IL DISEGNO DIGITALE DELLE CONICHE IN AUTOCAD

Indagine proiettiva e topologica dei luoghi geometrici sottesi.

Giovanni Anzani – 2026

Università degli Studi di Firenze - Dipartimento di Architettura DIDA

Via Della Mattonaia 8 - 50121 Firenze (FI)

Mob.: +39/339/7056663

giovanni.anzani@unifi.it

9 781291 657180

Premessa teorica

La Geometria Proiettiva e la Geometria Descrittiva si configurano storicamente come le discipline dello spazio astratto, fondate sull'indagine delle relazioni intercorrenti tra gli enti geometrici primari. In una prima istanza, lo studio si rivolge all'ente base del punto, inteso come entità atomica e adimensionale. Attraverso un primo atto di espansione spaziale, il passaggio dal singolo ente isolato al sistema continuo genera la retta, intesa come luogo geometrico lineare di primo grado, le cui interazioni spaziali sono rigorosamente ordinate dai criteri fondamentali di appartenenza, parallelismo e perpendicolarità. Da questo sostrato elementare, la trattazione si eleva alla determinazione delle curve di secondo grado attraverso un preciso passaggio dal singolo ente geometrico al sistema continuo: l'ellisse, la parabola e l'iperbole emergono come luoghi geometrici di secondo grado costituiti da punti generici di passaggio. Con l'introduzione di tali entità curvilinee, l'apparato teorico si integra mediante criteri funzionali più complessi, quali l'allineamento e la complanarietà, necessari a governare l'ampliamento delle tematiche e a verificarne le condizioni di esistenza strutturale. Nella pratica analogica, la determinazione grafica non ha mai costituito un puro esercizio di restituzione visiva, bensì la materializzazione di un processo speculativo: tracciare una conica richiedeva l'impiego di costrutti cinematici o di inviluppi proiettivi capaci di tradurre l'astrazione relazionale in una traiettoria verificabile sul piano.

L'indagine proiettiva assume la sua massima valenza euristica quando opera un ulteriore salto di scala, traslando dal singolo ente conica al sistema continuo superiore dei fasci di coniche. In questo dominio, l'attenzione si trasferisce dalla ricerca dei punti

generici di passaggio allo studio dei luoghi geometrici di terzo e quarto grado descritti dalle relazioni dei punti notevoli della curva, quali centri, fuochi e vertici. La costruzione si emancipa dal vincolo puramente metrico per affidarsi a princìpi di incidenza, tra i quali il Teorema di Pascal e il suo esagono mistico costituiscono un fondamento operativo centrale. Imponendo al piano una quaterna di punti base, si definisce un quadrangolo topologico la cui specifica morfologia – convessa, concava o trapezoidale – governa le proprietà dell'intero fascio sotteso. Il piano si configura così come un campo di relazioni continue, in cui l'infinita famiglia di curve condivide e specializza traiettorie spaziali rigorosamente interdipendenti in funzione della configurazione geometrica dei vertici originari.

L'introduzione dei sistemi CAD ha traslato questa sintassi all'interno di uno spazio vettoriale digitale. Tale transizione ha tuttavia comportato una precisa contrazione epistemologica, definibile in termini di automazione nativa. Nelle interfacce informatiche convenzionali, l'ente geometrico perde la propria flessibilità proiettiva per irrigidirsi in un costrutto statico. Per la generazione di un'ellisse, il software esige l'immissione preliminare dei suoi parametri metrici e posizionali, come il centro e la lunghezza dei semiassi, escludendo metodologie di tracciamento basate, ad esempio, sulle proprietà focali o su vincoli proiettivi parziali; al contempo, profili conici quali parabole e iperboli risultano del tutto assenti come primitive geometriche native all'interno dei medesimi ambienti. Questo paradigma impone un'inversione logica: la curva non viene dedotta dalle condizioni topologiche di contorno, ma deve resultar analiticamente nota prima della sua rappresentazione. L'automazione si limita pertanto in una mera esecuzione algebrica, che preclude lo spazio d'indagine.

Lo sviluppo di un'estesa libreria di funzioni in ambiente AutoLISP risponde all'esigenza metodologica di superare tale rigidità computazionale, restituendo all'ambiente digitale le prerogative di un'autentica Geometria Computazionale. Attraverso l'astrazione algoritmica, lo spazio di disegno viene riconfigurato come un sistema dinamico. Qualora il costrutto venga posto in una condizione di voluta sottodeterminazione – come nel caso dell'assegnazione di un quadrangolo topologico base – il software non si limita a calcolare un'unica soluzione predefinita, ma genera lo spettro delle curve appartenenti a quel fascio. Le variazioni morfologiche della conica non si manifestano in forma casuale, ma descrivono con continuità i luoghi geometrici dei propri elementi notevoli. La curva viene pertanto trattata non come un dato metrico isolato, ma come una componente variabile all'interno di un sistema relazionale continuo.

Tale ristrutturazione teorica si concretizza operativamente nel paradigma dell'Universal Loop, un modello fondato sull'interazione continua tra Luogo e Nodo. Attraverso la restituzione a schermo delle entità discrete e delle curve VLA che materializzano i luoghi geometrici, lo strumento informatico assume il ruolo di dispositivo per l'indagine proiettiva. In modalità "Luogo", l'operatore esplora con continuità le mappe di centri, fuochi o vertici, verificando in tempo reale l'adattamento morfologico della conica al variare dei parametri del fascio. Nella modalità "Nodo", l'interazione consente l'intercettazione esatta dei punti singolari e delle soluzioni analitiche precalcolate. L'esplicitazione grafica di questi loci restituisce rigore all'atto decisionale: la determinazione della conica, sia essa condotta per progressiva saturazione dei gradi di libertà o per vincoli di assialità, diviene l'esito di una scelta generativa fondata sul controllo razionale della struttura geometrica.

La libreria dei comandi AutoLISP: specifiche geometriche, vincoli operativi e casistica applicativa

I paragrafi che seguono descrivono in modo sistematico l'architettura applicativa del progetto informatico, traducendo i principi proiettivi della geometria computazionale in strumenti operativi integrati nell'ambiente AutoCAD. La trattazione è strutturata per accompagnare il lettore attraverso un percorso a complessità geometrica crescente, organizzando i comandi non in un semplice elenco di procedure, ma in un quadro organico di soluzioni algoritmiche applicate ai problemi di determinazione e tracciamento delle coniche.

L'apertura della sezione è dedicata alla definizione rigorosa delle convenzioni sintattiche e strutturali che governano l'intero impianto software. Viene formalizzato l'alfabeto geometrico del codice, esplicitando la nomenclatura dei punti di passaggio, dei centri, dei fuochi e dei vertici, distinguendo nettamente tra entità geometriche stabili (fisse) ed entità variabili (riposizionabili). Accanto alla notazione dei punti, viene illustrato il protocollo di gestione dell'ambiente di disegno: l'inizializzazione e il ripristino delle variabili di sistema tramite le routine Init_Conic_App e Restore_Conic_App, il controllo dinamico dei filtri di snap ad oggetto (OSMODE) per evitare interferenze durante le fasi di calcolo, e l'impostazione centralizzata degli undici layer standard deputati ad accogliere i vettori grafici, governata dal comando SLC. Vengono inoltre chiariti i meccanismi globali di interazione a schermo, quali le opzioni di accumulo o sostituzione delle curve (Modo) e i fattori di scala per l'estensione dei rami iperbolici oltre i vertici propri.

Il corpo centrale del manuale analizza singolarmente ogni applicativo sviluppato, secondo una griglia analitica costante che esamina i dati richiesti in input, la logica matematica del motore di calcolo, la struttura del loop interattivo e gli esiti grafici restituiti a schermo. Questa disamina parte dai comandi a determinazione bifocale e monofocale diretta (CON2V1P, CON2F1P, PARVF), in cui la natura e l'andamento della conica vengono derivati dalle coordinate locali di un punto in rapporto agli elementi canonici assegnati. La trattazione si estende poi ai sistemi complessi basati sulla sottodeterminazione dei vincoli e sulla conseguente emersione dei fasci di curve. Attraverso i comandi dedicati alle parabole per due e tre punti (PAR2PV, PAR2PFAPI, PAR3P) e alle ellissi e iperboli vincolate (ELL3PRD, IPE3PRD, CON3PC), viene documentata l'implementazione del paradigma "Universal Loop". Il testo descrive segnatamente come il codice generi le nuvole di punti discrete e le curve VLA che materializzano i loci geometrici di centri, fuochi e vertici, ed esplicita le modalità operative di interazione proiettiva (modalità Luogo per la navigazione continua e modalità Nodo o Target per l'intercettazione delle soluzioni analitiche esatte precalcolate).

La sezione culmina con l'analisi dei motori di calcolo universali per configurazioni a quattro e cinque punti (CON4P, CON5P, e le relative versioni reattive REA). Viene descritta dettagliatamente la logica di classificazione topologica automatica del quadrangolo base (quadrato, rettangolo, rombo, concavo ortocentrico, concavo generico, convesso, trapezio), da cui dipende l'attivazione di specifici algoritmi di scansione geometrica e la successiva estrazione di curve speciali come le iperboli equilatere o le ellissi pseudo-circolari. Infine, l'esame dei comandi a cinque punti illustra la saturazione definitiva dei gradi di libertà, mostrando come il software impieghi parabole guida e sezioni degeneri come confini geometrici del piano per guidare l'operatore nel tracciamento deterministico della curva finale. L'intero repertorio è supportato da un apparato iconografico che funge da verifica geometrica visiva, dimostrando l'esattezza delle traiettorie dei luoghi e la corretta separazione delle aree topologiche al variare delle condizioni geometriche imposte dal riposizionamento dei punti a schermo.

Notazioni simboliche di punti, rette e coniche

- P, P_1, P_2, P_3, P_4, P_5 punti generici di passaggio della conica o punti forniti per una qualche determinazione di condizioni vincolanti; se necessaria una distinzione si useranno le lettere es: P_{3a}, P_{3b}, P_{3c} o P_{4a}, P_{4b}, P_{4c}, o P_{5a}, P_{5b}, P_{5c}.
- C, punti identificati quali centri di una conica; se necessaria una distinzione si useranno le lettere o i numeri: C_A C_B C_C o C_1 C_2 C_3 a seconda delle necessità.
- V (parabola) o V_a e V_b (iperbole e ellisse) o V_a V_b V_c V_d (ellisse) punti identificati quali vertici di una conica; se necessaria una distinzione si useranno le lettere per distinguere i vertici e i numeri per distinguere le coniche: V_1 V_2 V_{1a} V_{1b} V_{2a} V_{2b}.
- F (parabola) o F_a e F_b (iperbole e ellisse) punti identificati quali fuochi di una conica; se necessaria una distinzione si useranno le lettere per distinguere i fuochi e i numeri per distinguere le coniche: F_1 F_2 o F_{1a} F_{1b} F_{2a} F_{2b}.
- ell, ell_1, ell_2, ell_3 le coniche se ellissi; par, par_1, par2, par_3 le coniche se parabole; ipe, ipe_1, ipe_2, ipe_3 le coniche se iperboli,
- (fi), (ri) i punti qualificati come fissi o come riposizionabili
- a d le rette quali asse e direttrice di una parabola, se necessaria una distinzione si useranno i numeri: a_1 a_2 d_1 d_2.
- a_{mag}, a_{min}, a_a, a_b le rette quali asse maggiore e minore di una ellissi, se necessaria una distinzione si useranno le lettere per distinguere gli assi e i numeri per distinguere le coniche: a_{1mag}, a_{1min}, a_{2mag}, a_{2min}, a_{1a},a_{1b}, a_{2a}, a_{2b}.
- a_{re} a_{im} le rette quali asse reale e immaginario di una iperbole, se necessaria una distinzione si useranno le lettere per distinguere gli assi e i numeri per distinguere le coniche: a_{1re} a_{1im} a_{2re} a_{2im}.
- a_{sa} a_{sb} le rette quali asintoti dell'iperbole, se necessaria una distinzione si useranno le lettere per distinguere gli asintoti e i numeri per distinguere le coniche: a_{1sa} a_{1sb} a_{2sa} a_{2sb}.

Funzionalità comuni ai vari comandi

Inizializzazione e Ripristino dell'Ambiente

Ogni comando inizia invocando **Init_Conic_App**, che: (1) salva le variabili di sistema OSMODE, PDMODE e CMDECHO; (2) imposta PDMODE a seconda dei casi a 32 o 0; (3) acquisisce i punti base tramite getpoint con il numero richiesto; (4) restituisce la lista dei punti e i valori originali delle variabili di sistema. Alla chiusura di ogni comando viene invocato **Restore_Conic_App** che ripristina le variabili ai valori originali.

Durante i loop interattivi, il comportamento dell'OSMODE segue un protocollo costante: il valore originale dell'utente viene riattivato prima della richiesta di un nuovo punto (per consentire gli snap), e viene azzerato subito dopo la ricezione dell'input (per evitare interferenze durante il calcolo e il disegno).

Opzione "Modo" (Sostituisci / Aggiungi)

Presente in quasi tutti i comandi. In **Sostituisci** (valore predefinito), ogni nuova conica cancella la precedente: il sistema usa un marcatore piazzato prima del disegno, raccoglie tutte le entità create dopo il marcatore e le elimina prima di disegnare la conica successiva. In **Aggiungi**, le curve si accumulano a schermo, consentendo di costruire visivamente famiglie o fasci di coniche.

Opzione "Estendi / Accorcia"

Presente nei comandi che generano iperboli. Controlla il fattore di scala **opt_e** (predefinito 1.0) con il quale vengono estesi i rami dell'iperbole oltre i vertici propri. Ogni "Estendi" incrementa di 0.5, ogni "Accorcia" decrementa di 0.5. Se una conica è già tracciata, il comando la ridisegna immediatamente con il nuovo fattore.

Opzione "Guide"

Attiva o disattiva entità ausiliarie di costruzione a schermo. Il contenuto varia: rette perpendicolari (CON2V1P), Dima topologica (PAR2PV, PAR2PFAPI Libero), parabole guida di separazione (CON5P), o luoghi geometrici rappresentati da nuvole di punti (PAR2PFAPI, PAR3P, ELL3PRD, CON4P).

Il Paradigma "Luogo / Nodo" (Universal Loop)

I comandi PAR3P, ELL3PRD, IPE3PRD e CON4P condividono un sistema di navigazione fondato su tre famiglie di luoghi geometrici: **Centri**, **Fuochi** e **Vertici**. Ciascuna può essere accesa o spenta indipendentemente. L'utente clicca un punto a schermo e il sistema cerca il punto più vicino tra tutti i luoghi attivi. In modalità **Luogo** (predefinita), il punto identifica una conica per interpolazione continua lungo la mappa dei luoghi. In modalità **Nodo** (attivabile con "Target"), il punto identifica il punto base più vicino e restituisce la soluzione analitica esatta precalcolata; click ripetuti sullo stesso nodo consentono di ciclare tra eventuali soluzioni multiple presenti.

Nomenclatura dei Layer

Le entità prodotte vengono distribuite su layer con nomi standardizzati definiti in variabili globali: *LaCo_Tracciati* (curva finale), *LaCo_Costruzioni* (assi, segmenti ausiliari), *LaCo_Punti_scelti* (marcatori), *LaCo_Poligoni* (triangoli, quadrangoli), *LaCo_Luoghi_geometrici* (nuvole di punti, loci). Ogni layer ha un nome ed un colore predefiniti eventualmente impostati all'avvio di ciascun comando se non già impostati preventivamente tramite l'uso del comando dedicato SLC (Settaggio Layer e Colori).

Quadro riepilogativo dei comandi realizzati

Al fine di restituire una lettura sistematica dell'intero impianto algoritmico, viene di seguito proposto (Tabella 1) il quadro sinottico e la sintesi operativa della libreria di comandi AutoLISP sviluppata per l'indagine proiettiva e topologica delle coniche in ambiente AutoCAD. Lo schema illustra in dettaglio l'architettura computazionale degli applicativi, esplicitando per ciascun comando le relazioni dirette tra i dati proiettivi richiesti per la risoluzione dell'algoritmo e le forme geometriche risultanti. Nello specifico, vengono rigorosamente riepilogati i gradi di libertà da saturare in input tramite l'immissione di punti fissi, punti riposizionabili o vincoli spaziali di assialità. Contestualmente, viene definita la natura dell'entità determinata, distinguendo nettamente tra il tracciamento deterministico di singole curve (ellissi, parabole, iperboli) e la generazione di puri luoghi geometrici. L'ultima colonna declina infine le interfacce di loop dinamico e le variabili di sistema: queste costituiscono lo strumento cardine per l'indagine, definendo le opzioni interattive a disposizione dell'operatore per la navigazione consapevole e l'esplorazione spaziale in tempo reale delle soluzioni ammissibili all'interno dei fasci di coniche.

Comando	Input Richiesto	Tipo di Conica	Opzioni di Menu
SLC	Nomi dei layers di sistema	Nessuna conica (solo scelta layers)	Si/No (riuso layer), 11 prompt individuali (se No)
CON2V1P	2 Vertici + N Punti	Ellisse/Iperbole	Guide, Estendi, Accorcia, Modo
CON2F1P	2 Fuochi + N Punti	Ellisse o Iperbole (scelta utente)	Ellisse/Iperbole, Quilatera, Estendi, Accorcia, Modo, Indietro
PARVF	1 Vertice + N Fuochi o viceversa	Parabola	Ordinamento (V-F / F-V), Modo
PAR2PV	2 Punti + N Vertici	Parabola (1/2/3 soluzioni)	Guide (Dima), Modo
PAR2PFAPI	2 Punti + Vincolo Asse + N Vertici o N Fuochi	Parabola	Fascio (Proprio/Improprio/Libero), Fuochi, Vertici, Modo; oppure Guide+Modo in Libero
PAR2PFAPIREA	2 Punti + Vincolo Asse (Reattivo)	Nessuna conica (solo luoghi geometrici)	Proprio/Improprio, Fuochi, Vertici, Spostamento dei 3 punti
PAR3P	3 Punti + N Vertici o N Fuochi	Parabola	Fuochi, Vertici, Target (Luogo/Nodo), Modo
PAR3PREA	3 Punti (Reattivo)	Nessuna conica (solo luoghi geometrici)	Fuochi, Vertici, Spostamento dei 3 punti, HD in uscita
ELL3PRD	3 Punti + Vincolo + N Centri o N Vertici o N Fuochi	Ellisse	Rapporto(b/a)/Direzione, Centri, Fuochi, Vertici, Target, Modo
IPE3PRD	3 Punti + Vincolo + N Centri o N Vertici o N Fuochi	Iperbole	Rapporto(b/a)/Direzione, Centri, Fuochi, Vertici, Target, Estendi, Accorcia, Modo
CON3PC	3 Punti + Centro (proprio o improprio)	Ellisse/Iperbole o Parabola (scelta utente)	Centro (Proprio/Improprio), Guide, Modo
CON4P	4 Punti + N Centri o N Vertici o N Fuochi	Ellisse/Parabola/Iperbole	Centri, Fuochi, Vertici, Target, Speciali (Ell/Ipe/Par), Estendi, Accorcia, Modo
CON5P	4 Punti + N 5° Punto	Ellisse/Parabola/Iperbole	Guide (parabole separatrici), Modo
CON5PREA	5 Punti (Reattivo)	Ellisse/Parabola/Iperbole	Guide, Spostamento di qualsiasi dei 5 punti

Tabella 1 *– Quadro sinottico degli applicativi AutoLISP: sintesi dei vincoli proiettivi in input, delle entità generate e delle opzioni interattive per l'esplorazione topologica dei luoghi geometrici.*

SLC – Configurazione dei Layer di Progetto

Configura gli 11 layer standard utilizzati dall'intera libreria di funzioni e comandi per organizzare le entità prodotte dai comandi di costruzione. Ogni layer è associato a una variabile globale (*LaCo_Punti_scelti*, *LaCo_Poligoni*, *LaCo_Centri*, *LaCo_Tracciati*, *LaCo_Luoghi_geometrici*, *LaCo_Costruzioni*, *LaCo_Assi*, *LaCo_Vertici*, *LaCo_Fuochi*, *LaCo_Asintoti*, *LaCo_Direttrice*) e riceve un nome predefinito (ad es. "Coniche - Punti scelti", "Coniche - Poligoni", ecc.).

Flusso interattivo

Il comando non è silenzioso: presenta un'interazione a due livelli.

Fase 1 — Verifica rapida (Load_Conic_Layers_Fast)

Il sistema verifica se tutti gli 11 layer predefiniti esistono già nel disegno (tramite tblsearch). Se esistono tutti, propone il loro riutilizzo:

Tutti i layer configurati sono gia' presenti. Usarli tutti? [Si/No] <Si>:

- **Si** (o Invio): accetta tutti i layer esistenti, li associa alle variabili globali e stampa "Layer inizializzati." — il setup è completato in un passaggio.
- **No**: prosegue alla Fase 2 per la configurazione individuale.

Fase 2 — Configurazione individuale (Config_Conic_Layers)

Se l'utente ha scelto "No" (o se nessun layer o non tutti i layer esistono nel disegno), il comando presenta 11 prompt consecutivi, uno per ciascun layer:

Layer Punti [Esistente] <Coniche - Punti scelti>:

Layer Poligoni [Esistente] <Coniche - Poligoni>:

Layer Centri [Esistente] <Coniche - Centri>:

Layer Tracciati (Curve) [Esistente] <Coniche - Tracciati>:

Layer Luoghi geometrici [Esistente] <Coniche - Luoghi geometrici>:

Layer Costruzioni [Esistente] <Coniche - Costruzioni>:

Layer Assi [Esistente] <Coniche - Assi>:

Layer Vertici [Esistente] <Coniche - Vertici>:

Layer Fuochi [Esistente] <Coniche - Fuochi>:

Layer Asintoti [Esistente] <Coniche - Asintoti>:

Layer Direttrice [Esistente] <Coniche - Direttrice>:

Per ciascun prompt l'utente può accettare il nome predefinito (premendo Invio) oppure digitare un nome alternativo. Il nome scelto viene salvato nel dizionario LDATA del disegno (chiave "GA_Disegno_Coniche") tramite la funzione Get_Layer_Smart, così che le sessioni successive possano ripristinarlo automaticamente.

Invocazione automatica

Il comando c:slc può anche essere invocato in modo trasparente: la funzione **Init_Conic_App**, chiamata all'avvio di ogni altro comando, verifica la presenza delle variabili globali *LaCo_* e, se mancanti, lancia automaticamente Config_Conic_Layers prima di proseguire.

CON2V1P – Conica bifocale dati due punti fissi quali vertici V_1 V_2 e un punto riposizionabile P_1

Assegnati due punti fissi quali vertici principali V_1 e V_2 di una conica bifocale (iperbole o ellisse) è possibile determinarne il centro proprio C nel punto medio dei due vertici, la direzione dell'asse principale nella retta congiungente i due vertici e il valore del semiasse a nella distanza tra i due vertici dimezzata. Per completare la determinazione della conica resta da trovare il semiasse minore b. Fornendo un terzo punto riposizionabile generico P_1, il sistema calcola le sue coordinate locali (u, v) rispetto al centro C: la coordinata 'u' è la proiezione di P lungo l'asse principale (direzione $V_1 – V_2$), mentre 'v' è la proiezione lungo l'asse a esso ortogonale.

In base alla coordinata 'u', si verificano tre casi mutuamente esclusivi che consentono di ricavare univocamente la curva e la sua natura sostituendo (u, v) nell'equazione canonica:

- Se la distanza $|u| < a$ (ovvero P è compreso nella striscia perpendicolare delimitata dai vertici), il punto appartiene a un'**ellisse** la cui equazione è $u^2/a^2 + v^2/b^2 = 1$. Da qui il sistema ricava univocamente $b = |v| / \sqrt{(1 - u^2/a^2)}$.
- Se la distanza $|u| > a$ (ovvero P è esterno a tale striscia), il punto appartiene a un'**iperbole** la cui equazione è $u^2/a^2 - v^2/b^2 = \pm 1$. Poiché i vertici giacciono sull'asse reale, l'equazione corretta è $u^2/a^2 - v^2/b^2 = 1$, da cui si ricava $b = |v| / \sqrt{(u^2/a^2 - 1)}$.
- Se $|u| = a$, il punto giace sulla perpendicolare esatta passante per un vertice e la conica degenera in due rette parallele, risultando incalcolabile: il comando la rifiuta segnalando "Punto sul confine".

Acquisizione iniziale

Richiede esattamente **2 punti** fissi quali vertici (V_1, V_2) della conica bifocale: se ellissi si intendono quali vertici dell'asse maggiore. Se coincidono, il comando segnala errore "I vertici non possono coincidere" e si arresta. Viene disegnato il segmento $V_1 – V_2$ sul layer Costruzioni.

Prompt del loop interattivo

GUIDE: off - EST: 1.0 - MODO: Sostituisci | Punto o [Guide/Estendi/Accorcia/Modo/Uscita] <Uscita>:

Le opzioni disponibili sono:

- **Guide** (ON/OFF): asse V1 e V2 + due rette perpendicolari all'asse passanti per i vertici.
- **Estendi / Accorcia**: controlla l'estensione dei rami iperbolici (solo se la conica è un'iperbole). Se la conica corrente è un'iperbole e il modo è Sostituisci, il ridisegno è immediato.
- **Modo**: commuta Sostituisci/Aggiungi.
- **Uscita** o Invio: termina il comando.
- **Punto cliccato**: il motore CON1P_Geom_Conic_2V_Engine calcola le coordinate locali (u, v) rispetto al centro C e determina automaticamente il tipo di conica.

Costruzioni a schermo

- Segmento $V_1 – V_2$ (asse principale) sul layer Costruzioni.
- Se Guide = ON: 2 rette perpendicolari passanti per i vertici.
- Ellisse: entità nativa AutoCAD con fuochi, vertici secondari, rapporto b/a (Draw_ConEll_CeAx_Foci).
- Iperbole: spline interpolata dei rami + asintoti (Draw_ConIpe_CeVeFu_Compl_Free).

Casi particolari

- Punto con $|u| = a$: semiasse b = 0, il comando segnala "Punto sul confine".
- Punto con $|u| < a$ ed $|u| > a$: determina automaticamente ellisse o iperbole.

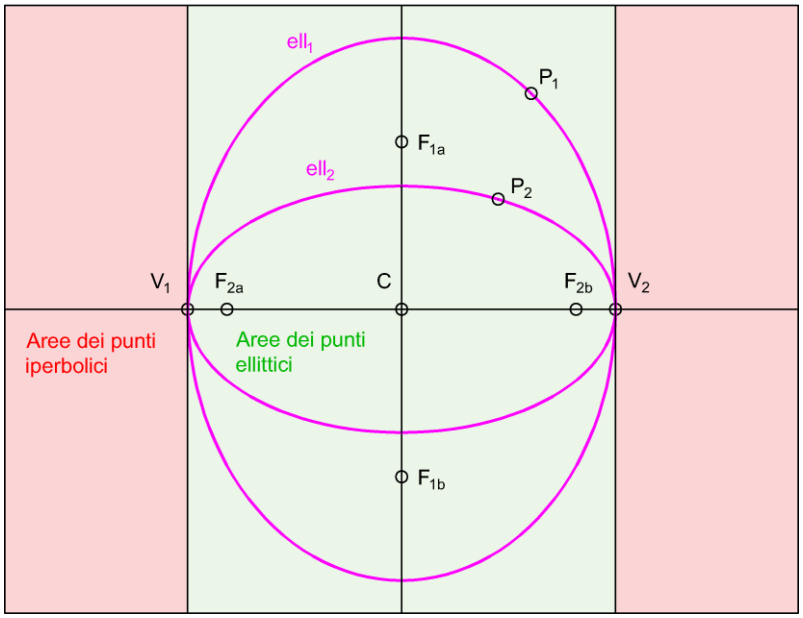

Figura 1 – CON2V1P (soluzione ellittica) – Dati i due vertici V_1 V_2 fissi e determinato il centro C fisso, sono dati i punti P_1 e P_2 riposizionabili tali da ricadere nell'area dei punti ellittici (verde) interna alle normali per i vertici V_1 e V_2 e tali da generare le ellissi ell_1 ell_2 passanti per i vertici assegnati. Sono altresì evidenziati i fuochi F_{1a} F_{1b} F_{2a} F_{2b} delle curve risultanti.

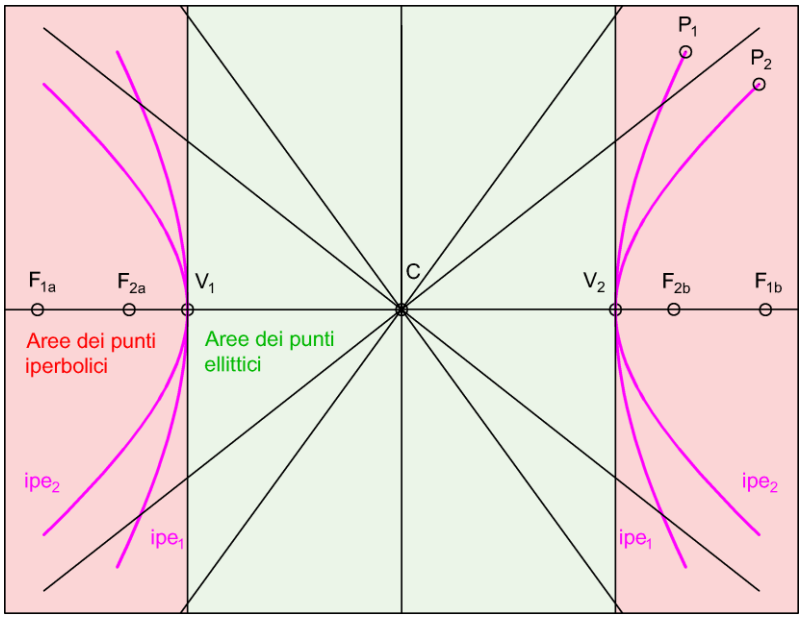

Figura 2 *– CON2V1P (soluzione iperbolica) – Dati i due vertici V_1 V_2 fissi e determinato il centro C fisso, sono dati i punti P_1 e P_2 riposizionabili tali da ricadere nell'area dei punti iperbolici (rossa) esterna alle normali per i vertici V_1 e V_2 e tali da generare delle iperboli ipe_1 ipe_2 passanti per i vertici assegnati. Sono altresì evidenziati i fuochi F_{1a} F_{1b} F_{2a} F_{2b} delle curve risultanti.*

CON2F1P – Conica bifocale dati due punti fissi quali fuochi F_1 F_2 e un punto riposizionabile P_1

Assegnati due punti quali fuochi F_1 e F_2 di una conica bifocale (iperbole o ellisse) è possibile determinarne il centro proprio C nel punto medio dei due fuochi, la direzione dell'asse focale nella retta congiungente i due fuochi e il valore della distanza semifocale c nella distanza tra i due fuochi dimezzata. Per qualsiasi punto P_1 del piano, la somma $P_1F_1 + P_1F_2$ definisce 2a per l'ellisse; la differenza $|P_1F_1 - P_1F_2|$ definisce 2a per l'iperbole. In entrambi i casi $b = \sqrt{|a^2-c^2|}$. Poiché un punto P_1 generico giace contemporaneamente su un'ellisse e su un'iperbole confocali, l'utente deve scegliere il tipo di conica desiderata prima di posizionare il punto.

Acquisizione iniziale

Acquisisce **2 punti** (F1, F2). Se coincidono: errore.

Menu principale (loop esterno)

MODO: Sostituisci | Scegli tipo [Ellisse/Iperbole/Modo/Uscita] <Uscita>:

- **Ellisse**: entra nel sotto-loop per ellissi confocali.
- **Iperbole**: entra nel sotto-loop per iperboli confocali.
- **Modo**: commuta Sostituisci/Aggiungi.
- **Uscita** o Invio: termina.

Sotto-loop Ellisse

MODO: Sostituisci | Punto Ellisse o [Modo/Indietro] <Indietro>:

- **Punto cliccato**: calcola $a = (d_1+d_2)/2$ e disegna l'ellisse con Draw_ConEll_CeAx_Foci.
- **Modo**: commuta.
- **Indietro** o Invio: torna al menu principale.

Sotto-loop Iperbole

MODO: Sostituisci - EST: 1.0 | Punto Iperbole o [Quilatera/Estendi/Accorcia/Modo/Indietro] <Indietro>:

- **Punto cliccato**: calcola $a = |d_1-d_2|/2$ e disegna l'iperbole con Draw_ConIpe_CeVeFu_Compl_Free.
- **Quilatera**: genera la posizione automatica del punto tale da produrre a = b (il punto viene posizionato a distanza $c/\sqrt{2}$ dal centro lungo l'asse principale). Disegna l'iperbole equilatera risultante.
- **Estendi / Accorcia**: modifica l'estensione dei rami.
- Modo / Indietro: come sopra.

Casi particolari

- Ellisse: se $a \leq c$ (punto troppo vicino al segmento focale): errore "ellisse impossibile".
- Iperbole: se $a < 1e\text{-}6$ oppure $a \geq c$: errore "fuori dominio iperbolico".

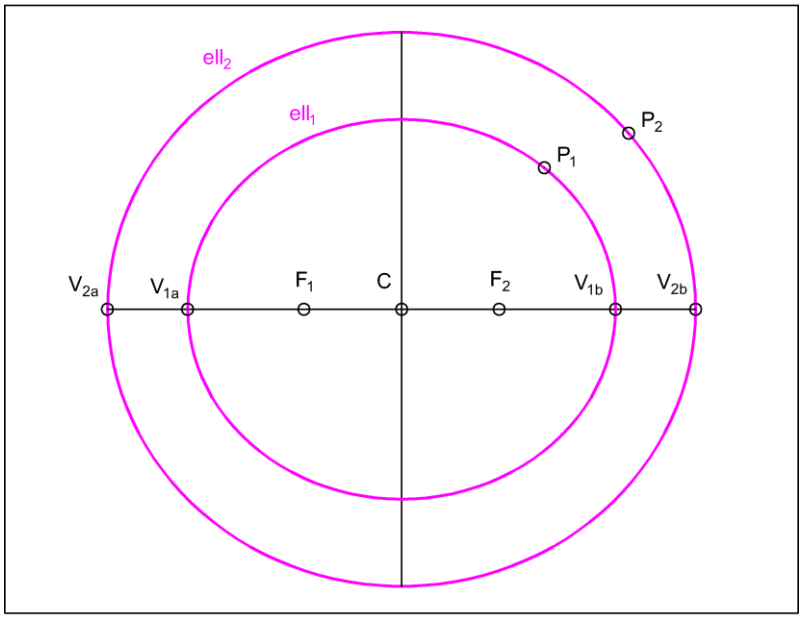

Figura 3 *– CON2F1P (costruzione ellittica) – Dati i due fuochi F_1 F_2 fissi, determinato il centro C fisso e scelta come tipo di conica bifocale desiderata l'ellisse, sono dati i punti P_1 e P_2 riposizionabili di passaggio per le ellissi ell_1 ell_2 associate ai due fuochi fissi assegnati. Sono altresì evidenziati i vertici V_{1a} V_{1b} V_{2a} V_{2b} delle curve risultanti.*

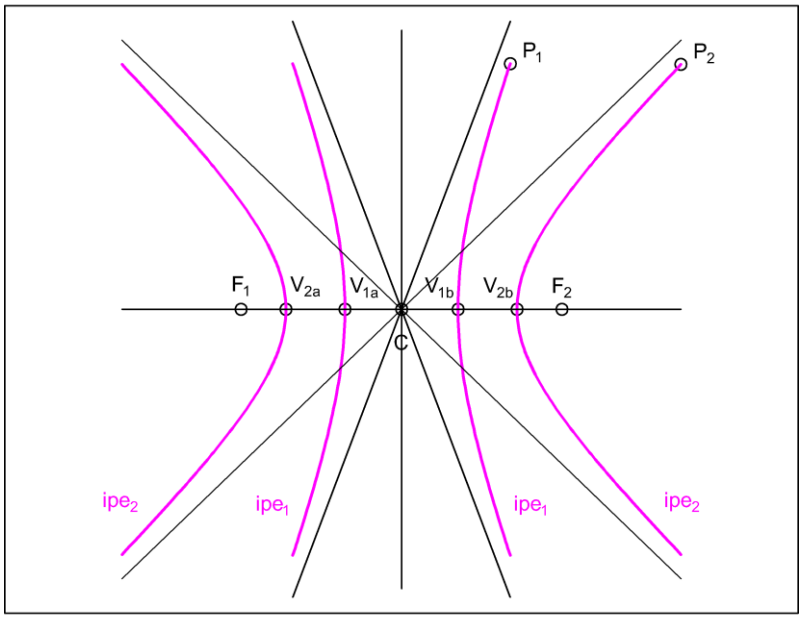

Figura 4 *– CON2F1P (costruzione iperbolica) – Dati i due fuochi F_1 F_2 fissi, determinato il centro C fisso e scelta come tipo di conica bifocale desiderata l'iperbole, sono dati i punti P_1 e P_2 riposizionabili di passaggio per le iperboli ipe_1 ipe_2 associate ai due fuochi fissi assegnati. Sono altresì evidenziati i vertici V_{1a} V_{1b} V_{2a} V_{2b} delle curve risultanti.*

PARVF – Parabola dati due punti quali vertice V e fuoco F alternativamente fissi o riposizionabili

Assegnati due punti quali fuoco F e vertice V di una conica monofocale (parabola), è possibile determinarne: l'asse di simmetria, coincidente con la retta passante per i punti F e V; la distanza focale (e conseguentemente l'apertura della curva), pari alla distanza intercorrente tra F e V; il verso della concavità, indicato dalla direzione del vettore VF; e infine la direttrice, quale retta ortogonale all'asse di simmetria e passante per il simmetrico di F rispetto a V.

Acquisizione iniziale

Il comando acquisisce **1 solo punto base** (detto "Ancora"), che per impostazione predefinita viene interpretato come il Vertice.

Prompt del loop interattivo

ORD: V-F - MODO: Sostituisci | Specifica Fuoco (F) o [Ordinamento/Modo/Fine] <Fine>:

La descrizione del punto richiesto cambia dinamicamente in funzione dell'ordinamento:

- Se ORD = **V-F**: l'Ancora è il **Vertice**, il click specifica il **Fuoco**. Il prompt dice "Specifica Fuoco (F)".
- Se ORD = **F-V**: l'Ancora è il **Fuoco**, il click specifica il **Vertice**. Il prompt dice "Specifica Vertice (V)". Nota: quando si cambia ordinamento, se esiste già una coppia di punti precedente, i ruoli di p_base e p_other vengono fisicamente scambiati, invertendo l'effetto di elastico dal nuovo punto di vista.
- **Ordinamento**: inverte il ruolo dell'Ancora tra Vertice e Fuoco.
- **Modo**: commuta Sostituisci/Aggiungi.
- **Fine** o Invio: termina.
- **Punto cliccato**: disegna la parabola con Draw_ConPar_VeFu_Compl. Il draw include un punto da 15× la distanza focale lungo l'asse per definire l'estensione del tracciamento.

PAR2PV – Parabola dati due punti fissi P_1 P_2 e un punto riposizionabile quale vertice V

Assegnati due punti fissi P_1 P_2 di passaggio per una parabola, la scelta di un ulteriore punto riposizionabile quale posizione del Vertice V di tale parabola consente di saturare i gradi di libertà residui con un numero variabile tra 0 e 3 possibili soluzioni in relazione alla ricerca degli zeri dell'equazione trigonometrica che definisce l'orientamento dell'asse θ per assorbire il passaggio sui due punti. Il sistema scandisce le direzioni con passo di 0.5° (motore Geom_Eval_Par_Error) e raffina gli zeri tramite bisezione. A causa del grado dell'equazione in θ, la posizione scelta per il Vertice riposizionabile determina il numero di direzioni reali per l'asse: a seconda della posizione scelta in rapporto alla Dima di suddivisione delle aree, il comando disegna **0, 1, 2 o 3 possibili parabole** distinte.

Acquisizione iniziale

Acquisisce **2 punti** (P1, P2).

Prompt del loop interattivo

GUIDE: off - MODO: Sostituisci | Specifica il Vertice (V) o [Guide/Modo/Uscita] <Uscita>:

- **Guide** (on/off): attiva/disattiva la **Dima**, ossia il confine topologico della regione entro la quale il Vertice produce almeno una parabola reale. Calcolata da Helper_Update_Dima_Par2pv.
- **Modo**: Sostituisci/Aggiungi.
- **Uscita** o Invio: termina.
- **Punto V cliccato**: scandisce 360° e disegna tutte le parabole trovate. Il prompt risponde: "Disegnate N parabole uniche valide."

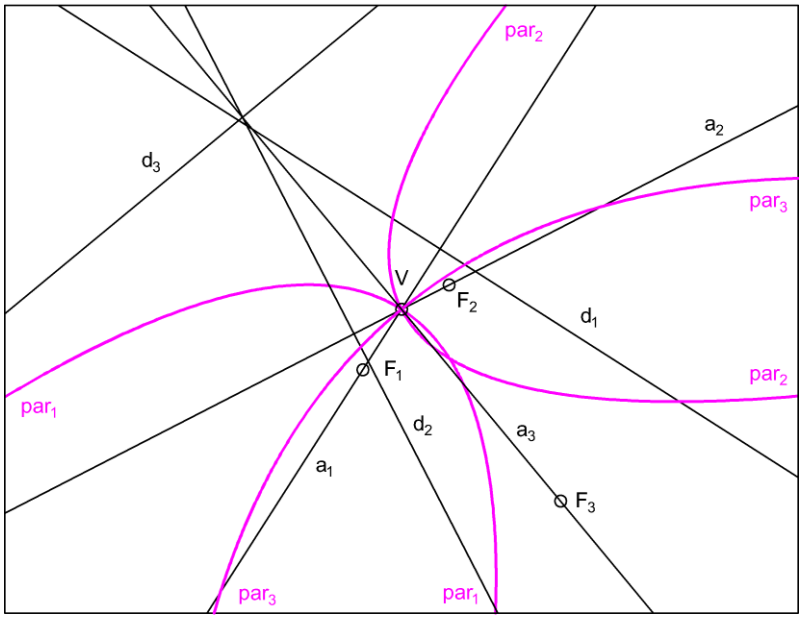

Figura 5 *– PARVF – Determinazione di una serie di parabole par₁ par₂ par₃ aventi vertice V fisso e fuoco F₁ F₂ F₃ variabile. Sono altresì evidenziati l'asse a1 a2 a3 e la direttrice d1 d2 d3 delle curve risultanti.*

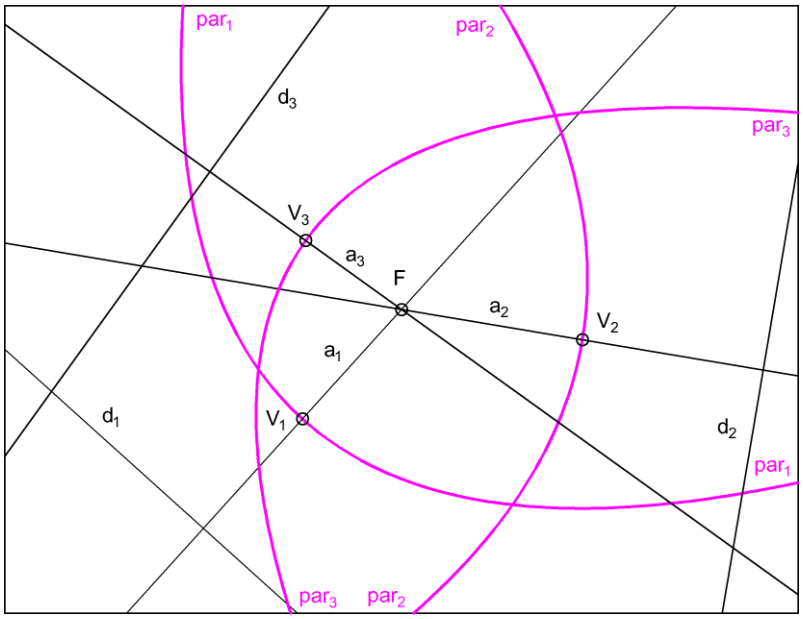

Figura 6 *– PARVF – Determinazione di una serie di parabole par₁ par₂ par₃ aventi Fuoco F fisso e vertice V₁ V₂ V₃ variabile. Sono altresì evidenziati l'asse a1 a2 a3 e la direttrice d1 d2 d3 delle curve risultanti.*

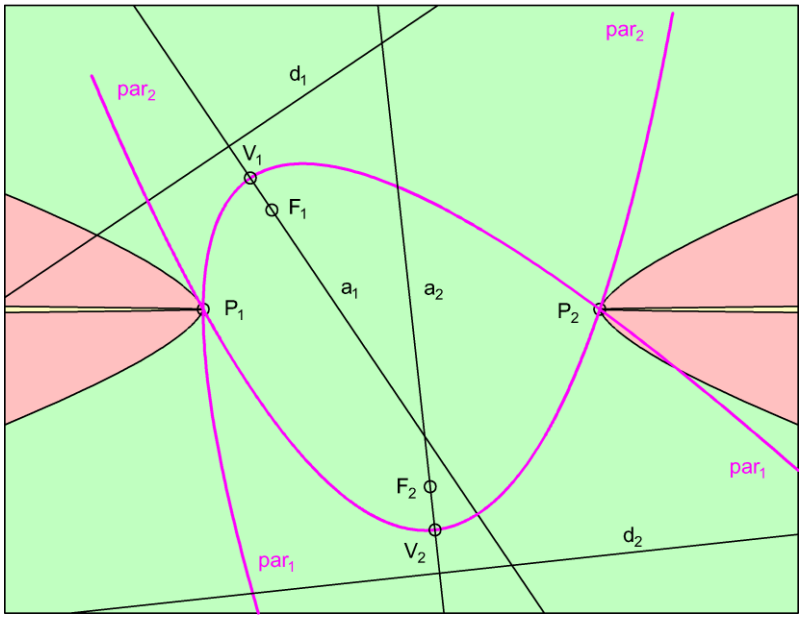

Figura 7 *– PAR2PV – Determinazione di una serie di parabole par₁ par₂ per i due punti fissi P₁ P₂ e per dei vertici variabili V₁ V₂ posti in un'area (verde) in cui si presenta un'unica soluzione. Per ogni parabola, oltre al vertice V₁ V₂, sono mostrati: il fuoco F₁ F₂, l'asse a₁ a₂ e la direttrice d₁ d₂. In rosso le aree che presentano tre soluzioni ed in giallo le aree che presentano due soluzioni.*

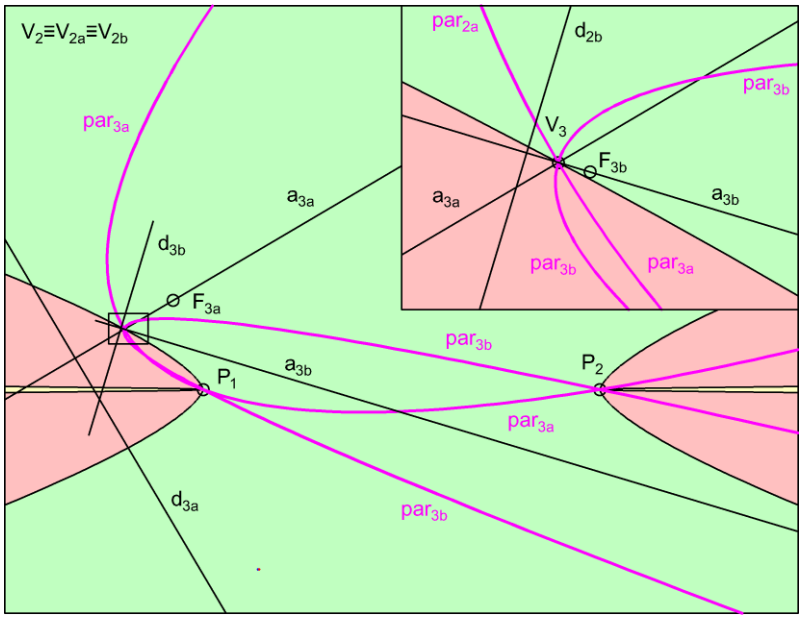

Figura 8 *– PAR2PV – Determinazione di una coppia di parabole par3a par3b per i due punti fissi P₁ P₂ e per il vertice variabile V₃ posto in un'area in cui si presenta una coppia di soluzioni (gialla). Per ogni parabola, oltre al vertice V₃ in comune, sono mostrati: il fuoco F₃a F₃b, l'asse a₃a a₃b e la direttrice d₃a d₃b. In rosso le aree che presentano tre soluzioni ed in verde le aree che presentano una soluzione.*

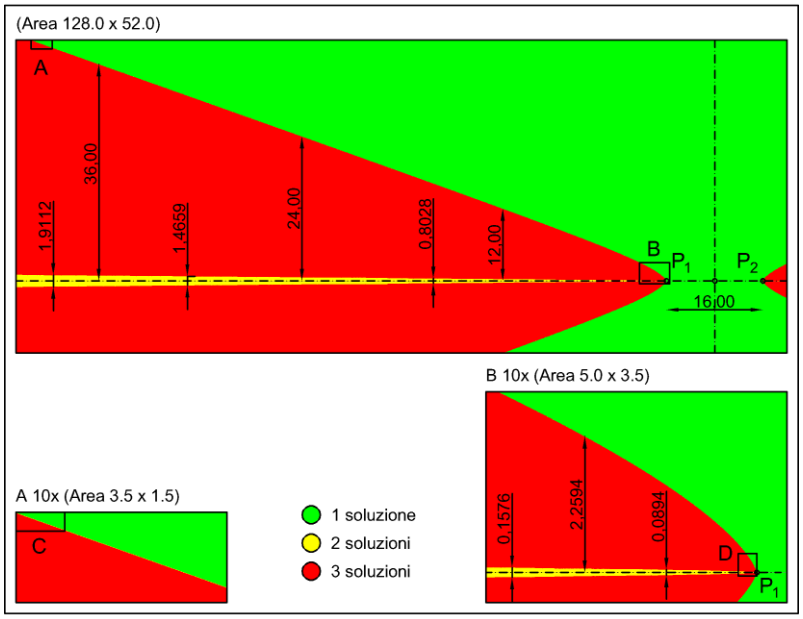

Figura 9 *– PAR2PV – Data una coppia di punti P_1 P_2 a distanza 16 unità grafiche, viene mostrata sopra una mappatura estesa ad un rettangolo di 128 x 52 unità grafiche delle aree in funzione del numero di soluzioni ammissibili (1, 2, 3), sotto le aree di dettaglio A e B con ingrandimento 10x, al loro interno l'indicazione di due ulteriori aree di dettaglio C e D con ingrandimento 100x poste a seguire in Figura 10.*

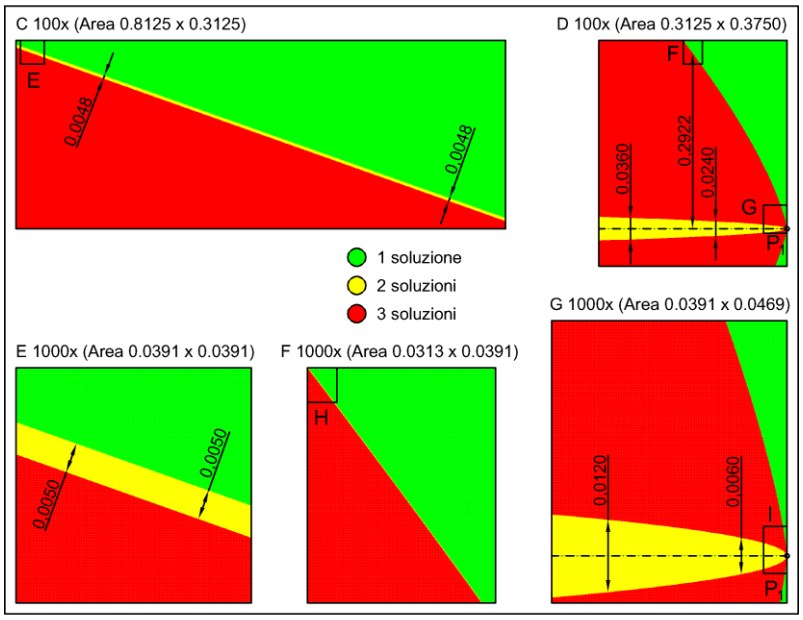

Figura 10 *– PAR2PV – In continuità con Figura 9 Sopra le aree di dettaglio C e D con ingrandimento 100x, al loro interno l'indicazione di tre ulteriori aree di dettaglio E F e G con ingrandimento 1000x riportate sotto, al loro interno l'indicazione di due ulteriori aree di dettaglio H e I con ingrandimento 10.000x poste a seguire in Figura 11.*

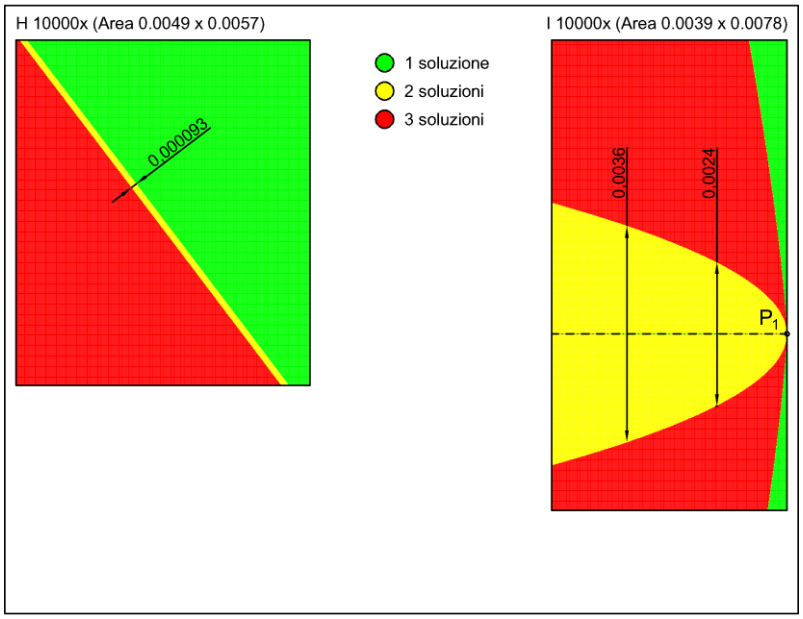

Figura 11 *– PAR2PV – In continuità con Figura 10 le aree di dettaglio H e I con ingrandimento 10.000x.*

La serie A C E e la serie B D F H mostrano, in due zone diverse, un'area a 2 soluzioni (gialla) posta tra le aree a 1 e 3 soluzioni (rispettivamente verde e rossa), la serie B D G I mostra l'area in prossimità del punto P1 e un'area a 2 soluzioni (gialla) posta internamente all'area a 3 soluzioni (rossa).

PAR2PFAPI – Parabola dati P1, P2 (fi) e F1 F2 F3 o V1 V2 V3 (ri) con a vincolato

L'acronimo FAPI, presente quale suffisso nel nome del comando, sta per "Fascio Asse Proprio Improprio". Assegnati due punti fissi di passaggio P_1 e P_2, il comando vincola l'asse di simmetria della parabola all'appartenenza ad un fascio di rette, distinguendo due modalità operative: nel caso di fascio proprio, l'asse viene costretto a passare per un punto P_3; nel caso di fascio improprio, l'asse viene orientato parallelamente alla direzione definita dalla retta passante per P_3 e per il punto medio P_4 del segmento P_1 – P_2. Questo vincolo consente al motore matematico di calcolare e definire a schermo i luoghi geometrici continui dei vertici e dei fuochi di tutte le infinite parabole compatibili con l'insieme dei dati. La selezione interattiva di un punto specifico su uno di questi luoghi, quale vertice o fuoco, colma l'ultimo grado di libertà, consentendo al sistema di addivenire all'identificazione dell'univoca parabola capace di onorare tutte le premesse associate. Resta altresì disponibile il percorso "Libero", che replica integralmente l'esplorazione a 360° del funzionamento di par2PV.

Acquisizione iniziale

2 punti base (P_1, P_2). Viene disegnato il segmento P_1 – P_2.

Scelta del fascio

Fascio: [Proprio/Improprio/Libero] <Libero>:

Fascio Proprio – Richiede un 3° punto P_3 detto **Perno di rotazione**: l'asse della parabola passa per P_3 e ruota attorno ad esso. Il motore PAR2PFAPI_Engine_Proprio calcola i luoghi.

Fascio Improprio – Richiede un punto P_3 per la definizione di una direzione d a partire dal **punto medio** di P_1 – P_2: l'asse della parabola è fissato parallelo a questa direzione e trasla rispetto ad essa. Il motore PAR2PFAPI_Engine_Improprio calcola i luoghi.

Fascio Libero – Non richiede un 3° punto di vincolo per l'asse. Lancia **PAR2PFAPI_Libero_Runner**, un loop alternativo identico nel funzionamento al comando par2PV: l'utente clicca un vertice, e il sistema scandisce e ne ricava, supportato dalla Dima, le 1, 2 o 3 parabole passanti ammissibili.

Il prompt del Fascio Libero è:

GUIDE: off - MODO: Sostituisci | Specifica il Vertice (V) o [Guide/Modo/Uscita] <Uscita>:

- **Guide**: attiva la Dima topologica (identica a c:par2PV).
- **Modo / Uscita**: come di consueto.

Loop interattivo (Fascio Proprio e Improprio)

Dopo il calcolo dei luoghi geometrici questi vengono visualizzati per punti a schermo: vertici in magenta e fuochi in verde. Il prompt diviene:

F:OFF V:ON | M:Sostituisci | Punto o [Fuochi/Vertici/Modo/Uscita] <Uscita>:

- **Fuochi** (ON/OFF): accende/spegne la nuvola dei Fuochi.
- **Vertici** (ON/OFF): accende/spegne la nuvola dei Vertici.
- **Modo**: Sostituisci/Aggiungi.
- **Punto cliccato**: il sistema cerca il punto più vicino nella(e) nuvola(e) attiva(e), lo raffina con PAR2PFAPI_Refine_Locus e disegna la parabola associata. Se entrambe le nuvole sono accese, sceglie il punto globalmente più vicino al click. Se nessuna nuvola è accesa, segnala "Attiva un luogo."
- **Uscita**: consolida e pulisce gli oggetti VLA interni.

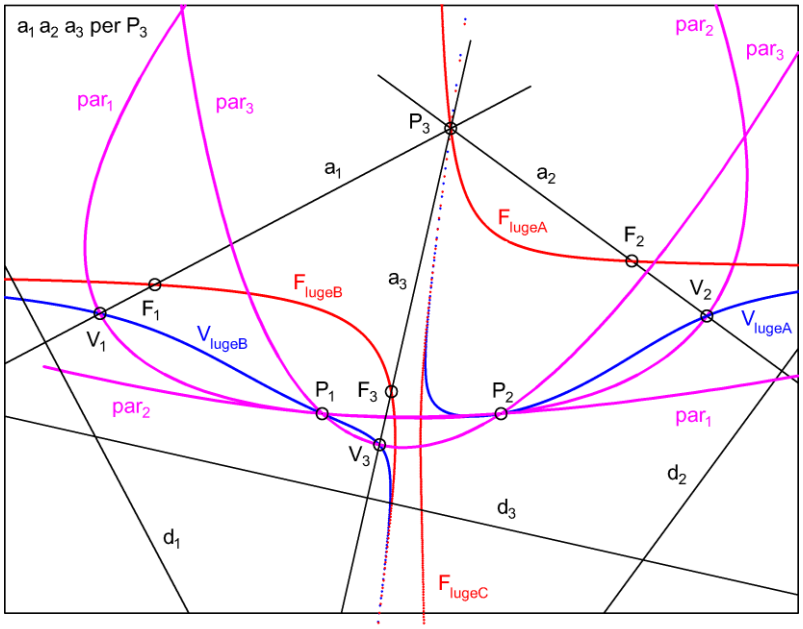

Figura 12 *– PAR2PFAPI – Una serie di parabole par_1 par_2 par_3 passanti dai punti fissi P_1 P_2 e per i vertici riposizionabili V_1 V_2 V_3 avendo imposto agli assi a_1 a_2 a_3 il passaggio per il punto fisso P_3. Per ogni parabola, oltre al vertice V_1 V_2 V_3, sono mostrati: il fuoco F_1 F_2 F_3, l'asse a_1 a_2 a_3 e la direttrice d_1 d_2 d_3. Stante il passaggio obbligato dell'asse in P_3 risultano determinati i luoghi geometrici F_{luge} V_{luge} in cui è possibile determinare la posizione del vertice e del fuoco delle parabole ammissibili.*

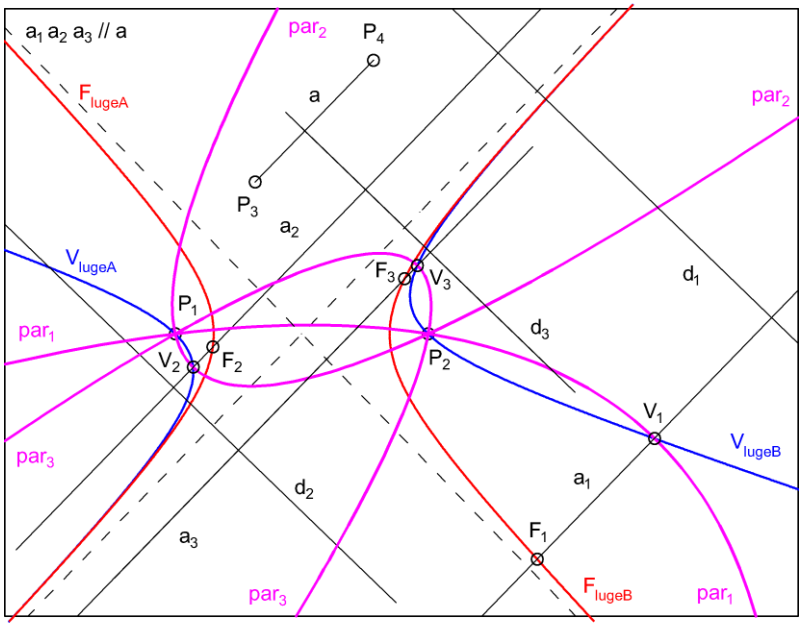

Figura 13 *– PAR2PFAPI – Una serie di parabole par_1 par2 par_3 passanti dai punti fissi P_1 P_2 e per i vertici riposizionabili V_1 V_2 V_3 avendo imposto agli assi a_1 a_2 a_3 la direzione a definita dai punti fissi P_3 P_4. Per ogni parabola, oltre al vertice V_1 V_2 V_3, sono mostrati: il fuoco F_1 F_2 F_3, l'asse a_1 a_2 a_3 e la direttrice d_1 d_2 d_3. Stante la direzione obbligata a dell'asse definita da P_3 P_4, risultano determinati i luoghi geometrici F_{luge} V_{luge} in cui è possibile determinare la posizione del vertice e del fuoco delle parabole ammissibili.*

PAR2PFAPIREA – Generazione dei luoghi geometrici riposizionabili per PAR2PFAPI

Il comando è identico alla parte iniziale del comando PAR2PFAPI, ma l'interazione è di tipo reattivo: anziché precalcolare i luoghi una volta, il sistema ricalcola e ridisegna tutto ad ogni spostamento di uno dei tre punti base senza poi proseguire nella determinazione di una specifica parabola.

Acquisizione iniziale

Richiede P_1 e P_2 (con getpoint in sequenza), poi:

Tipo fascio [Proprio/Improprio] <Proprio>:

Nota: la modalità **Libero** non è disponibile nella versione reattiva.

Infine acquisisce un 3° punto: "Perno rotazione P3" (se Proprio) oppure "Direzione asse P3" (se Improprio). Viene calcolato il vettore relativo v_rel tra P3 e il punto medio di P1-P2.

Loop reattivo

[REA-Proprio] F:ON V:ON | Seleziona P1/P2/P3 o [Fuochi/Vertici/Uscita] <Uscita>:

Ad ogni iterazione il sistema:

- 1. Cancella le entità del ciclo precedente (skeleton + nuvole di punti).
- 2. Ridisegna il triangolo $P_1 – P_2 – P_3$ con segmenti e marcatori tramite PAR2PFAPI_Draw_Skeleton_Rea.
- 3. Ricalcola i luoghi (Engine_Proprio o Engine_Improprio).
- 4. Visualizza le nuvole di punti Vertici e Fuochi e le raggruppa.

Le opzioni disponibili:

- **Fuochi** (ON/OFF): accende/spegne la nuvola dei Fuochi.
- **Vertici** (ON/OFF): accende/spegne la nuvola dei Vertici.
- **Punto cliccato**: il sistema identifica quale tra P1, P2 e P3 è il più vicino al click (variabile near_tag impostata a 'P1, 'P2 o 'P3). Poi chiede la nuova posizione con "Specifica nuova posizione". Se l'utente conferma, il punto viene spostato, la posizione relativa di P3 viene ricalcolata (nel caso Improprio la direzione dell'asse si aggiorna) e tutto il sistema viene ridisegnato.
- **Uscita**: effettua un ricalcolo finale ad alta risoluzione ("consolidamento finale") e termina.

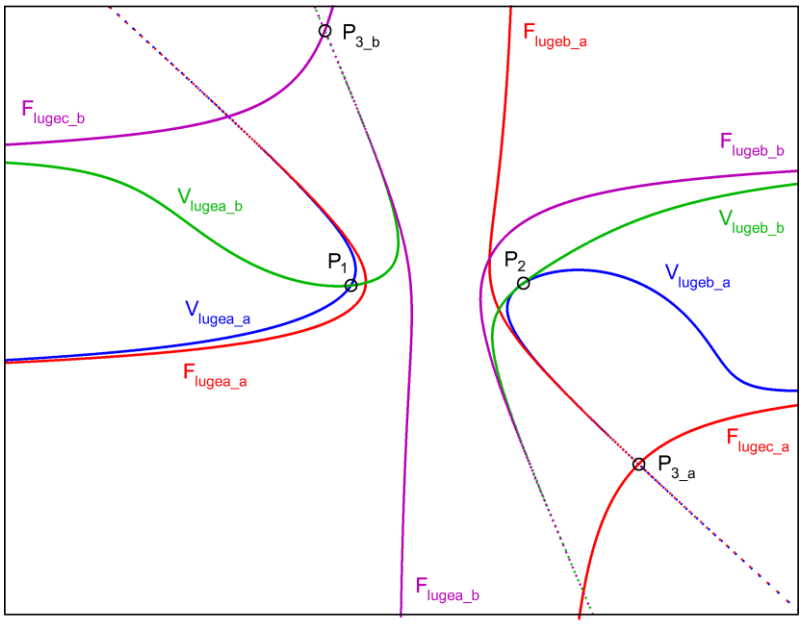

Figura 14 *– PAR2PFAPIREA (analisi interattiva) – Dati i punti fissi P_1 P_2 quali punti di passaggio per delle parabole ed il punto riposizionabile P_3 (P_{3_a} P_{3_b}) quale punto di passaggio per gli assi di tali parabole; risultano determinati i luoghi geometrici riposizionabili F_{luge} (F_{luge_a} F_{luge_b}) e V_{luge} (V_{luge_a} V_{luge_b}) in cui sarebbe possibile determinare la posizione del vertice e del fuoco delle parabole ammissibili.*

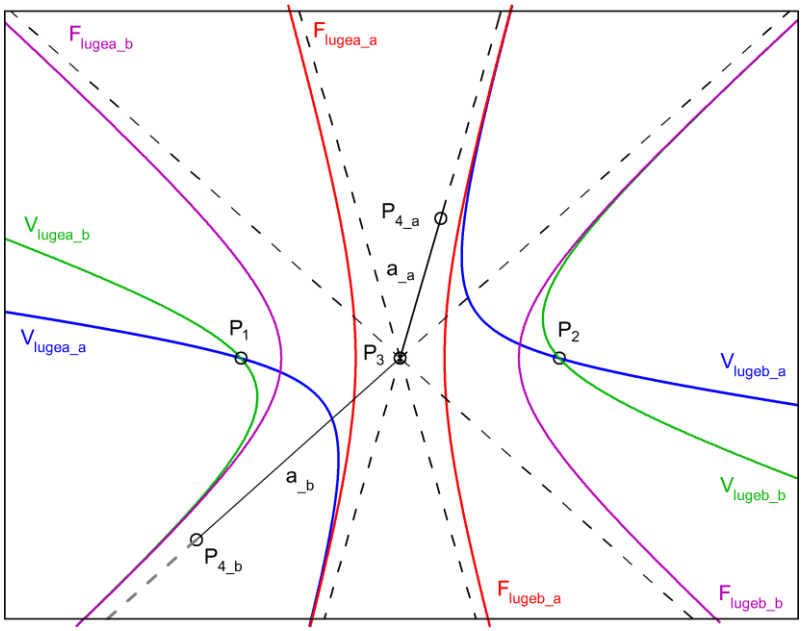

Figura 15 *– PAR2PFAPIREA – (analisi interattiva) – Dati i punti fissi P_1 P_2 quali punti di passaggio per delle parabole e la direzione a (a_a a_b) definita dal punto fisso P_3 e dal punto riposizionabile P_4 (P4_a P4_b) quale direzione per gli assi di tali parabole; risultano determinati i luoghi geometrici riposizionabili F_{luge} (F_{luge_a} F_{luge_b}) e V_{luge} (V_{luge_a} V_{luge_b}) in cui sarebbe possibile determinare la posizione del vertice e del fuoco delle parabole ammissibili; il luogo geometrico dei fuochi F_{luge} è in particolare definito da una iperbole avente i punti P_1 e P_2 quali fuochi.*

PAR3P – Parabola dati tre punti fissi P_1 P_2 P_3 e un punto riposizionabile quale vertice V o fuoco F

Tre punti P_1 P_2 P_3 non allineati lasciano un solo grado di libertà per la determinazione di un'univoca parabola: l'angolo θ dell'asse. Per ogni θ ammissibile esiste una parabola passante per i tre punti dotata di vertice e fuoco univoci. L'insieme dei vertici e fuochi di tali parabole genera un "luogo dei vertici" e un "luogo dei fuochi" mediante scansione adattiva.

Acquisizione iniziale

Acquisisce 3 punti P_1 P_2 P_3. Disegna il triangolo base. Calcola una distanza limite per delimitare i luoghi. Precalcola le soluzioni esatte per ciascun nodo.

Prompt del loop interattivo

F:OFF V:ON T:Luogo | M:Sostituisci | Punto o [Fuochi/Vertici/Target/Modo/Uscita] <Uscita>:

- **Fuochi** (ON/OFF): accende/spegne i luoghi dei Fuochi.
- **Vertici** (ON/OFF): accende/spegne i luoghi dei Vertici.
- **Target** (Luogo / Nodo): commuta tra snap continuo lungo la curva dei luoghi e snap esatto sui nodi precalcolati. In modalità Nodo, click ripetuti sul medesimo punto base ciclano tra le soluzioni multiple.
- **Modo**: Sostituisci/Aggiungi.
- **Uscita**: termina.
- **Punto cliccato**: in Luogo, cerca il punto più vicino nella nuvola attiva; in Nodo, cerca il nodo base più vicino tra i 3 punti e disegna le soluzioni esatte.

Nota sull'OSMODE nel loop

Il loop imposta OSMODE a 512 (snap "Vicino") prima del getpoint: questo consente all'utente di agganciarsi direttamente ai punti delle nuvole di luoghi a schermo.

PAR3PREA – Generazione dei luoghi geometrici (ri) per PAR3P

Il comando è identico alla parte iniziale del comando PAR3P, ma l'interazione è di tipo reattivo: anziché precalcolare i luoghi una volta, il sistema ricalcola e ridisegna tutto ad ogni spostamento di uno dei tre punti base senza poi proseguire nella determinazione di una specifica parabola. Definito in sole 3 righe di codice, lancia PAR3P_Master_Reactive_Universal_3P. Acquisisce 3 punti che consente all'utente di spostare interattivamente ricalcolando e ridisegnando l'intero sistema di luoghi ad ogni modifica.

Prompt del loop reattivo

F:ON V:ON | Muovi punto o [Fuochi/Vertici/Uscita] <Uscita per HD>:

- **Fuochi / Vertici** (ON/OFF): accende/spegne i relativi luoghi.
- **Punto cliccato**: OSMODE è forzato a 8 (snap "Nodo"), quindi il click aggancia direttamente il punto base AutoCAD più vicino. Il sistema chiede "Posizione:" e sposta il punto alla nuova posizione, poi ricalcola i luoghi con PAR3P_Master_Par_3P.
- **Uscita** o Invio: effettua un ricalcolo finale HD (passo 1.0) e termina.

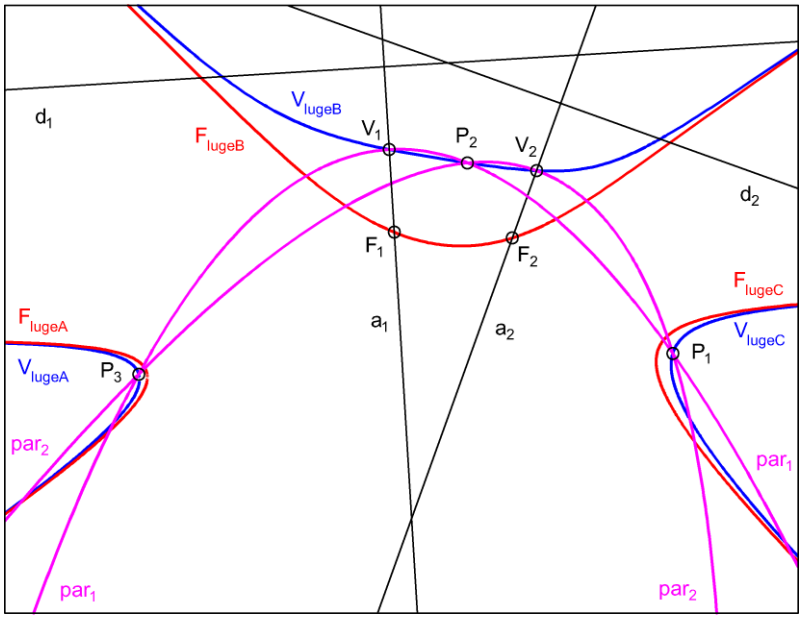

Figura 16 *– PAR3P – Una serie di parabole par_1 par_2 passanti dai punti fissi P_1 P_2 P_3 e per i vertici riposizionabili V_1 V_2. Per ogni parabola, oltre al vertice V_1 V_2, sono mostrati: il fuoco F_1 F_2, l'asse a_1 a_2 e la direttrice d_1 d_2. Stante il passaggio obbligato in P_1 P_2 P_3 risultano determinati i luoghi geometrici F_{luge} V_{luge} in cui è possibile determinare la posizione del vertice e del fuoco delle parabole ammissibili.*

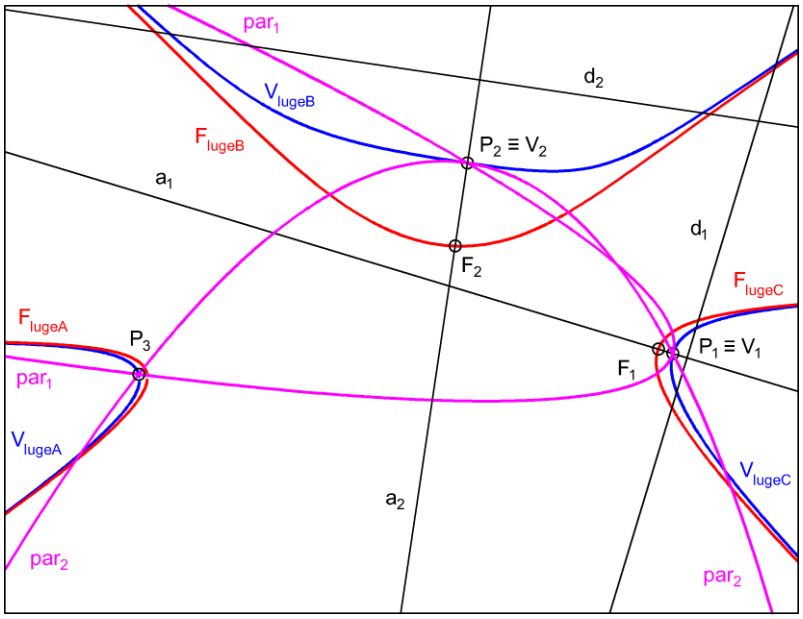

Figura 17 *– PAR3P – Una serie di parabole par_1 par_2 passanti dai punti fissi P_1 P_2 P_3 e per i vertici riposizionabili V_1 V_2 che, a differenza del disegno in Figura 16 risultano coincidere esattamente con P_1 e P_2. Per ogni parabola, oltre al vertice V_1 V_2, sono mostrati: il fuoco F_1 F_2, l'asse a_1 a_2 e la direttrice d_1 d_2. Stante il passaggio obbligato in P_1 P_2 P_3 risultano determinati i luoghi geometrici F_{luge} V_{luge} in cui è possibile determinare la posizione del vertice e del fuoco delle parabole ammissibili.*

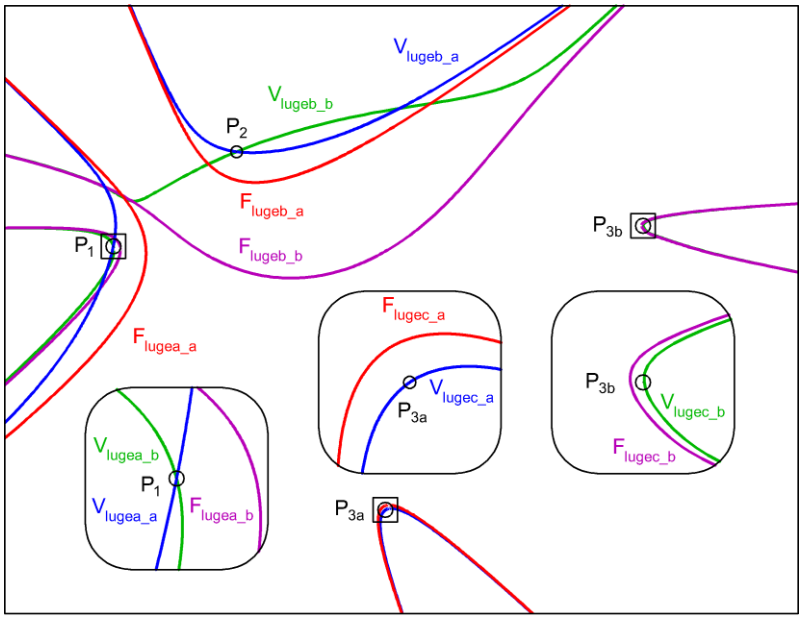

Figura 18 *– PAR3PREA (analisi interattiva) – Dati i punti fissi P_1 P_2 ed il punto riposizionabile P_3 (P_{3a} P_{3b}) quali punti di passaggio per delle parabole; risultano determinati i luoghi geometrici riposizionabili F_{luge} (F_{luge_a} F_{luge_b}) e V_{luge} (V_{luge_a} V_{luge_b}) in cui sarebbe possibile determinare la posizione del vertice e del fuoco delle parabole ammissibili.*

ELL3PRD – Ellisse dati tre punti fissi P1 P2 P3 e un punto riposizionabile quale centro C, vertice V o fuoco F e avendo vincolato il rapporto o la direzione degli assi

Tre punti fissi P_1 P_2 P_3 di passaggio forniscono tre dei cinque vincoli necessari a definire univocamente un'ellisse. Il primo dei due vincoli mancanti viene colmato con un parametro scelto dall'utente (Rapporto o Direzione); con un solo grado di libertà residuo non vincolato, il motore scansiona lo spazio delle soluzioni ammissibili e le presenta come luoghi geometrici navigabili per centro vertici e fuochi dell'ellisse. Con un ulteriore punto riposizionabile quale centro, vertice o fuoco dell'ellisse appartenente a tali luoghi geometrici, l'ellisse risulta determinata univocamente.

Acquisizione iniziale

Acquisisce **3 punti** (con OSMODE libero, PDMODE 0). Disegna il triangolo base e il **triangolo ortico** (altezze) come costruzione ausiliaria.

Scelta del vincolo

Vincolo: [Rapporto/Direzione] <Rapporto>:

- **Rapporto**: chiede il rapporto b/a (default 0.5). Se > 1.0, viene invertito automaticamente.
- **Direzione**: chiede 2 punti per definire la direzione dell'asse maggiore dell'ellisse.

Calcolo

Il motore ELL3P_Master_Engine calcola i luoghi geometrici dei **Centri**, **Fuochi** e **Vertici**, ciascuno sdoppiato in ramo "base" e "inverso". Vengono poi precalcolate le soluzioni analitiche esatte per ciascun nodo di passaggio: dove il luogo di un dato tipo attraversa esattamente uno dei tre punti base.

Prompt del loop interattivo

Se i Vertici sono accesi:

C:ON F:OFF V:ON T:Luogo | M:Sostituisci | Punto o [Centri/Fuochi/Vertici/Target/Modo/Uscita] <Uscita>:

Se i Vertici sono spenti, l'opzione Target viene rimossa dal prompt (la modalità Nodo richiede che i Vertici o i Fuochi siano accesi).

- **Centri** (ON/OFF): accende/spegne i punti dei Centri (colore ACI 1 — rosso).
- **Fuochi** (ON/OFF): accende/spegne i punti dei Fuochi (colore ACI 3 — verde).
- **Vertici** (ON/OFF): accende/spegne i punti dei Vertici (colore ACI 5 — blu). Quando viene spento, l'opzione Target torna automaticamente a "Luogo".
- **Target** (Luogo / Nodo): commuta tra snap continuo e soluzioni nodali esatte. In Nodo, per ogni punto base esistono potenzialmente più ellissi: click ripetuti ciclano.
- **Modo**: Sostituisci/Aggiungi.
- **Uscita**: termina.

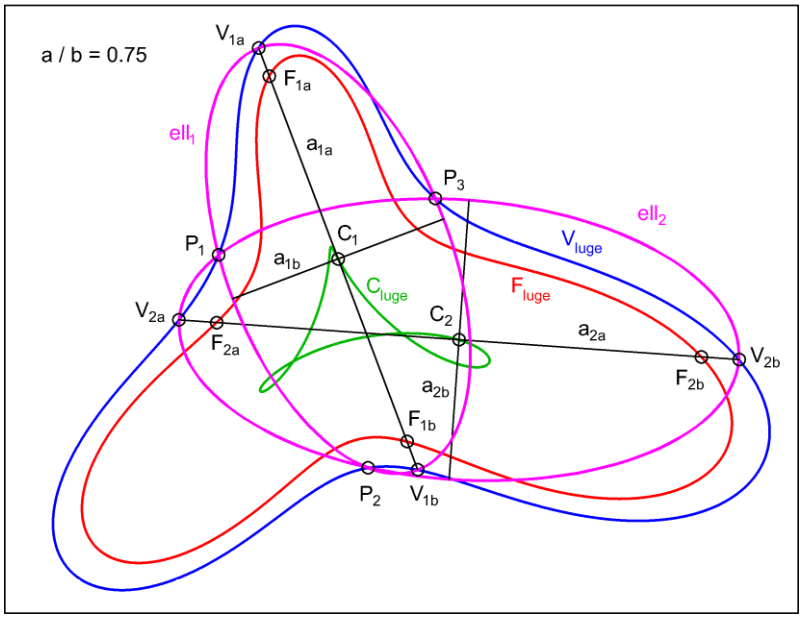

Figura 19 *– ELL3PRD – Una serie di ellissi ell_1 ell_2 passanti dai punti fissi P_1 P_2 P_3 e per i vertici riposizionabili V_a (V_{1a} V_{2a}) V_b (V_{1b} V_{2b}) e tali da avere un rapporto a/b assegnato tra semiassi, nell'esempio riportato pari a 0.75. Per ogni parabola, oltre ai vertici V_a V_b, sono mostrati: i fuochi F_a F_b, gli assi a_a a_b. Stante il passaggio obbligato in P_1 P_2 P_3 risultano determinati i luoghi geometrici C_{luge} F_{luge} V_{luge} in cui è possibile determinare la posizione del centro di un vertice e di un fuoco delle ellissi ammissibili.*

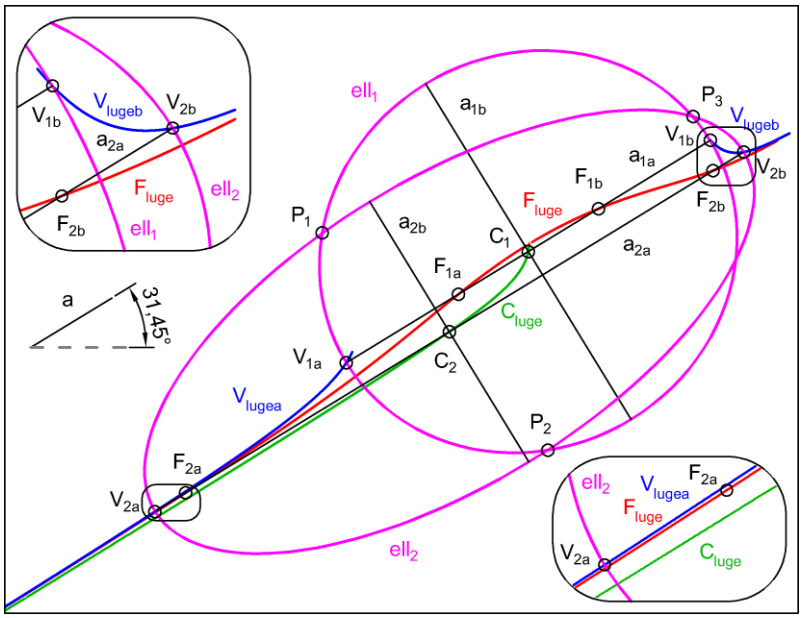

Figura 20 *– ELL3PRD – Una serie di ellissi ell_1 ell_2 passanti dai punti fissi P_1 P_2 P_3 e per i vertici riposizionabili V_a (V_{1a} V_{2a}) V_b (V_{1b} V_{2b}) e tali da avere una direzione a assegnata per i semiassi, nell'esempio riportato pari a 31,45°. Per ogni ellisse, oltre ai vertici V_a V_b, sono mostrati: i fuochi F_a F_b, gli assi a_a a_b. Stante il passaggio obbligato in P_1 P_2 P_3 risultano determinati i luoghi geometrici C_{luge} F_{luge} V_{luge} in cui è possibile determinare la posizione del centro di un vertice e di un fuoco delle ellissi ammissibili.*

IPE3PRD – Iperbole dati tre punti fissi P1 P2 P3 e un punto riposizionabile quale centro C, vertice V o fuoco F e avendo vincolato il rapporto o la direzione degli assi

Funzionamento analogo a ELL3PRD ma per iperboli. Tre punti fissi P_1 P_2 P_3 di passaggio forniscono tre dei cinque vincoli necessari a definire univocamente un'iperbole. Il primo dei due vincoli mancanti viene colmato con un parametro scelto dall'utente (Rapporto o Direzione); con un solo grado di libertà residuo non vincolato, il motore scansiona lo spazio delle soluzioni ammissibili e le presenta come luoghi geometrici navigabili per centro vertici e fuochi dell'iperbole. Con un ulteriore punto riposizionabile quale centro, vertice o fuoco dell'iperbole appartenente a tali luoghi geometrici, l'iperbole risulta determinata univocamente.

Scelta del vincolo

Vincolo: [Rapporto/Direzione] <Rapporto>:

- **Rapporto**: default 1.0 (diversamente da ell3PRD dove è 0.5). Non viene invertito se > 1.
- **Direzione**: chiede 2 punti per definire la direzione dell'asse reale (trasverso).

Prompt del loop interattivo

C:ON F:OFF V:OFF T:Luogo | M:Sostituisci | E:1.0 | Punto o [Centri/Fuochi/Vertici/Target/Modo/Estendi/Accorcia /Uscita] <Uscita>:

Rispetto al loop di c:ell3PRD, questo loop aggiunge:

- **Estendi / Accorcia**: controlla opt_e (estensione rami iperbolici). Se una conica è già tracciata, viene ridisegnata immediatamente.
- **Target** (Nodo): disponibile solo quando i Vertici sono accesi. Il ciclatore sui nodi funziona identicamente a c:ell3PRD.

Costruzioni a schermo

- Triangolo base e triangolo ortico.
- Fino a 6 rami di punti (2 per Centri, 2 per Fuochi, 2 per Vertici) indipendentemente accendibili.
- Iperbole risultante con rami, asintoti e fuochi tramite IPE3P_Draw_Engine.

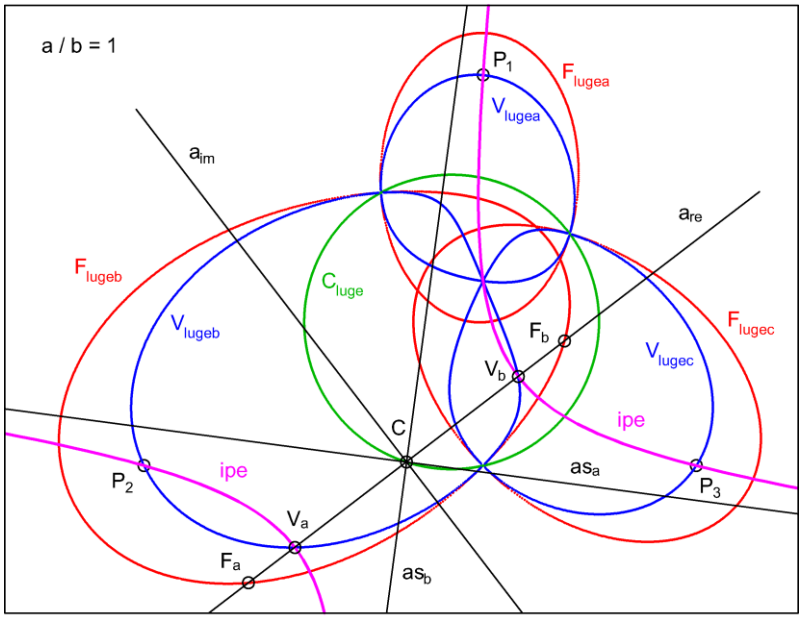

Figura 21 *– IPE3PRD – Un'iperbole ipe passante dai punti fissi P_1 P_2 P_3 e per i vertici riposizionabili V_a V_b e tale da avere un rapporto a/b assegnato tra semiassi, nell'esempio riportato pari a 1 (iperbole equilatera). Per l'iperbole, oltre ai vertici V_a V_b, sono mostrati: i fuochi F_a F_b, gli assi a_{re} a_{im}. Stante il passaggio obbligato in P_1 P_2 P_3 risultano determinati i luoghi geometrici C_{luge} (in questo caso descritto da una circonferenza) F_{luge} V_{luge} in cui è possibile determinare la posizione del centro di un vertice e di un fuoco delle iperboli (equilatere) ammissibili.*

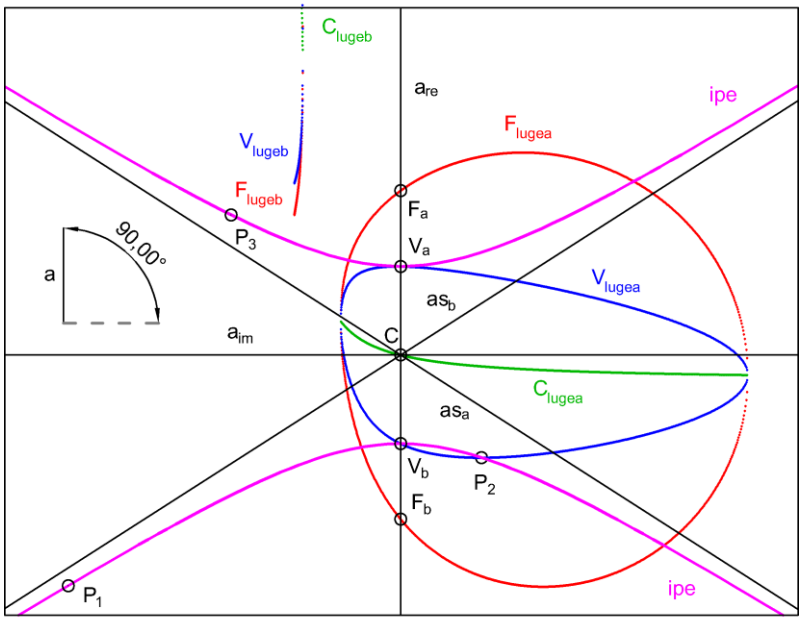

Figura 22 *– IPE3PRD – Un'iperbole ipe passante dai punti fissi P_1 P_2 P_3 e per i vertici riposizionabili V_a V_b e tali da avere una direzione a assegnata per i semiassi, nell'esempio riportato pari a 90,00°. Per l'iperbole, oltre ai vertici V_a V_b, sono mostrati: i fuochi F_a F_b, gli assi a_{re} a_{im}. Stante il passaggio obbligato in P_1 P_2 P_3 risultano determinati i luoghi geometrici C_{luge} F_{luge} V_{luge} in cui è possibile determinare la posizione del centro di un vertice e di un fuoco delle iperboli ammissibili.*

CON3PC – Conica dati tre punti fissi P_1 P_2 P_3 e un punto riposizionabile (proprio o improprio) quale centro C

Tre punti fissi P_1 P_2 P_3 di passaggio forniscono tre dei cinque vincoli necessari a definire univocamente una conica. Fissare il Centro C (proprio o improprio) di una conica impone la simmetria (polare o assiale) rispetto al dato assegnato: nel caso di centro proprio, per ogni punto P_1 P_2 P_3 è possibile determinare geometricamente il simmetrico P_4 P_5 P_6 sulla curva per soddisfare i cinque vincoli necessari; analiticamente, la centralità si traduce nel traslare il sistema in C assumendo la forma $aX^2 + bXY + cY^2 = 1$, dove le tre incognite lineari (a, b, c) si ricavano direttamente dal passaggio per i soli punti P_1 P_2 P_3 tramite un sistema di Cramer 3×3; nel caso in cui il centro C sia improprio, la conica degenera in una parabola: proiettivamente, assegnare il centro all'infinito equivale a imporre la tangenza alla retta impropria (condizione che vale due vincoli) definendo la direzione dell'asse di simmetria. I cinque gradi di libertà sono così univocamente saturati e l'equazione, avendo i termini di secondo grado vincolati in un quadrato perfetto noto dalla direzione assegnata, assume la forma $(\alpha X + \beta Y)^2 + dX + eY + f = 0$ dove i coefficienti noti (α, β) derivano dalla pendenza dell'asse; le tre incognite lineari residue (d, e, f) si ricavano agevolmente imponendo il passaggio per P_1, P_2, P_3, risolvendo anche in questo caso un diretto sistema lineare 3×3 di Cramer. Nel caso di centro C proprio il sistema è in grado di mappare il piano distinguendo le quattro aree dei centri ellittici dalle tre aree dei centri iperbolici.

Acquisizione iniziale

Acquisisce 3 punti fissi P_1 P_2 P_3 e disegna il triangolo base.

Interazione

Uno specifico motore gestisce un loop interattivo con la variabile **opt_c** che può assumere due valori: **Proprio** (un click indica il Centro della conica) o **Improprio** (una coppia di click indicano la direzione di un asse di simmetria). Il prompt varia dinamicamente:

Modalità Proprio (predefinita)

C:Proprio | G:ON | M:Sostituisci | Centro o [Centro/Guide/Modo/Fine] <Fine>:

- **Punto cliccato**: il sistema calcola la conica con quel centro (Geom_3Pts_Core_Proprio).

Modalità Improprio

C:Improprio | M:Sostituisci | 1° Punto Direzione asse o [Centro/Modo/Fine] <Fine>:

In questa modalità il sistema chiede 2 punti per definire la direzione dell'asse di simmetria e poi calcola la conica corrispondente (Geom_3Pts_Core_Improprio).

Le opzioni comuni:

- **Centro**: commuta tra Proprio e Improprio; quando cambia, anche le guide si aggiornano con Geom_3Pts_Update_Guides.
- **Guide** (ON/OFF): guide geometriche di supporto (solo in modo Proprio).
- **Modo**: Sostituisci/Aggiungi.
- **Fine** o Invio: termina.

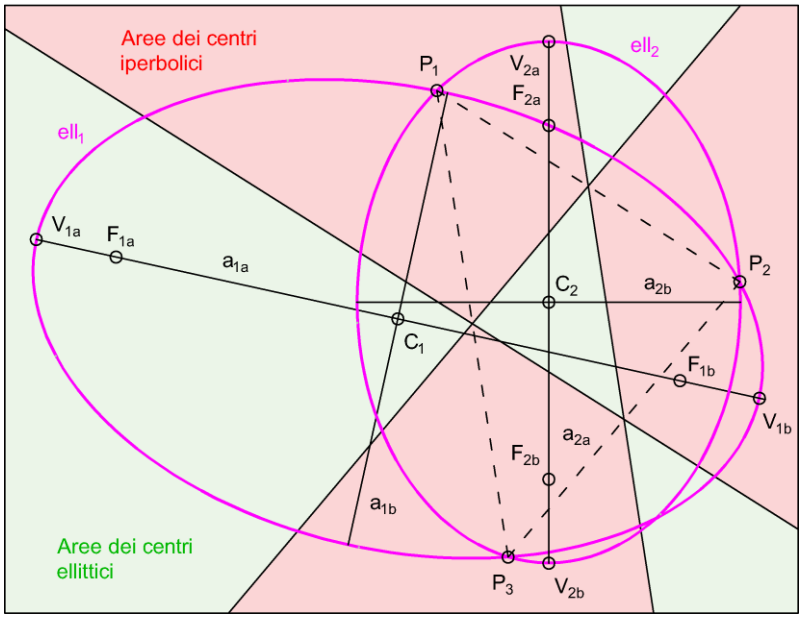

Figura 23 *– CON3PC – una serie di ellissi ell_1 ell_2 passanti dai punti fissi P_1 P_2 P_3 e per i centri propri riposizionabili C_1 C_2 e tali da ricadere nelle aree verdi di possibile collocazione dei centri ellittici ed avendo scelto di determinare una conica a centro proprio. Per ogni ellisse, oltre ai vertici V_a V_b, sono mostrati: i fuochi F_a F_b, gli assi a_a a_b.*

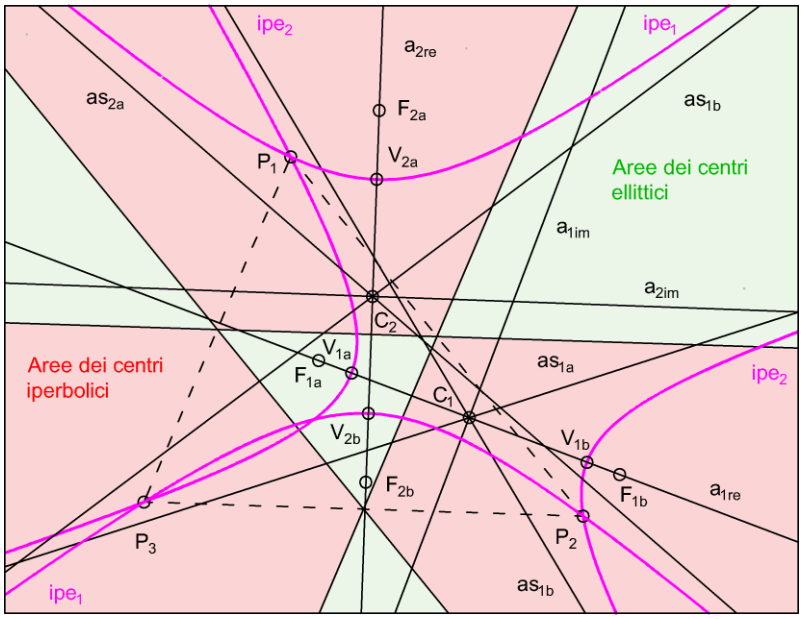

Figura 24 *– CON3PC – una serie di iperboli ipe_1 ipe_2 passanti dai punti fissi P_1 P_2 P_3 e per i centri propri riposizionabili C_1 C_2 e tali da ricadere nelle aree rosse di possibile collocazione dei centri iperbolici ed avendo scelto di determinare una conica a centro proprio. Per ogni iperbole, oltre ai vertici V_a V_b, sono mostrati: i fuochi F_a F_b, gli assi a_{re} a_{im}.*

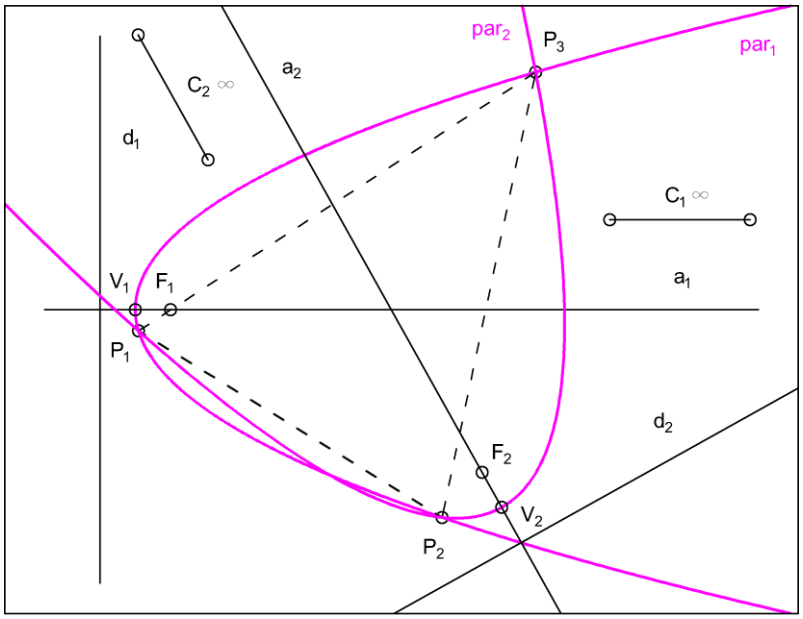

Figura 25 *– CON3PC – una serie di parabole par_1 par_2 passanti dai punti fissi P_1 P_2 P_3 e per i centri impropri riposizionabili C_1 C_2 ed avendo scelto di determinare una conica a centro improprio. Per ogni parabola, oltre al vertice V, sono mostrati: il fuoco F, l asse a e la direttrice d.*

CON4P – Conica dati quattro punti fissi P_1 P_2 P_3 P_4 e un punto riposizionabile quale centro C, fuoco F o vertice V

Quattro punti fissi P_1 P_2 P_3 P_4 di passaggio forniscono quattro dei cinque vincoli necessari a definire univocamente una conica ed in grado di definire un fascio di coniche (famiglia a un parametro λ). Al variare di λ si ottengono ellissi, iperboli e, per valori speciali, parabole di separazione. Il comando esplora questo fascio generando luoghi geometrici (Centri, Fuochi, Vertici) e consentendo all'utente di navigarli. Con un ulteriore punto riposizionabile quale centro, vertice o fuoco della conica desiderata scelto appartenente a tali luoghi geometrici, la conica risulta determinata univocamente.

Acquisizione iniziale e classificazione topologica

Acquisisce 4 punti tramite CON4P_Geom_4P_Master_Setup. I punti vengono classificati automaticamente (Math_Classify_Forma_4Pu) in una delle seguenti topologie, da ciascuna delle quali discende un diverso motore di calcolo:

- **QUADRATO, RETTANGOLO**: Motore ortogonale (CON4P_Draw_Orthogonal_Locus).
- **ROMBO, PARALLELOGRAMMA**: Motore parallelogramma (CON4P_Geom_Scan_Parallelogram_Locus).
- **CONCAVO_ORTOCENTRICO**: Motore ortocentrico (CON4P_Geom_Scan_Orthocentric_Circle).
- **CONCAVO_GENERICO**: Motore unificato concavo (CON4P_Geom_Scan_VLA_Centers).
- QUADRANGOLO_CICLICO, QUADRANGOLO_GENERICO: Motore unificato convesso.
- **TRAPEZIO_ISOSCELE, TRAPEZIO_SCALENO**: Motore unificato trapezoidale (con formattazione speciale di CON4P_Geom_Format_Trapezoid_Locus).
- **DEGENERE***: il comando si arresta segnalando l'impossibilità di determinare una conica se non degenere.

Fase 1: Pre-calcolo automatico

Prima dell'interazione, per tutti i motori diversi dall'Ortogonale, il sistema calcola:

- La curva VLA del luogo geometrico dei Centri (CON4P_Geom_Draw_Locus_And_Capture) disegnandola a schermo.
- I luoghi geometrici dei Fuochi (colori distinti per: ellisse, iperbole, parabola) e dei Vertici (colori distinti per: ellisse maggiori e minori, iperbole, parabola) disegnando le relative nuvole di punti di discretizzazione.
- Le soluzioni nodali esatte per ogni punto base (Geom_Precalc_Node_Conics).
- L'iperbole equilatera unica del fascio, se esiste (CON4P_Geom_Solve_Unique_Equilateral_Ipe).
- L'ellisse pseudo-circolare (eccentricità minima) del fascio, se esiste.
- Le parabole del fascio, se esistono (0 1 2 parabole a seconda delle configurazioni).

Il tempo di calcolo viene cronometrato e comunicato a schermo.

Le nuvole di punti vengono raggruppate in gruppi anonimi AutoCAD denominati "LUSNAP_Fuochi" e "LUSNAP_Vertici" per facilitare lo snap.

Ramo Ortogonale (Rettangoli/Quadrati)

[RETTANGOLO] FUOCHI_E | E:1.0 | M:Sostituisci | Click o [Fuochi/Vertici/Estendi/Accorcia/Modo/Uscita] <Uscita>:

- **Fuochi**: se già attivo (ON), commuta il tipo forzato tra ELLISSE e IPERBOLE (toggle opt_f_type) e ridisegna il luogo ortogonale. Se era spento (OFF), lo accende. Se era già attivo e la conica precedente esiste, la ridisegna con i nuovi parametri.
- **Vertici**: spegne i Fuochi e accende il luogo ortogonale dei Vertici.
- Estendi/Accorcia, Modo, Uscita: come standard.
- **Punto cliccato**: il sistema identifica il target più vicino sul luogo ortogonale tramite CON4P_Identify_Orthogonal_Target e disegna la conica corrispondente.

Loop principale (tutti gli altri tipi topologici)

C:ON F:ON V:ON T:Luogo | E:1.0 | M:Sostituisci | Punto o [Centri/Fuochi/Vertici/Estendi/Accorcia/Target/Speciali/Modo/Uscita] <Uscita>:

- **Centri** (opt_gc, ON/OFF): accende/spegne il luogo dei Centri (curva VLA).
- **Fuochi** (opt_gf, ON/OFF): accende/spegne la nuvola dei Fuochi.
- **Vertici** (opt_gv, ON/OFF): accende/spegne la nuvola dei Vertici.
- **Target** (opt_t, Luogo/Nodo): in Luogo, il click cerca il punto geometrico più vicino su qualunque luogo acceso (CON4P_Identify_Locus_Target). In Nodo, il click identifica il punto base più vicino e accede alle soluzioni nodali precalcolate, ciclando tra le soluzioni multiple ad ogni click ripetuto (CON4P_Identify_Node_Target).
- **Estendi/Accorcia**: modifica opt_e; se una conica esiste già la ridisegna (CON4P_Refresh_Last_Conic).
- **Speciali**: apre il sottomenu (vedi sotto).
- **Modo**: commuta Sostituisci/Aggiungi.
- **Uscita**: termina con pulizia tramite CON4P_Geom_Master_Exit_Handler.

Sottomenu "Speciali"

[TIPO] *** SPECIALI *** | M:Sostituisci | Scegli [Ellisse/Iperbole/Parabola/Modo/Generiche] <Generiche>:

- **Ellisse**: disegna l'ellisse pseudo-circolare del fascio (se disponibile), altrimenti segnala "Nessuna Ellisse Speciale disponibile per questa configurazione."
- **Iperbole**: disegna l'iperbole equilatera del fascio (se disponibile), altrimenti segnala "Nessuna Iperbole Equilatera disponibile per questa configurazione."
- **Parabola**: disegna una parabola del fascio corrente (se disponibile), altrimenti segnala "Nessuna Parabola disponibile per questa configurazione.". Se presenti due parabole, pressioni ripetute ciclano tra le due.
- **Modo**: commuta Sostituisci/Aggiungi all'interno del sottomenu.
- **Generiche** o Invio: torna al loop principale.

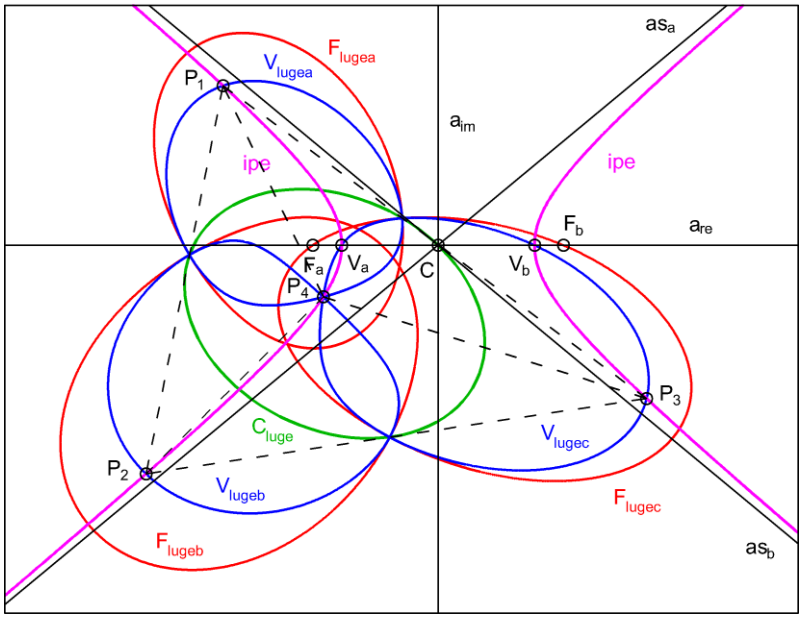

Figura 26 *– CON4P – Un'iperbole ipe passante da quattro punti fissi P_1 P_2 P_3 P_4 posti in una configurazione concava non ortocentrica e determinata tramite l'appartenenza del suo centro C al luogo geometrico C_{luge} o di uno dei suoi vertici V_a V_b al luogo geometrico V_{luge} o di uno dei suoi fuochi F_a F_b al luogo geometrico F_{luge}. Per ogni iperbole, oltre ai vertici V_a V_b, sono mostrati: i fuochi F_a F_b, gli assi a_{re} a_{im} e gli asintoti as_a as_b.*

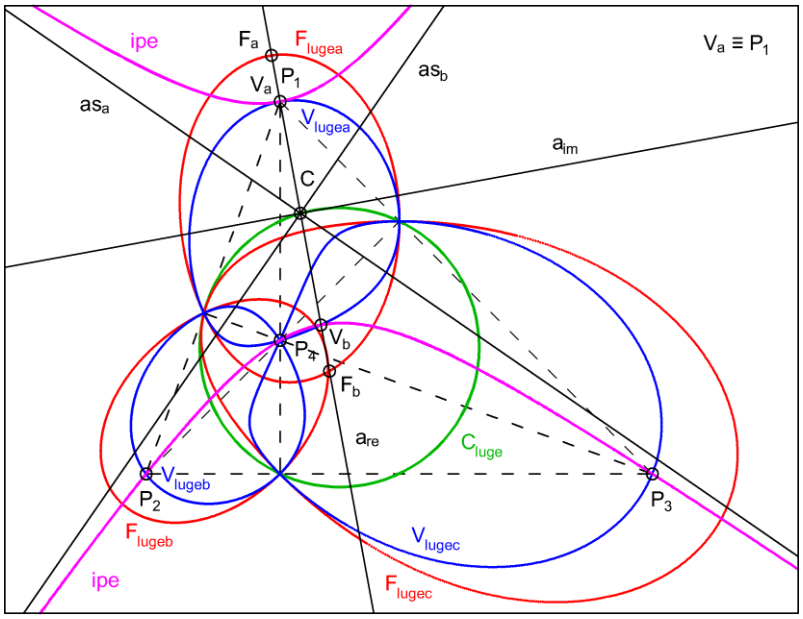

Figura 27 *– CON4P – Un'iperbole ipe passante da quattro punti fissi P_1 P_2 P_3 P_4 posti in una configurazione concava non ortocentrica e determinata tramite la coincidenza di uno dei suoi vertici (V_a) con uno dei punti fissi (P_1) assegnati. In figura sono presenti i luoghi geometrici C_{luge} F_{luge} V_{luge} di possibile collocazione di centro fuochi e vertici di una conica ammissibile per quella quaterna di punti assegnati. Per ogni iperbole, oltre ai vertici V_a V_b, sono indicati: i fuochi F_a F_b, gli assi a_{re} a_{im} e gli asintoti as_a as_b.*

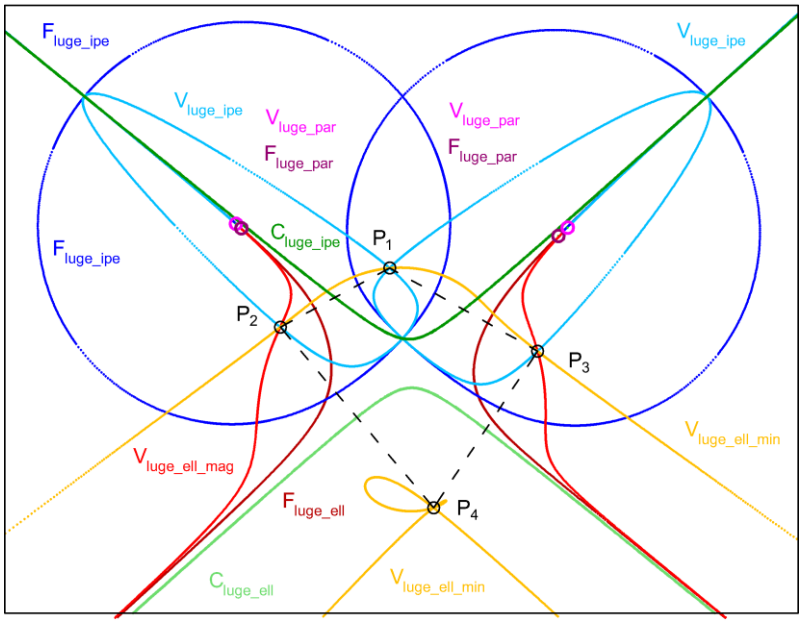

Figura 28 *– CON4P – Dati quattro punti fissi P_1 P_2 P_3 P_4 posti in una configurazione convessa generica sono determinati i luoghi geometrici di possibile collocazione di centro (C_{luge}) fuochi (F_{luge}) e vertici (V_{luge}) dei vari tipi di conica ammissibile (ell ipe par) per quella quaterna di punti assegnati. Tali luoghi geometrici sono distinti tramite colorazione sia per il tipo punto a cui fanno da supporto (C F V) sia per il tipo di conica (ell ipe par) che risulterà determinata: C_{luge_ell} C_{luge_ipe} F_{lug_ell} F_{luge_ipe} F_{lug_par} $V_{luge\_ell\_mag}$ $V_{luge\_ell\_min}$ V_{luge_ipe} V_{luge_par}. I luoghi geometrici relativi alla parabola sono definiti entrambi da soli due punti specificatamente evidenziati.*

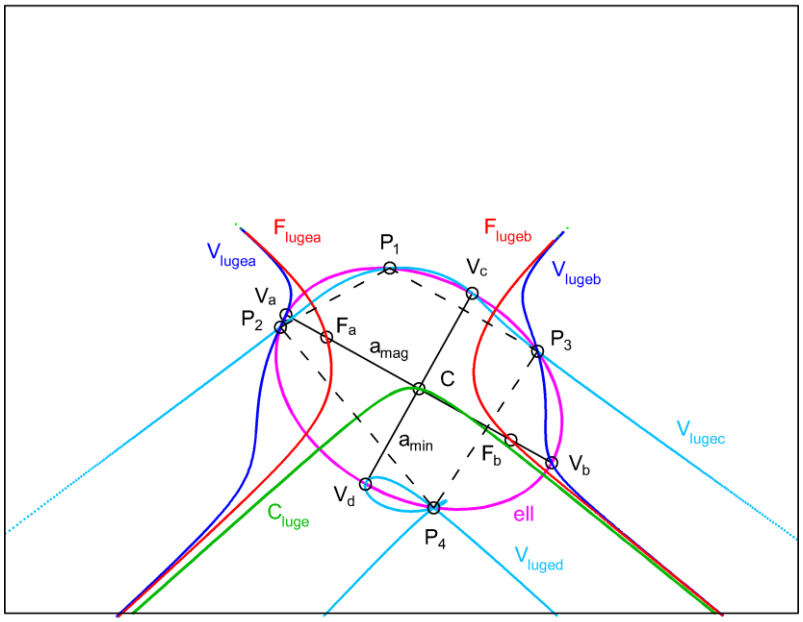

Figura 29 *– CON4P – Un'ellisse ell passante da quattro punti fissi P_1 P_2 P_3 P_4 posti in una configurazione convessa generica e determinata tramite l'appartenenza del suo centro C al luogo geometrico C_{luge} o di uno dei suoi vertici maggiori V_a V_b o minori V_c V_d al luogo geometrico V_{lugea} V_{lugeb} o V_{lugec} V_{luged} o di uno dei suoi fuochi F_a F_b al luogo geometrico F_{lugea} F_{lugeb}. Per ogni ellisse, oltre ai vertici V_a V_b, sono mostrati: i fuochi F_a F_b, gli assi a_{mag} a_{min}.*

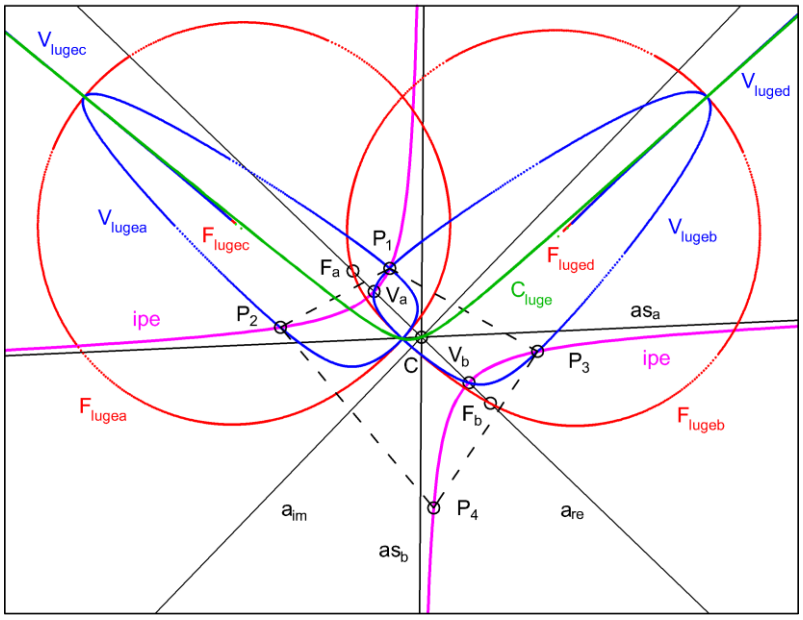

Figura 30 *– CON4P – Un'iperbole ipe passante da quattro punti fissi P_1 P_2 P_3 P_4 posti in una configurazione convessa generica e determinata tramite l'appartenenza del suo centro C al luogo geometrico C_{luge} o di uno dei suoi vertici V_a V_b al luogo geometrico V_{lugea} V_{lugeb} V_{lugec} V_{luged} o di uno dei suoi fuochi F_a F_b al luogo geometrico F_{lugea} F_{lugeb} F_{lugec} F_{luged}. Per ogni iperbole, oltre ai vertici V_a V_b, sono mostrati: i fuochi F_a F_b, gli assi a_{re} a_{im} e gli asintoti as_a as_b.*

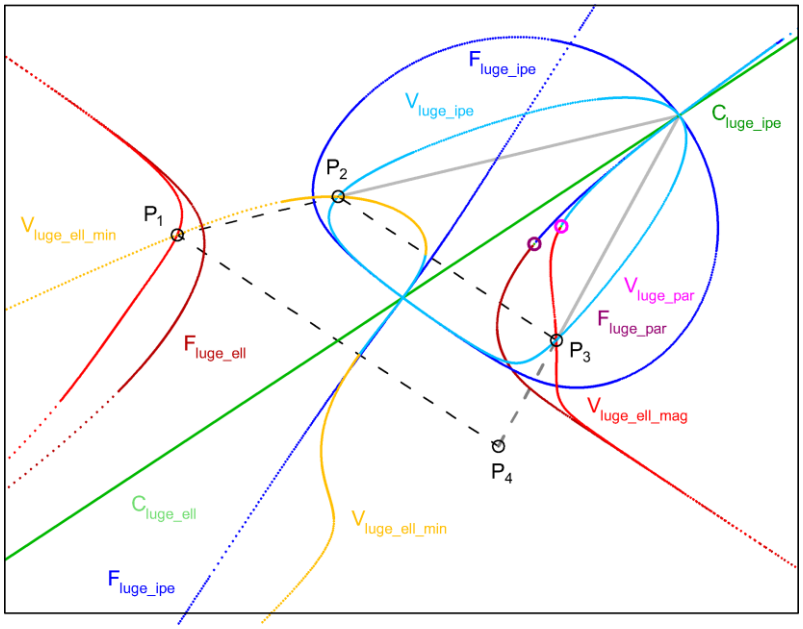

Figura 31 *– CON4P – Dati quattro punti fissi P_1 P_2 P_3 P_4 posti in una configurazione convessa a trapezio, sono determinati i luoghi geometrici di possibile collocazione di centro (C_{luge}) fuochi (F_{luge}) e vertici (V_{luge}) dei vari tipi di conica ammissibile (ell ipe par) per quella quaterna di punti assegnati. Tali luoghi geometrici sono distinti tramite colorazione sia per il tipo punto a cui fanno da supporto (C F V) sia per il tipo di conica (ell ipe par) che risulterà determinata: C_{luge_ell} C_{luge_ipe} F_{lug_ell} F_{luge_ipe} F_{lug_par} $V_{luge\_ell\_mag}$ $V_{luge\_ell\_min}$ V_{luge_ipe} V_{luge_par}. I luoghi geometrici relativi alla parabola sono definiti entrambi da un solo punto specificatamente evidenziato.*

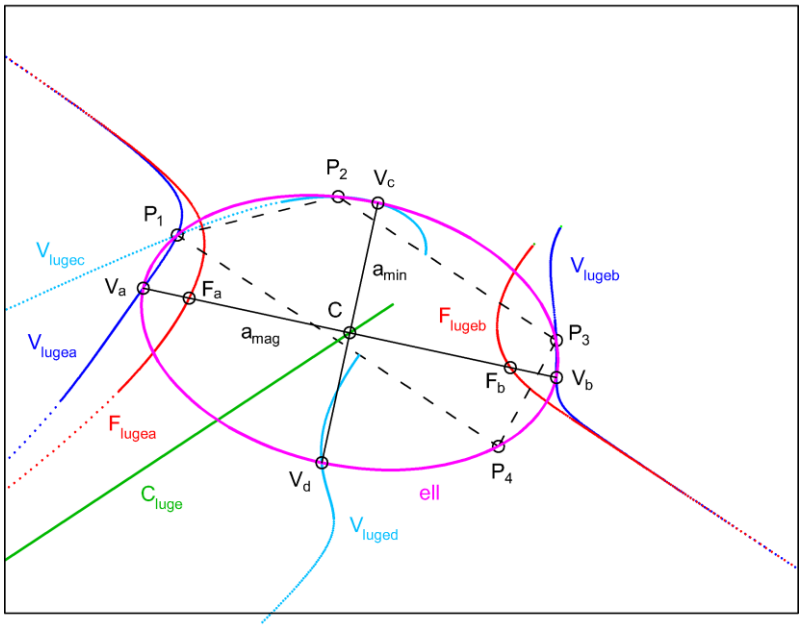

Figura 32 *– CON4P – Un'ellisse ell passante da quattro punti fissi P_1 P_2 P_3 P_4 posti in una configurazione convessa a trapezio e determinata tramite l'appartenenza del suo centro C al luogo geometrico C_{luge} o di uno dei suoi vertici maggiori V_a V_b o minori V_c V_d al luogo geometrico V_{lugea} V_{lugeb} o V_{lugec} V_{luged} o di uno dei suoi fuochi F_a F_b al luogo geometrico F_{lugea} F_{lugeb}. Per ogni ellisse, oltre ai vertici V_a V_b, sono mostrati: i fuochi F_a F_b, gli assi a_{mag} a_{min}.*

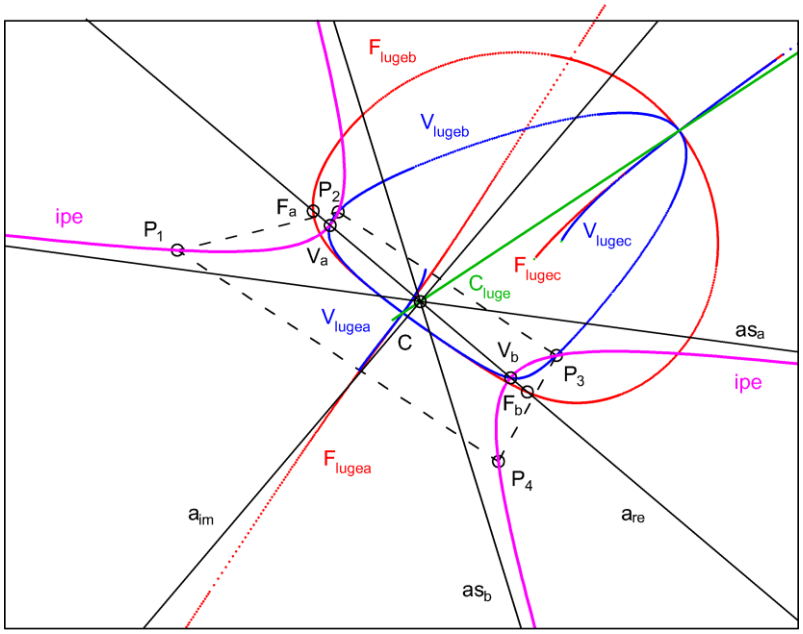

Figura 33 *– CON4P – Un'iperbole ipe passante da quattro punti fissi P_1 P_2 P_3 P_4 posti in una configurazione convessa a trapezio e determinata tramite l'appartenenza del suo centro C al luogo geometrico C_{luge} o di uno dei suoi vertici V_a V_b al luogo geometrico V_{lugea} V_{lugeb} V_{lugec} o di uno dei suoi fuochi F_a F_b al luogo geometrico F_{lugea} F_{lugeb} F_{lugec}. Per ogni iperbole, oltre ai vertici V_a V_b, sono mostrati: i fuochi F_a F_b, gli assi a_{re} a_{im} e gli asintoti as_a as_b.*

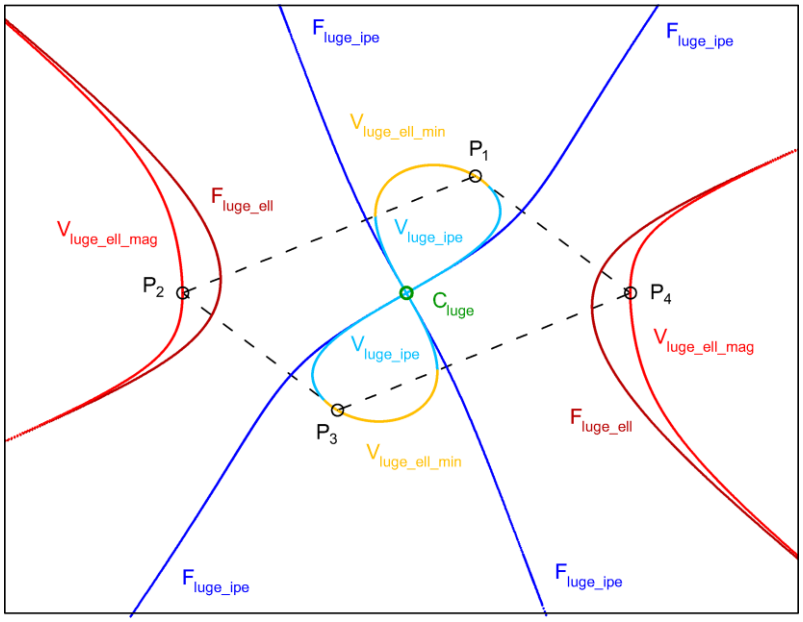

Figura 34 *– CON4P – Dati quattro punti fissi P_1 P_2 P_3 P_4 posti in una configurazione convessa a parallelogramma, sono determinati i luoghi geometrici di possibile collocazione di centro (C_{luge}) fuochi (F_{luge}) e vertici (V_{luge}) dei vari tipi di conica ammissibile (ell ipe) per quella quaterna di punti assegnati. Tali luoghi geometrici sono distinti tramite colorazione sia per il tipo punto a cui fanno da supporto (C F V) sia per il tipo di conica (ell ipe) che risulterà determinata: C_{luge} F_{lug_ell} F_{luge_ipe} $V_{luge\_ell\_mag}$ $V_{luge\_ell\_min}$ V_{luge_ipe}. Il luogo geometrico relativo al centro è definito da un solo punto specificatamente evidenziato. I luoghi geometrici relativi alla parabola sono assenti in quanto conica non ammissibile.*

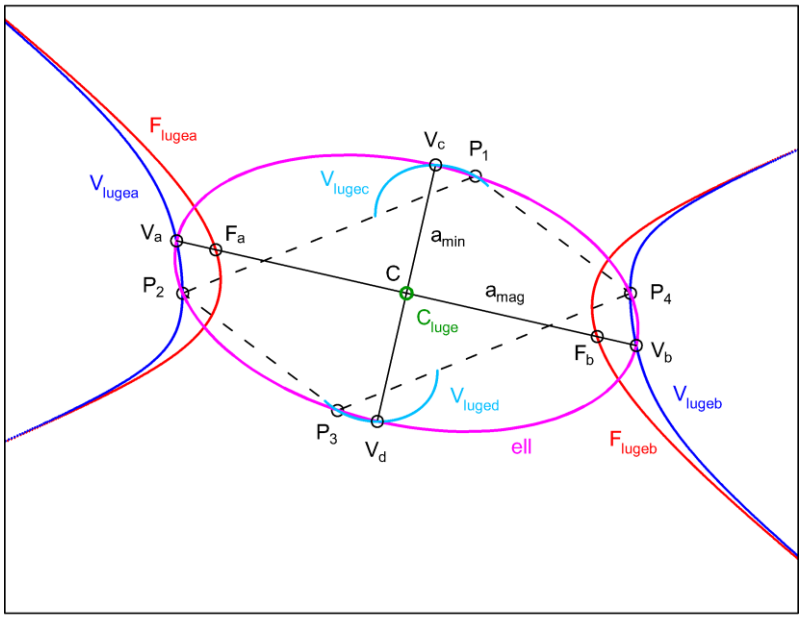

Figura 35 *– CON4P – Un'ellisse ell passante da quattro punti fissi P_1 P_2 P_3 P_4 disposti a parallelogramma e determinata tramite la coincidenza del suo centro C con il luogo geometrico puntiforme C_{luge} o tramite l'appartenenza di uno dei suoi vertici maggiori V_a V_b o minori V_c V_d al luogo geometrico V_{lugea} V_{lugeb} o V_{lugec} V_{luged} o di uno dei suoi fuochi F_a F_b al luogo geometrico F_{lugea} F_{lugeb}. Per ogni ellisse, oltre ai vertici V_a V_b, sono mostrati: i fuochi F_a F_b, gli assi a_{mag} a_{min}.*

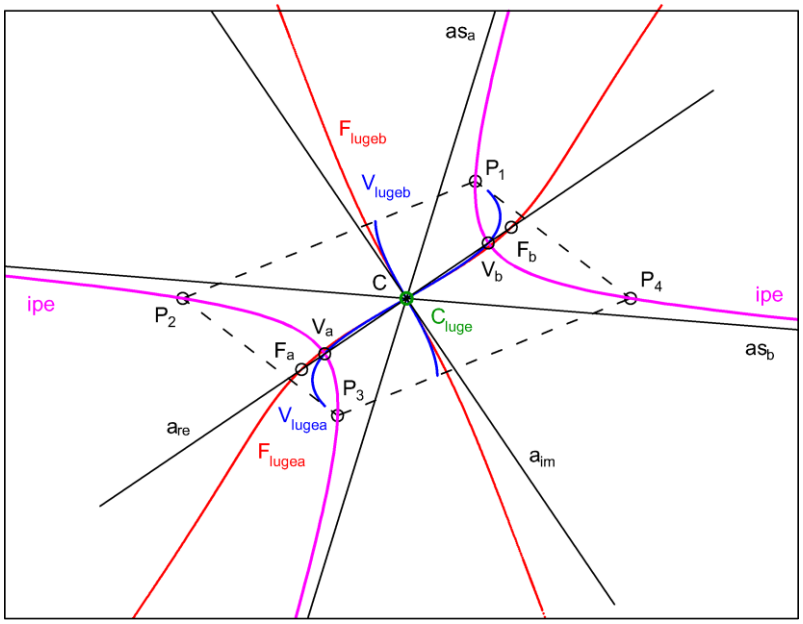

Figura 36 *– CON4P – Un'iperbole ipe passante da quattro punti fissi P_1 P_2 P_3 P_4 disposti a parallelogramma e determinata tramite la coincidenza del suo centro C al luogo geometrico puntiforme C_{luge} o tramite l'appartenenza di uno dei suoi vertici V_a V_b al luogo geometrico V_{lugea} V_{lugeb} o di uno dei suoi fuochi F_a F_b al luogo geometrico F_{lugea} F_{lugeb}. Per ogni iperbole, oltre ai vertici V_a V_b, sono mostrati: i fuochi F_a F_b, gli assi a_{re} a_{im} e gli asintoti as_a as_b.*

CON5P – Conica dati quattro punti fissi P_1 P_2 P_3 P_4 e un punto riposizionabile P_5

Quattro punti fissi P_1 P_2 P_3 P_4 di passaggio forniscono quattro dei cinque vincoli necessari a definire univocamente una conica e sono in grado di definire a seconda delle configurazioni del quadrangolo le due parabole o coniche degeneri (due rette parallele) di separazione tra aree di posizionamento di un 5° punto iperbolico, parabolico, ellittico. Un ulteriore punto riposizionabile P_5 di passaggio consente la determinazione univoca della conica.

Acquisizione iniziale

Acquisisce **4 punti base**. Disegna il quadrangolo (Geom_Draw_Quadrangolo_Smart) e le eventuali parabole guida (Geom_5P_Update_Guides) eventualmente degenerate in rette parallele.

Prompt del loop per il 5° punto

G:ON | M:Sostituisci | 5° punto o [Guide/Modo/Fine] <Fine>:

- **Guide** (ON/OFF): accende/spegne le **parabole guida**, ovvero le parabole del fascio passante per i 4 punti base. Queste parabole separano le regioni del piano dove il 5° punto produce un'ellisse da quelle dove produce un'iperbole e per appartenenza del 5° punto esso produce quella stessa parabola.
- **Modo**: Sostituisci/Aggiungi.
- **Fine** o Invio: termina.
- **Punto cliccato**: marca il punto (Draw_Point_R), calcola e disegna la conica completa (Draw_Con5P_Nor_Compl) con assi, fuochi e costruzioni. L'area di calcolo viene estesa tramite Util_Scale_Pts_From_Centroid (scala 5× dal baricentro).

CON5PREA – Conica dati 5 punti singolarmente riposizionabili P_1 P_2 P_3 P_4 P_5

Identico a con5p salvo consentire il riposizionamento di uno qualsiasi dei cinque punti forniti.

Flusso operativo

Fase A: Acquisizione del 5° punto

G:ON | Specifica 5° punto o [Guide/Fine] <Fine>:

- **Guide** (ON/OFF): parabole guida come in c:con5p.
- **Punto cliccato**: piazza il 5° punto e disegna la prima conica.
- **Fine**: termina senza entrare nella Fase B.

Fase B: Editing reattivo a impalcatura persistente

G:ON | Seleziona vertice da muovere o [Guide/Fine] <Fine>:

L'OSMODE viene forzato a 8 (snap "Nodo") per agganciare direttamente uno dei 5 punti. L'utente:

- Clicca vicino a uno dei 5 punti: il sistema identifica il più vicino (per distanza minima), separa i 4 punti residui (anchors), aggiorna il quadrangolo e le guide sui 4 punti residui, poi chiede "Nuova posizione:" con getpoint (con elastico dal punto originale).
- Se l'utente conferma la nuova posizione: il punto viene spostato nella lista dei 5, la conica vecchia viene cancellata, e la nuova viene calcolata e disegnata alla posizione aggiornata. Il ciclo si ripete.
- **Guide** (ON/OFF): commuta le parabole guida durante l'editing.
- **Fine**: termina.

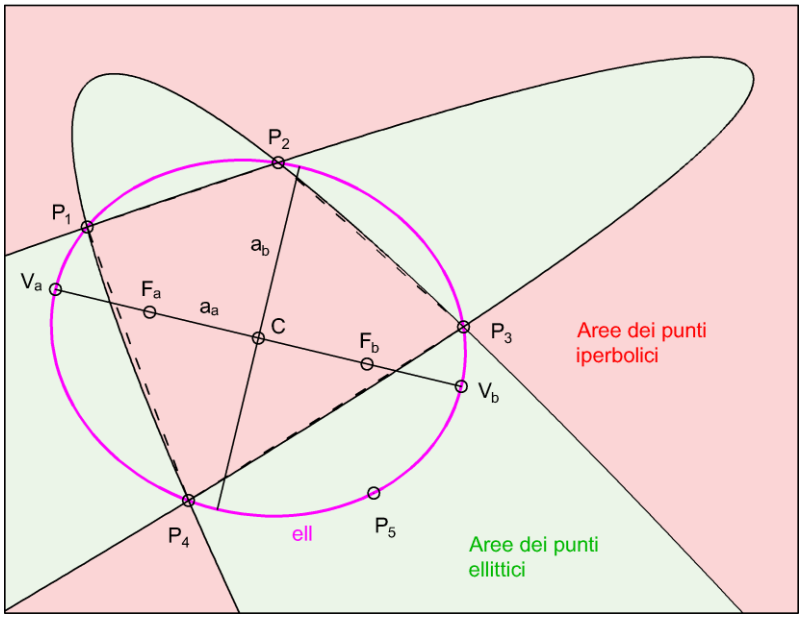

Figura 36 *– CON5P – Un'ellisse ell passante da quattro punti fissi P_1 P_2 P_3 P_4 disposti a quadrangolo convesso generico e da un punto riposizionabile P_5 posto in un'area (verde) di punti ellittici in grado di saturare i cinque gradi di libertà determinando un'ellisse.*

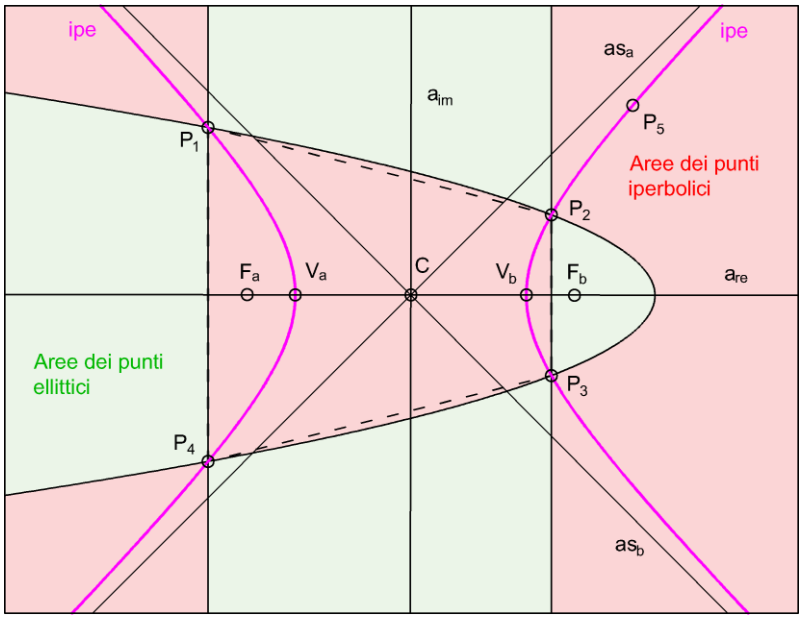

Figura 37 *– CON5P – Un'iperbole ipe passante da quattro punti fissi P_1 P_2 P_3 P_4 disposti a quadrangolo convesso a trapezio e da un punto riposizionabile P_5 posto in un'area (rossa) di punti iperbolici in grado di saturare i cinque gradi di libertà determinando un'iperbole.*

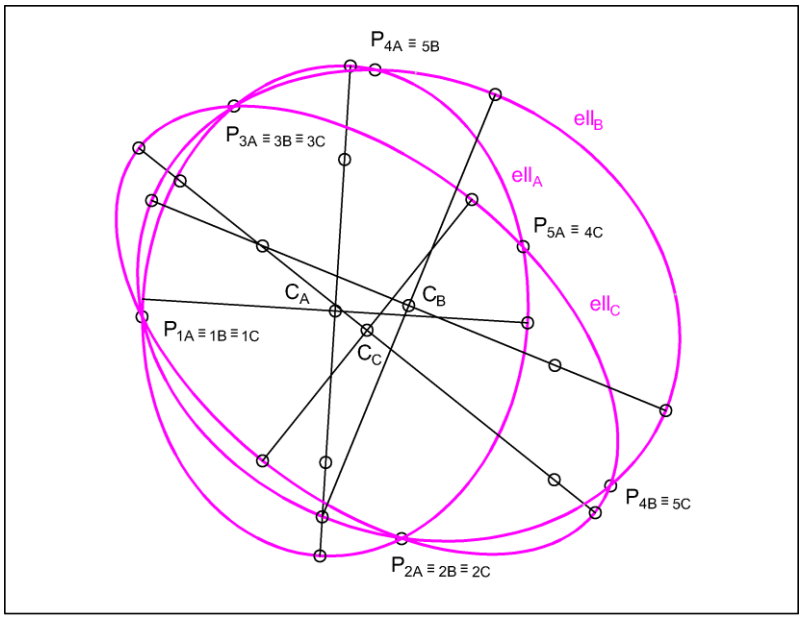

Figura 38 *– CON5PREA – Una serie di ellissi ell_A ell_B ell_C passanti da cinque punti riposizionabili P_1 P_2 P_3 P_4 P_5. Le tre ellissi condividono gli stessi primi tre punti (P_{1A} P_{1B} P_{1C}) (P_{2A} P_{2B} P_{2C}) (P_{3A} P_{3B} P_{3C}) che non sono stati riposizionati e condividono a coppie gli altri due punti (P_{4A} $P5_B$) (P_{5A} P_{4C}) (P_{4B} P_{5C}) che sono stati riposizionati.*

BIBLIOGRAFIA

- Anzani G., An algorithm in AutoLisp to draw an ellipse inscribed in a generic quadrangle based on the projective geometry, Lulu, Raleigh 2014.
- Aschieri F., Lezioni di geometria projettiva e descrittiva. Volume primo: Geometria projettiva, Hoepli, Milano 1883.
- Aschieri F., Lezioni di geometria projettiva e descrittiva. Volume secondo: Geometria descrittiva, Hoepli, Milano 1884.
- Aschieri F., Lezioni di geometria projettiva (seconda edizione con aggiunte e correzioni), Hoepli, Milano 1888.
- Campedelli L., Esercitazioni complementari di geometria (a cura di A. Barlotti), CEDAM, Padova 1952.
- Campedelli L., Esercitazioni di geometria analitica e proiettiva, CEDAM, Padova 1956.
- Campedelli L., Lezioni di geometria. Volume primo: La geometria analitica e gli elementi della geometria proiettiva, CEDAM, Padova 1967.
- Campedelli L., Lezioni di geometria. Volume secondo, parte prima: I metodi di rappresentazione della geometria descrittiva, CEDAM, Padova 1967.
- Campedelli L., Lezioni di geometria. Volume secondo, parte seconda: Le curve e le superficie, CEDAM, Padova 1966.
- Dino N. S., Elementi di geometria proiettiva, Morano, Napoli 1885.
- Enriques F., Lezioni di geometria descrittiva (a cura di U. Concina), Zanichelli, Bologna 1925.
- Enriques F., Lezioni di geometria proiettiva (ristampa anastatica della seconda edizione aumentata del 1904), Zanichelli, Bologna 1996.
- Fiedler G., Trattato di Geometria descrittiva (traduzione di A. Sayno e E. Padova), Le Monnier, Firenze 1874.
- Marletta G., Geometria proiettiva delle forme di 1a e 2a specie, Casa del Libro, Catania 1944.
- Morin U., Lezioni di geometria. Parte I: Elementi di geometria analitica, CEDAM, Padova 1953.
- Morin U., Lezioni di geometria. Parte II: Curve piane, CEDAM, Padova 1954.
- Morin U., Lezioni di geometria. Parte III: Elementi di geometria proiettiva, CEDAM, Padova 1955.
- Morin U., Lezioni di geometria. Parte IV: Superficie e curve sghembe, CEDAM, Padova 1953.
- Rollero A., Esercizi di geometria analitica e proiettiva. Parte terza: Geometria proiettiva, Edizioni Scientifiche, Genova 1951.
- Rollero A., Elementi di geometria analitica e proiettiva, Edizioni Scientifiche, Genova 1970.
- Villa M., Lezioni di Geometria. Volume primo, CEDAM, Padova 1965.

Appendici tecniche: documentazione algoritmica e riferimenti del codice

A fondamento dell'impianto teorico e metodologico fin qui esposto, l'apparato algoritmico sviluppato in ambiente AutoLISP non si configura come un mero espediente esecutivo, ma costituisce il nucleo dimostrativo dell'intera indagine proiettiva. Al fine di garantire la totale trasparenza logica e la rigorosa riproducibilità scientifica della ricerca, la presente pubblicazione si correda di quattro appendici tecniche, destinate alla scomposizione analitica e alla documentazione del software.

- **Implementazione algoritmica**, illustra l'architettura generale a strati del codice, definendo il flusso esecutivo che dal nucleo matematico puro conduce fino all'infrastruttura dei cicli interattivi per la navigazione topologica.
- **Categorizzazione funzionale**, ordina l'imponente libreria sviluppata, raggruppando l'intero set di funzioni in sedici distinte categorie operative omogenee.
- **Glossario delle funzioni**, costituisce il dizionario analitico del sistema, strutturato secondo un formato di catalogazione rigoroso che, per ogni singola routine, esplicita sistematicamente lo scopo, i parametri in input, i dati in output e la relativa logica interna.
- **Codice del listato**, espone integralmente la sintassi computazionale, consegnando al lettore la traduzione esatta delle regole geometriche descrittive nel linguaggio della macchina.

IMPLEMENTAZIONE ALGORITMICA

Il listato descritto è una libreria AutoLISP dedicata al disegno parametrico e interattivo delle sezioni coniche in AutoCAD. Il progetto implementa 15 comandi AutoCAD e 472 funzioni organizzate in 16 categorie funzionali, strutturate in strati sovrapposti dal nucleo matematico puro fino ai comandi interattivi registrati in AutoCAD.

Obiettivo generale

Lo scopo primario è risolvere il problema del disegno delle coniche in AutoCAD in modo completamente parametrico: l'utente può fornire combinazioni di punti, fuochi, vertici o direzioni d'asse, e il motore geometrico individua automaticamente la conica (o la famiglia) che soddisfa i vincoli. Il sistema distingue tra soluzioni proprie (centro finito: ellissi e iperboli) e improprie (centro all'infinito: parabole), e gestisce singolarità numeriche con protezioni integrate a ogni livello.

Architettura a strati

L'infrastruttura del sistema è rigorosamente stratificata al fine di isolare le responsabilità computazionali. Il flusso dei dati procede da un nucleo matematico e vettoriale puro – indipendente dall'ambiente CAD – attraversando una rete di guardie logiche e predicati topologici, fino a raggiungere i motori di scansione geometrica e i sottosistemi di disegno grafico. Questa architettura modulare, suddivisa in sedici distinte categorie funzionali (numerate da A ad I), garantisce la stabilità del ciclo esecutivo ed è oggetto di disamina analitica nella successiva appendice dedicata alla Categorizzazione funzionale.

Flusso esecutivo standard

- Inizializzazione: Salvataggio variabili di sistema (Init_Conic_App): OSMODE, PDMODE, CMDECHO.
- Acquisizione: Prompt interattivi (getpoint, entsel, getkword, getint).
- Validazione: Predicati P_ per coincidenza, allineamento, degenerazioni.

- Calcolo: Motore Geom_/Math_ specifico per tipo di conica e vincoli.
- Scansione: Scansione parametrica adattiva (D3) sul parametro angolare dell'asse.
- Disegno: Primitive Draw_ (E) → entità AutoCAD nel documento.
- Reattività: Loop ricalcolo+ridisegno a ogni spostamento cursore (ove presente).
- Cleanup: Ripristino ambiente (Restore_Conic_App), rimozione entità temporanee.

Scansione parametrica adattiva (categoria D3)

Il problema di trovare la conica che soddisfa N vincoli viene ridotto a una ricerca monodimensionale sul parametro angolare dell'asse. La scansione procede in tre fasi: (1) scan grossolano a passo variabile (più fine dove il vertice è vicino ai punti), (2) bracket topologico per isolare il minimo della funzione di errore, (3) raffinamento ricorsivo fino a soglia = max_dist/100000 (convergenza sub-pixel). La versione "Fuochi" duplica il meccanismo per ottimizzare la posizione dei fuochi indipendentemente da quella del vertice.

Sicurezza numerica

Protezioni sistematiche: guard (and p (listp p)) prima di ogni distanza; test (> (abs den) 1e-9) prima di ogni divisione; clamping del dominio di acos a [−1,1]; limitatori di densità della griglia di scansione; Math_Acos reimplementa l'arcocoseno mancante in AutoLISP; Geom_Refine_Internal_Recursive usa un threshold = max_dist/100000 come criterio di stop.

CATEGORIZZAZIONE FUNZIONALE

Le 472 funzioni del listato sono organizzate in 16 categorie funzionali che separano nettamente il calcolo geometrico dal controllo grafico in AutoCAD. Nelle tabelle che seguono suddivise per categorie ogni funzione è riportata con lo scopo effettivo derivato direttamente dall'analisi del codice sorgente.

A — Matematica di Base e Algebra [65 funzioni]

Il gruppo A costituisce il nucleo di calcolo elementare della libreria, indipendente dall'ambiente grafico AutoCAD. Queste 65 funzioni svolgono il ruolo di "primitive matematiche" invocate trasversalmente da ogni altro modulo del sistema. La logica di raggruppamento è finalizzata a isolare le operazioni di base su cui poggia l'intera geometria proiettiva del listato:

- Analisi numerica e algebrica: risoluzione di equazioni di secondo e terzo grado, gestione di determinanti ciclici e prodotti di somme/differenze.
- Geometria analitica elementare: calcolo di aree, angoli orientati, equazioni della retta e gestione della forma quadratica centrata.
- Operazioni su punti e vettori: rotazioni di punti (tramite seno/coseno precompilati), calcoli di ipotenusa (Pitagora) e trasformazioni lagrangiane.
- Classificazione geometrica: analisi preliminare dei punti per rilevare parallelismi, ortogonalità e coincidenze, fondamentali per la classificazione topologica richiesta dalle categorie D1-D4.

Questo modulo garantisce che i motori di calcolo di alto livello ricevano dati già coerenti e validati numericamente, centralizzando le operazioni aritmetiche per evitare ridondanze nei comandi.

B — Algebra Vettoriale e Simmetrie [12 funzioni]

Questo raggruppamento costituisce il modulo di manipolazione spaziale della libreria. Mentre la categoria A fornisce le primitive aritmetiche, il modulo B definisce le regole di interazione tra gli enti nello spazio 2D e 3D. Le 12 funzioni qui presenti sono indispensabili per tradurre le coordinate grezze di

AutoCAD in entità vettoriali manipolabili dal motore geometrico. La logica di questo modulo è duplice:

- Operazioni vettoriali fondamentali: Implementa il calcolo di somme, differenze, prodotti scalari, moduli (tramite routine Pitag_Ip) e normalizzazione dei versori (Vet_Uni), che fungono da base per ogni calcolo di distanza o direzione.
- Estensioni geometriche e simmetrie: Include le funzioni per la determinazione del punto medio e le routine avanzate per le simmetrie (es. Geom_1Pu_Simm_PuC e Geom_2Pu_Simm_PuC), che permettono di generare coppie di punti simmetrici rispetto a centri o perni con offset predefiniti.

Questo gruppo è essenziale per garantire che le trasformazioni spaziali avvengano con coerenza: ogni comando che necessita di posizionare un punto in relazione a un centro (come avviene nel calcolo dei vertici o dei fuochi nelle categorie D2-D4) deve necessariamente appoggiarsi a questo strato di algebra vettoriale.

C — Predicati Geometrici (P_…) [57 funzioni]

Questo modulo rappresenta la "logica di guardia" dell'intero sistema. Le 57 funzioni che lo compongono non eseguono calcoli di costruzione (affidati al Gruppo D), ma operano come verificatori booleani (risposta T/nil) per testare le proprietà intrinseche della configurazione geometrica fornita dall'utente. La funzione di questo raggruppamento è duplice:

- Validazione dei dati: Prima che i motori di tracciamento avviino cicli di calcolo intensivi, queste funzioni testano la fattibilità del problema, verificando ad esempio se tre o più punti sono allineati, se un sistema di quattro punti è degenere o se una configurazione ammette un sistema ortocentrico.
- Classificazione morfologica: Il gruppo include una suite completa di test per determinare la natura geometrica del quadrangolo base (distinzione tra trapezio scaleno/isoscele, parallelogramma, rettangolo, rombo, quadrato) e la posizione relativa dei punti (interno/esterno a triangoli o quadrangoli).

L'importanza di questo gruppo è critica per la stabilità dell'applicativo: agendo come filtro logico, impedisce la propagazione di errori matematici e orienta dinamicamente le scelte degli algoritmi di costruzione (Gruppo D) in base alle proprietà verificate, garantendo che ogni conica sia generata solo se le premesse geometriche sono matematicamente coerenti.

Gruppo D — Geometria delle Coniche [68 funzioni]

Il Gruppo D rappresenta il cuore computazionale della libreria 10020. Non si limita alla mera rappresentazione grafica, ma costituisce il motore di risoluzione proiettiva che traduce i vincoli geometrici in traiettorie definite. La suddivisione in quattro sotto-categorie riflette la stratificazione necessaria per gestire la complessità delle coniche, operando come un sistema a cascata dove i dati passano dall'analisi topologica alla risoluzione analitica e infine al raffinamento del luogo geometrico.

- **D1 — Topologia e Ordinamento Spaziale [7 funzioni]:** Queste routine fungono da pre-processori dell'intero sistema. Il loro compito fondamentale è l'analisi morfologica: ordinano i punti in senso antiorario (CCW), classificano la configurazione del quadrangolo (concavo, convesso, trapezio) e identificano le degenerazioni. La loro presenza è obbligatoria per garantire che i successivi motori di calcolo operino su strutture dati coerenti, eliminando alla radice l'ambiguità geometrica.
- **D2 — Calcolo Parametrico delle Coniche [48 funzioni]:** Il motore analitico centrale. Qui risiedono le routine che estraggono i parametri geometrici fondamentali (centro, vertici, fuochi) nel sistema globale WCS. La funzione di questo raggruppamento è la trasformazione dei vincoli in coefficienti algebrici: sono le funzioni che "risolvono" la conica, agendo come interfaccia diretta tra la teoria geometrica e la formulazione matematica.

- **D3 — Scansione Adattiva e Raffinamento Ricorsivo [9 funzioni]:** Questo gruppo gestisce la precisione del tracciamento nei casi critici. Implementano algoritmi di scansione adattiva e convergenza ricorsiva che operano in ambienti dove non è disponibile una soluzione chiusa immediata. Sono raggruppate perché condividono la logica di ottimizzazione "a forcella" (bracket), fondamentale per garantire che la conica rispetti i vincoli con tolleranza sub-pixel.
- **D4 — Motori di Azione Conica [4 funzioni]:** Gli orchestratori strategici. Non contengono logica di calcolo pura, ma definiscono la strategia di risoluzione (ad esempio, scegliendo tra il fascio di Pascal o la minimizzazione dell'eccentricità). La loro funzione è smistare i dati tra i risolutori D2 e gli algoritmi di raffinamento D3, coordinando il flusso di lavoro in base al tipo di conica rilevata dopo l'analisi topologica iniziale.

E — Disegno di Entità Grafiche (Draw_) [43 funzioni]

Questo modulo funge da strato di uscita verso l'ambiente AutoCAD. La funzione di queste 43 routine non è il calcolo geometrico (delegato ai gruppi D1-D4), ma la materializzazione grafica dei dati. Il raggruppamento è strutturato per diversificare l'output in base alle necessità del comando in esecuzione:

- Primitive complesse: routine che disegnano la conica insieme ai suoi elementi costruttivi (assi, fuochi, rette di guida, triangoli di costruzione), essenziali per la verifica visiva della correttezza proiettiva.
- Primitive ottimizzate: le varianti _Light ed eseguibili per i loop reattivi, progettate per ridurre il carico computazionale durante il disegno in tempo reale, emettendo solo la curva principale (pline o spline) senza elementi accessori.
- Sottosistemi di disegno: includono gestori specifici come Geom_Draw_Quadrangolo_Smart per il disegno del quadrangolo base, la gestione della sagoma Dima (Geom_Draw_Dima_Boundaries) e le routine di rendering VLA per ellissi e iperboli, garantendo che l'entità grafica risultante sia conforme alle specifiche richieste dall'algoritmo di calcolo.

Questo gruppo è il ponte tra l'astrazione algebrica e il disegno finito: ogni funzione qui contenuta traduce un parametro calcolato in una entità (LINE, LWPOLYLINE, SPLINE) all'interno del layer dedicato (SLC), garantendo una separazione netta tra la logica di progetto e la rappresentazione grafica finale.

F — Gestori di Topologia 4P (Handler_) [5 funzioni]

Queste 5 funzioni rappresentano il modulo di "smistamento decisionale" tra l'analisi topologica e l'esecuzione algoritmica. Il loro ruolo è tradurre la classificazione morfologica ottenuta dai test del Gruppo C e le analisi di D1 nella strategia operativa corretta per il tracciamento della conica del fascio. La logica di questo gruppo è puramente orchestrativa:

- Intermediazione operativa: Gli Handler ricevono in input i 4 punti già classificati (es. "configurazione concava", "trapezio", "simmetria centrale") e richiamano la funzione di calcolo (Geom_Conic_*_Action) corrispondente.
- Pre-ordinamento: Assicurano che il set di punti sia pre-ordinato correttamente prima di passare il controllo ai motori di calcolo, garantendo che le condizioni al contorno (come la posizione del punto interno in una configurazione concava) siano rispettate.
- Separazione delle responsabilità: Questo raggruppamento permette di isolare la logica dei "casi limite" (singolarità, configurazioni degeneri) dal motore di calcolo puro, rendendo il sistema modulare: se venisse aggiunta una nuova configurazione topologica, sarebbe sufficiente definire un nuovo Handler senza dover modificare il nucleo degli algoritmi di tracciamento.

Gruppo G — Sottosistemi per Comando [147 funzioni]

Il Gruppo G costituisce l'infrastruttura di coordinamento della libreria: ogni raggruppamento è dedicato a uno specifico comando AutoCAD (c:...),

isolando le routine che trasformano l'input utente in un flusso di lavoro geometrico. La logica di questo gruppo è la compartimentazione per competenza: ogni sottosistema gestisce internamente le proprie interfacce (prompt, variabili di stato, loop interattivi) delegando al Gruppo D (Geometria) il calcolo puro e al Gruppo E (Disegno) l'output grafico. Questa organizzazione permette di mantenere i comandi indipendenti, facilitando la gestione del codice e l'aggiunta di nuove funzionalità senza interferire con le routine già consolidate.

- **G1 — Sottosistema CON1P [6 funzioni]:** Gestisce la costruzione di coniche a partire da 2 punti fissi (vertici o fuochi) e 1 punto variabile. Il modulo è raggruppato qui perché condivide le logiche di calcolo dei vettori direttori degli assi e la gestione specifica dello stato dell'interfaccia utente (prompt dinamico).
- **G2 — Sottosistema CON4P [45 funzioni]:** Il motore più vasto della libreria. Include l'intera infrastruttura per l'ambiente protetto: motori di scansione del luogo geometrico, gestione del sistema "Matrioska" per le coniche annidate, analisi dei luoghi ortogonali e il rendering interattivo che copre ogni configurazione geometrica del quadrangolo base.
- **G3 — Sottosistema PAR2PFAPI [17 funzioni]:** Focalizzato sulle parabole con asse vincolato (proprio, improprio o libero). La logica di questo gruppo è incentrata sull'analisi del luogo geometrico tramite raffinamento frattale e sulla gestione della Dima topologica che definisce i limiti operativi della soluzione.
- **G4 — Sottosistemi ELL3P e IPE3P [32 funzioni]:** Gestisce ellissi e iperboli equilatere a 3 punti con raffinamento algebrico. La struttura è speculare per entrambi i comandi: acquisizione vincoli, estrazione parametri, calcolo dei candidati, valutazione e risoluzione algebrica dei nodi esatti per la massima precisione.
- **G5 — Sottosistema PAR3P [15 funzioni]:** Coordina i comandi per la parabola passante per 3 punti (sia in ambiente statico che reattivo). La logica è condivisa tra le due varianti per garantire coerenza nella gestione delle guide visive e nel calcolo del vertice e del semi-parametro.
- **G6 — Infrastruttura Comandi e Algoritmi Pascal [30 funzioni]:** Il modulo trasversale. Contiene le routine di sistema (Init/Restore) per la gestione delle variabili AutoCAD, il motore di algebra lineare (Cramer 3x3 e 4x4) e l'implementazione del teorema di Pascal per la determinazione del 6° punto, fornendo le basi di calcolo utilizzate indistintamente da tutti gli altri comandi.

Gruppo H — Utilità e Servizi [62 funzioni]

Questo gruppo agisce come libreria di supporto trasversale, fornendo gli strumenti necessari per la manipolazione dei dati in ogni fase del processo, dall'acquisizione dell'input utente alla gestione dell'output grafico. La logica di questo raggruppamento è la centralizzazione delle operazioni ripetitive di basso livello, garantendo che i motori geometrici (Gruppo D) e i gestori dei comandi (Gruppo G) possano operare senza dover implementare autonomamente routine di base.

- **H1 — Utilità Geometrico-Matematiche [29 funzioni]:** Questo modulo è dedicato alla trasformazione e normalizzazione dei dati numerici. Include le routine per traslazioni, rotazioni (dirette e inverse tramite cs), omotetie e calcoli di distanze medie. La sua funzione è standardizzare i dati geometrici in entrata e uscita dai motori di calcolo, rendendo le operazioni di manipolazione spaziale indipendenti dalla logica specifica del comando.
- **H2 — Servizi AutoCAD [19 funzioni]:** Costituisce l'interfaccia operativa con il database di AutoCAD. Contiene le funzioni per la gestione dei layer, la manipolazione dei SelectionSet, la creazione di gruppi anonimi e la formattazione dei dati per l'invio al motore VLA. È raggruppato qui per isolare le chiamate al database grafico del CAD, mantenendo il codice di calcolo "puro" e non contaminato dalle dipendenze specifiche dell'ambiente grafico.

- **H3 — Analisi e Manipolazione di Liste [14 funzioni]:** Il modulo di elaborazione dei dati strutturati. Include le pipeline di processing per la gestione dei rami dei luoghi geometrici (taglio, cucitura delle estremità, ordinamento) e le routine per l'indicizzazione avanzata delle liste (es. estrazione di coppie, rotazioni cicliche). La loro funzione è gestire il flusso dei dati durante le scansioni ricorsive, permettendo al sistema di mantenere l'integrità strutturale delle liste di punti durante le trasformazioni topologiche.

I — Comandi AutoCAD (c:…) [15 funzioni]

Questo raggruppamento costituisce l'interfaccia pubblica (entry-point) della libreria: le 15 funzioni qui raccolte sono i comandi che l'utente richiama direttamente dalla riga di comando di AutoCAD. La loro logica non risiede nel calcolo geometrico — che viene delegato interamente alle categorie A, B, C e D — ma nella gestione del ciclo di vita dell'operazione. Ogni comando Master all'interno di questo modulo è progettato per:

- Orchestrare il flusso di lavoro: Gestisce l'intero dialogo con l'utente, dalla richiesta di selezione dei punti (ancore) alla restituzione del risultato grafico.
- Coordinare le dipendenze: Inizializza l'ambiente tramite le routine di sistema (c:slc per i layer) e richiama in sequenza i motori di calcolo (es. D4 per la conica 4P o G5 per la parabola 3P) e le primitive di disegno (Gruppo E).
- Gestire l'interattività: Supporta le due modalità operative principali dell'applicativo: la modalità statica (calcolo puntuale) e la modalità reattiva (es. c:con5pRea, c:par3PRea), in cui il comando attiva reattori e loop che permettono il ricalcolo e il ridisegno istantaneo della conica ad ogni spostamento dei punti da parte dell'utente.
- Sicurezza: Integra i richiami all'Oracolo Geometrico per prevenire stalli matematici e gestire le eccezioni prima che il controllo venga restituito all'utente.

In sintesi, questo gruppo rappresenta il livello di astrazione superiore: nasconde la complessità algebrica sottostante, presentando all'utente un'interfaccia coerente, guidata da prompt adattivi e integrata con le funzionalità dinamiche di AutoCAD.

A — Matematica di Base e Algebra [65 funzioni]

Funzione	Scopo
Math_Classify_Forma_4Pu	Classificare la topologia geometrica formata da 4 punti spaziali, categorizzandola in base alle sue proprietà (degenerazioni, concavità, convessità, simmetrie) e riordinando i vertici di conseguenza.
Math_Circonferenza_3Pu	Calcolare analiticamente il centro e il raggio della circonferenza passante per tre punti dati (circocentro).
Math_Dist_2Pu	Calcolare in sicurezza la distanza reale tra due punti, prevenendo crash di esecuzione causati da argomenti nulli o malformati.
Math_PuntoMedio_2Pu	Calcolare le coordinate spaziali del punto medio esatto tra due estremi.
Math_lisPuntoMedio_lis2Pu	Calcolatore massivo (Array Processor) per estrarre i punti medi da una matrice di coppie.
Math_LisPuntiMedi_Quadrangolo_4Pu	Calcolare ed estrarre in un'unica operazione tutti i 6 punti medi di un quadrangolo completo (4 lati + 2 diagonali).
Math_Inters_Quaterna	Calcolare il punto di intersezione spaziale (fuga o incrocio diagonale) tra specifiche coppie di segmenti definiti dai quattro vertici di un quadrangolo.
Math_Inters_lisQuaterne	Eseguire il calcolo massivo delle intersezioni su un array di quaterne di punti, applicando a tutte la medesima regola topologica (es. tutti i lati opposti, oppure tutte le diagonali).
Math_Proiezione_1Pu_su_2Pu	Calcolare la proiezione ortogonale di un punto spaziale su una retta definita da due punti noti.
Math_Proiezione_lisPu_su_2Pu	Proiettare massivamente un array di punti (nuvola) su un'unica retta fissa di riferimento.
Math_Proiezione_lisTerne	Eseguire proiezioni ortogonali indipendenti per un array di terne, dove sia il punto che l'asse variano.
Math_Ortocentro_3Pu	Determinare la coordinata spaziale dell'ortocentro (intersezione delle altezze) di un triangolo.
Math_TriangoloDiagonale_4Pu	Determinare i 3 vertici del triangolo diagonale generato dalle intersezioni dei lati opposti e delle diagonali di un quadrangolo completo.
Math_ConicaCentri_9Pu	Generare la nuvola dei 9 punti notevoli (6 medi + 3 diagonali) per cui passa l'ellisse/iperbole dei centri del quadrangolo.
Math_Prod_Sum	Calcola il prodotto della somma di due coppie di valori: $(a + b) \times (c + d)$.
Math_Prod_Diff	Calcola il prodotto delle differenze: $(a - b) \times (c - d)$.
Math_Sum_Prod	Calcola la somma di due prodotti: $(a \times b) + (c \times d)$.
Math_Diff_Prod	Calcola la differenza di due prodotti: $(a \times b) - (c \times d)$.
Math_Sum_Squares	Calcola la somma dei quadrati di due valori numerici ($a^2 + b^2$).
Math_Diff_Squares	Calcola la differenza dei quadrati di due valori numerici ($a^2 - b^2$).

Math_Sqr	Calcola il quadrato di un valore numerico (x^2).
Math_Cyclic_Det	Calcola il determinante ciclico di sei valori, usato per il doppio dell'area orientata di un triangolo: x1(y2−y3) + x2(y3−y1) + x3(y1−y2).
Math_Inv_Sqrt	Calcola il reciproco della radice quadrata di un valore: 1/√x, formalmente equivalente all'elevamento a potenza −0.5.
Pitag_Ip	Calcola l'ipotenusa di un triangolo rettangolo date le due cateti, applicando il teorema di Pitagora: ip = √(ca1² + ca2²).
Pitag_Ca	Calcola la lunghezza di un cateto di un triangolo rettangolo data l'ipotenusa e l'altro cateto: ca = √(ip² − ca_noto²).
Math_Acos	Fornire la funzione matematica arcocoseno (acos), nativamente assente in AutoLISP, ricavandone il valore in radianti a partire dalla funzione arcotangente.
Math_Solve_5x5	Risolvere un sistema di equazioni lineari 5x5 utilizzando il metodo di eliminazione di Gauss con pivoting parziale.
Math_Extract_Coord_Pow	Estrae da una lista di punti le coordinate di un certo asse (X o Y) elevate a una potenza specificata, restituendole come lista di scalari.
Math_MinMax_List	Ricavare i valori estremi (minimo e/o massimo) all'interno di un array di numeri, fungendo da motore statistico di base per il rilevamento di bounding box e limiti dimensionali.
Math_Delta_List	Calcolare l'escursione totale assoluta (il Delta) tra il limite superiore e il limite inferiore presenti all'interno di un set di dati numerici.
Math_Solve_Oblique_2P	Risolve il sistema di equazioni che fornisce i coefficienti della parabola con asse inclinato di un angolo dato, passante per esattamente 2 punti fissi e con vertice lungo l'asse.
Math_Solve_Par_3P	Determina i coefficienti della parabola passante esattamente per 3 punti, con asse di simmetria orientato secondo un angolo parametrico dato.
Math_Calc_Par_Global_Error	Calcola l'errore quadratico complessivo della parabola con asse inclinato di Ang rispetto a tutti i punti del set, sommando i quadrati degli scarti residui.
Math_Det3x3	Calcolare il determinante di una matrice 3x3 ricevendo i nove coefficienti come argomenti scalari individuali, ottimizzando le prestazioni per chiamate dirette senza overhead di liste.
Math_Lagrange_Term	Calcola il termine di base di Lagrange L_i(x) usato nell'interpolazione polinomiale lagrangiana: il peso dell'i-esimo nodo rispetto agli altri due nodi xj e xk.
Calcola_Lagrange	Interpola il valore y in un punto x usando il polinomio di Lagrange di 2° grado costruito su 3 nodi (x1,y1), (x2,y2), (x3,y3).
Math_Rot_Pt_2D_CS	Ruota un punto 2D dato il coseno e il seno dell'angolo (forma "co-si"), restituendo le coordinate ruotate come lista 3D.
Calcola_Parabola	Calcola il valore y della parabola nel punto x usando l'interpolazione di Lagrange sui 3 punti di vincolo, restituendo le coordinate del punto sulla curva.

Math_Calc_Par_Error_4th_Pt	Valuta di quanto la parabola con asse inclinato di Ang, passante per i primi 3 punti, si discosta dal 4° punto: funzione di errore usata nella scansione parametrica.
Math_Line_General_Eq	Calcola i coefficienti (a,b,c) dell'equazione generale di una retta ax+by+c=0 passante per due punti dati.
Math_Multiply_Lines_To_Conic	Costruisce la conica degenere (coppia di rette) data dal prodotto algebrico di due rette in forma generale, esprimendo il risultato come conica $Ax^2+Bxy+Cy^2+Dx+Ey+F=0$.
Math_Combine_Conics	Combina linearmente due coniche K1 e K2 nella forma del fascio $K1+\lambda \cdot K2$, restituendo i coefficienti della conica combinata.
Math_Extract_Equilateral_Data	Estrae i dati geometrici (centro, direzione dell'asse, semi-asse) da una conica equilatera (iperbole rettangola, A+C=0) definita dai suoi 6 coefficienti.
Math_Get_Coeffs_4P_Axis Dir	Trova i coefficienti della conica passante per 4 punti con l'asse principale orientato nella direzione specificata da PuD1-PuD2, usando il fascio di coniche.
Math_Dist_Point_Line	Calcola la distanza euclidea di un punto Pu dalla retta definita dai coefficienti (a,b,c) dell'equazione generale ax+by+c=0.
Math_Solve_Par_Coeffs	Risolve il sistema lineare per determinare i coefficienti (A,B,C) della conica generica passante per 3 punti con centro fisso dato, nella forma $ax\_i^2+bx\_iy\_i+cy\_i^2=1$ (sistema centrato).
Math_Solve_Centered_Conic_3P_Ratio	Calcolare il centro locale e il semiasse maggiore di una conica a centro (ellisse o iperbole) passante per 3 punti, noto il rapporto assi.
Math_Solve_Ell_3P_Ratio	Fornire un'interfaccia d'uso rapida e tipizzata per la risoluzione specifica di ellissi (apertura fissa) e iperboli (inclinazione asintoti).
Math_Solve_Ipe_3P_Ratio	Fornire un'interfaccia d'uso rapida e tipizzata per la risoluzione specifica di ellissi (apertura fissa) e iperboli (inclinazione asintoti).
Math_Rotate_Pt_Origin	Ruota un punto 2D attorno all'origine degli assi di un angolo dato, restituendo le nuove coordinate.
Math_Uguaglianza_Relativa_2Re	Verifica se due valori reali sono uguali entro una tolleranza relativa proporzionale al loro ordine di grandezza, evitando i problemi di confronto in virgola mobile.
Math_Area_Triangolo	Calcola l'area del triangolo definito dai tre punti P1, P2, P3 usando la formula del determinante.
Math_Angolo_Minore_CCW	Calcola l'angolo minore compreso tra due angoli orientati in senso antiorario (CCW), restituendo il valore compreso tra 0 e π.
Math_Aree_Pu_vs_Tri	Calcola e confronta le aree dei sotto-triangoli formati da un punto P con i tre vertici del triangolo P1-P2-P3, per verificare la posizione relativa di P rispetto al triangolo.
Math_Check_Conciclicita_4Pu	Verifica se quattro punti giacciono su una stessa circonferenza (sono concicli), calcolando la circonferenza passante per tre di essi e verificando se il quarto la soddisfa.
Math_Conta_Paral_4Pu	Conta quante coppie di lati opposti sono parallele nel quadrangolo definito da 4 punti (0, 1 o 2 coppie).
Math_Conta_Ortogo_4Pu	Conta quante coppie di lati adiacenti sono ortogonali nel quadrangolo definito da 4 punti (0, 1, 2, 3 o 4 angoli retti).

Math_Conta_Coinc_4Pu	Conta quante coppie di punti coincidono nel set di 4 punti dato (0, 1, 2, 3 o 6 coppie).
F_func	Calcolare analiticamente i parametri esatti dell'unica parabola passante per 3 punti, forzando il primo come Vertice.
Math_Extract_Geom_Data_From_Coeffs	Estrae i dati geometrici di una conica (tipo, centro, semiassi, fuochi, eccentricità, orientamento) a partire dai suoi 6 coefficienti algebrici.
Math_ConIpeEqu_Coeffs	Calcolare i coefficienti dell'equazione generale (A, B, C, D, E, F) di un'iperbole equilatera passante per 3 punti.
Math_Solve_Cubic_Real	Calcolare analiticamente le radici reali di un'equazione polinomiale di terzo grado (A*x^3 + B*x^2 + C*x + D = 0).
Math_Classify_Forma_Lis4Pu	Funzione adattatore (Wrapper) per interfacciare liste singole con il classificatore matematico a parametri separati.
Math_Circonferenza_Eulero_4Pu	Calcolare la Circonferenza di Eulero (dei 9 punti) di un sistema ortocentrico, mantenendo la ridondanza analitica di calcolo e media.
Math_Solve_Quadratic	Risoluzione analitica di un'equazione polinomiale di secondo grado ($ax^2 + bx + c = 0$).

B — Algebra Vettoriale e Simmetrie [12 funzioni]

Funzione	**Scopo**
Geom_2Pu_Simm_PuC	Genera una coppia di punti simmetrici rispetto a un centro, disposti equidistanti lungo una direzione data.
Geom_1Pu_Simm_PuC	Calcola il punto simmetrico di un punto rispetto a un perno (punto di riflessione centrale).
Vet_Sum	Eseguire le operazioni aritmetiche vettoriali di base (somma e differenza) componente per componente tra due vettori o punti spaziali (2D o 3D).
Vet_Sub	Eseguire le operazioni aritmetiche vettoriali di base (somma e differenza) componente per componente tra due vettori o punti spaziali (2D o 3D).
Vet_Sca	Eseguire il prodotto vettoriale di uno spazio (o punto) per uno scalare, modulandone l'intensità (magnitudo) o invertendone il verso (se il fattore numerico à negativo) senza alterarne la linea d'azione.
Vet_Len	Calcolare la magnitudine (lunghezza o modulo) assoluta di un vettore nello spazio geometrico.
Vet_Mid	Calcolare le coordinate spaziali esatte del punto medio intermedio tra due coordinate (o vettori) fornite in ingresso.
Vet_Uni	Normalizza un vettore 2D o 3D al modulo unitario (versore), dividendo ogni componente per il modulo del vettore.
Vet_Dot	Calcolare il prodotto scalare (dot product) tra due vettori, restituendo un singolo valore numerico.
Vet_Cro	Calcolare il prodotto vettoriale (cross product) tra due vettori nello spazio tridimensionale, restituendo un terzo vettore strettamente ortogonale al piano da essi formato.

Vet_Uni_Bisect	Determinare la direzione esatta (versore) che taglia esattamente a metà l'angolo compreso tra due vettori generici, fondamentale per algoritmi di offset, centri di raccordo e normali ai vertici.
Vet_Perp_Sca	Calcolare il vettore perpendicolare (normale) nel piano XY rispetto a un vettore dato e scalarlo simultaneamente di un fattore definito.

C — Predicati Geometrici (P_…) [57 funzioni]

Funzione	Scopo
P_center-infinite	Determina se una conica ha il centro all'infinito, ovvero se è una parabola, verificando che il determinante della matrice 2×2 dei coefficienti quadratici sia nullo.
probe-conic-algebraic-tipo	Classifica algebricamente il tipo di conica (ellisse, iperbole o parabola) a partire dai suoi 6 coefficienti, calcolando il discriminante della forma quadratica.
inters_ver	Calcola il punto di intersezione tra due segmenti (o rette) verificando anche che l'intersezione cada nei tratti finiti dei segmenti.
inters_check_tristate	Verifica l'intersezione tra due segmenti restituendo uno stato a tre valori: intersezione nel tratto, intersezione sull'estensione, o nessuna intersezione.
P_Are_Vectors_Parallel	Verifica se due vettori 3D sono paralleli tra loro calcolando il modulo del loro prodotto vettoriale e confrontandolo con il prodotto dei moduli.
P_Is_Config_Concave	Verifica se 4 punti formano una configurazione concava (quadrangolo non convesso), cioè se uno dei punti è interno al triangolo formato dagli altri tre.
P_Are_Collinear	Verifica se 3 punti sono collineari (allineati sulla stessa retta). Alias di P_Allin_3Pu con nome inglese per retrocompatibilità.
P_Has_Collinear_Triplet	Verifica se in un set di punti esiste almeno una tripletta collineare (allineata).
P_Is_Orthocentric_System	Verifica se 4 punti formano un sistema ortocentrico: ogni punto è l'ortocentro del triangolo formato dagli altri tre.
P_Are_Vectors_Parallel_2D	Verifica il parallelismo di due vettori 2D confrontando le pendenze tramite prodotto incrociato (cross product planare).
P_Coinc_2Pu	Verifica se due punti coincidono, cioè se la loro distanza euclidea è inferiore alla tolleranza geometrica di coincidenza.
P_Coinc_List_2Pu	Verifica la coincidenza tra due punti quando almeno uno di essi può essere fornito come lista (eventualmente nidificata), gestendo la destrutturazione automatica.
P_Coinc_3Pu	Verifica se almeno due dei tre punti dati coincidono.
P_Coinc_4Pu	Verifica se almeno due dei quattro punti dati coincidono.
P_Coinc_2su3Pu	Verifica se esattamente due (e non tutti e tre) i punti dati coincidono.

P_Allin_3Pu	Verifica se tre punti sono allineati calcolando l'area del triangolo che formano; se l'area è nulla o trascurabile i punti sono collineari.
P_Allin_List_3Pu	Verifica l'allineamento di 3 punti ricevuti come lista unica anziché come argomenti separati.
P_Allin_4Pu	Verifica se tutti e quattro i punti dati sono allineati sulla stessa retta.
P_Allin_3su4Pu	Verifica se almeno 3 dei 4 punti dati sono allineati (configurazione degenere).
P_Pu_Inter_3Pu	Verifica se un punto P si trova all'interno (o sul bordo) del triangolo P1-P2-P3.
P_Pu_Ester_3Pu	Verifica se un punto P si trova all'esterno del triangolo P1-P2-P3.
P_Pu_Inter_List_4Pu	Verifica se un punto è interno al quadrangolo definito da una lista di 4 punti, testando l'appartenenza al triangolo formato da ogni tripla di vertici.
P_Pu_Ester_List_4Pu	Verifica se un punto è esterno al quadrangolo definito da una lista di 4 punti.
P_Conca_4Pu	Verifica se il quadrangolo formato da 4 punti è concavo, cioè se uno dei quattro punti è interno al triangolo formato dagli altri tre.
P_Conve_4Pu	Verifica se il quadrangolo formato da 4 punti è convesso, cioè se nessun punto è interno al triangolo degli altri tre.
P_0Paral_4Pu	Verifica che nessuna coppia di lati opposti del quadrangolo con 4 punti sia parallela.
P_1Paral_4Pu	Verifica che esattamente una coppia di lati opposti del quadrangolo sia parallela (trapezio).
P_2Paral_4Pu	Verifica che entrambe le coppie di lati opposti del quadrangolo siano parallele (parallelogramma o derivati).
P_0Orto_4Pu	Verifica che nessun angolo interno del quadrangolo sia retto.
P_1Orto_4Pu	Verifica che esattamente un angolo interno del quadrangolo sia retto.
P_2Orto_4Pu	Verifica che esattamente due angoli interni del quadrangolo siano retti.
P_3Orto_4Pu	Verifica che almeno 3 angoli interni del quadrangolo siano retti (equivalente a 4, poiché la somma degli angoli è sempre 360°).
P_0Coinc_4Pu	Verifica che nessuna coppia di punti tra i quattro dati sia coincidente.
P_1Coinc_4Pu	Verifica che esattamente una coppia di punti tra i quattro dati coincida.
P_2Coinc_4Pu	Verifica che esattamente due coppie di punti tra i quattro dati coincidano.
P_3Coinc_4Pu	Verifica che almeno 3 punti tra i quattro dati coincidano.
P_6Coinc_4Pu	Verifica l'improbabile caso in cui tutti e 4 i punti coincidano (6 coppie totali tutte coincidenti = degenere totale).
P_Paral_2Re	Verifica se due rette sono parallele tra loro, confrontando le loro direzioni.
P_Paral_distinte_2Re	Verifica se due rette sono parallele E distinte (non coincidenti).

P_Perpe_2Re	Verifica se due rette sono perpendicolari tra loro.
P_Paral_List_2Re	Verifica il parallelismo tra due rette definite come liste di punti.
P_Paral_distinte_List_2Re	Verifica se due rette (definite come liste di punti) sono parallele e distinte.
P_Perpe_List_2Re	Verifica la perpendicolarità tra due rette definite come liste di punti.
P_Paral_4Pu	Verifica se i lati opposti del quadrangolo formato da 4 punti sono paralleli (almeno una coppia), accettando 4 argomenti separati.
P_Paral_distinti_4Pu	Verifica se i lati opposti del quadrangolo sono paralleli E distinti (cioè il quadrangolo è un parallelogramma o derivato).
P_Quadrangolo_4Pu	Verifica che i 4 punti formino un quadrangolo valido: nessuna coppia coincidente, nessuna tripletta allineata, forma non degenere.
P_Trapezio_4Pu	Verifica se il quadrangolo formato da 4 punti è un trapezio: esattamente una coppia di lati opposti è parallela.
P_Parallelogramma_4Pu	Verifica se il quadrangolo è un parallelogramma: entrambe le coppie di lati opposti sono parallele, ma nessun angolo interno è retto.
P_Quadrangolo_Generico_4Pu	Verifica se il quadrangolo è generico: nessun lato parallelo, nessun angolo retto, nessuna degenerazione (la configurazione più generica possibile).
P_Trapezio_Scaleno_4Pu	Verifica se il quadrangolo è un trapezio scaleno: una sola coppia di lati opposti paralleli e i lati obliqui non uguali tra loro.
P_Trapezio_Isoscele_4Pu	Verifica se il quadrangolo è un trapezio isoscele: una sola coppia di lati opposti paralleli e i due lati obliqui di uguale lunghezza.
P_Rettangolo_4Pu	Verifica se il quadrangolo è un rettangolo: due coppie di lati paralleli e tutti e 4 gli angoli interni retti.
P_Rombo_4Pu	Verifica se il quadrangolo è un rombo: due coppie di lati paralleli, tutti i lati di uguale lunghezza, ma angoli non necessariamente retti.
P_Quadrato_4Pu	Verifica se il quadrangolo è un quadrato: rombo con tutti gli angoli retti.
P_Conciclici_4Pu	Verifica se i 4 punti giacciono tutti su una stessa circonferenza (concicli).
P_Concavo_Ortocentrico_4Pu	Verifica se il quadrangolo concavo è ortocentrico: il punto interno è l'ortocentro del triangolo formato dagli altri tre vertici.
P_Concavo_Generico_4Pu	Verifica se il quadrangolo formato da 4 punti è concavo ma non ortocentrico.

D — Geometria delle Coniche

D1 — Topologia e Ordinamento Spaziale [7 funzioni]

Funzione	Scopo
Geom_Analyze_Ipe_Topology	Analizza la configurazione topologica di 4 punti per determinare quante e quali iperboli equilatere li soddisfano: nessuna (degenere), infinite (ortocentrico) o una unica.
Geom_Check_Trapezoid_Direction	Verifica se i lati paralleli del trapezio definito da 4 punti sono orientati nella direzione attesa (primo o secondo paio di lati opposti).
Geom_Sort_Points_CCW	Ordina un insieme di punti in senso antiorario (Counter-Clock-Wise) rispetto al loro baricentro, per garantire la coerenza topologica negli algoritmi.
Geom_Ordina_CCW_4Pu	Riordina specificamente 4 punti in senso antiorario (CCW) rispetto al loro baricentro, garantendo la coerenza topologica per il quadrangolo convesso.
Geom_Ordina_Concavo_4Pu	Riordina 4 punti in configurazione concava mettendo il punto interno come ultimo elemento, per garantire la coerenza topologica nelle costruzioni concave.
Geom_Format_Map_Data	Strutturare una lista di punti geometrici grezzi trasformandola in pacchetti dati completi per il motore di aggancio LUSNAP e le meta-mappe.
Geom_Eval_Matrix_Parallelism	Valutare la condizione analitica di parallelismo tra il vettore normale a una conica in un punto P e il vettore che congiunge il centro della conica a P stesso.

D2 — Calcolo Parametrico delle Coniche [48 funzioni]

Funzione	Scopo
replace-nth	Ricostruire e restituire una nuova lista in cui l'elemento corrispondente all'indice specificato à stato sostituito dinamicamente con un nuovo valore, lasciando inalterati gli altri elementi.
get-all-permutations	Calcolare in modo esaustivo tutte le permutazioni possibili (combinazioni che tengono conto dell'ordine) degli elementi contenuti in una lista.
solve-system-cramer-2x2	Risolvere analiticamente un sistema lineare composto da due equazioni in due incognite (2x2) applicando il metodo algebrico di Cramer.
get-oblique-coords	Calcolare le coordinate (proiezioni scalari) di un punto spaziale rispetto a un sistema di riferimento obliquo, definito da un'origine e da due assi direzionali non necessariamente ortogonali.
get-unit-vector-from-pts	Determinare la direzione vettoriale pura (versore unitario) che congiunge spazialmente due punti, normalizzandone la lunghezza matematica a 1.0.
Lay_Make_Col	Creare silenziosamente un layer o forzarne l'aggiornamento cromatico, operando nel database senza alterare il layer corrente.
Get_N_Pts_Check	Gestire l'acquisizione sequenziale e interattiva di un numero esatto (N) di punti da parte dell'utente, fornendo un feedback visivo immediato sul CAD.
Get_1Pts_Check	Fornire interfacce rapide (scorciatoie sintattiche) per richiamare il motore principale di acquisizione punti con quantità numeriche prefissate (da 1 a 5).

Get_2Pts_Check	Fornire interfacce rapide (scorciatoie sintattiche) per richiamare il motore principale di acquisizione punti con quantità numeriche prefissate (da 1 a 5).
Get_3Pts_Check	Fornire interfacce rapide (scorciatoie sintattiche) per richiamare il motore principale di acquisizione punti con quantità numeriche prefissate (da 1 a 5).
Get_4Pts_Check	Fornire interfacce rapide (scorciatoie sintattiche) per richiamare il motore principale di acquisizione punti con quantità numeriche prefissate (da 1 a 5).
Get_5Pts_Check	Fornire interfacce rapide (scorciatoie sintattiche) per richiamare il motore principale di acquisizione punti con quantità numeriche prefissate (da 1 a 5).
Pu_Baricentro2D_LisPu	Calcolare il baricentro geometrico di un insieme di punti, determinando la media algebrica delle sole coordinate X e Y.
Geom_Extract_Par_World	Converte i parametri della parabola dal sistema locale dell'asse al sistema globale WCS di AutoCAD, applicando la trasformazione inversa.
Get_Pt_On_Axes	Trasformare coordinate locali (2D) in punti spaziali assoluti (3D) rispetto a un sistema di riferimento planare definito da un'origine e da due vettori direzionali (assi X e Y). Gestisce sia punti singoli che liste di coordinate.
Get_lisPt_On_Axes	Trasformare coordinate locali (2D) in punti spaziali assoluti (3D) rispetto a un sistema di riferimento planare definito da un'origine e da due vettori direzionali (assi X e Y). Gestisce sia punti singoli che liste di coordinate.
Calc_Par_Branch_CVs	Calcolare le coordinate spaziali WCS dei punti chiave necessari per la costruzione geometrica e la rappresentazione visiva di una parabola (punti di controllo della curva Spline, estremi della direttrice e limiti dell'asse di simmetria).
Calc_Par_Directrix_Pts	Calcolare le coordinate spaziali WCS dei punti chiave necessari per la costruzione geometrica e la rappresentazione visiva di una parabola (punti di controllo della curva Spline, estremi della direttrice e limiti dell'asse di simmetria).
Calc_Par_Axis_Pts	Calcolare le coordinate spaziali WCS dei punti chiave necessari per la costruzione geometrica e la rappresentazione visiva di una parabola (punti di controllo della curva Spline, estremi della direttrice e limiti dell'asse di simmetria).
Calc_Ipe_Branch_CVs	Determinare analiticamente le coordinate WCS dei punti strutturali per il tracciamento esatto di un'iperbole (punti di controllo dei rami NURBS/Spline, vertici limite per gli asintoti incrociati e confini per gli assi di simmetria maggio…
Calc_Ipe_Asym_Pts	Determinare analiticamente le coordinate WCS dei punti strutturali per il tracciamento esatto di un'iperbole (punti di controllo dei rami NURBS/Spline, vertici limite per gli asintoti incrociati e confini per gli assi di simmetria maggio…
Calc_Ipe_Axes_Pts	Determinare analiticamente le coordinate WCS dei punti strutturali per il tracciamento esatto di un'iperbole (punti di controllo dei rami NURBS/Spline, vertici limite per gli asintoti incrociati e confini per gli assi di simmetria maggio…
Geom_Helper_Find_Par_Axis_Angle	Eseguire la ricerca dicotomica per individuare l'inclinazione dell'asse di simmetria della parabola passante per il vertice e due punti.
Geom_Calc_Par_V_2P	Calcolare analiticamente il Fuoco e l'asse di simmetria di una parabola spaziale conoscendo le coordinate del Vertice e di 2 punti.

Load_Conic_Layers_Fast	Verificare la presenza nel disegno dei layer standard preconfigurati per la rappresentazione delle coniche e, se già esistenti, proporne all'utente il riutilizzo rapido per velocizzare il setup.
Config_Conic_Layers	Coordinare l'inizializzazione e la configurazione di tutti i layer operativi necessari all'ambiente di disegno delle coniche, tentando prima un caricamento rapido e passando, se necessario, alla definizione manuale interattiva.
Princ_1Pun	Fornire una suite di funzioni di log e stampa formattata per inviare output strutturati alla console di AutoCAD. Combinano etichette testuali personalizzate con la rappresentazione in stringa di vari tipi di dati (Punti, Simboli, Numeri,…
Princ_2Pun	Fornire una suite di funzioni di log e stampa formattata per inviare output strutturati alla console di AutoCAD. Combinano etichette testuali personalizzate con la rappresentazione in stringa di vari tipi di dati (Punti, Simboli, Numeri,…
Princ_1Sym	Fornire una suite di funzioni di log e stampa formattata per inviare output strutturati alla console di AutoCAD. Combinano etichette testuali personalizzate con la rappresentazione in stringa di vari tipi di dati (Punti, Simboli, Numeri,…
Princ_1Num	Fornire una suite di funzioni di log e stampa formattata per inviare output strutturati alla console di AutoCAD. Combinano etichette testuali personalizzate con la rappresentazione in stringa di vari tipi di dati (Punti, Simboli, Numeri,…
Princ_2Num	Fornire una suite di funzioni di log e stampa formattata per inviare output strutturati alla console di AutoCAD. Combinano etichette testuali personalizzate con la rappresentazione in stringa di vari tipi di dati (Punti, Simboli, Numeri,…
Princ_1Int	Fornire una suite di funzioni di log e stampa formattata per inviare output strutturati alla console di AutoCAD. Combinano etichette testuali personalizzate con la rappresentazione in stringa di vari tipi di dati (Punti, Simboli, Numeri,…
Geom_Co_Si_Ang	Precompila la coppia (coseno, seno) per un angolo in radianti, da passare ai motori di rotazione per evitare ricalcoli ripetuti.
rotate-pt-origin	Ruota un punto 2D attorno all'origine degli assi. Funzione a denominazione kebab-case (stile Scheme) usata in contesti legacy.
rotate-pt-90	Ruota un punto 2D di esattamente 90° in senso antiorario attorno all'origine.
Trans_Pt_Glo_To_Loc	Trasforma un singolo punto dal sistema globale WCS al sistema locale definito da origine e angolo, applicando traslazione e rotazione.
Trans_Pt_Loc_To_Glo	Trasforma un singolo punto dal sistema locale al sistema globale WCS, invertendo la trasformazione di Trans_Pt_Glo_To_Loc.
Geom_Count_Parallel_Pairs	Contare il numero di coppie di lati paralleli in un quadrangolo di 4 punti, garantendo un ritorno numerico intero anche in caso di figure concave o degeneri.
Geom_Find_Parabolic_Angles	Trova gli angoli dell'asse parabola che soddisfano i vincoli di passaggio per i punti dati, eseguendo una scansione angolare e raccogliendo tutti gli zeri della funzione di errore.
Geom_Trans_Pt	Trasla un singolo punto di un vettore dato; versione scalar per singolo punto del concetto implementato in lista da Util_Translate_Pts_Global.

Geom_Get_Projected_Center	Calcola la proiezione del centro geometrico di una conica sul luogo geometrico più vicino, per associarlo alla conica corretta nei contesti multi-ipotesi.
Geom_Ipe_4P_Eq	Calcola i parametri geometrici dell'iperbole equilatera passante per 4 punti: centro, direzione dell'asse semi-principale, semi-asse.
Geom_MinMax_Dist_LisPu	Calcolare e restituire le distanze minime e massime (e i relativi punti) da un set di coordinate.
Geom_ConCen_4PC	Calcola i coefficienti della conica passante per 4 punti con centro fisso dato, usando una strategia a torneo 3-su-4 per massimizzare la robustezza numerica.
Geom_Conic_4Pf_Engine	Motore grafico per la vettorializzazione a schermo delle coniche associate a un Fuoco target.
Geom_Conic_4Pv_Engine	Motore grafico specializzato per la vettorializzazione a schermo delle coniche associate a un Vertice target.
Geom_Precalc_Node_Conics	Precalcolare analiticamente le equazioni di tutte le coniche del fascio che possiedono uno dei vertici del quadrangolo come punto di tangenza/asse principale.
Geom_Master_Analysis_4P	Eseguire l'analisi topologica completa della quaterna di base e calcolarne i centri notevoli primari.

D3 — Scansione Adattiva e Raffinamento Ricorsivo [9 funzioni]

Funzione	**Scopo**
Geom_Scan_3P_Parabolas_Adaptive	Scanner adattivo della famiglia di parabole passanti per 3 punti: scansiona l'intervallo angolare [0°, max_angle] con passo variabile, riducendo il passo nelle zone di alta variazione (vicino ai vertici) per ottenere una copertura comple…
Geom_Refine_Internal_Scan	Scansione di raffinamento su un sotto-intervallo angolare [a_start, a_end] con n_steps passi uniformi, calcolando per ogni angolo la distanza del vertice della parabola dal punto target, per isoprare il minimo.
Geom_Refine_Internal_Bracket	Identifica il "bracket" (forcella) attorno al minimo di distanza dentro una lista di record angolari, estraendone gli estremi per il raffinamento successivo.
Geom_Refine_Internal_Recursive	Raffinamento ricorsivo ad albero della parabola ottimale: restringe progressivamente l'intervallo angolare attorno al minimo finché la larghezza spaziale del bracket è inferiore alla soglia o si raggiunge la profondità massima di ricorsi…
Geom_Refine_Par_Topological	Raffinamento topologico della parabola per un punto target: identifica il ramo del luogo a 3 punti più vicino al target, risolve l'ambiguità tra i due rami opposti (ang e ang+π) e lancia il raffinamento ricorsivo nella direzione corretta.
Geom_Refine_Internal_Scan_Fuochi	Eseguire una scansione angolare per valutare la distanza tra il fuoco di una serie di parabole e un punto bersaglio.
Geom_Refine_Internal_Bracket_Fuochi	Analizzare i risultati della scansione per individuare il "bracket" (l'intervallo angolare) in cui si trova il fuoco con la distanza minima.
Geom_Refine_Internal_Recursive_Fuochi	Motore di ricerca binaria ricorsiva per stringere iterativamente l'angolo fino a convergere sulla coordinata focale esatta.
Geom_Refine_Par_Topological_Fuochi	Funzione Wrapper/Entry-point per innescare l'affinamento topologico incentrato sul fuoco parabolico.

D4 — Motori di Azione Conica [4 funzioni]

Funzione	Scopo
Geom_Conic_Center_Proprio_Action	Motore di azione per la conica con centro finito (ellisse o iperbole) passante per 4 punti, con centro da minimizzare: trova il centro ottimale e calcola i coefficienti della conica.
Geom_Conic_Action_Par_Direct	Motore di azione per la parabola passante per 4 punti con il metodo diretto: classifica la topologia del quadrangolo, sceglie la diagonale corretta come asse naturale e chiama il disegnatore appropriato.
Geom_Conic_Action_MinEccentricity	Motore di azione per trovare la conica a eccentricità minima passante per 4 punti: nel fascio di coniche K1+λK2, sceglie il parametro λ che minimizza il rapporto tra gli autovalori (B^2-4AC minimale → conica più vicina al cerchio).
Geom_Master_Draw_Framework	Smistatore grafico (Router Polimorfico) primario per l'infrastruttura.

E — Disegno di Entità Grafiche (Draw_) [43 funzioni]

Funzione	Scopo
Draw_Point_R	Generare fisicamente e istantaneamente nel database DWG le entità grafiche primitive di base (punto, linea, cerchio, ellisse) bypassando l'uso della riga di comando per massimizzare le prestazioni di tracciamento.
Draw_Line_2Pu_R	Generare fisicamente e istantaneamente nel database DWG le entità grafiche primitive di base (punto, linea, cerchio, ellisse) bypassando l'uso della riga di comando per massimizzare le prestazioni di tracciamento.
Draw_Circle_R	Generare fisicamente e istantaneamente nel database DWG le entità grafiche primitive di base (punto, linea, cerchio, ellisse) bypassando l'uso della riga di comando per massimizzare le prestazioni di tracciamento.
Draw_ConEll_R	Generare fisicamente e istantaneamente nel database DWG le entità grafiche primitive di base (punto, linea, cerchio, ellisse) bypassando l'uso della riga di comando per massimizzare le prestazioni di tracciamento.
Draw_Lis_Line_Lis_Coppie_Pu_R	Fungono da interfacce di tracciamento massivo (wrapper) per generare simultaneamente interi array di linee o di punti, delegando la creazione fisica alle rispettive primitive di base e restituendo l'elenco dei puntatori generati.
Draw_Lis_Point_R	Fungono da interfacce di tracciamento massivo (wrapper) per generare simultaneamente interi array di linee o di punti, delegando la creazione fisica alle rispettive primitive di base e restituendo l'elenco dei puntatori generati.
Draw_Triangolo_Lis3Pu	Generare l'impalcatura grafica di base di un triangolo, tracciandone le tre linee perimetrali a partire da una lista contenente i suoi tre vertici spaziali.
Draw_Triangolo_Ortico_Lis3Pu	Calcolare analiticamente e tracciare a video il triangolo ortico (il triangolo i cui vertici coincidono con i piedi delle altezze) derivato da un triangolo base.
Draw_Quadrangolo_Convesso_Lis4Pu	Generare l'impalcatura grafica di base per un quadrangolo convesso, tracciandone esclusivamente il perimetro chiuso a partire dai suoi quattro vertici spaziali.
Draw_Quadrangolo_Concavo_Lis4Pu	Generare l'impalcatura grafica per un quadrangolo concavo, tracciando simultaneamente il perimetro esterno (il triangolo base) e i segmenti a stella interni.

Draw_Spline_Rational_3CV	Generare matematicamente una conica esatta (arco di ellisse, parabola o iperbole) tracciandola come curva B-Spline Razionale Quadratica basata su tre vertici di controllo spaziali.
Solve_And_Draw_From_Coeffs	Risolve i parametri geometrici dai coefficienti della conica e disegna la conica in AutoCAD, incluse le linee degli assi e le posizioni dei fuochi.
Draw_Curve_Segment_Spline	Generare un'entità grafica curvilinea (Spline) o rettilinea (Linea) a partire da una lista di punti, adattando automaticamente il tipo di oggetto CAD creato in base alla densità dei dati di input forniti.
Draw_ConCen_3P	Risolvere il sistema matriciale 3x3 per coniche a centro (Ellisse/Iperbole), unificando il calcolo di autovalori e autovettori, per poi delegare il disegno ai motori grafici in base al flag di vestizione anatomica.
Draw_ConCen_3P_Compl	Interfacce di retrocompatibilità per il solutore matriciale 3x3. NOTE SUI PARAMETRI: Le variabili 'lay_crv' e 'lay_constr' nella versione Compl sono mantenute volontariamente nella firma come "dummy arguments" per evitare errori di chiam…
Draw_ConCen_3P_Light	Interfacce di retrocompatibilità per il solutore matriciale 3x3. NOTE SUI PARAMETRI: Le variabili 'lay_crv' e 'lay_constr' nella versione Compl sono mantenute volontariamente nella firma come "dummy arguments" per evitare errori di chiam…
Draw_ConEll_CeVe_CoVe	Calcolare i parametri geometrici completi (assi e vertici mancanti) di un'ellisse a partire dal centro, un vertice e un fuoco, delegandone poi il tracciamento grafico.
Draw_ConEll_CeAx_Foci	Generare fisicamente nel database di AutoCAD l'ellisse completa della sua impalcatura geometrica (centri, assi, fuochi), smistando gli oggetti sui rispettivi layer/colori.
Draw_ConEll_CeAx_Light	Versione "light" del disegnatore di ellissi: disegna SOLO l'ellisse senza assi, fuochi o punti costruttivi aggiuntivi, per uso nei loop reattivi dove la velocità di ridisegno è critica.
Draw_ConIpe_CeVeFu	Motore universale per il tracciamento dell'iperbole. Gestisce centralmente il calcolo dei parametri, l'estensione dei rami e smista la generazione grafica tra anatomia completa e sola curva (Light).
Draw_ConIpe_CeVeFu_Compl	Interfacce specializzate per il tracciamento dell'iperbole, differenziate per estensione rami e livello di vestizione.
Draw_ConIpe_CeVeFu_Compl_Free	Interfacce specializzate per il tracciamento dell'iperbole, differenziate per estensione rami e livello di vestizione.
Draw_ConIpe_CeVeFu_Light	Interfacce specializzate per il tracciamento dell'iperbole, differenziate per estensione rami e livello di vestizione.
Draw_ConIpe_CeVeFu_Light_Free	Interfacce specializzate per il tracciamento dell'iperbole, differenziate per estensione rami e livello di vestizione.
Draw_ConIpe_4P_Eq	Disegna l'iperbole equilatera (rettangola) passante per 4 punti dati, calcolando prima i coefficienti con Math_Extract_Equilateral_Data e poi disegnando la curva con assi e asintoti.
Draw_ConPar_VeFu	Centralizzare il calcolo geometrico e il tracciamento della parabola, ottimizzando la logica di calcolo e smistando l'output tra rappresentazione completa (vestizione) e nuda curva (luogo geometrico).
Draw_ConPar_VeFu_Compl	Fornire interfacce specializzate (wrapper) per il disegno della parabola, permettendo di scegliere tra una rappresentazione completa di anatomia e una puramente geometrica, mantenendo la retrocompatibilità con i comandi esistenti.

Draw_ConPar_VeFu_Light	Fornire interfacce specializzate (wrapper) per il disegno della parabola, permettendo di scegliere tra una rappresentazione completa di anatomia e una puramente geometrica, mantenendo la retrocompatibilità con i comandi esistenti.
Draw_ConPar_Data_3P	Tracciare una parabola a partire da dati geometrici pre-calcolati (terna Vertice-Fuoco-Angolo), bypassando ricalcoli analitici onerosi e delegando il disegno anatomico al motore core.
Draw_ConPar_5P_Sub	Risolvere e tracciare la parabola da 5 punti (Pascal), ottimizzando le prestazioni tramite chiamata diretta al motore grafico core.
Draw_ConPar_5P_Sub_Compl	Wrapper di retrocompatibilità per il disegno di parabole da 5 punti (Analisi Pascal).
Draw_ConPar_5P_Sub_Light	Wrapper di retrocompatibilità per il disegno di parabole da 5 punti (Analisi Pascal).
Draw_ConPar_4P_Light	Versione "light" del disegnatore di parabola a 4 punti: disegna solo la curva parabolica senza guide, assi o costruzioni aggiuntive, ottimizzata per i loop reattivi.
Draw_ConPar_4P_ConVer_R	Disegna la parabola passante per 4 punti con la tecnica costruttiva del vertice reciproco (R): sfrutta la diagonale del quadrangolo come asse naturale della parabola, include assi e vertice nel disegno.
Draw_Conic_4P_AxisDir	Disegna la conica passante per 4 punti con l'asse principale orientato nella direzione specificata da due punti, usando il metodo del fascio di coniche.
Draw_Con5P_Nor	Orchestratore unico per il tracciamento di coniche da 5 punti (Analisi Pascal), centralizzando la normalizzazione, la diagnosi del tipo di curva e il torneo combinatorio per la risoluzione matriciale.
Draw_Con5P_Nor_Compl	Fornire punti di ingresso specializzati per il disegno di coniche da 5 punti, mantenendo le firme originali per retrocompatibilità.
Draw_Con5P_Nor_Light	Fornire punti di ingresso specializzati per il disegno di coniche da 5 punti, mantenendo le firme originali per retrocompatibilità.
Geom_Draw_Trapezoid_Smart	Disegna un trapezio (o altra figura quadrangolare) connettendo 4 punti con segmenti nell'ordine topologico corretto.
Geom_Draw_Dima_Boundaries	Disegna i confini topologici (bordi del dominio) della sagoma Dima, scalandoli e ruotandoli per adattarli ai due punti di ancoraggio forniti dall'utente.
Draw_ConCen_4PC	Disegna la conica passante per 4 punti con centro fisso dato, coordinando il calcolo matematico e il rendering grafico.
Draw_Guide_Parabolas_4P	Disegna le parabole guida nell'ambiente a 4 punti: traccia le parabole passanti per le coppie di punti che delimitano il fascio di coniche nel quadrangolo dato.
Geom_Draw_Quadrangolo_Smart	Disegna il quadrangolo definito da 4 punti connettendoli con segmenti nell'ordine topologico corretto determinato dalla classificazione geometrica.

F — Gestori di Topologia 4P (Handler_) [5 funzioni]

Funzione	Scopo
Handler_Concavo_4P	Gestore grafico per quadrilateri CONCAVI (P4 interno al triangolo P1-P2-P3).
Handler_Trapezio_4P	Gestore grafico per quadrilateri di tipo Trapezio (con una coppia di lati paralleli).
Handler_Convesso_4P	Gestore grafico per quadrilateri Convessi Generici e Ciclici.
Handler_Simmetria_Centrale_4P	Generare l'impalcatura grafica di base esclusivamente per i quadrilateri a simmetria centrale perfetti.
Handler_Degenere_4P	Gestione visiva di emergenza per gli input malformati o collassati (Casi Degeneri).

G — Sottosistemi per Comando [147 funzioni]

G1 — Sottosistema CON1P [6 funzioni]

Funzione	Scopo
CON1P_Helper_Base_Vectors	Centralizzare la costruzione dell'ossatura vettoriale base a partire da due punti forti (Vertici o Fuochi).
CON1P_Helper_Engine_Setup	Centralizzare la preparazione della memoria e dello spazio locale per i motori geometrici 1P.
CON1P_Helper_UI_State	"Potenziometro" universale per l'interfaccia dei comandi 1P.
CON1P_Helper_UI_Prompt	Generatore universale di prompt testuali interattivi per i loop LUSNAP dei comandi 1P.
CON1P_Geom_Conic_2V_Engine	Motore di Calcolo: 2 Vertici + 1 Punto.
CON1P_Geom_Conic_2F_Engine	Motore Bifocale: 2 Fuochi + 1 Punto.

G2 — Sottosistema CON4P [45 funzioni]

Funzione	Scopo
CON4P_Execute_Conic_Action	Smistare al CAD finale per il disegno delle coniche scelte.
CON4P_Execute_Center_Conic	Elaborare e disegnare una conica vincolata a un punto target trattato come centro puro.
CON4P_Refresh_Last_Conic	Ridisegnare l'ultima conica calcolata in seguito a un aggiornamento del parametro opt_e.
CON4P_Evaluate_LUSNAP_Target	Elaborare la priorità di aggancio tra i nodi parabolici discreti e la mappa densa del Micro-Scan frattale.

CON4P_Identify_Node_Target	Elaborare l'aggancio del cursore sui nodi analitici primari gestendo lo switch circolare delle soluzioni multiple.
CON4P_Identify_Locus_Target	Smistatore LUSNAP globale per i luoghi continui e discreti.
CON4P_Geom_Master_Exit_Handler	Pulizia dell'ambiente di lavoro al termine di un comando Master.
CON4P_Geom_4P_Master_Setup	Inizializzazione sicura e standardizzata dell'ambiente di calcolo per tutti i comandi della suite.
CON4P_Get_Config_String	Formattazione testuale rapida della topologia per i report a video.
CON4P_Geom_Format_Trapezoid_Locus	Ricostruire e segmentare in modo esatto la retta dei centri nei quadrangoli di tipo Trapezio.
CON4P_Capture_Locus_Entities	Intercettatore di entità sicuro e isolato per l'ambiente CON4P.
CON4P_Geom_Draw_Locus_And_Capture	Disegna fisicamente nell'ambiente CAD il luogo geometrico dei centri corrispondente alla topologia.
CON4P_Local_Collect	Estrazione sicura e cronologica degli oggetti disegnati nel database.
CON4P_Draw_Points_With_Colors	Disegnare una o più nuvole di punti discrete a schermo, colorandole dinamicamente.
CON4P_Geom_Get_Projected_Center	Ricerca della distanza ortogonale minima su geometrie continue.
CON4P_Geom_Find_Nearest_Data	LUSNAP discreto sui metadati (Parabole/Nodi). Ricerca euclidea diretta.
CON4P_Geom_Calc_Micro_Param_List	Genera un array densissimo di parametri T attorno a un nodo candidato su curva VLA.
CON4P_Geom_Eval_Micro_Param	Valutatore geometrico locale per raffinamento.
CON4P_Geom_MicroScan	Motore di raffinamento finale LUSNAP ibrido (Supporta mappe VLA e Mappe Discrete).
CON4P_Geom_Draw_Target_Parabola	Individuare e tracciare la parabola bersaglio prevenendo instabilità matematiche da scala.
CON4P_Geom_Update_MetaMap	Popola la Meta-Mappa globale iterando su una lista di punti e associando a ciascuno i riferimenti VLA/Metadati.
CON4P_Geom_Classify_And_Normalize	Helper. Isola la logica di classificazione e normalizzazione geometrica della conica.
CON4P_Process_Single_VLA_Point	Elaboratore atomico per isolare e calcolare i punti geometrici discreti (fuochi o vertici) estratti da un singolo parametro locale di una curva VLA.

CON4P_Elabora_Ramo_Partizionato	Iteratore di partizionamento logico. Scorre i segmenti sicuri di un ramo spezzato e innesca il Micro-Scan frattale.
CON4P_Geom_Scan_Parallelogram_Locus	Generatore discreto adattivo per calcolare analiticamente il luogo dei fuochi e dei vertici in Parallelogrammi e Rombi.
CON4P_Draw_Orthogonal_Locus	Generare i binari d'asse (luogo geometrico) per Quadrati e Rettangoli con settorializzazione cromatica.
CON4P_Solve_Orthogonal_Conic	Risolvere analiticamente i semiassi di una conica ortogonale dato il centro, un vertice del rettangolo e un punto proiettato sull'asse.
CON4P_Geom_Conic_4Pv_Engine_Local	Motore grafico specializzato per il rendering delle coniche del Ramo 2A (Rettangolo/Quadrato).
CON4P_Refresh_Orthogonal_Graph	Ricalcolare e ridisegnare istantaneamente la conica ortogonale corrente in seguito alla modifica dell'estensione.
CON4P_Identify_Orthogonal_Target	Interfaccia LUSNAP esclusiva per il Ramo 2A (Rettangoli). Proietta il click, risolve la conica e lancia il rendering.
CON4P_Helper_Pack_And_Draw	Accorpa l'estrazione delle mappe di scansione e il disegno dei punti notevoli a schermo.
CON4P_Helper_Auto_Matrioska	Calcolare dinamicamente lo step parametrico di crociera e innescare la scansione frattale (Matrioska) sui segmenti VLA continui di una curva.
CON4P_Helper_Solve_Normalized_Conic	Centralizzare e ottimizzare la normalizzazione spaziale preventiva e la risoluzione matriciale della conica passante per i vertici dato un centro specifico.
CON4P_Geom_Scan_Orthocentric_Circle	Scansione dell'anello di Eulero per la ricerca dei centri delle coniche degeneri, addensando il campionamento in prossimità dei punti critici.
CON4P_Core_Matrioska_Loop	Helper per eseguire il ciclo iterativo frattale (Matrioska) sia in avvicinamento che in allontanamento.
CON4P_Core_Matrioska_Engine	Motore matematico centralizzato per il calcolo della distribuzione frattale dei parametri.
CON4P_Process_Segment_Matrioska	Interfaccia di campionamento asimmetrico per la generazione fisica.
CON4P_Get_Matrioska_Angles	Interfaccia per il prelievo puramente matematico di parametri (angoli).
CON4P_Geom_Search_Grid_Ell_Param	Trovare il parametro sulla curva dove l'ellisse passante per i 4 punti raggiunge la massima circolarità (Ellisse Pseudo-Circolare CEPC).
CON4P_Draw_ConIpe_Equil	Analizzare una quaterna di punti e, se topologicamente idonea, calcolare e tracciare istantaneamente l'unica iperbole equilatera algebrica passante per essi.
CON4P_Draw_ConEll_Special	Estrarre e tracciare l'Ellisse Pseudo-Circolare (CEPC) recuperando il centro ottimale marchiato durante la scansione Matrioska del modulo CON4P.

CON4P_Get_Global_Config	Centralizzare e fornire i parametri numerici e la palette cromatica vitali all'intero motore geometrico della suite CON4P.
CON4P_Geom_Solve_Unique_Equilateral_Ipe	Calcolare la "carta d'identità" geometrica dell'unica Iperbole Equilatera passante per i 4 vertici base del quadrangolo.
CON4P_Helper_UI_Prompt	Centralizzare e generare dinamicamente le stringhe di prompt e i comandi testuali (initget) per i cicli interattivi del comando Master.
CON4P_Geom_Scan_VLA_Centers	Radar topologico per le coniche centrali. Scansiona i luoghi geometrici VLA per generare le mappe di fuochi e vertici applicando il partizionamento frattale. Isola ed estrae i dati dell'Ellisse Pseudo-Circolare (CEPC).

G3 — Sottosistema PAR2PFAPI [17 funzioni]

Funzione	Scopo
Get_Dima_Data	Restituisce l'array statico di punti 2D che definisce la curva di riferimento "Dima" (sagoma di calibrazione), usata come dato di test per i comandi di parabola a due punti fissi.
Helper_Update_Dima_Par2pv	Aggiorna i confini guida della Dima nel contesto del comando par2PV: rigenera o elimina i confini topologici al variare dei punti di ancoraggio.
Geom_Eval_Par_Error	Valuta l'errore geometrico di una parabola con asse inclinato di ang per due punti fissi: quantifica lo scarto dalla condizione di parabola esatta.
PAR2PFAPI_Helper_UI_Prompt	Generare in modo dinamico la stringa di prompt interattivo per il ciclo continuo LUSNAP, riflettendo lo stato dei toggle di visibilità.
PAR2PFAPI_Modulo_Pi	Normalizzare un valore angolare espresso in radianti all'interno del semipiano positivo standard [0, PI).
PAR2PFAPI_Fractal_Around_Param_Safe	Iniettare una cascata di micro-parametri di infittimento spaziale (struttura a "Matrioska") attorno a una singolarità matematica (asintoto).
PAR2PFAPI_Calc_F_Linear	Ricavare analiticamente la distanza focale 'f' di una parabola passante per due punti base fissi (P1, P2), dato un fuoco campionato.
PAR2PFAPI_Engine_Proprio	Generare le mappe discrete per il fascio proprio di parabole (asse rotante su P3), applicando l'infittimento "Matrioska" sulle singolarità.
PAR2PFAPI_Engine_Improprio	Generare le mappe discrete ad alta densità per il fascio improprio di parabole, sfruttando l'iperbole "Light" (solo rami) come luogo dei fuochi.
PAR2PFAPI_Universal_Loop	Gestore interattivo asincrono per la manipolazione in tempo reale della parabola tramite aggancio ai luoghi geometrici (LUSNAP).
PAR2PFAPI_Draw_Engine	Driver grafico deputato all'estrazione del payload e al rendering istantaneo della parabola "elastica".
PAR2PFAPI_Libero_Runner	Gestire la generazione interattiva di parabole in modalità libera, utilizzando la ricerca euristica a 360 gradi e la dima topologica.
PAR2PFAPI_Get_Top3_Indices	Individuare i tre migliori candidati (indici) nella mappa di payload per gestire zone di sovrapposizione (fiocchi) o rami multipli.

PAR2PFAPI_Refine_Locus_Ultra	Motore di raffinamento sub-pixel con gestione Fiocchi (Top 3) e correzione di fase per Vertici. LOGICA DI PRECISIONE: - PROPRIO: Scansione angolare tra i vicini (i-1, i+1). - IMPROPRIO: Se cerco V, uso il Fuoco del payload come ancora pe…
PAR2PFAPI_Helper_Nearest_Control_Pt	Identificare quale dei tre punti di controllo (P1, P2 o P3) à più vicino al clic dell'utente.
PAR2PFAPI_Draw_Skeleton_Rea	Feedback visivo del triangolo di controllo (P1, P2, P3).
PAR2PFAPI_Draw_Locus_Entities_Rea	Generare i luoghi geometrici come nuvole di punti colorati separati.

G4 — Sottosistemi ELL3P e IPE3P [32 funzioni]

Funzione	Scopo
ELL3P_Helper_UI_Prompt	Generare la stringa testuale dinamica per il prompt dei comandi in base allo stato delle opzioni visive e operative.
ELL3P_Master_Engine	Elaborare e generare in una singola passata le mappe di parametri e i gruppi di entità per Centri, Fuochi e Vertici dell'Ellisse.
ELL3P_Eval_Param_Distance	Risolvere matematicamente l'ellisse per un parametro istantaneo e restituire la distanza minima dal mouse.
ELL3P_Refine_Locus	Catturare ed affinare con elevata precisione l'interpolazione del luogo geometrico in funzione del movimento del cursore (LUSNAP continuo).
ELL3P_Math_Error_Func	Funzione di Costo per quantificare la discrepanza geometrica rispetto a un Nodo target.
ELL3P_Eval_Cands	Estrarre in real-time l'elenco dei punti notevoli candidati per un dato setup parametrico.
ELL3P_Solve_Exact_Node_Algebraic	Trovare radici puramente analitiche in modalità Pre-Flight (silente) per stazionare sui Nodi base del LUSNAP.
ELL3P_Precalc_Exact_Nodes	Calcolare preventivamente le radici analitiche per formare l'archivio volatile O(1).
ELL3P_Draw_Engine	Driver grafico deputato all'iniezione delle entità CAD (rendering e sostituzione schermo).
ELL3P_Get_Data	Acquisizione dell'impalcatura spaziale e delle variabili OSMODE ambientali.
ELL3P_Get_Constraints	Inizializzare le logiche e i vincoli operativi matematici forzati dall'utente (Rapporto o Direzione).
ELL3P_Universal_Loop	Gestore asincrono dell'I/O (LUSNAP e Ciclatore) per l'applicazione Ellisse.
IPE3P_Helper_UI_Prompt	Generare in modo dinamico la stringa di prompt interattivo per il ciclo continuo LUSNAP.
IPE3P_Find_Absolute_Nearest_Data	Cercare il punto discretizzato spazialmente più vicino al cursore del mouse.
IPE3P_Modulo_Pi	Normalizzare un angolo nel semipiano positivo standard [0, PI).

IPE3P_Helper_Orthogonal_Bisectors	Calcolare la coppia di bisettrici ortogonali date due direzioni angolari.
IPE3P_Helper_Calc_Angles_Singularity	Calcolare le bisettrici ortogonali (assi focali) per la conica degenere su un singolo lato del triangolo base.
IPE3P_Calc_Exact_Singularity_Params	Ricavare analiticamente i parametri angolari esatti in cui l'iperbole degenera in una coppia di rette (schianto sui lati).
IPE3P_Helper_Fractal_Band	Generare una singola "banda" di micro-parametri simmetrici attorno a una singolarità.
IPE3P_Fractal_Around_Param_Safe	Iniettare una cascata di micro-parametri di infittimento spaziale addossati a una singolarità ("Matrioska"), omettendola rigorosamente.
IPE3P_Helper_Draw_And_Group_Locus	Disegnare una nuvola di punti con un colore specifico e raggrupparla automaticamente.
IPE3P_Master_Engine_Pure	Generare le mappe discrete ad alta densità in modalità Single-Pass, pre-inglobando l'iniezione geometrica del Microscan.
IPE3P_Eval_Cands	Elaboratore algebrico real-time a supporto del raffinamento continuo interattivo LUSNAP.
IPE3P_Refine_Locus_Pure	Adattare in tempo reale e interpolare la griglia puramente algebrica per catturare il cursore utente (regola del braccio elastico).
IPE3P_Get_Best_Refined_Among_Lists	Eleggere il pacchetto visivo migliore in gara multicore interrogando simultaneamente la Mappa Base e la Mappa Inversa.
IPE3P_Math_Error_Func	Sensore di prossimità euristico finalizzato all'aggancio dei Nodi Esatti.
IPE3P_Solve_Exact_Node_Algebraic	Popolare l'archivio RAM di soluzioni chiuse per le stazioni d'arresto ciclabili dei Nodi di base (Vertici/Fuochi).
IPE3P_Precalc_Exact_Nodes	Strutturare la Mappa ad accesso istantaneo del Ciclattore LUSNAP Nodi.
IPE3P_Draw_Engine	Driver CAD per la vettorializzazione dell'iperbole a estensione dinamica asintotica.
IPE3P_Get_Data	Richiesta architettura spaziale in ingresso.
IPE3P_Get_Constraints	Selezione vincolo primario d'indagine.
IPE3P_Universal_Loop	Processore Master degli input utente asincroni durante la manipolazione dello spettro continuo.

G5 — Sottosistema PAR3P [15 funzioni]

Funzione	**Scopo**
update_3pv_guides	Comando principale interattivo per la generazione dinamica di coniche partendo da 2 Vertici e 1 Punto.
PAR3P_Calc_Par_V_2P	Calcolare analiticamente i parametri esatti dell'unica parabola passante per 3 punti, forzando il primo come Vertice.

PAR3P_Precalc_Exact_Nodes	Elaborare preventivamente l'Oracolo Geometrico per i Nodi base.
PAR3P_Master_Par_3P	Calcolo luoghi e generazione entità CAD raggruppate.
PAR3P_Helper_UI_Prompt	Centralizzare e generare la stringa di prompt per l'interfaccia testuale interattiva del comando Master.
PAR3P_Geom_Get_Closest_Guide_Target	Iterare su una lista di entità guida per trovare il punto più vicino al click dell'utente, aggiornando i record di distanza minima.
PAR3P_Universal_Loop	Controller interattivo per l'esplorazione e la generazione mirata (LUSNAP) delle parabole passanti per la terna di punti base.
PAR3P_Master_Reactive_Universal_3P	Controller per la modalità di editing reattivo (live editing) dei luoghi geometrici delle parabole passanti per 3 punti.
Geom_3Pts_Core_Proprio	Elaborare e tracciare la conica passante per 3 punti noti e un centro proprio (ellisse o iperbole).
Geom_3Pts_Core_Improprio	Elaborare e tracciare la parabola passante per 3 punti noti, vincolata da una direzione asintotica (centro improprio).
Geom_3Pts_Update_Guides	Generare e aggiornare dinamicamente il set di linee guida visive (mediane estese) del triangolo base durante l'esplorazione LUSNAP.
Geom_3Pts_Master_Engine	Gestire l'interfaccia utente dinamica (dashboard contestuale) e il ciclo di esecuzione per la risoluzione interattiva della conica a 3 punti.
Geom_Get_Par_Vertices_List	Calcolare analiticamente le posizioni assolute dei vertici delle parabole limitate dalla quaterna base, strutturando la mappa LUSNAP di interfaccia.
Geom_Get_Par_Foci_List	Calcolare e raggruppare le coordinate WCS dei fuochi delle parabole inscritte nella quaterna base, generando la relativa mappa dati per il LUSNAP.
Geom_5P_Update_Guides	Aggiorna in tempo reale le entità guida nell'ambiente interattivo a 5 punti, ridisegnandole ogni volta che il set di punti cambia.

G6 — Infrastruttura Comandi e Algoritmi Pascal [30 funzioni]

Funzione	**Scopo**
Get_String_Def	Chiede all'utente di inserire una stringa testuale con un valore predefinito: se l'utente preme Invio senza digitare nulla, restituisce il valore di default.
Get_Int_Pos	Chiede all'utente di inserire un numero intero positivo, validando che il valore sia ≥ 1 prima di restituirlo.
Pascal_Calcola_Primo_Diametro	Calcola il primo diametro del teorema di Pascal per una conica: l'intersezione dei lati P1P2 e P4P5 dell'esagono inscritto nella conica.
Pascal_Calcola_Corde_Coniugate	Calcola le corde coniugate di Pascal per i 5 punti dati, individuando le intersezioni tra i lati opposti dell'esagono parziale.
Solve_Par_Geometry	Risolve completamente la geometria di una parabola nota la sua equazione parametrica nel sistema locale: calcola fuoco, vertice, direttrice e semiparametro.

get-max-abs-projection-on-axis	Calcola la proiezione massima (in valore assoluto) di una lista di punti su una direzione d'asse data, usata per stimare l'estensione della conica.
Get_Layer_Smart	Recupera il nome del layer corrente o di un layer predefinito del sistema Coniche, verificando che esista nel documento; se non esiste, lo crea automaticamente.
Init_Conic_App	Funzione di inizializzazione comune a tutti i comandi del sistema Coniche: salva le variabili di sistema (OSMODE, PDMODE, CMDECHO), acquisisce il set iniziale di punti dall'utente e prepara l'ambiente.
Restore_Conic_App	Ripristina le variabili di sistema AutoCAD (OSMODE, PDMODE, CMDECHO) ai valori salvati prima dell'avvio del comando, garantendo il cleanup dell'ambiente alla conclusione.
Alg3x3_Trasla	Fornire un set di utilità algebriche e matriciali per supportare la costruzione e la risoluzione di sistemi lineari 3x3 (frequentemente impiegati nel calcolo analitico dei coefficienti delle coniche). Gestiscono vettori relativi, vincoli…
Alg3x3_Vincolo	Fornire un set di utilità algebriche e matriciali per supportare la costruzione e la risoluzione di sistemi lineari 3x3 (frequentemente impiegati nel calcolo analitico dei coefficienti delle coniche). Gestiscono vettori relativi, vincoli…
Alg3x3_Sostituisci_Colonna	Fornire un set di utilità algebriche e matriciali per supportare la costruzione e la risoluzione di sistemi lineari 3x3 (frequentemente impiegati nel calcolo analitico dei coefficienti delle coniche). Gestiscono vettori relativi, vincoli…
Alg3x3_Det	Completare il motore algebrico 3x3 fornendo le funzioni per il calcolo del determinante, la risoluzione di sistemi lineari tramite la regola di Cramer e l'estrazione diretta dei coefficienti quadratici di una conica a centro vincolata a …
Alg3x3_Risolvi	Risolve un sistema lineare 3×3 Mx=rhs con la regola di Cramer: calcola il determinante della matrice e divide i determinanti modificati.
Alg3x3_Calcola_Coeffs	Calcola i coefficienti (A,B,C) della forma quadratica centrata $Ax^2+Bxy+Cy^2=1$ della conica passante per 3 punti dati, con centro fisso noto.
Alg3x3_Autovalori	Determinare analiticamente gli autovalori e gli autovettori associati alla matrice della forma quadratica della conica, calcoli fondamentali per l'estrazione delle lunghezze dei semiassi principali e per definire il loro corretto orienta…
Alg3x3_Autovettore	Calcola l'autovettore normalizzato (versore) della matrice simmetrica 2×2 corrispondente all'autovalore lam dato, per trovare la direzione dell'asse principale di una conica.
Alg3x3_Punto_Su_Asse	Calcola il punto sull'asse della conica (definito dalla direzione dir) che si trova a distanza parametrica t dall'origine, usato per determinare i vertici della conica nel sistema di riferimento locale.
Alg4x4_Rimuovi_Colonna	Fornire un set di utilità per la manipolazione strutturale di righe e matrici di ordine 4 (4x4), permettendo l'estrazione rapida di minori complementari e la sostituzione dinamica di vettori colonna per la risoluzione dei sistemi.
Alg4x4_Sostituisci_Colonna	Sostituisce una colonna della matrice 4×4 con il vettore rhs fornito (passo preparatorio per la regola di Cramer).
Alg4x4_Det	Calcolare il determinante di una matrice quadrata di ordine 4 (4x4) e risolvere un sistema di equazioni lineari a 4 incognite tramite l'applicazione sistematica della regola di Cramer.

Alg4x4_Risolvi	Risolve un sistema lineare 4×4 Mx=rhs con la regola di Cramer, usato per trovare i 4 coefficienti omogenei di una conica passante per 4 punti.
analyze-conic-geometry	Analizza la geometria di una conica dai suoi 6 coefficienti e ne estrae tipo, centro, semiassi, fuochi, eccentricità e orientamento, con restituzione in formato strutturato a chiave-valore.
Pascal_Genera_Sesto_Punto	Applica il teorema di Pascal per determinare il sesto punto di una conica: dati 5 punti sulla conica, trova il sesto usando le intersezioni delle corde dell'esagono di Pascal.
Pascal_Calcola_Singola_Corda	Calcola una singola corda dell'esagono di Pascal: la retta passante per due dei sei vertici dati, restituita in forma di equazione generale.
Pascal_Intersezione_Diametri	Calcola il punto di intersezione tra le due rettediametri di Pascal, ricongiungendo i due assi dell'esagono per trovare il punto di Pascal.
pascal-core-attempt	Tentativo nucleo dell'algoritmo di Pascal: esegue un singolo tentativo di trovare il sesto punto a partire da una permutazione specifica dei 5 punti dati, restituendo il risultato o nil in caso di fallimento.
Log_Pascal_Iteration	Registra nel log i risultati di una singola iterazione dell'algoritmo di Pascal: il tipo di tentativo, la permutazione usata, il risultato e il tipo di conica trovata.
get-pascal-center-frame	Costruisce e restituisce il "frame" del centro di Pascal: le informazioni geometriche (centro, tipo, statistiche) derivate dall'analisi di tutti i tentativi dell'algoritmo di Pascal sui 5 punti dati.
Alg_Tournament_5P_Centered	Algoritmo a torneo per trovare la conica centrata (con centro fisso dato) che minimizza l'errore su 5 punti dati, usando una strategia 4-su-5: prova tutte le combinazioni di 4 punti e sceglie la soluzione migliore.

H — Utilità e Servizi [62 funzioni]

H1 — Utilità Geometrico-Matematiche [29 funzioni]

Funzione	Scopo
Util_Get_Max_Dist_From_Point	Determinare rispettivamente la distanza euclidea massima e minima intercorrente tra un punto di riferimento specifico e una nuvola o lista di punti spaziali assegnata.
Util_Get_Min_Dist_From_Point	Determinare rispettivamente la distanza euclidea massima e minima intercorrente tra un punto di riferimento specifico e una nuvola o lista di punti spaziali assegnata.
Util_Media_lisNum	Calcola la media aritmetica di una lista di valori numerici.
Util_Media_2Num	Calcola la media aritmetica di due valori numerici.
Util_Translate_Pts_Relative	Trasla una lista di punti sottraendo il punto base da ciascuno, portandoli nel sistema di riferimento locale centrato sul punto base.
Util_Translate_Pts_Global	Trasla una lista di punti nel sistema globale aggiungendo il punto base, invertendo la traslazione locale applicata da Util_Translate_Pts_Relative.
Util_Denorm_Pt	Denormalizza un singolo punto dal sistema normalizzato (baricentro=origine, max_dist=unità) al sistema reale delle coordinate di disegno.

Util_Denorm_List_Pts	Denormalizza una lista intera di punti dal sistema normalizzato al sistema reale.
Util_Rot_2D_Dir_CS	Ruota un punto 2D nella direzione diretta (senso positivo) dato il coseno e seno dell'angolo precompilati.
Util_Rot_2D_Inv_CS	Ruota un punto 2D nella direzione inversa (senso negativo) dato il coseno e seno dell'angolo precompilati, ovvero applica la rotazione di −ang.
Util_Rot_List_2D_CS	Ruota in senso diretto una lista intera di punti 2D usando coseno e seno precompilati per efficienza.
Util_Rot_List_2D_Inv_CS	Ruota in senso inverso una lista intera di punti 2D usando coseno e seno precompilati per efficienza.
Util_Trans_List_Glo_To_Loc	Trasforma una lista di punti dal sistema globale (WCS) al sistema locale definito da un'origine e un angolo di orientamento, applicando traslazione seguita da rotazione.
Util_Trans_List_Loc_To_Glo	Trasforma una lista di punti dal sistema locale al sistema globale (WCS), invertendo la trasformazione di Util_Trans_List_Glo_To_Loc.
Util_Distanza_Media_lisPu_Da_Cen	Calcola la distanza media di una lista di punti da un punto centro dato, usata come stima del raggio medio della conica.
Util_Scala_lisPu_Da_Cen	Scala una lista di punti rispetto a un centro dato, moltiplicando per un fattore le distanze al centro (ingrandimento/riduzione omotetica).
Util_Normalizza_lisPu_Da_Cen	Normalizza una lista di punti rispetto al loro baricentro e alla distanza massima, portando i punti in un sistema numerico stabile centrato sull'origine con raggio ≤ 1.
Util_Angle_Diff	Calcola la differenza angolare tra due angoli in radianti, restituendo il valore nell'intervallo [−π, π].
Util_Orthogonality_Score	Calcola un punteggio di ortogonalità per un insieme di direzioni angolari, misurando quanto ogni coppia di direzioni si discosta dalla perpendicolarità reciproca.
Util_Mean_Angle	Calcola la media angolare di una lista di angoli in radianti, gestendo correttamente il problema della discontinuità a ±π con il metodo vettoriale.
Util_Filter_Outliers_By_Centroid	Filtra i punti di una lista eliminando gli outlier (valori anomali) che superano una distanza soglia dal baricentro, proteggendo la qualità del fit.
Util_Sort_By_Orthogonality	Ordina una lista di ipotesi geometric (angoli o soluzioni) in base al loro punteggio di ortogonalità, mettendo prima le soluzioni più ortogonali.
Util_Sort_By_Angle_Deviation	Ordina una lista di soluzioni in base alla deviazione angolare rispetto a una direzione di riferimento, per trovare la conica con asse più allineato.
Util_List_Rotations	Genera tutte le rotazioni ciclice di una lista: per una lista di n elementi restituisce n liste, ciascuna ottenuta spostando il primo elemento in fondo.
Util_Trans_Pts_List	Trasla una lista di punti applicando un vettore di traslazione comune, alias semplificato di Util_Translate_Pts_Global per contesti specifici.
Util_Scale_Pts_From_Centroid	Scala una lista di punti rispetto al loro baricentro calcolato automaticamente, usato per estendere le fasce di ricerca del luogo geometrico.

Util_Seg_Crosses_Triangle	Verifica se un segmento interseca uno dei lati del triangolo dato, usato per rilevare attraversamenti del bordo durante la scansione del luogo.
Util_Normalize_Pts_Set	Normalizza un set di punti calcolando il baricentro e la distanza massima, restituendo i dati necessari per la normalizzazione e denormalizzazione.
get_ratio	Trovare il parametro sulla curva dove l'ellisse passante per i 4 punti raggiunge la massima circolarità (Ellisse Pseudo-Circolare CEPC).

H2 — Servizi AutoCAD [19 funzioni]

Funzione	Scopo
Util_Make_Anonymous_Group	Raggruppare dinamicamente una lista di entità vettoriali in un Gruppo Anonimo di AutoCAD per facilitarne la selezione e la manipolazione congiunta a video.
Util_Get_Pts_BBox_Size	Calcolare le dimensioni del bounding box (ingombro planare) di una nuvola di punti, determinando le escursioni massime (Delta) lungo gli assi X e Y.
Util_SafeLayer	Creare silenziosamente un layer o forzarne l'aggiornamento cromatico, operando nel database senza alterare il layer corrente.
Util_Pun_To_Str	Fornire un set unificato di utilità (wrapper) per la conversione sicura di vari tipi di dato LISP (Punti 3D, Simboli, Numeri Reali, Interi) in formato stringa testuale, standardizzando il logging e il feedback a video.
Util_Sym_To_Str	Fornire un set unificato di utilità (wrapper) per la conversione sicura di vari tipi di dato LISP (Punti 3D, Simboli, Numeri Reali, Interi) in formato stringa testuale, standardizzando il logging e il feedback a video.
Util_Num_To_Str	Fornire un set unificato di utilità (wrapper) per la conversione sicura di vari tipi di dato LISP (Punti 3D, Simboli, Numeri Reali, Interi) in formato stringa testuale, standardizzando il logging e il feedback a video.
Util_Int_To_Str	Fornire un set unificato di utilità (wrapper) per la conversione sicura di vari tipi di dato LISP (Punti 3D, Simboli, Numeri Reali, Interi) in formato stringa testuale, standardizzando il logging e il feedback a video.
Util_Print_And_Nil	Stampare un messaggio testuale sulla riga di comando di AutoCAD e restituire simultaneamente il valore nullo (nil), utile per gestire in modo compatto le uscite di errore o i fallback all'interno dei costrutti condizionali.
Util_Conv_List_To_VLA	Converte una lista LISP di punti in un array VLA (Visual LISP ActiveX) di varianti, formato richiesto da alcune API di AutoCAD quali AddPolyline.
Util_Capture_After_Entity	Raccoglie tutte le entità AutoCAD create dopo l'entità di riferimento data, includendo i blocchi anonimi e le sottoentità visibili.
Util_Collect_Ents	Raccoglie in una lista tutte le entità AutoCAD create dopo un'entità di riferimento data.
Util_Del_Ents_List	Elimina dal database AutoCAD tutte le entità contenute in una lista, verificando preventivamente la loro esistenza.
Util_Select_Point_From_List	Prompt interattivo che chiede all'utente di cliccare su un punto del disegno e verifica se il punto selezionato appartiene a una lista di candidati.

Util_Select_Winning_Hypothesis	Chiede all'utente di indicare quale tra le ipotesi geometriche mostrate (punti/vettori colorati) è quella corretta, restituendo l'ipotesi selezionata.
PrincPu	Stampa a schermo la rappresentazione testuale di un punto o di una lista di punti, utile per il debug interattivo in AutoCAD.
Util_Del_Ents_At_Pt	Elimina tutte le entità AutoCAD che si trovano entro una piccola area di tolleranza attorno a un punto dato, filtrate per layer.
Safe_Collect_Ents	Raccogliere in modo sicuro tutte le entità del database di AutoCAD create a partire da un determinato puntatore in poi.
Util_Update_Guides_Vis	Aggiornare lo stato di visibilità (acceso/spento) di un intero gruppo di entità grafiche nel disegno.
ssadd_list	Convertire una lista standard di nomi entità (ename) in un set di selezione (selection-set) AutoCAD.

H3 — Analisi e Manipolazione di Liste [14 funzioni]

Funzione	**Scopo**
Util_Reorder_List_By_Indices	Riordina una lista secondo un vettore di indici dato, restituendo una nuova lista in cui l'i-esimo elemento è l'elemento della lista originale alla posizione indices[i].
Util_Extract_Val	Estrae il valore di campo specificato da un record della lista dati strutturata del motore di scansione, usato per accedere ai campi dei risultati intermedi.
Util_Reset_Vars	Reimposta un insieme di variabili a nil in un'unica chiamata, usato nei cicli di init/reset dei loop interattivi.
Util_Filter_By_Nth	Filtra una lista di record mantenendo solo quelli il cui n-esimo elemento corrisponde a un valore specificato.
Util_Filter_And_Count_By_Nth	Filtra e conta contemporaneamente i record di una lista in base al valore dell'n-esimo elemento, restituendo sia la lista filtrata sia il conteggio.
Util_Scan_And_Cut_Branches	Scansiona i rami del luogo geometrico e li divide nei punti in cui la distanza tra punti consecutivi supera una soglia (salto non fisico).
Util_Stitch_Branches_HeadTail	Unisce rami contigui del luogo geometrico collegando la coda di un ramo alla testa del successivo, se la distanza tra gli estremi è inferiore alla soglia.
Util_Process_Branches_Universal	Pipeline di processing completo dei rami del luogo geometrico: prima taglia i rami ai salti (Util_Scan_And_Cut_Branches), poi cucisce i tronconi vicini (Util_Stitch_Branches_HeadTail), infine scarta i rami troppo corti come artefatti.
Util_Process_Branches_Data	Processa e consolida i dati dei rami di un luogo geometrico ramificato, eliminando i rami spuri e riassemblando quelli validi in ordine topologico.
Util_Process_Branches_Data_SmartStitch	Post-processare i dati grezzi dei rami parabolici, prima frammentandoli in base ai salti focali e poi tentando una saldatura intelligente (Smart Stitch) tra l'inizio del primo ramo e la fine dell'ultimo per chiudere i loop.
Util_Scan_And_Cut_Branches_Data_By_Focus	Analizzare sequenzialmente una lista di dati e frammentarla in rami discreti ogni volta che la distanza spaziale tra due fuochi consecutivi supera la soglia consentita, identificando così i salti topologici o gli asintoti.

Util_Get_All_Pairs	Genera tutte le coppie non ordinate possibili (combinazioni C(n,2)) da una lista di elementi.
Lis_2nth	Estrae due elementi da una lista usando due indici numerici.
Lis_Coppie_nth	Costruisce una lista di coppie di elementi estratti da una lista, usando un array di coppie di indici come selettore.

I — Comandi AutoCAD (c:…) [15 funzioni]

Funzione	Scopo
c:slc	Comando di configurazione: inizializza e crea i layer standard del sistema Coniche nel documento AutoCAD corrente.
c:CON2V1P	Comando principale interattivo per la generazione dinamica di coniche partendo da 2 Vertici e 1 Punto.
c:CON2F1P	Comando principale interattivo per la generazione dinamica di coniche partendo da 2 Fuochi e 1 Punto.
c:parVF	Comando laboratorio interattivo per disegnare una parabola specificando il punto BASE come ancora fissa (vertice o fuoco a scelta) e il secondo punto come mobile (fuoco o vertice rispettivamente), con loop continuo che permette di ridise…
c:par2PV	Comando per disegnare la parabola passante per 2 punti fissi P1 e P2 con un terzo punto mobile V come vertice. Ad ogni posizione di V il sistema scansiona gli assi possibili e trova la parabola (o le parabole) che passano per P1, P2 e ha…
c:par2PFAPI	Comando Master per la Parabola a 2 Punti fissi con Fascio di Assi Proprio, Improprio o Libero.
c:par2PFAPIRea	Laboratorio Reattivo con raggruppamento LV/LF automatico ad ogni step e triangolo persistente.
c:ell3PRD	Eseguibile Master per l'inizializzazione dell'ambiente sperimentale dell'Ellisse 3P.
c:ipe3PRD	Starter eseguibile root, agganciato al timer di perfomance e all'Oracolo Geometrico che previene gli stalli matematici nativi.
c:par3P	Comando Master per l'ambiente statico Parabola 3P. Gestisce l'acquisizione dei punti, il calcolo e il rendering del loop LUSNAP.
c:par3PRea	Avviare l'ambiente interattivo di editing reattivo per la generazione dinamica dei luoghi geometrici delle parabole passanti per 3 punti.
c:con3pc	Funzione Wrapper ed entry-point per l'avvio dell'applicazione interattiva di tracciamento "Conica per 3 Punti + Centro".
c:CON4P	Comando principale per l'Ambiente Protetto Conica 4P. Gestisce l'intero ciclo: Selezione -> Analisi Topologica -> Rendering Interattivo per TUTTE le figure e configurazioni geometriche.
c:con5p	Comando laboratorio per disegnare la conica passante per 5 punti in modalità "base 4 punti fissi + 5° punto mobile": l'utente seleziona 4 punti base, poi continua ad aggiungere un 5° punto in posizioni diverse vedendo la conica aggiornar…

c:con5pRea	Comando di editing reattivo per la conica a 5 punti: a differenza di c:con5p, il 5° punto è già selezionato e l'utente sposta interattivamente il cursore vedendo la conica ridisegnarsi in tempo reale ad ogni nuovo punto indicato, mantene…

GLOSSARIO DELLE FUNZIONI

Elenco alfabetico di tutte le 472 funzioni e comandi, con firma, scopo, input, output e logica interna.

A

Alg3x3_Autovalori

Firma: (Alg3x3_Autovalori A B C)

Scopo: Determinare analiticamente gli autovalori e gli autovettori associati alla matrice della forma quadratica della conica, calcoli fondamentali per l'estrazione delle lunghezze dei semiassi principali e per definire il loro corretto orientamento spaziale.

Input: A, B, C (Numeri reali corrispondenti ai coefficienti quadratici della forma). Per l'autovettore à richiesto anche 'lam' (il valore scalare dell'autovalore specifico da cui ricavare la direzione).

Output: Rispettivamente: una lista di due numeri reali (le due radici o autovalori lambda), oppure una lista tridimensionale (X Y 0.0) rappresentante il vettore direzionale normalizzato dell'asse.

Logica interna: 'Alg3x3_Autovalori' sfrutta l'invarianza della traccia e del determinante per risolvere l'equazione caratteristica della sottomatrice simmetrica 2x2. 'Alg3x3_Autovettore' calcola i vettori associati valutando il coefficiente misto B: se non nullo, ricava la componente Y ponendo X=1, altrimenti assegna i vettori base. In uscita esegue la normalizzazione euclidea.

Alg3x3_Autovettore

Firma: (Alg3x3_Autovettore A B C lam)

Scopo: Calcola l'autovettore normalizzato (versore) della matrice simmetrica 2×2 corrispondente all'autovalore lam dato, per trovare la direzione dell'asse principale di una conica.

Input: A: coefficiente x^2. B: coefficiente xy (termine misto). C: coefficiente y^2. lam: autovalore calcolato.

Output: Lista (vx vy 0.0) del versore dell'autovettore normalizzato a lunghezza unitaria.

Logica interna: Se B≠0: imposta vx=1, vy=(lam−A)/(B/2). Se B≈0: sceglie l'asse canonico più vicino all'autovalore confrontando |lam−A| e |lam−C|. Normalizza tramite Pitag_Ip.

Alg3x3_Calcola_Coeffs

Firma: (Alg3x3_Calcola_Coeffs Center P1 P2 P3)

Scopo: Calcola i coefficienti (A,B,C) della forma quadratica centrata $Ax^2+Bxy+Cy^2=1$ della conica passante per 3 punti dati, con centro fisso noto.

Input: Center: coordinate del centro fisso. P1, P2, P3: tre punti di vincolo.

Output: Lista (A B C) dei coefficienti della forma quadratica nel sistema centrato su Center; nil se degenere.

Logica interna: Trasla ogni punto nel sistema centrato su Center con Alg3x3_Trasla, costruisce il vincolo di passaggio con Alg3x3_Vincolo (lista x^2, xy, y^2), poi risolve il sistema 3×3 con termine noto (1 1 1) tramite Alg3x3_Risolvi.

Alg3x3_Det

Firma: (Alg3x3_Det M)

Scopo: Completare il motore algebrico 3x3 fornendo le funzioni per il calcolo del determinante, la risoluzione di sistemi lineari tramite la regola di Cramer e l'estrazione diretta dei coefficienti quadratici di una conica a centro vincolata a passare per 3 punti noti.

Input: L'argomento varia. Per il determinante: M (Matrice 3x3). Per la risoluzione: M e rhs (Vettore dei termini noti). Per i coefficienti: Center (Centro della conica) e P1, P2, P3 (Punti WCS di passaggio).

Output: Rispettivamente: un numero reale (determinante), una lista di 3 numeri reali (soluzioni del sistema lineare), oppure la lista dei 3 coefficienti (A, B, C) della conica. In caso di determinante nullo o matrice singolare, le funzioni restituiscono nil.

Logica interna: 'Alg3x3_Det' appiattisce la matrice e calcola il determinante per espansione. 'Alg3x3_Risolvi' applica il Teorema di Cramer, verificando che il determinante principale superi la tolleranza di 1e-9, per poi risolvere le incognite sostituendo iterativamente la colonna dei termini noti. 'Alg3x3_Calcola_Coeffs' agisce da orchestratore: trasla i punti sul centro, costruisce la

matrice dei vincoli e risolve il sistema uguagliando i termini noti a (1.0 1.0 1.0) per estrarre analiticamente i parametri di forma.

Alg3x3_Punto_Su_Asse

Firma: (Alg3x3_Punto_Su_Asse A B C dir)

Scopo: Calcola il punto sull'asse della conica (definito dalla direzione dir) che si trova a distanza parametrica t dall'origine, usato per determinare i vertici della conica nel sistema di riferimento locale.

Input: A, B, C: coefficienti della forma quadratica. dir: versore (vx vy) dell'asse.

Output: Coordinate (vx·t, vy·t, 0.0) del punto sull'asse; nil se la direzione è degenere.

Logica interna: Calcola il denominatore den = $A \cdot vx^2 + B \cdot vx \cdot vy + C \cdot vy^2$ (la forma quadratica valutata sulla direzione). Se |den|>1e-50, calcola t=1/√|den| e restituisce il punto scalato.

Alg3x3_Risolvi

Firma: (Alg3x3_Risolvi M rhs)

Scopo: Risolve un sistema lineare 3×3 Mx=rhs con la regola di Cramer: calcola il determinante della matrice e divide i determinanti modificati.

Input: M: matrice 3×3 (lista di 3 liste di 3 elementi). rhs: vettore termine noto (lista di 3 valori).

Output: Lista di 3 valori (x1 x2 x3) soluzione del sistema; nil se il determinante è nullo.

Logica interna: Calcola det=Alg3x3_Det(M). Se |det|>1e-9, risolve mediante la regola di Cramer: per ogni incognita sostituisce la colonna corrispondente con rhs tramite Alg3x3_Sostituisci_Colonna, ricalcola il determinante e divide per det.

Alg3x3_Sostituisci_Colonna

Firma: (Alg3x3_Sostituisci_Colonna mat col vec)

Scopo: Fornire un set di utilità algebriche e matriciali per supportare la costruzione e la risoluzione di sistemi lineari 3x3 (frequentemente impiegati nel calcolo analitico dei coefficienti delle coniche). Gestiscono vettori relativi, vincoli quadratici e sostituzioni in matrice.

Input: L'argomento varia per funzione. Traslazione: Pu1, Pu2 (coordinate). Vincolo: Pu (punto 2D o 3D). Matrice: mat (lista di liste, matrice 3x3), col (indice intero zero-based della colonna), vec (lista di valori del vettore colonna da inserire).

Output: Rispettivamente: un vettore relativo di traslazione (delta), una lista di tre termini quadratici geometrici, oppure una nuova matrice 3x3 aggiornata con la colonna in ingresso.

Logica interna: 'Alg3x3_Trasla' applica una semplice differenza vettoriale tramite 'mapcar'. 'Alg3x3_Vincolo' destruttura le componenti X e Y del punto e calcola i termini di secondo grado (x^2, xy, y^2) necessari per imporre il passaggio della conica per quel punto. Infine, 'Alg3x3_Sostituisci_Colonna' itera sulle righe della matrice sostituendo dinamicamente l'n-esimo elemento con i valori del nuovo vettore, operazione chiave per la risoluzione tramite regola di Cramer. (defun Alg3x3_Trasla (Pu1 Pu2) (mapcar '- Pu2 Pu1)) (defun Alg3x3_Vincolo (Pu / x y) (mapcar 'set '(x y) Pu) (list (Math_Sqr x) (* x y) (Math_Sqr y)))

Alg3x3_Trasla

Firma: (Alg3x3_Trasla Pu1 Pu2)

Scopo: Fornire un set di utilità algebriche e matriciali per supportare la costruzione e la risoluzione di sistemi lineari 3x3 (frequentemente impiegati nel calcolo analitico dei coefficienti delle coniche). Gestiscono vettori relativi, vincoli quadratici e sostituzioni in matrice.

Input: L'argomento varia per funzione. Traslazione: Pu1, Pu2 (coordinate). Vincolo: Pu (punto 2D o 3D). Matrice: mat (lista di liste, matrice 3x3), col (indice intero zero-based della colonna), vec (lista di valori del vettore colonna da inserire).

Output: Rispettivamente: un vettore relativo di traslazione (delta), una lista di tre termini quadratici geometrici, oppure una nuova matrice 3x3 aggiornata con la colonna in ingresso.

Logica interna: 'Alg3x3_Trasla' applica una semplice differenza vettoriale tramite 'mapcar'. 'Alg3x3_Vincolo' destruttura le componenti X e Y del punto e calcola i termini di secondo grado (x^2, xy, y^2) necessari per imporre il passaggio della conica per quel punto. Infine,

'Alg3x3_Sostituisci_Colonna' itera sulle righe della matrice sostituendo dinamicamente l'n-esimo elemento con i valori del nuovo vettore, operazione chiave per la risoluzione tramite regola di Cramer.

Alg3x3_Vincolo

Firma: (Alg3x3_Vincolo Pu)

Scopo: Fornire un set di utilità algebriche e matriciali per supportare la costruzione e la risoluzione di sistemi lineari 3x3 (frequentemente impiegati nel calcolo analitico dei coefficienti delle coniche). Gestiscono vettori relativi, vincoli quadratici e sostituzioni in matrice.

Input: L'argomento varia per funzione. Traslazione: Pu1, Pu2 (coordinate). Vincolo: Pu (punto 2D o 3D). Matrice: mat (lista di liste, matrice 3x3), col (indice intero zero-based della colonna), vec (lista di valori del vettore colonna da inserire).

Output: Rispettivamente: un vettore relativo di traslazione (delta), una lista di tre termini quadratici geometrici, oppure una nuova matrice 3x3 aggiornata con la colonna in ingresso.

Logica interna: 'Alg3x3_Trasla' applica una semplice differenza vettoriale tramite 'mapcar'. 'Alg3x3_Vincolo' destruttura le componenti X e Y del punto e calcola i termini di secondo grado (x^2, xy, y^2) necessari per imporre il passaggio della conica per quel punto. Infine, 'Alg3x3_Sostituisci_Colonna' itera sulle righe della matrice sostituendo dinamicamente l'n-esimo elemento con i valori del nuovo vettore, operazione chiave per la risoluzione tramite regola di Cramer. (defun Alg3x3_Trasla (Pu1 Pu2) (mapcar '- Pu2 Pu1))

Alg4x4_Det

Firma: (Alg4x4_Det M)

Scopo: Calcolare il determinante di una matrice quadrata di ordine 4 (4x4) e risolvere un sistema di equazioni lineari a 4 incognite tramite l'applicazione sistematica della regola di Cramer.

Input: M (Matrice 4x4 formata da 4 liste di numeri reali). Per la funzione di risoluzione si aggiunge 'rhs' (vettore dei 4 termini noti).

Output: Rispettivamente: un numero reale corrispondente al valore del determinante, oppure una lista di 4 numeri reali rappresentante il vettore soluzione. In caso di matrice singolare (determinante prossimo allo zero), la risoluzione restituisce nil.

Logica interna: 'Alg4x4_Det' applica lo sviluppo di Laplace lungo la prima riga, assemblando i quattro minori complementari 3x3 tramite l'helper 'Alg4x4_Rimuovi_Colonna' e sommandoli con segni alterni (+ - + -). 'Alg4x4_Risolvi' implementa il Teorema di Cramer: valuta il determinante principale rispetto a una tolleranza di 1e-9, quindi calcola le quattro incognite sostituendo iterativamente le singole colonne con il vettore dei termini noti e calcolandone il rapporto rispetto al determinante base.

Alg4x4_Rimuovi_Colonna

Firma: (Alg4x4_Rimuovi_Colonna row c)

Scopo: Fornire un set di utilità per la manipolazione strutturale di righe e matrici di ordine 4 (4x4), permettendo l'estrazione rapida di minori complementari e la sostituzione dinamica di vettori colonna per la risoluzione dei sistemi.

Input: L'argomento varia. Per la rimozione: row (lista di 4 elementi), c (indice intero 0-3 della colonna da scartare). Per la sostituzione: M (matrice 4x4 formata da 4 liste), col (indice 0-3 della colonna), rhs (lista di 4 valori del vettore da iniettare).

Output: Rispettivamente: una nuova riga ridotta di 3 elementi (pronta per il calcolo dei determinanti 3x3), oppure una nuova matrice 4x4 aggiornata con la colonna sostituita.

Logica interna: 'Alg4x4_Rimuovi_Colonna' utilizza un costrutto 'cond' per assemblare una nuova lista omettendo l'elemento posizionale richiesto. 'Alg4x4_Sostituisci_Colonna' sfrutta 'mapcar' iterando in parallelo sulle righe della matrice e sui valori del vettore 'rhs', applicando un 'cond' interno per iniettare il nuovo valore all'indice corretto di ciascuna riga.

Alg4x4_Risolvi

Firma: (Alg4x4_Risolvi M rhs)

Scopo: Risolve un sistema lineare 4×4 Mx=rhs con la regola di Cramer, usato per trovare i 4 coefficienti omogenei di una conica passante per 4 punti.

Input: M: matrice 4×4 (lista di 4 liste di 4 elementi). rhs: vettore termine noto (lista di 4 valori).

Output: Lista di 4 valori (x1 x2 x3 x4) soluzione del sistema; nil se determinante nullo.

Logica interna: Calcola det=Alg4x4_Det(M). Per ogni colonna, sostituisce con rhs (Alg4x4_Sostituisci_Colonna), ricalcola il determinante e divide per det.

Alg4x4_Sostituisci_Colonna

Firma: (Alg4x4_Sostituisci_Colonna M col rhs)

Scopo: Sostituisce una colonna della matrice 4×4 con il vettore rhs fornito (passo preparatorio per la regola di Cramer).

Input: M: matrice 4×4. col: indice della colonna da sostituire (0-3). rhs: lista di 4 valori.

Output: Nuova matrice 4×4 con la colonna col sostituita da rhs.

Logica interna: Usa mapcar su M e rhs in parallelo: per ogni riga, sostituisce l'elemento nella posizione col con il valore corrispondente di rhs tramite un cond a 4 rami.

Alg_Tournament_5P_Centered

Firma: (Alg_Tournament_5P_Centered pts Ce_real conic_type)

Scopo: Algoritmo a torneo per trovare la conica centrata (con centro fisso dato) che minimizza l'errore su 5 punti dati, usando una strategia 4-su-5: prova tutte le combinazioni di 4 punti e sceglie la soluzione migliore.

Input: pts: lista di 5 punti. Center: centro fisso della conica.

Output: Lista (errore_min coefficienti_ottimali) della configurazione con errore minimo sul punto escluso; nil se nessuna soluzione trovata.

Logica interna: Itera su tutte le C(5,4)=5 combinazioni di 4 punti su 5. Per ciascuna, risolve il sistema 4×4 con Alg4x4_Risolvi, valuta l'errore sul 5° punto e aggiorna il minimo.

analyze-conic-geometry

Firma: (analyze-conic-geometry pts)

Scopo: Analizza la geometria di una conica dai suoi 6 coefficienti e ne estrae tipo, centro, semiassi, fuochi, eccentricità e orientamento, con restituzione in formato strutturato a chiave-valore.

Input: K: lista [A B C D E F] dei coefficienti della conica. scale: fattore di scala per la denormalizzazione.

Output: Lista strutturata (alist) con i campi: tipo, centro, semi-asse-a, semi-asse-b, eccentricità, angolo-asse, fuochi.

Logica interna: Delega il calcolo a Math_Extract_Geom_Data_From_Coeffs e ristruttura il risultato in una alist con chiavi simboliche.

C

Calcola_Lagrange

Firma: (Calcola_Lagrange x x1 y1 x2 y2 x3 y3)

Scopo: Interpola il valore y in un punto x usando il polinomio di Lagrange di 2° grado costruito su 3 nodi (x1,y1), (x2,y2), (x3,y3).

Input: x: punto in cui valutare il polinomio. x1,y1,x2,y2,x3,y3: coordinate dei tre nodi.

Output: Valore interpolato y(x); nil se due o più nodi hanno la stessa ascissa (collineari in x).

Logica interna: Verifica che le ascisse siano distinte, poi applica la formula: y=y1·L1(x)+y2·L2(x)+y3·L3(x) dove Li(x) sono i termini di Lagrange calcolati da Math_Lagrange_Term.

Calcola_Parabola

Firma: (Calcola_Parabola lisPu axis_pair)

Scopo: Calcola il valore y della parabola nel punto x usando l'interpolazione di Lagrange sui 3 punti di vincolo, restituendo le coordinate del punto sulla curva.

Input: x: ascissa del punto cercato. P1, P2, P3: tre punti di vincolo della parabola.

Output: Lista (x y 0.0) con le coordinate del punto sulla parabola interpolata.

Logica interna: Estrae le ascisse e ordinate dai 3 punti e chiama Calcola_Lagrange per ottenere l'ordinata y corrispondente ad x.

Calc_Ipe_Asym_Pts

Firma: (Calc_Ipe_Asym_Pts C Vx Vy len h)

Scopo: Determinare analiticamente le coordinate WCS dei punti strutturali per il tracciamento esatto di un'iperbole (punti di controllo dei rami NURBS/Spline, vertici limite per gli asintoti incrociati e confini per gli assi di simmetria maggiore e minore).

Input: C (Coordinate WCS del Centro locale), Vx e Vy (Vettori direzionali degli assi). Parametri dimensionali scalari: dcv (distanza dal centro al vertice), len (estensione massima del ramo/asse), h (semi-apertura asintotica), dir (fattore di direzione 1.0 o -1.0).

Output: Matrici di punti WCS 3D progettate per la generazione diretta delle entità grafiche ausiliarie e principali dell'iperbole.

Logica interna: Costruisce la "gabbia" asintotica e i vertici di controllo fornendo le misure in coordinate locali 2D rispetto al centro C. Nel caso dei rami, applica il moltiplicatore 'dir' per specchiare simmetricamente la curva. Delega infine al motore 'Get_lisPt_On_Axes' l'ancoraggio e la rototraslazione della geometria nel sistema di coordinate globale. (defun Calc_Ipe_Branch_CVs (C Vx Vy dcv len h dir) (Get_lisPt_On_Axes C Vx Vy (list (list (* dir len) (- h)) (list (* dir dcv) 0.0) (list (* dir len) h))))

Calc_Ipe_Axes_Pts

Firma: (Calc_Ipe_Axes_Pts C Vx Vy len)

Scopo: Determinare analiticamente le coordinate WCS dei punti strutturali per il tracciamento esatto di un'iperbole (punti di controllo dei rami NURBS/Spline, vertici limite per gli asintoti incrociati e confini per gli assi di simmetria maggiore e minore).

Input: C (Coordinate WCS del Centro locale), Vx e Vy (Vettori direzionali degli assi). Parametri dimensionali scalari: dcv (distanza dal centro al vertice), len (estensione massima del ramo/asse), h (semi-apertura asintotica), dir (fattore di direzione 1.0 o -1.0).

Output: Matrici di punti WCS 3D progettate per la generazione diretta delle entità grafiche ausiliarie e principali dell'iperbole.

Logica interna: Costruisce la "gabbia" asintotica e i vertici di controllo fornendo le misure in coordinate locali 2D rispetto al centro C. Nel caso dei rami, applica il moltiplicatore 'dir' per specchiare simmetricamente la curva. Delega infine al motore 'Get_lisPt_On_Axes' l'ancoraggio e la rototraslazione della geometria nel sistema di coordinate globale. (defun Calc_Ipe_Branch_CVs (C Vx Vy dcv len h dir) (Get_lisPt_On_Axes C Vx Vy (list (list (* dir len) (- h)) (list (* dir dcv) 0.0) (list (* dir len) h)))) (defun Calc_Ipe_Asym_Pts (C Vx Vy len h)

(Get_lisPt_On_Axes C Vx Vy (list (list len h) (list (- len) (- h)) (list len (- h)) (list (- len) h))))

Calc_Ipe_Branch_CVs

Firma: (Calc_Ipe_Branch_CVs C Vx Vy dcv len h dir)

Scopo: Determinare analiticamente le coordinate WCS dei punti strutturali per il tracciamento esatto di un'iperbole (punti di controllo dei rami NURBS/Spline, vertici limite per gli asintoti incrociati e confini per gli assi di simmetria maggiore e minore).

Input: C (Coordinate WCS del Centro locale), Vx e Vy (Vettori direzionali degli assi). Parametri dimensionali scalari: dcv (distanza dal centro al vertice), len (estensione massima del ramo/asse), h (semi-apertura asintotica), dir (fattore di direzione 1.0 o -1.0).

Output: Matrici di punti WCS 3D progettate per la generazione diretta delle entità grafiche ausiliarie e principali dell'iperbole.

Logica interna: Costruisce la "gabbia" asintotica e i vertici di controllo fornendo le misure in coordinate locali 2D rispetto al centro C. Nel caso dei rami, applica il moltiplicatore 'dir' per specchiare simmetricamente la curva. Delega infine al motore 'Get_lisPt_On_Axes' l'ancoraggio e la rototraslazione della geometria nel sistema di coordinate globale.

Calc_Par_Axis_Pts

Firma: (Calc_Par_Axis_Pts V Ax No dfo len)

Scopo: Calcolare le coordinate spaziali WCS dei punti chiave necessari per la costruzione geometrica e la rappresentazione visiva di una parabola (punti di controllo della curva Spline, estremi della direttrice e limiti dell'asse di simmetria).

Input: V (Coordinate WCS del Vertice), Ax e No (Vettori direzionali normalizzati per asse X e asse Y locali). Parametri dimensionali scalari: len (lunghezza o estensione del ramo), h (semi-apertura trasversale), dfo (distanza focale).

Output: Liste separate di coordinate WCS 3D formattate e pronte per essere passate ai comandi di disegno nativi di AutoCAD.

Logica interna: Sfrutta la funzione 'Get_lisPt_On_Axes' come motore di trasformazione. Inietta le distanze e le semi-aperture come coordinate locali 2D (relative al vertice V) e le mappa vettorialmente, restituendo istantaneamente la geometria strutturale della parabola nello spazio tridimensionale assoluto WCS. (defun Calc_Par_Branch_CVs (V Ax No len h) (Get_lisPt_On_Axes V Ax No (list (list len (- h)) (list (- len) 0.0) (list len h)))) (defun Calc_Par_Directrix_Pts (V Ax No dfo h) (Get_lisPt_On_Axes V Ax No (list (list (- dfo) (- h)) (list (- dfo) h))))

Calc_Par_Branch_CVs

Firma: (Calc_Par_Branch_CVs V Ax No len h)

Scopo: Calcolare le coordinate spaziali WCS dei punti chiave necessari per la costruzione geometrica e la rappresentazione visiva di una parabola (punti di controllo della curva Spline, estremi della direttrice e limiti dell'asse di simmetria).

Input: V (Coordinate WCS del Vertice), Ax e No (Vettori direzionali normalizzati per asse X e asse Y locali). Parametri dimensionali scalari: len (lunghezza o estensione del ramo), h (semi-apertura trasversale), dfo (distanza focale).

Output: Liste separate di coordinate WCS 3D formattate e pronte per essere passate ai comandi di disegno nativi di AutoCAD.

Logica interna: Sfrutta la funzione 'Get_lisPt_On_Axes' come motore di trasformazione. Inietta le distanze e le semi-aperture come coordinate locali 2D (relative al vertice V) e le mappa vettorialmente, restituendo istantaneamente la geometria strutturale della parabola nello spazio tridimensionale assoluto WCS.

Calc_Par_Directrix_Pts

Firma: (Calc_Par_Directrix_Pts V Ax No dfo h)

Scopo: Calcolare le coordinate spaziali WCS dei punti chiave necessari per la costruzione geometrica e la rappresentazione visiva di una parabola (punti di controllo della curva Spline, estremi della direttrice e limiti dell'asse di simmetria).

Input: V (Coordinate WCS del Vertice), Ax e No (Vettori direzionali normalizzati per asse X e asse Y locali).

Parametri dimensionali scalari: len (lunghezza o estensione del ramo), h (semi-apertura trasversale), dfo (distanza focale).

Output: Liste separate di coordinate WCS 3D formattate e pronte per essere passate ai comandi di disegno nativi di AutoCAD.

Logica interna: Sfrutta la funzione 'Get_lisPt_On_Axes' come motore di trasformazione. Inietta le distanze e le semi-aperture come coordinate locali 2D (relative al vertice V) e le mappa vettorialmente, restituendo istantaneamente la geometria strutturale della parabola nello spazio tridimensionale assoluto WCS. (defun Calc_Par_Branch_CVs (V Ax No len h) (Get_lisPt_On_Axes V Ax No (list (list len (- h)) (list (- len) 0.0) (list len h))))

CON1P_Geom_Conic_2F_Engine

Firma: (CON1P_Geom_Conic_2F_Engine pt F1 F2 C c_dist V_dir V_perp type opt_m opt_e last_ents)

Scopo: Motore Bifocale: 2 Fuochi + 1 Punto.

Input: pt, F1, F2, C, c_dist, V_dir, V_perp, type, opt_m, opt_e, last_ents.

Output: Lista strutturata (TipoRichiesto . NuoveEntità).

Logica interna: Sfrutta la proprietà della somma (Ellisse) o differenza (Iperbole) costante delle distanze focali dal punto per derivare i semiassi 'a' e 'b'.

CON1P_Geom_Conic_2V_Engine

Firma: (CON1P_Geom_Conic_2V_Engine pt V1 V2 C a v_dir v_perp opt_m opt_e last_ents)

Scopo: Motore di Calcolo: 2 Vertici + 1 Punto.

Input: pt, V1, V2, C, a, v_dir, v_perp, opt_m, opt_e, last_ents.

Output: Lista strutturata (TipoTrovato . NuoveEntità).

Logica interna: Utilizza l'Helper Setup per la pulizia e le proiezioni. Deriva 'b' analiticamente invertendo l'equazione canonica dell'ellisse o dell'iperbole a seconda che il punto sia interno o esterno ad 'a'.

CON1P_Helper_Base_Vectors

Firma: (CON1P_Helper_Base_Vectors P1 P2)

Scopo: Centralizzare la costruzione dell'ossatura vettoriale base a partire da due punti forti (Vertici o Fuochi).

Input: P1, P2 (Coordinate 3D dei due punti).

Output: Lista formattata (Centro DistanzaTotale SemiDistanza V_dir V_perp) oppure nil se i punti coincidono.

Logica interna: Calcola il punto medio, le distanze e i versori ortonormali direttori, bloccando l'esecuzione e prevenendo errori matematici se l'utente clicca due volte lo stesso punto.

CON1P_Helper_Engine_Setup

Firma: (CON1P_Helper_Engine_Setup pt C v_dir v_perp opt_m last_ents)

Scopo: Centralizzare la preparazione della memoria e dello spazio locale per i motori geometrici 1P.

Input: pt (Punto click), C (Centro), v_dir (Vettore asse maggiore), v_perp (Vettore asse minore), opt_m (Modo), last_ents (Entità da cancellare).

Output: Lista formattata (e_marker P_u P_n) contenente il marker di memoria e le proiezioni locali assolute.

Logica interna: Spazza via le vecchie geometrie se in modalità "Sostituisci", inietta il nuovo marker nel database, disegna il punto per feedback visivo e scompone il click nei delta X e Y locali.

CON1P_Helper_UI_Prompt

Firma: (CON1P_Helper_UI_Prompt cmd_type state_list)

Scopo: Generatore universale di prompt testuali interattivi per i loop LUSNAP dei comandi 1P.

Input: cmd_type (Simbolo: 'CONVP, 'CONFP_MAIN, 'CONFP_SUB), state_list (Lista: opt_g, opt_m, opt_e, choice).

Output: Stringa formattata da passare direttamente a getpoint o getkword.

Logica interna: Inizializza i comandi testuali validi (initget) in base al contesto e concatena dinamicamente le variabili di stato per mostrare una riga di comando pulita.

CON1P_Helper_UI_State

Firma: (CON1P_Helper_UI_State input_str opt_m opt_e)

Scopo: "Potenziometro" universale per l'interfaccia dei comandi 1P.

Input: input_str (Comando utente: "Estendi", "Accorcia", "Modo"), opt_m (Stato attuale Modo), opt_e (Stato attuale Estensione).

Output: Lista formattata (nuovo_opt_m nuovo_opt_e trigger_redraw). Il trigger_redraw à T se la curva necessita di ricalcolo istantaneo.

Logica interna: Gestisce gli incrementi/decrementi di estensione (+0.5/-0.5) bloccando il limite minimo a 1.0, e innesca lo switch della stringa di modalità.

c:CON2F1P

Scopo: Comando principale interattivo per la generazione dinamica di coniche partendo da 2 Fuochi e 1 Punto.

Input: Nessuno (Interazione utente via console e mouse).

Output: Nessuno (Side effect: generazione entità su DWG).

Logica interna: Inizializza i vettori base. Attraverso due loop annidati (scelta modalità -> aggancio punto), gestisce Ellissi e Iperboli (inclusa la simulazione fittizia dell'Iperbole Equilatera per agganciarla all'estensione).

c:CON2V1P

Scopo: Comando principale interattivo per la generazione dinamica di coniche partendo da 2 Vertici e 1 Punto.

Input: Nessuno (Interazione utente via console e mouse).

Output: Nessuno (Side effect: generazione entità su DWG).

Logica interna: Inizializza i vettori base e innesca un ciclo while che sfrutta l'UI Prompt Helper e l'UI State Helper per aggiornare dinamicamente l'impalcatura e la curva, rispondendo agli hotkey.

c:con3pc

Scopo: Funzione Wrapper ed entry-point per l'avvio dell'applicazione interattiva di tracciamento "Conica per 3 Punti + Centro".

Input: Nessuno (Input richiesto interattivamente a schermo tramite clic).

Output: Nessuno (Ritorna silenziosamente al prompt dei comandi tramite `princ`).

Logica interna: Applica il protocollo standard di blindatura `Init_Conic_App` per salvare le variabili di sistema (OSMODE, PDMODE) e interrogare l'utente sui 3 punti base WCS. Se l'acquisizione ha successo, disegna il poligono di base (triangolo), spegne lo snap per evitare interferenze e delega l'intero ciclo interattivo al motore logico `Geom_3Pts_Master_Engine`. Alla conclusione del loop utente, invoca `Restore_Conic_App` per sanificare l'ambiente AutoCAD restituendo le configurazioni iniziali.

c:CON4P

Scopo: Comando principale per l'Ambiente Protetto Conica 4P. Gestisce l'intero ciclo: Selezione -> Analisi Topologica -> Rendering Interattivo per TUTTE le figure e configurazioni geometriche.

Input: Nessuno (Interattivo tramite console CAD e click del mouse).

Output: Nessuno (Side-effect su DWG tramite generazione di entità di costruzione e coniche finite).

Logica interna: Smista l'esecuzione verso il motore specializzato più adatto: - Ramo 2A (Rettangoli): Calcolo Ortogonale e Proiezione LUSNAP su assi. - Ramo 2B (Parallelogrammi): Generatore Discreto Adattivo (Radar Analitico sui nodi critici). - Ramo 3 (Ortocentrico): Estrattore Euleriano su Cerchio. - Ramo 4 (Generico e Trapezi): Partizionatore Matrioska su curve VLA. Genera Mappe dati globali (meta_map) e isola le soluzioni nobili (CEPC, Equilatera, Parabole) rendendole persistenti nella Fase 1 per l'accesso diretto tramite il sottomenu 'Speciali' in Fase 2. Gestisce il flusso Modo (Aggiungi/Sostituisci).

CON4P_Capture_Locus_Entities

Firma: (CON4P_Capture_Locus_Entities ent_start lay_target)

Scopo: Intercettatore di entità sicuro e isolato per l'ambiente CON4P.

Input: ent_start (ENAME di partenza), lay_target (Nome stringa del layer da filtrare).

Output: Lista di VLA-Objects generati dopo ent_start sul layer specificato.

Logica interna: Filtra rigorosamente per layer e per tipologia geometrica, castando i risultati in oggetti COM (VLA).

CON4P_Core_Matrioska_Engine

Firma: (CON4P_Core_Matrioska_Engine st en stp)

Scopo: Motore matematico centralizzato per il calcolo della distribuzione frattale dei parametri.

Input: st (Inizio T), en (Fine T), stp (Step base lineare).

Output: Lista di parametri pulita e ordinata.

Logica interna: Delega i cicli iterativi all'helper CON4P_Core_Matrioska_Loop, interfacciandosi solo con le chiamate di avvio e fine ramo. Il primo elemento (nil) garantisce l'incastro auto-calcolandosi.

CON4P_Core_Matrioska_Loop

Firma: (CON4P_Core_Matrioska_Loop base_val dir k_start k_step divisors_list L_init first_divisor cfg_iter param_list)

Scopo: Helper per eseguire il ciclo iterativo frattale (Matrioska) sia in avvicinamento che in allontanamento.

Input: base_val (st o en), dir (1 o -1 per l'offset), k_start (0 o cfg_len), k_step (1 o -1), divisors_list, L_init, first_divisor, cfg_iter, param_list.

Output: Lista aggiornata param_list con i nuovi punti calcolati.

Logica interna: Fonde i due cicli speculari usando moltiplicatori di direzione per riutilizzare la stessa identica matematica per l'espansione e la compressione.

CON4P_Draw_ConEll_Special

Firma: (CON4P_Draw_ConEll_Special pts_ord meta_map)

Scopo: Estrarre e tracciare l'Ellisse Pseudo-Circolare (CEPC) recuperando il centro ottimale marchiato durante la scansione Matrioska del modulo CON4P.

Input: pts_ord (4 vertici), meta_map (Mappa dei centri calcolati durante il radar topologico).

Output: Lista di ENAME disegnati (Curva + anatomia completa) o nil se il metadato non à presente.

Logica interna: Scansiona la 'meta_map' alla ricerca dell'entry specifica marchiata "CEPC" (generata per i rami Convesso Generico e Trapezio). Se trovata, scompone i parametri geometrici (Center, V_dir, semiassi) e invia il tutto al motore di disegno 'Draw_ConEll_CeVe_CoVe' per la restituzione grafica completa di assi e fuochi.

CON4P_Draw_ConIpe_Equil

Firma: (CON4P_Draw_ConIpe_Equil pts_ord)

Scopo: Analizzare una quaterna di punti e, se topologicamente idonea, calcolare e tracciare istantaneamente l'unica iperbole equilatera algebrica passante per essi.

Input: pts_ord (Lista delle coordinate WCS dei 4 punti base).

Output: Lista di entità ENAME disegnate (Curva + anatomia completa), oppure nil se la configurazione non lo consente.

Logica interna: 1. Gate Keeper Topologico: Analizza la forma e procede solo per topologie asimmetriche o trapezi (esclude concavi ortocentrici e parallelogrammi simmetrici). 2. Solutore Algebrico: Chiama il risolutore "one-shot" per estrarre Centro, Vettore Direzione e semiassi (a=b). 3. Traduttore Geometrico: Converte i parametri algebrici in Vertice (V1) e Fuoco (F1). 4. Driver Grafico: Invia i dati al wrapper unificato 'Draw_ConIpe_CeVeFu_Compl' per la generazione a video.

CON4P_Draw_Orthogonal_Locus

Firma: (CON4P_Draw_Orthogonal_Locus pts_ord mode Cg)

Scopo: Generare i binari d'asse (luogo geometrico) per Quadrati e Rettangoli con settorializzazione cromatica.

Input: pts_ord (4 vertici), mode ("FOCI" o "VERTICES"), Cg (Baricentro).

Output: Lista di ENAME dei segmenti generati.

Logica interna: Calcola i punti medi. Suddivide l'asse in un segmento interno (Iperbole) e segmenti esterni (Ellisse), assegnando il colore corrispondente definito nelle configurazioni.

CON4P_Draw_Points_With_Colors

Firma: (CON4P_Draw_Points_With_Colors pt_lists colors lay)

Scopo: Disegnare una o più nuvole di punti discrete a schermo, colorandole dinamicamente.

Input: pt_lists (Liste di punti), colors (Lista codici colore CECOLOR), lay (Layer target).

Output: Lista piana degli ENAME generati.

Logica interna: Sostituisce temporaneamente CECOLOR disegnando array su layer specificato.

CON4P_Elabora_Ramo_Partizionato

Firma: (CON4P_Elabora_Ramo_Partizionato obj ramo scan_mod pts_ord mode D_ref)

Scopo: Iteratore di partizionamento logico. Scorre i segmenti sicuri di un ramo spezzato e innesca il Micro-Scan frattale.

Input: obj (Curva VLA), ramo (Lista segmenti), scan_mod (Densità base), pts_ord (Vertici), mode ("FOCI" o "VERTICES"), D_ref (Scala).

Output: Mappe calcolate come side-effect globale.

Logica interna: Calcola lo step adattivo per densità estrema ai bordi della matrice leggendo il moltiplicatore dalle configurazioni globali.

CON4P_Evaluate_LUSNAP_Target

Firma: (CON4P_Evaluate_LUSNAP_Target p5 par_data_map meta_map pts_ord mode D_ref)

Scopo: Elaborare la priorità di aggancio tra i nodi parabolici discreti e la mappa densa del Micro-Scan frattale.

Input: p5, par_data_map, meta_map, pts_ord, mode, D_ref.

Output: Lista a 2 elementi: (Distanza_Minima Geometria_Vincente).

Logica interna: Identifica la parabola e l'innesco più vicini confrontandone le euclidee assolute.

CON4P_Execute_Center_Conic

Firma: (CON4P_Execute_Center_Conic action_data opt_m opt_e last_run_ents P1 P2 P3 P4 pts_ord Cg)

Scopo: Elaborare e disegnare una conica vincolata a un punto target trattato come centro puro.

Input: action_data (Dati LUSNAP), opt_m (Modo), opt_e (Estensione), last_run_ents, P1..P4, pts_ord, Cg.

Output: Restituisce la nuova lista di ename generati.

Logica interna: Calcola i parametri normalizzati della conica centrata e invia al motore focale.

CON4P_Execute_Conic_Action

Firma: (CON4P_Execute_Conic_Action geo_data mode P1 P2 P3 P4 Cg opt_m opt_e last_run_ents)

Scopo: Smistare al CAD finale per il disegno delle coniche scelte.

Input: geo_data (Struttura dati target), mode, base_pts, opt_m, opt_e, last_run_ents.

Output: ENAME della conica tracciata.

Logica interna: Router passivo per i motori grafici.

CON4P_Geom_4P_Master_Setup

Firma: (CON4P_Geom_4P_Master_Setup titolo snap_mode)

Scopo: Inizializzazione sicura e standardizzata dell'ambiente di calcolo per tutti i comandi della suite.

Input: titolo (Stringa prompt), snap_mode (Valore OSMODE).

Output: Lista contenente tutte le variabili di contesto, oppure nil se l'input fallisce.

Logica interna: Chiama il selezionatore unificato Init_Conic_App, scatena l'analizzatore topologico e attiva il Framework grafico.

CON4P_Geom_Calc_Micro_Param_List

Firma: (CON4P_Geom_Calc_Micro_Param_List obj p_center step_span ref_len phys_step)

Scopo: Genera un array densissimo di parametri T attorno a un nodo candidato su curva VLA.

Input: obj (Curva VLA), p_center (Parametro target), step_span (Margine parametrico), ref_len (Scala), phys_step (Risoluzione).

Output: Lista densa di parametri T.

Logica interna: Calcola limiti fisici isolando le estremità della curva ed applica tolleranza.

CON4P_Geom_Classify_And_Normalize

Firma: (CON4P_Geom_Classify_And_Normalize K_full g_data D_ref)

Scopo: Helper. Isola la logica di classificazione e normalizzazione geometrica della conica.

Input: K_full (coeff. matriciali), g_data (dati estratti), D_ref (scala).

Output: Lista formattata: (f_type V_dir a b).

Logica interna: Calcola il determinante per distinguere Ellissi da Iperboli e normalizza gli assi al volo.

CON4P_Geom_Conic_4Pv_Engine_Local

Firma: (CON4P_Geom_Conic_4Pv_Engine_Local P1 P2 P3 P4 Cg geo_packet opt_m opt_e last_ents)

Scopo: Motore grafico specializzato per il rendering delle coniche del Ramo 2A (Rettangolo/Quadrato).

Input: P1..P4 (Vertici), Cg (Centro), geo_packet (Dati conica), opt_m (Modo), opt_e (Estensione), last_ents.

Output: Lista degli ENAME generati.

Logica interna: Distingue tra Ellisse e Iperbole. Calcola dinamicamente l'estensione ottica per le iperboli (evitando rami troppo corti rispetto ai vertici) e lancia la routine di disegno a video.

CON4P_Geom_Draw_Locus_And_Capture

Firma: (CON4P_Geom_Draw_Locus_And_Capture pts_raw classification)

Scopo: Disegna fisicamente nell'ambiente CAD il luogo geometrico dei centri corrispondente alla topologia.

Input: pts_raw (Lista 4 punti), classification (Lista strutturata tipo/pts_ord).

Output: Lista (vla-objects) delle entità grafiche appena create sul layer dedicato.

Logica interna: Mappa in modo rigido i simboli restituiti dal classificatore verso le routine di disegno dedicate, catturando il risultato.

CON4P_Geom_Draw_Target_Parabola

Firma: (CON4P_Geom_Draw_Target_Parabola P1 P2 P3 P4 target_pt mode)

Scopo: Individuare e tracciare la parabola bersaglio prevenendo instabilità matematiche da scala.

Input: P1..P4 (Base), target_pt (Target mouse), mode.

Output: Lista di entità disegnate.

Logica interna: Forza la normalizzazione spaziale prima di interpolare su matrici quadrate, estraendo equazioni stabili tramite scarti dei minimi scostamenti.

CON4P_Geom_Eval_Micro_Param

Firma: (CON4P_Geom_Eval_Micro_Param obj param pts_ord mode ref_len p5)

Scopo: Valutatore geometrico locale per raffinamento.

Input: obj, param, pts_ord, mode, ref_len, p5.

Output: Lista (best_distance best_geometry) per il parametro t.

Logica interna: Calcola la conica centrata al parametro e ne estrae la distanza focale o i vertici.

CON4P_Geom_Find_Nearest_Data

Firma: (CON4P_Geom_Find_Nearest_Data pt map_list)

Scopo: LUSNAP discreto sui metadati (Parabole/Nodi). Ricerca euclidea diretta.

Input: pt (Coordinate 3D mouse), map_list (Lista dei pacchetti map-data).

Output: Elemento (item) geometrico più vicino.

Logica interna: Iterazione spaziale pura sui centroidi dell'array di mappe.

CON4P_Geom_Format_Trapezoid_Locus

Firma: (CON4P_Geom_Format_Trapezoid_Locus P1 P2 P3 P4 old_vla_list)

Scopo: Ricostruire e segmentare in modo esatto la retta dei centri nei quadrangoli di tipo Trapezio.

Input: P1..P4 (Vertici), old_vla_list (Liste linee grezze tracciate).

Output: Lista composta da: (NuovaListaVLA NuovaListaENAME).

Logica interna: Distrugge il luogo generico e ritraccia una catena di linee contigue estese per la scansione lineare dei centri.

CON4P_Geom_Get_Projected_Center

Firma: (CON4P_Geom_Get_Projected_Center pt_ucs vla_list extend)

Scopo: Ricerca della distanza ortogonale minima su geometrie continue.

Input: pt_ucs (Punto click), vla_list (Liste oggetti base), extend (Flag estensione curve VLA).

Output: Punto WCS più vicino.

Logica interna: Proiezione atomica di vlax-curve-getClosestPointTo catturando in sicurezza le eccezioni.

CON4P_Geom_Master_Exit_Handler

Firma: (CON4P_Geom_Master_Exit_Handler opt_gc opt_gr guide_ents_c guide_ents_r group_name)

Scopo: Pulizia dell'ambiente di lavoro al termine di un comando Master.

Input: opt_gc, opt_gr (Stato visibilità guide), guide_ents_c, guide_ents_r (Liste ENAME da eliminare), group_name (Nome modulo).

Output: Nessuno (Side-effect console e DWG).

Logica interna: Se la visibilità era in stato "OFF", elimina fisicamente i vettori di supporto dal database del disegno per non lasciare tracce.

CON4P_Geom_MicroScan

Scopo: Motore di raffinamento finale LUSNAP ibrido (Supporta mappe VLA e Mappe Discrete).

Input: p5 (Click), meta_list (Meta-mappa), pts_ord, mode, ref_len.

Output: Record assoluto (best_d best_geo).

Logica interna: Valuta la natura del luogo. Se l'oggetto à nullo (Ramo 2B - Parallelogramma), recupera gli angoli theta dei due punti più vicini e lancia un'interpolazione matematica pura (oversampling angolare locale) calcolando le coniche al volo. Altrimenti, esegue il Micro-Scan fisico esplorando i parametri parametrici delle Spline VLA di AutoCAD.

CON4P_Geom_Scan_Orthocentric_Circle

Firma: (CON4P_Geom_Scan_Orthocentric_Circle pts_ord locus_vla_list scan_mod mode D_ref)

Scopo: Scansione dell'anello di Eulero per la ricerca dei centri delle coniche degeneri, addensando il campionamento in prossimità dei punti critici.

Input: pts_ord (Punti base del quadrilatero), locus_vla_list (Lista delle curve del luogo, da cui estrarre il cerchio), scan_mod (Modificatore di risoluzione), mode, D_ref.

Output: Lista complessa contenente punti, mappe e metadati delle coniche valide trovate.

Logica interna: Identifica i 3 punti di intersezione delle diagonali, calcola i loro angoli rispetto al centro del cerchio e seziona la circonferenza in 3 archi. Su ogni arco applica il motore Matrioska per generare una distribuzione di angoli frattale continua, filtrando le cuciture ai bordi.

CON4P_Geom_Scan_Parallelogram_Locus

Scopo: Generatore discreto adattivo per calcolare analiticamente il luogo dei fuochi e dei vertici in Parallelogrammi e Rombi.

Input: pts_ord (4 vertici), mode (Stringa "FOCI" o "VERTICES"), D_ref (Fattore di scala per normalizzazione).

Output: Lista formattata compatibile con il Master (pts_1, pts_2, nil, pts_4, pts_map, count, meta_map).

Logica interna: Risolve l'equazione della conica centrata nel baricentro. Estrae in testata i parametri di configurazione densità (eliminando i magic numbers). Traccia le singolarità a +/- PI/2 e i nodi critici per convergere nel centro assoluto Cg tramite un compressore logaritmico modulabile.

CON4P_Geom_Scan_VLA_Centers

Firma: (CON4P_Geom_Scan_VLA_Centers pts_ord locus_vla_list scan_mod mode)

Scopo: Radar topologico per le coniche centrali. Scansiona i luoghi geometrici VLA per generare le mappe di fuochi e vertici applicando il partizionamento frattale. Isola ed estrae i dati dell'Ellisse Pseudo-Circolare (CEPC).

Input: pts_ord (Lista 4 vertici base), locus_vla_list (Lista curve VLA del luogo), scan_mod (Risoluzione), mode ("FOCI" o "VERTICES").

Output: Lista di 8 elementi: (pts_1 pts_2 nil pts_4 pts_map count meta_map cepc_packet) formattata per LUSNAP e menu Speciali.

Logica interna: Identifica la topologia analizzando i segmenti VLA. Per Quadrangoli Convessi e Trapezi, lancia una ricerca a griglia iterata per individuare il punto di massima circolarità (CEPC) e lo inietta come nodo di discontinuità nel motore Matrioska. Il partizionatore suddivide le curve e delega l'addensamento asintotico agli helper. A fine ciclo, calcola la matrice della CEPC, denormalizza i semiassi e accoda l'ottavo pacchetto (cepc_packet) alla mega-lista finale per renderlo persistente nel Master.

CON4P_Geom_Search_Grid_Ell_Param

Scopo: Trovare il parametro sulla curva dove l'ellisse passante per i 4 punti raggiunge la massima circolarità (Ellisse Pseudo-Circolare CEPC).

Input: curve_obj (Entità VLA), st_param/en_param (Limiti parametrici), pts_ord (4 vertici base), D_ref (Fattore scala), tol (Tolleranza).

Output: Valore reale (Float) che rappresenta il parametro ottimale t lungo la curva VLA.

Logica interna: Motore di ricerca iterativo a griglia con zoom progressivo. Suddivide il dominio in segmenti (CFG_DIVS), valuta il rapporto dei semiassi dell'ellisse per ogni punto e isola il valore massimo (più vicino a 1). Restringe poi il range di scansione attorno al candidato migliore, ripetendo l'operazione per un numero definito di cicli (CFG_ITERS) per convergere analiticamente verso il centro esatto.

CON4P_Geom_Solve_Unique_Equilateral_Ipe

Firma: (CON4P_Geom_Solve_Unique_Equilateral_Ipe pts_ord)

Scopo: Calcolare la "carta d'identità" geometrica dell'unica Iperbole Equilatera passante per i 4 vertici base del quadrangolo.

Input: pts_ord (Lista contenente le coordinate WCS dei 4 punti base).

Output: Pacchetto dati LUSNAP (Centro 'IPERBOLE Centro V_dir a b) pronto per il rendering, o nil in caso di degenerazione.

Logica interna: Esegue la normalizzazione spaziale dei punti rispetto al baricentro per stabilità numerica. Genera le due coniche base moltiplicando le rette dei lati. Risolve la condizione A+C=0 per trovare il parametro lambda del fascio. Ricava l'equazione finale, denormalizza i semiassi e i centri tramite D_ref, valida il determinante negativo e struttura l'output per il motore grafico unificato.

CON4P_Geom_Update_MetaMap

Firma: (CON4P_Geom_Update_MetaMap pts_list vla_obj p_val s_val map_entry current_map)

Scopo: Popola la Meta-Mappa globale iterando su una lista di punti e associando a ciascuno i riferimenti VLA/Metadati.

Input: pts_list, vla_obj, p_val, s_val, map_entry, current_map.

Output: Restituisce la meta_map aggiornata.

Logica interna: Aggiornamento O(1) tramite 'cons'.

CON4P_Get_Config_String

Firma: (CON4P_Get_Config_String tipo)

Scopo: Formattazione testuale rapida della topologia per i report a video.

Input: tipo (Simbolo classificazione).

Output: Stringa compatta (CC, TR, CV).

Logica interna: Mappatura tramite WCMATCH.

CON4P_Get_Global_Config

Scopo: Centralizzare e fornire i parametri numerici e la palette cromatica vitali all'intero motore geometrico della suite CON4P.

Input: Nessuno.

Output: Association List (A-List) contenente parametri di tolleranza, fattori di scala, densità di scansione e mappatura dei colori LUSNAP.

Logica interna: Raccoglie in un'unica struttura dati i magic numbers del codice per facilitarne la manutenzione globale. Definisce tolleranze, moltiplicatori di estensione asintotica per le topologie (Trapezi, Convessi, Concavi), i setup del partizionatore Matrioska e associa i codici colore ACI standard di AutoCAD per differenziare visivamente fuochi e vertici durante il rendering interattivo a schermo.

CON4P_Get_Matrioska_Angles

Firma: (CON4P_Get_Matrioska_Angles st en stp)

Scopo: Interfaccia per il prelievo puramente matematico di parametri (angoli).

Input: st (Angolo di inizio), en (Angolo di fine), stp (Risoluzione).

Output: Lista di angoli logaritmicamente addensati agli estremi. Bypassa l'uso di entità VLA.

Logica interna: Agisce come semplice wrapper trasparente verso CON4P_Core_Matrioska_Engine.

CON4P_Helper_Auto_Matrioska

Firma: (CON4P_Helper_Auto_Matrioska obj st en div_mod pts_ord mode D_ref)

Scopo: Calcolare dinamicamente lo step parametrico di crociera e innescare la scansione frattale (Matrioska) sui segmenti VLA continui di una curva.

Input: obj (Curva VLA), st (Parametro Start), en (Parametro End), div_mod (Modificatore densità di scansione), pts_ord (Lista vertici), mode ("FOCI" o "VERTICES"), D_ref (Fattore scala).

Output: Nessun output diretto (agisce tramite side-effect popolando le mappe LUSNAP globali).

Logica interna: Ricava la lunghezza matematica del tratto parametrico, deduce lo step base per il campionamento lineare usando div_mod come partizionatore locale e invia i dati al processore atomico di segmento per generare l'addensamento asintotico vero e proprio.

CON4P_Helper_Pack_And_Draw

Firma: (CON4P_Helper_Pack_And_Draw scan_res par_data colors lay)

Scopo: Accorpa l'estrazione delle mappe di scansione e il disegno dei punti notevoli a schermo.

Input: scan_res (Dati grezzi scanner), par_data (Mappa parabola o nil), colors (Palette), lay (Layer target).

Output: Lista compatta (guide_ents pts_map meta_map par_data).

Logica interna: Riduce il boilerplate nel Master spacchettando la tupla dello scanner e instradando subito i cluster di punti al motore grafico.

CON4P_Helper_Solve_Normalized_Conic

Firma: (CON4P_Helper_Solve_Normalized_Conic center pts_ord D_ref solver_type)

Scopo: Centralizzare e ottimizzare la normalizzazione spaziale preventiva e la risoluzione matriciale della conica passante per i vertici dato un centro specifico.

Input: center (Coordinate 3D del centro WCS candidato), pts_ord (Lista vertici WCS), D_ref (Fattore scala), solver_type (Simbolo 'CENTERED per matrici generali o 'EQUILATERAL per approccio Euleriano).

Output: Lista formattata (f_type V_dir a b) pronta per i generatori grafici, oppure nil se la matrice degenera.

Logica interna: Effettua un downscaling (D_ref) per portare le coordinate in range numerici sicuri (evitando overflow ed esplosioni sui determinanti polinomiali). A seconda del solver_type instradato, estrae la matrice e la classifica restituendo i semiassi denormalizzati in uscita.

CON4P_Helper_UI_Prompt

Firma: (CON4P_Helper_UI_Prompt menu_type tipo state_list)

Scopo: Centralizzare e generare dinamicamente le stringhe di prompt e i comandi testuali (initget) per i cicli interattivi del comando Master.

Input: menu_type (Contesto: 'ORTHO, 'SPECIALI, T), tipo (Topologia), state_list (Lista variabili: opt_gc, opt_gf, opt_gv, opt_e, opt_m, ecc.).

Output: Stringa di testo formattata pronta per le funzioni di acquisizione input del CAD (getpoint o getkword).

Logica interna: Mappa le variabili di stato per la lettura. Tramite un costrutto cond, valuta il tipo di menu richiesto e imposta le keyword univoche ammesse per l'initget (aggirando i conflitti alfanumerici). Concatena i dati in una stringa per offrire un HUD testuale costantemente aggiornato sullo stato dei layer, sul moltiplicatore asintotico (E:) e sul metodo di inserimento corrente (M:).

CON4P_Identify_Locus_Target

Firma: (CON4P_Identify_Locus_Target p5 opt_gc opt_gf opt_gv locus_vla_list par_data_f meta_map_f par_data_v meta_map_v pts_ord P1 P2 P3 P4 Cg opt_m opt_e last_run_ents)

Scopo: Smistatore LUSNAP globale per i luoghi continui e discreti.

Input: Tutte le mappe e stati di configurazione visiva.

Output: Flag finali per main loop.

Logica interna: Esegue check concorrenti su proiezioni VLA e meta-mappe frattali, assegnando la priorità al layer geograficamente più prossimo al cursore target.

CON4P_Identify_Node_Target

Firma: (CON4P_Identify_Node_Target p5 P1 P2 P3 P4 node_solutions last_node_idx sol_idx opt_m opt_e last_run_ents Cg)

Scopo: Elaborare l'aggancio del cursore sui nodi analitici primari gestendo lo switch circolare delle soluzioni multiple.

Input: Parametri spaziali e vettoriali del nodo.

Output: Nuova stringa LUSNAP con liste ents aggiornate.

Logica interna: Cliclando sullo stesso nodo il sistema cicla (modulo) tra le soluzioni precalcolate.

CON4P_Identify_Orthogonal_Target

Firma: (CON4P_Identify_Orthogonal_Target p5 pts_ord Cg mode opt_m opt_e last_ents f_target_type)

Scopo: Interfaccia LUSNAP esclusiva per il Ramo 2A (Rettangoli). Proietta il click, risolve la conica e lancia il rendering.

Input: p5 (Click mouse), pts_ord, Cg, mode, opt_m, opt_e, last_ents, f_target_type.

Output: Pacchetto completo dati conica e ENAME (append g_data new_ents).

Logica interna: Proietta ortogonalmente il punto del mouse sugli assi di simmetria del rettangolo. Invia il punto proiettato al risolutore analitico e aggiorna la grafica in un solo colpo.

CON4P_Local_Collect

Firma: (CON4P_Local_Collect marker)

Scopo: Estrazione sicura e cronologica degli oggetti disegnati nel database.

Input: marker (ENAME del punto di partenza).

Output: Lista di entità ENAME inserite dopo il marker.

Logica interna: Clicla con entnext accumulando nella lista in ordine inverso.

CON4P_Process_Segment_Matrioska

Firma: (CON4P_Process_Segment_Matrioska obj st en stp pts_ord mode D_ref)

Scopo: Interfaccia di campionamento asimmetrico per la generazione fisica.

Input: obj (Curva VLA), st (Inizio T), en (Fine T), stp (Step base lineare), pts_ord, mode, D_ref.

Output: Mappa punti via esecuzione atomica.

Logica interna: Delega il calcolo matematico a CON4P_Core_Matrioska_Engine, occupandosi esclusivamente del tracciamento a video (debug) e dell'invio alla funzione di generazione VLA.

CON4P_Process_Single_VLA_Point

Firma: (CON4P_Process_Single_VLA_Point vla_obj param_val pts_ord mode ref_len step_val)

Scopo: Elaboratore atomico per isolare e calcolare i punti geometrici discreti (fuochi o vertici) estratti da un singolo parametro locale di una curva VLA.

Input: vla_obj (Entità curva VLA di base), param_val (Parametro t analizzato), pts_ord (Lista vertici WCS), mode ("FOCI" o "VERTICES"), ref_len (Distanza di scala), step_val (Passo parametrico della griglia).

Output: Nessun output diretto (agisce per side-effect incrementando valid_centers_count e aggiornando gli array globali pts_1, pts_2, pts_4, pts_map, meta_map).

Logica interna: Estrae il punto vettoriale associato al parametro e calcola la conica passante per quel centro. In base al 'mode' impostato, spezza il centro per ricavare simmetricamente fuochi o vertici. Completa il processo formattando i payload informativi associando ai punti i rispettivi metadati di curve VLA per il tooltip LUSNAP.

CON4P_Refresh_Last_Conic

Firma: (CON4P_Refresh_Last_Conic last_geo opt_m opt_e last_run_ents P1 P2 P3 P4 pts_ord Cg pts_map_f)

Scopo: Ridisegnare l'ultima conica calcolata in seguito a un aggiornamento del parametro opt_e.

Input: Configurazione passata, base punti, stato estensione.

Output: ENAME.

Logica interna: Wrapper per i draw engine.

CON4P_Refresh_Orthogonal_Graph

Firma: (CON4P_Refresh_Orthogonal_Graph P1 P2 P3 P4 Cg g_data opt_m opt_e last_ents f_target_type)

Scopo: Ricalcolare e ridisegnare istantaneamente la conica ortogonale corrente in seguito alla modifica dell'estensione.

Input: P1..P4, Cg, g_data (Dati geometrici), opt_m, opt_e, last_ents, f_target_type.

Output: Lista ENAME aggiornata.

Logica interna: Ricava un punto di ricalcolo sull'asse usando i dati preesistenti e rilancia il motore grafico del rettangolo (Engine_Local).

CON4P_Solve_Orthogonal_Conic

Firma: (CON4P_Solve_Orthogonal_Conic Cg P_rect P_axis mode f_target_type)

Scopo: Risolvere analiticamente i semiassi di una conica ortogonale dato il centro, un vertice del rettangolo e un punto proiettato sull'asse.

Input: Cg (Baricentro), P_rect (Vertice rettangolo), P_axis (Punto proiettato sull'asse), mode ("F" o "V"), f_target_type ('ELLISSE o 'IPERBOLE).

Output: Lista (f_type V_dir a b) o nil se degenere.

Logica interna: Inverte le formule della distanza focale e dell'equazione canonica per trovare i semiassi esatti senza tentativi o scansioni LUSNAP empiriche.

c:con5p

Scopo: Comando laboratorio per disegnare la conica passante per 5 punti in modalità "base 4 punti fissi + 5° punto mobile": l'utente seleziona 4 punti base, poi continua ad aggiungere un 5° punto in posizioni diverse vedendo la conica aggiornarsi ad ogni nuova selezione, con opzioni Guide e Modo.

Input: Nessun parametro: selezione interattiva di 4 punti base, poi loop di selezione del 5° punto con getpoint. Opzioni: [Guide] ON/OFF, [Modo] Sostituisci/Aggiungi, [Fine].

Output: Conica passante per i 5 punti disegnata nel documento; quadrangolo guida e parabole guida visibili se Guide=ON; princ alla fine.

Logica interna: Per ogni 5° punto: Draw_Con5P_Nor_Compl chiama il motore globale con Alg_Tournament_5P_Centered e Draw_ appropriate; in modalità Sostituisci cancella le entità dell'iterazione precedente con Util_Collect_Ents/Del.

c:con5pRea

Scopo: Comando di editing reattivo per la conica a 5 punti: a differenza di c:con5p, il 5° punto è già selezionato e l'utente sposta interattivamente il cursore vedendo la conica ridisegnarsi in tempo reale ad ogni nuovo punto indicato, mantenendo l'impalcatura (quadrangolo + guide) persistente sullo schermo.

Input: Nessun parametro: selezione di 4 punti base, poi 5° punto iniziale, poi loop reattivo con getpoint (OSMODE=0) che ridisegna ad ogni movimento.

Output: Conica a 5 punti aggiornata in tempo reale; entità temporanee cancellate e ridisegnate ad ogni ciclo; princ alla fine del loop.

Logica interna: Fase A: disegna quadrangolo e guide iniziali. Fase B: loop reattivo che ad ogni nuovo punto cancella le entità conica precedenti, ricalcola e ridisegna tramite Draw_Con5P_Nor_Compl, mantenendo quadrangolo/guide invariati.

Config_Conic_Layers

Scopo: Coordinare l'inizializzazione e la configurazione di tutti i layer operativi necessari all'ambiente di disegno delle coniche, tentando prima un caricamento rapido e passando, se necessario, alla definizione manuale interattiva.

Input: Nessuno (l'acquisizione dei dati avviene interattivamente tramite riga di comando o lettura del dizionario LDATA).

Output: Nessun output diretto. Genera un side-effect valorizzando le variabili globali di sistema (es. *LaCo_Poligoni*).

Logica interna: Stampa l'intestazione di avvio e interroga 'Load_Conic_Layers_Fast'. Se l'esito à negativo (layer mancanti o scelta dell'utente di riconfigurare), avvia un blocco 'progn' che valorizza singolarmente l'intero set di 11 variabili globali dedicate ai layer, richiamando per ciascuna l'helper 'Get_Layer_Smart' con il relativo prompt, la chiave LDATA e il nome di default. Conclude stampando un messaggio di conferma a video.

D

Draw_Circle_R

Firma: (Draw_Circle_R Pu1 Rad Lay)

Scopo: Generare fisicamente e istantaneamente nel database DWG le entità grafiche primitive di base (punto, linea, cerchio, ellisse) bypassando l'uso della riga di comando per massimizzare le prestazioni di tracciamento.

Input: Variano per entità: coordinate base (Pu1, Pu2), parametri di forma (Rad, Vec_Maj, Ratio) e il nome testuale del layer di destinazione (se nullo, il sistema instrada automaticamente l'oggetto sul layer "0").

Output: Restituisce il puntatore Entity Name (ENAME) dell'oggetto vettoriale appena creato nel disegno.

Logica interna: Sfrutta la funzione nativa a basso livello `entmakex`. Per ciascuna primitiva, assembla al volo la relativa lista associativa DXF (associando i codici standard 0, 8, 10, 11, 40, ecc. ai parametri forniti) garantendo un inserimento silenzioso, diretto e indipendente dallo stato dell'interfaccia. (defun Draw_Point_R (Pu1 lay) (entmakex (list '(0 . "POINT") (cons 10 Pu1) (cons 8 (if lay lay "0"))))) (defun Draw_Line_2Pu_R (Pu1 Pu2 lay) (entmakex (list '(0 . "LINE") (cons 10 Pu1) (cons 11 Pu2) (cons 8 (if lay lay "0")))))

Draw_Con5P_Nor

Firma: (Draw_Con5P_Nor pts pts_limit full_mode lay_crv lay_constr)

Scopo: Orchestratore unico per il tracciamento di coniche da 5 punti (Analisi Pascal), centralizzando la normalizzazione, la diagnosi del tipo di curva e il torneo combinatorio per la risoluzione matriciale.

Input: pts (Lista 5 punti), pts_limit (Limiti estensione), full_mode (Flag T/nil), lay_crv/lay_constr (Layer dummy per retrocompatibilità).

Output: Lista di oggetti VLA (o singolo VLA) pronti per l'interazione CAD.

Logica interna: Esegue la normalizzazione unitaria del set di punti. Analizza la geometria per distinguere tra rami parabolici e centrali. Se 'full_mode' à attivo, abilita i feedback a riga di comando. Richiama poi direttamente i motori Core unificati (Draw_ConPar_5P_Sub o Draw_ConCen_3P) passandogli il flag per delegare la vestizione.

Draw_Con5P_Nor_Compl

Firma: (Draw_Con5P_Nor_Compl pts lay_crv lay_constr pts_limit)

Scopo: Fornire punti di ingresso specializzati per il disegno di coniche da 5 punti, mantenendo le firme originali per retrocompatibilità.

Input: pts (5 punti), pts_limit (estensione), [lay_crv lay_constr solo per versione Compl].

Output: Oggetti VLA convertiti.

Logica interna: Demandano l'esecuzione al motore 'Draw_Con5P_Nor'. La versione 'Compl' attiva la vestizione anatomica e i layer personalizzati, mentre la versione 'Light' forza il ritorno della sola geometria sui layer dei luoghi geometrici.

Draw_Con5P_Nor_Light

Firma: (Draw_Con5P_Nor_Light pts pts_limit)

Scopo: Fornire punti di ingresso specializzati per il disegno di coniche da 5 punti, mantenendo le firme originali per retrocompatibilità.

Input: pts (5 punti), pts_limit (estensione), [lay_crv lay_constr solo per versione Compl].

Output: Oggetti VLA convertiti.

Logica interna: Demandano l'esecuzione al motore 'Draw_Con5P_Nor'. La versione 'Compl' attiva la vestizione anatomica e i layer personalizzati, mentre la versione 'Light' forza il ritorno della sola geometria sui layer dei luoghi geometrici. (defun Draw_Con5P_Nor_Compl (pts lay_crv lay_constr pts_limit) (Draw_Con5P_Nor pts pts_limit T lay_crv lay_constr))

Draw_ConCen_3P

Firma: (Draw_ConCen_3P Center P1 P2 P3 pts_limit full_mode)

Scopo: Risolvere il sistema matriciale 3x3 per coniche a centro (Ellisse/Iperbole), unificando il calcolo di autovalori e autovettori, per poi delegare il disegno ai motori grafici in base al flag di vestizione anatomica.

Input: Center (Centro), P1, P2, P3 (Punti noti), pts_limit (Limiti estensione per Iperbole), full_mode (Flag T/nil).

Output: Lista di ename (con anatomia completa se full_mode=T, altrimenti solo le curve). Restituisce nil in caso di degenerazione.

Logica interna: Esegue un pre-check topologico per evitare sovrapposizioni. Normalizza i punti rispetto al centro. Calcola i coefficienti del fascio e ne estrae lo spettro (autovalori). In base al segno del determinante, smista su Ellisse o Iperbole. Calcola i semiassi e chiama i rispettivi motori (Draw_ConEll_... o Draw_ConIpe_...) iniettando il parametro 'full_mode' per governare l'output grafico finale.

Draw_ConCen_3P_Compl

Firma: (Draw_ConCen_3P_Compl Center P1 P2 P3 pts_limit)

Scopo: Interfacce di retrocompatibilità per il solutore matriciale 3x3. NOTE SUI PARAMETRI: Le variabili 'lay_crv' e 'lay_constr' nella versione Compl sono mantenute volontariamente nella firma come "dummy arguments" per evitare errori di chiamata 'too many arguments' dai comandi storici del listato.

Input: Parametri geometrici o di stato indicati nella firma.

Output: Risultato calcolato o effetto collaterale sul disegno.

Draw_ConCen_3P_Light

Firma: (Draw_ConCen_3P_Light Center P1 P2 P3 pts_limit)

Scopo: Interfacce di retrocompatibilità per il solutore matriciale 3x3. NOTE SUI PARAMETRI: Le variabili 'lay_crv' e 'lay_constr' nella versione Compl sono mantenute volontariamente nella firma come "dummy arguments" per evitare errori di chiamata 'too many arguments' dai comandi storici del listato. (defun Draw_ConCen_3P_Compl (Center P1 P2 P3 pts_limit) (Draw_ConCen_3P Center P1 P2 P3 pts_limit T))

Input: Parametri geometrici o di stato indicati nella firma.

Output: Risultato calcolato o effetto collaterale sul disegno.

Draw_ConCen_4PC

Firma: (Draw_ConCen_4PC cen pts lay)

Scopo: Disegna la conica passante per 4 punti con centro fisso dato, coordinando il calcolo matematico e il rendering grafico.

Input: cen: coordinate del centro fisso della conica. pts: lista di 4 punti di passaggio. lay: layer di destinazione.

Output: Entità AutoCAD dell'ellisse o iperbole disegnata; nil se il calcolo fallisce.

Logica interna: Chiama `Geom_ConCen_4PC` per ottenere i coefficienti, poi `Math_Extract_Geom_Data_From_Coeffs` per estrarre i parametri geometrici, infine instrada al disegnatore corretto (`Draw_ConEll_*` o `Draw_ConIpe_*`) secondo il tipo di conica.

Draw_ConEll_CeAx_Foci

Firma: (Draw_ConEll_CeAx_Foci Center A1 A2 B1 B2 F1 F2 Ratio)

Scopo: Generare fisicamente nel database di AutoCAD l'ellisse completa della sua impalcatura geometrica (centri, assi, fuochi), smistando gli oggetti sui rispettivi layer/colori.

Input: Center (Centro), A1, A2 (Vertici maggiori), B1, B2 (Vertici minori), F1, F2 (Fuochi), Ratio (Rapporto b/a).

Output: Lista aggregata contenente gli ename di tutte le entità appena generate a schermo.

Logica interna: Calcola il vettore direzionale maggiore partendo da centro e A1. Richiama in sequenza le primitive grafiche per tracciare la conica nativa (Draw_ConEll_R), il centro, i segmenti degli assi e i punti

focali. Raccoglie i puntatori in un'unica lista piatta tramite append, iniettando in ogni primitiva le costanti globali di tematizzazione (*LaCo_Tracciati*, *LaCo_Centri*, ecc.) per garantire la coerenza visiva dell'ambiente.

Draw_ConEll_CeAx_Light

Firma: (Draw_ConEll_CeAx_Light Center A1 Ratio)

Scopo: Versione "light" del disegnatore di ellissi: disegna SOLO l'ellisse senza assi, fuochi o punti costruttivi aggiuntivi, per uso nei loop reattivi dove la velocità di ridisegno è critica.

Input: cen: centro dell'ellisse. ax1, ax2: vettori dei due semiassi (direzione+lunghezza). lay: nome del layer di destinazione.

Output: Ename dell'ellisse disegnata; nil se i parametri sono degeneri.

Logica interna: Calcola la lunghezza dei semiassi con distance e l'angolo di orientamento con angle; chiama command "._ELLIPSE" con l'opzione centro-asse-semiasse.

Draw_ConEll_CeVe_CoVe

Firma: (Draw_ConEll_CeVe_CoVe Center V1 F1 pts)

Scopo: Calcolare i parametri geometrici completi (assi e vertici mancanti) di un'ellisse a partire dal centro, un vertice e un fuoco, delegandone poi il tracciamento grafico.

Input: Center (Coordinata 3D del centro), V1 (Coordinata 3D di un vertice sull'asse maggiore), F1 (Coordinata 3D di un fuoco), pts (Parametro di appoggio/legacy).

Output: Lista di ename (puntatori alle entità grafiche) restituita dalla funzione di disegno chiamata a valle.

Logica interna: Estrae il vettore direzionale normalizzato dell'asse maggiore. Calcola il semiasse maggiore (a) e la distanza focale (c), derivando il semiasse minore (b) tramite Pitagora (con fallback di sicurezza 0.1*a per ellissi degeneri). Ricava per simmetria vettoriale e proiezioni ortogonali il vertice opposto (A2), i co-vertici (B1, B2) e il secondo fuoco (F2). Calcola il rapporto (ratio) e passa l'intero set di dati al costruttore grafico.

Draw_ConEll_R

Firma: (Draw_ConEll_R Pu1 Vec_Maj Ratio lay)

Scopo: Generare fisicamente e istantaneamente nel database DWG le entità grafiche primitive di base (punto, linea, cerchio, ellisse) bypassando l'uso della riga di comando per massimizzare le prestazioni di tracciamento.

Input: Variano per entità: coordinate base (Pu1, Pu2), parametri di forma (Rad, Vec_Maj, Ratio) e il nome testuale del layer di destinazione (se nullo, il sistema instrada automaticamente l'oggetto sul layer "0").

Output: Restituisce il puntatore Entity Name (ENAME) dell'oggetto vettoriale appena creato nel disegno.

Logica interna: Sfrutta la funzione nativa a basso livello `entmakex`. Per ciascuna primitiva, assembla al volo la relativa lista associativa DXF (associando i codici standard 0, 8, 10, 11, 40, ecc. ai parametri forniti) garantendo un inserimento silenzioso, diretto e indipendente dallo stato dell'interfaccia. (defun Draw_Point_R (Pu1 lay) (entmakex (list '(0 . "POINT") (cons 10 Pu1) (cons 8 (if lay lay "0"))))) (defun Draw_Line_2Pu_R (Pu1 Pu2 lay) (entmakex (list '(0 . "LINE") (cons 10 Pu1) (cons 11 Pu2) (cons 8 (if lay lay "0")))))) (defun Draw_Circle_R (Pu1 Rad Lay) (entmakex (list '(0 . "CIRCLE") (cons 10 Pu1) (cons 40 Rad) (cons 8 (if Lay Lay "0")))))

Draw_Conic_4P_AxisDir

Firma: (Draw_Conic_4P_AxisDir Pu1 Pu2 Pu3 Pu4 PuD1 PuD2 lay_crv lay_constr lay_pts draw_input_geom)

Scopo: Disegna la conica passante per 4 punti con l'asse principale orientato nella direzione specificata da due punti, usando il metodo del fascio di coniche.

Input: Pu1, Pu2, Pu3, Pu4: quattro punti di vincolo. PuD1, PuD2: punti che definiscono la direzione dell'asse. lay_crv, lay_constr: layer di destinazione.

Output: Entità AutoCAD della conica (ellisse o iperbole) con i suoi elementi costruttivi; nil se la configurazione è degenere.

Logica interna: Chiama Math_Get_Coeffs_4P_AxisDir per ottenere i coefficienti del fascio con il vincolo di direzione, poi Solve_And_Draw_From_Coeffs per il disegno della conica risultante.

Draw_ConIpe_4P_Eq

Firma: (Draw_ConIpe_4P_Eq Pu1 Pu2 Pu3 Pu4 lay_crv lay_constr)

Scopo: Disegna l'iperbole equilatera (rettangola) passante per 4 punti dati, calcolando prima i coefficienti con Math_Extract_Equilateral_Data e poi disegnando la curva con assi e asintoti.

Input: Pu1, Pu2, Pu3, Pu4: quattro punti di vincolo. lay_crv: layer per la curva. lay_constr: layer per le costruzioni.

Output: Lista di entità AutoCAD disegnate (iperbole + asintoti + marcatori). nil se la configurazione è degenere o ortocentrica.

Logica interna: Verifica la topologia con Geom_Analyze_Ipe_Topology. Costruisce il fascio coniugato K1+λK2 con Math_Multiply_Lines_To_Conic, impone il vincolo A+C=0 (iperbole equilatera) per trovare λ, poi chiama Solve_And_Draw_From_Coeffs per il disegno finale.

Draw_ConIpe_CeVeFu

Firma: (Draw_ConIpe_CeVeFu Center V1 F1 pts free_mode full_mode)

Scopo: Motore universale per il tracciamento dell'iperbole. Gestisce centralmente il calcolo dei parametri, l'estensione dei rami e smista la generazione grafica tra anatomia completa e sola curva (Light).

Input: Center (Centro), V1 (Vertice), F1 (Fuoco), pts (Punti), free_mode (Flag estensione libera), full_mode (Flag vestizione anatomica).

Output: Lista di ename. Se full_mode=T, lista piatta con curva e anatomia; se full_mode=nil, lista di 2 elementi (solo i due rami).

Logica interna: Calcola l'apertura e i punti di controllo. Se 'full_mode' à attivo, elabora asintoti e assi, aggregando le entità tramite 'append' sui layer tematici. Se disattivato, traccia solo le due spline sul layer dei luoghi geometrici.

Draw_ConIpe_CeVeFu_Compl

Firma: (Draw_ConIpe_CeVeFu_Compl Center V1 F1 pts)

Scopo: Interfacce specializzate per il tracciamento dell'iperbole, differenziate per estensione rami e livello di vestizione.

Input: Center (Centro), V1 (Vertice), F1 (Fuoco), pts (Punti estensione).

Output: Le versioni 'Light' restituiscono una lista di 2 ename (solo i rami); le 'Compl' restituiscono l'intera anatomia in lista piatta.

Logica interna: Demandano al motore unificato 'Draw_ConIpe_CeVeFu' iniettando le 4 combinazioni possibili dei flag di controllo: - Compl: Estensione protetta (nil), vestizione completa (T). - Compl_Free: Estensione libera (T), vestizione completa (T). - Light: Estensione protetta (nil), nessuna vestizione (nil). - Light_Free: Estensione libera (T), nessuna vestizione (nil).

Draw_ConIpe_CeVeFu_Compl_Free

Firma: (Draw_ConIpe_CeVeFu_Compl_Free Center V1 F1 pts)

Scopo: Interfacce specializzate per il tracciamento dell'iperbole, differenziate per estensione rami e livello di vestizione.

Input: Center (Centro), V1 (Vertice), F1 (Fuoco), pts (Punti estensione).

Output: Le versioni 'Light' restituiscono una lista di 2 ename (solo i rami); le 'Compl' restituiscono l'intera anatomia in lista piatta.

Logica interna: Demandano al motore unificato 'Draw_ConIpe_CeVeFu' iniettando le 4 combinazioni possibili dei flag di controllo: - Compl: Estensione protetta (nil), vestizione completa (T). - Compl_Free: Estensione libera (T), vestizione completa (T). - Light: Estensione protetta (nil), nessuna vestizione (nil). - Light_Free: Estensione libera (T), nessuna vestizione (nil). (defun Draw_ConIpe_CeVeFu_Compl (Center V1 F1 pts) (Draw_ConIpe_CeVeFu Center V1 F1 pts nil T))

Draw_ConIpe_CeVeFu_Light

Firma: (Draw_ConIpe_CeVeFu_Light Center V1 F1 pts)

Scopo: Interfacce specializzate per il tracciamento dell'iperbole, differenziate per estensione rami e livello di vestizione.

Input: Center (Centro), V1 (Vertice), F1 (Fuoco), pts (Punti estensione).

Output: Le versioni 'Light' restituiscono una lista di 2 ename (solo i rami); le 'Compl' restituiscono l'intera anatomia in lista piatta.

Logica interna: Demandano al motore unificato 'Draw_ConIpe_CeVeFu' iniettando le 4 combinazioni possibili dei flag di controllo: - Compl: Estensione protetta (nil), vestizione completa (T). - Compl_Free: Estensione libera (T), vestizione completa (T). - Light: Estensione protetta (nil), nessuna vestizione (nil). - Light_Free: Estensione libera (T), nessuna vestizione (nil). (defun Draw_ConIpe_CeVeFu_Compl (Center V1 F1 pts) (Draw_ConIpe_CeVeFu Center V1 F1 pts nil T)) (defun Draw_ConIpe_CeVeFu_Compl_Free (Center V1 F1 pts) (Draw_ConIpe_CeVeFu Center V1 F1 pts T T))

Draw_ConIpe_CeVeFu_Light_Free

Firma: (Draw_ConIpe_CeVeFu_Light_Free Center V1 F1 pts)

Scopo: Interfacce specializzate per il tracciamento dell'iperbole, differenziate per estensione rami e livello di vestizione.

Input: Center (Centro), V1 (Vertice), F1 (Fuoco), pts (Punti estensione).

Output: Le versioni 'Light' restituiscono una lista di 2 ename (solo i rami); le 'Compl' restituiscono l'intera anatomia in lista piatta.

Logica interna: Demandano al motore unificato 'Draw_ConIpe_CeVeFu' iniettando le 4 combinazioni possibili dei flag di controllo: - Compl: Estensione protetta (nil), vestizione completa (T). - Compl_Free: Estensione libera (T), vestizione completa (T). - Light: Estensione protetta (nil), nessuna vestizione (nil). - Light_Free: Estensione libera (T), nessuna vestizione (nil). (defun Draw_ConIpe_CeVeFu_Compl (Center V1 F1 pts) (Draw_ConIpe_CeVeFu Center V1 F1 pts nil T)) (defun Draw_ConIpe_CeVeFu_Compl_Free (Center V1 F1 pts) (Draw_ConIpe_CeVeFu Center V1 F1 pts T T)) (defun Draw_ConIpe_CeVeFu_Light (Center V1 F1 pts) (Draw_ConIpe_CeVeFu Center V1 F1 pts nil nil))

Draw_ConPar_4P_ConVer_R

Firma: (Draw_ConPar_4P_ConVer_R P1 P2 P3 P4 lay_crv lay_constr)

Scopo: Disegna la parabola passante per 4 punti con la tecnica costruttiva del vertice reciproco (R): sfrutta la diagonale del quadrangolo come asse naturale della parabola, include assi e vertice nel disegno.

Input: V1, V2: coppia diagonale principale (asse naturale). P1, P2: coppia diagonale secondaria. lay_crv: layer curva. lay_constr: layer costruzioni.

Output: Lista di entità disegnate (spline parabola + asse + vertice). nil se degenere.

Logica interna: Calcola l'asse come retta V1-V2, trova il punto medio (vertice parametrico), costruisce la parabola per i 4 punti interpolando nel sistema locale dell'asse, poi disegna la spline con una serie di punti campionati lungo la curva.

Draw_ConPar_4P_Light

Firma: (Draw_ConPar_4P_Light P1 P2 P3 P4)

Scopo: Versione "light" del disegnatore di parabola a 4 punti: disegna solo la curva parabolica senza guide, assi o costruzioni aggiuntive, ottimizzata per i loop reattivi.

Input: Pu1, Pu2, Pu3, Pu4: quattro punti di passaggio. lay_crv: layer di destinazione della curva.

Output: Ename della spline parabolica disegnata; nil se la configurazione è degenere.

Logica interna: Chiama Draw_ConPar_4P_ConVer_R in modalità light, instradando solo il disegno della spline e omettendo tutto il layer costruzioni.

Draw_ConPar_5P_Sub

Firma: (Draw_ConPar_5P_Sub pts_norm analyze_res G_bary max_dist pts_limit full_mode)

Scopo: Risolvere e tracciare la parabola da 5 punti (Pascal), ottimizzando le prestazioni tramite chiamata diretta al motore grafico core.

Input: pts_norm (Punti normalizzati), analyze_res (Analisi Pascal), G_bary (Baricentro), max_dist (Scala), pts_limit (Estensione), full_mode (Flag T/nil).

Output: Lista di ename (sempre in formato lista: (ename...) o (ename)).

Logica interna: Esegue il solver geometrico e denormalizza i punti critici. Interpella direttamente 'Draw_ConPar_VeFu' passando il flag 'full_mode'. Per la modalità Light (nil), avvolge l'atomo restituito in una lista per mantenere la coerenza strutturale attesa dai chiamanti.

Draw_ConPar_5P_Sub_Compl

Firma: (Draw_ConPar_5P_Sub_Compl pts_norm analyze_res G_bary max_dist pts_limit)

Scopo: Wrapper di retrocompatibilità per il disegno di parabole da 5 punti (Analisi Pascal).

Input: pts_norm, analyze_res, G_bary, max_dist, pts_limit (Coordinate e parametri di trasformazione).

Output: Lista di ename (Entità grafiche generate).

Logica interna: Demandano il calcolo al motore 'Draw_ConPar_5P_Sub'. La versione '_Compl' richiede la vestizione anatomica, mentre la versione '_Light' restituisce la sola curva, mantenendo l'output originale (lista singola).

Draw_ConPar_5P_Sub_Light

Firma: (Draw_ConPar_5P_Sub_Light pts_norm analyze_res G_bary max_dist pts_limit)

Scopo: Wrapper di retrocompatibilità per il disegno di parabole da 5 punti (Analisi Pascal).

Input: pts_norm, analyze_res, G_bary, max_dist, pts_limit (Coordinate e parametri di trasformazione).

Output: Lista di ename (Entità grafiche generate).

Logica interna: Demandano il calcolo al motore 'Draw_ConPar_5P_Sub'. La versione '_Compl' richiede la vestizione anatomica, mentre la versione '_Light' restituisce la sola curva, mantenendo l'output originale (lista singola). (defun Draw_ConPar_5P_Sub_Compl (pts_norm analyze_res G_bary max_dist pts_limit) (Draw_ConPar_5P_Sub pts_norm analyze_res G_bary max_dist pts_limit T))

Draw_ConPar_Data_3P

Firma: (Draw_ConPar_Data_3P pts_3 refined_data)

Scopo: Tracciare una parabola a partire da dati geometrici pre-calcolati (terna Vertice-Fuoco-Angolo), bypassando ricalcoli analitici onerosi e delegando il disegno anatomico al motore core.

Input: pts_3 (Lista punti base per l'estensione), refined_data (Lista strutturata: Vertice, Fuoco, Angolo).

Output: Lista di ename (5 entità: curva e anatomia completa) generata dal motore grafico.

Logica interna: Verifica la validità dei dati affinati. Estrae le coordinate di Vertice (car) e Fuoco (cadr) e le passa direttamente alla funzione 'Draw_ConPar_VeFu' attivando il flag 'T' (full_mode). Questo garantisce il tracciamento automatico di assi e direttrici sui layer globali predefiniti senza passaggi intermedi.

Draw_ConPar_VeFu

Firma: (Draw_ConPar_VeFu PuV PuF pts full_mode)

Scopo: Centralizzare il calcolo geometrico e il tracciamento della parabola, ottimizzando la logica di calcolo e smistando l'output tra rappresentazione completa (vestizione) e nuda curva (luogo geometrico).

Input: PuV (Vertice), PuF (Fuoco), pts (Lista di punti per estensione), full_mode (Flag booleano: T per vestizione, nil per curva singola).

Output: Se full_mode = T: Lista di 5 ename (Curva, Asse, Direttrice, Fuoco, Vertice). Se full_mode = nil: Singolo ename (Atomo) della curva spline.

Logica interna: Calcola i parametri comuni (vettori, distanze, limiti di estensione). Tramite un singolo 'if', separa il tracciamento: il ramo 'true' calcola gli elementi anatomici e restituisce una lista aggregata sui layer

tematici; il ramo 'else' genera la sola spline sul layer dei luoghi geometrici, garantendo la perfetta compatibilità con i chiamanti "Light".

Draw_ConPar_VeFu_Compl

Firma: (Draw_ConPar_VeFu_Compl PuV PuF pts)

Scopo: Fornire interfacce specializzate (wrapper) per il disegno della parabola, permettendo di scegliere tra una rappresentazione completa di anatomia e una puramente geometrica, mantenendo la retrocompatibilità con i comandi esistenti.

Input: PuV (Vertice), PuF (Fuoco), pts (Lista di punti per il calcolo dell'estensione della curva).

Output: La versione '_Compl' restituisce una lista di ename; la versione '_Light' restituisce un singolo ename (atomo).

Logica interna: Entrambe le funzioni demandano l'intero processo di calcolo e tracciamento all'helper unificatore 'Draw_ConPar_VeFu'. La variante 'Compl' attiva il flag di disegno per assi, fuochi e direttrici sui layer tematici, mentre la variante 'Light' isola la sola curva sul layer dei luoghi geometrici, preservando l'output atomico originale.

Draw_ConPar_VeFu_Light

Firma: (Draw_ConPar_VeFu_Light PuV PuF pts)

Scopo: Fornire interfacce specializzate (wrapper) per il disegno della parabola, permettendo di scegliere tra una rappresentazione completa di anatomia e una puramente geometrica, mantenendo la retrocompatibilità con i comandi esistenti.

Input: PuV (Vertice), PuF (Fuoco), pts (Lista di punti per il calcolo dell'estensione della curva).

Output: La versione '_Compl' restituisce una lista di ename; la versione '_Light' restituisce un singolo ename (atomo).

Logica interna: Entrambe le funzioni demandano l'intero processo di calcolo e tracciamento all'helper unificatore 'Draw_ConPar_VeFu'. La variante 'Compl' attiva il flag di disegno per assi, fuochi e direttrici sui layer tematici, mentre la variante 'Light' isola la sola curva sul layer dei luoghi geometrici, preservando l'output atomico originale. (defun Draw_ConPar_VeFu_Compl (PuV PuF pts) (Draw_ConPar_VeFu PuV PuF pts T))

Draw_Curve_Segment_Spline

Firma: (Draw_Curve_Segment_Spline pts_list lay)

Scopo: Generare un'entità grafica curvilinea (Spline) o rettilinea (Linea) a partire da una lista di punti, adattando automaticamente il tipo di oggetto CAD creato in base alla densità dei dati di input forniti.

Input: pts_list (Lista di coordinate WCS dei punti), lay (Stringa indicante il layer di destinazione per l'entità generata).

Output: Restituisce l'identificatore dell'entità (ENAME) creata tramite entmakex, oppure nil se la lista contiene meno di 2 punti.

Logica interna: Determina la quantità di punti in ingresso. Se superiore a 2, assembla una lista DXF per la creazione di una SPLINE di grado 3 (cubica), mappando ogni punto di controllo sul codice 11 tramite 'mapcar'. Se i punti sono esattamente 2, ottimizza il database generando una primitiva LINE. Utilizza 'entmakex' per un inserimento diretto e silenzioso nel database del disegno.

Draw_Guide_Parabolas_4P

Firma: (Draw_Guide_Parabolas_4P pts)

Scopo: Disegna le parabole guida nell'ambiente a 4 punti: traccia le parabole passanti per le coppie di punti che delimitano il fascio di coniche nel quadrangolo dato.

Input: pts: lista di 4 punti del quadrangolo.

Output: Lista delle entità disegnate (spline/parabole guida). Effetto collaterale: guide disegnate nel documento.

Logica interna: Classifica i 4 punti con `Math_Classify_Forma_4Pu` per determinare l'ordine topologico. Seleziona le coppie di punti diagonali/opposti e chiama `Draw_ConPar_*` per ogni parabola guida del fascio.

Draw_Line_2Pu_R

Firma: (Draw_Line_2Pu_R Pu1 Pu2 lay)

Scopo: Generare fisicamente e istantaneamente nel database DWG le entità grafiche primitive di base (punto, linea, cerchio, ellisse) bypassando l'uso della riga di comando per massimizzare le prestazioni di tracciamento.

Input: Variano per entità: coordinate base (Pu1, Pu2), parametri di forma (Rad, Vec_Maj, Ratio) e il nome testuale del layer di destinazione (se nullo, il sistema instrada automaticamente l'oggetto sul layer "0").

Output: Restituisce il puntatore Entity Name (ENAME) dell'oggetto vettoriale appena creato nel disegno.

Logica interna: Sfrutta la funzione nativa a basso livello `entmakex`. Per ciascuna primitiva, assembla al volo la relativa lista associativa DXF (associando i codici standard 0, 8, 10, 11, 40, ecc. ai parametri forniti) garantendo un inserimento silenzioso, diretto e indipendente dallo stato dell'interfaccia. (defun Draw_Point_R (Pu1 lay) (entmakex (list '(0 . "POINT") (cons 10 Pu1) (cons 8 (if lay lay "0")))))

Draw_Lis_Line_Lis_Coppie_Pu_R

Firma: (Draw_Lis_Line_Lis_Coppie_Pu_R lis_coppie lay)

Scopo: Fungono da interfacce di tracciamento massivo (wrapper) per generare simultaneamente interi array di linee o di punti, delegando la creazione fisica alle rispettive primitive di base e restituendo l'elenco dei puntatori generati.

Input: lis_coppie o lis_punti (Lista di coordinate strutturate in base all'entità: coppie di punti per le linee, coordinate singole per i punti base), lay (Stringa indicante il layer di destinazione).

Output: Lista ordinata contenente gli Entity Name (ENAME) di tutte le entità vettoriali appena generate nel database DWG.

Logica interna: Sfrutta l'iteratore funzionale `mapcar` associato a una funzione `lambda` per scorrere massivamente l'array in ingresso. Per ogni elemento, invoca in modo trasparente la funzione primitiva sottostante (`Draw_Line_2Pu_R` o `Draw_Point_R`), accumulando i risultati.

Draw_Lis_Point_R

Firma: (Draw_Lis_Point_R lis_punti lay)

Scopo: Fungono da interfacce di tracciamento massivo (wrapper) per generare simultaneamente interi array di linee o di punti, delegando la creazione fisica alle rispettive primitive di base e restituendo l'elenco dei puntatori generati.

Input: lis_coppie o lis_punti (Lista di coordinate strutturate in base all'entità: coppie di punti per le linee, coordinate singole per i punti base), lay (Stringa indicante il layer di destinazione).

Output: Lista ordinata contenente gli Entity Name (ENAME) di tutte le entità vettoriali appena generate nel database DWG.

Logica interna: Sfrutta l'iteratore funzionale `mapcar` associato a una funzione `lambda` per scorrere massivamente l'array in ingresso. Per ogni elemento, invoca in modo trasparente la funzione primitiva sottostante (`Draw_Line_2Pu_R` o `Draw_Point_R`), accumulando i risultati. (defun Draw_Lis_Line_Lis_Coppie_Pu_R (lis_coppie lay) (mapcar '(lambda (coppia) (Draw_Line_2Pu_R (car coppia) (cadr coppia) lay)) lis_coppie))

Draw_Point_R

Firma: (Draw_Point_R Pu1 lay)

Scopo: Generare fisicamente e istantaneamente nel database DWG le entità grafiche primitive di base (punto, linea, cerchio, ellisse) bypassando l'uso della riga di comando per massimizzare le prestazioni di tracciamento.

Input: Variano per entità: coordinate base (Pu1, Pu2), parametri di forma (Rad, Vec_Maj, Ratio) e il nome testuale del layer di destinazione (se nullo, il sistema instrada automaticamente l'oggetto sul layer "0").

Output: Restituisce il puntatore Entity Name (ENAME) dell'oggetto vettoriale appena creato nel disegno.

Logica interna: Sfrutta la funzione nativa a basso livello `entmakex`. Per ciascuna primitiva, assembla al volo la relativa lista associativa DXF (associando i codici standard 0, 8, 10, 11, 40, ecc. ai parametri forniti) garantendo un

inserimento silenzioso, diretto e indipendente dallo stato dell'interfaccia.

Draw_Quadrangolo_Concavo_Lis4Pu

Firma: (Draw_Quadrangolo_Concavo_Lis4Pu pts lay)

Scopo: Generare l'impalcatura grafica per un quadrangolo concavo, tracciando simultaneamente il perimetro esterno (il triangolo base) e i segmenti a stella interni.

Input: pts (Lista ordinata di 4 coordinate WCS, dove la logica a monte garantisce che il 4à elemento, P4, sia il vertice concavo interno), lay (Stringa layer).

Output: Nessun output diretto. Agisce per side-effect invocando la creazione massiva dei vettori a schermo.

Logica interna: Estrae i 4 vertici. Struttura un array di 6 coppie di coordinate: le prime tre definiscono il contorno del triangolo base (P1-P2, P2-P3, P3-P1), mentre le restanti tre definiscono i "raggi" che congiungono il vertice concavo interno P4 ai tre estremi esterni (P1-P4, P2-P4, P3-P4). Invia l'array al CAD.

Draw_Quadrangolo_Convesso_Lis4Pu

Firma: (Draw_Quadrangolo_Convesso_Lis4Pu pts lay)

Scopo: Generare l'impalcatura grafica di base per un quadrangolo convesso, tracciandone esclusivamente il perimetro chiuso a partire dai suoi quattro vertici spaziali.

Input: pts (Lista ordinata di 4 coordinate WCS rappresentanti i vertici del quadrangolo), lay (Stringa indicante il nome del layer di destinazione).

Output: Nessun output diretto. Agisce per side-effect richiamando i motori grafici di basso livello per generare fisicamente le entità linea nel database DWG.

Logica interna: Decomprime la lista `pts` estraendo le quattro variabili puntuali (P1, P2, P3, P4) tramite `mapcar 'set`. Assembla una matrice contenente le 4 coppie di vertici adiacenti necessarie per sigillare il perimetro del poligono e la invia al wrapper massivo `Draw_Lis_Line_Lis_Coppie_Pu_R` per il rendering.

Draw_Spline_Rational_3CV

Firma: (Draw_Spline_Rational_3CV Pu1 Pu2 Pu3 w lay)

Scopo: Generare matematicamente una conica esatta (arco di ellisse, parabola o iperbole) tracciandola come curva B-Spline Razionale Quadratica basata su tre vertici di controllo spaziali.

Input: CV1, CV2, CV3 (Coordinate WCS dei tre punti di controllo, dove CV2 funge da vertice di tangenza), W (Numero reale che definisce il peso di CV2: <1 ellisse, =1 parabola, >1 iperbole), lay (Layer target).

Output: Puntatore Entity Name (ENAME) dell'oggetto Spline appena generato nel database DWG.

Logica interna: Costruisce l'entità a basso livello tramite `entmakex`. Imposta il grado della curva a 2 e definisce il vettore dei nodi (knot vector) per forzare il passaggio agli estremi (0 0 0 1 1 1). Assegna rigorosamente peso unitario (1.0) ai vertici di ancoraggio CV1 e CV3, mentre applica il parametro di tensione `W` esclusivamente al vertice centrale CV2 per modulare l'eccentricità della conica.

Draw_Triangolo_Lis3Pu

Firma: (Draw_Triangolo_Lis3Pu pts lay)

Scopo: Generare l'impalcatura grafica di base di un triangolo, tracciandone le tre linee perimetrali a partire da una lista contenente i suoi tre vertici spaziali.

Input: pts (Lista di tre coordinate WCS rappresentanti i vertici del triangolo), lay (Stringa indicante il layer di destinazione per le entità generate).

Output: Lista ordinata contenente gli Entity Name (ENAME) dei tre segmenti (linee) generati nel database DWG.

Logica interna: Decomprime la lista `pts` estraendo le singole coordinate (P1, P2, P3). Assembla una nuova lista composta dalle tre coppie di vertici adiacenti e la passa direttamente al motore grafico massivo `Draw_Lis_Line_Lis_Coppie_Pu_R` per eseguire il tracciamento sul layer specificato.

Draw_Triangolo_Ortico_Lis3Pu

Firma: (Draw_Triangolo_Ortico_Lis3Pu pts lay)

Scopo: Calcolare analiticamente e tracciare a video il triangolo ortico (il triangolo i cui vertici coincidono con i piedi delle altezze) derivato da un triangolo base.

Input: pts (Lista di tre coordinate WCS rappresentanti i vertici del triangolo base), lay (Stringa indicante il layer di destinazione per il tracciamento).

Output: Lista ordinata contenente gli Entity Name (ENAME) dei tre segmenti che compongono il triangolo ortico appena disegnato.

Logica interna: Estrae i tre vertici originari. Sfrutta il motore `Math_Proiezione_1Pu_su_2Pu` per calcolare le tre proiezioni ortogonali di ciascun vertice sul lato opposto, ricavando i punti H1, H2 e H3. Infine, organizza questi nuovi vertici in coppie e li delega al wrapper `Draw_Lis_Line_Lis_Coppie_Pu_R`.

E

c:ell3PRD

Scopo: Eseguibile Master per l'inizializzazione dell'ambiente sperimentale dell'Ellisse 3P.

Input: L'utente fornisce i vincoli geometrici (Rapporto o Direzione) via command line.

Output: Renderizzazione CAD assistita e controllo cursore.

Logica interna: Imposta il timer. Processa il Master Engine ricavando il Locus Base. Ordina la batteria analitica esatta per Nodi (Vertici, Fuochi). Dichiara i tempi operativi ed apre il loop interattivo. Ripristina l'ambiente in uscita.

ELL3P_Draw_Engine

Firma: (ELL3P_Draw_Engine geo_packet opt_m last_ents)

Scopo: Driver grafico deputato all'iniezione delle entità CAD (rendering e sostituzione schermo).

Input: geo_packet (Lista nativa di metadati geometrici calcolati), opt_m (Stringa Modalità Sostituzione), last_ents (Memoria delle entità dell'istanza precedente).

Output: Ename handle o lista di entità per lo storaggio a breve termine.

Logica interna: Spazza il buffer distruggendo i residui grafici precedenti qualora si operi in modalità "Sostituisci", ed estrapola i moduli e direzioni vettoriali dal payload, per poi richiedere la stesura dell'ellisse sul WCS.

ELL3P_Eval_Cands

Firma: (ELL3P_Eval_Cands pts cur_ratio ang_rad target_type)

Scopo: Estrarre in real-time l'elenco dei punti notevoli candidati per un dato setup parametrico.

Input: pts (Terna base), cur_ratio (Rapporto b/a), ang_rad (Angolo asse in radianti), target_type (Stringa per Fuochi o Vertici).

Output: Lista di pacchetti analitici formattati per i bersagli, o nil in caso di degenerazione spaziale.

Logica interna: Applica le funzioni di rototraslazione ed invoca il risolutore matematico, blindandolo. Moltiplica vettorialmente i moduli per restituire le coordinate esatte associate al payload informativo.

ELL3P_Eval_Param_Distance

Firma: (ELL3P_Eval_Param_Distance pts mode fixed_val param target_type pt_mouse)

Scopo: Risolvere matematicamente l'ellisse per un parametro istantaneo e restituire la distanza minima dal mouse.

Input: pts (Lista vertici base), mode (Vincolo Direzione o Rapporto), fixed_val (Vincolo bloccato), param (Parametro locale interpolato), target_type (Stringa target), pt_mouse (Punto utente).

Output: Lista contenente (Distanza_Minima (PuntoUtente . Pacchetto_Geometrico_Corrente)) o nil se calcolo fallito/degenere.

Logica interna: Esegue la rototraslazione algebrica per il parametro specifico e invoca il solver. Con il blocco listp protegge l'esito. In base al target_type calcola la distanza 3D tra il risultato matematico e il mouse.

ELL3P_Get_Constraints

Scopo: Inizializzare le logiche e i vincoli operativi matematici forzati dall'utente (Rapporto o Direzione).

Input: Nessuno.

Output: Lista con indice della modalità (1 o 2) e valore scalare/angolare del vincolo.

Logica interna: Clicla la stringa di branching (Rapporto o Direzione). Nel caso Rapporto blocca eventuali overflow ribaltando l'asse (if > 1.0 inverti).

ELL3P_Get_Data

Firma: (ELL3P_Get_Data app_title)

Scopo: Acquisizione dell'impalcatura spaziale e delle variabili OSMODE ambientali.

Input: app_title (Stringa di Benvenuto LUSNAP).

Output: Lista di Backup dati originari.

Logica interna: Sospende il sistema lanciando `Init_Conic_App` e, se l'utente fornisce i punti, spara a video i triangoli di base per contesto ottico.

ELL3P_Helper_UI_Prompt

Firma: (ELL3P_Helper_UI_Prompt opt_gc opt_gf opt_gv opt_t opt_m)

Scopo: Generare la stringa testuale dinamica per il prompt dei comandi in base allo stato delle opzioni visive e operative.

Input: opt_gc (Stato Centri), opt_gf (Stato Fuochi), opt_gv (Stato Vertici), opt_t (Target attuale), opt_m (Modo disegno), opt_e (Moltiplicatore estensione, per coerenza interfaccia).

Output: Stringa testuale pronta per la funzione getpoint.

Logica interna: Assembla le stringhe mostrando le opzioni correnti. Rimuove la possibilità di chiamare "Target" se la visualizzazione dei Vertici/Fuochi à spenta, prevenendo click a vuoto dell'utente.

ELL3P_Master_Engine

Firma: (ELL3P_Master_Engine pts mode fixed_val)

Scopo: Elaborare e generare in una singola passata le mappe di parametri e i gruppi di entità per Centri, Fuochi e Vertici dell'Ellisse.

Input: pts (Lista dei 3 vertici del triangolo), mode (Modalità di vincolo: 1=Direzione, 2=Rapporto), fixed_val (Valore del vincolo bloccato).

Output: Lista di 6 elementi contenente le mappe associative e gli handle (ename) dei gruppi generati (map_C guide_C map_F guide_F map_V guide_V).

Logica interna: Itera sul dominio di definizione (Rapporto o Angolo) per K_HD step. Risolve l'ellisse per ogni iterazione e, blindando la conversione con il controllo listp per scartare le degenerazioni, popola le liste punti se restano all'interno di una tolleranza spaziale (check_dist). Disegna le entità raggruppandole anonimamente.

ELL3P_Math_Error_Func

Firma: (ELL3P_Math_Error_Func pts th current_ratio target_pt target_type)

Scopo: Funzione di Costo per quantificare la discrepanza geometrica rispetto a un Nodo target.

Input: pts (Terna base), th (Angolo asse in radianti), current_ratio (Rapporto b/a), target_pt (Coordinate 3D del target), target_type (Stringa identificativa V_maj, V_min o Fuoco).

Output: Valore scalare assoluto dell'errore geometrico (distanza radiale residua), o nil se degenere.

Logica interna: Tenta la risoluzione algebrica dell'ellisse per quel preciso stato. Verifica validità tramite (listp). Calcola la grandezza vettoriale attesa (a_val, b_val, c_foc) e ne estrae la differenza in valore assoluto rispetto alla distanza tra il centro calcolato e il target fisico.

ELL3P_Precalc_Exact_Nodes

Firma: (ELL3P_Precalc_Exact_Nodes pts mode fixed_val node_type)

Scopo: Calcolare preventivamente le radici analitiche per formare l'archivio volatile O(1).

Input: pts (Terna base), mode (Vincolo), fixed_val (Valore fisso), node_type (Stringa descrittore).

Output: Mappa associativa distribuita per ciascun punto base.

Logica interna: Clicla la chiamata `ELL3P_Solve_Exact_Node_Algebraic` sui tre punti originali assemblando una lista accessibile e rapida.

ELL3P_Refine_Locus

Firma: (ELL3P_Refine_Locus pt_mouse pts map_data mode fixed_val target_type)

Scopo: Catturare ed affinare con elevata precisione l'interpolazione del luogo geometrico in funzione del movimento del cursore (LUSNAP continuo).

Input: pt_mouse (Mouse coordinate), pts (Terna base), map_data (Mappa discreta pura), mode (Vincolo), fixed_val (Valore fisso), target_type (Stringa target).

Output: La migliore interpolazione geometrica (distanza e dati) tra due punti adiacenti discreti.

Logica interna: Cerca i due punti discretizzati più vicini al cursore e determina lo span di interpolazione. Divide il differenziale parametrico in 15 sottosegmenti ed esegue il calcolo dinamico, estraendo il pacchetto geometrico con la minor deviazione euclidea.

ELL3P_Solve_Exact_Node_Algebraic

Firma: (ELL3P_Solve_Exact_Node_Algebraic p_target pts mode fixed_val active_node_type)

Scopo: Trovare radici puramente analitiche in modalità Pre-Flight (silente) per stazionare sui Nodi base del LUSNAP.

Input: p_target (Coordinate del Nodo spaziale), pts (Terna base), mode (Vincolo Operativo), fixed_val (Valore fisso), active_node_type (Tipologia di Nodi bersaglio).

Output: Lista di soluzioni analiticamente esatte e validate (errore inferiore a 1e-4) che collidono sul Nodo.

Logica interna: Implementa il metodo di esplorazione bruta dell'intero spettro parametrico. Isola i flessi discendenti di errore tramite comparazioni con i vicini e aggredisce il minimo locale con passate di precisione microscopiche (1000 step) restituendo solo i risultati validi spazialmente.

ELL3P_Universal_Loop

Firma: (ELL3P_Universal_Loop pts mode_idx f_val map_C ents_C map_F ents_F map_V ents_V exact_V_map exact_F_map)

Scopo: Gestore asincrono dell'I/O (LUSNAP e Ciclatore) per l'applicazione Ellisse.

Input: Database dei Nodi calcolati, Mappe visive e parametriche.

Output: Side effect di renderizzazione attiva.

Logica interna: Esegue un loop iterativo. Scansiona input testuali per variare lo stato dei gruppi CAD, oppure processa il posizionamento spaziale per innescare l'interpolazione a schermo (sia lato Luogo Continuo, sia lato Nodo Esatto).

F

F_func

Firma: (F_func th)

Scopo: Calcolare analiticamente i parametri esatti dell'unica parabola passante per 3 punti, forzando il primo come Vertice.

Input: pt_v (Vertice), pt_1 (Punto di passaggio 1), pt_2 (Punto di passaggio 2).

Output: Lista (Vertice Fuoco) o nil.

Logica interna: Sweep a 180à con dicotomica locale di precisione per trovare lo zero-crossing dell'equazione continua. (defun PAR3P_Calc_Par_V_2P (pt_v pt_1 pt_2 / x1 y1 x2 y2 step_val idx roots th1 th2 f1 f2 th_mid f_mid best_root best_a cos_th sin_th u1_best v1_best v2_best focal_dist F_wcs) (setq x1 (- (car pt_1) (car pt_v)) y1 (- (cadr pt_1) (cadr pt_v)) x2 (- (car pt_2) (car pt_v)) y2 (- (cadr pt_2) (cadr pt_v)))

G

Geom_1Pu_Simm_PuC

Firma: (Geom_1Pu_Simm_PuC Pu1 Pu2)

Scopo: Calcola il punto simmetrico di un punto rispetto a un perno (punto di riflessione centrale).

Input: Pu1: punto da riflettere. Pu2: perno (centro della simmetria).

Output: Coordinate del punto speculare di Pu1 rispetto a Pu2.

Logica interna: Applica la formula vettoriale della simmetria centrale: P_sym = Pu2 + (Pu2 − Pu1), ovvero somma il vettore dal punto al perno al perno stesso.

Geom_2Pu_Simm_PuC

Firma: (Geom_2Pu_Simm_PuC Pu1 vec len)

Scopo: Genera una coppia di punti simmetrici rispetto a un centro, disposti equidistanti lungo una direzione data.

Input: Pu1: punto centro della simmetria. vec: versore della direzione di disposizione. len: semi-distanza (offset) dal centro.

Output: Lista di due punti (Pu1+offset, Pu1−offset) simmetrici rispetto a Pu1 lungo vec.

Logica interna: Calcola il vettore offset come `Vet_Sca vec len`, poi restituisce la lista delle due posizioni: centro sommato e sottratto all'offset tramite `Vet_Sum`/`Vet_Sub`.

Geom_3Pts_Core_Improprio

Firma: (Geom_3Pts_Core_Improprio pts Pd1 Pd2)

Scopo: Elaborare e tracciare la parabola passante per 3 punti noti, vincolata da una direzione asintotica (centro improprio).

Input: pts (Lista di 3 coordinate WCS per i punti fissi), Pd1, Pd2 (Due coordinate WCS che definiscono il vettore direzionale dell'asse parabolico).

Output: Restituisce T (booleano vero) se il calcolo ha successo, nil in caso di configurazione impossibile.

Logica interna: Evidenzia a schermo l'asse direzionale selezionato. Espande il set iniziale a 5 punti sfruttando

`Calcola_Parabola`. Con la cinquina completa, estrae le invarianti geometriche primarie (Vertice e Fuoco) per ricavare il vettore unitario dell'asse. Infine, calcola le proiezioni ortogonali dei 3 punti originari sull'asse per generare i rispettivi punti simmetrici, completando il tracciamento grafico della parabola normalizzata.

Geom_3Pts_Core_Proprio

Firma: (Geom_3Pts_Core_Proprio pts center_pt)

Scopo: Elaborare e tracciare la conica passante per 3 punti noti e un centro proprio (ellisse o iperbole).

Input: pts (Lista di 3 coordinate WCS per i punti di passaggio fissi), center_pt (Coordinata WCS del centro proprio imposto dall'utente).

Output: Restituisce T (booleano vero) se il calcolo algebrico ha successo e la curva viene disegnata, nil in caso di degenerazione geometrica.

Logica interna: Traccia graficamente il punto centrale selezionato sul layer di input. Invoca il risolutore matriciale `Solve_And_Draw_Centered_3x3` per derivare l'equazione della conica. Se il calcolo va a buon fine, genera per simmetria i 3 punti opposti rispetto al centro (convalidando l'omotetia) e restituisce un flag di successo, altrimenti intercetta l'errore matematico avvisando l'utente a console.

Geom_3Pts_Master_Engine

Firma: (Geom_3Pts_Master_Engine pts old_os)

Scopo: Gestire l'interfaccia utente dinamica (dashboard contestuale) e il ciclo di esecuzione per la risoluzione interattiva della conica a 3 punti.

Input: pts (Lista di 3 coordinate WCS dei punti fissi), old_os (Valore OSMODE originario di AutoCAD da ripristinare ai prompt).

Output: Nessun output formale. Pilota l'intero flusso grafico finchà l'utente non termina la sessione.

Logica interna: Inizializza le variabili di stato (Centro, Guide, Modalità di Sostituzione) e disegna le guide preliminari. Innesca un ciclo `while` infinito catturando l'input dell'utente con opzioni testuali (initget). Attraverso un costrutto `cond`, smista le azioni: interruttori logici per cambiare modalità operativa, oppure input di coordinate 3D che innescano, a seconda dello stato di `opt_c`, il calcolo del ramo Proprio (centro cliccato) o Improprio (vettore direzionale). Raccoglie ed elimina al volo le entità generate per garantire il refresh grafico (Modalità Sostituisci).

Geom_3Pts_Update_Guides

Firma: (Geom_3Pts_Update_Guides pts opt_g opt_c old_guides)

Scopo: Generare e aggiornare dinamicamente il set di linee guida visive (mediane estese) del triangolo base durante l'esplorazione LUSNAP.

Input: pts (Lista di 3 coordinate WCS del triangolo), opt_g (Stringa di stato per le guide: "ON" o "OFF"), opt_c (Stringa di stato per la topologia: "Proprio" o "Improprio"), old_guides (Lista di ename delle vecchie linee guida da eliminare, se presenti).

Output: Lista contenente i nuovi ename delle entità linea appena tracciate (o nil se la generazione à soppressa dalle opzioni).

Logica interna: Esegue innanzitutto lo svuotamento del database cancellando le entità passate in `old_guides`. Se l'opzione grafica à attiva e si opera in modalità "Centro Proprio", calcola i punti medi dei tre lati, determinando una lunghezza limite (d_max) basata sul perimetro per tracciare estensioni simmetriche fisse. Disegna infine le tre mediane intersecanti, utili all'utente per localizzare il baricentro o punti notevoli.

Geom_5P_Update_Guides

Firma: (Geom_5P_Update_Guides pts opt_g old_guides)

Scopo: Aggiorna in tempo reale le entità guida nell'ambiente interattivo a 5 punti, ridisegnandole ogni volta che il set di punti cambia.

Input: pts: lista corrente di punti selezionati (2–4). opt_g: stato guide "ON"/"OFF". old_guides: lista delle entità guida precedenti.

Output: Lista delle nuove entità guida create. Effetto collaterale: vecchie guide eliminate, nuove disegnate.

Logica interna: Elimina le entità vecchie con `Util_Del_Ents_List`. Se opt_g = "ON" e ci sono ≥ 2 punti, ridisegna le guide appropriate (segmenti di connessione,

quadrangolo, parabole guida) in funzione del numero di punti disponibili.

Geom_Analyze_Ipe_Topology

Firma: (Geom_Analyze_Ipe_Topology pts)

Scopo: Analizza la configurazione topologica di 4 punti per determinare quante e quali iperboli equilatere li soddisfano: nessuna (degenere), infinite (ortocentrico) o una unica.

Input: pts: lista di 4 punti.

Output: Lista (tipo conteggio messaggio): tipo è 'DEGENERATE, 'ORTHOCENTRIC o 'UNIQUE; conteggio è 0, -1 (infinite) o 1.

Logica interna: Verifica in sequenza: P_Has_Collinear_Triplet (3 punti allineati → degenere), P_Is_Orthocentric_System (sistema ortocentrico → infinite soluzioni), altrimenti restituisce 'UNIQUE.

Geom_Calc_Par_V_2P

Firma: (Geom_Calc_Par_V_2P V P1 P2)

Scopo: Calcolare analiticamente il Fuoco e l'asse di simmetria di una parabola spaziale conoscendo le coordinate del Vertice e di 2 punti.

Input: V (Coordinate WCS Vertice), P1, P2 (Coordinate WCS dei punti sul ramo).

Output: Lista contenente (Vertice Fuoco) in WCS, oppure nil se la configurazione risulta allineata o degenere.

Logica interna: Delega la ricerca iterativa dell'angolo all'helper dicotomico. Ricavato l'asse, proietta un punto base nel sistema locale e ne spacchetta le coordinate. Calcola in sicurezza l'apertura 'a1', verifica la curva ed esegue la back-trasformazione del Fuoco nel WCS assoluto.

Geom_Check_Trapezoid_Direction

Firma: (Geom_Check_Trapezoid_Direction pts)

Scopo: Verifica se i lati paralleli del trapezio definito da 4 punti sono orientati nella direzione attesa (primo o secondo paio di lati opposti).

Input: Pu1, Pu2, Pu3, Pu4: vertici del trapezio. dir: direzione di riferimento attesa.

Output: T se il paio di lati paralleli è nella direzione dir; nil altrimenti.

Logica interna: Calcola la direzione del primo paio (P1-P2 vs P3-P4) e del secondo (P2-P3 vs P4-P1), poi confronta con dir tramite P_Are_Vectors_Parallel_2D.

Geom_ConCen_4PC

Firma: (Geom_ConCen_4PC cen pts)

Scopo: Calcola i coefficienti della conica passante per 4 punti con centro fisso dato, usando una strategia a torneo 3-su-4 per massimizzare la robustezza numerica.

Input: cen: coordinate del centro fisso. pts: lista di 4 punti di passaggio.

Output: Lista (errore_minimo coefficienti_ottimali) della migliore soluzione trovata; nil in caso di fallimento.

Logica interna: Prova tutte le combinazioni C(4,3) di triplette di punti, risolve il sistema lineare per ogni tripletta con `Math_Conics_From_3Pu_And_Center`, valuta l'errore sul quarto punto e restituisce i coefficienti che minimizzano l'errore residuo.

Geom_Conic_4Pf_Engine

Firma: (Geom_Conic_4Pf_Engine P1 P2 P3 P4 Cg geo_packet opt_m opt_e last_ents)

Scopo: Motore grafico per la vettorializzazione a schermo delle coniche associate a un Fuoco target.

Input: P1..P4 (Vertici quadrangolo base), Cg (Baricentro), geo_packet (Lista dati della conica target), opt_m (Modo disegno), opt_e (Estensione asintoti), last_ents (Entità precedenti).

Output: Lista di handle ENAME generati (tramite side-effect di Safe_Collect_Ents).

Logica interna: Valuta la modalità Sostituisci/Aggiungi pulendo il buffer grafico precedente. Estrae la topologia ('PARABOLA, 'ELLISSE, 'IPERBOLE) e i parametri WCS dal pacchetto dati tramite scomposizione mirata (cadr, caddr, cddr). Disegna la Parabola per direzione asse, l'Ellisse via semiassi e impone all'Iperbole un calcolo

dinamico dell'asintoto per evitare rami mozzi, passando il tutto ai driver di disegno finale.

Geom_Conic_4Pv_Engine

Firma: (Geom_Conic_4Pv_Engine P1 P2 P3 P4 Cg geo_packet opt_m opt_e last_ents)

Scopo: Motore grafico specializzato per la vettorializzazione a schermo delle coniche associate a un Vertice target.

Input: P1..P4 (Vertici quadrangolo base), Cg (Baricentro), geo_packet (Dati decodificati della conica), opt_m (Modo sostituzione/accumulo), opt_e (Moltiplicatore estensione), last_ents (Memoria buffer entità).

Output: Lista di handle ENAME degli oggetti vettoriali inseriti nel database DWG.

Logica interna: Esegue preliminarmente il flush delle entità precedenti se opt_m à "Sostituisci". Effettua uno switch-case analitico sulla tipologia della conica target: invia la Parabola al risolutore per raggio/direzione, l'Ellisse al renderer per semiassi estratti, e l'Iperbole a un limitatore asintotico che calcola la massima proiezione ortogonale dei 4 punti base sull'asse focale per generare il "dummy point" di estensione, scalandolo proporzionalmente al fattore opt_e.

Geom_Conic_Action_MinEccentricity

Scopo: Motore di azione per trovare la conica a eccentricità minima passante per 4 punti: nel fascio di coniche K1+λK2, sceglie il parametro λ che minimizza il rapporto tra gli autovalori (B^2-4AC minimale → conica più vicina al cerchio).

Input: pts: lista di 4 punti di vincolo (catturati dalla closure).

Output: Effetto collaterale: conica a eccentricità minima disegnata nel documento. Restituisce princ.

Logica interna: Normalizza i 4 punti. Costruisce K1=L1·L2 e K2=L3·L4 (prodotti delle due diagonali). Imposta il problema di minimizzazione dell'eccentricità come equazione quadratica in λ: risolve $Ka\cdot\lambda^2+Kb\cdot\lambda+Kc=0$ con Math_Solve_Quadratic. Per ogni λ candidato valuta il tipo di conica e sceglie il λ migliore (minima eccentricità). Chiama Solve_And_Draw_From_Coeffs con i coefficienti K_final.

Geom_Conic_Action_Par_Direct

Firma: (Geom_Conic_Action_Par_Direct pts)

Scopo: Motore di azione per la parabola passante per 4 punti con il metodo diretto: classifica la topologia del quadrangolo, sceglie la diagonale corretta come asse naturale e chiama il disegnatore appropriato.

Input: pts: lista di 4 punti di vincolo.

Output: Effetto collaterale: parabola disegnata nel documento. Restituisce princ.

Logica interna: Classifica i 4 punti con Math_Classify_Forma_4Pu. Se la configurazione è concava, ortocentrica o ha simmetria che rende la parabola impossibile, segnala e termina. Altrimenti chiama Draw_ConPar_4P_ConVer_R con le coppie diagonali appropriate.

Geom_Conic_Center_Proprio_Action

Firma: (Geom_Conic_Center_Proprio_Action p1 p2 p3 p4)

Scopo: Motore di azione per la conica con centro finito (ellisse o iperbole) passante per 4 punti, con centro da minimizzare: trova il centro ottimale e calcola i coefficienti della conica.

Input: pts: lista di 4 punti di vincolo.

Output: Lista (coefficienti centro scale tipo_conica) della soluzione ottimale; nil se fallisce.

Logica interna: Normalizza i punti, costruisce il fascio K1+λK2, risolve l'equazione quadratica in λ che minimizza l'eccentricità (condizione $(A-C)\cdot\sin 2\theta = B\cdot\cos 2\theta$), poi estrae i parametri con Math_Extract_Geom_Data_From_Coeffs.

Geom_Count_Parallel_Pairs

Firma: (Geom_Count_Parallel_Pairs pts)

Scopo: Contare il numero di coppie di lati paralleli in un quadrangolo di 4 punti, garantendo un ritorno numerico intero anche in caso di figure concave o degeneri.

Input: pts (Lista di 4 punti WCS).

Output: Numero intero (0, 1 o 2) indicante le coppie di parallelismi trovate.

Logica interna: Applica la lista dei punti direttamente alla funzione core 'Math_Conta_Paral_4Pu' tramite 'apply', eliminando variabili locali e passaggi intermedi. Utilizza l'operatore 'or' per intercettare l'eventuale ritorno nullo (nil) del classificatore e forzarlo a 0, assicurando la coerenza del tipo di dato in uscita.

Geom_Co_Si_Ang

Firma: (Geom_Co_Si_Ang ang)

Scopo: Precompila la coppia (coseno, seno) per un angolo in radianti, da passare ai motori di rotazione per evitare ricalcoli ripetuti.

Input: ang: angolo in radianti.

Output: Lista (cos(ang) sin(ang)).

Logica interna: Chiamata semplice a (cos ang) e (sin ang) con restituzione come lista.

Geom_Draw_Dima_Boundaries

Firma: (Geom_Draw_Dima_Boundaries pA pB lay)

Scopo: Disegna i confini topologici (bordi del dominio) della sagoma Dima, scalandoli e ruotandoli per adattarli ai due punti di ancoraggio forniti dall'utente.

Input: pA, pB: punti di ancoraggio che definiscono la posizione e la scala del dominio. lay: layer di destinazione.

Output: Lista di entità disegnate (segmenti del bordo). Effetto collaterale: confini Dima disegnati nel documento.

Logica interna: Calcola l'angolo e la scala dal segmento pA-pB rispetto al segmento base della Dima. Applica la trasformazione affine (rotazione + scala + traslazione) a ogni punto del bordo, poi disegna i segmenti trasformati.

Geom_Draw_Quadrangolo_Smart

Firma: (Geom_Draw_Quadrangolo_Smart pts lay)

Scopo: Disegna il quadrangolo definito da 4 punti connettendoli con segmenti nell'ordine topologico corretto determinato dalla classificazione geometrica.

Input: pts: lista di 4 punti. lay: layer di destinazione.

Output: Lista di entità segmento disegnate. Effetto collaterale: quadrangolo disegnato.

Logica interna: Classifica i 4 punti, determina l'ordine topologico (CCW o specifico per il tipo), poi disegna 4 segmenti di chiusura con `command "._LINE"` sul layer specificato.

Geom_Draw_Trapezoid_Smart

Firma: (Geom_Draw_Trapezoid_Smart pts_mid v_dir pts_ref lay_locus)

Scopo: Disegna un trapezio (o altra figura quadrangolare) connettendo 4 punti con segmenti nell'ordine topologico corretto.

Input: pts: lista di 4 punti. lay: nome del layer di destinazione.

Output: Lista di 4 entità segmento disegnate (ename).

Logica interna: Classifica la topologia del quadrangolo con `Math_Classify_Forma_4Pu`, poi connette i 4 vertici nell'ordine restituito con 4 chiamate `command "._LINE"` sul layer specificato.

Geom_Eval_Matrix_Parallelism

Firma: (Geom_Eval_Matrix_Parallelism M P)

Scopo: Valutare la condizione analitica di parallelismo tra il vettore normale a una conica in un punto P e il vettore che congiunge il centro della conica a P stesso.

Input: M (Lista dei primi 5 coefficienti dell'equazione generale: a, b, c, d, e), P (Coordinate 2D o 3D del punto).

Output: Numero reale (Float) corrispondente al prodotto vettoriale 2D (determinante incrociato).

Logica interna: Sfrutta le derivate parziali per estrarre la normale e ne calcola il cross-product 2D con il vettore posizione. L'annullamento identifica i vertici.

Geom_Eval_Par_Error

Firma: (Geom_Eval_Par_Error pts_rel c s)

Scopo: Valuta l'errore geometrico di una parabola con asse inclinato di ang per due punti fissi: quantifica lo scarto dalla condizione di parabola esatta.

Input: pts_rel: lista di punti trasformati nel sistema locale. c: coseno dell'angolo dell'asse. s: seno dell'angolo dell'asse.

Output: Valore numerico reale dell'errore (0.0 = parabola esatta, valori alti = grande discostamento).

Logica interna: Ruota i punti con `Util_Rot_List_2D_CS` usando c e s, poi valuta il residuo dell'equazione parabolica $y^2 = x$ applicata ai due punti ruotati, restituendo la differenza dei prodotti incrociati.

Geom_Extract_Par_World

Firma: (Geom_Extract_Par_World A B C origin theta)

Scopo: Converte i parametri della parabola dal sistema locale dell'asse al sistema globale WCS di AutoCAD, applicando la trasformazione inversa.

Input: par_local: parametri della parabola nel sistema locale (fuoco, vertice, asse). origin: origine del sistema locale. ang: orientamento dell'asse.

Output: Struttura dati con fuoco, vertice e asse della parabola in coordinate WCS.

Logica interna: Applica Util_Trans_List_Loc_To_Glo ai punti significativi (fuoco, vertice) della parabola, trasformandoli dal sistema ruotato al sistema globale.

Geom_Find_Parabolic_Angles

Firma: (Geom_Find_Parabolic_Angles Pu1 Pu2 Pu3 Pu4)

Scopo: Trova gli angoli dell'asse parabola che soddisfano i vincoli di passaggio per i punti dati, eseguendo una scansione angolare e raccogliendo tutti gli zeri della funzione di errore.

Input: pts: lista di punti di vincolo. scan_step: passo di scansione in radianti.

Output: Lista di angoli (in radianti) per cui la parabola passa per tutti i punti dati.

Logica interna: Scansiona l'intervallo [0, π] con passi scan_step, valuta Math_Calc_Par_Global_Error su ogni angolo e raccoglie i punti di cambio segno come candidati angoli soluzione.

Geom_Format_Map_Data

Firma: (Geom_Format_Map_Data v_list f_type g_data)

Scopo: Strutturare una lista di punti geometrici grezzi trasformandola in pacchetti dati completi per il motore di aggancio LUSNAP e le meta-mappe.

Input: v_list (Lista di coordinate 3D dei punti, es. fuochi o vertici), f_type (Simbolo della topologia, es. 'ELLISSE o 'IPERBOLE), g_data (Lista dei parametri geometrici base della conica: centro, asse, semiassi).

Output: Lista di sub-liste. Ogni elemento à un record dati formattato come: (CoordinatePunto TipoConica ParametriGeometrici...).

Logica interna: Utilizza un iteratore funzionale (mapcar) combinato con una funzione anonima (lambda) per fondere rapidamente la singola coordinata spaziale con la sua "carta d'identità" tipologica e geometrica, assemblando un payload informativo (la mappa) pronto ad essere digerito dal risolutore di affinamento al momento del clic.

Geom_Get_Par_Foci_List

Firma: (Geom_Get_Par_Foci_List P1 P2 P3 P4 tipo_f)

Scopo: Calcolare e raggruppare le coordinate WCS dei fuochi delle parabole inscritte nella quaterna base, generando la relativa mappa dati per il LUSNAP.

Input: P1, P2, P3, P4 (Coordinate WCS dei vertici del quadrangolo base), tipo_f (Simbolo della topologia geometrica).

Output: Lista a due elementi formattata come (Lista_Coordinate_Fuochi Lista_Meta_Mappa_Parabole), oppure (nil nil) se la topologia non ammette soluzioni paraboliche valide.

Logica interna: Esclude a priori figure incompatibili o degeneri (Concavi e Parallelogrammi). Estrae gli angoli di inclinazione asintotica e normalizza il sistema spaziale centrandolo sul baricentro e ruotandolo sull'asse di inviluppo principale (PCA). Per ogni angolo, calcola i coefficienti polinomiali (a,b,c) risolvendo un determinante ciclico per evitare divisioni per zero. Integra un filtro anti-ridondanza per scartare le parabole "gemelle" (sfalsate di 180à), calcola il fuoco e lo

ritrasforma nel sistema WCS assoluto per l'aggancio a schermo.

Geom_Get_Par_Vertices_List

Scopo: Calcolare analiticamente le posizioni assolute dei vertici delle parabole limitate dalla quaterna base, strutturando la mappa LUSNAP di interfaccia.

Input: P1, P2, P3, P4 (Coordinate WCS dei vertici base), tipo_f (Simbolo della classificazione topologica).

Output: Lista a due elementi formattata come (Lista_Coordinate_Vertici Lista_Meta_Mappa), oppure (nil nil) in caso di figure degeneri o asintoti incalcolabili.

Logica interna: Mette in sicurezza il calcolo bloccando le configurazioni precluse (Concavi, Parallelogrammi). Per evitare divisioni per zero ed eccezioni trigonometriche, esegue l'Analisi delle Componenti Principali (PCA), allinea localmente la nuvola di punti e risolve le curve tramite coefficienti di forma estraendo i vertici (Xv, Yv). Applica infine una back-trasformazione rototraslazionale per iniettare i nodi vertice nello spazio WCS definitivo. LOGICA DI FUNZIONAMENTO: 1. Filtro Topologico: Esclude configurazioni concave o parallelogrammi dove la parabola non à definita o à degenere. 2. Normalizzazione Spaziale: Proietta i punti sul piano della conica, trasla nel baricentro (Cg) e allinea il sistema tramite PCA (Principal Component Analysis) per stabilità numerica. 3. Risoluzione Analitica: Per ogni angolo parabolico trovato, ruota il sistema, calcola i coefficienti ($y = ax^2 + bx + c$) e individua il vertice locale (Xv, Yv). 4. Filtro "Twin": Evita doppioni geometrici dovuti alla simmetria a 180à degli angoli. 5. Back-Transformation: Riporta il vertice dal sistema locale ruotato alle coordinate WCS.

Geom_Get_Projected_Center

Firma: (Geom_Get_Projected_Center pt_ucs vla_list extend)

Scopo: Calcola la proiezione del centro geometrico di una conica sul luogo geometrico più vicino, per associarlo alla conica corretta nei contesti multi-ipotesi.

Input: cen: coordinate del centro calcolato. locus_ents: lista di entità che formano il luogo geometrico.

Output: Coordinate del punto del luogo geometrico più vicino al centro dato.

Logica interna: Itera sulle entità del luogo calcolando con `distance` la distanza dal centro; restituisce le coordinate dell'entità più vicina usando aggiornamento del minimo.

Geom_Helper_Find_Par_Axis_Angle

Firma: (Geom_Helper_Find_Par_Axis_Angle V P1 P2 a_min a_max)

Scopo: Eseguire la ricerca dicotomica per individuare l'inclinazione dell'asse di simmetria della parabola passante per il vertice e due punti.

Input: V (Vertice WCS), P1, P2 (Punti di passaggio WCS), a_min, a_max (Limiti angolari di ricerca in radianti).

Output: Valore angolare in radianti (Real) corrispondente all'inclinazione dell'asse di simmetria locale.

Logica interna: Itera 50 volte dimezzando il dominio angolare. Trasforma le coordinate dei punti nel sistema locale ruotato e confronta i coefficienti di apertura (a = y/x^2). Aggiusta i limiti a_min e a_max in base al segno dello scarto per forzare la convergenza dell'angolo.

Geom_Ipe_4P_Eq

Firma: (Geom_Ipe_4P_Eq Pu1 Pu2 Pu3 Pu4)

Scopo: Calcola i parametri geometrici dell'iperbole equilatera passante per 4 punti: centro, direzione dell'asse semi-principale, semi-asse.

Input: Pu1, Pu2, Pu3, Pu4: quattro punti di vincolo.

Output: Lista (centro versore_asse semi_asse) dell'iperbole equilatera; nil se degenere.

Logica interna: Costruisce il fascio K1+λK2, risolve per il λ che soddisfa il vincolo A+C=0 (iperbole equilatera), poi estrae i parametri geometrici con Math_Extract_Equilateral_Data.

Geom_Master_Analysis_4P

Firma: (Geom_Master_Analysis_4P pts)

Scopo: Eseguire l'analisi topologica completa della quaterna di base e calcolarne i centri notevoli primari.

Input: pts (Lista dei 4 punti originali).

Output: Lista strutturata (Tipo PuntiOrdinati Baricentro IntersezioneDiagonali).

Logica interna: Sfrutta il classificatore per identificare la topologia, estrae il baricentro e valuta l'intersezione logica della quaterna.

Geom_Master_Draw_Framework

Firma: (Geom_Master_Draw_Framework class P_diag)

Scopo: Smistatore grafico (Router Polimorfico) primario per l'infrastruttura.

Input: class (Lista Classificazione topologica: tipo, pts_ord), P_diag (Intersezione diagonali).

Output: Nessuno (Side-effect: genera le linee di costruzione base sul DWG).

Logica interna: Delega il rendering dei poligoni base all'Handler specifico. Passa i parametri in modo standardizzato (pts_ord, opzionali, layer) e isola i casi degeneri.

Geom_MinMax_Dist_LisPu

Firma: (Geom_MinMax_Dist_LisPu pts ref_pt opz)

Scopo: Calcolare e restituire le distanze minime e massime (e i relativi punti) da un set di coordinate.

Input: pts (Lista di punti), ref_pt (Punto singolo di riferimento, se nil esegue "Tutti contro Tutti"), opz (Intero da 1 a 7 per il formato di output).

Output: A seconda del parametro 'opz', restituisce distanze singole, coppie di punti o liste strutturate dei risultati estremi.

Logica interna: Utilizza un motore a doppio ramo. Se 'ref_pt' esiste, calcola le distanze radiali dal fulcro. Se à nil, innesca un doppio ciclo per confrontare ogni punto con i successivi, scartando i calcoli speculari. Salva i record in variabili scalari individuali per non pesare sulla RAM nei cicli stretti. OUTPUT (opz): 1: Distanza minima 2: Distanza massima 3: Lista coppia punti a dist minima (P_a P_b) - NB: se ref_pt esiste, P_a à ref_pt 4: Lista coppia punti a dist massima (P_c P_d) 5: Lista delle due distanze (Dist_Min Dist_Max) 6: Lista di 2 liste binate ((P_a P_b) (P_c P_d)) 7: Lista di 2 liste a terna ((P_a P_b Dist_Min) (P_c P_d Dist_Max))

Geom_Ordina_CCW_4Pu

Firma: (Geom_Ordina_CCW_4Pu Pu1 Pu2 Pu3 Pu4)

Scopo: Riordina specificamente 4 punti in senso antiorario (CCW) rispetto al loro baricentro, garantendo la coerenza topologica per il quadrangolo convesso.

Input: Pu1, Pu2, Pu3, Pu4: coordinate dei quattro vertici (in qualsiasi ordine).

Output: Lista ordinata (P_SW P_NW P_NE P_SE) o altro ordine CCW dei 4 punti.

Logica interna: Calcola il baricentro, poi ordina i 4 punti per angolo polare crescente con atan2 e li restituisce come lista CCW.

Geom_Ordina_Concavo_4Pu

Firma: (Geom_Ordina_Concavo_4Pu Pu1 Pu2 Pu3 Pu4)

Scopo: Riordina 4 punti in configurazione concava mettendo il punto interno come ultimo elemento, per garantire la coerenza topologica nelle costruzioni concave.

Input: Pu1, Pu2, Pu3, Pu4: coordinate dei quattro punti (uno è interno agli altri tre).

Output: Lista (P_ext1 P_ext2 P_ext3 P_interno) con il punto interno in ultima posizione.

Logica interna: Usa P_Conca_4Pu per trovare quale punto è interno, poi riordina mettendo i tre esterni in ordine CCW e il punto interno alla fine.

Geom_Precalc_Node_Conics

Firma: (Geom_Precalc_Node_Conics P1 P2 P3 P4)

Scopo: Precalcolare analiticamente le equazioni di tutte le coniche del fascio che possiedono uno dei vertici del quadrangolo come punto di tangenza/asse principale.

Input: P1, P2, P3, P4 (Coordinate 3D dei vertici del quadrangolo).

Output: master_solution_list (Lista nidificata per ogni nodo P1..P4, contenente i geo_packet validi).

Logica interna: Centra il sistema sul baricentro, calcola le coniche degeneri K1 e K2 e risolve il sistema cubico basato sul parallelismo matrice-vettore per isolare le radici (lambda) corrispondenti alle coniche vertice.

Geom_Refine_Internal_Bracket

Firma: (Geom_Refine_Internal_Bracket res_list)

Scopo: Identifica il "bracket" (forcella) attorno al minimo di distanza dentro una lista di record angolari, estraendone gli estremi per il raffinamento successivo.

Input: res_list: lista di record (distanza angolo ...) ordinata per angolo.

Output: Lista (a_sinistra a_destra larghezza_spaziale record_minimo) del bracket; nil se la lista ha meno di 3 elementi.

Logica interna: Trova il record con distanza minima (best_i). Estrae il precedente (prev_item) e il successivo (next_item) come estremi del bracket. Calcola la larghezza spaziale come distanza tra i vertici di prev e next.

Geom_Refine_Internal_Bracket_Fuochi

Firma: (Geom_Refine_Internal_Bracket_Fuochi res_list)

Scopo: Analizzare i risultati della scansione per individuare il "bracket" (l'intervallo angolare) in cui si trova il fuoco con la distanza minima.

Input: res_list (Lista di risultati strutturata e ordinata proveniente da Geom_Refine_Internal_Scan_Fuochi).

Output: Lista contenente i dati del bracket: (Angolo_Precedente Angolo_Successivo Distanza_Spaziale_Tra_Fuochi Dati_Miglior_Punto), oppure nil.

Logica interna: Identifica la sottolista avente la distanza minima dal target. Estrae quindi l'elemento precedente e successivo nell'array per definire il nuovo range angolare (il "bracket"). Calcola inoltre l'ampiezza spaziale effettiva misurando la distanza 3D tra il fuoco del punto precedente e quello del punto successivo.

Geom_Refine_Internal_Recursive

Firma: (Geom_Refine_Internal_Recursive ang_start ang_end depth max_depth steps pts_3 pt_target threshold)

Scopo: Raffinamento ricorsivo ad albero della parabola ottimale: restringe progressivamente l'intervallo angolare attorno al minimo finché la larghezza spaziale del bracket è inferiore alla soglia o si raggiunge la profondità massima di ricorsione.

Input: ang_start, ang_end: intervallo corrente. depth: profondità corrente. max_depth: profondità massima (tipicamente 15). steps: passi per Geom_Refine_Internal_Scan. pts_3: tre punti base. pt_target: punto target. threshold: soglia di convergenza.

Output: Lista (vertice fuoco angolo) della parabola ottimale; nil se nessuna soluzione.

Logica interna: Chiama Geom_Refine_Internal_Scan per la scansione dell'intervallo corrente. Chiama Geom_Refine_Internal_Bracket per trovare il bracket. Se spatial_width < threshold o depth ≥ max_depth, restituisce il vincitore. Altrimenti chiamata ricorsiva con il bracket ristretto e depth+1.

Geom_Refine_Internal_Recursive_Fuochi

Firma: (Geom_Refine_Internal_Recursive_Fuochi ang_start ang_end depth max_depth steps pts_3 pt_target threshold)

Scopo: Motore di ricerca binaria ricorsiva per stringere iterativamente l'angolo fino a convergere sulla coordinata focale esatta.

Input: ang_start, ang_end (Range angolare attuale), depth (Livello di ricorsione corrente), max_depth (Limite massimo di ricorsione), steps (Passi di scansione per ciclo), pts_3 (3 punti base WCS), pt_target (Punto bersaglio WCS), threshold (Tolleranza spaziale di convergenza).

Output: Lista di 3 elementi contenente la soluzione convergente: (Vertice Fuoco Angolo_Radianti), oppure nil se fallisce.

Logica interna: Invia il range corrente allo scanner (Geom_Refine_Internal_Scan_Fuochi) e ne fa valutare il risultato al selettore. Se l'ampiezza spaziale del bracket

scende sotto la threshold tollerata, o se si raggiunge il limite di ricorsione max_depth, interrompe il ciclo e restituisce il miglior risultato. Altrimenti, richiama se stessa passando i nuovi angoli ristretti (zoom in).

Geom_Refine_Internal_Scan

Firma: (Geom_Refine_Internal_Scan a_start a_end n_steps pts_3 pt_target)

Scopo: Scansione di raffinamento su un sotto-intervallo angolare [a_start, a_end] con n_steps passi uniformi, calcolando per ogni angolo la distanza del vertice della parabola dal punto target, per isoprare il minimo.

Input: a_start, a_end: estremi dell'intervallo angolare. n_steps: numero di passi. pts_3: lista di 3 punti base. pt_target: punto di cui minimizzare lo scarto.

Output: Lista di record (distanza_dal_target angolo vertice fuoco) ordinata per angolo; record vuoto se nessuna soluzione trovata nell'intervallo.

Logica interna: Suddivide [a_start, a_end] in n_steps+1 punti equidistanti. Per ogni angolo: chiama Calcola_Parabola → Solve_Par_Geometry, misura distance(vertice, pt_target) e accumula il record. Ordina il risultato per angolo con vl-sort.

Geom_Refine_Internal_Scan_Fuochi

Firma: (Geom_Refine_Internal_Scan_Fuochi a_start a_end n_steps pts_3 pt_target)

Scopo: Eseguire una scansione angolare per valutare la distanza tra il fuoco di una serie di parabole e un punto bersaglio.

Input: a_start, a_end (Angoli in radianti per il range di scansione), n_steps (Numero di divisioni), pts_3 (Lista di 3 punti base WCS), pt_target (Coordinata WCS del punto di mira).

Output: Lista di liste, ordinate in modo crescente per angolo. Ogni sottolista contiene: (Distanza_Fuoco_Target Angolo_Radianti Vertice Fuoco).

Logica interna: Calcola il passo angolare e itera per n_steps + 1 volte. Per ogni angolo, genera un vettore direzionale, calcola i 2 punti asintotici e risolve la parabola. Estrae il Fuoco dal risolutore, ne misura la distanza dal pt_target e accumula i risultati. Infine, riordina la lista crescente in base all'angolo testato.

Geom_Refine_Par_Topological

Firma: (Geom_Refine_Par_Topological pt_target spline_obj branch_data pts_3)

Scopo: Raffinamento topologico della parabola per un punto target: identifica il ramo del luogo a 3 punti più vicino al target, risolve l'ambiguità tra i due rami opposti (ang e ang+π) e lancia il raffinamento ricorsivo nella direzione corretta.

Input: pt_target: punto di vincolo aggiuntivo. spline_obj: entità spline del luogo (per analisi). branch_data: lista di record del luogo scansionato da Geom_Scan_3P_Parabolas_Adaptive. pts_3: tre punti base.

Output: Lista (vertice fuoco angolo_finale) della parabola che passa più vicino a pt_target; nil se nessuna soluzione convergente.

Logica interna: Trova il record in branch_data con distanza minima da pt_target (best_idx). Testa l'angolo best e best+π su Solve_Par_Geometry e sceglie il ramo con distanza minore. Definisce l'intervallo ±0.1 rad attorno all'angolo migliore e chiama Geom_Refine_Internal_Recursive con soglia = max_dist/100000.

Geom_Refine_Par_Topological_Fuochi

Firma: (Geom_Refine_Par_Topological_Fuochi pt_target spline_obj branch_data pts_3)

Scopo: Funzione Wrapper/Entry-point per innescare l'affinamento topologico incentrato sul fuoco parabolico.

Input: pt_target (Punto WCS bersaglio su cui far collassare il fuoco), spline_obj (Dato non usato nel corpo), branch_data (Lista preliminare di rami di parabola validi), pts_3 (Lista dei 3 punti master WCS).

Output: Lista ottimizzata dei parametri parabolici: (Vertice Fuoco Angolo_Direzione).

Logica interna: Estrae preliminarmente dalla branch_data il pacchetto che ha il fuoco più vicino al target per stabilire un angolo base di partenza. Esegue un test di direzionalità valutando l'angolo e il suo opposto (+

pi) per scartare il ramo gemello divergente. Trovata la direzione corretta, definisce un bracket iniziale grezzo di +/- 0.1 radianti e lancia il motore ricorsivo con profondità fissa a 15 cicli.

Geom_Scan_3P_Parabolas_Adaptive

Scopo: Scanner adattivo della famiglia di parabole passanti per 3 punti: scansiona l'intervallo angolare [0°, max_angle] con passo variabile, riducendo il passo nelle zone di alta variazione (vicino ai vertici) per ottenere una copertura completa del luogo geometrico.

Input: pts: lista di 3 punti di vincolo. max_angle: angolo massimo di scansione (gradi). molt: moltiplicatore che controlla la densità della scansione adattiva.

Output: Lista di record (vertice fuoco angolo_asse) di tutte le parabole trovate, ordinata per angolo crescente.

Logica interna: Loop while sull'angolo da 0 a max_angle. Per ogni angolo: calcola Calcola_Parabola e Solve_Par_Geometry. Il passo successivo è moltiplicato per un fattore adattivo basato sulla distanza del vertice dalla nuvola di punti: più il vertice è vicino ai punti, più il passo è piccolo (gradiente fine) per non perdere soluzioni.

Geom_Sort_Points_CCW

Firma: (Geom_Sort_Points_CCW pts)

Scopo: Ordina un insieme di punti in senso antiorario (Counter-Clock-Wise) rispetto al loro baricentro, per garantire la coerenza topologica negli algoritmi.

Input: pts: lista di punti 2D.

Output: Lista di punti riordinata in senso antiorario attorno al baricentro.

Logica interna: Calcola il baricentro, poi ordina i punti per angolo crescente (atan2(y−cy, x−cx)) tramite un ordinamento per chiave.

Geom_Trans_Pt

Firma: (Geom_Trans_Pt Pu1 ang PuC)

Scopo: Trasla un singolo punto di un vettore dato; versione scalar per singolo punto del concetto implementato in lista da Util_Translate_Pts_Global.

Input: pt: coordinate del punto da traslare. offset_vec: vettore di traslazione.

Output: Coordinate del punto traslato.

Logica interna: Delega a Vet_Sum(pt, offset_vec).

get-all-permutations

Firma: (get-all-permutations lst)

Scopo: Calcolare in modo esaustivo tutte le permutazioni possibili (combinazioni che tengono conto dell'ordine) degli elementi contenuti in una lista.

Input: lst (Lista generica contenente gli elementi, di qualsiasi natura o tipo di dato, da permutare).

Output: Lista di liste, dove ogni sottolista rappresenta una singola permutazione unica e completa degli elementi della lista originaria.

Logica interna: Implementa un rigoroso algoritmo ricorsivo. Come caso base, se la lista à vuota restituisce una lista contenente `nil`. Nel passo iterativo, scorre ogni elemento tramite `mapcar`, lo isola rimuovendolo dal set (`vl-remove`), calcola le permutazioni dei rimanenti chiamando se stessa e lo reinserisce in testa (`cons`) a ciascuna di esse. Appiattisce l'albero risultante con `apply 'append`.

get-max-abs-projection-on-axis

Firma: (get-max-abs-projection-on-axis pts origin axis_vec)

Scopo: Calcola la proiezione massima (in valore assoluto) di una lista di punti su una direzione d'asse data, usata per stimare l'estensione della conica.

Input: pts: lista di punti. axis_dir: versore della direzione d'asse.

Output: Valore reale della massima proiezione in valore assoluto.

Logica interna: Per ogni punto calcola il prodotto scalare con axis_dir (proiezione), prende il valore assoluto e restituisce il massimo tra tutti i punti.

get-oblique-coords

Firma: (get-oblique-coords Pu1 Pu2 Pu3 Pu4)

Scopo: Calcolare le coordinate (proiezioni scalari) di un punto spaziale rispetto a un sistema di riferimento obliquo, definito da un'origine e da due assi direzionali non necessariamente ortogonali.

Input: Pu1 (Coordinate 3D del punto target), Pu2 (Coordinate 3D dell'origine del sistema locale), Pu3, Pu4 (Punti spaziali che, congiunti a Pu2, definiscono la direzione dei due assi obliqui).

Output: Lista estesa contenente le coordinate scalari trovate e le componenti dei versori, nel formato: (Sol_X Sol_Y ux uy vx vy). Restituisce nil se la trasformazione à impossibile (es. assi paralleli).

Logica interna: Calcola i versori unitari dei due assi (u, v) e il vettore distanza tra il punto Pu1 e l'origine Pu2. Imposta le equazioni per le proiezioni e delega la soluzione al motore `solve-system-cramer-2x2`. In caso di successo, concatena tramite `append` i coefficienti calcolati e i versori degli assi base.

get-pascal-center-frame

Firma: (get-pascal-center-frame pts)

Scopo: Costruisce e restituisce il "frame" del centro di Pascal: le informazioni geometriche (centro, tipo, statistiche) derivate dall'analisi di tutti i tentativi dell'algoritmo di Pascal sui 5 punti dati.

Input: logbook: lista di tutti i record Log_Pascal_Iteration accumulati.

Output: Struttura dati riepilogativa con: tipo di conica prevalente, centro calcolato per voto di maggioranza, statistiche (n ellissi, n iperboli, n parabole, n errori).

Logica interna: Aggrega i risultati del logbook usando Util_Filter_And_Count_By_Nth per estrarre i voti per tipo, determina la maggioranza, calcola il centro medio con Util_Media_lisNum separatamente su X e Y.

get-unit-vector-from-pts

Firma: (get-unit-vector-from-pts Pu1 Pu2)

Scopo: Determinare la direzione vettoriale pura (versore unitario) che congiunge spazialmente due punti, normalizzandone la lunghezza matematica a 1.0.

Input: Pu1 (Coordinate 3D del punto di origine), Pu2 (Coordinate 3D del punto di destinazione del vettore).

Output: Lista di coordinate reali (3D) rappresentante il versore. Restituisce nil o zero geometrico se i punti sono coincidenti (comportamento ereditato da Vet_Uni).

Logica interna: Calcola il vettore differenza da `Pu1` a `Pu2` utilizzando la funzione `Vet_Sub`. Invia immediatamente il vettore spaziale risultante alla funzione `Vet_Uni`, che provvede a estrarne il modulo e a scalarlo per restituire la sola direzione normalizzata.

Get_1Pts_Check

Firma: (Get_1Pts_Check Title lay)

Scopo: Fornire interfacce rapide (scorciatoie sintattiche) per richiamare il motore principale di acquisizione punti con quantità numeriche prefissate (da 1 a 5).

Input: Title (Stringa di prompt iniziale per istruire l'utente), lay (Stringa del layer di destinazione per i punti tracciati).

Output: Lista di 1, 2, 3, 4 o 5 punti WCS (a seconda della funzione chiamata), oppure nil se l'operazione viene annullata.

Logica interna: Funzioni wrapper pure che iniettano un intero cablato (da 1 a 5) come parametro 'numPu' nella chiamata diretta alla funzione core 'Get_N_Pts_Check', riducendo il boilerplate nel codice dei comandi principali.

Get_2Pts_Check

Firma: (Get_2Pts_Check Title lay)

Scopo: Fornire interfacce rapide (scorciatoie sintattiche) per richiamare il motore principale di acquisizione punti con quantità numeriche prefissate (da 1 a 5).

Input: Title (Stringa di prompt iniziale per istruire l'utente), lay (Stringa del layer di destinazione per i punti tracciati).

Output: Lista di 1, 2, 3, 4 o 5 punti WCS (a seconda della funzione chiamata), oppure nil se l'operazione viene annullata.

Logica interna: Funzioni wrapper pure che iniettano un intero cablato (da 1 a 5) come parametro 'numPu' nella chiamata diretta alla funzione core 'Get_N_Pts_Check', riducendo il boilerplate nel codice dei comandi principali. (defun Get_1Pts_Check (Title lay) (Get_N_Pts_Check Title 1 lay))

Get_3Pts_Check

Firma: (Get_3Pts_Check Title lay)

Scopo: Fornire interfacce rapide (scorciatoie sintattiche) per richiamare il motore principale di acquisizione punti con quantità numeriche prefissate (da 1 a 5).

Input: Title (Stringa di prompt iniziale per istruire l'utente), lay (Stringa del layer di destinazione per i punti tracciati).

Output: Lista di 1, 2, 3, 4 o 5 punti WCS (a seconda della funzione chiamata), oppure nil se l'operazione viene annullata.

Logica interna: Funzioni wrapper pure che iniettano un intero cablato (da 1 a 5) come parametro 'numPu' nella chiamata diretta alla funzione core 'Get_N_Pts_Check', riducendo il boilerplate nel codice dei comandi principali. (defun Get_1Pts_Check (Title lay) (Get_N_Pts_Check Title 1 lay)) (defun Get_2Pts_Check (Title lay) (Get_N_Pts_Check Title 2 lay))

Get_4Pts_Check

Firma: (Get_4Pts_Check Title lay)

Scopo: Fornire interfacce rapide (scorciatoie sintattiche) per richiamare il motore principale di acquisizione punti con quantità numeriche prefissate (da 1 a 5).

Input: Title (Stringa di prompt iniziale per istruire l'utente), lay (Stringa del layer di destinazione per i punti tracciati).

Output: Lista di 1, 2, 3, 4 o 5 punti WCS (a seconda della funzione chiamata), oppure nil se l'operazione viene annullata.

Logica interna: Funzioni wrapper pure che iniettano un intero cablato (da 1 a 5) come parametro 'numPu' nella chiamata diretta alla funzione core 'Get_N_Pts_Check', riducendo il boilerplate nel codice dei comandi principali. (defun Get_1Pts_Check (Title lay) (Get_N_Pts_Check Title 1 lay)) (defun Get_2Pts_Check (Title lay) (Get_N_Pts_Check Title 2 lay)) (defun Get_3Pts_Check (Title lay) (Get_N_Pts_Check Title 3 lay))

Get_5Pts_Check

Firma: (Get_5Pts_Check Title lay)

Scopo: Fornire interfacce rapide (scorciatoie sintattiche) per richiamare il motore principale di acquisizione punti con quantità numeriche prefissate (da 1 a 5).

Input: Title (Stringa di prompt iniziale per istruire l'utente), lay (Stringa del layer di destinazione per i punti tracciati).

Output: Lista di 1, 2, 3, 4 o 5 punti WCS (a seconda della funzione chiamata), oppure nil se l'operazione viene annullata.

Logica interna: Funzioni wrapper pure che iniettano un intero cablato (da 1 a 5) come parametro 'numPu' nella chiamata diretta alla funzione core 'Get_N_Pts_Check', riducendo il boilerplate nel codice dei comandi principali. (defun Get_1Pts_Check (Title lay) (Get_N_Pts_Check Title 1 lay)) (defun Get_2Pts_Check (Title lay) (Get_N_Pts_Check Title 2 lay)) (defun Get_3Pts_Check (Title lay) (Get_N_Pts_Check Title 3 lay)) (defun Get_4Pts_Check (Title lay) (Get_N_Pts_Check Title 4 lay))

Get_Dima_Data

Scopo: Restituisce l'array statico di punti 2D che definisce la curva di riferimento "Dima" (sagoma di calibrazione), usata come dato di test per i comandi di parabola a due punti fissi.

Input: Nessun parametro: la funzione restituisce always gli stessi dati hard-coded.

Output: Lista di coppie di coordinate (x y) che formano la curva di riferimento Dima.

Logica interna: Restituisce direttamente una lista quotata di coppie di coordinate, pre-calcolate e codificate nel sorgente come dato statico di test geometrico.

Get_Int_Pos

Firma: (Get_Int_Pos msg def)

Scopo: Chiede all'utente di inserire un numero intero positivo, validando che il valore sia ≥ 1 prima di restituirlo.

Input: prompt: testo del prompt di acquisizione. default_val: valore intero di default.

Output: Intero positivo inserito dall'utente (≥ 1); default_val se l'utente preme Invio.

Logica interna: Usa getint con initget per rifiutare valori ≤ 0 e accettare il default; restituisce il valore o default_val se nil.

Get_Layer_Smart

Firma: (Get_Layer_Smart Msg Dict_Key Std_Lay)

Scopo: Recupera il nome del layer corrente o di un layer predefinito del sistema Coniche, verificando che esista nel documento; se non esiste, lo crea automaticamente.

Input: layer_name: nome del layer da recuperare o creare. color_idx: indice colore AutoCAD per la creazione (se il layer non esiste).

Output: Nome del layer pronto per l'uso (stringa).

Logica interna: Usa tblsearch per verificare l'esistenza del layer; se assente chiama Lay_Make_Col per crearlo con il colore specificato; restituisce il nome del layer.

Get_lisPt_On_Axes

Firma: (Get_lisPt_On_Axes Origin V_x V_y local_coords)

Scopo: Trasformare coordinate locali (2D) in punti spaziali assoluti (3D) rispetto a un sistema di riferimento planare definito da un'origine e da due vettori direzionali (assi X e Y). Gestisce sia punti singoli che liste di coordinate.

Input: Origin (Coordinate WCS dell'origine locale), V_x e V_y (Vettori direzionali normalizzati degli assi X e Y locali). Per la funzione singola: dx, dy (scarti locali scalari). Per la funzione array: local_coords (Lista di coppie di coordinate locali).

Output: Coordinate assolute WCS del punto trasformato (per la funzione singola), oppure una lista di coordinate WCS (per la versione array).

Logica interna: La funzione base esegue una combinazione lineare vettoriale scalando i vettori asse per gli scarti locali e sommandoli all'origine tramite 'Vet_Sum' e 'Vet_Sca'. La funzione array funge da wrapper vettorializzatore: utilizza 'mapcar' e una funzione lambda per mappare l'intera lista di coordinate locali iterando la trasformazione base su ogni singolo elemento. (defun Get_Pt_On_Axes (Origin V_x V_y dx dy) (Vet_Sum (Vet_Sum Origin (Vet_Sca V_x dx)) (Vet_Sca V_y dy)))

Get_N_Pts_Check

Firma: (Get_N_Pts_Check Title numPu lay)

Scopo: Gestire l'acquisizione sequenziale e interattiva di un numero esatto (N) di punti da parte dell'utente, fornendo un feedback visivo immediato sul CAD.

Input: Title (Stringa di prompt iniziale), numPu (Numero intero di punti da acquisire), lay (Stringa indicante il layer per il feedback visivo).

Output: Lista ordinata dei punti 3D acquisiti se l'input à completo (lunghezza pari a numPu), altrimenti nil.

Logica interna: Avvia un ciclo di acquisizione getpoint appoggiandosi all'ultimo punto cliccato per generare l'effetto "rubber-band" (linea elastica). Stampa le coordinate in console e traccia un punto sul layer a ogni click. Interrompe il ciclo se l'utente annulla o raggiunge numPu. Restituisce la lista invertita per ripristinare il corretto ordine cronologico dei click.

Get_Pt_On_Axes

Firma: (Get_Pt_On_Axes Origin V_x V_y dx dy)

Scopo: Trasformare coordinate locali (2D) in punti spaziali assoluti (3D) rispetto a un sistema di riferimento planare definito da un'origine e da due vettori direzionali (assi X e Y). Gestisce sia punti singoli che liste di coordinate.

Input: Origin (Coordinate WCS dell'origine locale), V_x e V_y (Vettori direzionali normalizzati degli assi X e Y locali). Per la funzione singola: dx, dy (scarti locali scalari).

Per la funzione array: local_coords (Lista di coppie di coordinate locali).

Output: Coordinate assolute WCS del punto trasformato (per la funzione singola), oppure una lista di coordinate WCS (per la versione array).

Logica interna: La funzione base esegue una combinazione lineare vettoriale scalando i vettori asse per gli scarti locali e sommandoli all'origine tramite 'Vet_Sum' e 'Vet_Sca'. La funzione array funge da wrapper vettorializzatore: utilizza 'mapcar' e una funzione lambda per mappare l'intera lista di coordinate locali iterando la trasformazione base su ogni singolo elemento.

get_ratio

Firma: (get_ratio param)

Scopo: Trovare il parametro sulla curva dove l'ellisse passante per i 4 punti raggiunge la massima circolarità (Ellisse Pseudo-Circolare CEPC).

Input: curve_obj (Entità VLA), st_param/en_param (Limiti parametrici), pts_ord (4 vertici base), D_ref (Fattore scala), tol (Tolleranza).

Output: Valore reale (Float) che rappresenta il parametro ottimale t lungo la curva VLA.

Logica interna: Motore di ricerca iterativo a griglia con zoom progressivo. Suddivide il dominio in segmenti (CFG_DIVS), valuta il rapporto dei semiassi dell'ellisse per ogni punto e isola il valore massimo (più vicino a 1). Restringe poi il range di scansione attorno al candidato migliore, ripetendo l'operazione per un numero definito di cicli (CFG_ITERS) per convergere analiticamente verso il centro esatto. (defun CON4P_Geom_Search_Grid_Ell_Param (curve_obj st_param en_param pts_ord D_ref tol / CFG_DIVS CFG_ITERS get_ratio pt_catch a b iter step_p i curr_p curr_r max_r best_p new_a new_b) (princ "\n[DEBUG-GRID] Avvio motore a griglia per CEPC...") (setq CFG_DIVS 16.0 CFG_ITERS 4)

Get_String_Def

Firma: (Get_String_Def msg def)

Scopo: Chiede all'utente di inserire una stringa testuale con un valore predefinito: se l'utente preme Invio senza digitare nulla, restituisce il valore di default.

Input: prompt: testo del prompt. default_val: stringa da restituire se l'utente non digita nulla.

Output: Stringa inserita dall'utente, oppure default_val se ha premuto solo Invio.

Logica interna: Chiama getstring con il prompt. Se il risultato è una stringa vuota ("") restituisce default_val, altrimenti restituisce la stringa digitata.

H

Handler_Concavo_4P

Firma: (Handler_Concavo_4P pts_ord layPol layCos)

Scopo: Gestore grafico per quadrilateri CONCAVI (P4 interno al triangolo P1-P2-P3).

Input: pts_ord (Lista vertici), layPol (Layer poligoni), layCos (Layer costruzioni).

Output: Nessuno (Side-effect su DWG).

Logica interna: Decompone i segmenti, calcola le intersezioni certe dei raggi dal vertice interno ai lati opposti, e traccia perimetro e costruzioni.

Handler_Convesso_4P

Firma: (Handler_Convesso_4P pts_ord layPol layCos)

Scopo: Gestore grafico per quadrilateri Convessi Generici e Ciclici.

Input: pts_ord (Lista vertici), layPol (Layer poligoni), layCos (Layer costruzioni).

Output: Nessuno (Side-effect su DWG).

Logica interna: Calcola le due fughe esterne date dall'incrocio delle due coppie di lati opposti. Traccia il perimetro e le proiezioni esterne verso le fughe.

Handler_Degenere_4P

Firma: (Handler_Degenere_4P pts_ord tipo_f layPol layCos)

Scopo: Gestione visiva di emergenza per gli input malformati o collassati (Casi Degeneri).

Input: pts_ord (Lista vertici), tipo_f (Simbolo topologia), layPol, layCos.

Output: Nessuno (Side-effect su DWG).

Logica interna: Blocca la generazione di perimetri e diagonali per prevenire errori di lunghezza zero nel CAD. Si limita a marcare i punti battuti per fornire un feedback visivo dell'errore all'utente.

Handler_Simmetria_Centrale_4P

Firma: (Handler_Simmetria_Centrale_4P pts_ord tipo_f I layPol layCos)

Scopo: Generare l'impalcatura grafica di base esclusivamente per i quadrilateri a simmetria centrale perfetti.

Input: pts_ord (Lista vertici), tipo_f (Simbolo topologia), I (Centro/Diagonali), layPol, layCos.

Output: Nessuno (Side-effect su DWG).

Logica interna: Traccia il perimetro chiuso, l'ossatura della simmetria (diagonali) e marca il centro esatto.

Handler_Trapezio_4P

Firma: (Handler_Trapezio_4P pts_ord I layPol layCos)

Scopo: Gestore grafico per quadrilateri di tipo Trapezio (con una coppia di lati paralleli).

Input: pts_ord (Lista vertici), I (Centro/Diagonali), layPol (Layer poligoni), layCos (Layer costruzioni).

Output: Nessuno (Side-effect su DWG).

Logica interna: Identifica la fuga calcolando l'intersezione dei lati convergenti. Disegna il perimetro e proietta le estensioni dei lati verso la fuga. Evidenzia fuga e centro.

Helper_Update_Dima_Par2pv

Firma: (Helper_Update_Dima_Par2pv opt_g guide_ents P1 P2)

Scopo: Aggiorna i confini guida della Dima nel contesto del comando par2PV: rigenera o elimina i confini topologici al variare dei punti di ancoraggio.

Input: opt_g: opzione guide ("Si"/"No"). guide_ents: lista delle entità guida esistenti. P1, P2: punti di ancoraggio correnti.

Output: Lista aggiornata delle entità guida. Effetto collaterale: confini disegnati o rimossi.

Logica interna: Se opt_g = "Si" e guide_ents è nil, genera i confini chiamando `Geom_Draw_Dima_Boundaries`. Se opt_g = "Si" e guide_ents esiste, elimina le vecchie entità e rigenera. Se opt_g = "No", elimina eventuali guide esistenti.

I

Init_Conic_App

Firma: (Init_Conic_App title_str num_pts prompt_msg pdm)

Scopo: Funzione di inizializzazione comune a tutti i comandi del sistema Coniche: salva le variabili di sistema (OSMODE, PDMODE, CMDECHO), acquisisce il set iniziale di punti dall'utente e prepara l'ambiente.

Input: title: titolo del comando da stampare. num_pts: numero di punti da acquisire. prompt_msg: testo del prompt di acquisizione punti. osmode_val: valore OSMODE da impostare.

Output: Lista (pts old_os old_pd old_ce) con i punti acquisiti e i valori salvati delle variabili di sistema; pts=nil se l'utente annulla.

Logica interna: Stampa il titolo. Salva le 3 variabili di sistema. Imposta OSMODE=osmode_val. Chiama Get_NptS_Check per acquisire i punti e li restituisce insieme ai valori salvati.

inters_check_tristate

Firma: (inters_check_tristate p1 p2 p3 p4 fuzz)

Scopo: Verifica l'intersezione tra due segmenti restituendo uno stato a tre valori: intersezione nel tratto, intersezione sull'estensione, o nessuna intersezione.

Input: P1, P2, P3, P4: estremi dei due segmenti.

Output: Simbolo: 'INNER (dentro entrambi i segmenti), 'EXTENDED (sull'estensione), 'NONE (paralleli).

Logica interna: Chiama inters con e senza l'opzione di fine-segmento e confronta i risultati per determinare se il punto di intersezione è interno, esterno o inesistente.

inters_ver

Firma: (inters_ver p1 p2 p3 p4 fuzz)

Scopo: Calcola il punto di intersezione tra due segmenti (o rette) verificando anche che l'intersezione cada nei tratti finiti dei segmenti.

Input: P1, P2: estremi del primo segmento. P3, P4: estremi del secondo segmento.

Output: Coordinate del punto di intersezione se cade entro i segmenti; nil se paralleli o fuori.

Logica interna: Usa la funzione nativa inters di AutoLISP con l'opzione di verifica sulle estensioni, poi controlla che il punto cada nell'intervallo dei segmenti.

c:ipe3PRD

Scopo: Starter eseguibile root, agganciato al timer di perfomance e all'Oracolo Geometrico che previene gli stalli matematici nativi.

Input: Nessuno.

Output: Nessuno.

Logica interna: Innesca il clock. Processa la famiglia Master Base tramite Master_Engine_Pure infondendole le Matrioske dell'Oracolo Geometrico. Ricicla per Inversa. Genera Mappe Nodi. Emette Delta Tempo. Richiama LUSNAP.

IPE3P_Calc_Exact_Singularity_Params

Firma: (IPE3P_Calc_Exact_Singularity_Params lisPu ratio_val)

Scopo: Ricavare analiticamente i parametri angolari esatti in cui l'iperbole degenera in una coppia di rette (schianto sui lati).

Input: lisPu (Terna dei vertici base), ratio_val (Vincolo di rapporto b/a).

Output: Lista di parametri angolari in radianti, rappresentanti gli inneschi esatti dei collassi geometrici.

Logica interna: Scompone la lista iniziale lisPu nei singoli punti Pu1, Pu2 e Pu3 tramite mapcar. Invoca quindi la funzione esterna IPE3P_Helper_Calc_Angles_Singularity per ciascuno dei tre lati del triangolo base, aggregando i risultati in un'unica lista piatta tramite append.

IPE3P_Draw_Engine

Firma: (IPE3P_Draw_Engine pts_tri geo_packet opt_m opt_e last_ents)

Scopo: Driver CAD per la vettorializzazione dell'iperbole a estensione dinamica asintotica.

Input: pts_tri, geo_packet, opt_m, opt_e, last_ents.

Output: Ename handle su AutoCAD.

Logica interna: Calcola la propulsione d'estensione della conica tramite proiezioni limitanti, evitando il sovraccarico di buffer sui tracciati illimitati classici.

IPE3P_Eval_Cands

Firma: (IPE3P_Eval_Cands pts_list cur_ratio ang_rad target_type)

Scopo: Elaboratore algebrico real-time a supporto del raffinamento continuo interattivo LUSNAP.

Input: pts_list (Vertici), cur_ratio (Rapporto), ang_rad (Angolo asse), target_type (Tipologia di Output).

Output: Generazione di un pacchetto di candidati formativi (Coordinate e Metadati) specifici per lo sweep richiesto.

Logica interna: Trasla e ruota le coordinate base per il micro-schermo risolvendo l'iperbole istantanea. Filtra con listp crash scalari.

IPE3P_Find_Absolute_Nearest_Data

Firma: (IPE3P_Find_Absolute_Nearest_Data pt_mouse map_data)

Scopo: Cercare il punto discretizzato spazialmente più vicino al cursore del mouse.

Input: pt_mouse (Coordinate spaziali utente), map_data (Mappa Pura dei punti elaborati).

Output: Lista strutturata (DatiElemento Indice Lista DistanzaMinima).

Logica interna: Scansiona la mappa associativa integralmente, estraendo tramite formula euclidea il record con lo scostamento planimetrico inferiore.

IPE3P_Fractal_Around_Param_Safe

Firma: (IPE3P_Fractal_Around_Param_Safe param_center step_base)

Scopo: Iniettare una cascata di micro-parametri di infittimento spaziale addossati a una singolarità ("Matrioska"), omettendola rigorosamente.

Input: param_center (Parametro di schianto esatto), step_base (Passo di crociera master).

Output: Lista di micro-parametri addensati in modo logaritmico.

Logica interna: Delega la generazione delle singole rampe simmetriche (1/16, 1/8, 1/4, 1/2) all'helper IPE3P_Helper_Fractal_Band, unificando i risultati in un'unica lista piatta tramite append. Omette deliberatamente il valore param_center per evitare il collasso del risolutore matematico sulla conica degenere.

IPE3P_Get_Best_Refined_Among_Lists

Firma: (IPE3P_Get_Best_Refined_Among_Lists pt_mouse pts_list lists_of_maps mode target_type)

Scopo: Eleggere il pacchetto visivo migliore in gara multicore interrogando simultaneamente la Mappa Base e la Mappa Inversa.

Input: pt_mouse, pts_list, lists_of_maps (le mappe parallele), mode, target_type.

Output: Il candidato vincente formattato o nil.

Logica interna: Clicla il refined locale su tutte le liste accese ed esporta univocamente il record euclideo minore.

IPE3P_Get_Constraints

Scopo: Selezione vincolo primario d'indagine.

Input: Nessuno.

Output: Modificatore d'indice e scala d'ancoraggio.

Logica interna: Guida initget sul branching Rapporto o Direzione.

IPE3P_Get_Data

Firma: (IPE3P_Get_Data app_title)

Scopo: Richiesta architettura spaziale in ingresso.

Input: app_title (Stringa Master).

Output: Punti d'appoggio primari e backup sistema ambientale.

Logica interna: Avvia la logica di base per pre-caricare l'ortico d'orientamento.

IPE3P_Helper_Calc_Angles_Singularity

Firma: (IPE3P_Helper_Calc_Angles_Singularity pt_A pt_B pt_opp ratio_val)

Scopo: Calcolare le bisettrici ortogonali (assi focali) per la conica degenere su un singolo lato del triangolo base.

Input: pt_A, pt_B (Estremi del lato), pt_opp (Vertice opposto), ratio_val (Vincolo di rapporto b/a).

Output: Lista di parametri angolari in radianti.

Logica interna: Calcola lo schianto sul lato. Sfrutta l'helper IPE3P_Helper_Orthogonal_Bisectors valutando in linea gli angoli necessari, restituendo la singola coppia o la fusione (append) delle due coppie se lo schianto à doppio.

IPE3P_Helper_Draw_And_Group_Locus

Firma: (IPE3P_Helper_Draw_And_Group_Locus pt_list col_val layer_name group_name)

Scopo: Disegnare una nuvola di punti con un colore specifico e raggrupparla automaticamente.

Input: pt_list (Lista di coordinate 3D), col_val (Variabile colore), layer_name (Nome del layer), group_name (Stringa nome gruppo).

Output: Lista delle entità (ename) appena generate.

Logica interna: Registra l'ultimo ename, forza il colore, itera la lista per tracciare i punti, raccoglie le nuove entità e le compatta in un gruppo anonimo.

IPE3P_Helper_Fractal_Band

Firma: (IPE3P_Helper_Fractal_Band param_center step_base max_steps base_mult)

Scopo: Generare una singola "banda" di micro-parametri simmetrici attorno a una singolarità.

Input: param_center (Centro dello schianto), step_base (Passo master), max_steps (Numero di partizioni), base_mult (Moltiplicatore offset base).

Output: Lista di parametri angolari addensati per la specifica banda.

Logica interna: Calcola l'offset incrementale e genera simmetricamente i parametri a destra e a sinistra del centro, restituendoli in una lista piatta.

IPE3P_Helper_Orthogonal_Bisectors

Firma: (IPE3P_Helper_Orthogonal_Bisectors ang_1 ang_2)

Scopo: Calcolare la coppia di bisettrici ortogonali date due direzioni angolari.

Input: ang_1, ang_2 (Valori angolari in radianti).

Output: Lista di due parametri angolari normalizzati [0, PI).

Logica interna: Calcola la bisettrice media e la sua ortogonale (+ PI/2), delegando la normalizzazione a IPE3P_Modulo_Pi. Restituisce direttamente la coppia in una lista.

IPE3P_Helper_UI_Prompt

Firma: (IPE3P_Helper_UI_Prompt opt_gc opt_gf opt_gv opt_t opt_m opt_e)

Scopo: Generare in modo dinamico la stringa di prompt interattivo per il ciclo continuo LUSNAP.

Input: opt_gc (Visibilità Centri), opt_gf (Visibilità Fuochi), opt_gv (Visibilità Vertici), opt_t (Target attivo), opt_m (Modo operativo), opt_e (Moltiplicatore estensione).

Output: Stringa testuale formattata per AutoCAD.

Logica interna: Formatica le opzioni attive correnti in base ai toggle selezionati. Disabilita la voce di menu "Target" se la visualizzazione dei Vertici à inibita.

IPE3P_Master_Engine_Pure

Firma: (IPE3P_Master_Engine_Pure pts_list mode forced_val)

Scopo: Generare le mappe discrete ad alta densità in modalità Single-Pass, pre-inglobando l'iniezione geometrica del Microscan.

Input: pts_list (Vertici), mode (Modo Operativo), forced_val (Vincolo di Rapporto o Direzione).

Output: Lista annidata con le mappe coordinate e i puntatori dei gruppi di Locus.

Logica interna: Dispone lo spettro continuo in array lineare. Qualora il modo operativo sia 2 (Rapporto), interroga l'Oracolo Geometrico per le singolarità, addensa le rampe frattali sui bersagli e ordina la lista

ripulendo sovrapposizioni. Passa al risolutore filtrando residui spuri listp.

IPE3P_Math_Error_Func

Firma: (IPE3P_Math_Error_Func pts_list angle_val current_ratio target_pt target_type)

Scopo: Sensore di prossimità euristico finalizzato all'aggancio dei Nodi Esatti.

Input: pts_list, angle_val, current_ratio, target_pt, target_type.

Output: Margine numerico reale rappresentante l'errore d'allineamento.

Logica interna: Valuta geometricamente il distacco della conica virtuale dal target_pt esatto in fase di blind-shooting per minimizzare lo sweep dei nodi.

IPE3P_Modulo_Pi

Firma: (IPE3P_Modulo_Pi ang_val)

Scopo: Normalizzare un angolo nel semipiano positivo standard [0, PI).

Input: ang_val (Valore angolare in radianti).

Output: Valore angolare normalizzato (Real).

Logica interna: Applica l'operatore resto (rem) con pi greco. Se il risultato cade in campo negativo, lo trasla nel semipiano positivo sommandogli pi greco.

IPE3P_Precalc_Exact_Nodes

Firma: (IPE3P_Precalc_Exact_Nodes pts_list mode fixed_val node_type)

Scopo: Strutturare la Mappa ad accesso istantaneo del Ciclattore LUSNAP Nodi.

Input: pts_list, mode, fixed_val, node_type.

Output: Mappa associativa per ogni vertice.

Logica interna: Chiama la routine IPE3P_Solve_Exact_Node_Algebraic distribuendola su tutta l'impalcatura di base.

IPE3P_Refine_Locus_Pure

Firma: (IPE3P_Refine_Locus_Pure pt_mouse pts_list pure_map mode target_type)

Scopo: Adattare in tempo reale e interpolare la griglia puramente algebrica per catturare il cursore utente (regola del braccio elastico).

Input: pt_mouse (Mouse), pts_list (Vertici), pure_map (Mappa guida), mode (Vincolo), target_type.

Output: Lista di esito per il ranking LUSNAP (Distanza Minima, Miglior Candidato).

Logica interna: Cerca il pivot discreto più vicino al cursore. Estrapola lo step di convergenza verso i pivot contigui, scindendolo internamente in 25 passate locali per spremere il target senza sovraccaricare il buffer.

IPE3P_Solve_Exact_Node_Algebraic

Firma: (IPE3P_Solve_Exact_Node_Algebraic p_target pts_list mode fixed_val active_node_type)

Scopo: Popolare l'archivio RAM di soluzioni chiuse per le stazioni d'arresto ciclabili dei Nodi di base (Vertici/Fuochi).

Input: p_target (Nodo base), pts_list, mode, fixed_val, active_node_type.

Output: Lista di soluzioni ottimali.

Logica interna: Indaga tutto il dominio riducendo il campo per ricerca dei minimi in dissezione fine, salvaguardando il vincolo 1e-4.

IPE3P_Universal_Loop

Firma: (IPE3P_Universal_Loop pts_list mode_idx f_val m_C_base e_C_base m_C_inv e_C_inv m_F_base e_F_base m_F_inv e_F_inv m_V_base e_V_base m_V_inv e_V_inv exact_V_map exact_F_map)

Scopo: Processore Master degli input utente asincroni durante la manipolazione dello spettro continuo.

Input: Metadati globali delle pre-spazzate Mappe e Vincoli.

Output: Gestione interattiva.

Logica interna: Gestisce lo switch su Luogo o Nodo, interrogando il Multicore su Luogo, e la Memoria O(1) sul Target discreto.

L

Lay_Make_Col

Firma: (Lay_Make_Col lay col)

Scopo: Creare silenziosamente un layer o forzarne l'aggiornamento cromatico, operando nel database senza alterare il layer corrente.

Input: lay (Stringa col nome del layer), col (Intero 1-255 per il colore AutoCAD).

Output: Restituisce il nome del layer processato (Stringa).

Logica interna: Estrae il safe_col. Cerca il layer via tblobjname. Se esiste, ne legge il codice 62. Esegue entmod SOLO se il colore attuale differisce da quello richiesto, risparmiando scritture inutili. Se non esiste, lo crea da zero tramite entmake.

Lis_2nth

Firma: (Lis_2nth id1 id2 lisPu)

Scopo: Estrae due elementi da una lista usando due indici numerici.

Input: id1, id2: indici interi (0-based). lisPu: lista di elementi.

Output: Lista di due elementi: (elemento[id1], elemento[id2]).

Logica interna: Chiama `nth` due volte con gli indici forniti e raccoglie i risultati in una lista di due elementi.

Lis_Coppie_nth

Firma: (Lis_Coppie_nth lisco lisPu)

Scopo: Costruisce una lista di coppie di elementi estratti da una lista, usando un array di coppie di indici come selettore.

Input: lisco: lista di coppie di indici ((i1 j1) (i2 j2) ...). lisPu: lista sorgente di elementi.

Output: Lista di coppie di elementi selezionati da lisPu secondo gli indici in lisco.

Logica interna: Mappa `Lis_2nth` su ogni coppia di indici in lisco, passando lisPu come lista sorgente.

Load_Conic_Layers_Fast

Scopo: Verificare la presenza nel disegno dei layer standard preconfigurati per la rappresentazione delle coniche e, se già esistenti, proporne all'utente il riutilizzo rapido per velocizzare il setup.

Input: Nessuno (la funzione non richiede argomenti formali, ma attinge i dati dal dizionario LDATA del DWG e dall'input utente).

Output: Restituisce il valore booleano T se tutti i layer esistono e l'utente accetta di usarli, altrimenti nil.

Logica interna: Costruisce tre liste parallele per mappare le variabili globali, i nomi di default e le chiavi del dizionario LDATA. Scorre le chiavi estraendo i nomi personalizzati (o applicando i default) e usa 'tblsearch' per confermarne l'esistenza. Se il set à completo, lancia un prompt [Si/No]. Se confermato, associa massivamente i nomi estratti ai simboli globali (*LaCo_*) tramite 'mapcar set', salva lo stato nel dizionario e restituisce T, bypassando così il motore di creazione layer lento.

Log_Pascal_Iteration

Firma: (Log_Pascal_Iteration counter res_temp)

Scopo: Registra nel log i risultati di una singola iterazione dell'algoritmo di Pascal: il tipo di tentativo, la permutazione usata, il risultato e il tipo di conica trovata.

Input: record: lista (tipo_tentativo punti_perm punto_trovato tipo_conica).

Output: Effetto collaterale: aggiunge il record alla lista di log globale. Restituisce il record.

Logica interna: Usa cons o append per aggiungere il record alla variabile globale di log, mantenendo la cronologia completa di tutti i tentativi pascaliani.

M

Math_Acos

Firma: (Math_Acos val)

Scopo: Fornire la funzione matematica arcocoseno (acos), nativamente assente in AutoLISP, ricavandone il valore in radianti a partire dalla funzione arcotangente.

Input: val (Numero reale, rigorosamente compreso nel dominio tra -1.0 e 1.0, rappresentante il valore del coseno di un angolo).

Output: Valore numerico reale corrispondente all'angolo espresso in radianti (compreso tra 0 e pi greco).

Logica interna: Valuta preventivamente tramite `cond` se il valore in ingresso raggiunge o eccede i limiti del dominio [-1.0, 1.0], restituendo in modo forzato 0.0 o `pi` per scongiurare errori di dominio. Per i valori interni validi, applica la formula trigonometrica equivalente basata sull'istruzione nativa `atan` unita alla radice di (1 - val^2).

Math_Angolo_Minore_CCW

Firma: (Math_Angolo_Minore_CCW Cen P1 P2)

Scopo: Calcola l'angolo minore compreso tra due angoli orientati in senso antiorario (CCW), restituendo il valore compreso tra 0 e π.

Input: a1, a2: angoli in radianti.

Output: Valore numerico reale tra 0 e π rappresentante l'angolo minore tra i due orientamenti.

Logica interna: Calcola la differenza assoluta tra i due angoli, poi applica `min` con il suo supplemento (2π − diff) per garantire che il risultato sia sempre l'angolo geometrico minore.

Math_Area_Triangolo

Firma: (Math_Area_Triangolo Pu1 Pu2 Pu3)

Scopo: Calcola l'area del triangolo definito dai tre punti P1, P2, P3 usando la formula del determinante.

Input: P1, P2, P3: liste di coordinate 2D o 3D dei tre vertici del triangolo.

Output: Valore numerico reale positivo dell'area del triangolo.

Logica interna: Sfrutta `Math_Cyclic_Det` per il calcolo del doppio dell'area orientata (formula del "shoelace"), poi ne prende il valore assoluto e lo divide per 2.

Math_Aree_Pu_vs_Tri

Firma: (Math_Aree_Pu_vs_Tri Pu1 Pu2 Pu3 Pu4)

Scopo: Calcola e confronta le aree dei sotto-triangoli formati da un punto P con i tre vertici del triangolo P1-P2-P3, per verificare la posizione relativa di P rispetto al triangolo.

Input: P, P1, P2, P3: liste di coordinate 2D o 3D. P è il punto da testare, P1-P2-P3 i vertici del triangolo.

Output: Lista delle tre aree dei sotto-triangoli (P-P1-P2, P-P2-P3, P-P3-P1) o risultato del confronto.

Logica interna: Chiama `Math_Area_Triangolo` per ciascuno dei tre sotto-triangoli generati associando P con ciascun lato del triangolo, permettendo al chiamante di verificare se la somma uguaglia l'area totale (punto interno) o la supera (punto esterno).

Math_Calc_Par_Error_4th_Pt

Firma: (Math_Calc_Par_Error_4th_Pt Ang pts)

Scopo: Valuta di quanto la parabola con asse inclinato di Ang, passante per i primi 3 punti, si discosta dal 4° punto: funzione di errore usata nella scansione parametrica.

Input: Ang: angolo dell'asse. pts: lista di 4 punti (i primi 3 definiscono la parabola, il 4° è il punto di verifica).

Output: Valore numerico reale dello scarto residuo nel 4° punto (y_calc − y4); 0.0 = parabola esatta sul 4° punto. 1e9 se degenere.

Logica interna: Ruota tutti e 4 i punti con Util_Rot_List_2D_CS, risolve il sistema 3×3 per i coefficienti a,b,c tramite determinante ciclico, poi valuta y_calc = a·x4²+b·x4+c e restituisce la differenza con y4_reale.

Math_Calc_Par_Global_Error

Firma: (Math_Calc_Par_Global_Error pts_local A B C)

Scopo: Calcola l'errore quadratico complessivo della parabola con asse inclinato di Ang rispetto a tutti i punti del set, sommando i quadrati degli scarti residui.

Input: Ang: angolo dell'asse in radianti. pts: lista di punti da fittare.

Output: Valore numerico reale non negativo: somma dei quadrati degli scarti. Più basso = migliore fit. 0.0 = interpolazione esatta.

Logica interna: Chiama Math_Solve_Par_3P per ottenere i coefficienti della parabola, poi valuta il residuo su ogni punto aggiuntivo (y_calc − y_reale)² e ne somma i quadrati.

Math_Check_Conciclicita_4Pu

Firma: (Math_Check_Conciclicita_4Pu Pu1 Pu2 Pu3 Pu4)

Scopo: Verifica se quattro punti giacciono su una stessa circonferenza (sono concicli), calcolando la circonferenza passante per tre di essi e verificando se il quarto la soddisfa.

Input: P1, P2, P3, P4: liste di coordinate 2D o 3D dei quattro punti da testare.

Output: T se i quattro punti sono concicli entro la tolleranza numerica; nil altrimenti.

Logica interna: Calcola la circonferenza passante per P1, P2, P3 tramite `Math_Circonferenza_3Pu`, poi misura la distanza di P4 dal centro e la confronta con il raggio; se la differenza è inferiore a una soglia di tolleranza i quattro punti sono considerati concicli.

Math_Circonferenza_3Pu

Firma: (Math_Circonferenza_3Pu P1 P2 P3)

Scopo: Calcolare analiticamente il centro e il raggio della circonferenza passante per tre punti dati (circocentro).

Input: P1, P2, P3 (Liste di coordinate 3D dei tre punti di passaggio).

Output: Lista a due elementi contenente le coordinate del Centro e il valore del Raggio, formattata come: ((Cx Cy 0.0) Raggio). Restituisce nil se i punti sono allineati o la configurazione à degenere.

Logica interna: Verifica preliminarmente l'allineamento tramite la funzione `P_Allin_3Pu`. Decompone i punti nelle componenti X e Y calcolandone la somma dei quadrati. Sfrutta la funzione `Math_Cyclic_Det`per determinare il doppio dell'area orientata (D). Se D supera la soglia di tolleranza numerica (1e-12), risolve il sistema algebrico per ricavare il centro planare (cx, cy) e calcola la distanza euclidea da P1 per ottenere il raggio.

Math_Circonferenza_Eulero_4Pu

Firma: (Math_Circonferenza_Eulero_4Pu Pu1 Pu2 Pu3 Pu4)

Scopo: Calcolare la Circonferenza di Eulero (dei 9 punti) di un sistema ortocentrico, mantenendo la ridondanza analitica di calcolo e media.

Input: Pu1, Pu2, Pu3, Pu4 (Quaterna di punti formante un sistema ortocentrico).

Output: Lista contenente (CentroMedio RaggioMedio) oppure nil in caso di sistema non valido o degenere.

Logica interna: Estrae l'ortocentro e il triangolo base. Calcola le tre circonferenze (cir1, cir2, cir3). Se valide, ne spacchetta i centri (Ce1, Ce2, Ce3) e i raggi (ra1, ra2, ra3). Sfrutta gli helper Pu_Baricentro2D_LisPu e Util_Media_lisNum per delegare totalmente i calcoli medi.

Math_Classify_Forma_4Pu

Firma: (Math_Classify_Forma_4Pu Pu1 Pu2 Pu3 Pu4)

Scopo: Classificare la topologia geometrica formata da 4 punti spaziali, categorizzandola in base alle sue proprietà (degenerazioni, concavità, convessità, simmetrie) e riordinando i vertici di conseguenza.

Input: Pu1, Pu2, Pu3, Pu4 (Coordinate dei quattro punti base).

Output: Lista strutturata a due elementi: (Simbolo_Tipologia Lista_Punti_Ordinati). Esempio: ('QUADRANGOLO_GENERICO (p1 p2 p3 p4)).

Logica interna: Sfrutta un costrutto 'cond' a cascata che valuta una serie gerarchica di predicati booleani, procedendo dal caso più degenere e specifico (Livello 0:

punti coincidenti o allineati) fino ai casi più generali (Livello 2: quadrangoli convessi). Trovata la corrispondenza topologica, affida la quaterna al motore di ordinamento specifico (es. CCW per i convessi, orientamento interno per i concavi) per garantire la coerenza vettoriale nei calcoli a valle.

Math_Classify_Forma_Lis4Pu

Firma: (Math_Classify_Forma_Lis4Pu pts)

Scopo: Funzione adattatore (Wrapper) per interfacciare liste singole con il classificatore matematico a parametri separati.

Input: pts (Lista contenente 4 coordinate 3D grezze).

Output: Lista strutturata di classificazione (TipoTopologico ListaPuntiOrdinati).

Logica interna: Inietta l'array in ingresso direttamente negli argomenti del motore base tramite l'operatore funzionale 'apply'.

Math_Combine_Conics

Firma: (Math_Combine_Conics K1 K2 lam)

Scopo: Combina linearmente due coniche K1 e K2 nella forma del fascio K1+λ·K2, restituendo i coefficienti della conica combinata.

Input: K1, K2: liste di 6 coefficienti [A B C D E F] delle due coniche. lam: parametro λ del fascio.

Output: Lista di 6 coefficienti della conica K1+λ·K2.

Logica interna: Moltiplica componente per componente K2 per λ, poi somma elemento per elemento con K1 tramite mapcar.

Math_ConicaCentri_9Pu

Firma: (Math_ConicaCentri_9Pu Pu1 Pu2 Pu3 Pu4)

Scopo: Generare la nuvola dei 9 punti notevoli (6 medi + 3 diagonali) per cui passa l'ellisse/iperbole dei centri del quadrangolo.

Input: Pu1, Pu2, Pu3, Pu4 (Coordinate 3D dei vertici del quadrangolo).

Output: Lista ordinata di 9 coordinate 3D.

Logica interna: Massimizza l'efficienza unendo i risultati di due sottomotori specializzati. Sfrutta il risolutore diretto 'Math_LisPuntiMedi_Quadrangolo_4Pu' per estrarre a bassissimo costo computazionale (bare-metal) i 6 punti medi, e lo concatena (append) all'output della funzione 'Math_TriangoloDiagonale_4Pu' per i 3 centri d'intersezione, evitando duplicazioni di codice.

Math_ConIpeEqu_Coeffs

Firma: (Math_ConIpeEqu_Coeffs cen pts)

Scopo: Calcolare i coefficienti dell'equazione generale (A, B, C, D, E, F) di un'iperbole equilatera passante per 3 punti.

Input: cen (Coordinate del centro dell'iperbole), pts (Lista di 3 punti di passaggio WCS).

Output: Lista di 6 coefficienti (A B C D E F), oppure nil se la conica degenera (determinante prossimo a zero).

Logica interna: Decomprime la lista punti ed esegue la traslazione scalare rispetto al centro per massima velocità. Srotola il calcolo combinatorio delle 3 coppie per pre-calcolare algebricamente i termini u, v una sola volta. Identifica il determinante massimo in valore assoluto per garantire stabilità numerica e ricava i parametri best_A e best_B.

Math_Conta_Coinc_4Pu

Firma: (Math_Conta_Coinc_4Pu Pu1 Pu2 Pu3 Pu4)

Scopo: Conta quante coppie di punti coincidono nel set di 4 punti dato (0, 1, 2, 3 o 6 coppie).

Input: P1, P2, P3, P4: coordinate dei quattro punti da analizzare.

Output: Intero tra 0 e 6: numero di coppie di punti coincidenti.

Logica interna: Confronta tutte le 6 combinazioni possibili di coppie (C(4,2)) usando `P_Coinc_2Pu` o la distanza < tolleranza, sommando i risultati booleani.

Math_Conta_Ortogo_4Pu

Firma: (Math_Conta_Ortogo_4Pu Pu1 Pu2 Pu3 Pu4)

Scopo: Conta quante coppie di lati adiacenti sono ortogonali nel quadrangolo definito da 4 punti (0, 1, 2, 3 o 4 angoli retti).

Input: P1, P2, P3, P4: coordinate dei quattro vertici del quadrangolo.

Output: Intero tra 0 e 4: numero di angoli interni del quadrangolo che sono retti (90°).

Logica interna: Valuta l'ortogonalità di ciascuna coppia di lati adiacenti (P1-P2:P2-P3, P2-P3:P3-P4, ecc.) tramite prodotto scalare o `P_Perpe_2Re` e somma i risultati.

Math_Conta_Paral_4Pu

Firma: (Math_Conta_Paral_4Pu Pu1 Pu2 Pu3 Pu4)

Scopo: Conta quante coppie di lati opposti sono parallele nel quadrangolo definito da 4 punti (0, 1 o 2 coppie).

Input: P1, P2, P3, P4: coordinate dei quattro vertici del quadrangolo nell'ordine dato.

Output: Intero 0, 1 o 2: numero di coppie di lati opposti paralleli.

Logica interna: Verifica il parallelismo tra i lati P1-P2 e P3-P4, poi tra P2-P3 e P4-P1 usando il predicato `P_Paral_2Re` o la verifica dell'angolo tra direzioni. Somma i risultati booleani convertiti a intero.

Math_Cyclic_Det

Firma: (Math_Cyclic_Det k1 k2 k3 v1 v2 v3)

Scopo: Calcola il determinante ciclico di sei valori, usato per il doppio dell'area orientata di un triangolo: x1(y2−y3) + x2(y3−y1) + x3(y1−y2).

Input: x1, x2, x3, y1, y2, y3: coordinate X e Y dei tre vertici del triangolo.

Output: Valore numerico reale (doppio dell'area orientata; positivo in senso antiorario).

Logica interna: Applica direttamente la formula del determinante ciclico senza ricorrere a strutture dati ausiliarie.

Math_Delta_List

Firma: (Math_Delta_List lst)

Scopo: Calcolare l'escursione totale assoluta (il Delta) tra il limite superiore e il limite inferiore presenti all'interno di un set di dati numerici.

Input: lst (Lista monodimensionale di valori numerici).

Output: Valore numerico (reale o intero) che rappresenta l'ampiezza dell'intervallo (Max - Min).

Logica interna: Delega l'estrazione degli estremi matematici all'helper `Math_MinMax_List`, iniettando il flag 2 per farsi restituire la coppia ordinata di valori. Successivamente, tramite una semplice differenza algebrica (`cadr` meno `car`), ricava l'escursione isolata restituendone il modulo scalare.

Math_Det3x3

Firma: (Math_Det3x3 a1 a2 a3 b1 b2 b3 c1 c2 c3)

Scopo: Calcolare il determinante di una matrice 3x3 ricevendo i nove coefficienti come argomenti scalari individuali, ottimizzando le prestazioni per chiamate dirette senza overhead di liste.

Input: a1, a2, a3 (Coefficienti 1à riga), b1, b2, b3 (Coefficienti 2à riga), c1, c2, c3 (Coefficienti 3à riga).

Output: Valore reale (float) corrispondente al determinante della matrice.

Logica interna: Implementa lo sviluppo di Laplace lungo la prima riga. Calcola i tre minori complementari di ordine 2 sfruttando la funzione 'Math_Diff_Prod' per determinare i prodotti incrociati e restituisce la somma algebrica pesata dei risultati. Rispetto alla versione matriciale, evita lo spacchettamento delle liste migliorando i tempi di esecuzione in cicli intensivi.

Math_Diff_Prod

Firma: (Math_Diff_Prod a b c d)

Scopo: Calcola la differenza di due prodotti: (a × b) − (c × d).

Input: a, b, c, d: valori numerici reali.

Output: Valore reale risultante dalla differenza dei due prodotti.

Logica interna: Singola espressione (- (* a b) (* c d)); usata nel determinante 2×2 e nelle rotazioni.

Math_Diff_Squares

Firma: (Math_Diff_Squares a b)

Scopo: Calcola la differenza dei quadrati di due valori numerici ($a^2 - b^2$).

Input: a, b: valori numerici reali.

Output: Valore numerico reale corrispondente ad $a^2 - b^2$.

Logica interna: Moltiplica ogni termine per se stesso e ne calcola la differenza diretta.

Math_Dist_2Pu

Firma: (Math_Dist_2Pu Pu1 Pu2)

Scopo: Calcolare in sicurezza la distanza reale tra due punti, prevenendo crash di esecuzione causati da argomenti nulli o malformati.

Input: Pu1, Pu2 (Coordinate spaziali dei due punti da misurare, attese rigorosamente in formato lista).

Output: Valore numerico reale rappresentante la distanza euclidea. Restituisce nil se uno o entrambi gli argomenti risultano mancanti o non validi.

Logica interna: Sfrutta gli operatori logici `and` e `listp` per eseguire un type-checking preventivo sugli argomenti in ingresso. Solo se entrambi i parametri superano la validazione, delega il calcolo alla funzione nativa `distance` di AutoLISP.

Math_Dist_Point_Line

Firma: (Math_Dist_Point_Line Pu1 Pu2 Pu3)

Scopo: Calcola la distanza euclidea di un punto Pu dalla retta definita dai coefficienti (a,b,c) dell'equazione generale ax+by+c=0.

Input: Pu: coordinate 2D del punto. L: lista (a b c) dei coefficienti della retta.

Output: Valore numerico reale della distanza punto-retta; 0.0 se il punto giace sulla retta.

Logica interna: Applica la formula analitica: d = $|a \cdot x + b \cdot y + c| \,/\, \mathrm{sqrt}(a^2+b^2)$, usando Math_Sqr e Pitag_Ip per il denominatore.

Math_Extract_Coord_Pow

Firma: (Math_Extract_Coord_Pow pts idx pow)

Scopo: Estrae da una lista di punti le coordinate di un certo asse (X o Y) elevate a una potenza specificata, restituendole come lista di scalari.

Input: pts: lista di punti 2D/3D. axis: indice dell'asse (0=X, 1=Y). pow: esponente (1 o 2).

Output: Lista di valori numerici, ciascuno pari a coord_i^pow del punto i-esimo.

Logica interna: Usa mapcar con una lambda che applica nth per estrarre la componente e Math_Sqr (se pow=2) o l'identità (se pow=1) per elevarla a potenza.

Math_Extract_Equilateral_Data

Firma: (Math_Extract_Equilateral_Data Coeffs)

Scopo: Estrae i dati geometrici (centro, direzione dell'asse, semi-asse) da una conica equilatera (iperbole rettangola, A+C=0) definita dai suoi 6 coefficienti.

Input: Coeffs: lista [A B C D E F] dei coefficienti di una conica equilatera.

Output: Lista (Centro Versore_Asse Semi_Asse) con il centro come lista 3D, il versore dell'asse principale e il semi-asse in unità reali; nil se degenere.

Logica interna: Calcola il centro dalla formula diretta usando il determinante $4AC-B^2$. Calcola il valore proprio $\lambda 1$ dalla traccia della matrice della forma quadratica, il semi-asse da $F'/\lambda 1$, e l'orientamento da atan(B, A−C)/2.

Math_Extract_Geom_Data_From_Coeffs

Firma: (Math_Extract_Geom_Data_From_Coeffs K PuC m_dist)

Scopo: Estrae i dati geometrici di una conica (tipo, centro, semiassi, fuochi, eccentricità, orientamento) a partire dai suoi 6 coefficienti algebrici.

Input: K: lista [A B C D E F] dei coefficienti. PuC: centro di normalizzazione (punto origine). m_dist: fattore di scala per la denormalizzazione.

Output: Lista strutturata contenente: tipo di conica ('ellisse/'iperbole), centro in WCS, semiassi, angolo di orientamento dell'asse maggiore, fuochi, eccentricità.

Logica interna: Calcola il determinante $4AC-B^2$ per classificare la conica. Completa il quadrato per ricavare il centro (h,k). Diagonalizza la forma quadratica tramite rotazione dell'angolo $\theta = 0.5 \cdot \text{atan2}(B, A-C)$. Calcola i semiassi dalla forma canonica e ricava fuochi ed eccentricità dalle relazioni geometriche standard.

Math_Get_Coeffs_4P_AxisDir

Firma: (Math_Get_Coeffs_4P_AxisDir Pu1 Pu2 Pu3 Pu4 PuD1 PuD2)

Scopo: Trova i coefficienti della conica passante per 4 punti con l'asse principale orientato nella direzione specificata da PuD1-PuD2, usando il fascio di coniche.

Input: Pu1,Pu2,Pu3,Pu4: quattro punti di vincolo. PuD1,PuD2: due punti che definiscono la direzione dell'asse della conica cercata.

Output: Lista (K_finale PuC max_dist tipo_conica) con i coefficienti della conica, il centro di normalizzazione, il fattore di scala e il simbolo 'ELLISSE/'IPERBOLE/'PARABOLA; nil se il sistema è degenere.

Logica interna: Normalizza i 4 punti, costruisce il fascio $K=K1+\lambda \cdot K2$ (prodotto delle diagonali), imposta la condizione sull'angolo dell'asse ($B \cdot \cos 2\alpha = (A-C) \cdot \sin 2\alpha$) per ricavare λ, poi classifica la conica tramite il discriminante B^2-4AC.

Math_Inters_lisQuaterne

Firma: (Math_Inters_lisQuaterne lst_quaterne tipo)

Scopo: Eseguire il calcolo massivo delle intersezioni su un array di quaterne di punti, applicando a tutte la medesima regola topologica (es. tutti i lati opposti, oppure tutte le diagonali).

Input: lst_quaterne (Lista di sottoliste, dove ogni elemento à una quaterna di coordinate 3D), tipo (Intero 1, 2 o 3 che definisce la regola di intersezione da applicare globalmente).

Output: Lista ordinata dei punti di intersezione spaziale (coordinate 3D, oppure nil in caso di rette parallele) calcolati per ciascuna quaterna.

Logica interna: Utilizza la funzione `mapcar` associata a una lambda function per iterare sull'intero array in ingresso. Ad ogni ciclo, delega la singola quaterna corrente e il parametro `tipo` invariato alla funzione di calcolo base `Math_Inters_Quaterna`.

Math_Inters_Quaterna

Firma: (Math_Inters_Quaterna quaterna tipo)

Scopo: Calcolare il punto di intersezione spaziale (fuga o incrocio diagonale) tra specifiche coppie di segmenti definiti dai quattro vertici di un quadrangolo.

Input: quaterna (Lista ordinata di 4 coordinate 3D dei vertici), tipo (Numero intero 1, 2 o 3 che fa da selettore per decidere quali coppie di punti collegare e intersecare).

Output: Lista di coordinate 3D del punto di intersezione individuato. Restituisce nil se le rette risultano parallele.

Logica interna: Decomprime la lista `quaterna` nelle quattro variabili puntuali (P1, P2, P3, P4) tramite `mapcar 'set`. Valuta poi l'argomento `tipo` in un costrutto `cond` per instradare i punti alla funzione nativa `inters` (usata con parametro nil per operare su rette infinite). Il tipo 1 e 2 incrociano i lati opposti, il tipo 3 le diagonali.

Math_Inv_Sqrt

Firma: (Math_Inv_Sqrt x)

Scopo: Calcola il reciproco della radice quadrata di un valore: $1/\sqrt{x}$, formalmente equivalente all'elevamento a potenza -0.5.

Input: x: valore numerico reale positivo.

Output: Valore reale $1/\sqrt{x}$; nil o errore se $x \leq 0$.

Logica interna: Esegue (/ 1.0 (sqrt x)) con eventuale guard (> x 0.0) per proteggere dalla radice di zero o di un valore negativo.

Math_Lagrange_Term

Firma: (Math_Lagrange_Term val xi xj xk)

Scopo: Calcola il termine di base di Lagrange L_i(x) usato nell'interpolazione polinomiale lagrangiana: il peso dell'i-esimo nodo rispetto agli altri due nodi xj e xk.

Input: val: punto x in cui valutare il termine. xi: ascissa del nodo base (i). xj, xk: ascisse degli altri due nodi.

Output: Valore reale del termine L_i(val) = (val−xj)(val−xk) / (xi−xj)(xi−xk).

Logica interna: Chiama Math_Prod_Diff due volte: una per il numeratore (val−xj)(val−xk) e una per il denominatore (xi−xj)(xi−xk). Restituisce il quoziente.

Math_Line_General_Eq

Firma: (Math_Line_General_Eq Pu1 Pu2)

Scopo: Calcola i coefficienti (a,b,c) dell'equazione generale di una retta ax+by+c=0 passante per due punti dati.

Input: Pu1, Pu2: coordinate 2D dei due punti che definiscono la retta.

Output: Lista (a b c) con i coefficienti dell'equazione generale della retta.

Logica interna: Calcola a = y1−y2, b = x2−x1 e c = x1·y2−x2·y1 tramite Math_Diff_Prod, senza normalizzare i coefficienti.

Math_LisPuntiMedi_Quadrangolo_4Pu

Firma: (Math_LisPuntiMedi_Quadrangolo_4Pu P1 P2 P3 P4)

Scopo: Calcolare ed estrarre in un'unica operazione tutti i 6 punti medi di un quadrangolo completo (4 lati + 2 diagonali).

Input: P1, P2, P3, P4 (Coordinate spaziali dei 4 vertici del quadrangolo).

Output: Lista ordinata di 6 elementi spaziali corrispondenti ai punti medi: (M12 M23 M34 M41 M13 M24).

Logica interna: Riceve i vertici già separati in ingresso, saltando la fase di decompressione. Invoca direttamente il motore base Math_PuntoMedio_2Pu per generare la lista di output combinando le coppie fisiche, con la massima efficienza "bare metal".

Math_lisPuntoMedio_lis2Pu

Firma: (Math_lisPuntoMedio_lis2Pu lst_pairs)

Scopo: Calcolatore massivo (Array Processor) per estrarre i punti medi da una matrice di coppie.

Input: lst_pairs (Lista di sottoliste, dove ogni sottolista contiene esattamente una coppia di punti: '((P1 P2) (P3 P4) ...)).

Output: Lista ordinata dei punti medi calcolati.

Logica interna: Itera sull'array principale murendo il destrutturatore 'apply', che "spalma" istantaneamente gli elementi della coppia corrente sugli argomenti attesi dalla funzione base Math_PuntoMedio_2Pu.

Math_MinMax_List

Firma: (Math_MinMax_List lst flag)

Scopo: Ricavare i valori estremi (minimo e/o massimo) all'interno di un array di numeri, fungendo da motore statistico di base per il rilevamento di bounding box e limiti dimensionali.

Input: lst (Lista monodimensionale di valori numerici), flag (Intero selettore: 0 per il solo Minimo, 1 per il solo Massimo, 2 per estrarre la coppia combinata Min/Max).

Output: Numero singolo (se flag à 0 o 1) oppure una lista di due numeri `(Min Max)` (se flag à 2). Restituisce nil se la lista in ingresso risulta vuota o malformata.

Logica interna: Esegue un type-check logico iniziale sulla validità della lista. Sfruttando un costrutto `cond`valuta il `flag` e instrada l'array all'operatore di collasso nativo appropriato, applicando in modo fulmineo le primitive `min` o `max` tramite la funzione di spalmatura `apply`.

Math_Multiply_Lines_To_Conic

Firma: (Math_Multiply_Lines_To_Conic L1 L2)

Scopo: Costruisce la conica degenere (coppia di rette) data dal prodotto algebrico di due rette in forma generale, esprimendo il risultato come conica $Ax^2+Bxy+Cy^2+Dx+Ey+F=0$.

Input: L1, L2: liste (a1 b1 c1) e (a2 b2 c2) dei coefficienti delle due rette.

Output: Lista di 6 coefficienti [A B C D E F] della conica degenere L1·L2=0.

Logica interna: Effettua il prodotto simbolico dei due binomi: A=a1·a2, B=a1·b2+a2·b1, C=b1·b2, D=a1·c2+a2·c1, E=b1·c2+b2·c1, F=c1·c2, usando Math_Sum_Prod.

Math_Ortocentro_3Pu

Firma: (Math_Ortocentro_3Pu Pu1 Pu2 Pu3)

Scopo: Determinare la coordinata spaziale dell'ortocentro (intersezione delle altezze) di un triangolo.

Input: Pu1, Pu2, Pu3 (Coordinate 3D dei tre vertici del triangolo).

Output: Lista di coordinate 3D dell'ortocentro, oppure nil in caso di punti degeneri/allineati.

Logica interna: Risolve il punto calcolando al volo le due proiezioni ortogonali e incrociandole direttamente con 'inters' senza allocare variabili intermedie, massimizzando l'efficienza funzionale.

Math_Prod_Diff

Firma: (Math_Prod_Diff a b c d)

Scopo: Calcola il prodotto delle differenze: (a − b) × (c − d).

Input: a, b, c, d: valori numerici reali.

Output: Valore reale (a-b)·(c-d).

Logica interna: Usata la formula espansa come mattone per il termine di Lagrange.

Math_Prod_Sum

Firma: (Math_Prod_Sum a b c d)

Scopo: Calcola il prodotto della somma di due coppie di valori: (a + b) × (c + d).

Input: a, b, c, d: valori numerici reali.

Output: Valore numerico reale risultante dal prodotto delle due somme.

Logica interna: Somma le due coppie e moltiplica i risultati in un'unica espressione.

Math_Proiezione_1Pu_su_2Pu

Firma: (Math_Proiezione_1Pu_su_2Pu pup pul1 pul2)

Scopo: Calcolare la proiezione ortogonale di un punto spaziale su una retta definita da due punti noti.

Input: pup (Coordinate 3D del punto da proiettare), pul1 (Coordinate 3D del primo punto retta), pul2 (Coordinate 3D del secondo punto retta).

Output: Lista di coordinate 3D rappresentante il punto proiettato ortogonalmente.

Logica interna: Calcola i vettori differenza (v, u) e il prodotto scalare (d). Esegue la proiezione solo se il modulo quadro della direzione supera 1e-12, per prevenire divisioni per zero in caso di punti asse coincidenti.

Math_Proiezione_lisPu_su_2Pu

Firma: (Math_Proiezione_lisPu_su_2Pu lst_Pu Pu2 Pu3)

Scopo: Proiettare massivamente un array di punti (nuvola) su un'unica retta fissa di riferimento.

Input: lst_Pu (Lista di punti da proiettare), Pu2, Pu3 (Punti master che definiscono l'asse fisso).

Output: Lista ordinata di coordinate 3D dei punti proiettati sull'asse.

Logica interna: Sfrutta l'iteratore mapcar per scorrere la lista in ingresso. Ad ogni ciclo invia il punto corrente e l'asse fisso al motore di calcolo ad alte prestazioni Math_Proiezione_1Pu_su_2Pu.

Math_Proiezione_lisTerne

Firma: (Math_Proiezione_lisTerne lst_terne)

Scopo: Eseguire proiezioni ortogonali indipendenti per un array di terne, dove sia il punto che l'asse variano.

Input: lst_terne (Lista di sottoliste, dove ogni sottolista contiene '(Punto_da_Proiettare Asse_P1 Asse_P2)).

Output: Lista ordinata di coordinate 3D dei punti proiettati.

Logica interna: Utilizza mapcar per scorrere la lista madre. Ad ogni iterazione, usa 'apply' per spalmare istantaneamente la terna direttamente sugli argomenti del motore di base Math_Proiezione_1Pu_su_2Pu.

Math_PuntoMedio_2Pu

Firma: (Math_PuntoMedio_2Pu Pu1 Pu2)

Scopo: Calcolare le coordinate spaziali del punto medio esatto tra due estremi.

Input: Pu1, Pu2 (Liste di coordinate omologhe, 2D o 3D).

Output: Lista di coordinate del punto medio.

Logica interna: Sfrutta il parallelismo di 'mapcar' per mappare la lambda function sulle coordinate X, Y (e Z). La somma viene moltiplicata per 0.5 per massimizzare la velocità di calcolo in virgola mobile rispetto alla classica divisione.

Math_Rotate_Pt_Origin

Firma: (Math_Rotate_Pt_Origin Pu1 ang)

Scopo: Ruota un punto 2D attorno all'origine degli assi di un angolo dato, restituendo le nuove coordinate.

Input: pt: lista (x y) o (x y z) del punto da ruotare. ang: angolo di rotazione in radianti.

Output: Lista (x' y' 0.0) con le coordinate del punto ruotato.

Logica interna: Applica la formula della matrice di rotazione planare: x' = x·cos(ang) − y·sin(ang), y' = x·sin(ang) + y·cos(ang). Estrae le componenti x, y tramite `car`/`cadr` e combina i risultati in una lista 3D.

Math_Rot_Pt_2D_CS

Firma: (Math_Rot_Pt_2D_CS x y co si)

Scopo: Ruota un punto 2D dato il coseno e il seno dell'angolo (forma "co-si"), restituendo le coordinate ruotate come lista 3D.

Input: x, y: coordinate del punto. co: coseno dell'angolo. si: seno dell'angolo.

Output: Lista (x' y' 0.0) con le coordinate del punto ruotato.

Logica interna: Applica la matrice di rotazione 2D: x'= x·co − y·si, y'= x·si + y·co, usando Math_Diff_Prod e Math_Sum_Prod per efficienza numerica.

Math_Solve_5x5

Firma: (Math_Solve_5x5 mat vec)

Scopo: Risolvere un sistema di equazioni lineari 5x5 utilizzando il metodo di eliminazione di Gauss con pivoting parziale.

Input: mat (Lista di 5 liste, rappresentante la matrice dei coefficienti 5x5), vec (Lista di 5 numeri reali, rappresentante il vettore dei termini noti).

Output: Lista di 5 numeri reali rappresentante il vettore soluzione delle incognite, oppure nil se la matrice risulta singolare (determinante prossimo a zero).

Logica interna: Costruisce inizialmente la matrice aumentata affiancando il vettore dei termini noti. Applica l'eliminazione gaussiana con pivoting parziale (scambio delle righe per portare il valore assoluto massimo sulla diagonale) per garantire la massima stabilità numerica. Se un pivot scende sotto la soglia di tolleranza (1e-12), il sistema interrompe il calcolo per singolarità. Infine, esegue la sostituzione all'indietro (back substitution) per ricavare sequenzialmente il vettore delle incognite.

Math_Solve_Centered_Conic_3P_Ratio

Firma: (Math_Solve_Centered_Conic_3P_Ratio pts ratio conic_type)

Scopo: Calcolare il centro locale e il semiasse maggiore di una conica a centro (ellisse o iperbole) passante per 3 punti, noto il rapporto assi.

Input: pts (Lista 3 punti in coordinate locali), ratio (Numero reale, rapporto b/a), conic_type (Flag mnemonico 'ELLISSE o 'IPERBOLE).

Output: Lista contenente (Centro_Locale Semiasse_A) oppure nil in caso di configurazione geometrica degenere o soluzione non reale.

Logica interna: 1. PESO W: Determina il coefficiente relativo dell'asse Y (W = à1/r²). Il segno distingue tra forme chiuse (ellisse, +) e aperte (iperbole, -). 2. SISTEMA: Imposta un sistema 3x3 basato sull'equazione

generalizzata della conica centrata traslata: $x^2 + Wy^2 + xD' + yE' + F' = 0$. 3. RISOLUZIONE: Utilizza il solver di Cramer (Alg3x3_Risolvi) per determinare i parametri incogniti D', E', F'. 4. ESTRAZIONE: Converte i coefficienti in parametri fisici: Centro (Cx = -D'/2, Cy = -E'/(2W)) e Semiasse Maggiore (aà = $Cx^2 + W*Cy^2 - F'$).

Math_Solve_Cubic_Real

Firma: (Math_Solve_Cubic_Real A B_coeff C D)

Scopo: Calcolare analiticamente le radici reali di un'equazione polinomiale di terzo grado (A*x^3 + B*x^2 + C*x + D = 0).

Input: A, B_coeff, C, D (Numeri reali corrispondenti ai coefficienti del polinomio).

Output: Lista contenente esclusivamente le radici reali trovate (1, 2 o 3 elementi), oppure nil se nessuna radice reale à applicabile/computabile.

Logica interna: Prevede un fallback automatico per equazioni di grado inferiore. Nella risoluzione cubica pura, normalizza i coefficienti e valuta il discriminante (D_isc). Applica Viàte trigonometrico o Cardano algebrico ignorando le radici complesse.

Math_Solve_Ell_3P_Ratio

Firma: (Math_Solve_Ell_3P_Ratio pts ratio)

Scopo: Fornire un'interfaccia d'uso rapida e tipizzata per la risoluzione specifica di ellissi (apertura fissa) e iperboli (inclinazione asintoti).

Input: pts (Lista di 3 punti in coordinate locali), ratio (Numero reale, rapporto metrico b/a).

Output: Lista contenente (Centro_Locale Semiasse_A), passata in cascata dal risolutore principale, oppure nil.

Logica interna: Wrapper puri di delega. Semplificano la chiamata esterna invocando il motore algebrico unificato 'Math_Solve_Centered_Conic_3P_Ratio' e iniettando automaticamente, in trasparenza, il flag ('ELLISSE o 'IPERBOLE) corretto per l'instradamento logico del calcolo del peso W.

Math_Solve_Ipe_3P_Ratio

Firma: (Math_Solve_Ipe_3P_Ratio pts ratio)

Scopo: Fornire un'interfaccia d'uso rapida e tipizzata per la risoluzione specifica di ellissi (apertura fissa) e iperboli (inclinazione asintoti).

Input: pts (Lista di 3 punti in coordinate locali), ratio (Numero reale, rapporto metrico b/a).

Output: Lista contenente (Centro_Locale Semiasse_A), passata in cascata dal risolutore principale, oppure nil.

Logica interna: Wrapper puri di delega. Semplificano la chiamata esterna invocando il motore algebrico unificato 'Math_Solve_Centered_Conic_3P_Ratio' e iniettando automaticamente, in trasparenza, il flag ('ELLISSE o 'IPERBOLE) corretto per l'instradamento logico del calcolo del peso W. (defun Math_Solve_Ell_3P_Ratio (pts ratio) (Math_Solve_Centered_Conic_3P_Ratio pts ratio 'ELLISSE))

Math_Solve_Oblique_2P

Firma: (Math_Solve_Oblique_2P coords_A coords_B)

Scopo: Risolve il sistema di equazioni che fornisce i coefficienti della parabola con asse inclinato di un angolo dato, passante per esattamente 2 punti fissi e con vertice lungo l'asse.

Input: Ang: angolo dell'asse parabola in radianti. pts: lista dei 2 punti fissi.

Output: Lista (A_coeff p_dist) con il coefficiente angolare della parabola e la distanza fuoco-vertice; nil se il sistema è degenere.

Logica interna: Ruota i punti nel sistema locale dell'asse con Util_Rot_List_2D_CS, imposta e risolve il sistema lineare 2×2 che lega le coordinate locali all'equazione canonica $y^2 = 4px$ della parabola con asse orizzontale.

Math_Solve_Par_3P

Firma: (Math_Solve_Par_3P p1 p2 p3)

Scopo: Determina i coefficienti della parabola passante esattamente per 3 punti, con asse di simmetria orientato secondo un angolo parametrico dato.

Input: Ang: angolo dell'asse in radianti. P1, P2, P3: tre punti di vincolo.

Output: Lista dei coefficienti (a b c) dell'equazione y = ax^2+bx+c nel sistema locale dell'asse; nil se il determinante è nullo.

Logica interna: Ruota i 3 punti nel sistema locale tramite Util_Rot_List_2D_CS, poi risolve il sistema 3×3 con Alg3x3_Risolvi impostando le tre equazioni del tipo $y_i = a \cdot x_i^2+b \cdot x_i+c$.

Math_Solve_Par_Coeffs

Firma: (Math_Solve_Par_Coeffs Pu1 Pu2 Pu3)

Scopo: Risolve il sistema lineare per determinare i coefficienti (A,B,C) della conica generica passante per 3 punti con centro fisso dato, nella forma $ax_i^2+bx_iy_i+cy_i^2=1$ (sistema centrato).

Input: Center: coordinate del centro fisso. P1, P2, P3: tre punti di vincolo.

Output: Lista (A B C) dei coefficienti della forma quadratica centrata; nil se il determinante è nullo.

Logica interna: Traduce il centro e i punti nel sistema locale centrato sul centro (Alg3x3_Trasla), costruisce il vincolo di passaggio per ogni punto (Alg3x3_Vincolo), poi risolve il sistema 3×3 con Alg3x3_Risolvi.

Math_Solve_Quadratic

Firma: (Math_Solve_Quadratic a b c)

Scopo: Risoluzione analitica di un'equazione polinomiale di secondo grado (ax^2 + bx + c = 0).

Input: a, b, c (Coefficienti reali).

Output: Lista delle radici reali trovate, nil se il discriminante à negativo.

Logica interna: Calcola il Delta. Applica la formula risolutiva classica gestendo il caso di radice doppia o singola.

Math_Sqr

Firma: (Math_Sqr a)

Scopo: Calcola il quadrato di un valore numerico (x^2).

Input: x: valore numerico reale.

Output: Valore reale x^2.

Logica interna: Esegue una singola moltiplicazione (* x x), più veloce di (expt x 2).

Math_Sum_Prod

Firma: (Math_Sum_Prod a b c d)

Scopo: Calcola la somma di due prodotti: (a × b) + (c × d).

Input: a, b, c, d: valori numerici reali.

Output: Valore reale risultante dalla somma dei due prodotti.

Logica interna: Singola espressione (+ (* a b) (* c d)); usata come mattone nelle rotazioni 2D.

Math_Sum_Squares

Firma: (Math_Sum_Squares a b)

Scopo: Calcola la somma dei quadrati di due valori numerici ($a^2 + b^2$).

Input: a, b: valori numerici reali.

Output: Valore numerico reale corrispondente ad $a^2 + b^2$.

Logica interna: Esegue la moltiplicazione di ogni addendo per se stesso e ne restituisce la somma; evita la chiamata a `expt` per maggiore velocità.

Math_TriangoloDiagonale_4Pu

Firma: (Math_TriangoloDiagonale_4Pu Pu1 Pu2 Pu3 Pu4)

Scopo: Determinare i 3 vertici del triangolo diagonale generato dalle intersezioni dei lati opposti e delle diagonali di un quadrangolo completo.

Input: Pu1, Pu2, Pu3, Pu4 (Coordinate 3D dei vertici del quadrangolo).

Output: Lista di 3 coordinate 3D corrispondenti ai punti diagonali.

Logica interna: Invia al motore massivo 'Math_Inters_lisQuaterne' le tre permutazioni specifiche dei vertici necessarie per intersecare: lati opposti (1-2 con 3-4), diagonali (1-3 con 2-4) e lati opposti alternati (1-4 con 2-3).

Math_Uguaglianza_Relativa_2Re

Firma: (Math_Uguaglianza_Relativa_2Re Pu1 Pu2 Pu3 Pu4)

Scopo: Verifica se due valori reali sono uguali entro una tolleranza relativa proporzionale al loro ordine di grandezza, evitando i problemi di confronto in virgola mobile.

Input: a, b: valori numerici reali da confrontare.

Output: T se i due valori sono considerati uguali entro la tolleranza relativa; nil altrimenti.

Logica interna: Calcola il valore assoluto della differenza e lo confronta con il prodotto di una tolleranza relativa (tipicamente 1e-9) per il massimo dei valori assoluti dei due addendi.

P

c:par2PFAPI

Scopo: Comando Master per la Parabola a 2 Punti fissi con Fascio di Assi Proprio, Improprio o Libero.

Input: Parametri geometrici o di stato indicati nella firma.

Output: Risultato calcolato o effetto collaterale sul disegno.

Logica interna: Acquisisce P1/P2, imposta i layer (Poligoni/Costruzioni) e smista sui motori. Per l'improprio usa il punto medio come origine della direzione asse. Genera LV/LF e passa il controllo all'Universal Loop per l'interazione dinamica sub-pixel. MANUALE OPERATIVO: - CONFIGURAZIONE: Selezionare P1-P2. Scegliere 'Proprio' (asse rotante su P3) o 'Improprio' (asse parallelo a direzione definita da Mid P1P2). - INTERAZIONE: Nel loop, muovere il mouse vicino ai luoghi LV/LF per agganciare la parabola. 'V' o 'F' per toggler la visibilità. - MODALITA': Usare 'Modo' per passare tra 'Sostituisci' (curve mobili) e 'Aggiungi' (per creare famiglie di parabole depositate). - USCITA: Digitare 'U' o Invio per chiudere, consolidare le entità e pulire i supporti invisibili.

c:par2PFAPIRea

Scopo: Laboratorio Reattivo con raggruppamento LV/LF automatico ad ogni step e triangolo persistente.

Input: Coordinate interattive (P1, P2, P3) e opzioni di visibilità.

Output: Luoghi geometrici aggiornati in tempo reale e consolidati in HD.

Logica interna: Gestisce il disaccoppiamento della direzione tramite traslazione rigida di P3. Esegue raggruppamento dei punti ad ogni rigenerazione per sicurezza del database.

PAR2PFAPI_Calc_F_Linear

Firma: (PAR2PFAPI_Calc_F_Linear pt_F P1 P2 u_x u_y n_x n_y const_num const_den)

Scopo: Ricavare analiticamente la distanza focale 'f' di una parabola passante per due punti base fissi (P1, P2), dato un fuoco campionato.

Input: pt_F (Coordinate fuoco), P1/P2 (Punti base), u_x/u_y (Versore asse), n_x/n_y (Versore normale), const_num/const_den (Costanti pre-calcolate).

Output: Distanza focale calcolata (Real con segno relativo all'apertura).

Logica interna: Esegue la proiezione dei punti base nel sistema di coordinate locale centrato sul fuoco. Sfrutta una derivazione lineare delle equazioni di appartenenza per estrarre 'f' senza l'ausilio di radici quadrate, risolvendo l'ambiguità del verso di apertura.

PAR2PFAPI_Draw_Engine

Firma: (PAR2PFAPI_Draw_Engine P1 P2 payload opt_m last_ents)

Scopo: Driver grafico deputato all'estrazione del payload e al rendering istantaneo della parabola "elastica".

Input: P1/P2 (Punti base), payload (Pacchetto geometrico), opt_m (Modo), last_ents (Memoria entità).

Output: Handle (ename) delle nuove entità tracciate.

Logica interna: Gestisce la sostituzione delle entità precedenti se in modo 'Sostituisci'. Estrae Vertice e Fuoco dal payload e invoca il driver standard Draw_ConPar_VeFu_Compl fissando l'estensione geometrica della curva.

PAR2PFAPI_Draw_Locus_Entities_Rea

Firma: (PAR2PFAPI_Draw_Locus_Entities_Rea res_map opt_gv opt_gf)

Scopo: Generare i luoghi geometrici come nuvole di punti colorati separati.

Input: res_map (Payload), opt_gv (Visibilità V), opt_gf (Visibilità F).

Output: Lista di due liste: (Lista_Ename_V Lista_Ename_F).

Logica interna: Cicla sui payload e disegna ogni nodo come punto discreto. Vertici in Celeste (Colore 5), Fuochi in Verde (Colore 3).

PAR2PFAPI_Draw_Skeleton_Rea

Firma: (PAR2PFAPI_Draw_Skeleton_Rea P1 P2 P3 f_type)

Scopo: Feedback visivo del triangolo di controllo (P1, P2, P3).

Input: P1, P2, P3 (Punti), f_type (Fascio).

Output: Lista di ename delle entità di costruzione create.

Logica interna: Traccia la base P1-P2 (Layer Poligoni) e i lati P1-P3, P2-P3 (Layer Costruzioni). Include il vettore direzione dal punto medio se in modalità Improprio.

PAR2PFAPI_Engine_Improprio

Firma: (PAR2PFAPI_Engine_Improprio P1 P2 vec_dir opt_e)

Scopo: Generare le mappe discrete ad alta densità per il fascio improprio di parabole, sfruttando l'iperbole "Light" (solo rami) come luogo dei fuochi.

Input: P1 e P2 (Punti fissi), vec_dir (Vettore direzione asse), opt_e (Fattore estensione).

Output: Lista strutturata (Lista_Payload_V Lista_Payload_F Lista_Oggetti_VLA_Iperbole).

Logica interna: Calcola i parametri dell'iperbole esatta passante per P1 e P2. Disegna i soli rami iperbolici (Light) e li campiona via VLA. Per ogni nodo, calcola f via PAR2PFAPI_Calc_F_Linear e retrocede V sul vettore asse. Le iperboli sono rese invisibili e conservate nella vla_list per permettere il raffinamento continuo (proiezione su curva) durante il loop interattivo.

PAR2PFAPI_Engine_Proprio

Firma: (PAR2PFAPI_Engine_Proprio P1 P2 P3 opt_e)

Scopo: Generare le mappe discrete per il fascio proprio di parabole (asse rotante su P3), applicando l'infittimento "Matrioska" sulle singolarità.

Input: P1 e P2 (Punti fissi), P3 (Perno di rotazione), opt_e (Fattore estensione).

Output: Lista strutturata (Lista_Payload_V Lista_Payload_F nil).

Logica interna: Calcola gli angoli critici asintotici. Genera uno spettro angolare [0, PI) infittito localmente. Risolve algebricamente la parabola estrando il payload completo (V, F, angolo, f). Applica tagli di distanza massima per evitare la dispersione infinita dei luoghi.

PAR2PFAPI_Fractal_Around_Param_Safe

Firma: (PAR2PFAPI_Fractal_Around_Param_Safe param_center step_base)

Scopo: Iniettare una cascata di micro-parametri di infittimento spaziale (struttura a "Matrioska") attorno a una singolarità matematica (asintoto).

Input: param_center (Parametro di schianto o asintoto), step_base (Passo di scansione master del motore).

Output: Lista di micro-parametri addensati in modo logaritmico.

Logica interna: Genera rampe simmetriche di avvicinamento al valore critico applicando frazioni progressive (1/16, 1/8, 1/4) del passo base. Questa tecnica permette di mappare i rami infiniti con alta densità senza causare il crash del risolutore per denominatore nullo.

PAR2PFAPI_Get_Top3_Indices

Firma: (PAR2PFAPI_Get_Top3_Indices pt_mouse flat_map)

Scopo: Individuare i tre migliori candidati (indici) nella mappa di payload per gestire zone di sovrapposizione (fiocchi) o rami multipli.

Input: pt_mouse (Punto cursore), flat_map (Lista di payload campionati).

Output: Lista di 3 elementi, ognuno dei quali à una sottolista: (Distanza Indice_nella_mappa).

Logica interna: Scansiona l'intera mappa calcolando le distanze e mantiene una classifica ordinata dei 3 punti più vicini. Questo permette al raffinamento successivo di analizzare fino a tre rami diversi contemporaneamente se non sono contigui.

PAR2PFAPI_Helper_Nearest_Control_Pt

Firma: (PAR2PFAPI_Helper_Nearest_Control_Pt pt_click P1 P2 P3)

Scopo: Identificare quale dei tre punti di controllo (P1, P2 o P3) à più vicino al clic dell'utente.

Input: pt_click (Punto), P1, P2, P3 (Punti attuali).

Output: Simbolo del punto più vicino ('P1, 'P2 o 'P3).

Logica interna: Calcola le distanze euclidee e restituisce il tag del punto con distanza minima per la ridefinizione dinamica.

PAR2PFAPI_Helper_UI_Prompt

Firma: (PAR2PFAPI_Helper_UI_Prompt opt_gf opt_gv opt_m)

Scopo: Generare in modo dinamico la stringa di prompt interattivo per il ciclo continuo LUSNAP, riflettendo lo stato dei toggle di visibilità.

Input: opt_gf (Stato visibilità Fuochi), opt_gv (Stato visibilità Vertici), opt_m (Modo operativo corrente).

Output: Stringa testuale formattata per la riga di comando AutoCAD.

Logica interna: Inizializza le parole chiave ammissibili tramite initget. Costruisce una stringa complessa che concatena le variabili di stato (ON/OFF) e la modalità (Sostituisci/Aggiungi) per fornire un feedback visivo immediato all'utente durante il puntamento del mouse.

PAR2PFAPI_Libero_Runner

Firma: (PAR2PFAPI_Libero_Runner P1 P2)

Scopo: Gestire la generazione interattiva di parabole in modalità libera, utilizzando la ricerca euristica a 360 gradi e la dima topologica.

Input: P1 e P2 (Punti fissi di passaggio).

Output: Tracciamento dinamico di parabole basato sulla posizione del vertice selezionato.

Logica interna: Implementa la ricerca dicotomica per trovare l'inclinazione dell'asse compatibile con V. Visualizza i confini topologici (dima) tramite Helper_Update_Dima_Par2pv per guidare l'utente verso le zone di esistenza della conica.

PAR2PFAPI_Modulo_Pi

Firma: (PAR2PFAPI_Modulo_Pi ang_val)

Scopo: Normalizzare un valore angolare espresso in radianti all'interno del semipiano positivo standard [0, PI).

Input: ang_val (Valore numerico angolare in radianti).

Output: Valore angolare normalizzato (Real).

Logica interna: Applica l'operatore resto (rem) rispetto alla costante pi greco. Qualora il risultato sia negativo, lo trasla nel semipiano positivo sommandogli pi greco, garantendo la coerenza dei vettori direzione dell'asse della parabola.

PAR2PFAPI_Refine_Locus_Ultra

Firma: (PAR2PFAPI_Refine_Locus_Ultra pt_mouse P1 P2 map_data f_type vla_list P3_ref vec_ref active_loc)

Scopo: Motore di raffinamento sub-pixel con gestione Fiocchi (Top 3) e correzione di fase per Vertici. LOGICA DI PRECISIONE: - PROPRIO: Scansione angolare tra i vicini (i-1, i+1). - IMPROPRIO: Se cerco V, uso il Fuoco del payload come ancora per lo scan sull'iperbole.

Input: Parametri geometrici o di stato indicati nella firma.

Output: Risultato calcolato o effetto collaterale sul disegno.

PAR2PFAPI_Universal_Loop

Firma: (PAR2PFAPI_Universal_Loop P1 P2 flat_map_V ents_V flat_map_F ents_F f_type vla_list P3_ref vec_ref)

Scopo: Gestore interattivo asincrono per la manipolazione in tempo reale della parabola tramite aggancio ai luoghi geometrici (LUSNAP).

Input: P1/P2 (Base), flat_map_V/F (Mappe), ents_V/F (Entità punti), f_type (Fascio), vla_list (Supporti VLA), P3_ref (Perno), vec_ref (Direzione).

Output: Esecuzione dinamica e gestione dell'interfaccia utente.

Logica interna: Gestisce i toggle di visibilità e la modalità operativa. Interroga PAR2PFAPI_Refine_Locus ad ogni movimento del mouse per piloter il Draw_Engine. In uscita garantisce la distruzione degli oggetti VLA invisibili per pulire il database del disegno.

c:par2PV

Scopo: Comando per disegnare la parabola passante per 2 punti fissi P1 e P2 con un terzo punto mobile V come vertice. Ad ogni posizione di V il sistema scansiona gli assi possibili e trova la parabola (o le parabole) che passano per P1, P2 e hanno V come vertice, con opzione Guide per mostrare i confini topologici della sagoma Dima.

Input: Nessun parametro: selezione interattiva di 2 punti fissi, poi getpoint agganciato per il vertice V. Opzioni da tastiera: [Guide] ON/OFF, [Modo], [Uscita].

Output: Parabola/e disegnate nel documento per la posizione corrente di V; princ alla fine.

Logica interna: Scansiona l'intervallo angolare [0,π] con passi Step, valuta Math_Solve_Oblique_2P per ogni Ang, raccoglie le soluzioni, filtra i duplicati, poi chiama Draw_ConPar_VeFu_Compl per ogni soluzione trovata.

c:par3P

Scopo: Comando Master per l'ambiente statico Parabola 3P. Gestisce l'acquisizione dei punti, il calcolo e il rendering del loop LUSNAP.

Input: Nessuno (Interattivo da riga di comando CAD, richiede la selezione manuale di 3 punti base a schermo).

Output: Nessuno (Side-effect: disegna il triangolo base, i luoghi VLA e avvia l'interfaccia di puntamento analitico su nodi e curve).

Logica interna: Inizializza le variabili di sistema e i colori, poi acquisisce la terna. Definisce il limite di estensione asintotica in base alla dimensione del triangolo. Genera le curve VLA (luoghi di vertici e fuochi) e precalcola le parabole esatte passanti per i nodi. Cronometra le operazioni di calcolo e cede il controllo al ciclo interattivo universale per l'esplorazione geometrica a video.

c:par3PRea

Scopo: Avviare l'ambiente interattivo di editing reattivo per la generazione dinamica dei luoghi geometrici delle parabole passanti per 3 punti.

Input: Nessuno (Comando lanciato da riga di comando CAD).

Output: Nessuno (Avvia il loop interattivo tramite side-effect).

Logica interna: Inizializza i codici colore standard (ACI) per i vertici e i fuochi delle parabole. Innesca il master reattivo universale che gestisce l'acquisizione dei punti base e l'aggiornamento in tempo reale dei luoghi grafici in risposta allo spostamento dei nodi nello spazio.

PAR3P_Calc_Par_V_2P

Firma: (PAR3P_Calc_Par_V_2P pt_v pt_1 pt_2)

Scopo: Calcolare analiticamente i parametri esatti dell'unica parabola passante per 3 punti, forzando il primo come Vertice.

Input: pt_v (Vertice), pt_1 (Punto di passaggio 1), pt_2 (Punto di passaggio 2).

Output: Lista (Vertice Fuoco) o nil.

Logica interna: Sweep a 180à con dicotomica locale di precisione per trovare lo zero-crossing dell'equazione continua.

PAR3P_Geom_Get_Closest_Guide_Target

Firma: (PAR3P_Geom_Get_Closest_Guide_Target pt_in guide_list min_d best_c pt_proj)

Scopo: Iterare su una lista di entità guida per trovare il punto più vicino al click dell'utente, aggiornando i record di distanza minima.

Input: pt_in (Punto mouse), guide_list (Lista ENAME delle curve), min_d (Distanza minima corrente), best_c (Entità vincente), pt_proj (Punto proiettato).

Output: Lista aggiornata (min_d best_c pt_proj).

PAR3P_Helper_UI_Prompt

Firma: (PAR3P_Helper_UI_Prompt opt_gf opt_gv opt_t opt_m)

Scopo: Centralizzare e generare la stringa di prompt per l'interfaccia testuale interattiva del comando Master.

Input: opt_gf, opt_gv (Stato visibilità Fuochi/Vertici), opt_t (Modalità Target), opt_m (Modalità inserimento Aggiungi/Sostituisci).

Output: Stringa di testo formattata, pronta per l'utilizzo all'interno della funzione getpoint nel loop principale.

Logica interna: Inizializza le keyword accettate dal motore CAD tramite initget per prevenire input errati. Costruisce e restituisce una stringa dinamica (HUD testuale) che mostra all'utente lo stato corrente delle variabili di sistema e le opzioni di interazione disponibili.

PAR3P_Master_Par_3P

Firma: (PAR3P_Master_Par_3P pts dist_limit)

Scopo: Calcolo luoghi e generazione entità CAD raggruppate.

Input: pts (3 Punti), dist_limit (Taglio rami).

Output: Lista (guide_V map_V guide_F map_F).

Logica interna: Utilizza Safe_Collect_Ents e Anonymous_Group per l'allineamento con ELL3P/IPE3P.

PAR3P_Master_Reactive_Universal_3P

Firma: (PAR3P_Master_Reactive_Universal_3P titolo)

Scopo: Controller per la modalità di editing reattivo (live editing) dei luoghi geometrici delle parabole passanti per 3 punti.

Input: titolo (Stringa testuale per l'intestazione del prompt di inizializzazione del comando).

Output: Nessuno (Agisce per side-effect aggiornando e rigenerando dinamicamente le entità grafiche sullo schermo durante il trascinamento).

Logica interna: Acquisisce la terna di base e definisce un helper (update_locus) per il ricalcolo rapido. Avvia un loop interattivo in cui l'utente può accendere/spegnere i livelli visivi o selezionare un nodo per spostarlo. A ogni spostamento, la routine aggiorna la lista base, distrugge

i vecchi luoghi e rigenera l'impalcatura in tempo reale. All'uscita dal ciclo, forza un ricalcolo finale per consolidare il disegno.

PAR3P_Precalc_Exact_Nodes

Firma: (PAR3P_Precalc_Exact_Nodes pts)

Scopo: Elaborare preventivamente l'Oracolo Geometrico per i Nodi base.

Input: pts (Lista dei 3 vertici base).

Output: Mappa associativa Point -> Soluzione analitica.

Logica interna: Cicla i 3 punti come vertici potenziali calcolando la soluzione esatta.

PAR3P_Universal_Loop

Firma: (PAR3P_Universal_Loop pts guide_V map_V guide_F map_F tri_scale exact_map)

Scopo: Controller interattivo per l'esplorazione e la generazione mirata (LUSNAP) delle parabole passanti per la terna di punti base.

Input: pts (Terna base), guide_V/F e map_V/F (Entità e mappe dati per vertici/fuochi), tri_scale (Scala tolleranza), exact_map (Nodi precalcolati).

Output: Nessuno (Side-effect: genera le parabole a schermo, gestisce la pulizia temporanea e distrugge le guide grafiche all'uscita).

Logica interna: Loop infinito interrotto da "Uscita". Gestisce l'interfaccia CAD per i toggle di visibilità e i metodi di input (Nodo/Luogo). Selezionando un punto, calcola la distanza minima dalle spline VLA attive, ne estrae il segmento topologico base, lo affina tramite proiezione matematica WCS e lancia il motore di disegno. In modalità Nodo, pesca direttamente la soluzione analitica esatta per un aggancio assoluto.

c:parVF

Scopo: Comando laboratorio interattivo per disegnare una parabola specificando il punto BASE come ancora fissa (vertice o fuoco a scelta) e il secondo punto come mobile (fuoco o vertice rispettivamente), con loop continuo che permette di ridisegnare la parabola cambiando il secondo punto senza dover riselezionare l'ancora.

Input: Nessun parametro: input interattivo (getpoint per il punto base, poi getpoint agganciato al punto base per il secondo punto). Opzioni da tastiera: [Ordinamento] scambia ruolo vertice/fuoco, [Modo] alterna Sostituisci/Aggiungi, [Fine] termina.

Output: Parabola disegnata nel documento AutoCAD con assi e fuoco; princ alla fine del loop.

Logica interna: Init_Conic_App salva le variabili di sistema. Il loop while principale aggiorna opt_ord tra "V-F" e "F-V"; per ogni nuovo punto chiama Draw_ConPar_VeFu_Compl con vertice e fuoco scambiati secondo opt_ord; in modalità Sostituisci cancella l'entità precedente prima di ridisegnare.

pascal-core-attempt

Firma: (pascal-core-attempt curr_pts)

Scopo: Tentativo nucleo dell'algoritmo di Pascal: esegue un singolo tentativo di trovare il sesto punto a partire da una permutazione specifica dei 5 punti dati, restituendo il risultato o nil in caso di fallimento.

Input: perm: lista di 5 punti in un ordine specifico (una delle 12 permutazioni utili).

Output: Coordinate del sesto punto calcolato; nil se la geometria è degenere.

Logica interna: Chiama Pascal_Calcola_Corde_Coniugate e Pascal_Intersezione_Diametri sulla permutazione data e ne restituisce il risultato.

Pascal_Calcola_Corde_Coniugate

Firma: (Pascal_Calcola_Corde_Coniugate pts ang_dir)

Scopo: Calcola le corde coniugate di Pascal per i 5 punti dati, individuando le intersezioni tra i lati opposti dell'esagono parziale.

Input: P1..P5: cinque punti della conica su cui applicare il teorema di Pascal parzialmente.

Output: Lista di due punti (diametri intermedi di Pascal) usati per determinare il 6° punto.

Logica interna: Chiama Pascal_Calcola_Primo_Diametro per ogni coppia di lati opposti dell'esagono parziale a 5 punti e restituisce i due diametri calcolati.

Pascal_Calcola_Primo_Diametro

Firma: (Pascal_Calcola_Primo_Diametro pts_6)

Scopo: Calcola il primo diametro del teorema di Pascal per una conica: l'intersezione dei lati P1P2 e P4P5 dell'esagono inscritto nella conica.

Input: P1, P2, P4, P5: quattro dei sei vertici dell'esagono di Pascal.

Output: Coordinate del punto di intersezione dei due lati dati (primo diametro di Pascal); nil se paralleli.

Logica interna: Calcola le equazioni generali delle rette P1P2 e P4P5 con Math_Line_General_Eq, poi risolve il sistema 2×2 con Alg3x3_Risolvi per trovare il punto di intersezione.

Pascal_Calcola_Singola_Corda

Firma: (Pascal_Calcola_Singola_Corda pA pB pC pD pE ang_dir)

Scopo: Calcola una singola corda dell'esagono di Pascal: la retta passante per due dei sei vertici dati, restituita in forma di equazione generale.

Input: PA, PB: due punti (vertici dell'esagono di Pascal).

Output: Lista (a b c) dei coefficienti della retta PA-PB in forma generale ax+by+c=0.

Logica interna: Delega a Math_Line_General_Eq(PA, PB).

Pascal_Genera_Sesto_Punto

Firma: (Pascal_Genera_Sesto_Punto pts)

Scopo: Applica il teorema di Pascal per determinare il sesto punto di una conica: dati 5 punti sulla conica, trova il sesto usando le intersezioni delle corde dell'esagono di Pascal.

Input: P1..P5: cinque punti sulla conica. dir (opzionale): direzione preferenziale.

Output: Coordinate del sesto punto della conica costruito per via pascaliana; nil se degenere.

Logica interna: Calcola i due diametri di Pascal (intersezioni di lati opposti), traccia la retta che li congiunge (asse di Pascal), poi interseca con la retta P5-P1 per trovare P6.

Pascal_Intersezione_Diametri

Firma: (Pascal_Intersezione_Diametri diam1_pts chords_pts pts_base)

Scopo: Calcola il punto di intersezione tra le due rettediametri di Pascal, ricongiungendo i due assi dell'esagono per trovare il punto di Pascal.

Input: D1, D2: liste (a1 b1 c1) e (a2 b2 c2) dei coefficienti dei due diametri in forma generale.

Output: Coordinate del punto di Pascal (intersezione dei due diametri); nil se paralleli.

Logica interna: Risolve il sistema 2×2 lineare D1·P = D2·P con Alg3x3_Risolvi o equivalente formula di Cramer diretta.

Pitag_Ca

Firma: (Pitag_Ca ip ca)

Scopo: Calcola la lunghezza di un cateto di un triangolo rettangolo data l'ipotenusa e l'altro cateto: ca = √(ip² − ca_noto²).

Input: ip: ipotenusa. ca: cateto noto.

Output: Valore reale della lunghezza del cateto incognito; nil se ip ≤ ca o se gli argomenti sono nil/non validi.

Logica interna: Verifica che ip > ca prima di calcolare (protezione dalla radice di numero negativo). Usa Math_Diff_Squares per ip²-ca², poi ne estrae la radice quadrata.

Pitag_Ip

Firma: (Pitag_Ip ca1 ca2)

Scopo: Calcola l'ipotenusa di un triangolo rettangolo date le due cateti, applicando il teorema di Pitagora: ip = √(ca1² + ca2²).

Input: ca1, ca2: lunghezze dei due cateti (valori numerici reali).

Output: Valore reale della lunghezza dell'ipotenusa; nil se un argomento è nil.

Logica interna: Chiama Math_Sum_Squares per ottenere ca1²+ca2², poi ne estrae la radice quadrata con sqrt. Usa and per proteggere dalla chiamata con nil.

PrincPu

Firma: (PrincPu txt pu dec)

Scopo: Stampa a schermo la rappresentazione testuale di un punto o di una lista di punti, utile per il debug interattivo in AutoCAD.

Input: pu: punto (lista di coordinate) o lista di punti da stampare.

Output: Effetto collaterale: testo del punto stampato sulla command line di AutoCAD. Restituisce nil.

Logica interna: Usa `princ` con `strcat` per formattare le coordinate del punto in una stringa leggibile e stamparla sulla riga di comando di AutoCAD.

Princ_1Int

Firma: (Princ_1Int txt n)

Scopo: Fornire una suite di funzioni di log e stampa formattata per inviare output strutturati alla console di AutoCAD. Combinano etichette testuali personalizzate con la rappresentazione in stringa di vari tipi di dati (Punti, Simboli, Numeri, Interi).

Input: txt (Stringa di prefisso opzionale, es. un'etichetta o un messaggio). Seguono i valori specifici da stampare: Pu1, Pu2 (Coordinate), sym (Simbolo LISP), num, n1, n2 (Numeri reali), n (Numero Intero).

Output: Restituisce il valore stringa stampato. Genera come side-effect visivo la stampa del messaggio assemblato direttamente sulla riga di comando di AutoCAD.

Logica interna: Ciascun wrapper verifica innanzitutto l'esistenza del prefisso testuale 'txt' (sostituendolo con stringa vuota se nil). Utilizza quindi l'operatore 'strcat' per concatenare il prefisso con il risultato delle utilità di conversione 'Util_*_To_Str' dedicate. Il pacchetto stringa finale viene inviato alla console tramite la funzione primitiva 'princ'. (defun Princ_1Pun (txt Pu1) (princ (strcat (if txt txt "") (Util_Pun_To_Str Pu1)))) (defun Princ_2Pun (txt Pu1 Pu2) (princ (strcat (if txt txt "") (Util_Pun_To_Str Pu1) " - " (Util_Pun_To_Str Pu2)))) (defun Princ_1Sym (txt sym) (princ (strcat (if txt txt "") (Util_Sym_To_Str sym)))) (defun Princ_1Num (txt num) (princ (strcat (if txt txt "") (Util_Num_To_Str num)))) (defun Princ_2Num (txt n1 n2) (princ (strcat (if txt txt "") (Util_Num_To_Str n1) " , " (Util_Num_To_Str n2))))

Princ_1Num

Firma: (Princ_1Num txt num)

Scopo: Fornire una suite di funzioni di log e stampa formattata per inviare output strutturati alla console di AutoCAD. Combinano etichette testuali personalizzate con la rappresentazione in stringa di vari tipi di dati (Punti, Simboli, Numeri, Interi).

Input: txt (Stringa di prefisso opzionale, es. un'etichetta o un messaggio). Seguono i valori specifici da stampare: Pu1, Pu2 (Coordinate), sym (Simbolo LISP), num, n1, n2 (Numeri reali), n (Numero Intero).

Output: Restituisce il valore stringa stampato. Genera come side-effect visivo la stampa del messaggio assemblato direttamente sulla riga di comando di AutoCAD.

Logica interna: Ciascun wrapper verifica innanzitutto l'esistenza del prefisso testuale 'txt' (sostituendolo con stringa vuota se nil). Utilizza quindi l'operatore 'strcat' per concatenare il prefisso con il risultato delle utilità di conversione 'Util_*_To_Str' dedicate. Il pacchetto stringa finale viene inviato alla console tramite la funzione primitiva 'princ'. (defun Princ_1Pun (txt Pu1) (princ (strcat (if txt txt "") (Util_Pun_To_Str Pu1)))) (defun Princ_2Pun (txt Pu1 Pu2) (princ (strcat (if txt txt "") (Util_Pun_To_Str Pu1) " - " (Util_Pun_To_Str Pu2)))) (defun Princ_1Sym (txt sym) (princ (strcat (if txt txt "") (Util_Sym_To_Str sym))))

Princ_1Pun

Firma: (Princ_1Pun txt Pu1)

Scopo: Fornire una suite di funzioni di log e stampa formattata per inviare output strutturati alla console di AutoCAD. Combinano etichette testuali personalizzate con la rappresentazione in stringa di vari tipi di dati (Punti, Simboli, Numeri, Interi).

Input: txt (Stringa di prefisso opzionale, es. un'etichetta o un messaggio). Seguono i valori specifici da stampare: Pu1, Pu2 (Coordinate), sym (Simbolo LISP), num, n1, n2 (Numeri reali), n (Numero Intero).

Output: Restituisce il valore stringa stampato. Genera come side-effect visivo la stampa del messaggio assemblato direttamente sulla riga di comando di AutoCAD.

Logica interna: Ciascun wrapper verifica innanzitutto l'esistenza del prefisso testuale 'txt' (sostituendolo con stringa vuota se nil). Utilizza quindi l'operatore 'strcat' per concatenare il prefisso con il risultato delle utilità di conversione 'Util_*_To_Str' dedicate. Il pacchetto stringa finale viene inviato alla console tramite la funzione primitiva 'princ'.

Princ_1Sym

Firma: (Princ_1Sym txt sym)

Scopo: Fornire una suite di funzioni di log e stampa formattata per inviare output strutturati alla console di AutoCAD. Combinano etichette testuali personalizzate con la rappresentazione in stringa di vari tipi di dati (Punti, Simboli, Numeri, Interi).

Input: txt (Stringa di prefisso opzionale, es. un'etichetta o un messaggio). Seguono i valori specifici da stampare: Pu1, Pu2 (Coordinate), sym (Simbolo LISP), num, n1, n2 (Numeri reali), n (Numero Intero).

Output: Restituisce il valore stringa stampato. Genera come side-effect visivo la stampa del messaggio assemblato direttamente sulla riga di comando di AutoCAD.

Logica interna: Ciascun wrapper verifica innanzitutto l'esistenza del prefisso testuale 'txt' (sostituendolo con stringa vuota se nil). Utilizza quindi l'operatore 'strcat' per concatenare il prefisso con il risultato delle utilità di conversione 'Util_*_To_Str' dedicate. Il pacchetto stringa finale viene inviato alla console tramite la funzione primitiva 'princ'. (defun Princ_1Pun (txt Pu1) (princ (strcat (if txt txt "") (Util_Pun_To_Str Pu1)))) (defun Princ_2Pun (txt Pu1 Pu2) (princ (strcat (if txt txt "") (Util_Pun_To_Str Pu1) " - " (Util_Pun_To_Str Pu2))))

Princ_2Num

Firma: (Princ_2Num txt n1 n2)

Scopo: Fornire una suite di funzioni di log e stampa formattata per inviare output strutturati alla console di AutoCAD. Combinano etichette testuali personalizzate con la rappresentazione in stringa di vari tipi di dati (Punti, Simboli, Numeri, Interi).

Input: txt (Stringa di prefisso opzionale, es. un'etichetta o un messaggio). Seguono i valori specifici da stampare: Pu1, Pu2 (Coordinate), sym (Simbolo LISP), num, n1, n2 (Numeri reali), n (Numero Intero).

Output: Restituisce il valore stringa stampato. Genera come side-effect visivo la stampa del messaggio assemblato direttamente sulla riga di comando di AutoCAD.

Logica interna: Ciascun wrapper verifica innanzitutto l'esistenza del prefisso testuale 'txt' (sostituendolo con stringa vuota se nil). Utilizza quindi l'operatore 'strcat' per concatenare il prefisso con il risultato delle utilità di conversione 'Util_*_To_Str' dedicate. Il pacchetto stringa finale viene inviato alla console tramite la funzione primitiva 'princ'. (defun Princ_1Pun (txt Pu1) (princ (strcat (if txt txt "") (Util_Pun_To_Str Pu1)))) (defun Princ_2Pun (txt Pu1 Pu2) (princ (strcat (if txt txt "") (Util_Pun_To_Str Pu1) " - " (Util_Pun_To_Str Pu2)))) (defun Princ_1Sym (txt sym) (princ (strcat (if txt txt "") (Util_Sym_To_Str sym)))) (defun Princ_1Num (txt num) (princ (strcat (if txt txt "") (Util_Num_To_Str num))))

Princ_2Pun

Firma: (Princ_2Pun txt Pu1 Pu2)

Scopo: Fornire una suite di funzioni di log e stampa formattata per inviare output strutturati alla console di AutoCAD. Combinano etichette testuali personalizzate con la rappresentazione in stringa di vari tipi di dati (Punti, Simboli, Numeri, Interi).

Input: txt (Stringa di prefisso opzionale, es. un'etichetta o un messaggio). Seguono i valori specifici da stampare: Pu1, Pu2 (Coordinate), sym (Simbolo LISP), num, n1, n2 (Numeri reali), n (Numero Intero).

Output: Restituisce il valore stringa stampato. Genera come side-effect visivo la stampa del messaggio assemblato direttamente sulla riga di comando di AutoCAD.

Logica interna: Ciascun wrapper verifica innanzitutto l'esistenza del prefisso testuale 'txt' (sostituendolo con stringa vuota se nil). Utilizza quindi l'operatore 'strcat' per concatenare il prefisso con il risultato delle utilità di

conversione 'Util_*_To_Str' dedicate. Il pacchetto stringa finale viene inviato alla console tramite la funzione primitiva 'princ'. (defun Princ_1Pun (txt Pu1) (princ (strcat (if txt txt "") (Util_Pun_To_Str Pu1))))

probe-conic-algebraic-tipo

Firma: (probe-conic-algebraic-tipo pts center ref_p1 ref_p2)

Scopo: Classifica algebricamente il tipo di conica (ellisse, iperbole o parabola) a partire dai suoi 6 coefficienti, calcolando il discriminante della forma quadratica.

Input: K: lista [A B C D E F] dei coefficienti della conica.

Output: Simbolo: 'ELLISSE se $B^2-4AC < 0$; 'IPERBOLE se $B^2-4AC > 0$; 'PARABOLA se ≈ 0.

Logica interna: Calcola il discriminante delta = B^2-4AC con Math_Sqr e confronta con ±1e-9 per classificare la conica.

Pu_Baricentro2D_LisPu

Firma: (Pu_Baricentro2D_LisPu LisPu)

Scopo: Calcolare il baricentro geometrico di un insieme di punti, determinando la media algebrica delle sole coordinate X e Y.

Input: LisPu (Lista di coordinate, bidimensionali o tridimensionali, rappresentanti i vertici del poligono o la nuvola di punti).

Output: Una lista di tre numeri reali (X Y 0.0) corrispondente alle coordinate del baricentro, schiacciato con quota Z forzata a zero.

Logica interna: Estrae la quantità di punti in float. Applica un nesting funzionale diretto ('apply' e 'mapcar') per isolare, sommare e mediare al volo le componenti X e Y. La nomenclatura '2D' esplicita l'appiattimento sul piano, prevenendo ambiguità d'uso nel 3D.

P_0Coinc_4Pu

Firma: (P_0Coinc_4Pu Pu1 Pu2 Pu3 Pu4)

Scopo: Verifica che nessuna coppia di punti tra i quattro dati sia coincidente.

Input: Pu1, Pu2, Pu3, Pu4: coordinate dei quattro punti.

Output: T se tutti e 4 i punti sono distinti; nil se almeno due coincidono.

Logica interna: Usa Math_Conta_Coinc_4Pu e verifica che il risultato sia 0.

P_0Orto_4Pu

Firma: (P_0Orto_4Pu Pu1 Pu2 Pu3 Pu4)

Scopo: Verifica che nessun angolo interno del quadrangolo sia retto.

Input: Pu1, Pu2, Pu3, Pu4: coordinate dei quattro punti.

Output: T se 0 angoli interni sono retti; nil altrimenti.

Logica interna: Usa Math_Conta_Ortogo_4Pu e verifica che il risultato sia 0.

P_0Paral_4Pu

Firma: (P_0Paral_4Pu Pu1 Pu2 Pu3 Pu4)

Scopo: Verifica che nessuna coppia di lati opposti del quadrangolo con 4 punti sia parallela.

Input: Pu1, Pu2, Pu3, Pu4: coordinate dei quattro punti.

Output: T se 0 coppie di lati opposti sono parallele; nil altrimenti.

Logica interna: Usa Math_Conta_Paral_4Pu e verifica che il risultato sia 0.

P_1Coinc_4Pu

Firma: (P_1Coinc_4Pu Pu1 Pu2 Pu3 Pu4)

Scopo: Verifica che esattamente una coppia di punti tra i quattro dati coincida.

Input: Pu1, Pu2, Pu3, Pu4: coordinate dei quattro punti.

Output: T se esattamente 1 coppia di punti coincide; nil altrimenti.

Logica interna: Usa Math_Conta_Coinc_4Pu e verifica che il risultato sia 1.

P_1Orto_4Pu

Firma: (P_1Orto_4Pu Pu1 Pu2 Pu3 Pu4)

Scopo: Verifica che esattamente un angolo interno del quadrangolo sia retto.

Input: Pu1, Pu2, Pu3, Pu4: coordinate dei quattro punti.

Output: T se esattamente 1 angolo è retto; nil altrimenti.

Logica interna: Usa Math_Conta_Ortogo_4Pu e verifica che il risultato sia 1.

P_1Paral_4Pu

Firma: (P_1Paral_4Pu Pu1 Pu2 Pu3 Pu4)

Scopo: Verifica che esattamente una coppia di lati opposti del quadrangolo sia parallela (trapezio).

Input: Pu1, Pu2, Pu3, Pu4: coordinate dei quattro punti.

Output: T se esattamente 1 coppia di lati opposti è parallela (figura è un trapezio); nil altrimenti.

Logica interna: Usa Math_Conta_Paral_4Pu e verifica che il risultato sia esattamente 1.

P_2Coinc_4Pu

Firma: (P_2Coinc_4Pu Pu1 Pu2 Pu3 Pu4)

Scopo: Verifica che esattamente due coppie di punti tra i quattro dati coincidano.

Input: Pu1, Pu2, Pu3, Pu4: coordinate dei quattro punti.

Output: T se esattamente 2 coppie distinte coincidono; nil altrimenti.

Logica interna: Usa Math_Conta_Coinc_4Pu e verifica che il risultato sia 2.

P_2Orto_4Pu

Firma: (P_2Orto_4Pu Pu1 Pu2 Pu3 Pu4)

Scopo: Verifica che esattamente due angoli interni del quadrangolo siano retti.

Input: Pu1, Pu2, Pu3, Pu4: coordinate dei quattro punti.

Output: T se esattamente 2 angoli sono retti; nil altrimenti.

Logica interna: Usa Math_Conta_Ortogo_4Pu e verifica che il risultato sia 2.

P_2Paral_4Pu

Firma: (P_2Paral_4Pu Pu1 Pu2 Pu3 Pu4)

Scopo: Verifica che entrambe le coppie di lati opposti del quadrangolo siano parallele (parallelogramma o derivati).

Input: Pu1, Pu2, Pu3, Pu4: coordinate dei quattro punti.

Output: T se 2 coppie di lati opposti sono parallele (parallelogramma, rettangolo, rombo, quadrato); nil altrimenti.

Logica interna: Usa Math_Conta_Paral_4Pu e verifica che il risultato sia 2.

P_3Coinc_4Pu

Firma: (P_3Coinc_4Pu Pu1 Pu2 Pu3 Pu4)

Scopo: Verifica che almeno 3 punti tra i quattro dati coincidano.

Input: Pu1, Pu2, Pu3, Pu4: coordinate dei quattro punti.

Output: T se almeno 3 punti coincidono; nil altrimenti.

Logica interna: Usa Math_Conta_Coinc_4Pu e verifica che il risultato sia ≥ 3.

P_3Orto_4Pu

Firma: (P_3Orto_4Pu Pu1 Pu2 Pu3 Pu4)

Scopo: Verifica che almeno 3 angoli interni del quadrangolo siano retti (equivalente a 4, poiché la somma degli angoli è sempre 360°).

Input: Pu1, Pu2, Pu3, Pu4: coordinate dei quattro punti.

Output: T se ≥ 3 angoli sono retti (figura è un rettangolo o quadrato); nil altrimenti.

Logica interna: Usa Math_Conta_Ortogo_4Pu e verifica che il risultato sia ≥ 3.

P_6Coinc_4Pu

Firma: (P_6Coinc_4Pu Pu1 Pu2 Pu3 Pu4)

Scopo: Verifica l'improbabile caso in cui tutti e 4 i punti coincidano (6 coppie totali tutte coincidenti = degenere totale).

Input: Pu1, Pu2, Pu3, Pu4: coordinate dei quattro punti.

Output: T se tutti e 4 i punti sono coincidenti (unico punto); nil altrimenti.

Logica interna: Usa Math_Conta_Coinc_4Pu e verifica che il risultato sia 6 (tutte le C(4,2) coppie).

P_Allin_3Pu

Firma: (P_Allin_3Pu Pu1 Pu2 Pu3)

Scopo: Verifica se tre punti sono allineati calcolando l'area del triangolo che formano; se l'area è nulla o trascurabile i punti sono collineari.

Input: Pu1, Pu2, Pu3: coordinate 2D o 3D dei tre punti.

Output: T se i tre punti sono allineati entro la tolleranza; nil se formano un triangolo reale.

Logica interna: Calcola la somma ciclica (area orientata) con Math_Cyclic_Det e la confronta con la soglia 1e-10; oppure usa distance e triangle-area se disponibile.

P_Allin_3su4Pu

Firma: (P_Allin_3su4Pu Pu1 Pu2 Pu3 Pu4 Only)

Scopo: Verifica se almeno 3 dei 4 punti dati sono allineati (configurazione degenere).

Input: Pu1, Pu2, Pu3, Pu4: coordinate dei quattro punti.

Output: T se almeno una tripletta di punti è allineata; nil se tutti i punti sono in posizione generica.

Logica interna: Verifica le 4 triplette C(4,3) possibili con P_Allin_3Pu e le unisce con or.

P_Allin_4Pu

Firma: (P_Allin_4Pu Pu1 Pu2 Pu3 Pu4)

Scopo: Verifica se tutti e quattro i punti dati sono allineati sulla stessa retta.

Input: Pu1, Pu2, Pu3, Pu4: coordinate dei quattro punti.

Output: T se i 4 punti sono collineari; nil altrimenti.

Logica interna: Verifica l'allineamento di due triplette (P1-P2-P3 e P1-P2-P4) con P_Allin_3Pu; restituisce T solo se entrambi gli allineamenti sono veri.

P_Allin_List_3Pu

Firma: (P_Allin_List_3Pu lst_triples)

Scopo: Verifica l'allineamento di 3 punti ricevuti come lista unica anziché come argomenti separati.

Input: lst: lista di 3 elementi, ciascuno una lista di coordinate.

Output: T se i tre punti sono allineati; nil altrimenti.

Logica interna: Destruttura la lista con mapcar'set e delega a P_Allin_3Pu.

P_Are_Collinear

Firma: (P_Are_Collinear Pu1 Pu2 Pu3)

Scopo: Verifica se 3 punti sono collineari (allineati sulla stessa retta). Alias di P_Allin_3Pu con nome inglese per retrocompatibilità.

Input: Pu1, Pu2, Pu3: coordinate dei tre punti.

Output: T se i punti sono allineati; nil altrimenti.

Logica interna: Delega direttamente a P_Allin_3Pu; fornisce un'interfaccia di denominazione alternativa con il nome inglese Are_Collinear per uniformità con altri predicati.

P_Are_Vectors_Parallel

Firma: (P_Are_Vectors_Parallel v1 v2)

Scopo: Verifica se due vettori 3D sono paralleli tra loro calcolando il modulo del loro prodotto vettoriale e confrontandolo con il prodotto dei moduli.

Input: v1, v2: liste di coordinate 3D dei due vettori.

Output: T se i vettori sono paralleli entro la tolleranza relativa; nil altrimenti.

Logica interna: Calcola il modulo del prodotto vettoriale (Vet_Cro) tramite distance dall'origine. Se il rapporto cp_mag/(|v1|·|v2|) < 1e-4 i vettori sono paralleli.

P_Are_Vectors_Parallel_2D

Firma: (P_Are_Vectors_Parallel_2D v1 v2 tol)

Scopo: Verifica il parallelismo di due vettori 2D confrontando le pendenze tramite prodotto incrociato (cross product planare).

Input: v1, v2: liste di coordinate 2D (o 3D con z ignorata) dei due vettori.

Output: T se i vettori 2D sono paralleli entro la tolleranza; nil altrimenti.

Logica interna: Calcola il prodotto vettoriale 2D z = v1x·v2y − v1y·v2x e verifica che il suo valore assoluto sia inferiore alla soglia di tolleranza.

P_center-infinite

Firma: (P_center-infinite pts center threshold)

Scopo: Determina se una conica ha il centro all'infinito, ovvero se è una parabola, verificando che il determinante della matrice 2×2 dei coefficienti quadratici sia nullo.

Input: K: lista [A B C D E F] dei coefficienti della conica.

Output: T se $4AC-B^2 \approx 0$ (centro all'infinito, parabola); nil se la conica ha centro finito.

Logica interna: Calcola det = $4AC-B^2$ e lo confronta con la soglia 1e-9; se |det| < soglia restituisce T.

P_Coinc_2Pu

Firma: (P_Coinc_2Pu Pu1 Pu2)

Scopo: Verifica se due punti coincidono, cioè se la loro distanza euclidea è inferiore alla tolleranza geometrica di coincidenza.

Input: Pu1, Pu2: coordinate 2D o 3D dei due punti.

Output: T se i punti sono coincidenti entro la tolleranza; nil se distinti.

Logica interna: Confronta la distanza calcolata da Math_Dist_2Pu con una soglia fissa (tipicamente 1e-6 unità di disegno).

P_Coinc_2su3Pu

Firma: (P_Coinc_2su3Pu Pu1 Pu2 Pu3 Only)

Scopo: Verifica se esattamente due (e non tutti e tre) i punti dati coincidono.

Input: Pu1, Pu2, Pu3: coordinate dei tre punti.

Output: T se esattamente 2 punti su 3 coincidono; nil altrimenti.

Logica interna: Valuta separatamente ogni coppia e verifica che almeno una coppia sia coincidente ma non tutte e tre; esclude il caso di tripla coincidenza.

P_Coinc_3Pu

Firma: (P_Coinc_3Pu Pu1 Pu2 Pu3)

Scopo: Verifica se almeno due dei tre punti dati coincidono.

Input: Pu1, Pu2, Pu3: coordinate dei tre punti.

Output: T se almeno una coppia di punti è coincidente; nil se tutti distinti.

Logica interna: Valuta le tre coppie (P1-P2, P1-P3, P2-P3) con P_Coinc_2Pu e le unisce con or.

P_Coinc_4Pu

Firma: (P_Coinc_4Pu Pu1 Pu2 Pu3 Pu4)

Scopo: Verifica se almeno due dei quattro punti dati coincidono.

Input: Pu1, Pu2, Pu3, Pu4: coordinate dei quattro punti.

Output: T se almeno una coppia di punti è coincidente; nil se tutti distinti.

Logica interna: Valuta le sei coppie C(4,2) con P_Coinc_2Pu e le unisce con or.

P_Coinc_List_2Pu

Firma: (P_Coinc_List_2Pu lst_pairs)

Scopo: Verifica la coincidenza tra due punti quando almeno uno di essi può essere fornito come lista (eventualmente nidificata), gestendo la destrutturazione automatica.

Input: Pu1: punto o lista contenente un punto. Pu2: punto.

Output: T se il punto estratto da Pu1 e Pu2 coincidono; nil altrimenti.

Logica interna: Controlla se Pu1 è una lista di liste (punto nidificato) e lo destruttura con car; poi delega il confronto a P_Coinc_2Pu.

P_Concavo_Generico_4Pu

Firma: (P_Concavo_Generico_4Pu Pu1 Pu2 Pu3 Pu4)

Scopo: Verifica se il quadrangolo formato da 4 punti è concavo ma non ortocentrico.

Input: Pu1, Pu2, Pu3, Pu4: coordinate dei quattro punti.

Output: T se concavo e non ortocentrico; nil altrimenti.

Logica interna: AND di P_Conca_4Pu (concavo) con la negazione di P_Concavo_Ortocentrico_4Pu.

P_Concavo_Ortocentrico_4Pu

Firma: (P_Concavo_Ortocentrico_4Pu Pu1 Pu2 Pu3 Pu4)

Scopo: Verifica se il quadrangolo concavo è ortocentrico: il punto interno è l'ortocentro del triangolo formato dagli altri tre vertici.

Input: Pu1, Pu2, Pu3, Pu4: coordinate dei quattro punti.

Output: T se la configurazione è concava con il punto interno uguale all'ortocentro; nil altrimenti.

Logica interna: Calcola l'ortocentro del triangolo degli altri tre punti e verifica con P_Coinc_2Pu se coincide con il punto interno rilevato da P_Conca_4Pu.

P_Conca_4Pu

Firma: (P_Conca_4Pu Pu1 Pu2 Pu3 Pu4)

Scopo: Verifica se il quadrangolo formato da 4 punti è concavo, cioè se uno dei quattro punti è interno al triangolo formato dagli altri tre.

Input: Pu1, Pu2, Pu3, Pu4: coordinate dei quattro punti.

Output: T se il quadrangolo è concavo; nil se convesso o degenere.

Logica interna: Testa tutte e 4 le combinazioni "un punto dentro gli altri tre" usando P_Pu_Inter_3Pu; restituisce T se almeno una è verificata.

P_Conciclici_4Pu

Firma: (P_Conciclici_4Pu Pu1 Pu2 Pu3 Pu4)

Scopo: Verifica se i 4 punti giacciono tutti su una stessa circonferenza (concicli).

Input: Pu1, Pu2, Pu3, Pu4: coordinate dei quattro punti.

Output: T se i 4 punti sono concicli entro la tolleranza numerica; nil altrimenti.

Logica interna: Delega a Math_Check_Conciclicita_4Pu passando i 4 punti come argomenti.

P_Conve_4Pu

Firma: (P_Conve_4Pu Pu1 Pu2 Pu3 Pu4)

Scopo: Verifica se il quadrangolo formato da 4 punti è convesso, cioè se nessun punto è interno al triangolo degli altri tre.

Input: Pu1, Pu2, Pu3, Pu4: coordinate dei quattro punti.

Output: T se convesso; nil se concavo o degenere.

Logica interna: Negazione di P_Conca_4Pu: T se e solo se nessuno dei 4 punti è interno al triangolo degli altri tre.

P_Has_Collinear_Triplet

Firma: (P_Has_Collinear_Triplet pts)

Scopo: Verifica se in un set di punti esiste almeno una tripletta collineare (allineata).

Input: pts: lista di punti (3 o più).

Output: T se almeno 3 punti del set sono allineati; nil se tutti in posizione generica.

Logica interna: Itera su tutte le triplette del set con un ciclo annidato e applica P_Allin_3Pu; restituisce T al primo allineamento trovato.

P_Is_Config_Concave

Firma: (P_Is_Config_Concave pts)

Scopo: Verifica se 4 punti formano una configurazione concava (quadrangolo non convesso), cioè se uno dei punti è interno al triangolo formato dagli altri tre.

Input: pts: lista di 4 punti.

Output: T se la configurazione è concava; nil se convessa o degenere.

Logica interna: Wrapper di P_Conca_4Pu che accetta lista invece di 4 argomenti separati, dopo aver verificato che la lista abbia esattamente 4 elementi.

P_Is_Orthocentric_System

Firma: (P_Is_Orthocentric_System pts)

Scopo: Verifica se 4 punti formano un sistema ortocentrico: ogni punto è l'ortocentro del triangolo formato dagli altri tre.

Input: pts: lista di 4 punti.

Output: T se i 4 punti formano un sistema ortocentrico; nil altrimenti.

Logica interna: Calcola l'ortocentro del triangolo formato da 3 dei 4 punti e verifica con P_Coinc_2Pu se coincide con il 4° punto; ripete per tutte le combinazioni.

P_Parallelogramma_4Pu

Firma: (P_Parallelogramma_4Pu Pu1 Pu2 Pu3 Pu4)

Scopo: Verifica se il quadrangolo è un parallelogramma: entrambe le coppie di lati opposti sono parallele, ma nessun angolo interno è retto.

Input: Pu1, Pu2, Pu3, Pu4: coordinate dei quattro punti.

Output: T se è un parallelogramma generico; nil altrimenti.

Logica interna: AND di P_2Paral_4Pu (entrambe le coppie parallele) e P_0Orto_4Pu (nessun angolo retto).

P_Paral_2Re

Firma: (P_Paral_2Re Re1A Re1B Re2A Re2B)

Scopo: Verifica se due rette sono parallele tra loro, confrontando le loro direzioni.

Input: L1, L2: coppie di punti o coefficienti che definiscono le due rette.

Output: T se le rette sono parallele; nil altrimenti.

Logica interna: Calcola la direzione (angolo) di ciascuna retta tramite la funzione angle di AutoLISP; verifica che la differenza angolare (mod π) sia inferiore alla tolleranza angolare.

P_Paral_4Pu

Firma: (P_Paral_4Pu Pu1 Pu2 Pu3 Pu4)

Scopo: Verifica se i lati opposti del quadrangolo formato da 4 punti sono paralleli (almeno una coppia), accettando 4 argomenti separati.

Input: Pu1, Pu2, Pu3, Pu4: coordinate dei quattro vertici.

Output: T se almeno una coppia di lati opposti è parallela; nil altrimenti.

Logica interna: Verifica il parallelismo tra P1-P2 e P3-P4, e tra P2-P3 e P4-P1 tramite P_Paral_2Re; restituisce T se almeno una verifica è positiva.

P_Paral_distinte_2Re

Firma: (P_Paral_distinte_2Re Re1A Re1B Re2A Re2B)

Scopo: Verifica se due rette sono parallele E distinte (non coincidenti).

Input: L1, L2: definizioni delle due rette (coppie di punti o coefficienti).

Output: T se le rette sono parallele ma non coincidenti; nil altrimenti.

Logica interna: AND di P_Paral_2Re (parallele) con il complemento di P_Coinc per la verifica che non siano la stessa retta (distanza tra le rette ≠ 0).

P_Paral_distinte_List_2Re

Firma: (P_Paral_distinte_List_2Re lst_pairs_pairs)

Scopo: Verifica se due rette (definite come liste di punti) sono parallele e distinte.

Input: L1, L2: liste (p1 p2) delle due rette.

Output: T se parallele e distinte; nil altrimenti.

Logica interna: Estrae i punti e delega a P_Paral_distinte_2Re.

P_Paral_distinti_4Pu

Firma: (P_Paral_distinti_4Pu Pu1 Pu2 Pu3 Pu4)

Scopo: Verifica se i lati opposti del quadrangolo sono paralleli E distinti (cioè il quadrangolo è un parallelogramma o derivato).

Input: Pu1, Pu2, Pu3, Pu4: coordinate dei quattro vertici.

Output: T se almeno una coppia di lati opposti è parallela e distinta; nil altrimenti.

Logica interna: Come P_Paral_4Pu ma usando P_Paral_distinte_2Re invece di P_Paral_2Re.

P_Paral_List_2Re

Firma: (P_Paral_List_2Re lst_pairs_pairs)

Scopo: Verifica il parallelismo tra due rette definite come liste di punti.

Input: L1, L2: liste (p1 p2) che definiscono le due rette come coppie di punti.

Output: T se le rette definite dalle liste sono parallele; nil altrimenti.

Logica interna: Estrae i punti dalle liste e delega a P_Paral_2Re.

P_Perpe_2Re

Firma: (P_Perpe_2Re Re1A Re1B Re2A Re2B)

Scopo: Verifica se due rette sono perpendicolari tra loro.

Input: L1, L2: definizioni delle due rette (coppie di punti o coefficienti).

Output: T se le rette sono perpendicolari; nil altrimenti.

Logica interna: Calcola la differenza angolare tra le direzioni delle due rette e verifica che il valore assoluto (mod π) sia prossimo a π/2 entro la tolleranza angolare.

P_Perpe_List_2Re

Firma: (P_Perpe_List_2Re lst_pairs_pairs)

Scopo: Verifica la perpendicolarità tra due rette definite come liste di punti.

Input: L1, L2: liste (p1 p2) delle due rette.

Output: T se le rette sono perpendicolari; nil altrimenti.

Logica interna: Estrae i punti e delega a P_Perpe_2Re.

P_Pu_Ester_3Pu

Firma: (P_Pu_Ester_3Pu Pu1 Pu2 Pu3 Pu4)

Scopo: Verifica se un punto P si trova all'esterno del triangolo P1-P2-P3.

Input: P: punto da testare. P1, P2, P3: vertici del triangolo.

Output: T se P è esterno al triangolo; nil se interno o sul bordo.

Logica interna: Negazione logica di P_Pu_Inter_3Pu: restituisce T se e solo se la somma delle aree dei sotto-triangoli supera l'area totale del triangolo originale.

P_Pu_Ester_List_4Pu

Firma: (P_Pu_Ester_List_4Pu lst_quartets)

Scopo: Verifica se un punto è esterno al quadrangolo definito da una lista di 4 punti.

Input: P: punto da testare. pts: lista di 4 punti del quadrangolo.

Output: T se P è esterno al quadrangolo; nil se interno.

Logica interna: Negazione di P_Pu_Inter_List_4Pu.

P_Pu_Inter_3Pu

Firma: (P_Pu_Inter_3Pu Pu1 Pu2 Pu3 Pu4)

Scopo: Verifica se un punto P si trova all'interno (o sul bordo) del triangolo P1-P2-P3.

Input: P: punto da testare. P1, P2, P3: vertici del triangolo.

Output: T se P è interno o sul bordo del triangolo; nil se esterno.

Logica interna: Confronta la somma delle aree dei tre sotto-triangoli P-P1-P2, P-P2-P3, P-P3-P1 con l'area del triangolo originale con Math_Aree_Pu_vs_Tri; T se la somma ≈ area totale (entro tolleranza).

P_Pu_Inter_List_4Pu

Firma: (P_Pu_Inter_List_4Pu lst_quartets)

Scopo: Verifica se un punto è interno al quadrangolo definito da una lista di 4 punti, testando l'appartenenza al triangolo formato da ogni tripla di vertici.

Input: P: punto da testare. pts: lista di 4 punti del quadrangolo.

Output: T se P è interno al quadrangolo; nil se esterno.

Logica interna: Suddivide il quadrangolo nelle due diagonali triangolari e usa P_Pu_Inter_3Pu su entrambe: T se P è interno ad almeno una delle due triangolazioni.

P_Quadrangolo_4Pu

Firma: (P_Quadrangolo_4Pu Pu1 Pu2 Pu3 Pu4)

Scopo: Verifica che i 4 punti formino un quadrangolo valido: nessuna coppia coincidente, nessuna tripletta allineata, forma non degenere.

Input: Pu1, Pu2, Pu3, Pu4: coordinate dei quattro punti.

Output: T se i 4 punti formano un quadrangolo valido; nil se la configurazione è degenere.

Logica interna: AND di P_0Coinc_4Pu (nessuna coincidenza) con not(P_Allin_3su4Pu) (nessun allineamento di triplette).

P_Quadrangolo_Generico_4Pu

Firma: (P_Quadrangolo_Generico_4Pu Pu1 Pu2 Pu3 Pu4)

Scopo: Verifica se il quadrangolo è generico: nessun lato parallelo, nessun angolo retto, nessuna degenerazione (la configurazione più generica possibile).

Input: Pu1, Pu2, Pu3, Pu4: coordinate dei quattro punti.

Output: T se quadrangolo generico (0 paralleli, 0 ortogonali, non degenere); nil altrimenti.

Logica interna: AND di P_Quadrangolo_4Pu, P_0Paral_4Pu e P_0Orto_4Pu.

P_Quadrato_4Pu

Firma: (P_Quadrato_4Pu Pu1 Pu2 Pu3 Pu4)

Scopo: Verifica se il quadrangolo è un quadrato: rombo con tutti gli angoli retti.

Input: Pu1, Pu2, Pu3, Pu4: coordinate dei quattro punti.

Output: T se è un quadrato; nil altrimenti.

Logica interna: AND di P_Rettangolo_4Pu (rettangolo) con la verifica che tutti i lati abbiano la stessa lunghezza (rombo).

P_Rettangolo_4Pu

Firma: (P_Rettangolo_4Pu Pu1 Pu2 Pu3 Pu4)

Scopo: Verifica se il quadrangolo è un rettangolo: due coppie di lati paralleli e tutti e 4 gli angoli interni retti.

Input: Pu1, Pu2, Pu3, Pu4: coordinate dei quattro punti.

Output: T se è un rettangolo; nil altrimenti.

Logica interna: AND di P_2Paral_4Pu (parallelogramma) e P_3Orto_4Pu (tutti angoli retti).

P_Rombo_4Pu

Firma: (P_Rombo_4Pu Pu1 Pu2 Pu3 Pu4)

Scopo: Verifica se il quadrangolo è un rombo: due coppie di lati paralleli, tutti i lati di uguale lunghezza, ma angoli non necessariamente retti.

Input: Pu1, Pu2, Pu3, Pu4: coordinate dei quattro punti.

Output: T se è un rombo; nil altrimenti.

Logica interna: AND di P_2Paral_4Pu con la verifica che tutti i lati abbiano la stessa lunghezza entro tolleranza, e NOT P_Rettangolo_4Pu.

P_Trapezio_4Pu

Firma: (P_Trapezio_4Pu Pu1 Pu2 Pu3 Pu4)

Scopo: Verifica se il quadrangolo formato da 4 punti è un trapezio: esattamente una coppia di lati opposti è parallela.

Input: Pu1, Pu2, Pu3, Pu4: coordinate dei quattro punti.

Output: T se la figura è un trapezio; nil altrimenti.

Logica interna: AND di P_Quadrangolo_4Pu (validità) e P_1Paral_4Pu (una sola coppia parallela).

P_Trapezio_Isoscele_4Pu

Firma: (P_Trapezio_Isoscele_4Pu Pu1 Pu2 Pu3 Pu4)

Scopo: Verifica se il quadrangolo è un trapezio isoscele: una sola coppia di lati opposti paralleli e i due lati obliqui di uguale lunghezza.

Input: Pu1, Pu2, Pu3, Pu4: coordinate dei quattro punti.

Output: T se è trapezio isoscele; nil altrimenti.

Logica interna: AND di P_Trapezio_4Pu con la verifica che i due lati obliqui abbiano la stessa lunghezza entro tolleranza (equal (distance P1 P4) (distance P2 P3) tol).

P_Trapezio_Scaleno_4Pu

Firma: (P_Trapezio_Scaleno_4Pu Pu1 Pu2 Pu3 Pu4)

Scopo: Verifica se il quadrangolo è un trapezio scaleno: una sola coppia di lati opposti paralleli e i lati obliqui non uguali tra loro.

Input: Pu1, Pu2, Pu3, Pu4: coordinate dei quattro punti.

Output: T se è trapezio scaleno; nil altrimenti.

Logica interna: AND di P_Trapezio_4Pu (è un trapezio) con la negazione di P_Trapezio_Isoscele_4Pu.

R

replace-nth

Firma: (replace-nth lst n val)

Scopo: Ricostruire e restituire una nuova lista in cui l'elemento corrispondente all'indice specificato à stato sostituito dinamicamente con un nuovo valore, lasciando inalterati gli altri elementi.

Input: lst (La lista di partenza da elaborare), n (Numero intero zero-based indicante l'indice dell'elemento da sostituire), val (Il nuovo valore o elemento da inserire nella posizione target).

Output: Una nuova lista clonata contenente la sostituzione richiesta.

Logica interna: Itera la lista in ingresso tramite `foreach` e un contatore `i`. Un costrutto `if` valuta l'indice: se `i` uguaglia `n`, inietta `val`, altrimenti conserva l'elemento originale. Per massimizzare le prestazioni, utilizza `cons` per l'inserimento in testa (evitando il costoso `append` nei cicli) e ripristina l'ordine cronologico tramite `reverse` al termine del ciclo.

Restore_Conic_App

Firma: (Restore_Conic_App old_os old_pd old_ce)

Scopo: Ripristina le variabili di sistema AutoCAD (OSMODE, PDMODE, CMDECHO) ai valori salvati prima dell'avvio del comando, garantendo il cleanup dell'ambiente alla conclusione.

Input: old_os, old_pd, old_ce: valori originali di OSMODE, PDMODE, CMDECHO restituiti da Init_Conic_App; possono essere nil se non erano stati modificati.

Output: Effetto collaterale: variabili di sistema ripristinate. Restituisce princ.

Logica interna: Usa setvar condizionale (if old_os ...) per ripristinare solo i valori che erano stati effettivamente modificati.

rotate-pt-90

Firma: (rotate-pt-90 pt center)

Scopo: Ruota un punto 2D di esattamente 90° in senso antiorario attorno all'origine.

Input: pt: lista (x y) del punto.

Output: Lista (−y x 0.0) del punto ruotato di 90° CCW.

Logica interna: Applica la formula semplificata della rotazione di 90°: (x,y) → (−y, x), senza calcolare seno/coseno grazie al caso speciale.

rotate-pt-origin

Firma: (rotate-pt-origin pt ang)

Scopo: Ruota un punto 2D attorno all'origine degli assi. Funzione a denominazione kebab-case (stile Scheme) usata in contesti legacy.

Input: pt: lista (x y) del punto. ang: angolo di rotazione in radianti.

Output: Lista (x' y' 0.0) del punto ruotato.

Logica interna: Estrae x e y, applica la matrice di rotazione planare con Math_Sum_Prod e Math_Diff_Prod.

S

Safe_Collect_Ents

Firma: (Safe_Collect_Ents e_start)

Scopo: Raccogliere in modo sicuro tutte le entità del database di AutoCAD create a partire da un determinato puntatore in poi.

Input: e_start (Ename dell'entità di partenza. Se nil, la scansione parte dalla prima entità in assoluto presente nel file DWG).

Output: Lista contenente gli ename di tutte le entità successive a e_start (salvate in ordine inverso di estrazione tramite cons).

Logica interna: Valuta l'entità di partenza e posiziona il cursore di lettura del database (entnext). Attraversa sequenzialmente il file DWG con un ciclo while, accumulando progressivamente i puntatori in una lista fino all'esaurimento degli oggetti validi.

c:slc

Scopo: Comando di configurazione: inizializza e crea i layer standard del sistema Coniche nel documento AutoCAD corrente.

Input: Nessun input richiesto: esecuzione diretta da prompt AutoCAD.

Output: Layer di lavoro creati nel documento (Luoghi geometrici, Coniche, Guide, ecc.). Princ alla conclusione.

Logica interna: Chiama `Config_Conic_Layers` che itera sulla lista dei layer standard del sistema e li crea con `Lay_Check`/`Lay_Make_Col` se non esistono già.

solve-system-cramer-2x2

Firma: (solve-system-cramer-2x2 a1 b1 c1 a2 b2 c2)

Scopo: Risolvere analiticamente un sistema lineare composto da due equazioni in due incognite (2x2) applicando il metodo algebrico di Cramer.

Input: a1, b1, c1, a2, b2, c2 (Numeri reali che rappresentano i coefficienti e i termini noti delle due equazioni nella forma standard ax + by = c).

Output: Lista di due numeri reali (X Y) corrispondenti ai valori delle incognite risolte. Restituisce nil se la matrice à singolare (determinante prossimo allo zero), indicando rette parallele o coincidenti.

Logica interna: Calcola inizialmente il determinante della matrice principale delegando il prodotto incrociato all'helper `Math_Diff_Prod`. Se il valore assoluto supera la soglia di tolleranza di sicurezza (1e-9), procede a calcolare i determinanti delle matrici sostituite, dividendoli per il principale per estrarre X e Y.

Solve_And_Draw_From_Coeffs

Firma: (Solve_And_Draw_From_Coeffs Coeffs PuC_Offset lay_crv lay_constr conic_type scale_factor)

Scopo: Risolve i parametri geometrici dai coefficienti della conica e disegna la conica in AutoCAD, incluse le linee degli assi e le posizioni dei fuochi.

Input: Coeffs: lista [A B C D E F]. PuC_Offset: centro di denormalizzazione. lay_crv: layer conica. lay_constr: layer costruzioni. conic_type: tipo conica. scale_factor: fattore di scala.

Output: Entità AutoCAD disegnate (conica + assi + fuochi). Effetto collaterale sul disegno.

Logica interna: Estrae i dati geometrici con Math_Extract_Geom_Data_From_Coeffs, denormalizza fuochi e vertici con Util_Denorm_Pt, poi chiama le primitive Draw_ appropriate per tipo (ellisse o iperbole), aggiungendo gli assi e marcatori dei fuochi sul layer costruzioni.

Solve_Par_Geometry

Firma: (Solve_Par_Geometry pts ref1 ref2)

Scopo: Risolve completamente la geometria di una parabola nota la sua equazione parametrica nel sistema locale: calcola fuoco, vertice, direttrice e semiparametro.

Input: a: coefficiente principale. b: termine lineare. c: termine costante (nel sistema locale).

Output: Lista (fuoco vertice semi-parametro angolo-asse) della parabola nel sistema locale.

Logica interna: Dalla forma canonica $y=ax^2+bx+c$ ricava il vertice come $-b/(2a)$, il semi-parametro $p=1/(4a)$, il fuoco spostato di p dal vertice.

ssadd_list

Firma: (ssadd_list ent_list)

Scopo: Convertire una lista standard di nomi entità (ename) in un set di selezione (selection-set) AutoCAD.

Input: ent_list (Lista di ename).

Output: Set di selezione (Pickset).

Logica interna: Inizializza un set vuoto (ssadd) e vi inserisce ciclicamente ogni entità della lista previa verifica di esistenza nel database.

T

Trans_Pt_Glo_To_Loc

Firma: (Trans_Pt_Glo_To_Loc pt origin ang)

Scopo: Trasforma un singolo punto dal sistema globale WCS al sistema locale definito da origine e angolo, applicando traslazione e rotazione.

Input: pt: punto in WCS. origin: origine del sistema locale. ang: angolo di orientamento.

Output: Coordinate del punto nel sistema locale.

Logica interna: Sottrae origin al punto (traslazione) e poi applica la rotazione inversa di −ang con rotate-pt-origin o Math_Rot_Pt_2D_CS.

Trans_Pt_Loc_To_Glo

Firma: (Trans_Pt_Loc_To_Glo pt origin ang)

Scopo: Trasforma un singolo punto dal sistema locale al sistema globale WCS, invertendo la trasformazione di Trans_Pt_Glo_To_Loc.

Input: pt: punto nel sistema locale. origin: origine del sistema locale. ang: angolo.

Output: Coordinate del punto in WCS.

Logica interna: Applica la rotazione di +ang e poi aggiunge origin (traslazione inversa).

U

update_3pv_guides

Scopo: Comando principale interattivo per la generazione dinamica di coniche partendo da 2 Vertici e 1 Punto.

Input: Nessuno (Interazione utente via console e mouse).

Output: Nessuno (Side effect: generazione entità su DWG).

Logica interna: Inizializza i vettori base e innesca un ciclo while che sfrutta l'UI Prompt Helper e l'UI State Helper per aggiornare dinamicamente l'impalcatura e la curva, rispondendo agli hotkey. (defun c:CON2V1P (/ pts old_os old_pd old_ce V1 V2 base_vecs C dist_V a v_dir v_perp opt_g opt_m opt_e guide_ents last_run_ents temp_axis loop_main P last_P last_type res ui_state redraw_req) (mapcar 'set '(pts old_os old_pd old_ce) (Init_Conic_App "CONICA PER 2 VERTICI E PUNTO" 2 "\nSeleziona i due vertici (V1, V2): " 32)) (if pts (progn (mapcar 'set '(V1 V2) pts) (if (setq base_vecs (CON1P_Helper_Base_Vectors V1 V2)) (progn (mapcar 'set '(C dist_V a v_dir v_perp) base_vecs) (setvar "OSMODE" 0) (setvar "CMDECHO" 0) (setq opt_g "off" opt_m "Sostituisci" opt_e 1.0 guide_ents nil last_run_ents nil last_P nil last_type nil temp_axis (Draw_Line_2Pu_R V1 V2 *LaCo_Costruzioni*))

Util_Angle_Diff

Firma: (Util_Angle_Diff a1 a2)

Scopo: Calcola la differenza angolare tra due angoli in radianti, restituendo il valore nell'intervallo [−π, π].

Input: ang1, ang2: angoli in radianti.

Output: Differenza angolare (ang1 − ang2) normalizzata a [−π, π].

Logica interna: Sottrae gli angoli e applica una normalizzazione per riportare il risultato nell'intervallo [−π, π] tramite operatori mod e correzione di segno.

Util_Capture_After_Entity

Firma: (Util_Capture_After_Entity ent_start lay_target)

Scopo: Raccoglie tutte le entità AutoCAD create dopo l'entità di riferimento data, includendo i blocchi anonimi e le sottoentità visibili.

Input: anchor_ent: entità di riferimento (ename). Chi chiama deve aver salvato entlast prima.

Output: Lista di ename di tutte le nuove entità create dopo anchor_ent nel database.

Logica interna: Itera con entnext, raccogliendo ogni entità successiva fino alla fine del database. A differenza di Util_Collect_Ents, gestisce anche le entità composte.

Util_Collect_Ents

Firma: (Util_Collect_Ents last_ent)

Scopo: Raccoglie in una lista tutte le entità AutoCAD create dopo un'entità di riferimento data.

Input: last_ent: nome entità (ename) usata come punto di partenza; la raccolta inizia dalla successiva.

Output: Lista ordinata di ename di tutte le entità create dopo last_ent.

Logica interna: Itera con `entnext` partendo da last_ent fino alla fine del database disegno, accumulando i nomi entità in una lista che viene invertita con `reverse` per preservare l'ordine cronologico.

Util_Conv_List_To_VLA

Firma: (Util_Conv_List_To_VLA ents)

Scopo: Converte una lista LISP di punti in un array VLA (Visual LISP ActiveX) di varianti, formato richiesto da alcune API di AutoCAD quali AddPolyline.

Input: lst: lista di punti (liste di coordinate).

Output: Array VLA (safearray di varianti) compatibile con le funzioni ActiveX di AutoCAD.

Logica interna: Alloca un safearray con vlax-make-safearray, poi usa vlax-safearray-fill per popolare ogni elemento con le coordinate del punto corrispondente.

Util_Del_Ents_At_Pt

Firma: (Util_Del_Ents_At_Pt pt layer)

Scopo: Elimina tutte le entità AutoCAD che si trovano entro una piccola area di tolleranza attorno a un punto dato, filtrate per layer.

Input: pt: coordinate del punto di riferimento. layer: nome del layer su cui cercare le entità.

Output: Effetto collaterale: entità rimosse dal disegno. Restituisce nil.

Logica interna: Costruisce un box di ricerca di 2e-4 unità intorno a pt, usa `ssget "C"` con filtro layer per raccogliere le entità nella zona, poi le elimina una per una con `entdel`.

Util_Del_Ents_List

Firma: (Util_Del_Ents_List lst_ents)

Scopo: Elimina dal database AutoCAD tutte le entità contenute in una lista, verificando preventivamente la loro esistenza.

Input: lst_ents: lista di nomi entità (ename) da eliminare.

Output: Effetto collaterale: entità eliminate dal disegno. Restituisce nil.

Logica interna: Itera sulla lista con `foreach`; per ogni entità verifica con `entget` che esista ancora nel database prima di chiamare `entdel`, proteggendo da errori su entità già cancellate.

Util_Denorm_List_Pts

Firma: (Util_Denorm_List_Pts pts_list G_bary max_dist)

Scopo: Denormalizza una lista intera di punti dal sistema normalizzato al sistema reale.

Input: pts_list: lista di punti normalizzati. G_bary: baricentro. max_dist: fattore di scala.

Output: Lista di punti in coordinate reali.

Logica interna: Applica mapcar con Util_Denorm_Pt a ogni elemento della lista.

Util_Denorm_Pt

Firma: (Util_Denorm_Pt pt_norm G_bary max_dist)

Scopo: Denormalizza un singolo punto dal sistema normalizzato (baricentro=origine, max_dist=unità) al sistema reale delle coordinate di disegno.

Input: pt_norm: punto normalizzato. G_bary: baricentro reale. max_dist: distanza massima di normalizzazione.

Output: Coordinate del punto reale corrispondente: G_bary + pt_norm · max_dist.

Logica interna: Scala il punto normalizzato per max_dist (Vet_Sca) e trasla al baricentro (Vet_Sum).

Util_Distanza_Media_lisPu_Da_Cen

Firma: (Util_Distanza_Media_lisPu_Da_Cen pts center)

Scopo: Calcola la distanza media di una lista di punti da un punto centro dato, usata come stima del raggio medio della conica.

Input: lisPu: lista di punti. Cen: punto centro di riferimento.

Output: Valore reale della distanza media.

Logica interna: Calcola distance(Cen, p) per ogni punto p con mapcar, poi chiama Util_Media_lisNum sulla lista di distanze risultante.

Util_Extract_Val

Firma: (Util_Extract_Val func item)

Scopo: Estrae il valore di campo specificato da un record della lista dati strutturata del motore di scansione, usato per accedere ai campi dei risultati intermedi.

Input: record: lista strutturata (associazione chiave-valore). key: simbolo chiave da estrarre.

Output: Valore associato alla chiave; nil se la chiave non esiste nel record.

Logica interna: Usa assoc per trovare la coppia chiave-valore nel record e car/cdr per estrarre il valore.

Util_Filter_And_Count_By_Nth

Firma: (Util_Filter_And_Count_By_Nth lst n target)

Scopo: Filtra e conta contemporaneamente i record di una lista in base al valore dell'n-esimo elemento, restituendo sia la lista filtrata sia il conteggio.

Input: lst: lista di record. n: indice. val: valore ricercato.

Output: Lista di due elementi: (lista_filtrata conteggio).

Logica interna: Chiama Util_Filter_By_Nth per la lista e (length ...) per il conteggio, restituendo entrambi in una lista.

Util_Filter_By_Nth

Firma: (Util_Filter_By_Nth lst n target)

Scopo: Filtra una lista di record mantenendo solo quelli il cui n-esimo elemento corrisponde a un valore specificato.

Input: lst: lista di record (liste). n: indice dell'elemento da verificare. val: valore cercato.

Output: Lista filtrata dei record dove (nth n record) = val.

Logica interna: Applica foreach o vl-remove-if-not con una lambda che testa (equal (nth n item) val).

Util_Filter_Outliers_By_Centroid

Firma: (Util_Filter_Outliers_By_Centroid data_list center_extractor_func tolerance_factor)

Scopo: Filtra i punti di una lista eliminando gli outlier (valori anomali) che superano una distanza soglia dal baricentro, proteggendo la qualità del fit.

Input: lisPu: lista di punti. sigma_thresh: moltiplicatore di deviazione standard per la soglia.

Output: Lista di punti filtrati (senza outlier).

Logica interna: Calcola il baricentro e la distanza media dal baricentro. Elimina i punti la cui distanza supera media + sigma_thresh × deviazione_standard.

Util_Get_All_Pairs

Firma: (Util_Get_All_Pairs lst)

Scopo: Genera tutte le coppie non ordinate possibili (combinazioni C(n,2)) da una lista di elementi.

Input: lst: lista di n elementi.

Output: Lista di tutte le coppie (n*(n-1)/2 elementi), ciascuna come lista a due elementi.

Logica interna: Doppio ciclo `while` che scorre la lista diminuendo progressivamente il sottolista interno; accumula ogni coppia in una lista risultato.

Util_Get_Max_Dist_From_Point

Firma: (Util_Get_Max_Dist_From_Point pts ref_pt)

Scopo: Determinare rispettivamente la distanza euclidea massima e minima intercorrente tra un punto di riferimento specifico e una nuvola o lista di punti spaziali assegnata.

Input: pts (Lista di coordinate WCS dei punti da analizzare), ref_pt (Coordinate WCS del punto di riferimento da cui misurare le distanze).

Output: Un numero reale (float) corrispondente al valore scalare della distanza euclidea estrema (massima o minima) individuata.

Logica interna: Mappa una funzione anonima (lambda) sull'intera lista di target per generare un array monodimensionale contenente tutte le distanze euclidee calcolate. Utilizza quindi l'operatore funzionale 'apply' combinato con i predicati primitivi 'max' o 'min' per collassare istantaneamente l'array numerico ed estrarre il valore richiesto con la massima efficienza.

Util_Get_Min_Dist_From_Point

Firma: (Util_Get_Min_Dist_From_Point pts ref_pt)

Scopo: Determinare rispettivamente la distanza euclidea massima e minima intercorrente tra un punto di riferimento specifico e una nuvola o lista di punti spaziali assegnata.

Input: pts (Lista di coordinate WCS dei punti da analizzare), ref_pt (Coordinate WCS del punto di riferimento da cui misurare le distanze).

Output: Un numero reale (float) corrispondente al valore scalare della distanza euclidea estrema (massima o minima) individuata.

Logica interna: Mappa una funzione anonima (lambda) sull'intera lista di target per generare un array monodimensionale contenente tutte le distanze euclidee calcolate. Utilizza quindi l'operatore funzionale 'apply' combinato con i predicati primitivi 'max' o 'min' per collassare istantaneamente l'array numerico ed estrarre il valore richiesto con la massima efficienza. (defun Util_Get_Max_Dist_From_Point (pts ref_pt) (apply 'max (mapcar '(lambda (p) (distance p ref_pt)) pts)))

Util_Get_Pts_BBox_Size

Firma: (Util_Get_Pts_BBox_Size pts)

Scopo: Calcolare le dimensioni del bounding box (ingombro planare) di una nuvola di punti, determinando le escursioni massime (Delta) lungo gli assi X e Y.

Input: pts (Lista di coordinate o punti 2D/3D rappresentanti la geometria spaziale da analizzare).

Output: Lista di due numeri reali formattata come `(dx dy)`, che rappresenta rispettivamente la larghezza (Delta X) e l'altezza (Delta Y) dell'ingombro totale.

Logica interna: Sfrutta l'helper `Math_Extract_Coord_Pow` (con potenza 1) per isolare massivamente in due array separati tutte le coordinate X (indice 0) e Y (indice 1) della nuvola. Invia poi questi due vettori alla funzione `Math_Delta_List` per calcolarne l'escursione termica (Max - Min), assemblando infine i risultati in una coppia.

Util_Int_To_Str

Firma: (Util_Int_To_Str n)

Scopo: Fornire un set unificato di utilità (wrapper) per la conversione sicura di vari tipi di dato LISP (Punti 3D, Simboli, Numeri Reali, Interi) in formato stringa testuale, standardizzando il logging e il feedback a video.

Input: L'argomento specifico da convertire: Pu1 (lista di coordinate), sym (simbolo), num (numero reale), n (numero intero).

Output: La stringa formattata rappresentante il dato in ingresso (con precisione decimale a 4 cifre per reali e

coordinate), oppure la stringa letterale "NIL" se l'input risulta nullo o di tipo non valido.

Logica interna: Le funzioni eseguono un type-checking preventivo (listp, numberp o un semplice controllo di nullità). A seconda del tipo, i punti vengono destrutturati e concatenati con parentesi tramite 'rtos', i reali processati direttamente con 'rtos', gli interi con 'itoa' e i simboli tradotti con 'vl-symbol-name'. Il meccanismo di fallback globale gestisce silenziosamente le eccezioni. (defun Util_Pun_To_Str (Pu1) (if (and Pu1 (listp Pu1)) (strcat "(" (rtos (car Pu1) 2 4) " " (rtos (cadr Pu1) 2 4) " " (rtos (caddr Pu1) 2 4) ")") "NIL")) (defun Util_Sym_To_Str (sym) (if sym (vl-symbol-name sym) "NIL")) (defun Util_Num_To_Str (num) (if (numberp num) (rtos num 2 4) "NIL"))

Util_List_Rotations

Firma: (Util_List_Rotations lst)

Scopo: Genera tutte le rotazioni ciclice di una lista: per una lista di n elementi restituisce n liste, ciascuna ottenuta spostando il primo elemento in fondo.

Input: lst: lista di n elementi.

Output: Lista di n liste, ciascuna una rotazione ciclica della lista originale.

Logica interna: Loop repeat con length(lst) iterazioni: ad ogni passo aggiunge la lista corrente alla lista risultato, poi ruota la lista spostando (car lst) in fondo con (append (cdr lst) (list (car lst))).

Util_Make_Anonymous_Group

Firma: (Util_Make_Anonymous_Group lst_ents descr)

Scopo: Raggruppare dinamicamente una lista di entità vettoriali in un Gruppo Anonimo di AutoCAD per facilitarne la selezione e la manipolazione congiunta a video.

Input: lst_ents (Lista di handle ENAME delle entità da raggruppare), descr (Stringa descrittiva associata al gruppo).

Output: Nessuno (Restituisce un'uscita silenziosa con `princ`, agisce per side-effect sul database DWG).

Logica interna: Valida la lista in ingresso (richiedendo almeno 2 entità). Inizializza un Selection Set vuoto e lo popola scorrendo gli handle validi verificati tramite `entget`. Disabilita temporaneamente l'eco dei comandi CAD (`CMDECHO`) per non inquinare la riga di comando, invoca il comando nativo `_.-GROUP` con parametro `*` per forzare la generazione di un nome anonimo, ed infine ripristina lo stato originale della variabile di sistema.

Util_Mean_Angle

Firma: (Util_Mean_Angle angle_list)

Scopo: Calcola la media angolare di una lista di angoli in radianti, gestendo correttamente il problema della discontinuità a ±π con il metodo vettoriale.

Input: angles: lista di angoli in radianti.

Output: Angolo medio in radianti, normalizzato a [−π, π].

Logica interna: Converte ogni angolo in un versore 2D (cos, sin), ne calcola la media vettoriale (somma divisa per n) e riconverte a angolo con atan2.

Util_Media_2Num

Firma: (Util_Media_2Num num1 num2)

Scopo: Calcola la media aritmetica di due valori numerici.

Input: num1, num2: due valori numerici reali.

Output: Valore reale della media (num1+num2)/2.

Logica interna: Costruisce la lista (num1 num2) e delega a Util_Media_lisNum.

Util_Media_lisNum

Firma: (Util_Media_lisNum lst)

Scopo: Calcola la media aritmetica di una lista di valori numerici.

Input: lst: lista non vuota di valori numerici reali.

Output: Valore reale della media aritmetica; nil se la lista è vuota.

Logica interna: Somma tutti gli elementi con (apply'+ lst) e divide per (float (length lst)) per ottenere la media in virgola mobile.

Util_Normalize_Pts_Set

Firma: (Util_Normalize_Pts_Set pts)

Scopo: Normalizza un set di punti calcolando il baricentro e la distanza massima, restituendo i dati necessari per la normalizzazione e denormalizzazione.

Input: pts: lista di punti da normalizzare.

Output: Lista (baricentro max_dist punti_normalizzati) con i dati del sistema normalizzato.

Logica interna: Calcola il baricentro come media delle coordinate X e Y. Calcola la distanza massima dal baricentro. Normalizza ogni punto sottraendo il baricentro e dividendo per max_dist.

Util_Normalizza_lisPu_Da_Cen

Firma: (Util_Normalizza_lisPu_Da_Cen pts center)

Scopo: Normalizza una lista di punti rispetto al loro baricentro e alla distanza massima, portando i punti in un sistema numerico stabile centrato sull'origine con raggio ≤ 1.

Input: lisPu: lista di punti. Cen(opzionale): centro di normalizzazione (se nil usa il baricentro).

Output: Lista di punti normalizzati, con coordinate comprese tipicamente tra -1 e 1.

Logica interna: Calcola il baricentro con Util_Media_lisNum sulle X e Y, la distanza massima con Util_Distanza_Media_lisPu_Da_Cen, poi trasla e scala ogni punto.

Util_Num_To_Str

Firma: (Util_Num_To_Str num)

Scopo: Fornire un set unificato di utilità (wrapper) per la conversione sicura di vari tipi di dato LISP (Punti 3D, Simboli, Numeri Reali, Interi) in formato stringa testuale, standardizzando il logging e il feedback a video.

Input: L'argomento specifico da convertire: Pu1 (lista di coordinate), sym (simbolo), num (numero reale), n (numero intero).

Output: La stringa formattata rappresentante il dato in ingresso (con precisione decimale a 4 cifre per reali e coordinate), oppure la stringa letterale "NIL" se l'input risulta nullo o di tipo non valido.

Logica interna: Le funzioni eseguono un type-checking preventivo (listp, numberp o un semplice controllo di nullità). A seconda del tipo, i punti vengono destrutturati e concatenati con parentesi tramite 'rtos', i reali processati direttamente con 'rtos', gli interi con 'itoa' e i simboli tradotti con 'vl-symbol-name'. Il meccanismo di fallback globale gestisce silenziosamente le eccezioni. (defun Util_Pun_To_Str (Pu1) (if (and Pu1 (listp Pu1)) (strcat "(" (rtos (car Pu1) 2 4) " " (rtos (cadr Pu1) 2 4) " " (rtos (caddr Pu1) 2 4) ")") "NIL")) (defun Util_Sym_To_Str (sym) (if sym (vl-symbol-name sym) "NIL"))

Util_Orthogonality_Score

Firma: (Util_Orthogonality_Score ang)

Scopo: Calcola un punteggio di ortogonalità per un insieme di direzioni angolari, misurando quanto ogni coppia di direzioni si discosta dalla perpendicolarità reciproca.

Input: angles: lista di angoli in radianti.

Output: Valore numerico reale: 0.0 = perfettamente ortogonale; valori alti = scarsa ortogonalità.

Logica interna: Per ogni coppia di angoli calcola la differenza con Util_Angle_Diff e misura lo scarto da π/2; somma gli scarti al quadrato come punteggio complessivo.

Util_Print_And_Nil

Firma: (Util_Print_And_Nil txt)

Scopo: Stampare un messaggio testuale sulla riga di comando di AutoCAD e restituire simultaneamente il valore nullo (nil), utile per gestire in modo compatto le uscite di errore o i fallback all'interno dei costrutti condizionali.

Input: txt (Stringa testuale contenente il messaggio, l'avviso o l'errore da mostrare all'utente).

Output: Restituisce sempre e rigorosamente il valore logico nil.

Logica interna: Sfrutta la funzione primitiva 'princ' per stampare il testo in ingresso a schermo.

Successivamente, la routine à strutturata per valutare esplicitamente 'nil' come ultima espressione. Questo forza il ritorno nullo al chiamante, permettendo di unire il feedback visivo e l'interruzione logica in una singola riga di codice molto pulita.

Util_Process_Branches_Data

Firma: (Util_Process_Branches_Data data_list dist_limit)

Scopo: Processa e consolida i dati dei rami di un luogo geometrico ramificato, eliminando i rami spuri e riassemblando quelli validi in ordine topologico.

Input: branches: lista di rami (ciascuno una lista di punti o entità). thresh: soglia di distanza per la giunzione dei rami.

Output: Lista di rami processati e ordinati pronta per il disegno finale.

Logica interna: Valuta la distanza tra i punti estremi dei rami adiacenti; se inferiore a thresh, unisce i rami in un unico percorso continuo. Scarta i rami troppo corti (< min_len) come artefatti della scansione.

Util_Process_Branches_Data_SmartStitch

Firma: (Util_Process_Branches_Data_SmartStitch raw_data dist_limit)

Scopo: Post-processare i dati grezzi dei rami parabolici, prima frammentandoli in base ai salti focali e poi tentando una saldatura intelligente (Smart Stitch) tra l'inizio del primo ramo e la fine dell'ultimo per chiudere i loop.

Input: raw_data (Lista grezza di pacchetti dati, dove il 2à elemento à il Fuoco), dist_limit (Tolleranza di taglio).

Output: Lista strutturata di rami discreti (matrice di pacchetti dati), eventualmente ricucita alle estremità.

Logica interna: Invoca lo scanner per dividere i rami. Se ne ottiene almeno due, valuta la distanza 3D tra il fuoco finale dell'ultimo ramo e il fuoco iniziale del primo. Se questo "gap" di chiusura à inferiore al triplo (factor 3.0) del passo locale adiacente, fonde l'ultimo e il primo ramo garantendo la continuità della curva senza interruzioni.

Util_Process_Branches_Universal

Firma: (Util_Process_Branches_Universal pt_list dist_limit pts_tri flag_smart_stitch flag_no_cross)

Scopo: Pipeline di processing completo dei rami del luogo geometrico: prima taglia i rami ai salti (Util_Scan_And_Cut_Branches), poi cucisce i tronconi vicini (Util_Stitch_Branches_HeadTail), infine scarta i rami troppo corti come artefatti.

Input: branches: lista di rami grezzi. cut_thresh, stitch_thresh: soglie rispettive.

Output: Lista di rami processati e pronti per il disegno.

Logica interna: Applica in sequenza: taglio → cucitura → filtraggio per lunghezza minima.

Util_Pun_To_Str

Firma: (Util_Pun_To_Str Pu1)

Scopo: Fornire un set unificato di utilità (wrapper) per la conversione sicura di vari tipi di dato LISP (Punti 3D, Simboli, Numeri Reali, Interi) in formato stringa testuale, standardizzando il logging e il feedback a video.

Input: L'argomento specifico da convertire: Pu1 (lista di coordinate), sym (simbolo), num (numero reale), n (numero intero).

Output: La stringa formattata rappresentante il dato in ingresso (con precisione decimale a 4 cifre per reali e coordinate), oppure la stringa letterale "NIL" se l'input risulta nullo o di tipo non valido.

Logica interna: Le funzioni eseguono un type-checking preventivo (listp, numberp o un semplice controllo di nullità). A seconda del tipo, i punti vengono destrutturati e concatenati con parentesi tramite 'rtos', i reali processati direttamente con 'rtos', gli interi con 'itoa' e i simboli tradotti con 'vl-symbol-name'. Il meccanismo di fallback globale gestisce silenziosamente le eccezioni.

Util_Reorder_List_By_Indices

Firma: (Util_Reorder_List_By_Indices lst idx_list)

Scopo: Riordina una lista secondo un vettore di indici dato, restituendo una nuova lista in cui l'i-esimo elemento è l'elemento della lista originale alla posizione indices[i].

Input: lst: lista di elementi da riordinare. indices: lista di indici interi (0-based).

Output: Lista riordinata della stessa lunghezza.

Logica interna: Applica mapcar con (lambda (i) (nth i lst)) sulla lista indices per estrarre gli elementi nell'ordine specificato.

Util_Reset_Vars

Firma: (Util_Reset_Vars sym_list)

Scopo: Reimposta un insieme di variabili a nil in un'unica chiamata, usato nei cicli di init/reset dei loop interattivi.

Input: var_names: lista di simboli di variabili da azzerare.

Output: Effetto collaterale: le variabili listate vengono impostate a nil.

Logica interna: Usa foreach e set su ogni simbolo della lista per assegnare nil a ciascuna variabile.

Util_Rot_2D_Dir_CS

Firma: (Util_Rot_2D_Dir_CS Pu1 co si)

Scopo: Ruota un punto 2D nella direzione diretta (senso positivo) dato il coseno e seno dell'angolo precompilati.

Input: Pu1: punto 2D. co: coseno dell'angolo. si: seno dell'angolo.

Output: Lista (x' y') del punto ruotato in senso diretto.

Logica interna: Calcola x'= x·co + y·si, y'= y·co − x·si tramite Math_Sum_Prod e Math_Diff_Prod.

Util_Rot_2D_Inv_CS

Firma: (Util_Rot_2D_Inv_CS Pu1 co si)

Scopo: Ruota un punto 2D nella direzione inversa (senso negativo) dato il coseno e seno dell'angolo precompilati, ovvero applica la rotazione di −ang.

Input: Pu1: punto 2D. co: coseno dell'angolo. si: seno dell'angolo.

Output: Lista (x' y') del punto ruotato in senso inverso.

Logica interna: Calcola x'= x·co − y·si, y'= x·si + y·co (rotazione inversa rispetto a Util_Rot_2D_Dir_CS).

Util_Rot_List_2D_CS

Firma: (Util_Rot_List_2D_CS lisPu co si)

Scopo: Ruota in senso diretto una lista intera di punti 2D usando coseno e seno precompilati per efficienza.

Input: lisPu: lista di punti 2D. co: coseno. si: seno.

Output: Lista di punti 2D ruotati.

Logica interna: Applica mapcar con Util_Rot_2D_Dir_CS a ogni punto della lista.

Util_Rot_List_2D_Inv_CS

Firma: (Util_Rot_List_2D_Inv_CS lisPu co si)

Scopo: Ruota in senso inverso una lista intera di punti 2D usando coseno e seno precompilati per efficienza.

Input: lisPu: lista di punti 2D. co: coseno. si: seno.

Output: Lista di punti 2D ruotati in senso inverso.

Logica interna: Applica mapcar con Util_Rot_2D_Inv_CS a ogni punto della lista.

Util_SafeLayer

Firma: (Util_SafeLayer lay)

Scopo: Creare silenziosamente un layer o forzarne l'aggiornamento cromatico, operando nel database senza alterare il layer corrente.

Input: lay (Stringa col nome del layer), col (Intero 1-255 per il colore AutoCAD).

Output: Restituisce il nome del layer processato (Stringa).

Logica interna: Estrae il safe_col. Cerca il layer via tblobjname. Se esiste, ne legge il codice 62. Esegue entmod solo se il colore attuale differisce da quello richiesto, risparmiando scritture inutili. Se non esiste, lo crea da zero tramite entmake.

Util_Scala_lisPu_Da_Cen

Firma: (Util_Scala_lisPu_Da_Cen pts center scale_factor)

Scopo: Scala una lista di punti rispetto a un centro dato, moltiplicando per un fattore le distanze al centro (ingrandimento/riduzione omotetica).

Input: lisPu: lista di punti. Cen: centro dell'omotetia. scala: fattore di scala.

Output: Lista di punti scalati rispetto a Cen.

Logica interna: Per ogni punto, calcola il vettore v = p − Cen, lo scala con Vet_Sca(v, scala) e riposiziona sommando Cen.

Util_Scale_Pts_From_Centroid

Firma: (Util_Scale_Pts_From_Centroid pts factor)

Scopo: Scala una lista di punti rispetto al loro baricentro calcolato automaticamente, usato per estendere le fasce di ricerca del luogo geometrico.

Input: pts: lista di punti. scala: fattore di scala omotetico.

Output: Lista di punti scalati rispetto al baricentro calcolato.

Logica interna: Calcola il baricentro come media delle coordinate, poi delega a Util_Scala_lisPu_Da_Cen.

Util_Scan_And_Cut_Branches

Firma: (Util_Scan_And_Cut_Branches pt_list dist_limit pts_tri flag_no_cross)

Scopo: Scansiona i rami del luogo geometrico e li divide nei punti in cui la distanza tra punti consecutivi supera una soglia (salto non fisico).

Input: branches: lista di rami. thresh: soglia di distanza per il taglio.

Output: Lista di rami tagliati, senza salti artificiali.

Logica interna: Per ogni ramo, itera sui punti consecutivi e calcola la distanza; quando supera thresh taglia il ramo in quel punto, generando sotto-rami separati.

Util_Scan_And_Cut_Branches_Data_By_Focus

Firma: (Util_Scan_And_Cut_Branches_Data_By_Focus pt_list dist_limit)

Scopo: Analizzare sequenzialmente una lista di dati e frammentarla in rami discreti ogni volta che la distanza spaziale tra due fuochi consecutivi supera la soglia consentita, identificando così i salti topologici o gli asintoti.

Input: pt_list (Lista di pacchetti, dove il 'cadr' contiene le coordinate del Fuoco), dist_limit (Soglia di taglio).

Output: Lista di rami (ogni ramo à una sottolista contenente i pacchetti dati geometricamente contigui).

Logica interna: Itera su ogni pacchetto estraendo il Fuoco. Misura la distanza dal fuoco del punto precedente. Se tale distanza à maggiore del dist_limit, archivia il ramo accumulato finora e inizia a registrare un nuovo ramo. Al termine del ciclo, applica un 'reverse' alle liste per ripristinare il corretto ordine cronologico dei punti.

Util_Seg_Crosses_Triangle

Firma: (Util_Seg_Crosses_Triangle Pu1 Pu2 tri_pts)

Scopo: Verifica se un segmento interseca uno dei lati del triangolo dato, usato per rilevare attraversamenti del bordo durante la scansione del luogo.

Input: p1, p2: estremi del segmento. tri_pts: lista di 3 punti del triangolo.

Output: T se il segmento interseca almeno un lato del triangolo; nil altrimenti.

Logica interna: Verifica l'intersezione del segmento p1-p2 con ciascuno dei tre lati del triangolo tramite inters_ver o una funzione equivalente di intersezione segmento-segmento.

Util_Select_Point_From_List

Firma: (Util_Select_Point_From_List msg lst_pts)

Scopo: Prompt interattivo che chiede all'utente di cliccare su un punto del disegno e verifica se il punto selezionato appartiene a una lista di candidati.

Input: msg: stringa del prompt da mostrare all'utente. lst_pts: lista di punti candidati da cui verificare la corrispondenza.

Output: Il punto della lista più vicino al punto selezionato dall'utente; nil se non trovato entro la tolleranza.

Logica interna: Loop `while` con `entsel` che attende la selezione di un'entità. Confronta la posizione selezionata con i punti della lista tramite distanza; se la distanza è

inferiore alla soglia restituisce il candidato corrispondente.

Util_Select_Winning_Hypothesis

Firma: (Util_Select_Winning_Hypothesis L1 L2 L3)

Scopo: Chiede all'utente di indicare quale tra le ipotesi geometriche mostrate (punti/vettori colorati) è quella corretta, restituendo l'ipotesi selezionata.

Input: L1, L2, L3: liste di entità corrispondenti a tre ipotesi geometriche disegnate.

Output: Intero 1, 2 o 3 identificante l'ipotesi selezionata dall'utente.

Logica interna: Loop `while` con `entsel`. Per ogni entità selezionata verifica in quale delle tre liste L1/L2/L3 è presente; al match esce dal loop restituendo l'indice corrispondente.

Util_Sort_By_Angle_Deviation

Firma: (Util_Sort_By_Angle_Deviation data_list angle_extractor target_angle)

Scopo: Ordina una lista di soluzioni in base alla deviazione angolare rispetto a una direzione di riferimento, per trovare la conica con asse più allineato.

Input: solutions: lista di record con campo angolo. ref_ang: angolo di riferimento.

Output: Lista ordinata per deviazione angolare crescente rispetto a ref_ang.

Logica interna: Calcola Util_Angle_Diff(sol_ang, ref_ang) per ogni soluzione e ordina per valore assoluto crescente.

Util_Sort_By_Orthogonality

Firma: (Util_Sort_By_Orthogonality data_list angle_extractor)

Scopo: Ordina una lista di ipotesi geometric (angoli o soluzioni) in base al loro punteggio di ortogonalità, mettendo prima le soluzioni più ortogonali.

Input: hypotheses: lista di record con campo di angolo. pts: punti di riferimento.

Output: Lista ordinata dalla soluzione più ortogonale alla meno ortogonale.

Logica interna: Calcola Util_Orthogonality_Score per ogni ipotesi, poi ordina tramite un algoritmo di ordinamento (bubble sort o insertion sort) per score crescente.

Util_Stitch_Branches_HeadTail

Firma: (Util_Stitch_Branches_HeadTail branches)

Scopo: Unisce rami contigui del luogo geometrico collegando la coda di un ramo alla testa del successivo, se la distanza tra gli estremi è inferiore alla soglia.

Input: branches: lista di rami. thresh: soglia di distanza per la giunzione.

Output: Lista di rami uniti, con meno interruzioni.

Logica interna: Per ogni coppia di rami, confronta la distanza tra (last ramo_i) e (car ramo_j); se inferiore a thresh concatena i due rami con append.

Util_Sym_To_Str

Firma: (Util_Sym_To_Str sym)

Scopo: Fornire un set unificato di utilità (wrapper) per la conversione sicura di vari tipi di dato LISP (Punti 3D, Simboli, Numeri Reali, Interi) in formato stringa testuale, standardizzando il logging e il feedback a video.

Input: L'argomento specifico da convertire: Pu1 (lista di coordinate), sym (simbolo), num (numero reale), n (numero intero).

Output: La stringa formattata rappresentante il dato in ingresso (con precisione decimale a 4 cifre per reali e coordinate), oppure la stringa letterale "NIL" se l'input risulta nullo o di tipo non valido.

Logica interna: Le funzioni eseguono un type-checking preventivo (listp, numberp o un semplice controllo di nullità). A seconda del tipo, i punti vengono destrutturati e concatenati con parentesi tramite 'rtos', i reali processati direttamente con 'rtos', gli interi con 'itoa' e i simboli tradotti con 'vl-symbol-name'. Il meccanismo di fallback globale gestisce silenziosamente le eccezioni. (defun Util_Pun_To_Str (Pu1) (if (and Pu1 (listp Pu1)) (strcat "(" (rtos (car Pu1) 2 4) " " (rtos (cadr Pu1) 2 4) " " (rtos (caddr Pu1) 2 4) ")") "NIL"))

Util_Translate_Pts_Global

Firma: (Util_Translate_Pts_Global pts_list base_pt)

Scopo: Trasla una lista di punti nel sistema globale aggiungendo il punto base, invertendo la traslazione locale applicata da Util_Translate_Pts_Relative.

Input: pts_list: lista di punti nel sistema locale. base_pt: punto origine del sistema locale.

Output: Lista di punti nel sistema globale: ogni punto p diventa p + base_pt.

Logica interna: Applica mapcar con Vet_Sum per sommare base_pt a ogni punto della lista.

Util_Translate_Pts_Relative

Firma: (Util_Translate_Pts_Relative pts_list base_pt)

Scopo: Trasla una lista di punti sottraendo il punto base da ciascuno, portandoli nel sistema di riferimento locale centrato sul punto base.

Input: pts_list: lista di punti 2D o 3D. base_pt: punto origine del sistema locale.

Output: Lista di punti traslati: ogni punto p diventa p − base_pt.

Logica interna: Applica mapcar con Vet_Sub per sottrarre base_pt da ogni punto della lista.

Util_Trans_List_Glo_To_Loc

Firma: (Util_Trans_List_Glo_To_Loc pts origin ang)

Scopo: Trasforma una lista di punti dal sistema globale (WCS) al sistema locale definito da un'origine e un angolo di orientamento, applicando traslazione seguita da rotazione.

Input: pts: lista di punti WCS. origin: punto origine del sistema locale. ang: angolo di orientamento.

Output: Lista di punti nel sistema locale (traslato e ruotato).

Logica interna: Precompila co/si da ang con Geom_Co_Si_Ang. Per ogni punto: sottrae origin con Vet_Sub (traslazione) poi ruota con Math_Rot_Pt_2D_CS.

Util_Trans_List_Loc_To_Glo

Firma: (Util_Trans_List_Loc_To_Glo pts origin ang)

Scopo: Trasforma una lista di punti dal sistema locale al sistema globale (WCS), invertendo la trasformazione di Util_Trans_List_Glo_To_Loc.

Input: pts: lista di punti nel sistema locale. origin: origine del sistema. ang: angolo.

Output: Lista di punti in coordinate WCS.

Logica interna: Precompila co/si da −ang (angolo negativo = rotazione inversa). Per ogni punto: ruota con Math_Rot_Pt_2D_CS poi somma origin con Vet_Sum.

Util_Trans_Pts_List

Firma: (Util_Trans_Pts_List pts from_CS to_CS)

Scopo: Trasla una lista di punti applicando un vettore di traslazione comune, alias semplificato di Util_Translate_Pts_Global per contesti specifici.

Input: pts: lista di punti. offset: vettore di traslazione.

Output: Lista di punti traslati.

Logica interna: Applica mapcar con Vet_Sum a ogni punto della lista.

Util_Update_Guides_Vis

Firma: (Util_Update_Guides_Vis stato_opt lista_ent)

Scopo: Aggiornare lo stato di visibilità (acceso/spento) di un intero gruppo di entità grafiche nel disegno.

Input: stato_opt (Stringa che definisce lo stato desiderato, es. "ON" o altro per "OFF"), lista_ent (Lista di ename delle entità di cui alterare la visibilità).

Output: Nessun output esplicito (ritorna la funzione silenziosa princ). Agisce direttamente in modifica sul database grafico di AutoCAD.

Logica interna: Traduce la stringa testuale di stato nel codice numerico DXF corrispondente (0 per visibile, 1 per invisibile). Scorre iterativamente la lista delle entità tramite foreach, verificandone prima l'esistenza nel database con entget per prevenire crash su oggetti precedentemente eliminati. Infine, inietta o sovrascrive il codice di gruppo DXF 60 tramite entmod, aggiornando istantaneamente la visualizzazione a schermo.

V

Vet_Cro

Firma: (Vet_Cro v1 v2)

Scopo: Calcolare il prodotto vettoriale (cross product) tra due vettori nello spazio tridimensionale, restituendo un terzo vettore strettamente ortogonale al piano da essi formato.

Input: v1, v2 (Liste di coordinate numeriche reali rappresentanti le componenti 3D dei due vettori originari).

Output: Lista di tre numeri reali corrispondente alle coordinate del nuovo vettore ortogonale.

Logica interna: Unisce temporaneamente le due liste tramite `append` e utilizza `mapcar 'set` per assegnare simultaneamente le 6 componenti spaziali. Applica infine la regola dei determinanti incrociati, delegando il calcolo all'helper `Math_Diff_Prod` per mantenere la sintassi compatta e sicura.

Vet_Dot

Firma: (Vet_Dot v1 v2)

Scopo: Calcolare il prodotto scalare (dot product) tra due vettori, restituendo un singolo valore numerico.

Input: v1, v2 (Liste di numeri reali che rappresentano le componenti spaziali dei due vettori).

Output: Valore numerico reale corrispondente al prodotto scalare.

Logica interna: Esegue la somma dei prodotti delle componenti omologhe. Sfrutta l'iteratore `mapcar` con l'operatore `*` per moltiplicare parallelamente le liste vettoriali, e la funzione `apply` associata all'operatore `+` per sommare istantaneamente i risultati in un'unica e velocissima istruzione nativa.

Vet_Len

Firma: (Vet_Len v)

Scopo: Calcolare la magnitudine (lunghezza o modulo) assoluta di un vettore nello spazio geometrico.

Input: v (Lista di coordinate numeriche reali rappresentanti le componenti spaziali del vettore 2D o 3D).

Output: Valore numerico reale (float) indicante la lunghezza euclidea del vettore.

Logica interna: Ottimizza il calcolo computazionale sfruttando la funzione nativa `distance` di AutoLISP per misurare la distanza del vettore (trattato matematicamente come un punto) dall'origine assoluta delle coordinate `(0.0 0.0 0.0)`. Questo approccio bypassa l'implementazione esplicita del Teorema di Pitagora, garantendo tempi di esecuzione minimi.

Vet_Mid

Firma: (Vet_Mid p1 p2)

Scopo: Calcolare le coordinate spaziali esatte del punto medio intermedio tra due coordinate (o vettori) fornite in ingresso.

Input: p1, p2 (Liste di numeri reali che rappresentano le coordinate spaziali 2D o 3D dei due estremi del segmento).

Output: Lista di coordinate reali corrispondente al vettore (o punto) medio calcolato.

Logica interna: Applica l'algebra vettoriale assemblando in cascata gli helper nativi. Esegue prima la somma vettoriale delle coordinate tramite `Vet_Sum` e ne dimezza istantaneamente il risultato moltiplicandolo per il fattore scalare 0.5 tramite il motore `Vet_Sca`.

Vet_Perp_Sca

Firma: (Vet_Perp_Sca v s)

Scopo: Calcolare il vettore perpendicolare (normale) nel piano XY rispetto a un vettore dato e scalarlo simultaneamente di un fattore definito.

Input: v (Lista rappresentante il vettore direzionale di partenza), s (Numero reale, fattore di scala o magnitudo da applicare).

Output: Lista di tre elementi (vettore 3D) rappresentante la normale scalata.

Logica interna: Applica la trasformazione geometrica algebrica per la rotazione ortogonale 2D (x, y) -> (-y, x).

Estrae l'ordinata (cadr), la inverte di segno e la moltiplica per lo scalare s per generare la nuova ascissa. L'ascissa originale (car) viene scalata per s e diventa la nuova ordinata. La componente Z viene forzata rigidamente a 0.0 per confinare il risultato sul piano di lavoro bidimensionale.

Vet_Sca

Firma: (Vet_Sca v s)

Scopo: Eseguire il prodotto vettoriale di uno spazio (o punto) per uno scalare, modulandone l'intensità (magnitudo) o invertendone il verso (se il fattore numerico à negativo) senza alterarne la linea d'azione.

Input: v (Lista di numeri reali che rappresenta le componenti del vettore base), s (Numero reale indicante il fattore di moltiplicazione o scala).

Output: Lista di coordinate reali corrispondente al nuovo vettore scalato.

Logica interna: Sfrutta l'iteratore `mapcar` per scorrere sequenzialmente tutte le componenti della lista vettoriale `v`. Per ogni elemento, applica una funzione `lambda` anonima in linea che esegue la moltiplicazione per la costante `s`, distribuendo l'operazione su tutti gli assi con il minimo overhead.

Vet_Sub

Firma: (Vet_Sub v1 v2)

Scopo: Eseguire le operazioni aritmetiche vettoriali di base (somma e differenza) componente per componente tra due vettori o punti spaziali (2D o 3D).

Input: v1, v2 (Liste di numeri reali che rappresentano le coordinate dei vettori).

Output: Lista di numeri reali rappresentante il vettore risultante.

Logica interna: Sfruttano l'iteratore funzionale 'mapcar' associato agli operatori aritmetici primitivi ('+' per la somma, '-' per la sottrazione) per elaborare simultaneamente le coordinate con il minimo overhead computazionale. (defun Vet_Sum (v1 v2) (mapcar '+ v1 v2))

Vet_Sum

Firma: (Vet_Sum v1 v2)

Scopo: Eseguire le operazioni aritmetiche vettoriali di base (somma e differenza) componente per componente tra due vettori o punti spaziali (2D o 3D).

Input: v1, v2 (Liste di numeri reali che rappresentano le coordinate dei vettori).

Output: Lista di numeri reali rappresentante il vettore risultante.

Logica interna: Sfruttano l'iteratore funzionale 'mapcar' associato agli operatori aritmetici primitivi ('+' per la somma, '-' per la sottrazione) per elaborare simultaneamente le coordinate con il minimo overhead computazionale.

Vet_Uni

Firma: (Vet_Uni v)

Scopo: Normalizza un vettore 2D o 3D al modulo unitario (versore), dividendo ogni componente per il modulo del vettore.

Input: v: lista di coordinate del vettore da normalizzare.

Output: Lista delle coordinate del versore (vettore con modulo 1.0); nil o vettore nullo se il modulo è inferiore alla soglia di singolarità.

Logica interna: Calcola il modulo con Pitag_Ip (o distance dall'origine), verifica che sia > 1e-12, poi divide ogni componente per il modulo con mapcar.

Vet_Uni_Bisect

Firma: (Vet_Uni_Bisect v1 v2)

Scopo: Determinare la direzione esatta (versore) che taglia esattamente a metà l'angolo compreso tra due vettori generici, fondamentale per algoritmi di offset, centri di raccordo e normali ai vertici.

Input: v1, v2 (Liste di coordinate numeriche reali rappresentanti i due vettori di riferimento).

Output: Lista di coordinate reali corrispondente al versore puro della bisettrice.

Logica interna: Applica algebricamente la regola del parallelogramma su vettori di modulo equiparato. Estrae

i versori unitari di entrambi gli argomenti tramite `Vet_Uni`, li somma vettorialmente con `Vet_Sum` e normalizza la risultante finale richiamando nuovamente `Vet_Uni` per generare il versore puro.

CODICE DEL LISTATO

```
;;-------------------------------------------------------------------------------------
 (vl-load-com)
;;-------------------------------------------------------------------------------------
(defun Math_Classify_Forma_4Pu (Pu1 Pu2 Pu3 Pu4)
  (cond ((P_6Coinc_4Pu                Pu1 Pu2 Pu3 Pu4)
     (list 'DEGENERE_PUNTO_4          (list                     Pu1 Pu2 Pu3 Pu4)))
        ((P_3Coinc_4Pu                Pu1 Pu2 Pu3 Pu4)
     (list 'DEGENERE_SEGMENTO_3_1     (list                     Pu1 Pu2 Pu3 Pu4)))
        ((P_2Coinc_4Pu                Pu1 Pu2 Pu3 Pu4)
     (list 'DEGENERE_SEGMENTO_2_2     (list                     Pu1 Pu2 Pu3 Pu4)))
        ((P_1Coinc_4Pu                Pu1 Pu2 Pu3 Pu4)
     (list 'DEGENERE_TRIANGOLO_2_1_1 (list                      Pu1 Pu2 Pu3 Pu4)))
        ((P_Allin_4Pu                 Pu1 Pu2 Pu3 Pu4)
     (list 'DEGENERE_ALLINEATI_4      (list                     Pu1 Pu2 Pu3 Pu4)))
        ((P_Allin_3su4Pu              Pu1 Pu2 Pu3 Pu4 T)
     (list 'DEGENERE_ALLINEATI_3_1    (list                     Pu1 Pu2 Pu3 Pu4)))
        ((P_Concavo_Ortocentrico_4Pu Pu1 Pu2 Pu3 Pu4)
     (list 'CONCAVO_ORTOCENTRICO      (Geom_Ordina_Concavo_4Pu Pu1 Pu2 Pu3 Pu4)))
        ((P_Conca_4Pu                 Pu1 Pu2 Pu3 Pu4)
     (list 'CONCAVO_GENERICO          (Geom_Ordina_Concavo_4Pu Pu1 Pu2 Pu3 Pu4)))
        ((P_Quadrato_4Pu              Pu1 Pu2 Pu3 Pu4)
     (list 'QUADRATO                  (Geom_Ordina_CCW_4Pu     Pu1 Pu2 Pu3 Pu4)))
        ((P_Rettangolo_4Pu            Pu1 Pu2 Pu3 Pu4)
     (list 'RETTANGOLO                (Geom_Ordina_CCW_4Pu     Pu1 Pu2 Pu3 Pu4)))
        ((P_Rombo_4Pu                 Pu1 Pu2 Pu3 Pu4)
     (list 'ROMBO                     (Geom_Ordina_CCW_4Pu     Pu1 Pu2 Pu3 Pu4)))
        ((P_Parallelogramma_4Pu       Pu1 Pu2 Pu3 Pu4)
     (list 'PARALLELOGRAMMA           (Geom_Ordina_CCW_4Pu     Pu1 Pu2 Pu3 Pu4)))
        ((P_Trapezio_Isoscele_4Pu     Pu1 Pu2 Pu3 Pu4)
     (list 'TRAPEZIO_ISOSCELE         (Geom_Ordina_CCW_4Pu     Pu1 Pu2 Pu3 Pu4)))
        ((P_Trapezio_Scaleno_4Pu      Pu1 Pu2 Pu3 Pu4)
     (list 'TRAPEZIO_SCALENO          (Geom_Ordina_CCW_4Pu     Pu1 Pu2 Pu3 Pu4)))
        ((P_Conciclici_4Pu            Pu1 Pu2 Pu3 Pu4)
     (list 'QUADRANGOLO_CICLICO       (Geom_Ordina_CCW_4Pu     Pu1 Pu2 Pu3 Pu4)))
        ((P_Quadrangolo_Generico_4Pu Pu1 Pu2 Pu3 Pu4)
     (list 'QUADRANGOLO_GENERICO      (Geom_Ordina_CCW_4Pu     Pu1 Pu2 Pu3 Pu4)))
        (T                                                  (list 'ERRORE_CLASSIFICAZIONE   (list
     Pu1 Pu2 Pu3 Pu4)))))
;;-------------------------------------------------------------------------------------
(defun Geom_2Pu_Simm_PuC (Pu1 vec len / v_offset)  (setq v_offset (Vet_Sca vec len))
     (list (Vet_Sum Pu1 v_offset) (Vet_Sub Pu1 v_offset)))
;;-------------------------------------------------------------------------------------
(defun Geom_1Pu_Simm_PuC (Pu1 Pu2) (Vet_Sum Pu2 (Vet_Sub Pu2 Pu1)))
```

```
;;-------------------------------------------------------------------------------------
(defun Math_Circonferenza_3Pu (P1 P2 P3 / x1 y1 x2 y2 x3 y3 sq1 sq2 sq3 D cx cy center)
  (if (not (P_Allin_3Pu P1 P2 P3))
    (progn
      (mapcar 'set '(x1 y1) P1)
      (mapcar 'set '(x2 y2) P2)
      (mapcar 'set '(x3 y3) P3)
      (setq sq1 (Math_Sum_Squares x1 y1)
            sq2 (Math_Sum_Squares x2 y2)
            sq3 (Math_Sum_Squares x3 y3))
      (setq D   (* 2.0 (Math_Cyclic_Det x1 x2 x3 y1 y2 y3)))

      (if (> (abs D) 1e-12)
        (progn
          (setq cx     (/    (Math_Cyclic_Det sq1 sq2 sq3 y1 y2 y3)  D))
          (setq cy     (/ (- (Math_Cyclic_Det sq1 sq2 sq3 x1 x2 x3)) D))
          (setq center (list cx cy 0.0))
          (list center (distance center P1)))))))
;;-------------------------------------------------------------------------------------
(defun Math_Dist_2Pu (Pu1 Pu2)  (if (and Pu1 (listp Pu1) Pu2 (listp Pu2)) (distance Pu1 Pu2)))
;;-------------------------------------------------------------------------------------
(defun Math_PuntoMedio_2Pu (Pu1 Pu2) (mapcar '(lambda (x1 x2) (* 0.5 (+ x1 x2))) Pu1 Pu2))
;;-------------------------------------------------------------------------------------
(defun Math_lisPuntoMedio_lis2Pu (lst_pairs) (mapcar '(lambda (pair)
     (apply 'Math_PuntoMedio_2Pu pair)) lst_pairs))
;;-------------------------------------------------------------------------------------
(defun Math_LisPuntiMedi_Quadrangolo_4Pu (P1 P2 P3 P4)
  (list (Math_PuntoMedio_2Pu P1 P2) (Math_PuntoMedio_2Pu P2 P3) (Math_PuntoMedio_2Pu P3 P4)
     (Math_PuntoMedio_2Pu P4 P1)
        (Math_PuntoMedio_2Pu P1 P3) (Math_PuntoMedio_2Pu P2 P4)))
;;-------------------------------------------------------------------------------------
(defun Math_Inters_Quaterna (quaterna tipo / P1 P2 P3 P4)
  (mapcar 'set '(P1 P2 P3 P4) quaterna)
  (cond  ((= tipo 1) (inters P1 P2 P3 P4 nil))
         ((= tipo 2) (inters P2 P3 P4 P1 nil))
         ((= tipo 3) (inters P1 P3 P2 P4 nil))))
;;-------------------------------------------------------------------------------------
(defun Math_Inters_lisQuaterne (lst_quaterne tipo)  (mapcar '(lambda (q)
     (Math_Inters_Quaterna q tipo)) lst_quaterne))
;;-------------------------------------------------------------------------------------
(defun Math_Proiezione_1Pu_su_2Pu (pup pul1 pul2 / v u d)
  (setq v (Vet_Sub pup  pul1)
        u (Vet_Sub pul2 pul1)
        d (Vet_Dot u u))
  (if (> d 1e-12)
```

```
      (Vet_Sum pul1 (Vet_Sca u (/ (Vet_Dot v u) d)))
      pul1))
;;-------------------------------------------------------------------------------------------
(defun Math_Proiezione_lisPu_su_2Pu (lst_Pu Pu2 Pu3)  (mapcar '(lambda (p)
     (Math_Proiezione_1Pu_su_2Pu p Pu2 Pu3)) lst_Pu))
;;-------------------------------------------------------------------------------------------
(defun Math_Proiezione_lisTerne (lst_terne) (mapcar '(lambda (terna)
     (apply 'Math_Proiezione_1Pu_su_2Pu terna)) lst_terne))
;;-------------------------------------------------------------------------------------------
(defun Math_Ortocentro_3Pu (Pu1 Pu2 Pu3)
  (inters Pu3 (Math_Proiezione_1Pu_su_2Pu Pu3 Pu1 Pu2)
          Pu1 (Math_Proiezione_1Pu_su_2Pu Pu1 Pu2 Pu3) nil))
;;-------------------------------------------------------------------------------------------
(defun Math_TriangoloDiagonale_4Pu (Pu1 Pu2 Pu3 Pu4)
  (Math_Inters_lisQuaterne (list (list Pu1 Pu2 Pu3 Pu4) (list Pu1 Pu3 Pu2 Pu4)
     (list Pu1 Pu4 Pu2 Pu3)) 1))
;;-------------------------------------------------------------------------------------------
(defun Math_ConicaCentri_9Pu (Pu1 Pu2 Pu3 Pu4)
  (append (Math_LisPuntiMedi_Quadrangolo_4Pu Pu1 Pu2 Pu3 Pu4)
          (Math_TriangoloDiagonale_4Pu       Pu1 Pu2 Pu3 Pu4)))
;;-------------------------------------------------------------------------------------------
(defun Math_Prod_Sum     (a b c d)    (* (+ a b) (+ c d)))
;;-------------------------------------------------------------------------------------------
(defun Math_Prod_Diff    (a b c d)    (* (- a b) (- c d)))
;;-------------------------------------------------------------------------------------------
(defun Math_Sum_Prod     (a b c d)    (+ (* a b) (* c d)))
;;-------------------------------------------------------------------------------------------
(defun Math_Diff_Prod    (a b c d)    (- (* a b) (* c d)))
;;-------------------------------------------------------------------------------------------
(defun Math_Sum_Squares  (a b)        (Math_Sum_Prod a a b b))
;;-------------------------------------------------------------------------------------------
(defun Math_Diff_Squares (a b)        (Math_Diff_Prod a a b b))
;;-------------------------------------------------------------------------------------------
(defun Math_Sqr          (a)          (* a a))
;;-------------------------------------------------------------------------------------------
(defun Math_Cyclic_Det   (k1 k2 k3 v1 v2 v3)  (+ (* k1 (- v2 v3)) (* k2 (- v3 v1))
     (* k3 (- v1 v2))))
;;-------------------------------------------------------------------------------------------
(defun Math_Inv_Sqrt (x)    (if (and (numberp x) (> x 1e-12))      (sqrt (/ 1.0 x))     0.0))
;;-------------------------------------------------------------------------------------------
(defun Pitag_Ip (ca1 ca2)   (if (and ca1 ca2)            (sqrt (Math_Sum_Squares ca1 ca2))))
;;-------------------------------------------------------------------------------------------
(defun Pitag_Ca (ip ca)     (if (and ip ca (> ip ca))    (sqrt (Math_Diff_Squares ip ca))))
;;-------------------------------------------------------------------------------------------
(defun Math_Acos (val)
```

```
  (cond ((>= val  1.0) 0.0)
        ((<= val -1.0) pi)
        (T (atan (sqrt (max 0.0 (- 1.0 (* val val)))) val))))
;;-------------------------------------------------------------------------------------
(defun Vet_Sum (v1 v2)     (mapcar '+ v1 v2))
;;-------------------------------------------------------------------------------------
(defun Vet_Sub (v1 v2)     (mapcar '- v1 v2))
;;-------------------------------------------------------------------------------------
(defun Vet_Sca (v s)       (mapcar '(lambda (x) (* x s)) v))
;;-------------------------------------------------------------------------------------
(defun Vet_Len (v)         (distance '(0.0 0.0 0.0) v))
;;-------------------------------------------------------------------------------------
(defun Vet_Mid (p1 p2)     (mapcar '(lambda (a b) (/ (+ a b) 2.0)) p1 p2))
;;-------------------------------------------------------------------------------------
(defun Vet_Mid (p1 p2)     (Vet_Sca (Vet_Sum p1 p2) 0.5))
;;-------------------------------------------------------------------------------------
(defun Vet_Uni (v / dist)  (setq dist (Vet_Len v)) (if (> dist 1e-12) (Vet_Sca v (/ 1.0 dist))
     nil))
;;-------------------------------------------------------------------------------------
(defun Vet_Uni (v / d)     (setq d    (distance '(0 0 0) v)) (if (> d 1e-12)
     (Vet_Sca v (/ 1.0 d)) v))
;;-------------------------------------------------------------------------------------
(defun Vet_Uni (v / len)   (setq len  (sqrt (Vet_Dot v v))) (if (> len 1e-14)
     (Vet_Sca v (/ 1.0 len)) '(0.0 0.0 0.0)))
;;-------------------------------------------------------------------------------------
(defun Vet_Dot (v1 v2)  (apply '+ (mapcar '* v1 v2)))
;;-------------------------------------------------------------------------------------
(defun Vet_Cro (v1 v2 / x1 y1 z1 x2 y2 z2)
  (mapcar 'set '(x1 y1 z1 x2 y2 z2) (append v1 v2))
  (list (Math_Diff_Prod y1 z2 z1 y2)
        (Math_Diff_Prod z1 x2 x1 z2)
        (Math_Diff_Prod x1 y2 y1 x2)))
;;-------------------------------------------------------------------------------------
(defun Vet_Uni_Bisect (v1 v2)  (Vet_Uni (Vet_Sum (Vet_Uni v1) (Vet_Uni v2))))
;;-------------------------------------------------------------------------------------
(defun Vet_Perp_Sca (v s) (list (* (- (cadr v)) s) (* (car v) s) 0.0))
;;-------------------------------------------------------------------------------------
(defun Util_Make_Anonymous_Group (lst_ents descr / ss e old_cmd)
  (if (and lst_ents (> (length lst_ents) 1))
    (progn
      (setq ss (ssadd))
      (foreach e lst_ents (if (entget e) (ssadd e ss)))
      (if (> (sslength ss) 0)
        (progn
          (setq old_cmd (getvar "CMDECHO"))
```

```
          (setvar "CMDECHO" 0)
          (vl-cmdf "_.-GROUP" "_Create" "*" descr ss "")
          (setvar "CMDECHO" old_cmd)))))
  (princ))
;;-----------------------------------------------------------------------------
(defun Math_Solve_5x5 (mat vec / n aug i j k max_r pivot factor temp_aug new_row x_list sum x_v
     al temp_res)
  (setq n 5)
  (setq aug (mapcar 'append mat (mapcar 'list vec)))

  (setq i 0)
  (while (< i n)
    (setq max_r i j (1+ i))
    (while (< j n)
      (if (> (abs (nth i (nth j aug))) (abs (nth i (nth max_r aug))))
          (setq max_r j))
      (setq j (1+ j)))

    (if (/= i max_r)
        (setq aug (mapcar '(lambda (idx row)
                             (cond ((= idx i) (nth max_r aug))
                                   ((= idx max_r) (nth i aug))
                                   (T row)))
                          '(0 1 2 3 4) aug)))

    (setq pivot (nth i (nth i aug)))
    (if (< (abs pivot) 1e-12)
        (setq i 99)
        (progn
          (setq temp_aug nil j 0)
          (while (< j n)
            (if (<= j i)
                (setq temp_aug (append temp_aug (list (nth j aug))))
                (progn
                  (setq factor (/ (float (nth i (nth j aug))) (float pivot)))
                  (setq new_row nil k 0)
                  (while (<= k n)
                    (setq new_row (append new_row (list (- (nth k (nth j aug))
     (* factor (nth k (nth i aug)))))))
                    (setq k (1+ k)))
                  (setq temp_aug (append temp_aug (list new_row)))
                ))
            (setq j (1+ j)))
          (setq aug temp_aug)
        ))
```

```
    (setq i (1+ i)))

  (if (> i 90)
      nil
      (progn
        (setq x_list '(0.0 0.0 0.0 0.0 0.0) i (1- n))
        (while (>= i 0)
          (setq sum (nth n (nth i aug)) j (1+ i))
          (while (< j n)
            (setq sum (- sum (* (nth j (nth i aug)) (nth j x_list))))
            (setq j (1+ j)))
          (setq x_val (/ sum (float (nth i (nth i aug)))))

          (setq temp_res nil k 0)
          (while (< k n)
            (if (= k i)
                (setq temp_res (append temp_res (list x_val)))
                (setq temp_res (append temp_res (list (nth k x_list)))))
            (setq k (1+ k)))
          (setq x_list temp_res)
          (setq i (1- i)))
        x_list
      )))
;;-------------------------------------------------------------------------------------------
(defun replace-nth (lst n val / i out)
  (setq out '() i 0)
  (foreach el lst
    (setq out (cons (if (= i n) val el) out))
    (setq i (1+ i)))
  (reverse out))
;;-------------------------------------------------------------------------------------------
(defun get-all-permutations (lst)
  (if (null lst) '(nil)
    (apply 'append (mapcar '(lambda (el) (mapcar '(lambda (p) (cons el p))
     (get-all-permutations (vl-remove el lst)))) lst))))
;;-------------------------------------------------------------------------------------------
(defun solve-system-cramer-2x2 (a1 b1 c1 a2 b2 c2 / det)
  (setq det (Math_Diff_Prod a1 b2 a2 b1))
  (if (> (abs det) 1e-9) (list (/ (Math_Diff_Prod c1 b2 c2 b1) det)
     (/ (Math_Diff_Prod a1 c2 a2 c1) det))))
;;-------------------------------------------------------------------------------------------
(defun get-oblique-coords (Pu1 Pu2 Pu3 Pu4 / ux uy vx vy dx dy sol)
  (mapcar 'set '(ux uy) (Vet_Uni (Vet_Sub Pu3 Pu2)))
  (mapcar 'set '(vx vy) (Vet_Uni (Vet_Sub Pu4 Pu2)))
  (mapcar 'set '(dx dy) (Vet_Sub Pu1 Pu2))
```

```
  (setq sol (if (and ux vx) (solve-system-cramer-2x2 ux vx dx uy vy dy)))
  (if sol (append sol (list ux uy vx vy))))
;;-------------------------------------------------------------------------------------------
(defun Geom_2Pu_Simm_PuC (Pu1 vec len / v_offset)  (setq v_offset (Vet_Sca vec len))
     (list (Vet_Sum Pu1 v_offset) (Vet_Sub Pu1 v_offset)))
;;-------------------------------------------------------------------------------------------
(defun Geom_1Pu_Simm_PuC (Pu1 PuC)  (Vet_Sum PuC (Vet_Sub PuC Pu1)))
;;-------------------------------------------------------------------------------------------
(defun get-unit-vector-from-pts (Pu1 Pu2) (Vet_Uni (Vet_Sub Pu2 Pu1)))
;;-------------------------------------------------------------------------------------------
(defun Math_Extract_Coord_Pow (pts idx pow)  (mapcar '(lambda (p) (expt (nth idx p) pow)) pts))
;;-------------------------------------------------------------------------------------------
(defun Math_MinMax_List (lst flag)
  (if (and lst (listp lst))
      (cond ((eq flag 0) (apply 'min lst))
            ((eq flag 1) (apply 'max lst))
            ((eq flag 2) (list (apply 'min lst) (apply 'max lst))))))
;;-------------------------------------------------------------------------------------------
(defun Math_Delta_List (lst / res)  (if (setq res (Math_MinMax_List lst 2)) (- (cadr res)
     (car res))))
;;-------------------------------------------------------------------------------------------
(defun Util_Get_Pts_BBox_Size (pts)  (list (Math_Delta_List (Math_Extract_Coord_Pow pts 0 1))
                                           (Math_Delta_List (Math_Extract_Coord_Pow pts 1 1))))
;;-------------------------------------------------------------------------------------------
(defun Draw_Point_R    (Pu1 lay)               (entmakex (list '(0 . "POINT")  (cons 10 Pu1)
     (cons 8 (if lay lay "0")))))
;;-------------------------------------------------------------------------------------------
(defun Draw_Line_2Pu_R (Pu1 Pu2 lay)           (entmakex (list '(0 . "LINE")   (cons 10 Pu1)
     (cons 11 Pu2) (cons 8 (if lay lay "0")))))
;;-------------------------------------------------------------------------------------------
(defun Draw_Circle_R   (Pu1 Rad Lay)           (entmakex (list '(0 . "CIRCLE") (cons 10 Pu1)
     (cons 40 Rad) (cons 8 (if Lay Lay "0")))))
;;-------------------------------------------------------------------------------------------
(defun Draw_ConEll_R   (Pu1 Vec_Maj Ratio lay)  (entmakex (list '(0 . "ELLIPSE")
     '(100 . "AcDbEntity") '(100 . "AcDbEllipse") (cons 8 (if lay lay "0")) (cons 10 Pu1)
     (cons 11 Vec_Maj) (cons 40 Ratio) '(41 . 0.0) (cons 42 (* 2.0 pi)))))
;;-------------------------------------------------------------------------------------------
(defun Draw_Lis_Line_Lis_Coppie_Pu_R (lis_coppie lay)  (mapcar '(lambda (coppia)
     (Draw_Line_2Pu_R (car coppia) (cadr coppia) lay)) lis_coppie))
;;-------------------------------------------------------------------------------------------
(defun Draw_Lis_Point_R              (lis_punti lay)   (mapcar '(lambda (pt)
     (Draw_Point_R pt lay)) lis_punti))
;;-------------------------------------------------------------------------------------------
(defun Draw_Triangolo_Lis3Pu (pts lay / P1 P2 P3)
  (mapcar 'set '(P1 P2 P3) pts)
```

```
  (Draw_Lis_Line_Lis_Coppie_Pu_R (list (list P1 P2) (list P2 P3) (list P3 P1)) lay))
;;-------------------------------------------------------------------------------------
(defun Draw_Triangolo_Ortico_Lis3Pu (pts lay / P1 P2 P3 H1 H2 H3)
  (mapcar 'set '(P1 P2 P3) pts)
  (setq H1 (Math_Proiezione_1Pu_su_2Pu P1 P2 P3)
        H2 (Math_Proiezione_1Pu_su_2Pu P2 P3 P1)
        H3 (Math_Proiezione_1Pu_su_2Pu P3 P1 P2))
  (Draw_Lis_Line_Lis_Coppie_Pu_R (list (list H1 H2) (list H2 H3) (list H3 H1)) lay))
;;-------------------------------------------------------------------------------------
(defun Draw_Quadrangolo_Convesso_Lis4Pu (pts lay / P1 P2 P3 P4)
  (mapcar 'set '(P1 P2 P3 P4) pts)  (Draw_Lis_Line_Lis_Coppie_Pu_R (list (list P1 P2)
     (list P2 P3) (list P3 P4)(list P4 P1)) lay))
;;-------------------------------------------------------------------------------------
(defun Draw_Quadrangolo_Concavo_Lis4Pu (pts lay / P1 P2 P3 P4)
  (mapcar 'set '(P1 P2 P3 P4) pts)  (Draw_Lis_Line_Lis_Coppie_Pu_R (list (list P1 P2)
     (list P1 P4) (list P2 P3) (list P2 P4)(list P3 P1)(list P3 P4)) lay))
;;-------------------------------------------------------------------------------------
(defun Draw_Spline_Rational_3CV (Pu1 Pu2 Pu3 w lay) (entmakex (list '(0 . "SPLINE")
     '(100 . "AcDbEntity") '(100 . "AcDbSpline")
                                                          '(70 . 4) '(71 . 2) '(72 . 6) '(73 . 3)
'(74 . 0) (cons 8 (if lay lay "0"))
                                                          '(40 . 0.0) '(40 . 0.0) '(40 . 0.0)
'(40 . 1.0) '(40 . 1.0) '(40 . 1.0) '(41 . 1.0)
                                                          (cons 10 Pu1) (cons 41 w) (cons 10 Pu2)
'(41 . 1.0) (cons 10 Pu3))))
;;-------------------------------------------------------------------------------------
(defun Get_String_Def (msg def / val)  (setq val (getstring T (strcat msg " <" def ">: ")))
     (if (= val "") def val))
;;-------------------------------------------------------------------------------------
(defun Get_Int_Pos (msg def / val)  (initget 6)  (setq val (getint (strcat msg " <" (itoa def)
     ">: ")))  (if val val def))
;;-------------------------------------------------------------------------------------
(defun Lay_Make_Col (lay col / lay_ename lay_data safe_col curr_col)
  (setq safe_col (if (and col (> col 0) (< col 256)) col 7))
  (if (setq lay_ename (tblobjname "LAYER" lay))
      (progn
        (setq lay_data (entget lay_ename)
              curr_col (cdr (assoc 62 lay_data)))
        (if (/= curr_col safe_col)
            (entmod (subst (cons 62 safe_col) (assoc 62 lay_data) lay_data))))
      (entmake (list '(0 . "LAYER")
                     '(100 . "AcDbSymbolTableRecord")
                     '(100 . "AcDbLayerTableRecord")
                     (cons 2 lay)
                     '(70 . 0)
```

```
                    (cons 62 safe_col))))
  lay)
;;-------------------------------------------------------------------------------------
(defun Util_SafeLayer (lay)
  (if (not (tblsearch "LAYER" lay))
      (entmake (list '(0 . "LAYER") '(100 . "AcDbSymbolTableRecord")
     '(100 . "AcDbLayerTableRecord") (cons 2 lay) '(70 . 0))))  lay)
;;-------------------------------------------------------------------------------------
(defun Get_N_Pts_Check (Title numPu lay / Pu1 pts)
  (if Title (princ Title))
  (if (setq Pu1 (getpoint "\nPunto 1: "))
    (progn
      (Princ_1Pun " >> " Pu1)
      (Draw_Point_R Pu1 lay)
      (setq pts (list Pu1))
      (while (and (< (length pts) numPu) (setq Pu1 (getpoint (car pts)
     (strcat "\nPunto " (itoa (1+ (length pts))) ": "))))
        (Princ_1Pun " >> " Pu1)
        (Draw_Point_R Pu1 lay)
        (setq pts (cons Pu1 pts)))
      (if (= (length pts) numPu) (reverse pts) (progn (princ "\nInput incompleto.") nil)))))
;;-------------------------------------------------------------------------------------
(defun Get_1Pts_Check (Title lay) (Get_N_Pts_Check Title 1 lay))
;;-------------------------------------------------------------------------------------
(defun Get_2Pts_Check (Title lay) (Get_N_Pts_Check Title 2 lay))
;;-------------------------------------------------------------------------------------
(defun Get_3Pts_Check (Title lay) (Get_N_Pts_Check Title 3 lay))
;;-------------------------------------------------------------------------------------
(defun Get_4Pts_Check (Title lay) (Get_N_Pts_Check Title 4 lay))
;;-------------------------------------------------------------------------------------
(defun Get_5Pts_Check (Title lay) (Get_N_Pts_Check Title 5 lay))
;;-------------------------------------------------------------------------------------
(defun Pu_Baricentro2D_LisPu (LisPu / n)  (setq n (float (length LisPu)))
     (list (/ (apply '+ (mapcar 'car  LisPu)) n) (/ (apply '+ (mapcar 'cadr LisPu)) n) 0.0))
;;-------------------------------------------------------------------------------------
(defun Util_Get_Max_Dist_From_Point (pts ref_pt)  (apply 'max (mapcar '(lambda (p)
     (distance p ref_pt)) pts)))
;;-------------------------------------------------------------------------------------
(defun Util_Get_Min_Dist_From_Point (pts ref_pt)  (apply 'min (mapcar '(lambda (p)
     (distance p ref_pt)) pts)))
;;-------------------------------------------------------------------------------------
(defun P_center-infinite (pts center threshold / G max_dist dist_center)
  (setq G        (Pu_Baricentro2D_LisPu pts))
  (setq max_dist (Util_Get_Max_Dist_From_Point pts G))
  (setq dist_center (distance G center))
```

```
  (and (> max_dist 1e-6) (> dist_center (* max_dist threshold))))
;;-------------------------------------------------------------------------------------
(defun Pascal_Calcola_Primo_Diametro (pts_6 / p1 p2 p3 p4 p5 p6 p7 p8)
  (mapcar 'set '(p1 p2 p3 p4 p5 p6) pts_6)
  (setq p7 (Math_PuntoMedio_2Pu p3 p4)
        p8 (Math_PuntoMedio_2Pu p1 p6))
  (if (not (equal p7 p8 1e-6))
      (list p7 p8 (angle p7 p8))))
;;-------------------------------------------------------------------------------------
(defun Pascal_Calcola_Corde_Coniugate (pts ang_dir / p1 p2 p3 p4 p5 p9 p10)
  (mapcar 'set '(p1 p2 p3 p4 p5) pts)
  (setq p9  (Pascal_Calcola_Singola_Corda p1 p2 p3 p4 p5 ang_dir))
  (setq p10 (Pascal_Calcola_Singola_Corda p4 p3 p2 p1 p5 ang_dir))
  (if (and p9 p10) (list p9 p10)))
;;-------------------------------------------------------------------------------------
(defun Util_Reorder_List_By_Indices (lst idx_list)  (mapcar '(lambda (i) (nth i lst))
     idx_list))
;;-------------------------------------------------------------------------------------
(defun Math_Solve_Oblique_2P (coords_A coords_B / u1 v1 u2 v2)
  (mapcar 'set '(u1 v1) coords_A)
  (mapcar 'set '(u2 v2) coords_B)
  (solve-system-cramer-2x2 (Math_Sqr u1) (Math_Sqr v1) 1.0 (Math_Sqr u2) (Math_Sqr v2) 1.0))
;;-------------------------------------------------------------------------------------
(defun probe-conic-algebraic-tipo (pts center ref_p1 ref_p2 / i j found ptA ptB coords_A coords
     _B cramer_res result_type)
  (cond
    ((null center) 'PARABOLA)
    (T
     (setq found nil i 0 result_type nil)
     (while (and (< i 4) (not found))
       (setq j (+ i 1))
       (while (and (< j 5) (not found))
          (setq ptA (nth i pts)
                ptB (nth j pts)
                coords_A (get-oblique-coords ptA center ref_p1 ref_p2)
                coords_B (get-oblique-coords ptB center ref_p1 ref_p2))
          (if (and coords_A coords_B)
            (progn
              (setq cramer_res (Math_Solve_Oblique_2P coords_A coords_B))
              (if cramer_res
                (progn
                  (setq result_type (if (> (apply '* cramer_res) 0) 'ELLISSE 'IPERBOLE))
                  (setq found T)))))
          (setq j (+ j 1)))
       (setq i (+ i 1)))
```

```
    result_type)))
;;-------------------------------------------------------------------------------------
(defun Math_Solve_Par_3P (p1 p2 p3 / x1 y1 x2 y2 x3 y3 D A B C)
  (mapcar 'set '(x1 y1) p1)
  (mapcar 'set '(x2 y2) p2)
  (mapcar 'set '(x3 y3) p3)
  (setq D (+ (* x1 x1 (- x2 x3))
             (* x2 x2 (- x3 x1))
             (* x3 x3 (- x1 x2))))
  (if (> (abs D) 1e-9)
    (progn
      (setq A (/ (+ (* y1 (- x2 x3)) (* y2 (- x3 x1)) (* y3 (- x1 x2))) D)
            B (/ (+ (* x1 x1 (- y2 y3)) (* x2 x2 (- y3 y1)) (* x3 x3 (- y1 y2))) D)
            C (/ (+ (* x1 x1 (- (* x2 y3) (* x3 y2)))
                    (* x2 x2 (- (* x3 y1) (* x1 y3)))
                    (* x3 x3 (- (* x1 y2) (* x2 y1)))) D))
      (if (> (abs A) 1e-12)
        (list A B C)
        nil))))
;;-------------------------------------------------------------------------------------
(defun Math_Calc_Par_Global_Error (pts_local A B C / total_err xt yt val_test)
  (setq total_err 0.0)
  (foreach pt_test pts_local
    (mapcar 'set '(xt yt) pt_test)
    (setq val_test  (+ (* A xt xt) (* B xt) C)
          total_err (+ total_err (abs (- val_test yt)))))
  total_err)
;;-------------------------------------------------------------------------------------
(defun Geom_Extract_Par_World (A B C origin theta / xv_loc yv_loc p_focal V_loc F_loc V_world F
    _world)
  (setq xv_loc  (/ (- B) (* 2.0 A))
        yv_loc  (+ (* A xv_loc xv_loc) (* B xv_loc) C)
        p_focal (/ 1.0 (* 4.0 A))
        V_loc   (list xv_loc yv_loc 0.0)
        F_loc   (list xv_loc (+ yv_loc p_focal) 0.0)
        V_world (Trans_Pt_Loc_To_Glo V_loc origin theta)
        F_world (Trans_Pt_Loc_To_Glo F_loc origin theta))
  (list V_world F_world (abs p_focal)))
;;-------------------------------------------------------------------------------------
(defun Solve_Par_Geometry (pts ref1 ref2 / u_axis theta origin pts_local i j k p1 p2 p3 abc_res
    A B C total_err geom candidate_list best_candidate debug_log V_world F_world p_focal)
  (setq debug_log T)
  (if debug_log (princ "\n--- INIZIO TORNEO PARABOLA (TRIPLETTE) ---"))
  (setq candidate_list nil)
  (setq u_axis    (get-unit-vector-from-pts ref1 ref2)
```

```
        theta     (- (/ pi 2.0) (angle '(0 0 0) u_axis))
        origin    (car pts)
        pts_local (mapcar '(lambda (p) (Trans_Pt_Glo_To_Loc p origin theta)) pts))
  (setq i 0)
  (while (< i 3)
    (setq j (+ i 1))
    (while (< j 4)
      (setq k (+ j 1))
      (while (< k 5)
        (setq p1 (nth i pts_local)
              p2 (nth j pts_local)
              p3 (nth k pts_local)
              abc_res (Math_Solve_Par_3P p1 p2 p3))
        (if abc_res
          (progn
            (mapcar 'set '(A B C) abc_res)
            (setq total_err (Math_Calc_Par_Global_Error pts_local A B C)
                  geom      (Geom_Extract_Par_World A B C origin theta)
                  candidate_list (cons (list total_err geom) candidate_list))
            (if debug_log (princ (strcat "\n[DEBUG] Tripla " (itoa i) "-" (itoa j) "-" (itoa k)
     " -> A:" (rtos A 2 16) " | Errore Residuo: " (rtos total_err 2 16)))))
            (if debug_log (princ (strcat "\n[DEBUG] Tripla " (itoa i) "-" (itoa j) "-" (itoa k)
     " -> SCARTATA (Det nullo o Degenere in retta)"))))
          (setq k (+ k 1)))
        (setq j (+ j 1)))
      (setq i (+ i 1)))
  (if candidate_list
    (progn
      (setq candidate_list (vl-sort candidate_list '(lambda (a b) (< (car a) (car b))))
            best_candidate (car candidate_list))
      (if debug_log (princ (strcat "\n[VINCITORE] Errore Minore: " (rtos (car best_candidate)
     2 16))))
      (mapcar 'set '(V_world F_world p_focal) (cadr best_candidate))
      (princ "\n--- RISULTATO PARABOLA (OPTIMIZED) ---")
      (Princ_1Pun "\n  Vertice: " V_world)
      (Princ_1Pun "\n  Fuoco  : " F_world)
      (Princ_1Num "\n  Distanza Focale (p): " p_focal)
      (cadr best_candidate))
    (progn
      (if debug_log (princ "\n[FALLIMENTO] Impossibile risolvere sistema parabola."))
      nil)))
;;-------------------------------------------------------------------------------------
(defun get-max-abs-projection-on-axis (pts origin axis_vec)  (apply 'max (mapcar '(lambda (pt)
     (abs (Vet_Dot (Vet_Sub pt origin) axis_vec))) pts)))
```

```
;;-------------------------------------------------------------------------------------------
(defun Get_Pt_On_Axes    (Origin V_x V_y dx dy)
     (Vet_Sum (Vet_Sum Origin (Vet_Sca V_x dx)) (Vet_Sca V_y dy)))
;;-------------------------------------------------------------------------------------------
(defun Get_lisPt_On_Axes (Origin V_x V_y local_coords)  (mapcar '(lambda (loc)
     (Get_Pt_On_Axes Origin V_x V_y (car loc) (cadr loc))) local_coords))
;;-------------------------------------------------------------------------------------------
(defun Calc_Par_Branch_CVs    (V Ax No len h)
     (Get_lisPt_On_Axes V Ax No (list (list len         (- h)) (list (- len)      0.0)
     (list len          h))))
;;-------------------------------------------------------------------------------------------
(defun Calc_Par_Directrix_Pts (V Ax No dfo h)
     (Get_lisPt_On_Axes V Ax No (list (list (- dfo)     (- h)) (list (- dfo)        h)
     )))
;;-------------------------------------------------------------------------------------------
(defun Calc_Par_Axis_Pts      (V Ax No dfo len)
     (Get_lisPt_On_Axes V Ax No (list (list (* dfo -9.0)  0.0) (list len          0.0)
     )))
;;-------------------------------------------------------------------------------------------
(defun Calc_Ipe_Branch_CVs    (C Vx Vy dcv len h dir)
     (Get_lisPt_On_Axes C Vx Vy (list (list (* dir len) (- h)) (list (* dir dcv)  0.0)
     (list (* dir len)  h)                          )))
;;-------------------------------------------------------------------------------------------
(defun Calc_Ipe_Asym_Pts      (C Vx Vy len h)
     (Get_lisPt_On_Axes C Vx Vy (list (list len             h) (list (- len)    (- h))
     (list len      (- h))   (list (- len)      h))))
;;-------------------------------------------------------------------------------------------
(defun Calc_Ipe_Axes_Pts      (C Vx Vy len)
     (Get_lisPt_On_Axes C Vx Vy (list (list len           0.0) (list (- len)      0.0)
     (list 0.0        len)   (list 0.0    (- len)))))
;;-------------------------------------------------------------------------------------------
(defun Geom_Helper_Find_Par_Axis_Angle (V P1 P2 a_min a_max / a_mid p1_rot p2_rot a1 a2 err)
  (repeat 50
    (setq a_mid (/ (+ a_min a_max) 2.0)
          p1_rot (Trans_Pt_Glo_To_Loc P1 V a_mid)
          p2_rot (Trans_Pt_Glo_To_Loc P2 V a_mid))
    (if (and (> (abs (car p1_rot)) 1e-6) (> (abs (car p2_rot)) 1e-6))
        (progn
          (setq a1 (/ (cadr p1_rot) (* (car p1_rot) (car p1_rot)))
                a2 (/ (cadr p2_rot) (* (car p2_rot) (car p2_rot)))
                err (- a1 a2))
          (if (> err 0) (setq a_max a_mid) (setq a_min a_mid)))
        (setq a_min a_mid a_max a_mid)))
  a_mid)
;;-------------------------------------------------------------------------------------------
```

```
(defun Geom_Calc_Par_V_2P (V P1 P2 / ang_axis p1_loc x1 y1 z1 a1 p_focal F_loc F_world res)
  (setq ang_axis (Geom_Helper_Find_Par_Axis_Angle V P1 P2 0.0 pi)
        p1_loc   (Trans_Pt_Glo_To_Loc P1 V ang_axis))
  (mapcar 'set '(x1 y1 z1) p1_loc)
  (setq a1 (if (> (abs x1) 1e-6)
               (/ y1 (* x1 x1))
               0.0))
  (if (not (equal a1 0.0 1e-9))
      (setq p_focal (abs (/ 1.0 (* 4.0 a1)))
            F_loc   (list 0.0 (if (> a1 0) p_focal (- p_focal)) 0.0)
            F_world (Trans_Pt_Loc_To_Glo F_loc V ang_axis)
            res     (list V F_world))))
;;-----------------------------------------------------------------------------------------
(defun Get_Layer_Smart (Msg Dict_Key Std_Lay / proposed_lay msg_mod lay_name col)
  (setq proposed_lay (vlax-ldata-get "GA_Disegno_Coniche" Dict_Key))
  (if (or (not proposed_lay) (= proposed_lay "")) (setq proposed_lay Std_Lay))
  (if (tblsearch "LAYER" proposed_lay)
      (setq msg_mod (strcat Msg " [Esistente]"))
      (setq msg_mod (strcat Msg " [Da creare]")))
  (setq lay_name (Get_String_Def msg_mod proposed_lay))
  (if (not (tblsearch "LAYER" lay_name))
      (progn (setq col (Get_Int_Pos (strcat "\nLayer '" lay_name "' non esiste. Colore (1-255)"
     ) 7))
             (Lay_Make_Col lay_name col)
             (princ (strcat "\nCreato layer: " lay_name))))
  (vlax-ldata-put "GA_Disegno_Coniche" Dict_Key lay_name)
  lay_name)
;;-----------------------------------------------------------------------------------------
(defun Load_Conic_Layers_Fast ( / sym_list key_list name_list layer_specs all_exist ans lay_nam
     e lay_list)
  (setq sym_list  '(*LaCo_Punti_scelti* *LaCo_Poligoni* *LaCo_Centri* *LaCo_Tracciati*
                    *LaCo_Luoghi_geometrici* *LaCo_Costruzioni* *LaCo_Assi* *LaCo_Vertici*
                    *LaCo_Fuochi* *LaCo_Asintoti* *LaCo_Direttrice*)
        name_list '("Coniche - Punti scelti" "Coniche - Poligoni" "Coniche - Centri" "Coniche -
     Tracciati"
                    "Coniche - Luoghi geometrici" "Coniche - Costruzioni" "Coniche - Assi" "Coniche -
Vertici
     "
                    "Coniche - Fuochi" "Coniche - Asintoti" "Coniche - Direttrice")
        key_list  '("K_Pun" "K_Pol" "K_Cen" "K_Tra"
                    "K_Lug" "K_Cos" "K_Ass" "K_Ver"
                    "K_Fuo" "K_Asi" "K_Dir")
        layer_specs (mapcar 'list key_list name_list)
        all_exist T
        lay_list nil)
```

```
  (foreach spec layer_specs
    (setq lay_name (vlax-ldata-get "GA_Disegno_Coniche" (car spec)))
    (if (or (not lay_name) (= lay_name "")) (setq lay_name (cadr spec)))
    (setq lay_list (append lay_list (list lay_name)))
    (if (not (tblsearch "LAYER" lay_name)) (setq all_exist nil)))
  (if all_exist
      (progn  (initget "Si No")
              (setq ans (getkword "\nTutti i layer configurati sono gia' presenti. Usarli tutti
    ? [Si/No] <Si>: "))
              (if (not ans) (setq ans "Si"))
              (if (= ans "Si")
                  (progn  (mapcar 'set sym_list lay_list)
                          (mapcar '(lambda (k v) (vlax-ldata-put "GA_Disegno_Coniche" k v))
    key_list lay_list)
                          (princ "\nLayer inizializzati.")
                          T)))))
;;-----------------------------------------------------------------------------------------
(defun Config_Conic_Layers ( / )
  (princ "\n--- Configurazione Layer Coniche ---")
  (if (not (Load_Conic_Layers_Fast))
    (progn
      (setq *LaCo_Punti_scelti*      (Get_Layer_Smart "\nLayer Punti" "K_Pun" "Coniche - Punti
    scelti")
            *LaCo_Poligoni*          (Get_Layer_Smart "\nLayer Poligoni" "K_Pol" "Coniche - Pol
    igoni")
            *LaCo_Centri*            (Get_Layer_Smart "\nLayer Centri" "K_Cen" "Coniche - Centr
    i")
            *LaCo_Tracciati*         (Get_Layer_Smart "\nLayer Tracciati (Curve)" "K_Tra" "Coni
    che - Tracciati")
            *LaCo_Luoghi_geometrici* (Get_Layer_Smart "\nLayer Luoghi geometrici" "K_Lug" "Coni
    che - Luoghi geometrici")
            *LaCo_Costruzioni*       (Get_Layer_Smart "\nLayer Costruzioni" "K_Cos" "Coniche -
    Costruzioni")
            *LaCo_Assi*              (Get_Layer_Smart "\nLayer Assi" "K_Ass" "Coniche - Assi")
            *LaCo_Vertici*           (Get_Layer_Smart "\nLayer Vertici" "K_Ver" "Coniche - Vert
    ici")
            *LaCo_Fuochi*            (Get_Layer_Smart "\nLayer Fuochi" "K_Fuo" "Coniche - Fuoch
    i")
            *LaCo_Asintoti*          (Get_Layer_Smart "\nLayer Asintoti" "K_Asi" "Coniche - Asi
    ntoti")
            *LaCo_Direttrice*        (Get_Layer_Smart "\nLayer Direttrice" "K_Dir" "Coniche - D
    irettrice"))))
  (princ "\nConfigurazione completata.")
  (princ))
;;-----------------------------------------------------------------------------------------
```

```
(defun Init_Conic_App (title_str num_pts prompt_msg pdm / old_os old_pd old_ce pts)
  (vl-load-com)
  (setq old_os (getvar "OSMODE")
        old_pd (getvar "PDMODE")
        old_ce (getvar "CMDECHO"))
  (setvar "PDMODE" pdm)
  (setvar "CMDECHO" 0)
  (if (not (and *LaCo_Punti_scelti*      *LaCo_Centri* *LaCo_Tracciati* *LaCo_Costruzioni*
             *LaCo_Luoghi_geometrici* *LaCo_Fuochi* *LaCo_Assi*      *LaCo_Asintoti*))
    (Config_Conic_Layers))
  (if title_str (princ (strcat "\n=== " title_str " ===")))
  (setq pts (cond ((= num_pts 1) (Get_1Pts_Check prompt_msg *LaCo_Punti_scelti*))
                  ((= num_pts 2) (Get_2Pts_Check prompt_msg *LaCo_Punti_scelti*))
                  ((= num_pts 3) (Get_3Pts_Check prompt_msg *LaCo_Punti_scelti*))
                  ((= num_pts 4) (Get_4Pts_Check prompt_msg *LaCo_Punti_scelti*))
                  ((= num_pts 5) (Get_5Pts_Check prompt_msg *LaCo_Punti_scelti*))
                  ((= num_pts nil) nil)))
  (list pts old_os old_pd old_ce))
;;-------------------------------------------------------------------------------------
(defun Restore_Conic_App (old_os old_pd old_ce)
  (if old_os (setvar "OSMODE"  old_os))
  (if old_pd (setvar "PDMODE"  old_pd))
  (if old_ce (setvar "CMDECHO" old_ce))
  (princ))
;;-------------------------------------------------------------------------------------
(defun Util_Pun_To_Str (Pu1)  (if (and Pu1 (listp Pu1)) (strcat "(" (rtos (car Pu1) 2 4)
     " " (rtos (cadr Pu1) 2 4) " " (rtos (caddr Pu1) 2 4) ")") "NIL"))
;;-------------------------------------------------------------------------------------
(defun Util_Sym_To_Str (sym)  (if sym                   (vl-symbol-name sym)
     "NIL"))
;;-------------------------------------------------------------------------------------
(defun Util_Num_To_Str (num)  (if (numberp num)         (rtos num 2 4)
     "NIL"))
;;-------------------------------------------------------------------------------------
(defun Util_Int_To_Str (n)    (if (numberp n)           (itoa n)
     "NIL"))
;;-------------------------------------------------------------------------------------
(defun Util_Print_And_Nil (txt) (princ txt) nil)
;;-------------------------------------------------------------------------------------
(defun Princ_1Pun (txt Pu1)     (princ (strcat (if txt txt "")   (Util_Pun_To_Str Pu1))))
;;-------------------------------------------------------------------------------------
(defun Princ_2Pun (txt Pu1 Pu2) (princ (strcat (if txt txt "")   (Util_Pun_To_Str Pu1)
     " - " (Util_Pun_To_Str Pu2))))
;;-------------------------------------------------------------------------------------
(defun Princ_1Sym (txt sym)     (princ (strcat (if txt txt "")   (Util_Sym_To_Str sym))))
```

```
;;-------------------------------------------------------------------------------------
(defun Princ_1Num (txt num)     (princ (strcat (if txt txt "")   (Util_Num_To_Str num))))
;;-------------------------------------------------------------------------------------
(defun Princ_2Num (txt n1 n2)   (princ (strcat (if txt txt "")   (Util_Num_To_Str n1)
     " , " (Util_Num_To_Str n2))))
;;-------------------------------------------------------------------------------------
(defun Princ_1Int (txt n)       (princ (strcat (if txt txt "")   (Util_Int_To_Str n))))
;;-------------------------------------------------------------------------------------
(defun Alg3x3_Trasla              (Pu1 Pu2)     (mapcar '- Pu2 Pu1))
;;-------------------------------------------------------------------------------------
(defun Alg3x3_Vincolo             (Pu / x y)    (mapcar 'set '(x y) Pu) (list (Math_Sqr x)
     (* x y) (Math_Sqr y)))
;;-------------------------------------------------------------------------------------
(defun Alg3x3_Sostituisci_Colonna (mat col vec) (mapcar '(lambda (row v)
     (replace-nth row col v)) mat vec))
;;-------------------------------------------------------------------------------------
(defun Alg3x3_Det (M / a1 b1 c1 a2 b2 c2 a3 b3 c3)
  (mapcar 'set '(a1 b1 c1 a2 b2 c2 a3 b3 c3) (apply 'append M))
  (+ (* a1 (Math_Diff_Prod b2 c3 c2 b3))
     (* b1 (Math_Diff_Prod c2 a3 a2 c3))
     (* c1 (Math_Diff_Prod a2 b3 b2 a3))))
;;-------------------------------------------------------------------------------------
(defun Alg3x3_Risolvi (M rhs / det)
  (setq det (Alg3x3_Det M))
  (if (not (equal det 0.0 1e-9))
      (list (/ (Alg3x3_Det (Alg3x3_Sostituisci_Colonna M 0 rhs)) det)
            (/ (Alg3x3_Det (Alg3x3_Sostituisci_Colonna M 1 rhs)) det)
            (/ (Alg3x3_Det (Alg3x3_Sostituisci_Colonna M 2 rhs)) det))))
;;-------------------------------------------------------------------------------------
(defun Alg3x3_Calcola_Coeffs (Center P1 P2 P3)
  (Alg3x3_Risolvi
    (list (Alg3x3_Vincolo (Alg3x3_Trasla Center P1))
          (Alg3x3_Vincolo (Alg3x3_Trasla Center P2))
          (Alg3x3_Vincolo (Alg3x3_Trasla Center P3)))
    '(1.0 1.0 1.0)))
;;-------------------------------------------------------------------------------------
(defun Math_Det3x3 (a1 a2 a3 b1 b2 b3 c1 c2 c3)
  (+ (* a1 (Math_Diff_Prod b2 c3 b3 c2))
     (* a2 (Math_Diff_Prod b3 c1 b1 c3))
     (* a3 (Math_Diff_Prod b1 c2 b2 c1))))
;;-------------------------------------------------------------------------------------
(defun Alg3x3_Autovalori (A B C / tr det rad)
  (setq tr  (+ A C)
        det (- (* A C) (/ (Math_Sqr B) 4.0))
        rad (sqrt (abs (- (Math_Sqr tr) (* 4.0 det)))))
```

```
  (list (/ (+ tr rad) 2.0) (/ (- tr rad) 2.0)))
;;-------------------------------------------------------------------------------------
(defun Alg3x3_Autovettore (A B C lam / vx vy norm)
  (if (not (equal B 0.0 1e-9))
      (setq vx 1.0 vy (/ (- lam A) (/ B 2.0)))
      (if (< (abs (- lam A)) (abs (- lam C)))
          (setq vx 1.0 vy 0.0)
          (setq vx 0.0 vy 1.0)))
  (setq norm (Pitag_Ip vx vy))
  (list (/ vx norm) (/ vy norm) 0.0))
;;-------------------------------------------------------------------------------------
(defun Alg3x3_Punto_Su_Asse (A B C dir / vx vy den tval)
  (mapcar 'set '(vx vy) dir)
  (setq den (+ (* A (Math_Sqr vx)) (* B vx vy) (* C (Math_Sqr vy))))
  (if (> (abs den) 1e-50)
      (progn (setq tval (sqrt (abs (/ 1.0 den))))
             (list (* tval vx) (* tval vy) 0.0))))
;;-------------------------------------------------------------------------------------
(defun Alg4x4_Rimuovi_Colonna (row c)
  (cond ((= c 0) (list (cadr row) (caddr row) (cadddr row)))
        ((= c 1) (list (car  row) (caddr row) (cadddr row)))
        ((= c 2) (list (car  row) (cadr  row) (cadddr row)))
        ((= c 3) (list (car  row) (cadr  row) (caddr  row)))))
;;-------------------------------------------------------------------------------------
(defun Alg4x4_Sostituisci_Colonna (M col rhs)
  (mapcar '(lambda (row val)
             (cond  ((= col 0) (list val       (cadr row) (caddr row) (cadddr row)))
                    ((= col 1) (list (car row) val        (caddr row) (cadddr row)))
                    ((= col 2) (list (car row) (cadr row) val         (cadddr row)))
                    ((= col 3) (list (car row) (cadr row) (caddr row) val))))
          M rhs))
;;-------------------------------------------------------------------------------------
(defun Alg4x4_Det (M / row0 m0 m1 m2 m3)
  (setq row0 (car M))
  (setq m0 (list (Alg4x4_Rimuovi_Colonna (nth 1 M) 0) (Alg4x4_Rimuovi_Colonna (nth 2 M) 0)
     (Alg4x4_Rimuovi_Colonna (nth 3 M) 0)))
  (setq m1 (list (Alg4x4_Rimuovi_Colonna (nth 1 M) 1) (Alg4x4_Rimuovi_Colonna (nth 2 M) 1)
     (Alg4x4_Rimuovi_Colonna (nth 3 M) 1)))
  (setq m2 (list (Alg4x4_Rimuovi_Colonna (nth 1 M) 2) (Alg4x4_Rimuovi_Colonna (nth 2 M) 2)
     (Alg4x4_Rimuovi_Colonna (nth 3 M) 2)))
  (setq m3 (list (Alg4x4_Rimuovi_Colonna (nth 1 M) 3) (Alg4x4_Rimuovi_Colonna (nth 2 M) 3)
     (Alg4x4_Rimuovi_Colonna (nth 3 M) 3)))
  (+    (* (nth 0 row0) (Alg3x3_Det m0))
     (- (* (nth 1 row0) (Alg3x3_Det m1)))
        (* (nth 2 row0) (Alg3x3_Det m2))
```

```
     (- (* (nth 3 row0) (Alg3x3_Det m3)))))
;;-------------------------------------------------------------------------------------
(defun Alg4x4_Risolvi (M rhs / det)
  (setq det (Alg4x4_Det M))
  (if (not (equal det 0.0 1e-9))
      (list (/ (Alg4x4_Det (Alg4x4_Sostituisci_Colonna M 0 rhs)) det)
            (/ (Alg4x4_Det (Alg4x4_Sostituisci_Colonna M 1 rhs)) det)
            (/ (Alg4x4_Det (Alg4x4_Sostituisci_Colonna M 2 rhs)) det)
            (/ (Alg4x4_Det (Alg4x4_Sostituisci_Colonna M 3 rhs)) det))))
;;-------------------------------------------------------------------------------------
(defun Calcola_Lagrange (x x1 y1 x2 y2 x3 y3)
  (if (not (or (equal x1 x2 1e-6) (equal x2 x3 1e-6) (equal x1 x3 1e-6)))
      (+ (* y1 (/ (Math_Prod_Diff x x2 x x3) (Math_Prod_Diff x1 x2 x1 x3)))
         (* y2 (/ (Math_Prod_Diff x x1 x x3) (Math_Prod_Diff x2 x1 x2 x3)))
         (* y3 (/ (Math_Prod_Diff x x1 x x2) (Math_Prod_Diff x3 x1 x3 x2))))))
;;-------------------------------------------------------------------------------------
(defun Util_Media_lisNum (lst)        (if lst  (/ (apply '+ lst) (float (length lst)))))
;;-------------------------------------------------------------------------------------
(defun Util_Media_2Num   (num1 num2)  (Util_Media_lisNum (list num1 num2)))
;;-------------------------------------------------------------------------------------
(defun Util_Translate_Pts_Relative (pts_list base_pt)  (mapcar '(lambda (p)
     (Vet_Sub p base_pt))         pts_list))
;;-------------------------------------------------------------------------------------
(defun Util_Translate_Pts_Global   (pts_list base_pt)  (mapcar '(lambda (p)
     (Vet_Sum p base_pt))         pts_list))
;;-------------------------------------------------------------------------------------
(defun Util_Denorm_Pt       (pt_norm G_bary max_dist)
     (Vet_Sum G_bary (Vet_Sca pt_norm max_dist)))
;;-------------------------------------------------------------------------------------
(defun Util_Denorm_List_Pts (pts_list G_bary max_dist)  (mapcar '(lambda (p)
     (Util_Denorm_Pt p G_bary max_dist)) pts_list))
;;-------------------------------------------------------------------------------------
(defun Math_Rot_Pt_2D_CS  (x y co si)
     (list (Math_Diff_Prod x co y si) (Math_Sum_Prod  x si y co) 0.0))
;;-------------------------------------------------------------------------------------
(defun Util_Rot_2D_Dir_CS (Pu1 co si / x y) (mapcar 'set '(x y) Pu1)
     (list (Math_Sum_Prod  x co y si) (Math_Diff_Prod y co x si)))
;;-------------------------------------------------------------------------------------
(defun Util_Rot_2D_Inv_CS (Pu1 co si / x y) (mapcar 'set '(x y) Pu1)
     (list (Math_Diff_Prod x co y si) (Math_Sum_Prod  x si y co)))
;;-------------------------------------------------------------------------------------
(defun Util_Rot_List_2D_CS        (lisPu co si)       (mapcar '(lambda (Pu)
     (Util_Rot_2D_Dir_CS Pu co si)) lisPu))
;;-------------------------------------------------------------------------------------
(defun Util_Rot_List_2D_Inv_CS    (lisPu co si)       (mapcar '(lambda (Pu)
```

```
     (Util_Rot_2D_Inv_CS Pu co si)) lisPu))
;;-------------------------------------------------------------------------------------
(defun Geom_Co_Si_Ang (ang) (list (cos ang) (sin ang)))
;;-------------------------------------------------------------------------------------
(defun Util_Trans_List_Glo_To_Loc (pts origin ang / co si) (mapcar 'set '(co si)
     (Geom_Co_Si_Ang ang))     (mapcar '(lambda (p / v) (setq v (Vet_Sub p origin))
     (Math_Rot_Pt_2D_CS (car v) (cadr v) co si)) pts))
;;-------------------------------------------------------------------------------------
(defun Util_Trans_List_Loc_To_Glo (pts origin ang / co si) (mapcar 'set '(co si)
     (Geom_Co_Si_Ang (- ang))) (mapcar '(lambda (p) (Vet_Sum origin (Math_Rot_Pt_2D_CS (car p)
     (cadr p) co si))) pts))
;;-------------------------------------------------------------------------------------
(defun rotate-pt-origin (pt ang / x y co si)
  (mapcar 'set '(x y) pt)
  (mapcar 'set '(co si) (Geom_Co_Si_Ang ang))
  (Math_Rot_Pt_2D_CS x y co si))
;;-------------------------------------------------------------------------------------
(defun rotate-pt-90 (pt center)    (mapcar '+ center (rotate-pt-origin (mapcar '- pt center)
     (/ pi 2.0))))
;;-------------------------------------------------------------------------------------
(defun Trans_Pt_Glo_To_Loc (pt origin ang) (rotate-pt-origin (mapcar '- pt origin) ang))
;;-------------------------------------------------------------------------------------
(defun Trans_Pt_Loc_To_Glo (pt origin ang) (mapcar '+ origin (rotate-pt-origin pt (- ang))))
;;-------------------------------------------------------------------------------------
(defun Calcola_Parabola (lisPu axis_pair / Pu1 Pu2 Pu3 dir_p1 dir_p2 rot_ang  lisPur x1 x2 x3 y
     1 y2 y3 x4 y4 x5 y5)
  (mapcar 'set '(Pu1 Pu2 Pu3) lisPu)
  (if (not (P_Allin_3Pu Pu1 Pu2 Pu3))
    (progn
      (mapcar 'set '(dir_p1 dir_p2) axis_pair)
      (setq rot_ang  (- (/ pi 2.0) (angle dir_p1 dir_p2)))
      (setq lisPur (Util_Trans_List_Glo_To_Loc lisPu Pu1 rot_ang))
      (mapcar 'set '(x1 x2 x3) (Math_Extract_Coord_Pow lisPur 0 1))
      (mapcar 'set '(y1 y2 y3) (Math_Extract_Coord_Pow lisPur 1 1))
      (setq x4 (Util_Media_2Num x1 x2)
            x5 (Util_Media_2Num x2 x3)
            y4 (Calcola_Lagrange x4 x1 y1 x2 y2 x3 y3)
            y5 (Calcola_Lagrange x5 x1 y1 x2 y2 x3 y3))
      (if (and y4 y5)
        (Util_Trans_List_Loc_To_Glo (list (list x4 y4 0.0) (list x5 y5 0.0)) Pu1 rot_ang)))))
;;-------------------------------------------------------------------------------------
(defun Util_Distanza_Media_lisPu_Da_Cen (pts center)  (/ (apply '+ (mapcar '(lambda (p)
     (distance p center)) pts))      (float (length pts))))
;;-------------------------------------------------------------------------------------
(defun Util_Scala_lisPu_Da_Cen (pts center scale_factor)  (mapcar '(lambda (p)
```

```
     (Vet_Sca (Vet_Sub p center) scale_factor)) pts))
;;-------------------------------------------------------------------------------------
(defun Util_Normalizza_lisPu_Da_Cen (pts center / avg_len)
  (setq avg_len (Util_Distanza_Media_lisPu_Da_Cen pts center))
  (if (> avg_len 1e-12) (list avg_len (Util_Scala_lisPu_Da_Cen pts center (/ 1.0 avg_len)))))
;;-------------------------------------------------------------------------------------
(defun Util_Angle_Diff (a1 a2 / d)  (setq d (abs (- a1 a2)))  (if (> d pi) (- (* 2.0 pi) d) d))
;;-------------------------------------------------------------------------------------
(defun Util_Orthogonality_Score (ang) (abs (sin ang)))
;;-------------------------------------------------------------------------------------
(defun Util_Mean_Angle (angle_list / sum_sin sum_cos n)
  (setq sum_sin 0.0 sum_cos 0.0 n (float (length angle_list)))
  (foreach a angle_list
    (setq sum_sin (+ sum_sin (sin (* a 2.0)))
          sum_cos (+ sum_cos (cos (* a 2.0)))))
  (/ (atan sum_sin sum_cos) 2.0))
;;-------------------------------------------------------------------------------------
(defun Util_Filter_Outliers_By_Centroid (data_list center_extractor_func tolerance_factor / cen
     ters G dists avg_dist valid_data)
  (setq centers  (mapcar center_extractor_func data_list)
        G        (Pu_Baricentro2D_LisPu centers)
        dists    (mapcar '(lambda (p) (distance p G)) centers)
        avg_dist (/ (apply '+ dists) (float (length dists))))
  (if (zerop avg_dist) (setq avg_dist 1.0))
  (setq valid_data nil)
  (mapcar '(lambda (item d) (if (<= d (* avg_dist tolerance_factor))
     (setq valid_data (cons item valid_data)))) data_list dists)
  (reverse valid_data))
;;-------------------------------------------------------------------------------------
(defun Util_Extract_Val (func item)  (apply func (list item)))
;;-------------------------------------------------------------------------------------
(defun Util_Sort_By_Orthogonality (data_list angle_extractor)
  (vl-sort data_list '(lambda (a b)
     (> (Util_Orthogonality_Score (Util_Extract_Val angle_extractor a))
                                      (Util_Orthogonality_Score (Util_Extract_Val angle_extrac
     tor b))))))
;;-------------------------------------------------------------------------------------
(defun Util_Sort_By_Angle_Deviation (data_list angle_extractor target_angle)
  (vl-sort data_list '(lambda (a b) (< (Util_Angle_Diff (Util_Extract_Val angle_extractor a)
     target_angle)
                                      (Util_Angle_Diff (Util_Extract_Val angle_extractor b)
     target_angle)))))
;;-------------------------------------------------------------------------------------
(defun analyze-conic-geometry (pts / pascal_data)
  (setq pascal_data (get-pascal-center-frame pts))
```

```
  (if pascal_data (princ (strcat "\n[ANALYZE] Tipo confermato: " (vl-symbol-name (car pascal_da
     ta)))))
  pascal_data)
;;-------------------------------------------------------------------------------------------
(defun inters_ver (p1 p2 p3 p4 fuzz)
  (if (not fuzz) (setq fuzz 1e-4))
  (if (and (> (distance p1 p2) fuzz) (> (distance p3 p4) fuzz))
      (inters p1 p2 p3 p4 nil)))
;;-------------------------------------------------------------------------------------------
(defun inters_check_tristate (p1 p2 p3 p4 fuzz / int)
  (if (not fuzz) (setq fuzz 1e-4))
  (if (or (< (distance p1 p2) fuzz)
          (< (distance p3 p4) fuzz))
      (list 'INVALID nil nil)
      (progn  (setq int (inters p1 p2 p3 p4 nil))
              (if int   (list 'INTERSECT int (abs (- (angle p1 p2) (angle p3 p4))))
                        (list 'PARALLEL nil  (rem    (angle p1 p2) pi))))))
;;-------------------------------------------------------------------------------------------
(defun Pascal_Genera_Sesto_Punto (pts / p1 p2 p3 p4 p5 K ang34 L line_dir p6)
  (mapcar 'set '(p1 p2 p3 p4 p5) pts)
  (if (setq K (inters_ver p1 p2 p4 p5 1e-4))
      (progn
        (setq ang34 (angle p3 p4)
              L     (inters_ver K (polar K ang34 10.0) p2 p3 1e-4))
        (if L
            (progn
              (setq line_dir (polar p1 ang34 10.0)
                    p6       (inters_ver L p5 p1 line_dir 1e-4))
              (if p6 (append pts (list p6))))))))
;;-------------------------------------------------------------------------------------------
(defun Pascal_Calcola_Singola_Corda (pA pB pC pD pE ang_dir / K dir M L)
  (setq K (inters_ver pA pB pD pE 1e-4)
        dir (polar pA ang_dir 10.0)
        M (inters_ver pC pD pA dir 1e-4)
        L (if (and K M) (inters_ver K M pB pC 1e-4)))
  (if L (inters_ver L pE pA dir 1e-4)))
;;-------------------------------------------------------------------------------------------
(defun Pascal_Intersezione_Diametri (diam1_pts chords_pts pts_base / p1 p2 p3 p4 p5 p7 p8 p9 p1
     0 p11 p12 check_res status center angle_ref)
  (mapcar 'set '(p7 p8) diam1_pts)
  (mapcar 'set '(p9 p10) chords_pts)
  (mapcar 'set '(p1 p2 p3 p4 p5) pts_base)
  (setq p11 (Math_PuntoMedio_2Pu p1 p9)
        p12 (Math_PuntoMedio_2Pu p4 p10))
  (setq check_res (inters_check_tristate p7 p8 p11 p12 1e-4))
```

```
  (mapcar 'set '(status center angle_ref) check_res)
  (list status center angle_ref (list center p7 p8 p11 p12)))
;;-------------------------------------------------------------------------------------
(defun pascal-core-attempt (curr_pts / pts_6 p7 p8 ang78 chord_pts diam_res status center angle
     _ref geom_data tipo)
  (setq pts_6 (Pascal_Genera_Sesto_Punto curr_pts))
  (if pts_6
      (progn
        (mapcar 'set '(p7 p8 ang78) (Pascal_Calcola_Primo_Diametro pts_6))
        (if p7
            (progn
              (setq chord_pts (Pascal_Calcola_Corde_Coniugate curr_pts ang78))
              (if chord_pts
                  (progn
                    (setq diam_res (Pascal_Intersezione_Diametri (list p7 p8)
     chord_pts curr_pts))
                    (mapcar 'set '(status center angle_ref geom_data) diam_res)
                    (cond
                      ((eq status 'INVALID)  (setq tipo 'NIL))
                      ((eq status 'PARALLEL) (setq tipo 'PARABOLA))
                      ((eq status 'INTERSECT)
                       (if (P_center-infinite curr_pts center 5000.0)
                           (setq tipo 'PARABOLA status 'PARALLEL angle_ref ang78)
                           (setq tipo (probe-conic-algebraic-tipo curr_pts center (cadr geom_da
     ta) (cadddr geom_data))))))
                    (list status center angle_ref tipo geom_data))
                  (list 'PARALLEL nil ang78 'PARABOLA (list nil p7 p8 nil nil))))
            (list 'FAIL_DIAM1 nil nil 'NIL nil)))
       (list 'FAIL_P6 nil nil 'NIL nil)))
;;-------------------------------------------------------------------------------------
(defun Util_Reset_Vars (sym_list)  (foreach sym sym_list (set sym nil)))
;;-------------------------------------------------------------------------------------
(defun Util_Filter_By_Nth (lst n target)  (vl-remove-if-not '(lambda (item) (eq (nth n item)
     target)) lst))
;;-------------------------------------------------------------------------------------
(defun Util_Filter_And_Count_By_Nth (lst n target / subset)
     (setq subset (Util_Filter_By_Nth lst n target))  (list subset (length subset)))
;;-------------------------------------------------------------------------------------
(defun Log_Pascal_Iteration (counter res_temp / status center ang tipo partial_geom center_geom
     p7 p8 p11 p12)
  (Princ_1Sym (strcat "\n[LOG] #" (itoa counter) " ") (car res_temp))
  (cond
    ((eq (car res_temp) 'FAIL_P6)    (princ " -> Fallimento Costruzione P6 (Steiner)"))
    ((eq (car res_temp) 'FAIL_DIAM1) (princ " -> Fallimento Primo Diametro"))
    ((eq (car res_temp) 'INVALID)    (princ " -> Punti coincidenti"))
```

```
    (T
     (Util_Reset_Vars '(status center ang tipo partial_geom center_geom p7 p8 p11 p12))
     (mapcar 'set '(status center ang tipo partial_geom) res_temp)
     (mapcar 'set '(center_geom p7 p8 p11 p12) partial_geom)
     (Princ_1Num " | Ang: " ang)
     (Princ_1Sym " | Tipo: " tipo)
     (Princ_1Pun " | C: " center)
     (Princ_2Pun " | D1: " p7 p8)
     (Princ_2Pun " | D2: " p11 p12))))
;;-------------------------------------------------------------------------------------
(defun get-pascal-center-frame (pts / idx_list res_temp logbook lis_finite lis_infiniti lis_par
     abola lis_ellisse lis_iperbole lis_fail_p6 lis_fail_diam lis_invalid winner_data G_bary pt
     s_loc final_res counter win_type num_finite num_infiniti num_parabola num_ellisse num_iper
     bole num_fail_p6 num_fail_diam num_inv angles mean_dir deviation G dist_from_G)
  (setq logbook nil
        counter 0
        G_bary (Pu_Baricentro2D_LisPu pts)
        pts_loc (mapcar '(lambda (p) (mapcar '- p G_bary)) pts))
  (princ "\n--- ANALISI DIAGNOSTICA (Space Locale) ---")
  (foreach idx_list (get-all-permutations '(0 1 2 3 4))
    (setq res_temp (pascal-core-attempt (Util_Reorder_List_By_Indices pts_loc idx_list))
          logbook (cons res_temp logbook)))
  (princ "\n--- FINE RACCOLTA ---")
  (mapcar 'set '(lis_finite    num_finite)
     (Util_Filter_And_Count_By_Nth logbook 0 'INTERSECT))
  (mapcar 'set '(lis_infiniti  num_infiniti)
     (Util_Filter_And_Count_By_Nth logbook 0 'PARALLEL))
  (mapcar 'set '(lis_fail_p6   num_fail_p6)
     (Util_Filter_And_Count_By_Nth logbook 0 'FAIL_P6))
  (mapcar 'set '(lis_fail_diam num_fail_diam)
     (Util_Filter_And_Count_By_Nth logbook 0 'FAIL_DIAM1))
  (mapcar 'set '(lis_invalid   num_inv)
     (Util_Filter_And_Count_By_Nth logbook 0 'INVALID))
  (mapcar 'set '(lis_ellisse   num_ellisse)
     (Util_Filter_And_Count_By_Nth logbook 3 'ELLISSE))
  (mapcar 'set '(lis_iperbole  num_iperbole)
     (Util_Filter_And_Count_By_Nth logbook 3 'IPERBOLE))
  (mapcar 'set '(lis_parabola  num_parabola)
     (Util_Filter_And_Count_By_Nth logbook 3 'PARABOLA))
  (princ "\n--- STATISTICHE VOTO ---")
  (Princ_1Int "\n[STAT] Totale Centri Finiti  : " num_finite)
  (Princ_1Int "\n[STAT] -> Voti Ellisse       : " num_ellisse)
  (Princ_1Int "\n[STAT] -> Voti Iperbole      : " num_iperbole)
  (Princ_1Int "\n[STAT] Totale Centri Infiniti: " num_infiniti)
  (Princ_1Int "\n[STAT] -> Voti Parabola      : " num_parabola)
```

```
  (Princ_1Int "\n[STAT] Totale Errori e Scarti: " (+ num_fail_p6 num_fail_diam num_inv))
  (Princ_1Int "\n[STAT] -> Fail P6 (Steiner)  : " num_fail_p6)
  (Princ_1Int "\n[STAT] -> Fail Diametro 1    : " num_fail_diam)
  (Princ_1Int "\n[STAT] -> Punti Coincidenti  : " num_inv)

  (cond
    ((and (> num_finite 0) (>= num_finite num_parabola))
     (princ "\n[DECISIONE] Vince Percorso Conica a Centro.")
     (if (>= num_ellisse num_iperbole)
         (setq win_type 'ELLISSE lis_finite lis_ellisse)
         (setq win_type 'IPERBOLE lis_finite lis_iperbole))
     (Princ_1Sym "\n[SOTTOTIPO] Vince " win_type)
     (Princ_1Int "\nCandidati totali grezzi: " (length lis_finite))
     (setq lis_finite (Util_Filter_Outliers_By_Centroid lis_finite 'cadr 2.5)
           lis_finite (Util_Sort_By_Orthogonality lis_finite 'caddr)
           winner_data (car lis_finite))
     (Princ_1Int "\nCandidati totali filtrati: " (length lis_finite))
     (if (> (length lis_finite) 1)
         (progn
           (setq G (Pu_Baricentro2D_LisPu (mapcar 'cadr lis_finite))
                 dist_from_G (distance (cadr winner_data) G))
           (Princ_1Num "\n[AFFIDABILITA] Dist. dal Baricentro: " dist_from_G))
         (princ "\n[AFFIDABILITA] Campione unico."))
     (Princ_1Num " | Score Ortogonalità: " (Util_Orthogonality_Score (caddr winner_data)))
     (setq final_res (cons win_type (nth 4 winner_data))))
    ((> num_parabola 0)
     (princ "\n[DECISIONE] Vince Percorso Parabola (Maggioranza Parallela).")
     (setq angles (mapcar 'caddr lis_parabola)
           mean_dir (Util_Mean_Angle angles)
           lis_parabola (Util_Sort_By_Angle_Deviation lis_parabola 'caddr mean_dir)
           winner_data (car lis_parabola))
     (Princ_1Num "\n[AFFIDABILITA] Deviazione: " (Util_Angle_Diff (caddr winner_data)
     mean_dir))
     (setq final_res (cons 'PARABOLA (nth 4 winner_data))))
    (T
     (princ "\n[ERRORE] Nessuna geometria valida determinabile.")
     (setq final_res nil)))
  (if final_res (cons (car final_res) (mapcar '(lambda (pt) (if pt (mapcar '+ pt G_bary) nil))
     (cdr final_res)))))
;;-------------------------------------------------------------------------------------
(defun Math_Calc_Par_Error_4th_Pt (Ang pts / P_rot x1 x2 x3 x4 y1 y2 y3 y4 sq_x1 sq_x2 sq_x3 sq
     _x4 Det a b c_coeff)
  (setq P_rot (Util_Rot_List_2D_CS pts (cos Ang) (sin Ang)))
  (mapcar 'set '(x1 x2 x3 x4)             (Math_Extract_Coord_Pow P_rot 0 1))
  (mapcar 'set '(y1 y2 y3 y4)             (Math_Extract_Coord_Pow P_rot 1 1))
```

```
  (mapcar 'set '(sq_x1 sq_x2 sq_x3 sq_x4) (Math_Extract_Coord_Pow P_rot 0 2))
  (setq Det (Math_Cyclic_Det sq_x1 sq_x2 sq_x3 x1 x2 x3))
  (if (< (abs Det) 1e-6)
      1.0e9
      (progn
        (setq a (/ (Math_Cyclic_Det y1 y2 y3 x1 x2 x3) Det)
              b (/ (Math_Cyclic_Det sq_x1 sq_x2 sq_x3 y1 y2 y3) Det)
              c_coeff (/ (+ (* sq_x1 (Math_Diff_Prod x2 y3 x3 y2))
                            (* sq_x2 (Math_Diff_Prod x3 y1 x1 y3))
                            (* sq_x3 (Math_Diff_Prod x1 y2 x2 y1))) Det))
        (- (+ (Math_Sum_Prod a (Math_Sqr x4) b x4) c_coeff) y4))))
;;-------------------------------------------------------------------------------------
(defun Util_List_Rotations (lst / res)  (repeat (length lst) (setq res (cons lst res)
     lst (append (cdr lst) (list (car lst)))))  (reverse res))
;;-------------------------------------------------------------------------------------
(defun P_Are_Vectors_Parallel (v1 v2 / cp_mag len_prod)
  (setq cp_mag (distance '(0 0 0) (Vet_Cro v1 v2))
        len_prod (* (distance '(0 0 0) v1) (distance '(0 0 0) v2)))
  (or (< len_prod 1e-12)
      (< (/ cp_mag len_prod) 1e-4)))
;;-------------------------------------------------------------------------------------
(defun P_Is_Config_Concave (pts)  (and (= (length pts) 4) (apply 'P_Conca_4Pu pts)))
;;-------------------------------------------------------------------------------------
(defun Geom_Count_Parallel_Pairs (pts)  (or (apply 'Math_Conta_Paral_4Pu pts) 0))
;;-------------------------------------------------------------------------------------
(defun P_Are_Collinear (Pu1 Pu2 Pu3)  (P_Allin_3Pu Pu1 Pu2 Pu3))
;;-------------------------------------------------------------------------------------
(defun P_Has_Collinear_Triplet (pts)  (apply 'P_Allin_3su4Pu (append pts (list nil))))
;;-------------------------------------------------------------------------------------
(defun P_Is_Orthocentric_System (pts)  (apply 'P_3Orto_4Pu pts))
;;-------------------------------------------------------------------------------------
(defun Geom_Analyze_Ipe_Topology (pts)
  (cond ((P_Has_Collinear_Triplet pts)
     (list 'DEGENERATE 0 "3 punti sono allineati. Iperbole degenere o impossibile."))
        ((P_Is_Orthocentric_System pts)
     (list 'ORTHOCENTRIC -1 "Sistema Ortocentrico rilevato. Infinite Iperboli Equilatere possib
     ili."))
        (T (list 'UNIQUE 1 "Configurazione valida. Esiste 1 Iperbole Equilatera unica."))))
;;-------------------------------------------------------------------------------------
(defun Math_Line_General_Eq (Pu1 Pu2 / a b c)
  (setq a (- (cadr Pu1) (cadr Pu2))
        b (- (car Pu2) (car Pu1))
        c (Math_Diff_Prod (car Pu1) (cadr Pu2) (car Pu2) (cadr Pu1)))
  (list a b c))
;;-------------------------------------------------------------------------------------
```

```
(defun Math_Multiply_Lines_To_Conic (L1 L2 / a1 b1 c1 a2 b2 c2)
  (mapcar 'set '(a1 b1 c1 a2 b2 c2) (append L1 L2))
  (list (* a1 a2) (Math_Sum_Prod a1 b2 a2 b1) (* b1 b2) (Math_Sum_Prod a1 c2 a2 c1)
     (Math_Sum_Prod b1 c2 b2 c1) (* c1 c2)))
;;-------------------------------------------------------------------------------------
(defun Math_Combine_Conics (K1 K2 lam)  (mapcar '+ K1 (mapcar '(lambda (x) (* x lam)) K2)))
;;-------------------------------------------------------------------------------------
(defun Math_Extract_Equilateral_Data (Coeffs / A B C D E F det Cx Cy PuC F_prime lambda1 a_semi
     axis_ang)
  (mapcar 'set '(A B C D E F) Coeffs)
  (setq det (- (* 4 A C) (Math_Sqr B)))
  (if (> (abs det) 1e-9)
      (progn
        (setq Cx (/ (Math_Diff_Prod B E (* 2 C) D) det)
              Cy (/ (Math_Diff_Prod B D (* 2 A) E) det)
              PuC (list Cx Cy 0.0)
              F_prime (+ (/ (Math_Sum_Prod D Cx E Cy) 2.0) F)
              lambda1 (sqrt (+ (Math_Sqr A) (/ (Math_Sqr B) 4.0)))
              a_semi (sqrt (abs (/ F_prime lambda1)))
              axis_ang (if (> (/ (- F_prime) lambda1) 0)
                           (* 0.5 (atan B (- A C)))
                           (+ (* 0.5 (atan B (- A C))) (/ pi 2.0))))
        (list PuC (list (cos axis_ang) (sin axis_ang) 0.0) a_semi))
      nil))
;;-------------------------------------------------------------------------------------
(defun Math_Get_Coeffs_4P_AxisDir (Pu1 Pu2 Pu3 Pu4 PuD1 PuD2 / norm_data PuC max_dist P1n P2n P
     3n P4n K1 K2 ang_axis sin2 cos2 A1 B1 C1 A2 B2 C2 num den lam K_final delta found_type)
  (setq norm_data (Util_Normalize_Pts_Set (list Pu1 Pu2 Pu3 Pu4))
        PuC (car norm_data)
        max_dist (cadr norm_data))
  (mapcar 'set '(P1n P2n P3n P4n) (caddr norm_data))
  (setq K1 (Math_Multiply_Lines_To_Conic (Math_Line_General_Eq P1n P2n)
     (Math_Line_General_Eq P3n P4n))
        K2 (Math_Multiply_Lines_To_Conic (Math_Line_General_Eq P1n P3n)
     (Math_Line_General_Eq P2n P4n))
        ang_axis (angle PuD1 PuD2)
        sin2 (sin (* 2.0 ang_axis))
        cos2 (cos (* 2.0 ang_axis)))
  (mapcar 'set '(A1 B1 C1) K1)
  (mapcar 'set '(A2 B2 C2) K2)
  (setq num (Math_Diff_Prod B1 cos2 (- A1 C1) sin2)
        den (Math_Diff_Prod (- A2 C2) sin2 B2 cos2))
  (if (< (abs den) 1e-9)
      nil
      (progn
```

```
        (setq lam (/ num den)
              K_final (Math_Combine_Conics K1 K2 lam)
              delta (- (Math_Sqr (nth 1 K_final)) (* 4.0 (nth 0 K_final) (nth 2 K_final))))
        (cond ((< delta -1e-9)  (setq found_type 'ELLISSE))
              ((> delta 1e-9)   (setq found_type 'IPERBOLE))
              (T                (setq found_type 'PARABOLA)))
        (list K_final PuC max_dist found_type))))
;;-------------------------------------------------------------------------------------
(defun Solve_And_Draw_From_Coeffs (Coeffs PuC_Offset lay_crv lay_constr conic_type scale_factor
     / scale_v A B C D E F det Cx Cy PuC F_prime trace_v discr lambda1 lambda2 a_raw b_raw V_E1
     V_E2 V_Maj V_Min a_fin b_fin c_foc Ax1 Ax2 Ax3 Ax4 F1 F2 V1)
  (setq scale_v (if scale_factor scale_factor 1.0))
  (mapcar 'set '(A B C D E F) Coeffs)
  (setq det (- (* 4 A C) (Math_Sqr B)))
  (if (< (abs det) 1e-9)
      (princ "\n[ERRORE] Conica degenere (Determinante nullo).")
      (progn
        (setq Cx (/ (Math_Diff_Prod B E (* 2 C) D) det)
              Cy (/ (Math_Diff_Prod B D (* 2 A) E) det)
              PuC (Vet_Sum (Vet_Sca (list Cx Cy 0.0) scale_v) PuC_Offset)
              F_prime (+ (/ (Math_Sum_Prod D Cx E Cy) 2.0) F))
        (if (> (abs F_prime) 1e-9)
            (progn
              (setq trace_v (+ A C)
                    discr (sqrt (Math_Sum_Squares (- A C) B))
                    lambda1 (/ (+ trace_v discr) 2.0)
                    lambda2 (/ (- trace_v discr) 2.0)
                    a_raw (sqrt (abs (/ F_prime lambda1)))
                    b_raw (sqrt (abs (/ F_prime lambda2)))
                    V_E1 (Alg3x3_Autovettore A B C lambda1)
                    V_E2 (list (- (cadr V_E1)) (car V_E1) 0.0))
              (cond
                ((eq conic_type 'ELLISSE)
                 (if (> a_raw b_raw)
                     (setq a_fin a_raw V_Maj V_E1 b_fin b_raw V_Min V_E2)
                     (setq a_fin b_raw V_Maj V_E2 b_fin a_raw V_Min V_E1))
                 (setq a_fin (* a_fin scale_v)
                       b_fin (* b_fin scale_v))

                 (setq c_foc (if (setq tmp_c (Pitag_Ca a_fin b_fin)) tmp_c 0.0))

                 (mapcar 'set '(Ax1 Ax2) (Geom_2Pu_Simm_PuC PuC V_Maj a_fin))
                 (mapcar 'set '(Ax3 Ax4) (Geom_2Pu_Simm_PuC PuC V_Min b_fin))
                 (mapcar 'set '(F1 F2)   (Geom_2Pu_Simm_PuC PuC V_Maj c_foc))
```

```
                (Draw_ConEll_CeAx_Foci PuC Ax1 Ax2 Ax3 Ax4 F1 F2 (/ b_fin a_fin)))
               ((eq conic_type 'IPERBOLE)
                (if (> (/ (- F_prime) lambda1) 0)
                    (setq a_fin a_raw V_Maj V_E1 b_fin b_raw)
                    (setq a_fin b_raw V_Maj V_E2 b_fin a_raw))
                (setq a_fin (* a_fin scale_v)
                      b_fin (* b_fin scale_v)
                      c_foc (Pitag_Ip a_fin b_fin)
                      V1 (Vet_Sum PuC (Vet_Sca V_Maj a_fin))
                      F1 (Vet_Sum PuC (Vet_Sca V_Maj c_foc)))

                (Draw_ConIpe_CeVeFu_Compl PuC V1 F1 nil))))
            (princ "\n[ERRORE] Conica degenere (F' nullo).")))))
;;-------------------------------------------------------------------------------------
(defun Util_Trans_Pts_List (pts from_CS to_CS)  (mapcar '(lambda (p) (trans p from_CS to_CS))
     pts))
;;-------------------------------------------------------------------------------------
(defun Geom_Find_Parabolic_Angles (Pu1 Pu2 Pu3 Pu4 / V_Norm pts dx dy ang_pca c_pre s_pre pts_a
     ligned Step Ang prev_err curr_err min_a max_a mid_a mid_err Solutions is_dup)
  (setq V_Norm (Vet_Uni (Vet_Cro (Vet_Sub Pu2 Pu1) (Vet_Sub Pu3 Pu1)))
        pts (caddr (Util_Normalize_Pts_Set (Util_Trans_Pts_List (list Pu1 Pu2 Pu3 Pu4)
     0 V_Norm))))
  (mapcar 'set '(dx dy) (Util_Get_Pts_BBox_Size pts))
  (setq ang_pca (if (> dy dx) (/ pi 2.0) (angle '(0 0) (Vet_Sub (nth 2 pts) (nth 0 pts))))
        c_pre (cos ang_pca)
        s_pre (sin ang_pca)
        pts_aligned (Util_Rot_List_2D_CS pts c_pre s_pre)
        Step 2.0
        Ang 0.0
        prev_err (Math_Calc_Par_Error_4th_Pt 0.0 pts_aligned)
        Solutions nil)
  (repeat 180
    (setq Ang (+ Ang Step)
          curr_err (Math_Calc_Par_Error_4th_Pt (* Ang (/ pi 180.0)) pts_aligned))
    (if (< (* prev_err curr_err) 0)
        (progn
          (setq min_a (- Ang Step)
                max_a Ang)
          (repeat 20
            (setq mid_a (* 0.5 (+ min_a max_a))
                  mid_err (Math_Calc_Par_Error_4th_Pt (* mid_a (/ pi 180.0)) pts_aligned))
            (if (< (* mid_err prev_err) 0) (setq max_a mid_a) (setq min_a mid_a)))
          (if (< (abs mid_err) 1e-2)
              (progn
                (setq is_dup nil)
```

```
                (foreach s Solutions (if (< (abs (- mid_a s)) 1.0) (setq is_dup T)))
                (if (not is_dup) (setq Solutions (cons mid_a Solutions)))))))
    (setq prev_err curr_err))
  (vl-sort Solutions '<))
;;-------------------------------------------------------------------------------------------
(defun Util_Scale_Pts_From_Centroid (pts factor / PuC)  (setq PuC (Pu_Baricentro2D_LisPu pts))
     (mapcar '(lambda (p) (Vet_Sum PuC (Vet_Sca (Vet_Sub p PuC) factor))) pts))
;;-------------------------------------------------------------------------------------------
(defun Math_Dist_Point_Line (Pu1 Pu2 Pu3 / x1 y1 x2 y2 x3 y3 num den)
  (mapcar 'set '(x1 y1) Pu1)
  (mapcar 'set '(x2 y2) Pu2)
  (mapcar 'set '(x3 y3) Pu3)
  (setq num (abs (+ (* x3 (- y2 y1))
                    (* x2 (- y1 y3))
                    (* x1 (- y3 y2))))
        den (distance Pu2 Pu3))
  (if (> den 1e-9) (/ num den) 0.0))
;;-------------------------------------------------------------------------------------------
(defun Geom_Check_Trapezoid_Direction (pts / P1 P2 P3 P4)
  (mapcar 'set '(P1 P2 P3 P4) pts)
  (cond  ((P_Are_Vectors_Parallel (Vet_Sub P2 P1) (Vet_Sub P3 P4)) (Vet_Sub P2 P1))
         ((P_Are_Vectors_Parallel (Vet_Sub P4 P1) (Vet_Sub P3 P2)) (Vet_Sub P4 P1))))
;;-------------------------------------------------------------------------------------------
(defun P_Are_Vectors_Parallel_2D (v1 v2 tol / x1 y1 x2 y2)
  (mapcar 'set '(x1 y1) v1)
  (mapcar 'set '(x2 y2) v2)
  (< (abs (Math_Diff_Prod x1 y2 x2 y1)) tol))
;;-------------------------------------------------------------------------------------------
(defun Math_Solve_Par_Coeffs (Pu1 Pu2 Pu3 / x1 y1 x2 y2 x3 y3 sx1 sx2 sx3 denom a b c)
  (mapcar 'set '(x1 y1) Pu1)
  (mapcar 'set '(x2 y2) Pu2)
  (mapcar 'set '(x3 y3) Pu3)

  (setq sx1 (Math_Sqr x1)
        sx2 (Math_Sqr x2)
        sx3 (Math_Sqr x3))

  (setq denom (Math_Cyclic_Det sx1 sx2 sx3 x1 x2 x3))

  (if (not (equal denom 0.0 1e-9))
      (progn
        (setq a (/ (Math_Cyclic_Det y1  y2  y3  x1 x2 x3) denom)
              b (/ (Math_Cyclic_Det sx1 sx2 sx3 y1 y2 y3) denom)
              c (/ (+ (* sx1 (Math_Diff_Prod x2 y3 x3 y2))
                      (* sx2 (Math_Diff_Prod x3 y1 x1 y3))
```

```
                    (* sx3 (Math_Diff_Prod x1 y2 x2 y1))) denom))
        (list a b c))))
;;-------------------------------------------------------------------------------------------
(defun Math_Solve_Centered_Conic_3P_Ratio (pts ratio conic_type / Pu1 Pu2 Pu3 x1 y1 x2 y2 x3 y3
     W M rhs sol D_prime E_prime F_prime Cx Cy a_sq)
  (mapcar 'set '(Pu1 Pu2 Pu3) pts)
  (mapcar 'set '(x1 y1)       Pu1)
  (mapcar 'set '(x2 y2)       Pu2)
  (mapcar 'set '(x3 y3)       Pu3)
  (setq W (cond ((eq conic_type 'ELLISSE)     (/ 1.0 (Math_Sqr ratio)))
                ((eq conic_type 'IPERBOLE) (- (/ 1.0 (Math_Sqr ratio))))))
  (if W
      (progn (setq M   (list (list x1 y1 1.0) (list x2 y2 1.0) (list x3 y3 1.0))
                   rhs (list (- (Math_Sum_Prod x1 x1 y1 (* y1 W)))
                             (- (Math_Sum_Prod x2 x2 y2 (* y2 W)))
                             (- (Math_Sum_Prod x3 x3 y3 (* y3 W))))
                   sol (Alg3x3_Risolvi M rhs))
             (if sol
                (progn  (mapcar 'set '(D_prime E_prime F_prime) sol)
                         (setq Cx   (/ D_prime -2.0)
                               Cy   (/ E_prime (* -2.0 W))
                               a_sq (- (Math_Sum_Prod Cx Cx Cy (* Cy W)) F_prime))
                         (if (> a_sq 1e-6)
                             (list (list Cx Cy 0.0) (sqrt a_sq))))))))
;;-------------------------------------------------------------------------------------------
(defun Math_Solve_Ell_3P_Ratio (pts ratio)
     (Math_Solve_Centered_Conic_3P_Ratio pts ratio 'ELLISSE))
;;-------------------------------------------------------------------------------------------
(defun Math_Solve_Ipe_3P_Ratio (pts ratio)
     (Math_Solve_Centered_Conic_3P_Ratio pts ratio 'IPERBOLE))
;;-------------------------------------------------------------------------------------------
(defun Draw_Curve_Segment_Spline (pts_list lay / pts_num)
  (setq pts_num (length pts_list))
  (cond ((> pts_num 2) (entmakex (append  (list '(0 . "SPLINE") '(100 . "AcDbEntity")
     (cons 8 lay)
                                             '(100 . "AcDbSpline") '(71 . 3) '(70 . 0)
     (cons 74 pts_num))
                                       (mapcar '(lambda (pt) (cons 11 pt)) pts_list))))
        ((= pts_num 2) (entmakex          (list '(0 . "LINE")   '(100 . "AcDbEntity")
     (cons 8 lay)
                                       (cons 10 (car pts_list))
     (cons 11 (cadr pts_list)))))))
;;-------------------------------------------------------------------------------------------
(defun Geom_Trans_Pt (Pu1 ang PuC / x1 y1 xc yc co si)
  (mapcar 'set '(x1 y1) Pu1)
```

```
  (mapcar 'set '(xc yc) PuC)
  (mapcar 'set '(co si) (list (cos ang) (sin ang)))
  (list (+ xc (Math_Diff_Prod x1 co y1 si))
        (+ yc (Math_Sum_Prod  x1 si y1 co))
        0.0))
;;-------------------------------------------------------------------------------------
(defun Math_Rotate_Pt_Origin (Pu1 ang)  (Geom_Trans_Pt Pu1 ang '(0.0 0.0 0.0)))
;;-------------------------------------------------------------------------------------
(defun Util_Seg_Crosses_Triangle (Pu1 Pu2 tri_pts / Pu3 Pu4 Pu5)
  (mapcar 'set '(Pu3 Pu4 Pu5) tri_pts)
  (or (inters Pu1 Pu2 Pu3 Pu4 T)
      (inters Pu1 Pu2 Pu4 Pu5 T)
      (inters Pu1 Pu2 Pu5 Pu3 T)))
;;-------------------------------------------------------------------------------------
(defun Util_Scan_And_Cut_Branches (pt_list dist_limit pts_tri flag_no_cross / branches curr_bra
     nch last_pt cut_now)
  (setq branches nil curr_branch nil last_pt nil)
  (foreach pt pt_list
    (setq cut_now nil)
    (if last_pt
        (progn
          (if (> (distance pt last_pt) dist_limit) (setq cut_now T))
          (if (and flag_no_cross pts_tri (not cut_now))
              (if (Util_Seg_Crosses_Triangle last_pt pt pts_tri) (setq cut_now T)))))
    (if cut_now
        (progn
          (if curr_branch (setq branches (cons (reverse curr_branch) branches)))
          (setq curr_branch nil)))
    (setq curr_branch (cons pt curr_branch)
          last_pt        pt))
  (if curr_branch (setq branches (cons (reverse curr_branch) branches)))
  (reverse branches))
;;-------------------------------------------------------------------------------------
(defun Util_Stitch_Branches_HeadTail (branches / seg_start seg_end p_first p_second p_last p_pr
     ev d_closing d_start d_end d_ref factor merged_branch middle_branches)
  (if (>= (length branches) 2)
      (progn
        (setq seg_start (car branches)
              seg_end   (last branches))
        (if (and (>= (length seg_start) 2) (>= (length seg_end) 2))
            (progn
              (setq p_first   (nth 0 seg_start)
                    p_second  (nth 1 seg_start)
                    p_last    (last seg_end)
                    p_prev    (cadr (reverse seg_end))
```

```
                    d_closing (distance p_last p_first)
                    d_start   (distance p_first p_second)
                    d_end     (distance p_prev p_last)
                    d_ref     (max d_start d_end)
                    factor    3.0)
               (if (< d_closing (* d_ref factor))
                   (progn
                    (princ (strcat "\n[INFO] Saldatura Smart: Gap (" (rtos d_closing 2 2) ")
     unito."))
                    (setq merged_branch (append seg_end seg_start))
                    (if (= (length branches) 2)
                        (setq branches (list merged_branch))
                        (setq middle_branches (cdr branches)
                              middle_branches (reverse (cdr (reverse middle_branches)))
                              branches        (cons merged_branch middle_branches)))))))))
  branches)
;;-------------------------------------------------------------------------------------------
(defun Util_Process_Branches_Universal (pt_list dist_limit pts_tri flag_smart_stitch flag_no_cr
     oss / branches)
  (setq branches (Util_Scan_And_Cut_Branches pt_list dist_limit pts_tri flag_no_cross))
  (if flag_smart_stitch (setq branches (Util_Stitch_Branches_HeadTail branches)))
  branches)
;;-------------------------------------------------------------------------------------------
(defun Util_Normalize_Pts_Set (pts / PuC max_dist scale_f)
  (setq PuC (Pu_Baricentro2D_LisPu pts)
        max_dist (apply 'max (mapcar '(lambda (p) (distance p PuC)) pts)))
  (if (< max_dist 1e-9) (setq max_dist 1.0))
  (setq scale_f (/ 1.0 max_dist))
  (list PuC max_dist (mapcar '(lambda (p) (Vet_Sca (Vet_Sub p PuC) scale_f)) pts)))
;;-------------------------------------------------------------------------------------------
(defun Alg_Tournament_5P_Centered (pts Ce_real conic_type / best_err best_res i j k p1 p2 p3 re
     s_matrix A B C delta_m err_curr pt_test x y val_test)
  (setq best_err 1.0e99
        best_res nil
        i        0)
  (while (< i 3)
    (setq j (1+ i))
    (while (< j 4)
      (setq k (1+ j))
      (while (< k 5)
        (setq p1 (nth i pts) p2 (nth j pts) p3 (nth k pts)
              res_matrix (Alg3x3_Calcola_Coeffs Ce_real p1 p2 p3))
        (if res_matrix
          (progn
            (mapcar 'set '(A B C) res_matrix)
```

```
            (setq delta_m (- (* B B) (* 4.0 A C)))
            (if (or (and (eq conic_type 'ELLISSE) (< delta_m 0)) (and (eq conic_type 'IPERBOLE)
     (> delta_m 0)))
                (progn
                  (setq err_curr 0.0)
                  (foreach idx '(0 1 2 3 4)
                    (if (and (/= idx i) (/= idx j) (/= idx k))
                        (progn
                          (setq pt_test (mapcar '- (nth idx pts) Ce_real))
                          (mapcar 'set '(x y) pt_test)
                          (setq val_test (+ (* A x x) (* B x y) (* C y y))
                                err_curr (+ err_curr (abs (- val_test 1.0)))))))
                  (if (< err_curr best_err)
                      (setq best_err err_curr
                            best_res (list Ce_real p1 p2 p3)))))
        (setq k (1+ k)))
      (setq j (1+ j)))
    (setq i (1+ i)))
  (if (and best_res (< best_err 1.0))
      (progn (princ (strcat "\n[DECISIONE] Residuo: " (rtos best_err 2 16))) best_res)
      nil))
;;-------------------------------------------------------------------------------------------
(defun Util_Conv_List_To_VLA (ents / vla_list)
  (setq vla_list nil)
  (if (listp ents)
      (foreach ent ents
        (if (and (= (type ent) 'ENAME) (member (cdr (assoc 0 (entget ent)))
     '("SPLINE" "ELLIPSE")))
            (setq vla_list (cons (vlax-ename->vla-object ent) vla_list)))))
  (reverse vla_list))
;;-------------------------------------------------------------------------------------------
;;-------------------------------------------------------------------------------------------

;;-------------------------------------------------------------------------------------------
;;-------------------------------------------------------------------------------------------
(progn
;;-------------------------------------------------------------------------------------------
(defun Geom_Sort_Points_CCW (pts / cen)
  (if (and pts (> (length pts) 1))
      (progn (setq cen (Pu_Baricentro2D_LisPu pts))
             (vl-sort pts '(lambda (a b) (Math_Angolo_Minore_CCW cen a b))))
      pts))
;;-------------------------------------------------------------------------------------------
(defun Geom_Ordina_CCW_4Pu (Pu1 Pu2 Pu3 Pu4)  (if (P_0Coinc_4Pu Pu1 Pu2 Pu3 Pu4)
     (Geom_Sort_Points_CCW (list Pu1 Pu2 Pu3 Pu4))))
```

```
;;-------------------------------------------------------------------------------------
(defun Geom_Ordina_Concavo_4Pu (Pu1 Pu2 Pu3 Pu4 / Pu_In Pu_Out_List)
  (if (P_Conca_4Pu Pu1 Pu2 Pu3 Pu4)
    (progn
      (cond ((P_Pu_Inter_3Pu Pu1 Pu2 Pu3 Pu4) (setq Pu_In Pu1 Pu_Out_List (list Pu2 Pu3 Pu4)))
            ((P_Pu_Inter_3Pu Pu2 Pu1 Pu3 Pu4) (setq Pu_In Pu2 Pu_Out_List (list Pu1 Pu3 Pu4)))
            ((P_Pu_Inter_3Pu Pu3 Pu1 Pu2 Pu4) (setq Pu_In Pu3 Pu_Out_List (list Pu1 Pu2 Pu4)))
            ((P_Pu_Inter_3Pu Pu4 Pu1 Pu2 Pu3) (setq Pu_In Pu4 Pu_Out_List (list Pu1 Pu2 Pu3))))
      (append Pu_Out_List (list Pu_In)))))
;;-------------------------------------------------------------------------------------
(defun Math_Uguaglianza_Relativa_2Re (Pu1 Pu2 Pu3 Pu4 / d1 d2 media)
  (setq d1 (distance Pu1 Pu2)
        d2 (distance Pu3 Pu4)
        media (/ (+ d1 d2) 2.0))
  (if (> media 1e-12)
      (< (/ (abs (- d1 d2)) media) 1e-6)
      T))
;;-------------------------------------------------------------------------------------
(defun Math_Area_Triangolo (Pu1 Pu2 Pu3)  (* 0.5 (distance '(0 0 0) (Vet_Cro (Vet_Sub Pu2 Pu1)
     (Vet_Sub Pu3 Pu1)))))
;;-------------------------------------------------------------------------------------
(defun Math_Angolo_Minore_CCW (Cen P1 P2)  (< (angle Cen P1) (angle Cen P2)))
;;-------------------------------------------------------------------------------------
(defun Math_Aree_Pu_vs_Tri (Pu1 Pu2 Pu3 Pu4)
  (list (Math_Area_Triangolo Pu2 Pu3 Pu4)
        (Math_Area_Triangolo Pu1 Pu2 Pu3)
        (Math_Area_Triangolo Pu1 Pu3 Pu4)
        (Math_Area_Triangolo Pu1 Pu2 Pu4)))
;;-------------------------------------------------------------------------------------
(defun Math_Check_Conciclicita_4Pu (Pu1 Pu2 Pu3 Pu4 / Pu5 Pu6 Pu7 Pu8 dx6 dy6 dx7 dy7 dx8 dy8 a
     b c d e f g h i det area scala)
  (if (P_0Coinc_4Pu Pu1 Pu2 Pu3 Pu4)
    (progn
      (mapcar 'set '(Pu5 Pu6 Pu7 Pu8) (Geom_Ordina_CCW_4Pu Pu1 Pu2 Pu3 Pu4))
      (mapcar 'set '(dx6 dy6) (Vet_Sub Pu6 Pu5))
      (mapcar 'set '(dx7 dy7) (Vet_Sub Pu7 Pu5))
      (mapcar 'set '(dx8 dy8) (Vet_Sub Pu8 Pu5))
      (setq a dx6 b dy6 c (Math_Sum_Squares dx6 dy6)
            d dx7 e dy7 f (Math_Sum_Squares dx7 dy7)
            g dx8 h dy8 i (Math_Sum_Squares dx8 dy8)
            det (Math_Det3x3 a b c d e f g h i)
            area (+ (Math_Area_Triangolo Pu5 Pu6 Pu7) (Math_Area_Triangolo Pu5 Pu7 Pu8))
            scala (* area (distance Pu5 Pu7)))
      (if (> scala 1e-12) (< (/ (abs det) scala) 1e-4) T))))
;;-------------------------------------------------------------------------------------
```

```
(defun Math_Conta_Paral_4Pu (Pu1 Pu2 Pu3 Pu4)
  (if (P_Conve_4Pu Pu1 Pu2 Pu3 Pu4)
     (length (vl-remove nil (P_Paral_distinte_List_2Re (list (list Pu1 Pu2 Pu3 Pu4)
     (list Pu1 Pu3 Pu2 Pu4) (list Pu1 Pu4 Pu2 Pu3)))))))
;;-------------------------------------------------------------------------------------
(defun Math_Conta_Ortogo_4Pu (Pu1 Pu2 Pu3 Pu4 / lista Pu5 Pu6 Pu7 Pu8)
  (if (setq lista (Geom_Ordina_Concavo_4Pu Pu1 Pu2 Pu3 Pu4))
      (progn (mapcar 'set '(Pu5 Pu6 Pu7 Pu8) lista)
             (length (vl-remove nil (P_Perpe_List_2Re (list (list Pu5 Pu8 Pu6 Pu7)
     (list Pu6 Pu8 Pu5 Pu7) (list Pu7 Pu8 Pu5 Pu6)))))))
;;-------------------------------------------------------------------------------------
(defun Math_Conta_Coinc_4Pu (Pu1 Pu2 Pu3 Pu4)
     (length (vl-remove nil (P_Coinc_List_2Pu (list (list Pu1 Pu2) (list Pu1 Pu3)
     (list Pu1 Pu4) (list Pu2 Pu3) (list Pu2 Pu4) (list Pu3 Pu4))))))
;;-------------------------------------------------------------------------------------
(defun P_Coinc_2Pu      (Pu1 Pu2)    (if (and Pu1 Pu2) (< (distance Pu1 Pu2) 1e-6)))
;;-------------------------------------------------------------------------------------
(defun P_Coinc_List_2Pu (lst_pairs)  (mapcar '(lambda (pair) (P_Coinc_2Pu (car pair)
     (cadr pair))) lst_pairs))
;;-------------------------------------------------------------------------------------
(defun P_Coinc_3Pu (Pu1 Pu2 Pu3)      (apply 'and (P_Coinc_List_2Pu (list (list Pu1 Pu2)
     (list Pu2 Pu3) (list Pu1 Pu3)))))
;;-------------------------------------------------------------------------------------
(defun P_Coinc_4Pu (Pu1 Pu2 Pu3 Pu4)  (apply 'and (P_Coinc_List_2Pu (list (list Pu1 Pu2)
     (list Pu1 Pu3) (list Pu1 Pu4) (list Pu2 Pu3) (list Pu2 Pu4) (list Pu3 Pu4)))))
;;-------------------------------------------------------------------------------------
(defun P_Coinc_2su3Pu (Pu1 Pu2 Pu3 Only)
     (and (apply 'or (P_Coinc_List_2Pu (list (list Pu1 Pu2) (list Pu2 Pu3) (list Pu1 Pu3))))
     (if Only (not (P_Coinc_3Pu Pu1 Pu2 Pu3)) T)))
;;-------------------------------------------------------------------------------------
(defun P_Allin_3Pu (Pu1 Pu2 Pu3 / v1 v2 cp_mag len_prod)
  (if (and Pu1 Pu2 Pu3)
    (or
      (P_Coinc_2su3Pu Pu1 Pu2 Pu3 nil)
      (progn
        (setq v1       (mapcar '- Pu2 Pu1)
              v2       (mapcar '- Pu3 Pu1)
              cp_mag   (distance '(0 0 0) (Vet_Cro v1 v2))
              len_prod (* (distance Pu1 Pu2) (distance Pu1 Pu3)))
        (< (/ cp_mag len_prod) 1e-4)))))
;;-------------------------------------------------------------------------------------
(defun P_Allin_List_3Pu (lst_triples)  (mapcar '(lambda (tri) (P_Allin_3Pu (car tri) (cadr tri)
     (caddr tri))) lst_triples))
;;-------------------------------------------------------------------------------------
(defun P_Allin_4Pu (Pu1 Pu2 Pu3 Pu4)  (apply 'and (P_Allin_List_3Pu (list (list Pu1 Pu2 Pu3)
```

```
     (list Pu1 Pu2 Pu4) (list Pu1 Pu3 Pu4) (list Pu2 Pu3 Pu4)))))
;;-------------------------------------------------------------------------------------
(defun P_Allin_3su4Pu (Pu1 Pu2 Pu3 Pu4 Only)
  (and (apply 'or (P_Allin_List_3Pu (list (list Pu1 Pu2 Pu3) (list Pu1 Pu2 Pu4)
     (list Pu1 Pu3 Pu4) (list Pu2 Pu3 Pu4)))) (if Only (not (P_Allin_4Pu Pu1 Pu2 Pu3 Pu4)) T)))
;;-------------------------------------------------------------------------------------
(defun P_Pu_Inter_3Pu (Pu1 Pu2 Pu3 Pu4 / a234 a123 a134 a124)
  (mapcar 'set '(a234 a123 a134 a124) (Math_Aree_Pu_vs_Tri Pu1 Pu2 Pu3 Pu4))
  (and (not (P_Allin_3su4Pu Pu1 Pu2 Pu3 Pu4 nil))
       (equal a234 (+ a123 a134 a124) 1e-5)))
;;-------------------------------------------------------------------------------------
(defun P_Pu_Ester_3Pu (Pu1 Pu2 Pu3 Pu4 / a234 a123 a134 a124)
  (mapcar 'set '(a234 a123 a134 a124) (Math_Aree_Pu_vs_Tri Pu1 Pu2 Pu3 Pu4))
  (not (or (P_Allin_3su4Pu Pu1 Pu2 Pu3 Pu4 nil)
           (equal a234 (+ a123 a134 a124) 1e-5))))
;;-------------------------------------------------------------------------------------
(defun P_Pu_Inter_List_4Pu (lst_quartets)  (mapcar '(lambda (qua) (P_Pu_Inter_3Pu (car qua)
     (cadr qua) (caddr qua) (cadddr qua))) lst_quartets))
;;-------------------------------------------------------------------------------------
(defun P_Pu_Ester_List_4Pu (lst_quartets)  (mapcar '(lambda (qua) (P_Pu_Ester_3Pu (car qua)
     (cadr qua) (caddr qua) (cadddr qua))) lst_quartets))
;;-------------------------------------------------------------------------------------
(defun P_Conca_4Pu (Pu1 Pu2 Pu3 Pu4)
     (apply 'or  (P_Pu_Inter_List_4Pu (list (list Pu1 Pu2 Pu3 Pu4) (list Pu2 Pu3 Pu4 Pu1)
     (list Pu3 Pu4 Pu1 Pu2) (list Pu4 Pu1 Pu2 Pu3)))))
;;-------------------------------------------------------------------------------------
(defun P_Conve_4Pu (Pu1 Pu2 Pu3 Pu4)
     (apply 'and (P_Pu_Ester_List_4Pu (list (list Pu1 Pu2 Pu3 Pu4) (list Pu2 Pu3 Pu4 Pu1)
     (list Pu3 Pu4 Pu1 Pu2) (list Pu4 Pu1 Pu2 Pu3)))))
;;-------------------------------------------------------------------------------------
(defun P_0Paral_4Pu (Pu1 Pu2 Pu3 Pu4) (equal (Math_Conta_Paral_4Pu Pu1 Pu2 Pu3 Pu4) 0))
;;-------------------------------------------------------------------------------------
(defun P_1Paral_4Pu (Pu1 Pu2 Pu3 Pu4) (equal (Math_Conta_Paral_4Pu Pu1 Pu2 Pu3 Pu4) 1))
;;-------------------------------------------------------------------------------------
(defun P_2Paral_4Pu (Pu1 Pu2 Pu3 Pu4) (equal (Math_Conta_Paral_4Pu Pu1 Pu2 Pu3 Pu4) 2))
;;-------------------------------------------------------------------------------------
(defun P_0Orto_4Pu (Pu1 Pu2 Pu3 Pu4) (equal (Math_Conta_Ortogo_4Pu Pu1 Pu2 Pu3 Pu4) 0))
;;-------------------------------------------------------------------------------------
(defun P_1Orto_4Pu (Pu1 Pu2 Pu3 Pu4) (equal (Math_Conta_Ortogo_4Pu Pu1 Pu2 Pu3 Pu4) 1))
;;-------------------------------------------------------------------------------------
(defun P_2Orto_4Pu (Pu1 Pu2 Pu3 Pu4) (equal (Math_Conta_Ortogo_4Pu Pu1 Pu2 Pu3 Pu4) 2))
;;-------------------------------------------------------------------------------------
(defun P_3Orto_4Pu (Pu1 Pu2 Pu3 Pu4) (equal (Math_Conta_Ortogo_4Pu Pu1 Pu2 Pu3 Pu4) 3))
;;-------------------------------------------------------------------------------------
(defun P_0Coinc_4Pu (Pu1 Pu2 Pu3 Pu4) (equal (Math_Conta_Coinc_4Pu Pu1 Pu2 Pu3 Pu4) 0))
```

```
;;-------------------------------------------------------------------------------------------
(defun P_1Coinc_4Pu (Pu1 Pu2 Pu3 Pu4) (equal (Math_Conta_Coinc_4Pu Pu1 Pu2 Pu3 Pu4) 1))
;;-------------------------------------------------------------------------------------------
(defun P_2Coinc_4Pu (Pu1 Pu2 Pu3 Pu4) (equal (Math_Conta_Coinc_4Pu Pu1 Pu2 Pu3 Pu4) 2))
;;-------------------------------------------------------------------------------------------
(defun P_3Coinc_4Pu (Pu1 Pu2 Pu3 Pu4) (equal (Math_Conta_Coinc_4Pu Pu1 Pu2 Pu3 Pu4) 3))
;;-------------------------------------------------------------------------------------------
(defun P_6Coinc_4Pu (Pu1 Pu2 Pu3 Pu4) (equal (Math_Conta_Coinc_4Pu Pu1 Pu2 Pu3 Pu4) 6))
;;-------------------------------------------------------------------------------------------
(defun P_Paral_2Re (Re1A Re1B Re2A Re2B / v1 v2 cp_mag len_prod)
  (setq v1       (Vet_Sub Re1B Re1A)
        v2       (Vet_Sub Re2B Re2A)
        cp_mag   (distance '(0 0 0) (Vet_Cro v1 v2))
        len_prod (* (distance Re1A Re1B) (distance Re2A Re2B)))
  (if (> len_prod 1e-12)
    (< (/ cp_mag len_prod) 1e-4)))
;;-------------------------------------------------------------------------------------------
(defun P_Paral_distinte_2Re (Re1A Re1B Re2A Re2B)
  (and (P_Paral_2Re Re1A Re1B Re2A Re2B)
       (not (P_Allin_4Pu Re1A Re1B Re2A Re2B))))
;;-------------------------------------------------------------------------------------------
(defun P_Perpe_2Re (Re1A Re1B Re2A Re2B / v1 v2 dot len_prod)
  (setq v1       (Vet_Sub Re1B Re1A)
        v2       (Vet_Sub Re2B Re2A)
        dot      (Vet_Dot v1 v2)
        len_prod (* (distance Re1A Re1B) (distance Re2A Re2B)))
  (if (> len_prod 1e-12)
    (< (/ (abs dot) len_prod) 1e-4)))
;;-------------------------------------------------------------------------------------------
(defun P_Paral_List_2Re          (lst_pairs_pairs)  (mapcar '(lambda (pp)
     (P_Paral_2Re          (car pp) (cadr pp) (caddr pp) (cadddr pp))) lst_pairs_pairs))
;;-------------------------------------------------------------------------------------------
(defun P_Paral_distinte_List_2Re (lst_pairs_pairs)  (mapcar '(lambda (pp)
     (P_Paral_distinte_2Re (car pp) (cadr pp) (caddr pp) (cadddr pp))) lst_pairs_pairs))
;;-------------------------------------------------------------------------------------------
(defun P_Perpe_List_2Re          (lst_pairs_pairs)  (mapcar '(lambda (pp)
     (P_Perpe_2Re          (car pp) (cadr pp) (caddr pp) (cadddr pp))) lst_pairs_pairs))
;;-------------------------------------------------------------------------------------------
(defun P_Paral_4Pu          (Pu1 Pu2 Pu3 Pu4)
     (apply 'or (P_Paral_List_2Re          (list (list Pu1 Pu2 Pu3 Pu4) (list Pu2 Pu3 Pu4 Pu1)
     (list Pu1 Pu3 Pu2 Pu4)))))
;;-------------------------------------------------------------------------------------------
(defun P_Paral_distinti_4Pu (Pu1 Pu2 Pu3 Pu4)
     (apply 'or (P_Paral_distinte_List_2Re (list (list Pu1 Pu2 Pu3 Pu4)
     (list Pu2 Pu3 Pu4 Pu1)))))
```

```
;;-------------------------------------------------------------------------------------------
(defun P_Quadrangolo_4Pu          (Pu1 Pu2 Pu3 Pu4) (P_0Paral_4Pu Pu1 Pu2 Pu3 Pu4))
;;-------------------------------------------------------------------------------------------
(defun P_Trapezio_4Pu             (Pu1 Pu2 Pu3 Pu4) (P_1Paral_4Pu Pu1 Pu2 Pu3 Pu4))
;;-------------------------------------------------------------------------------------------
(defun P_Parallelogramma_4Pu      (Pu1 Pu2 Pu3 Pu4) (P_2Paral_4Pu Pu1 Pu2 Pu3 Pu4))
;;-------------------------------------------------------------------------------------------
(defun P_Quadrangolo_Generico_4Pu (Pu1 Pu2 Pu3 Pu4) (and (P_0Paral_4Pu Pu1 Pu2 Pu3 Pu4)
     (not (P_Conciclici_4Pu Pu1 Pu2 Pu3 Pu4))))
;;-------------------------------------------------------------------------------------------
(defun P_Trapezio_Scaleno_4Pu     (Pu1 Pu2 Pu3 Pu4) (and (P_Trapezio_4Pu Pu1 Pu2 Pu3 Pu4)
     (not (P_Trapezio_Isoscele_4Pu Pu1 Pu2 Pu3 Pu4))))
;;-------------------------------------------------------------------------------------------
(defun P_Trapezio_Isoscele_4Pu (Pu1 Pu2 Pu3 Pu4 / Pu5 Pu6 Pu7 Pu8)
  (mapcar 'set '(Pu5 Pu6 Pu7 Pu8) (Geom_Ordina_CCW_4Pu Pu1 Pu2 Pu3 Pu4))
     (and (P_Trapezio_4Pu Pu5 Pu6 Pu7 Pu8) (Math_Uguaglianza_Relativa_2Re Pu5 Pu7 Pu6 Pu8)))
;;-------------------------------------------------------------------------------------------
(defun P_Rettangolo_4Pu (Pu1 Pu2 Pu3 Pu4 / Pu5 Pu6 Pu7 Pu8)
  (mapcar 'set '(Pu5 Pu6 Pu7 Pu8) (Geom_Ordina_CCW_4Pu Pu1 Pu2 Pu3 Pu4))
     (and (P_Parallelogramma_4Pu Pu5 Pu6 Pu7 Pu8) (P_Perpe_2Re Pu5 Pu6 Pu6 Pu7)))
;;-------------------------------------------------------------------------------------------
(defun P_Rombo_4Pu (Pu1 Pu2 Pu3 Pu4 / Pu5 Pu6 Pu7 Pu8)
  (mapcar 'set '(Pu5 Pu6 Pu7 Pu8) (Geom_Ordina_CCW_4Pu Pu1 Pu2 Pu3 Pu4))
     (and (P_Parallelogramma_4Pu Pu5 Pu6 Pu7 Pu8)
     (Math_Uguaglianza_Relativa_2Re Pu5 Pu6 Pu6 Pu7)))
;;-------------------------------------------------------------------------------------------
(defun P_Quadrato_4Pu (Pu1 Pu2 Pu3 Pu4 / Pu5 Pu6 Pu7 Pu8)
  (mapcar 'set '(Pu5 Pu6 Pu7 Pu8) (Geom_Ordina_CCW_4Pu Pu1 Pu2 Pu3 Pu4))
     (and (P_Rettangolo_4Pu Pu5 Pu6 Pu7 Pu8) (Math_Uguaglianza_Relativa_2Re Pu5 Pu6 Pu6 Pu7)))
;;-------------------------------------------------------------------------------------------
(defun P_Conciclici_4Pu (Pu1 Pu2 Pu3 Pu4 / Pu5 Pu6 Pu7 Pu8)
  (mapcar 'set '(Pu5 Pu6 Pu7 Pu8) (Geom_Ordina_CCW_4Pu Pu1 Pu2 Pu3 Pu4))
     (and (P_Conve_4Pu Pu5 Pu6 Pu7 Pu8) (Math_Check_Conciclicita_4Pu Pu5 Pu6 Pu7 Pu8)))
;;-------------------------------------------------------------------------------------------
(defun P_Concavo_Ortocentrico_4Pu (Pu1 Pu2 Pu3 Pu4) (P_3Orto_4Pu Pu1 Pu2 Pu3 Pu4))
;;-------------------------------------------------------------------------------------------
(defun P_Concavo_Generico_4Pu     (Pu1 Pu2 Pu3 Pu4) (or (P_0Orto_4Pu Pu1 Pu2 Pu3 Pu4)
     (P_1Orto_4Pu Pu1 Pu2 Pu3 Pu4) (P_2Orto_4Pu Pu1 Pu2 Pu3 Pu4)))
)
;;-------------------------------------------------------------------------------------------
;;-------------------------------------------------------------------------------------------

;;-------------------------------------------------------------------------------------------
;;-------------------------------------------------------------------------------------------
(progn
```

```
;;-------------------------------------------------------------------------------
(defun Draw_ConCen_3P (Center P1 P2 P3 pts_limit full_mode / scale_factor lisPu_norm P1_n P2_n
     P3_n coeffs A B C l1 l2 vec1 vec2 len1_norm len2_norm a b c V_Maj V_Min Ax1 Ax2 Ax3 Ax4 F1
     F2 ratio l_real l_imag V1 vec_Real vec_Perp pts_ext ents tmp_c)
  (if (P_0Coinc_4Pu Center P1 P2 P3)
    (progn
      (mapcar 'set '(scale_factor lisPu_norm) (Util_Normalizza_lisPu_Da_Cen (list P1 P2 P3)
     Center))
      (mapcar 'set '(P1_n P2_n P3_n) lisPu_norm)
      (setq coeffs (Alg3x3_Calcola_Coeffs '(0.0 0.0 0.0) P1_n P2_n P3_n))
      (if coeffs
        (progn
          (mapcar 'set '(A B C) coeffs)
          (mapcar 'set '(l1 l2) (Alg3x3_Autovalori A B C))
          (cond
            ((and (> l1 0) (> l2 0))
             (if full_mode (princ "\n[INFO] Rilevata Ellisse (o Cerchio)."))
             (setq vec1 (Alg3x3_Autovettore A B C l1)
                   vec2 (Alg3x3_Autovettore A B C l2)
                   len1_norm (* (Math_Inv_Sqrt l1) scale_factor)
                   len2_norm (* (Math_Inv_Sqrt l2) scale_factor))
             (if (> len1_norm len2_norm)
               (setq a len1_norm   b len2_norm   V_Maj vec1   V_Min vec2)
               (setq a len2_norm   b len1_norm   V_Maj vec2   V_Min vec1))
             (setq ratio (/ b a))

             (if full_mode
                 (progn
                   (setq c (if (setq tmp_c (Pitag_Ca a b)) tmp_c 0.0))
                   (mapcar 'set '(Ax1 Ax2) (Geom_2Pu_Simm_PuC Center V_Maj a))
                   (mapcar 'set '(Ax3 Ax4) (Geom_2Pu_Simm_PuC Center V_Min b))
                   (mapcar 'set '(F1 F2)   (Geom_2Pu_Simm_PuC Center V_Maj c))
                   (setq ents (Draw_ConEll_CeAx_Foci Center Ax1 Ax2 Ax3 Ax4 F1 F2 ratio)))
                 (setq ents (list (Draw_ConEll_CeAx_Light Center (Vet_Sum Center (Vet_Sca V_Maj
     a)) ratio))))
             ents)
            ((< (* l1 l2) 0)
             (if full_mode (princ "\n[INFO] Rilevata Iperbole."))
             (if (> l1 0)
                 (setq vec_Real (Alg3x3_Autovettore A B C l1)   l_real l1   l_imag l2)
                 (setq vec_Real (Alg3x3_Autovettore A B C l2)   l_real l2   l_imag l1))
             (setq a        (* (Math_Inv_Sqrt (abs l_real)) scale_factor)
                   b        (* (Math_Inv_Sqrt (abs l_imag)) scale_factor)
                   c        (Pitag_Ip a b)
                   vec_Perp (list (- (cadr vec_Real)) (car vec_Real) 0.0)
```

```
                V1       (Get_Pt_On_Axes Center vec_Real vec_Perp a 0.0)
                F1       (Get_Pt_On_Axes Center vec_Real vec_Perp c 0.0))
            (if (not pts_limit) (setq pts_limit (list P1 P2 P3)))
            (setq pts_ext (Util_Scale_Pts_From_Centroid pts_limit 6.0))

            (if full_mode
                (setq ents (Draw_ConIpe_CeVeFu_Compl Center V1 F1 pts_ext))
                (setq ents (Draw_ConIpe_CeVeFu_Light Center V1 F1 pts_ext)))
            ents)

           (T (if full_mode (princ "\n[ERRORE] Configurazione degenere o immaginaria."))
     nil)))
        (progn (if full_mode (princ "\n[ERRORE] Matrice singolare (punti allineati col centro o
     impossibili)."")) nil)))
    (progn (if full_mode (princ "\n[ERRORE] Punti coincidenti tra loro o sovrapposti al centro.
     "")) nil)))
;;-------------------------------------------------------------------------------------------
(defun Draw_ConCen_3P_Compl (Center P1 P2 P3 pts_limit)
     (Draw_ConCen_3P Center P1 P2 P3 pts_limit T))
;;-------------------------------------------------------------------------------------------
(defun Draw_ConCen_3P_Light (Center P1 P2 P3 pts_limit)
     (Draw_ConCen_3P Center P1 P2 P3 pts_limit nil))
;;-------------------------------------------------------------------------------------------
(defun Draw_ConEll_CeVe_CoVe (Center V1 F1 pts / a c b V_Axis A1 A2 B1 B2 F2 ratio)
  (setq V_Axis (Vet_Uni (Vet_Sub V1 Center))
        a (distance Center V1)
        c (distance Center F1)
         b (if (> a c) (sqrt (- (* a a) (* c c))) (* a 0.1))
         A1 V1
        A2 (Vet_Sum Center (Vet_Sca V_Axis (- a)))
         B1 (Vet_Sum Center (Vet_Perp_Sca V_Axis b))
        B2 (Vet_Sum Center (Vet_Perp_Sca V_Axis (- b)))
         F2 (Vet_Sum Center (Vet_Sca V_Axis (- c))) ratio (/ b a))
  (Draw_ConEll_CeAx_Foci Center A1 A2 B1 B2 F1 F2 ratio))
;;-------------------------------------------------------------------------------------------
(defun Draw_ConEll_CeAx_Foci (Center A1 A2 B1 B2 F1 F2 Ratio / Vec_Maj Ents)
  (setq Vec_Maj (Vet_Sub A1 Center))
  (setq Ents (append (list (Draw_ConEll_R Center Vec_Maj Ratio *LaCo_Tracciati*)
                           (Draw_Point_R Center *LaCo_Centri*))
                     (Draw_Lis_Line_Lis_Coppie_Pu_R (list (list A1 A2) (list B1 B2))
     *LaCo_Assi*)
                     (Draw_Lis_Point_R              (list F1 F2)
     *LaCo_Fuochi*))))
;;-------------------------------------------------------------------------------------------
(defun Draw_ConEll_CeAx_Light (Center A1 Ratio / Vec_Maj)
```

```
    (Draw_ConEll_R Center (Vet_Sub A1 Center) Ratio *LaCo_Luoghi_geometrici*))
;;-------------------------------------------------------------------------------------
(defun Draw_ConIpe_CeVeFu (Center V1 F1 pts free_mode full_mode / V_Axis V_Perp a c b max_proj
    len_draw w y_lim D_cv y_asym F2 P_S1 P_CV1 P_E1 P_S2 P_CV2 P_E2 As1A As1B As2A As2B AxReA
    AxReB AxImA AxImB)
  (setq V_Axis   (Vet_Uni (Vet_Sub V1 Center))
        V_Perp   (list (- (cadr V_Axis)) (car V_Axis) 0.0)
        a        (distance Center V1)
        c        (distance Center F1)
        b        (Pitag_Ca c a)
        max_proj (get-max-abs-projection-on-axis pts Center V_Axis)
        len_draw (if free_mode
                     (max (* a 1.001) max_proj)
                     (if (> max_proj (* c 2.0)) (* max_proj 1.1) (* c 2.0)))
        w        (/ len_draw a)
        y_lim    (* b (sqrt (- (Math_Sqr w) 1.0)))
        D_cv     (/ (Math_Sqr a) len_draw)
        y_asym   (* len_draw (/ b a))
        F2       (Vet_Sum Center (Vet_Sca V_Axis (- c))))
  (mapcar 'set '(P_S1 P_CV1 P_E1)
     (Calc_Ipe_Branch_CVs Center V_Axis V_Perp D_cv len_draw y_lim 1.0))
  (mapcar 'set '(P_S2 P_CV2 P_E2)
     (Calc_Ipe_Branch_CVs Center V_Axis V_Perp D_cv len_draw y_lim -1.0))
  (if full_mode
      (progn
        (mapcar 'set '(As1A As1B As2A As2B)
     (Calc_Ipe_Asym_Pts Center V_Axis V_Perp len_draw y_asym))
        (mapcar 'set '(AxReA AxReB AxImA AxImB)
     (Calc_Ipe_Axes_Pts Center V_Axis V_Perp len_draw))
        (append (list (Draw_Spline_Rational_3CV P_S1 P_CV1 P_E1 w *LaCo_Tracciati*)
                      (Draw_Spline_Rational_3CV P_S2 P_CV2 P_E2 w *LaCo_Tracciati*)
                      (Draw_Point_R Center *LaCo_Centri*))
                (Draw_Lis_Line_Lis_Coppie_Pu_R (list (list As1A As1B)   (list As2A As2B))
     *LaCo_Asintoti*)
                (Draw_Lis_Line_Lis_Coppie_Pu_R (list (list AxReA AxReB) (list AxImA AxImB))
     *LaCo_Assi*)
                (Draw_Lis_Point_R              (list F1 F2)
     *LaCo_Fuochi*)))
      (list (Draw_Spline_Rational_3CV P_S1 P_CV1 P_E1 w *LaCo_Luoghi_geometrici*)
            (Draw_Spline_Rational_3CV P_S2 P_CV2 P_E2 w *LaCo_Luoghi_geometrici*))))
;;-------------------------------------------------------------------------------------
(defun Draw_ConIpe_CeVeFu_Compl       (Center V1 F1 pts)
    (Draw_ConIpe_CeVeFu Center V1 F1 pts nil T))
;;-------------------------------------------------------------------------------------
(defun Draw_ConIpe_CeVeFu_Compl_Free  (Center V1 F1 pts)
```

```
     (Draw_ConIpe_CeVeFu Center V1 F1 pts T   T))
;;-------------------------------------------------------------------------------------
(defun Draw_ConIpe_CeVeFu_Light       (Center V1 F1 pts)
     (Draw_ConIpe_CeVeFu Center V1 F1 pts nil nil))
;;-------------------------------------------------------------------------------------
(defun Draw_ConIpe_CeVeFu_Light_Free  (Center V1 F1 pts)
     (Draw_ConIpe_CeVeFu Center V1 F1 pts T   nil))
;;-------------------------------------------------------------------------------------
(defun Draw_ConIpe_4P_Eq (Pu1 Pu2 Pu3 Pu4 lay_crv lay_constr / math_data PuC V_dir a b c V1 F1)
  (setq math_data (Geom_Ipe_4P_Eq Pu1 Pu2 Pu3 Pu4))
  (if math_data
      (progn
        (mapcar 'set '(PuC V_dir a b) math_data)
        (setq c  (Pitag_Ip a b)
              V1 (Vet_Sum PuC (Vet_Sca V_dir a))
              F1 (Vet_Sum PuC (Vet_Sca V_dir c)))
        (Draw_ConIpe_CeVeFu_Compl PuC V1 F1 (list Pu1 Pu2 Pu3 Pu4))
        (princ "\n[INFO] Iperbole Equilatera generata con successo."))
      (princ "\n[ERRORE] Impossibile generare iperbole (Configurazione degenere)."))
  (princ))
;;-------------------------------------------------------------------------------------
(defun Draw_ConPar_VeFu (PuV PuF pts full_mode / V_Asse V_Norm p max_proj X Y P_Start P_CV P_En
     d Ax1 Ax2 DirB DirA)
  (setq V_Asse (Vet_Uni (Vet_Sub PuF PuV))
        p      (distance PuV PuF)
        V_Norm (Vet_Uni (Vet_Cro V_Asse (if (and (< (abs (car V_Asse)) 1e-6)
     (< (abs (cadr V_Asse)) 1e-6)) '(0 1 0) '(0 0 1))))
        max_proj (get-max-abs-projection-on-axis pts PuV V_Asse)
        Y      (if (> max_proj (* p 2.0)) (* max_proj 2.0) (* p 4.0))
        X      (sqrt (* 4.0 p Y)))
  (mapcar 'set '(P_Start P_CV P_End) (Calc_Par_Branch_CVs PuV V_Asse V_Norm Y X))
  (if full_mode
      (progn (mapcar 'set '(Ax1 Ax2)   (Calc_Par_Axis_Pts PuV V_Asse V_Norm p Y))
             (mapcar 'set '(DirA DirB) (Calc_Par_Directrix_Pts PuV V_Asse V_Norm p X))
             (list (Draw_Spline_Rational_3CV P_Start P_CV P_End 1.0 *LaCo_Tracciati*)
                   (Draw_Line_2Pu_R Ax1 Ax2 *LaCo_Assi*)
                   (Draw_Line_2Pu_R DirA DirB *LaCo_Direttrice*)
                   (Draw_Point_R PuF *LaCo_Fuochi*)
                   (Draw_Point_R PuV *LaCo_Vertici*)))
      (progn (Draw_Spline_Rational_3CV P_Start P_CV P_End 1.0 *LaCo_Luoghi_geometrici*))))
;;-------------------------------------------------------------------------------------
(defun Draw_ConPar_VeFu_Compl (PuV PuF pts) (Draw_ConPar_VeFu PuV PuF pts T))
;;-------------------------------------------------------------------------------------
(defun Draw_ConPar_VeFu_Light (PuV PuF pts) (Draw_ConPar_VeFu PuV PuF pts nil))
;;-------------------------------------------------------------------------------------
```

```
(defun Draw_ConPar_Data_3P (pts_3 refined_data)
     (if refined_data (Draw_ConPar_VeFu (car refined_data) (cadr refined_data) pts_3 T)))
;;-------------------------------------------------------------------------------------------
(defun Draw_ConPar_5P_Sub (pts_norm analyze_res G_bary max_dist pts_limit full_mode / Ve Fu p s
     olve_res pts_ext)
  (setq solve_res (Solve_Par_Geometry pts_norm (nth 2 analyze_res) (nth 3 analyze_res)))
  (if solve_res
      (progn  (mapcar 'set '(Ve Fu p) solve_res)
              (mapcar 'set '(Ve Fu) (Util_Denorm_List_Pts (list Ve Fu) G_bary max_dist))
              (setq pts_ext (if pts_limit pts_limit (Util_Denorm_List_Pts pts_norm G_bary max_d
     ist)))
              (if full_mode
                  (Draw_ConPar_VeFu Ve Fu pts_ext T)
                  (list (Draw_ConPar_VeFu Ve Fu pts_ext nil))))))
;;-------------------------------------------------------------------------------------------
(defun Draw_ConPar_5P_Sub_Compl (pts_norm analyze_res G_bary max_dist pts_limit)
     (Draw_ConPar_5P_Sub pts_norm analyze_res G_bary max_dist pts_limit T))
;;-------------------------------------------------------------------------------------------
(defun Draw_ConPar_5P_Sub_Light (pts_norm analyze_res G_bary max_dist pts_limit)
     (Draw_ConPar_5P_Sub pts_norm analyze_res G_bary max_dist pts_limit nil))
;;-------------------------------------------------------------------------------------------
(defun Draw_ConPar_4P_Light (P1 P2 P3 P4 / Angles_Deg V_Norm z_orig pts_o Cg dx dy ang_pca c_pr
     e s_pre c s pts_rot x1 x2 x3 x4 y1 y2 y3 y4 sq_x1 sq_x2 sq_x3 sq_x4 D ka kb kc max_err err
     Xv Yv p_dist pts_temp V_wcs F_wcs l_ents unique_count is_geom_twin processed_angles)
  (setq Angles_Deg (Geom_Find_Parabolic_Angles P1 P2 P3 P4)
        V_Norm     (Vet_Uni (Vet_Cro (Vet_Sub P2 P1) (Vet_Sub P3 P1)))
        z_orig     (caddr (trans P1 0 V_Norm))
        pts_o      (Util_Trans_Pts_List (list P1 P2 P3 P4) 0 V_Norm)
        Cg         (Pu_Baricentro2D_LisPu pts_o)
        pts_o      (Util_Translate_Pts_Relative pts_o Cg))
  (mapcar 'set '(dx dy) (Util_Get_Pts_BBox_Size pts_o))
  (setq ang_pca (if (> dy dx) (/ pi 2.0) (angle '(0 0) (Vet_Sub (nth 2 pts_o) (nth 0 pts_o))))
        c_pre   (cos ang_pca)
        s_pre   (sin ang_pca)
        pts_o   (Util_Rot_List_2D_CS pts_o c_pre s_pre)
        l_ents  nil
        processed_angles nil
        unique_count 0)
  (if Angles_Deg
      (foreach Theta_Deg Angles_Deg
        (setq c       (cos (* Theta_Deg (/ pi 180.0)))
              s       (sin (* Theta_Deg (/ pi 180.0)))
              pts_rot (Util_Rot_List_2D_CS pts_o c s))
        (mapcar 'set '(x1 x2 x3 x4)             (Math_Extract_Coord_Pow pts_rot 0 1))
        (mapcar 'set '(y1 y2 y3 y4)             (Math_Extract_Coord_Pow pts_rot 1 1))
```

```
      (mapcar 'set '(sq_x1 sq_x2 sq_x3 sq_x4) (Math_Extract_Coord_Pow pts_rot 0 2))
      (setq D (Math_Cyclic_Det sq_x1 sq_x2 sq_x3 x1 x2 x3))
      (if (> (abs D) 1e-9)
          (progn
            (setq ka (/ (Math_Cyclic_Det y1 y2 y3 x1 x2 x3) D)
                  kb (/ (Math_Cyclic_Det sq_x1 sq_x2 sq_x3 y1 y2 y3) D)
                  kc (/ (+ (* sq_x1 (Math_Diff_Prod x2 y3 x3 y2))
                           (* sq_x2 (Math_Diff_Prod x3 y1 x1 y3))
                           (* sq_x3 (Math_Diff_Prod x1 y2 x2 y1))) D)
                  max_err 0.0)
            (foreach p_check pts_rot
              (setq err (abs (- (+ (Math_Sum_Prod ka (Math_Sqr (car p_check))
    kb (car p_check)) kc) (cadr p_check))))
              (if (> err max_err) (setq max_err err)))
            (if (< max_err 1e-2)
                (progn
                  (setq is_geom_twin (vl-some '(lambda (pa) (< (abs (- (abs (- pa Theta_Deg))
    180.0)) 2.0)) processed_angles))
                  (if (not is_geom_twin)
                      (progn
                        (setq unique_count     (1+ unique_count)
                              processed_angles (cons Theta_Deg processed_angles))
                        (setq Xv       (/ (- kb) (* 2.0 ka))
                              Yv       (- kc (/ (* kb kb) (* 4.0 ka)))
                              p_dist   (/ 1.0 (* 4.0 ka))
                              pts_temp (Util_Rot_List_2D_Inv_CS (Util_Rot_List_2D_Inv_CS (lis
    t (list Xv Yv) (list Xv (+ Yv p_dist))) c s) c_pre s_pre)
                              V_wcs    (trans (Vet_Sum (list (car (nth 0 pts_temp))
    (cadr (nth 0 pts_temp)) z_orig) Cg) V_Norm 0)
                              F_wcs    (trans (Vet_Sum (list (car (nth 1 pts_temp))
    (cadr (nth 1 pts_temp)) z_orig) Cg) V_Norm 0)
                              l_ents   (append l_ents (list  (Draw_ConPar_VeFu_Light V_wcs F_
    wcs (list P1 P2 P3 P4))))
                        ))))))))) l_ents)
;;-------------------------------------------------------------------------------------------
(defun Draw_ConPar_4P_ConVer_R (P1 P2 P3 P4 lay_crv lay_constr / Angles_Deg V_Norm z_orig pts_o
    Cg dx dy ang_pca c_pre s_pre c s pts_rot x1 x2 x3 x4 y1 y2 y3 y4 sq_x1 sq_x2 sq_x3 sq_x4 D
    ka kb kc max_err err Xv Yv p_dist pts_temp V_wcs F_wcs l_ents unique_count is_geom_twin pr
    ocessed_angles)
  (setq Angles_Deg (Geom_Find_Parabolic_Angles P1 P2 P3 P4)
        V_Norm     (Vet_Uni (Vet_Cro (Vet_Sub P2 P1) (Vet_Sub P3 P1)))
        z_orig     (caddr (trans P1 0 V_Norm))
        pts_o      (Util_Trans_Pts_List (list P1 P2 P3 P4) 0 V_Norm)
        Cg         (Pu_Baricentro2D_LisPu pts_o)
        pts_o      (Util_Translate_Pts_Relative pts_o Cg))
```

```
(mapcar 'set '(dx dy) (Util_Get_Pts_BBox_Size pts_o))
(setq ang_pca (if (> dy dx) (/ pi 2.0) (angle '(0 0) (Vet_Sub (nth 2 pts_o) (nth 0 pts_o))))
      c_pre   (cos ang_pca)
      s_pre   (sin ang_pca)
      pts_o   (Util_Rot_List_2D_CS pts_o c_pre s_pre)
      l_ents  nil
      processed_angles nil
      unique_count 0)
(if Angles_Deg
    (foreach Theta_Deg Angles_Deg
      (setq c       (cos (* Theta_Deg (/ pi 180.0)))
            s       (sin (* Theta_Deg (/ pi 180.0)))
            pts_rot (Util_Rot_List_2D_CS pts_o c s))
      (mapcar 'set '(x1 x2 x3 x4)             (Math_Extract_Coord_Pow pts_rot 0 1))
      (mapcar 'set '(y1 y2 y3 y4)             (Math_Extract_Coord_Pow pts_rot 1 1))
      (mapcar 'set '(sq_x1 sq_x2 sq_x3 sq_x4) (Math_Extract_Coord_Pow pts_rot 0 2))
      (setq D (Math_Cyclic_Det sq_x1 sq_x2 sq_x3 x1 x2 x3))
      (if (> (abs D) 1e-9)
          (progn
            (setq ka (/ (Math_Cyclic_Det y1 y2 y3 x1 x2 x3) D)
                  kb (/ (Math_Cyclic_Det sq_x1 sq_x2 sq_x3 y1 y2 y3) D)
                  kc (/ (+ (* sq_x1 (Math_Diff_Prod x2 y3 x3 y2))
                           (* sq_x2 (Math_Diff_Prod x3 y1 x1 y3))
                           (* sq_x3 (Math_Diff_Prod x1 y2 x2 y1))) D)
                  max_err 0.0)
            (foreach p_check pts_rot
              (setq err (abs (- (+ (Math_Sum_Prod ka (Math_Sqr (car p_check))
   kb (car p_check)) kc) (cadr p_check))))
              (if (> err max_err) (setq max_err err)))
            (if (< max_err 1e-2)
                (progn
                  (setq is_geom_twin (vl-some '(lambda (pa) (< (abs (- (abs (- pa Theta_Deg))
   180.0)) 2.0)) processed_angles))
                  (if (not is_geom_twin)
                      (progn
                        (setq unique_count     (1+ unique_count)
                              processed_angles (cons Theta_Deg processed_angles))
                        (princ (strcat "\n[DRAW] Generazione Parabola " (itoa unique_count)
   "... "))
                        (setq Xv       (/ (- kb) (* 2.0 ka))
                              Yv       (- kc (/ (* kb kb) (* 4.0 ka)))
                              p_dist   (/ 1.0 (* 4.0 ka))
                              pts_temp (Util_Rot_List_2D_Inv_CS (Util_Rot_List_2D_Inv_CS (lis
   t (list Xv Yv) (list Xv (+ Yv p_dist))) c s) c_pre s_pre)
                              V_wcs    (trans (Vet_Sum (list (car (nth 0 pts_temp))
```

```
    (cadr (nth 0 pts_temp)) z_orig) Cg) V_Norm 0)
                              F_wcs    (trans (Vet_Sum (list (car (nth 1 pts_temp))
    (cadr (nth 1 pts_temp)) z_orig) Cg) V_Norm 0)
                              l_ents   (append l_ents (Draw_ConPar_VeFu_Compl V_wcs F_wcs (li
    st P1 P2 P3 P4))))
                         (princ "Fatto.")))))))))
     (princ "\n[DRAW] Nessuna parabola trovata."))
  l_ents)
;;-------------------------------------------------------------------------------------
(defun Draw_Conic_4P_AxisDir (Pu1 Pu2 Pu3 Pu4 PuD1 PuD2 lay_crv lay_constr lay_pts draw_input_g
    eom / l_constr norm_data PuC max_dist P1n P2n P3n P4n K1 K2 ang_axis sin2 cos2 A1 B1 C1 A2
    B2 C2 num den lam K_final delta l_curve found_type)
  (setq l_constr (if draw_input_geom (list (Draw_Point_R PuD1 lay_pts)
                                     (Draw_Point_R PuD2 lay_pts)
                                     (Draw_Line_2Pu_R PuD1 PuD2 lay_constr)) nil)
       norm_data (Util_Normalize_Pts_Set (list Pu1 Pu2 Pu3 Pu4))
       PuC (car norm_data)
       max_dist (cadr norm_data))
  (mapcar 'set '(P1n P2n P3n P4n) (caddr norm_data))
  (setq K1 (Math_Multiply_Lines_To_Conic (Math_Line_General_Eq P1n P2n)
    (Math_Line_General_Eq P3n P4n))
       K2 (Math_Multiply_Lines_To_Conic (Math_Line_General_Eq P1n P3n)
    (Math_Line_General_Eq P2n P4n))
       ang_axis (angle PuD1 PuD2)
       sin2 (sin (* 2.0 ang_axis))
       cos2 (cos (* 2.0 ang_axis)))
  (mapcar 'set '(A1 B1 C1) K1)
  (mapcar 'set '(A2 B2 C2) K2)
  (setq num (Math_Diff_Prod B1 cos2 (- A1 C1) sin2)
       den (Math_Diff_Prod (- A2 C2) sin2 B2 cos2))
  (if (< (abs den) 1e-9)
      (list 'INDETERMINATO l_constr)
      (progn
        (setq lam (/ num den)
              K_final (Math_Combine_Conics K1 K2 lam)
              delta (- (Math_Sqr (nth 1 K_final)) (* 4.0 (nth 0 K_final) (nth 2 K_final))))
        (cond ((< delta -1e-9)  (setq found_type 'ELLISSE
                                     l_curve (Solve_And_Draw_From_Coeffs K_final PuC lay_crv l
    ay_constr 'ELLISSE max_dist)))
              ((> delta 1e-9)   (setq found_type 'IPERBOLE
                                     l_curve (Solve_And_Draw_From_Coeffs K_final PuC lay_crv l
    ay_constr 'IPERBOLE max_dist)))
              (T                (setq found_type 'PARABOLA
                                     l_curve nil)))
```

```
        (list found_type (append l_constr l_curve)))))
;;-------------------------------------------------------------------------------------
(defun Draw_Con5P_Nor (pts pts_limit full_mode lay_crv lay_constr / G_bary max_dist pts_norm an
     alyze_res conic_type best_res created_ents)
  (mapcar 'set '(G_bary max_dist pts_norm) (Util_Normalize_Pts_Set pts))
  (if (setq analyze_res (analyze-conic-geometry pts_norm))
      (progn
        (setq conic_type (car analyze_res))
        (cond
          ((eq conic_type 'PARABOLA)
           (setq created_ents (Draw_ConPar_5P_Sub pts_norm analyze_res G_bary max_dist pts_limi
     t full_mode)))

          ((or (eq conic_type 'ELLISSE) (eq conic_type 'IPERBOLE))
           (if full_mode (princ (strcat "\n[MATRICE] Torneo Combinatorio (" (vl-symbol-name con
     ic_type) ")...")))
           (setq best_res (Alg_Tournament_5P_Centered pts (Util_Denorm_Pt (cadr analyze_res)
     G_bary max_dist) conic_type))
           (if best_res
               (setq created_ents
                     (apply 'Draw_ConCen_3P
                            (append best_res (list (if full_mode pts pts_limit)
     full_mode)))))))
        (Util_Conv_List_To_VLA created_ents))
      (if full_mode (Util_Print_And_Nil "\n[WARN] Analisi preliminare fallita.") nil)))
;;-------------------------------------------------------------------------------------
(defun Draw_Con5P_Nor_Compl (pts lay_crv lay_constr pts_limit)
     (Draw_Con5P_Nor pts pts_limit T lay_crv lay_constr))
;;-------------------------------------------------------------------------------------
(defun Draw_Con5P_Nor_Light (pts pts_limit)
     (Draw_Con5P_Nor pts pts_limit nil nil nil))

)
;;-------------------------------------------------------------------------------------
(defun Util_Capture_After_Entity (ent_start lay_target / ename new_objs ent_data obj)
  (setq new_objs nil)
  (if ent_start
      (setq ename (entnext ent_start))
      (setq ename (entnext)))
  (while ename
    (setq ent_data (entget ename))
    (if (and (= (strcase (cdr (assoc 8 ent_data))) (strcase lay_target))
             (member (cdr (assoc 0 ent_data)) '("LINE" "XLINE" "SPLINE" "ELLIPSE")))
        (progn
          (setq obj (vlax-ename->vla-object ename))
```

```
          (setq new_objs (cons obj new_objs))))
    (setq ename (entnext ename)))
  (reverse new_objs))
;;-------------------------------------------------------------------------------------
(defun Geom_Get_Projected_Center (pt_ucs vla_list extend / pt_wcs best_pt_wcs min_dist proj_pt_
     wcs dist pt_ret)
  (setq pt_wcs      (trans pt_ucs 1 0)
        best_pt_wcs pt_wcs
        min_dist    1e99)
  (foreach obj vla_list
    (if (not (vl-catch-all-error-p
               (setq proj_pt_wcs (vl-catch-all-apply 'vlax-curve-getClosestPointTo (list obj pt
     _wcs extend)))))
        (if proj_pt_wcs
            (progn
              (setq dist (distance pt_wcs proj_pt_wcs))
              (if (< dist min_dist)
                  (setq min_dist    dist
                        best_pt_wcs proj_pt_wcs))))))
  (setq pt_ret (trans best_pt_wcs 0 1))
  pt_ret)
;;-------------------------------------------------------------------------------------
(defun Geom_Draw_Trapezoid_Smart (pts_mid v_dir pts_ref lay_locus / p1 p2 p3 p4 m1 m2 dist_base
     center v_axis pts_ends)
  (mapcar 'set '(p1 p2 p3 p4) pts_ref)
  (if (P_Are_Vectors_Parallel_2D (mapcar '- p2 p1) v_dir 1e-3)
      (setq m1 (Vet_Mid p1 p2) m2 (Vet_Mid p3 p4))
      (setq m1 (Vet_Mid p1 p4) m2 (Vet_Mid p2 p3)))
  (setq dist_base (distance m1 m2)
        center    (Vet_Mid m1 m2)
        v_axis    (Vet_Uni (mapcar '- m2 m1))
        pts_ends  (Geom_2Pu_Simm_PuC center v_axis (* dist_base 12.0)))
  (if pts_ends (Draw_Line_2Pu_R (car pts_ends) (cadr pts_ends) lay_locus)))
;;-------------------------------------------------------------------------------------
(defun Geom_Ipe_4P_Eq (Pu1 Pu2 Pu3 Pu4 / PuC_Ref P1n P2n P3n P4n K1 K2 A1 B1 C1 A2 B2 C2 lam K_
     final Geom_Data PuC V_dir a b)
  (setq PuC_Ref (Pu_Baricentro2D_LisPu (list Pu1 Pu2 Pu3 Pu4)))
  (mapcar 'set '(P1n P2n P3n P4n) (Util_Translate_Pts_Relative (list Pu1 Pu2 Pu3 Pu4) PuC_Ref))
  (setq K1 (Math_Multiply_Lines_To_Conic (Math_Line_General_Eq P1n P2n)
     (Math_Line_General_Eq P3n P4n))
        K2 (Math_Multiply_Lines_To_Conic (Math_Line_General_Eq P1n P3n)
     (Math_Line_General_Eq P2n P4n)))
  (mapcar 'set '(A1 B1 C1) K1)
  (mapcar 'set '(A2 B2 C2) K2)
  (if (< (abs (+ A2 C2)) 1e-9)
```

```
      (setq K_final K2)
      (setq lam (/ (- (+ A1 C1)) (+ A2 C2))
            K_final (Math_Combine_Conics K1 K2 lam)))
  (setq Geom_Data (Math_Extract_Equilateral_Data K_final))
  (if Geom_Data
      (progn
        (setq PuC   (Vet_Sum (car Geom_Data) PuC_Ref)
              V_dir (cadr Geom_Data)
              a     (caddr Geom_Data)
              b     a)
        (list PuC V_dir a b))
      nil))
;;-------------------------------------------------------------------------------------
(defun Geom_Conic_Center_Proprio_Action (p1 p2 p3 p4 / cen pts best_err best_res i j k triplet
     pA pB pC coeffs A B C delta err_curr pt_test x y val_test)
  (princ "\nSpecifica il Centro della conica: ")
  (setq cen (getpoint))
  (if cen
    (progn
      (Draw_Point_R cen *LaCo_Punti_scelti*)
      (setq pts      (list p1 p2 p3 p4)
            best_err 1.0e99
            best_res nil
            i        0)
      (princ "\n[SYSTEM] Torneo 3-su-4 per Centro Fisso...")
      (while (< i 2)
        (setq j (+ i 1))
        (while (< j 3)
          (setq k (+ j 1))
          (while (< k 4)
            (setq pA (nth i pts) pB (nth j pts) pC (nth k pts))
            (setq coeffs (Alg3x3_Calcola_Coeffs cen pA pB pC))
            (if coeffs
              (progn
                (mapcar 'set '(A B C) coeffs)
                (setq err_curr 0.0)
                (foreach idx (list 0 1 2 3)
                  (if (and (/= idx i) (/= idx j) (/= idx k))
                      (progn
                        (setq pt_test (mapcar '- (nth idx pts) cen))
                        (mapcar 'set '(x y) pt_test)
                        (setq val_test (+ (* A x x) (* B x y) (* C y y))
                              err_curr (abs (- val_test 1.0))))))
                (if (< err_curr best_err)
                    (setq best_err err_curr
```

```
                        best_res (list pA pB pC)))))
            (setq k (1+ k)))
          (setq j (1+ j)))
        (setq i (1+ i)))
      (if (and best_res (< best_err 1.0))
          (progn
            (princ (strcat " -> Vincitore (Err: " (rtos best_err 2 6) ")."))
            (Draw_ConCen_3P_Compl cen (car best_res) (cadr best_res) (caddr best_res)  pts))
          (princ "\n[ERRORE] Geometria impossibile per il centro dato.")))
    (princ "\nNessun centro selezionato."))
  (princ))
;;-----------------------------------------------------------------------------------------
(defun Geom_MinMax_Dist_LisPu (pts ref_pt opz / min_val max_val min_pa min_pb max_pa max_pb p1
     p2 d tail)
  (setq min_val 1e99
        max_val -1.0)

  (if ref_pt
    (foreach p2 pts
      (setq d (distance ref_pt p2))
      (if (< d min_val) (setq min_val d min_pa ref_pt min_pb p2))
      (if (> d max_val) (setq max_val d max_pa ref_pt max_pb p2)))
    (while (cdr pts)
      (setq p1   (car pts)
            tail (cdr pts))
      (foreach p2 tail
        (setq d (distance p1 p2))
        (if (< d min_val) (setq min_val d min_pa p1 min_pb p2))
        (if (> d max_val) (setq max_val d max_pa p1 max_pb p2)))
      (setq pts (cdr pts))))

  (cond
    ((= opz 1) min_val)
    ((= opz 2) max_val)
    ((= opz 3) (list min_pa min_pb))
    ((= opz 4) (list max_pa max_pb))
    ((= opz 5) (list min_val max_val))
    ((= opz 6) (list (list min_pa min_pb)         (list max_pa max_pb)))
    ((= opz 7) (list (list min_pa min_pb min_val) (list max_pa max_pb max_val)))))
;;-----------------------------------------------------------------------------------------
(defun Geom_Conic_Action_Par_Direct (pts / topo_type p_ord Pu1 Pu2 Pu3 Pu4)
  (mapcar 'set '(topo_type p_ord) (apply 'Math_Classify_Forma_4Pu pts))

  (if (or (wcmatch (vl-symbol-name topo_type) "CONCAVO*")
          (wcmatch (vl-symbol-name topo_type) "DEGENERE*")
```

```
          (member topo_type '(QUADRATO RETTANGOLO ROMBO PARALLELOGRAMMA)))
      (princ (strcat "\n[INFO] Configurazione " (vl-symbol-name topo_type)
     ": Parabola impossibile."))

      (progn
        (mapcar 'set '(Pu1 Pu2 Pu3 Pu4) p_ord)
        (Draw_ConPar_4P_ConVer_R Pu1 Pu3 Pu2 Pu4 *LaCo_Tracciati* *LaCo_Costruzioni*)))
  (princ))
;;-----------------------------------------------------------------------------------
(defun Geom_Scan_3P_Parabolas_Adaptive (pts max_angle molt / centroid d_max ang step raw_data o
     rigin deg2rad dir_vec ang_rad
                                                    pts_extra solve_res p_data vtx d_cen d_clo p )
  (setq centroid   (Pu_Baricentro2D_LisPu pts)
        d_max      (Geom_MinMax_Dist_LisPu pts nil 2)
        ang        0.0
       origin    '(0.0 0.0 0.0)
       deg2rad    (/ pi 180.0)
       molt      molt)
  (while (<= ang max_angle)
    (setq ang_rad                (* ang deg2rad)
          dir_vec                (list (cos ang_rad) (sin ang_rad) 0.0)
          pts_extra              (Calcola_Parabola pts (list origin dir_vec))
          solve_res (if pts_extra (Solve_Par_Geometry (append pts pts_extra) origin dir_vec))
          p_data    (if solve_res (list (car solve_res) (cadr solve_res) ang_rad))
          raw_data  (if p_data    (cons p_data raw_data) raw_data)
          vtx                    (car p_data)
         d_cen     (if vtx       (distance centroid vtx))
          d_clo     (if vtx       (Util_Get_Min_Dist_From_Point pts vtx)))

        (setq step (if (and d_cen d_clo)
                     (* molt (cond ((<  d_clo (* d_max 0.05)) 0.40)
                                   ((<  d_clo (* d_max 0.10)) 0.50)
                                   ((<  d_clo (* d_max 0.15)) 0.60)
                                   ((<  d_clo (* d_max 0.20)) 0.70)
                                   ((<  d_clo (* d_max 0.25)) 0.80)
                                   ((<  d_cen (* d_max 1.0))  0.80)
                                   ((<  d_cen (* d_max 2.0))  0.40)
                                   ((<  d_cen (* d_max 4.0))  0.20)
                                   ((>= d_cen (* d_max 4.0))  0.10)))
                   0.25))
    (setq ang (+ ang step)))
  (reverse raw_data))
;;-----------------------------------------------------------------------------------
(defun Geom_Refine_Internal_Scan (a_start a_end n_steps pts_3 pt_target / step a_curr res dir_v
     ec pts_extra solve_res V d origin)
```

```
  (setq origin '(0.0 0.0 0.0) step (/ (- a_end a_start) n_steps))
  (if (equal step 0.0 1e-15) (setq step 1e-15))
  (setq a_curr a_start res '())
  (repeat (fix (+ n_steps 1))
    (setq dir_vec (list (cos a_curr) (sin a_curr) 0.0))
    (setq pts_extra (Calcola_Parabola pts_3 (list origin dir_vec)))
    (if (and pts_extra (car pts_extra))
        (progn
           (setq solve_res (Solve_Par_Geometry (append pts_3 pts_extra) origin dir_vec))
           (if solve_res
               (progn
                 (setq V (car solve_res) d (distance V pt_target))
                 (setq res (cons (list d a_curr V (cadr solve_res)) res))))))
    (setq a_curr (+ a_curr step)))
  (vl-sort res (function (lambda (e1 e2) (< (cadr e1) (cadr e2))))))
;;-------------------------------------------------------------------------------------
(defun Geom_Refine_Internal_Bracket (res_list / best_i min_d i curr_d len_res prev_item next_it
     em best_item)
  (setq len_res (length res_list))
  (if (< len_res 3) nil
    (progn
      (setq min_d 1e99 best_i 0 i 0)
      (foreach item res_list
         (setq curr_d (car item))
         (if (< curr_d min_d) (setq min_d curr_d best_i i))
         (setq i (1+ i)))
      (setq best_item (nth best_i res_list))
      (setq prev_item (if (> best_i 0) (nth (1- best_i) res_list) (nth 0 res_list)))
      (setq next_item (if (< best_i (1- len_res)) (nth (1+ best_i) res_list) (last res_list)))
      (list (cadr prev_item) (cadr next_item) (distance (caddr prev_item) (caddr next_item))
     best_item))))
;;-------------------------------------------------------------------------------------
(defun Geom_Refine_Internal_Recursive (ang_start ang_end depth max_depth steps pts_3 pt_target
     threshold / results bracket_data spatial_width winner)
  (setq results (Geom_Refine_Internal_Scan ang_start ang_end steps pts_3 pt_target))
  (setq bracket_data (Geom_Refine_Internal_Bracket results))
  (if bracket_data
    (progn
      (setq spatial_width (caddr bracket_data) winner (cadddr bracket_data))
      (if (or (< spatial_width threshold) (>= depth max_depth))
        (list (caddr winner) (cadddr winner) (cadr winner))
        (Geom_Refine_Internal_Recursive (car bracket_data) (cadr bracket_data) (1+ depth)
     max_depth steps pts_3 pt_target threshold)))
    nil))
;;-------------------------------------------------------------------------------------
```

```
(defun Geom_Refine_Par_Topological (pt_target spline_obj branch_data pts_3 / len threshold best
     _idx min_dist_global idx d item_best ang_best ang_start ang_end test_norm test_twin d_norm
     d_twin origin dir_norm dir_twin)
  (setq len        (length branch_data)
       origin    '(0.0 0.0 0.0))
  (setq threshold  (/ (Geom_MinMax_Dist_LisPu pts_3 nil 1) 100000.0))
  (setq best_idx        0
       min_dist_global 1e99
       idx             0)
  (foreach item branch_data
    (setq d (distance (car item) pt_target))
    (if (< d min_dist_global) (setq min_dist_global d best_idx idx))
    (setq idx (1+ idx)))
  (setq item_best (nth best_idx branch_data)
       ang_best  (caddr item_best))
  (setq dir_norm  (list (cos ang_best) (sin ang_best) 0.0))
  (setq test_norm (Solve_Par_Geometry (append pts_3 (Calcola_Parabola pts_3 (list origin dir_no
     rm))) origin dir_norm))
  (setq d_norm    (if test_norm (distance (car test_norm) pt_target) 1e99))
  (setq ang_best  (+ ang_best pi)
       dir_twin  (list (cos ang_best) (sin ang_best) 0.0))
  (setq test_twin (Solve_Par_Geometry (append pts_3 (Calcola_Parabola pts_3 (list origin dir_tw
     in))) origin dir_twin))
  (setq d_twin    (if test_twin (distance (car test_twin) pt_target) 1e99))
  (if (< d_norm d_twin) (setq ang_best (- ang_best pi)))
  (setq ang_start (- ang_best 0.1) ang_end (+ ang_best 0.1))
  (Geom_Refine_Internal_Recursive ang_start ang_end 1 15 10.0 pts_3 pt_target threshold))
;;-------------------------------------------------------------------------------------
(defun Util_Process_Branches_Data (data_list dist_limit / branches curr_branch last_V V_curr se
     g_first seg_last p_start p_end p_next p_prev d_gap d_step_in d_step_out merged_branch midd
     le_branches)
  (setq branches nil curr_branch nil last_V nil)
  (foreach item data_list
    (setq V_curr (car item))
    (if last_V
      (if (> (distance V_curr last_V) dist_limit)
        (progn
          (if curr_branch (setq branches (cons (reverse curr_branch) branches)))
          (setq curr_branch nil))))
    (setq curr_branch (cons item curr_branch) last_V V_curr))
  (if curr_branch (setq branches (cons (reverse curr_branch) branches)))
  (setq branches (reverse branches))
  (if (>= (length branches) 2)
    (progn
       (setq seg_first (car branches) seg_last (last branches))
```

```
       (if (and (>= (length seg_first) 2) (>= (length seg_last) 2))
         (progn
            (setq p_start (car (nth 0 seg_first)) p_next (car (nth 1 seg_first)))
            (setq p_end (car (last seg_last)) p_prev (car (cadr (reverse seg_last))))
            (setq d_gap (distance p_end p_start) d_step_in (distance p_start p_next)
     d_step_out (distance p_prev p_end))
            (if (< d_gap (* 3.0 (max d_step_in d_step_out)))
                (progn
                   (princ "\n[INFO] Saldatura Ciclica attivata (Loop Chiuso).")
                   (setq merged_branch (append seg_last seg_first))
                   (if (= (length branches) 2)
                       (setq branches (list merged_branch))
                       (progn
                          (setq middle_branches (cdr branches)
     middle_branches (reverse (cdr (reverse middle_branches))))
                          (setq branches (cons merged_branch middle_branches)))))
                nil)))))
  branches)
;;-------------------------------------------------------------------------------------
(defun Geom_Refine_Internal_Scan_Fuochi (a_start a_end n_steps pts_3 pt_target / step a_curr re
     s dir_vec pts_extra solve_res F d origin)
  (setq origin '(0.0 0.0 0.0) step (/ (- a_end a_start) n_steps))
  (if (equal step 0.0 1e-15) (setq step 1e-15))
  (setq a_curr a_start res '())
  (repeat (fix (+ n_steps 1))
    (setq dir_vec (list (cos a_curr) (sin a_curr) 0.0))
    (setq pts_extra (Calcola_Parabola pts_3 (list origin dir_vec)))
    (if (and pts_extra (car pts_extra))
        (progn
           (setq solve_res (Solve_Par_Geometry (append pts_3 pts_extra) origin dir_vec))
           (if solve_res
               (progn
                 (setq F (cadr solve_res) d (distance F pt_target))
                 (setq res (cons (list d a_curr (car solve_res) F) res))))))
    (setq a_curr (+ a_curr step)))
  (vl-sort res (function (lambda (e1 e2) (< (cadr e1) (cadr e2))))))
;;-------------------------------------------------------------------------------------
(defun Geom_Refine_Internal_Bracket_Fuochi (res_list / best_i min_d i curr_d len_res prev_item
     next_item best_item)
  (setq len_res (length res_list))
  (if (< len_res 3) nil
    (progn
      (setq min_d 1e99 best_i 0 i 0)
      (foreach item res_list
         (setq curr_d (car item))
```

```
        (if (< curr_d min_d) (setq min_d curr_d best_i i))
        (setq i (1+ i)))
      (setq best_item (nth best_i res_list))
      (setq prev_item (if (> best_i 0) (nth (1- best_i) res_list) (nth 0 res_list)))
      (setq next_item (if (< best_i (1- len_res)) (nth (1+ best_i) res_list) (last res_list)))
      (list (cadr prev_item) (cadr next_item) (distance (cadddr prev_item) (cadddr next_item))
     best_item))))
;;-------------------------------------------------------------------------------------
(defun Geom_Refine_Internal_Recursive_Fuochi (ang_start ang_end depth max_depth steps pts_3 pt_
     target threshold / results bracket_data spatial_width winner)
  (setq results (Geom_Refine_Internal_Scan_Fuochi ang_start ang_end steps pts_3 pt_target))
  (setq bracket_data (Geom_Refine_Internal_Bracket_Fuochi results))
  (if bracket_data
    (progn
      (setq spatial_width (caddr bracket_data) winner (cadddr bracket_data))
      (if (or (< spatial_width threshold) (>= depth max_depth))
        (list (caddr winner) (cadddr winner) (cadr winner))
        (Geom_Refine_Internal_Recursive_Fuochi (car bracket_data) (cadr bracket_data)
     (1+ depth) max_depth steps pts_3 pt_target threshold)))
    nil))
;;-------------------------------------------------------------------------------------
(defun Geom_Refine_Par_Topological_Fuochi (pt_target spline_obj branch_data pts_3 / len thresho
     ld best_idx min_dist_global idx d item_best ang_best ang_start ang_end test_norm test_twin
     d_norm d_twin origin dir_norm dir_twin)
  (setq len        (length branch_data)
        origin    '(0.0 0.0 0.0))
  (setq threshold  (/ (Geom_MinMax_Dist_LisPu pts_3 nil 1) 100000.0))
  (setq best_idx        0
        min_dist_global 1e99
        idx             0)
  (foreach item branch_data
    (setq d (distance (cadr item) pt_target))
    (if (< d min_dist_global) (setq min_dist_global d best_idx idx))
    (setq idx (1+ idx)))
  (setq item_best (nth best_idx branch_data)
        ang_best  (caddr item_best))

  (setq dir_norm  (list (cos ang_best) (sin ang_best) 0.0))
  (setq test_norm (Solve_Par_Geometry (append pts_3 (Calcola_Parabola pts_3 (list origin dir_no
     rm))) origin dir_norm))
  (setq d_norm    (if test_norm (distance (cadr test_norm) pt_target) 1e99))

  (setq ang_best  (+ ang_best pi)
        dir_twin  (list (cos ang_best) (sin ang_best) 0.0))
  (setq test_twin (Solve_Par_Geometry (append pts_3 (Calcola_Parabola pts_3 (list origin dir_tw
```

```
    in))) origin dir_twin))
  (setq d_twin    (if test_twin (distance (cadr test_twin) pt_target) 1e99))

  (if (< d_norm d_twin) (setq ang_best (- ang_best pi)))
  (setq ang_start (- ang_best 0.1) ang_end (+ ang_best 0.1))

  (Geom_Refine_Internal_Recursive_Fuochi ang_start ang_end 1 15 10.0 pts_3 pt_target threshold)
    )
;;-------------------------------------------------------------------------------------
(defun Util_Process_Branches_Data_SmartStitch (raw_data dist_limit / branches p_first p_second
    p_last p_prev d_closing d_start d_end d_ref factor merged_branch middle_branches seg_start
    seg_end)
  (setq branches (Util_Scan_And_Cut_Branches_Data_By_Focus raw_data dist_limit))

  (if (>= (length branches) 2)
      (progn
        (setq seg_start (car branches)
              seg_end   (last branches))
        (if (and (>= (length seg_start) 2) (>= (length seg_end) 2))
            (progn
              (setq p_first   (cadr (nth 0 seg_start))
                    p_second  (cadr (nth 1 seg_start))
                    p_last    (cadr (last seg_end))
                    p_prev    (cadr (cadr (reverse seg_end)))
                    d_closing (distance p_last p_first)
                    d_start   (distance p_first p_second)
                    d_end     (distance p_prev p_last)
                    d_ref     (max d_start d_end)
                    factor    3.0)
              (if (< d_closing (* d_ref factor))
                  (progn
                    (princ (strcat "\n[INFO] Saldatura Smart Fuochi: Gap (" (rtos d_closing 2 2
    ) ") unito."))
                    (setq merged_branch (append seg_end seg_start))
                    (if (= (length branches) 2)
                        (setq branches (list merged_branch))
                        (setq middle_branches (cdr branches)
                              middle_branches (reverse (cdr (reverse middle_branches)))
                              branches        (cons merged_branch middle_branches)))))))))
  branches)
;;-------------------------------------------------------------------------------------
(defun Util_Scan_And_Cut_Branches_Data_By_Focus (pt_list dist_limit / branches curr_branch last
    _pt cut_now focus)
  (setq branches nil curr_branch nil last_pt nil)
  (foreach pt_packet pt_list
```

```
    (setq cut_now nil focus (cadr pt_packet))
    (if last_pt
        (if (> (distance focus last_pt) dist_limit) (setq cut_now T)))
    (if cut_now
        (progn
          (if curr_branch (setq branches (cons (reverse curr_branch) branches)))
          (setq curr_branch nil)))
    (setq curr_branch (cons pt_packet curr_branch)
          last_pt      focus))
  (if curr_branch (setq branches (cons (reverse curr_branch) branches)))
  (reverse branches))
;;-------------------------------------------------------------------------------------
(defun Geom_Conic_Action_MinEccentricity (pts / norm_data PuC max_dist P1n P2n P3n P4n K1 K2
                                          A1 B1 C1 A2 B2 C2 d1 d2 s1 s2 ap bp cp aq bq cq
                                          ka kb kc discr lams best_lam best_score score K_final
delta)
  (setq norm_data (Util_Normalize_Pts_Set pts)
        PuC (car norm_data)
        max_dist (cadr norm_data))
  (mapcar 'set '(P1n P2n P3n P4n) (caddr norm_data))
  (setq K1 (Math_Multiply_Lines_To_Conic (Math_Line_General_Eq P1n P2n)
     (Math_Line_General_Eq P3n P4n))
        K2 (Math_Multiply_Lines_To_Conic (Math_Line_General_Eq P1n P3n)
     (Math_Line_General_Eq P2n P4n)))
  (mapcar 'set '(A1 B1 C1) K1)
  (mapcar 'set '(A2 B2 C2) K2)

  (setq d1 (- A1 C1) d2 (- A2 C2)
        s1 (+ A1 C1) s2 (+ A2 C2))
  (setq ap (+ (Math_Sqr d2) (Math_Sqr B2))
        bp (* 2.0 (+ (* d1 d2) (* B1 B2)))
        cp (+ (Math_Sqr d1) (Math_Sqr B1))
        aq (Math_Sqr s2)
        bq (* 2.0 s1 s2)
        cq (Math_Sqr s1))

  (setq ka (- (* ap bq) (* aq bp))
        kb (* 2.0 (- (* ap cq) (* aq cp)))
        kc (- (* bp cq) (* cp bq)))

  (setq lams nil)
  (if (< (abs ka) 1e-9)
      (if (> (abs kb) 1e-9) (setq lams (list (/ (- kc) kb))))
      (progn
        (setq discr (- (Math_Sqr kb) (* 4.0 ka kc)))
```

```
        (if (>= discr 0.0)
            (setq lams (list (/ (+ (- kb) (sqrt discr)) (* 2.0 ka))
                             (/ (- (- kb) (sqrt discr)) (* 2.0 ka)))))))

  (setq best_lam nil best_score 1e99)
  (foreach lam lams
    (setq K_final (Math_Combine_Conics K1 K2 lam))
    (setq delta (- (Math_Sqr (nth 1 K_final)) (* 4.0 (nth 0 K_final) (nth 2 K_final))))
    (if (< delta -1e-9)
        (progn
           (setq score (/ (+ (Math_Sqr (- (nth 0 K_final) (nth 2 K_final)))
     (Math_Sqr (nth 1 K_final)))
                          (Math_Sqr (+ (nth 0 K_final) (nth 2 K_final)))))
           (if (< score best_score)
               (setq best_score score best_lam lam)))))
  (if best_lam
      (progn
        (setq K_final (Math_Combine_Conics K1 K2 best_lam))
        (if (< best_score 1e-6)
            (princ "\n[MODE] Trovato CERCHIO (I punti sono conciclici!).")
            (princ "\n[MODE] Trovata ELLISSE PAFFUTA (Minima eccentricita' nel fascio)."))
        (Solve_And_Draw_From_Coeffs K_final PuC nil nil 'ELLISSE max_dist))
       (princ "\n[WARN] Nessuna ellisse di minima eccentricita' determinabile in questa configur
      azione.")))
;;-------------------------------------------------------------------------------------------
(defun Util_Get_All_Pairs (lst / res sub)
  (while (cdr lst)
    (setq sub (cdr lst))
    (foreach item sub
      (setq res (cons (list (car lst) item) res)))
    (setq lst sub))
  (reverse res))
;;-------------------------------------------------------------------------------------------
(defun Lis_2nth (id1 id2 lisPu)  (list (nth id1 lisPu) (nth id2 lisPu)))
;;-------------------------------------------------------------------------------------------
(defun Lis_Coppie_nth (lisco lisPu) (mapcar '(lambda (coppia) (Lis_2nth (car coppia)
     (cadr coppia) lisPu)) lisco))
;;-------------------------------------------------------------------------------------------
(defun Util_Collect_Ents (last_ent / ent lst)
  (setq ent last_ent)
  (while (setq ent (entnext ent)) (setq lst (cons ent lst)))
  (reverse lst))
;;-------------------------------------------------------------------------------------------
(defun Util_Del_Ents_List (lst_ents)  (foreach ent lst_ents (if (entget ent) (entdel ent))))
;;-------------------------------------------------------------------------------------------
```

```
(defun Util_Select_Point_From_List (msg lst_pts / sel ent_data pt match)
  (while (not match)
    (setq sel (entsel (strcat "\n" msg)))
    (if sel
      (progn
        (setq ent_data (entget (car sel)))
        (if (= (cdr (assoc 0 ent_data)) "POINT")
          (progn
            (setq pt (cdr (assoc 10 ent_data)))
            (if (vl-some '(lambda (p) (< (distance p pt) 1e-6)) lst_pts)
              (setq match pt)
              (princ "\n[!] Errore: Il punto selezionato non appartiene al sistema ortocentrico
     ."))))
          (princ "\n[!] Errore: Seleziona un'entità di tipo PUNTO.")))
      (setq match 'cancel)))
  (if (eq match 'cancel) nil match))
;;-----------------------------------------------------------------------------------------
(defun Util_Select_Winning_Hypothesis (L1 L2 L3 / sel ent match)
  (while (not match)
    (setq sel (entsel "\nSeleziona l'ipotesi corretta (clicca su un punto blu o su un vettore):
     "))
    (if sel
      (progn
        (setq ent (car sel))
        (cond   ((vl-position ent L1) (setq match 1))
                ((vl-position ent L2) (setq match 2))
                ((vl-position ent L3) (setq match 3))
          (T (princ "\n[!] Errore: L'oggetto non fa parte delle ipotesi proposte."))))
      (setq match 'cancel)))
  (if (eq match 'cancel) nil match))
;;-----------------------------------------------------------------------------------------
(defun PrincPu (txt pu dec)
  (princ (if (caddr pu) (strcat txt (rtos (car pu) 2 dec) " , " (rtos (cadr pu) 2 dec)
     " , " (rtos (caddr pu) 2 dec))
                        (strcat txt (rtos (car pu) 2 dec) " , " (rtos (cadr pu) 2 dec)))))
;;-----------------------------------------------------------------------------------------
(defun Util_Del_Ents_At_Pt (pt layer / d p1 p2 ss)
  (setq d 1e-4)
  (setq p1 (list (- (car pt) d) (- (cadr pt) d) 0.0)
        p2 (list (+ (car pt) d) (+ (cadr pt) d) 0.0))

  (if (setq ss (ssget "_C" p1 p2 (list '(0 . "POINT") (cons 8 layer))))
      (command "_.ERASE" ss ""))
  (princ))
;;-----------------------------------------------------------------------------------------
```

```
(defun Safe_Collect_Ents (e_start / e_curr lst)
  (setq e_curr (if e_start (entnext e_start) (entnext)) lst nil)
  (while e_curr
    (setq lst (cons e_curr lst) e_curr (entnext e_curr)))
  lst)
;;-------------------------------------------------------------------------------------------
(defun Util_Update_Guides_Vis (stato_opt lista_ent / mode e)
  (setq mode (if (= stato_opt "ON") 0 1))  (foreach e lista_ent (if (entget e)
     (entmod (append (entget e) (list (cons 60 mode)))))) (princ))
;;-------------------------------------------------------------------------------------------
(defun c:slc ()  (Config_Conic_Layers))
;;-------------------------------------------------------------------------------------------
;;-------------------------------------------------------------------------------------------

;;-------------------------------------------------------------------------------------------
;;-------------------------------------------------------------------------------------------
(progn
;;-------------------------------------------------------------------------------------------
(defun CON1P_Helper_Base_Vectors (P1 P2 / C dist_tot semi_dist v_dir v_perp)
  (setq dist_tot (distance P1 P2))
  (if (> dist_tot 1e-6)
      (progn
        (setq C         (Vet_Mid P1 P2)
              semi_dist (* 0.5 dist_tot)
              v_dir     (Vet_Uni (Vet_Sub P2 C))
              v_perp    (list (- (cadr v_dir)) (car v_dir) 0.0))
        (list C dist_tot semi_dist v_dir v_perp))
      nil))
;;-------------------------------------------------------------------------------------------
(defun CON1P_Helper_Engine_Setup (pt C v_dir v_perp opt_m last_ents / e_marker P_local P_u P_n)
  (if (and (= opt_m "Sostituisci") last_ents)
      (foreach e last_ents (if (entget e) (entdel e))))

  (setq e_marker (entlast))
  (Draw_Point_R pt *LaCo_Punti_scelti*)

  (setq P_local (Vet_Sub pt C)
        P_u     (abs (Vet_Dot P_local v_dir))
        P_n     (abs (Vet_Dot P_local v_perp)))

  (list e_marker P_u P_n))
;;-------------------------------------------------------------------------------------------
(defun CON1P_Helper_UI_State (input_str opt_m opt_e / new_m new_e trigger_redraw)
  (setq new_m opt_m new_e opt_e trigger_redraw nil)
  (cond
```

```
    ((= input_str "Modo")       (setq new_m (if (= opt_m "Sostituisci")
     "Aggiungi" "Sostituisci")))
    ((= input_str "Estendi")    (setq new_e (+ opt_e 0.5) trigger_redraw T)
                                (princ (strcat "\n[+] Estensione: " (rtos new_e 2 1))))
    ((= input_str "Accorcia")   (setq new_e (max 1.0 (- opt_e 0.5)))
                                (if (/= opt_e new_e)
                                (progn (princ (strcat "\n[-] Estensione: " (rtos new_e 2 1)))
     (setq trigger_redraw T))
                                (princ "\n[INFO] Limite minimo (1.0)."))))
  (list new_m new_e trigger_redraw))
;;-------------------------------------------------------------------------------------------
(defun CON1P_Helper_UI_Prompt (cmd_type state_list / opt_g opt_m opt_e choice)
  (mapcar 'set '(opt_g opt_m opt_e choice) state_list)
  (cond
    ((= cmd_type 'CONVP)        (initget "Guide Estendi Accorcia Modo Uscita")
                                (strcat "\nGUIDE: " opt_g " - EST: " (rtos opt_e 2 1)
     " - MODO: " opt_m " | Punto o [Guide/Estendi/Accorcia/Modo/Uscita] <Uscita>: "))
    ((= cmd_type 'CONFP_MAIN)   (initget "Ellisse Iperbole Modo Uscita")
                                (strcat "\nMODO: " opt_m " | Scegli tipo [Ellisse/Iperbole/Modo
     /Uscita] <Uscita>: "))
    ((= cmd_type 'CONFP_SUB)    (if (= choice "Iperbole")
                                    (initget "Quilatera Estendi Accorcia Modo Indietro")
                                    (initget "Modo Indietro"))
                                (strcat "\nMODO: " opt_m
                                        (if (= choice "Iperbole")
     (strcat " - EST: " (rtos opt_e 2 1)) "")
                                         " | Punto " choice " o ["
                                        (if (= choice "Iperbole")
     "Quilatera/Estendi/Accorcia/" "")
                                        "Modo/Indietro] <Indietro>: "))))
;;-------------------------------------------------------------------------------------------
(defun CON1P_Geom_Conic_2V_Engine (pt V1 V2 C a v_dir v_perp opt_m opt_e last_ents / setup_data
     e_marker P_u P_n b c_foc F1 F2 ratio A1 A2 B1 B2 x_limit dummy_pt type_found new_ents)
  (setq setup_data (CON1P_Helper_Engine_Setup pt C v_dir v_perp opt_m last_ents))
  (mapcar 'set '(e_marker P_u P_n) setup_data)

  (if (equal P_u a 1e-6)
    (progn (princ "\n[ERR] Punto sul confine.") (setq type_found "ERRORE"))
    (progn
      (if (< P_u a)
        (progn
          (setq type_found "ELLISSE"
                b (sqrt (/ (* P_n P_n) (- 1.0 (Math_Sqr (/ P_u a))))))
          (if (>= a b)
            (setq c_foc (sqrt (- (* a a) (* b b)))
```

```
                F1 (Vet_Sum C (Vet_Sca v_dir c_foc)) F2 (Vet_Sub C (Vet_Sca v_dir c_foc))
                ratio (/ b a) A1 V1 A2 V2
                B1 (Vet_Sum C (Vet_Sca v_perp b)) B2 (Vet_Sub C (Vet_Sca v_perp b)))
          (setq c_foc (sqrt (- (* b b) (* a a)))
                F1 (Vet_Sum C (Vet_Sca v_perp c_foc)) F2 (Vet_Sub C (Vet_Sca v_perp c_foc))
                ratio (/ a b) A1 (Vet_Sum C (Vet_Sca v_perp b))
    A2 (Vet_Sub C (Vet_Sca v_perp b))
                B1 V1 B2 V2))
        (Draw_ConEll_CeAx_Foci C A1 A2 B1 B2 F1 F2 ratio))

       (progn
         (setq type_found "IPERBOLE"
               b (sqrt (/ (* P_n P_n) (- (Math_Sqr (/ P_u a)) 1.0)))
               c_foc (Pitag_Ip a b)
               F1 (Vet_Sum C (Vet_Sca v_dir c_foc)))

         (setq x_limit (* P_u opt_e)
               dummy_pt (Vet_Sum C (Vet_Sca v_dir x_limit)))

         (Draw_ConIpe_CeVeFu_Compl_Free C V2 F1 (list V1 V2 pt dummy_pt))))
     (setq new_ents (Util_Collect_Ents e_marker))))
  (cons type_found new_ents))
;;-------------------------------------------------------------------------------------
(defun CON1P_Geom_Conic_2F_Engine (pt F1 F2 C c_dist V_dir V_perp type opt_m opt_e last_ents /
    setup_data e_marker P_u P_n d1 d2 a b A1 A2 B1 B2 V1 x_limit dummy_pt res_type new_ents)
  (setq setup_data (CON1P_Helper_Engine_Setup pt C V_dir V_perp opt_m last_ents)
        res_type "NULL")
  (mapcar 'set '(e_marker P_u P_n) setup_data)

  (cond
    ((= type "Ellisse")
     (setq d1 (distance pt F1) d2 (distance pt F2) a (* 0.5 (+ d1 d2)))
     (if (> a c_dist)
         (progn
           (setq b (sqrt (- (* a a) (* c_dist c_dist)))
                 A1 (Vet_Sum C (Vet_Sca V_dir a)) A2 (Vet_Sub C (Vet_Sca V_dir a))
                 B1 (Vet_Sum C (Vet_Sca V_perp b)) B2 (Vet_Sub C (Vet_Sca V_perp b)))
           (Draw_ConEll_CeAx_Foci C A1 A2 B1 B2 F1 F2 (/ b a))
           (setq res_type "Ellisse"))
         (princ "\n[ERR] Punto interno ai fuochi: ellisse impossibile.")))

    ((= type "Iperbole")
     (setq d1 (distance pt F1) d2 (distance pt F2) a (* 0.5 (abs (- d1 d2))))
     (if (and (> a 1e-6) (< a c_dist))
         (progn
```

```
            (setq b (sqrt (- (* c_dist c_dist) (* a a)))
                  V1 (Vet_Sum C (Vet_Sca V_dir a)))

            (if (equal P_u a 1e-4)
                (setq x_limit (* a opt_e))
                (setq x_limit (+ a (* (- P_u a) opt_e))))

            (setq dummy_pt (Vet_Sum C (Vet_Sca V_dir x_limit)))
            (Draw_ConIpe_CeVeFu_Compl_Free C V1 F1 (list V1 pt dummy_pt))
            (setq res_type "Iperbole"))
          (princ "\n[ERR] Punto fuori dominio iperbolico."))))

  (setq new_ents (Util_Collect_Ents e_marker))
  (cons res_type new_ents))
;;-------------------------------------------------------------------------------------------
(defun c:CON2V1P (/ pts old_os old_pd old_ce V1 V2 base_vecs C dist_V a v_dir v_perp opt_g opt_
     m opt_e guide_ents last_run_ents temp_axis loop_main P last_P last_type res ui_state redra
     w_req)
  (mapcar 'set '(pts old_os old_pd old_ce)
     (Init_Conic_App "CONICA PER 2 VERTICI E PUNTO" 2 "\nSeleziona i due vertici (V1, V2): " 32
     ))
  (if pts
    (progn
      (mapcar 'set '(V1 V2) pts)
      (if (setq base_vecs (CON1P_Helper_Base_Vectors V1 V2))
        (progn
          (mapcar 'set '(C dist_V a v_dir v_perp) base_vecs)
          (setvar "OSMODE" 0) (setvar "CMDECHO" 0)
          (setq opt_g "off" opt_m "Sostituisci" opt_e 1.0
                guide_ents nil last_run_ents nil last_P nil last_type nil
                temp_axis (Draw_Line_2Pu_R V1 V2 *LaCo_Costruzioni*))
;;-------------------------------------------------------------------------------------------
          (defun update_3pv_guides ()
            (if guide_ents (progn (foreach e guide_ents (if (entget e) (entdel e)))
     (setq guide_ents nil)))
            (if (= opt_g "on")
              (setq guide_ents (list (Draw_Line_2Pu_R (Vet_Sum V1 (Vet_Sca v_perp (* dist_V 5.0
     ))) (Vet_Sub V1 (Vet_Sca v_perp (* dist_V 5.0))) *LaCo_Luoghi_geometrici*)
                                    (Draw_Line_2Pu_R (Vet_Sum V2 (Vet_Sca v_perp (* dist_V 5.0
     ))) (Vet_Sub V2 (Vet_Sca v_perp (* dist_V 5.0))) *LaCo_Luoghi_geometrici*)))))

          (setq loop_main T)
          (while loop_main
            (setvar "OSMODE" (if old_os old_os 0))
            (setq P (getpoint C (CON1P_Helper_UI_Prompt 'CONVP (list opt_g opt_m opt_e nil))))
```

```
            (setvar "OSMODE" 0)

            (cond
              ((or (null P) (= P "Uscita")) (setq loop_main nil))
              ((= P "Guide") (setq opt_g (if (= opt_g "on") "off" "on")) (update_3pv_guides))

              ((member P '("Estendi" "Accorcia" "Modo"))
               (setq ui_state (CON1P_Helper_UI_State P opt_m opt_e))
               (mapcar 'set '(opt_m opt_e redraw_req) ui_state)
               (if (= opt_m "Aggiungi") (setq last_run_ents nil))
               (if (and redraw_req last_P (= last_type "IPERBOLE") (= opt_m "Sostituisci"))
                   (setq last_run_ents (cdr (CON1P_Geom_Conic_2V_Engine last_P V1 V2 C a v_dir
    v_perp opt_m opt_e last_run_ents)))))

              ((listp P)
               (if temp_axis (progn (if (entget temp_axis) (entdel temp_axis))
    (setq temp_axis nil)))
               (setq last_P P res (CON1P_Geom_Conic_2V_Engine P V1 V2 C a v_dir v_perp opt_m op
    t_e last_run_ents)
                     last_type (car res) last_run_ents (cdr res)))
            )
          )
          (if guide_ents (foreach e guide_ents (if (entget e) (entdel e))))
          (if temp_axis (if (entget temp_axis) (entdel temp_axis)))
          (princ "\nComando terminato."))
        (princ "\n[ERRORE] I vertici non possono coincidere.")))
    (princ "\nInput annullato."))
  (Restore_Conic_App old_os old_pd old_ce))
;;-------------------------------------------------------------------------------------------
(defun c:CON2F1P (/ pts old_os old_pd old_ce F1 F2 base_vecs C dist_F c_dist V_dir V_perp loop_
     main opt_m opt_e last_run_ents choice loop_sub P last_P last_type res ui_state redraw_req)
  (mapcar 'set '(pts old_os old_pd old_ce)
     (Init_Conic_App "CONICA BIFOCALE (LABORATORIO)" 2 "\nSpecifica i due Fuochi (F1, F2): " 32
     ))
  (if pts
    (progn
      (mapcar 'set '(F1 F2) pts)
      (if (setq base_vecs (CON1P_Helper_Base_Vectors F1 F2))
        (progn
          (mapcar 'set '(C dist_F c_dist V_dir V_perp) base_vecs)
          (setq opt_m "Sostituisci" opt_e 1.0
                last_run_ents nil last_P nil last_type nil
                loop_main T)

          (while loop_main
```

```
       (setq choice (getkword (CON1P_Helper_UI_Prompt 'CONFP_MAIN (list nil opt_m nil nil)
)))

       (cond
         ((or (null choice) (= choice "Uscita")) (setq loop_main nil))

         ((= choice "Modo")
          (setq ui_state (CON1P_Helper_UI_State "Modo" opt_m opt_e) opt_m (car ui_state))
          (if (= opt_m "Aggiungi") (setq last_run_ents nil)))

         ((or (= choice "Ellisse") (= choice "Iperbole"))
          (setq loop_sub T last_P nil last_type nil)
          (while loop_sub
            (setvar "OSMODE" (if old_os old_os 0))
            (setq P (getpoint C (CON1P_Helper_UI_Prompt 'CONFP_SUB (list nil opt_m opt_e c
hoice))))
            (setvar "OSMODE" 0)

            (cond
              ((or (null P) (= P "Indietro")) (setq loop_sub nil))

              ((member P '("Estendi" "Accorcia" "Modo"))
               (setq ui_state (CON1P_Helper_UI_State P opt_m opt_e))
               (mapcar 'set '(opt_m opt_e redraw_req) ui_state)
               (if (= opt_m "Aggiungi") (setq last_run_ents nil))
               (if (and redraw_req last_P (= opt_m "Sostituisci"))
                   (progn (setq res (CON1P_Geom_Conic_2F_Engine last_P F1 F2 C c_dist V_di
r V_perp choice opt_m opt_e last_run_ents))
                          (setq last_type (car res) last_run_ents (cdr res)))))

              ((= P "Quilatera")
               (setq last_P (Vet_Sum C (Vet_Sca V_dir (/ c_dist (sqrt 2.0)))))
               (setq res (CON1P_Geom_Conic_2F_Engine last_P F1 F2 C c_dist V_dir V_perp ch
oice opt_m opt_e last_run_ents))
               (setq last_type (car res) last_run_ents (cdr res)))

              ((listp P)
               (setq last_P P)
               (setq res (CON1P_Geom_Conic_2F_Engine P F1 F2 C c_dist V_dir V_perp choice
opt_m opt_e last_run_ents))
               (setq last_type (car res) last_run_ents (cdr res)))
            )
          )
         )
       )
```

```
          )
        )
        (princ "\n[ERRORE] I fuochi non possono coincidere.")))
    (princ "\nInput annullato."))
  (Restore_Conic_App old_os old_pd old_ce))
)
;;-------------------------------------------------------------------------------------------
;;-------------------------------------------------------------------------------------------

;;-------------------------------------------------------------------------------------------
;;-------------------------------------------------------------------------------------------
(progn
;;-------------------------------------------------------------------------------------------
(defun c:parVF (/ pts old_os old_pd old_ce p_base p_other opt_ord opt_m loop_main p_input puV p
     uF p_dist dir_asse dummy_pt e_marker last_run_ents)

  (mapcar 'set '(pts old_os old_pd old_ce)
     (Init_Conic_App "LABORATORIO PARABOLA 2P" 1 "\nSpecifica il punto BASE (Ancora): " 32))

  (if pts
    (progn
      (setq p_base (car pts)
            p_other nil
            opt_ord "V-F"
            opt_m "Sostituisci"
            last_run_ents nil
            loop_main T)

      (while loop_main
        (setvar "OSMODE" (if old_os old_os 0))
        (initget "Ordinamento Modo Fine")

        (setq p_input
          (getpoint p_base
            (strcat "\nORD: " opt_ord " - MODO: " opt_m
                    " | Specifica " (if (= opt_ord "V-F") "Fuoco (F)" "Vertice (V)")
                    " o [Ordinamento/Modo/Fine] <Fine>: ")))

        (setvar "OSMODE" 0)

        (cond
          ((or (null p_input) (= p_input "Fine"))
           (setq loop_main nil))

          ((= p_input "Ordinamento")
```

```
          (setq opt_ord (if (= opt_ord "V-F") "F-V" "V-F"))
          (if p_other
              (setq tmp p_base p_base p_other p_other tmp))
          (princ (strcat "\n[INFO] Ordinamento cambiato in: " opt_ord)))

         ((= p_input "Modo")
          (setq opt_m (if (= opt_m "Sostituisci") "Aggiungi" "Sostituisci"))
          (if (= opt_m "Aggiungi") (setq last_run_ents nil))
          (princ (strcat "\n[INFO] Modalità impostata su: " opt_m)))

         ((listp p_input)
          (setq p_other p_input)
          (setq p_dist (distance p_base p_other))

          (if (> p_dist 1e-6)
              (progn
                (if (and (= opt_m "Sostituisci") last_run_ents)
                    (foreach e last_run_ents (if (entget e) (entdel e))))

                (setq e_marker (entlast))

                (if (= opt_ord "V-F")
                    (setq puV p_base puF p_other)
                    (setq puV p_other puF p_base))

                (setq dir_asse (Vet_Uni (Vet_Sub puF puV)))
                (setq dummy_pt (Vet_Sum puV (Vet_Sca dir_asse (* p_dist 15.0))))

                (Draw_Point_R p_other *LaCo_Punti_scelti*)
                (Draw_ConPar_VeFu_Compl puV puF (list puV puF dummy_pt))

                (if (= opt_m "Sostituisci")
                    (setq last_run_ents (Util_Collect_Ents e_marker))
                    (setq last_run_ents nil))

                (princ "\n[OK] Parabola generata."))
              (princ "\n[ERRORE] I punti non possono coincidere!")))        )      )
      (princ "\nLaboratorio terminato."))
    (princ "\nInput annullato."))

  (Restore_Conic_App old_os old_pd old_ce)
  (princ))
;;-------------------------------------------------------------------------------
(defun Get_Dima_Data ()
  '((-478.0131 -229.5335) (-477.8495 -229.482) (-477.5733 -229.3508) (-477.4317 -229.2573)
```

(-477.2971 -229.2195) (-477.0209 -229.0883) (-476.8502 -228.9811) (-476.7447 -228.9571)
(-476.4684 -228.8259) (-476.2687 -228.7048) (-476.1922 -228.6947) (-475.916 -228.5634)
(-475.6872 -228.4286) (-475.6398 -228.4322) (-475.6322 -228.4286) (-475.6398 -228.4061)
(-475.3636 -228.301) (-475.1058 -228.1524) (-475.0874 -228.1698) (-475.0874 -228.1437)
(-475.0508 -228.1524) (-474.5349 -227.9074) (-474.5243 -227.8762) (-474.2587 -227.7762)
(-473.9825 -227.6449) (-473.9427 -227.6) (-473.7063 -227.5137) (-473.4301 -227.3825)
(-473.3612 -227.3238) (-473.1539 -227.2513) (-472.8777 -227.1201) (-472.7797 -227.0476)
(-472.6014 -226.9889) (-472.3252 -226.8577) (-472.1982 -226.7714) (-472.049 -226.7265)
(-471.7728 -226.5953) (-471.6166 -226.4951) (-471.4966 -226.4641) (-471.2204 -226.3329)
(-471.0351 -226.2189) (-470.9442 -226.2017) (-470.668 -226.0705) (-470.4535 -225.9427)
(-470.3917 -225.9393) (-470.1155 -225.8081) (-469.872 -225.6665) (-469.8393 -225.6769)
(-469.8393 -225.651) (-469.8175 -225.6665) (-469.2869 -225.4145) (-469.2904 -225.3903)
(-469.2869 -225.3886) (-469.2359 -225.3903) (-468.7345 -225.1521) (-468.7088 -225.1141)
(-468.4582 -225.0209) (-468.182 -224.8897) (-468.1272 -224.8379) (-467.9058 -224.7585)
(-467.6296 -224.6273) (-467.5456 -224.5616) (-467.3534 -224.4961) (-467.0772 -224.3649)
(-466.964 -224.2854) (-466.801 -224.2337) (-466.5247 -224.1026) (-466.3824 -224.0092)
(-466.2485 -223.9714) (-465.9723 -223.8402) (-465.8008 -223.733) (-465.6961 -223.709)
(-465.4199 -223.5778) (-465.2192 -223.4568) (-465.1437 -223.4466) (-464.8675 -223.3154)
(-464.6375 -223.1806) (-464.5913 -223.1843) (-464.5835 -223.1806) (-464.5913 -223.1586)
(-464.315 -223.0531) (-464.0559 -222.9044) (-464.0388 -222.9219) (-464.0388 -222.8963)
(-464.0019 -222.9044) (-463.4864 -222.6595) (-463.4742 -222.6282) (-463.2102 -222.5284)
(-462.934 -222.3972) (-462.8926 -222.3519) (-462.6578 -222.266) (-403.5119 -194.1782)
(-340.4875 -164.3471) (-287.7745 -139.4879) (-243.0313 -118.4956) (-242.7917 -118.3994)
(-242.5155 -118.2702) (-242.4406 -118.2194) (-242.2393 -118.141) (-241.9631 -118.0119)
(-241.8498 -117.9432) (-241.6869 -117.8827) (-241.4107 -117.7536) (-241.259 -117.667)
(-241.1345 -117.6245) (-240.8582 -117.4953) (-240.6681 -117.3908) (-240.582 -117.3662)
(-240.3058 -117.2371) (-240.0771 -117.1146) (-240.0296 -117.108) (-239.7534 -116.9789)
(-239.4861 -116.8383) (-239.4772 -116.8498) (-239.4772 -116.8342) (-239.4527 -116.8383)
(-238.9248 -116.5916) (-238.8949 -116.5621) (-238.6485 -116.4625) (-238.3723 -116.3334)
(-238.3038 -116.2859) (-238.0961 -116.2044) (-237.8199 -116.0753) (-237.7125 -116.0097)
(-237.5437 -115.9463) (-237.2675 -115.8172) (-237.1212 -115.7335) (-236.9913 -115.6882)
(-236.715 -115.5592) (-236.5299 -115.4573) (-236.4388 -115.4301) (-236.1626 -115.3011)
(-235.9384 -115.1811) (-235.8864 -115.1721) (-235.6102 -115.0431) (-235.3469 -114.9049)
(-235.334 -114.9141) (-235.334 -114.8988) (-235.3141 -114.9049) (-234.7815 -114.6561)
(-234.7554 -114.6286) (-234.5053 -114.5272) (-234.2291 -114.3982) (-234.1637 -114.3524)
(-233.9529 -114.2692) (-233.6767 -114.1403) (-233.572 -114.0762) (-233.4005 -114.0113)
(-233.1243 -113.8824) (-232.9802 -113.8) (-232.8481 -113.7535) (-232.5718 -113.6245)
(-232.3884 -113.5238) (-232.2956 -113.4956) (-232.0194 -113.3667) (-231.7965 -113.2476)
(-231.7432 -113.2378) (-231.467 -113.1089) (-231.2045 -112.9714) (-231.1908 -112.98)
(-231.1908 -112.965) (-231.1722 -112.9714) (-230.6383 -112.7223) (-230.6124 -112.6951)
(-230.3621 -112.5934) (-230.0859 -112.4645) (-230.0203 -112.4189) (-229.8097 -112.3357)
(-229.5335 -112.2068) (-229.428 -112.1427) (-229.2573 -112.078) (-228.9811 -111.9492)
(-228.8358 -111.8665) (-228.7048 -111.8203) (-228.4286 -111.6915) (-228.2434 -111.5903)
(-228.1524 -111.5627) (-227.8762 -111.4339) (-227.651 -111.3141) (-227.6 -111.3051)
(-227.3238 -111.1763) (-227.0585 -111.0379) (-227.0476 -111.0476) (-227.0476 -111.0328)

(-227.0268 -111.0379) (-226.4951 -110.79) (-226.4659 -110.7616) (-226.2189 -110.6613)
(-225.9427 -110.5325) (-225.8732 -110.4854) (-225.6665 -110.4038) (-225.3903 -110.2751)
(-225.2805 -110.2092) (-225.1141 -110.1463) (-224.8379 -110.0176) (-224.6876 -109.933)
(-224.5616 -109.8889) (-224.2854 -109.7602) (-224.0947 -109.6568) (-224.0092 -109.6315)
(-223.733 -109.5028) (-223.5017 -109.3806) (-223.4568 -109.3742) (-222.9044 -109.1168)
(-222.9087 -109.1044) (-222.9044 -109.1024) (-222.8776 -109.1044) (-222.3519 -108.8595)
(-222.3155 -108.8282) (-222.0757 -108.7309) (-221.7995 -108.6023) (-221.7223 -108.5519)
(-221.5233 -108.4737) (-221.2471 -108.345) (-221.129 -108.2757) (-220.9709 -108.2164)
(-220.6947 -108.0879) (-220.5356 -107.9995) (-220.4184 -107.9593) (-220.1422 -107.8307)
(-219.9421 -107.7233) (-219.866 -107.7021) (-219.5898 -107.5736) (-219.3486 -107.4471)
(-219.3136 -107.445) (-218.7612 -107.1879) (-218.7549 -107.1709) (-218.4849 -107.0594)
(-218.2087 -106.9309) (-218.1612 -106.8947) (-217.9325 -106.8024) (-217.6563 -106.6739)
(-217.5674 -106.6184) (-217.3801 -106.5454) (-217.1039 -106.4169) (-216.9735 -106.3422)
(-216.8277 -106.2885) (-216.5515 -106.16) (-216.3795 -106.066) (-216.2752 -106.0315)
(-215.999 -105.9031) (-215.7854 -105.7898) (-215.7228 -105.7746) (-215.4466 -105.6462)
(-215.1912 -105.5136) (-215.1704 -105.5178) (-215.1704 -105.5039) (-215.1613 -105.5136)
(-214.618 -105.261) (-214.597 -105.2374) (-214.3417 -105.1326) (-214.0655 -105.0042)
(-214.0026 -104.9612) (-213.7893 -104.8758) (-213.5131 -104.7475) (-213.4082 -104.6849)
(-213.2369 -104.6191) (-212.9607 -104.4908) (-212.8136 -104.4087) (-212.6845 -104.3624)
(-212.4082 -104.2341) (-212.219 -104.1325) (-212.132 -104.1058) (-211.8558 -103.9775)
(-211.6242 -103.8563) (-211.5796 -103.8492) (-211.0272 -103.5926) (-211.0294 -103.5801)
(-211.0272 -103.5791) (-211.0002 -103.5801) (-210.4748 -103.3361) (-210.4345 -103.3039)
(-210.1985 -103.2078) (-209.9223 -103.0796) (-209.8395 -103.0277) (-209.6461 -102.9514)
(-209.3699 -102.8232) (-209.2444 -102.7515) (-209.0937 -102.6949) (-208.8175 -102.5667)
(-208.6491 -102.4752) (-208.5413 -102.4385) (-208.265 -102.3104) (-208.0538 -102.199)
(-207.9888 -102.1822) (-207.7126 -102.054) (-207.4584 -101.9228) (-207.4364 -101.9259)
(-207.4364 -101.9126) (-207.4298 -101.9228) (-206.884 -101.6696) (-206.8629 -101.6466)
(-206.6078 -101.5415) (-206.3315 -101.4134) (-206.2672 -101.3704) (-206.0553 -101.2853)
(-205.7791 -101.1572) (-205.6715 -101.0942) (-205.5029 -101.0291) (-205.2267 -100.901)
(-205.0757 -100.818) (-204.9505 -100.773) (-204.6743 -100.6449) (-204.4797 -100.5417)
(-204.3981 -100.5169) (-204.1218 -100.3889) (-203.8837 -100.2655) (-203.8456 -100.2609)
(-203.2932 -100.0049) (-203.2875 -99.9893) (-203.017 -99.8769) (-202.7408 -99.7489)
(-202.6913 -99.7131) (-202.4646 -99.6209) (-202.1883 -99.493) (-202.0949 -99.4369)
(-201.9121 -99.365) (-201.6359 -99.2371) (-201.4984 -99.1607) (-201.3597 -99.1092)
(-201.0835 -98.9813) (-200.9018 -98.8845) (-200.8073 -98.8534) (-200.5311 -98.7255)
(-200.3051 -98.6083) (-200.2549 -98.5976) (-199.9786 -98.4698) (-199.7083 -98.332)
(-199.7024 -98.3419) (-199.7024 -98.3293) (-199.6811 -98.332) (-199.15 -98.0862)
(-199.1113 -98.0558) (-198.8738 -97.9584) (-198.5976 -97.8306) (-198.5143 -97.7796)
(-198.3214 -97.7028) (-198.0451 -97.575) (-197.9171 -97.5034) (-197.7689 -97.4473)
(-197.4927 -97.3195) (-197.3198 -97.2272) (-197.2165 -97.1918) (-196.9403 -97.064)
(-196.7224 -96.951) (-196.6641 -96.9363) (-196.3879 -96.8086) (-196.1248 -96.6748)
(-196.1116 -96.6809) (-196.1116 -96.6687) (-196.0984 -96.6748) (-195.5592 -96.4255)
(-195.5272 -96.3985) (-195.283 -96.2978) (-195.0068 -96.1702) (-194.9294 -96.1223)
(-194.7306 -96.0426) (-194.4544 -95.9149) (-194.3315 -95.8461) (-194.1782 -95.7873)
(-193.9019 -95.6597) (-193.7334 -95.5699) (-193.6257 -95.5321) (-193.3495 -95.4045)

```
    (-193.1353 -95.2937) (-193.0733 -95.277) (-192.7971 -95.1494) (-192.537 -95.0175)
   (-192.5209 -95.0219) (-192.5209 -95.01) (-192.5113 -95.0175) (-191.9684 -94.7668)
    (-191.9386 -94.7413) (-191.6922 -94.6393) (-191.416 -94.5118) (-191.34 -94.465)
   (-191.1398 -94.3844) (-190.8636 -94.2569) (-190.7413 -94.1888) (-190.5874 -94.1294)
    (-190.3112 -94.002) (-190.1425 -93.9126) (-190.0349 -93.8746) (-189.7587 -93.7472)
   (-189.5436 -93.6364) (-189.4825 -93.6198) (-189.2063 -93.4924) (-188.9445 -93.3602)
    (-188.9301 -93.365) (-188.9301 -93.3536) (-188.9196 -93.3602) (-188.3777 -93.1103)
   (-188.3452 -93.084) (-173.315 -86.1786) (-158.7808 -79.5495) (-145.9529 -73.749)
    (-134.223 -68.501) (-123.5946 -63.8053) (-113.4412 -59.3859) (-108.9412 -57.4524)
   (-108.8282 -57.4106) (-108.5519 -57.2926) (-108.2948 -57.1762) (-108.2757 -57.1746)
    (-107.7233 -56.9389) (-107.6474 -56.9) (-107.4471 -56.8211) (-107.1709 -56.7035)
   (-106.9991 -56.6238) (-106.8947 -56.5859) (-106.6184 -56.4683) (-106.3497 -56.3476)
    (-106.3422 -56.3509) (-106.3422 -56.3444) (-106.3344 -56.3476) (-105.7898 -56.1162)
   (-105.6993 -56.0714) (-105.5136 -55.999) (-105.2374 -55.8819) (-105.0479 -55.7951)
    (-104.9612 -55.7648) (-104.4087 -55.5309) (-104.3954 -55.5189) (-99.7954 -53.5854)
   (-95.8035 -51.9282) (-91.7608 -50.2709) (-88.3411 -48.8898) (-84.1652 -47.2325)
    (-80.619 -45.8515) (-77.7229 -44.7466) (-74.0191 -43.3655) (-70.9699 -42.2607)
   (-67.8276 -41.1558) (-65.3891 -40.3272) (-62.8703 -39.4985) (-60.2403 -38.6699)
    (-60.2146 -38.665) (-59.3332 -38.3937) (-57.4643 -37.8413) (-55.5026 -37.2888)
   (-54.4799 -37.0126) (-53.4195 -36.7364) (-52.3181 -36.4602) (-51.1633 -36.184)
    (-49.9398 -35.9078) (-49.166 -35.7441) (-48.6265 -35.6316) (-47.1855 -35.3553)
   (-46.4039 -35.2206) (-45.4717 -35.0697) (-44.6377 -34.9497) (-43.7492 -34.8374)
    (-43.2749 -34.7845) (-42.8099 -34.7382) (-42.0423 -34.6734) (-41.2832 -34.626)
   (-40.6034 -34.5992) (-40.2279 -34.5913) (-39.8783 -34.5898) (-39.5417 -34.5912)
    (-39.1878 -34.5993) (-38.8411 -34.613) (-38.5371 -34.6303) (-38.2448 -34.6519)
   (-37.9626 -34.6778) (-37.69 -34.708) (-37.4269 -34.7425) (-37.1766 -34.7808)
    (-36.9064 -34.8288) (-36.6362 -34.8849) (-36.3855 -34.9453) (-36.1538 -35.0098)
   (-35.9638 -35.0705) (-35.843 -35.1136) (-35.7653 -35.1437) (-35.679 -35.1794)
    (-35.597 -35.2161) (-35.5193 -35.2543) (-35.4526 -35.2906) (-35.3939 -35.3267)
   (-35.3723 -35.3418) (-35.3553 -35.3553) (-35.3564 -35.3546) (-35.3578 -35.3541)
    (-35.3613 -35.3541) (-35.3661 -35.3553) (-35.3723 -35.3578) (-35.3881 -35.3664)
   (-35.4089 -35.3799) (-35.4632 -35.4201) (-35.5409 -35.4828) (-35.7524 -35.6649)
    (-36.0419 -35.925) (-36.8616 -36.6851) (-38.0055 -37.7679) (-41.3244 -40.9573)
   (-46.0842 -45.5752) (-60.51 -59.6621) (-83.6216 -82.3116) (-356.5917 -350.4529)))
;;-------------------------------------------------------------------------------------
(defun Geom_Draw_Dima_Boundaries (pA pB lay / pts_raw p1_base p2_base d_base ang_base d_user an
     g_user scale d_ang final_pts ent1 ent2 pM pPerp obj1 obj2)
  (setq p1_base  '(-35.35533906 -35.35533906)
        p2_base  '(35.35533906 35.35533906)
        d_base   (distance p1_base p2_base)
        ang_base (angle p1_base p2_base)
        d_user   (distance pA pB)
        ang_user (angle pA pB)
        scale    (/ d_user d_base)
        d_ang    (- ang_user ang_base)
```

```
        pts_raw  (Get_Dima_Data))

  (setq final_pts
    (mapcar
      '(lambda (pt)
         (polar pA (+ (angle p1_base pt) d_ang) (* (distance p1_base pt) scale)))
      pts_raw))

  (setq ent1 (entmakex (append (list '(0 . "LWPOLYLINE") '(100 . "AcDbEntity")
     '(100 . "AcDbPolyline") (cons 90 (length final_pts)) '(70 . 0) (cons 8 lay))
                               (mapcar '(lambda (p) (list 10 (car p) (cadr p))) final_pts))))

  (setq obj1  (vlax-ename->vla-object ent1))
  (setq obj2  (vla-mirror obj1 (vlax-3d-point pA) (vlax-3d-point pB)))
  (setq pM    (polar pA ang_user (/ d_user 2.0))
        pPerp (polar pM (+ ang_user (/ pi 2.0)) 10.0))
  (vla-mirror obj1 (vlax-3d-point pM) (vlax-3d-point pPerp))
  (vla-mirror obj2 (vlax-3d-point pM) (vlax-3d-point pPerp))
  (princ))
;;-------------------------------------------------------------------------------------
(defun Helper_Update_Dima_Par2pv (opt_g guide_ents P1 P2 / e_start e_curr)
  (cond
    ((and (= opt_g "Si") (not guide_ents))
     (setq e_start (entlast))
     (princ "\n[GUIDA] Generazione confini topologici in corso...")
     (Geom_Draw_Dima_Boundaries P1 P2 *LaCo_Luoghi_geometrici*)
     (setq e_curr (entnext e_start))
     (while e_curr
       (setq guide_ents (cons e_curr guide_ents)
             e_curr (entnext e_curr)))
     (princ " Fatto.")
     guide_ents)
    ((and (= opt_g "No") guide_ents)
     (foreach e guide_ents (if (entget e) (entdel e)))
     (princ "\n[INFO] Dima topologica rimossa.")
     nil)
    (T guide_ents)))
;;-------------------------------------------------------------------------------------
(defun Geom_Eval_Par_Error (pts_rel c s / pr)
  (setq pr (Util_Rot_List_2D_CS pts_rel c s))
  (- (* (caar pr) (caar pr) (cadadr pr)) (* (caadr pr) (caadr pr) (cadar pr))))
;;-------------------------------------------------------------------------------------
(defun c:par2PV (/ pts old_os old_pd old_ce P1 P2 loop_active V pts_rel Step Ang deg2rad prev_e
     rr curr_err min_a max_a mid_a mid_err
                 Solutions is_dup c s x1 y1 A_val p_dist F_loc F_wcs num_found opt_g opt_m guid
```

```
   e_ents last_run_ents e_marker)

(mapcar 'set '(pts old_os old_pd old_ce)
   (Init_Conic_App "PARABOLA PER 2 PUNTI FISSI E VERTICE" 2 "\nSpecifica i due Punti di passa
   ggio fissi: " 32))
(if pts
  (progn
    (mapcar 'set '(P1 P2) pts)
    (setvar "OSMODE" 0)
    (setvar "CMDECHO" 0)

    (setq opt_g "off"
          opt_m "Sostituisci"
          guide_ents nil
          last_run_ents nil
          deg2rad (/ pi 180.0))

    (setq guide_ents (Helper_Update_Dima_Par2pv "No" guide_ents P1 P2))
    (setq loop_active T)
    (while loop_active
      (setvar "OSMODE" (if old_os old_os 0))
      (initget "Guide Modo Uscita")
      (setq V (getpoint (strcat "\nGUIDE: " opt_g " - MODO: " opt_m " | Specifica il Vertice
   (V) o [Guide/Modo/Uscita] <Uscita>: ")))
      (setvar "OSMODE" 0)
      (cond
        ((or (null V) (= V "Uscita")) (setq loop_active nil))

        ((= V "Guide")              (setq opt_g (if (= opt_g "on") "off" "on"))
                                    (setq guide_ents (Helper_Update_Dima_Par2pv (if (= opt_g
   "on") "Si" "No") guide_ents P1 P2)))

        ((= V "Modo")               (setq opt_m (if (= opt_m "Sostituisci")
   "Aggiungi" "Sostituisci")))

        ((listp V)
         (if (and (= opt_m "Sostituisci") last_run_ents)
             (foreach e last_run_ents (if (entget e) (entdel e))))

         (setq e_marker (entlast))
         (Draw_Point_R V *LaCo_Punti_scelti*)

         (setq pts_rel (list (Vet_Sub P1 V) (Vet_Sub P2 V))
               Step 0.5
               Ang 0.0
```

```
            Solutions nil)

      (setq prev_err (Geom_Eval_Par_Error pts_rel 1.0 0.0))
      (repeat 360
        (setq Ang      (+ Ang Step)
              curr_err (Geom_Eval_Par_Error pts_rel (cos (* Ang deg2rad))
(sin (* Ang deg2rad))))

        (if (< (* prev_err curr_err) 0)
          (progn
            (setq min_a (- Ang Step) max_a Ang)
            (repeat 20
              (setq mid_a (* 0.5 (+ min_a max_a))
                    mid_err (Geom_Eval_Par_Error pts_rel (cos (* mid_a deg2rad))
(sin (* mid_a deg2rad))))
              (if (< (* mid_err prev_err) 0)
                  (setq max_a mid_a)
                  (setq min_a mid_a prev_err mid_err)))

            (setq is_dup nil)
            (foreach s_val Solutions (if (< (abs (- mid_a s_val)) 1.0) (setq is_dup T)))
            (if (not is_dup) (setq Solutions (cons mid_a Solutions)))))
        (setq prev_err curr_err))

      (setq num_found 0)
      (foreach sol Solutions
        (setq c (cos (* sol deg2rad))
              s (sin (* sol deg2rad))
              x1 (caar (Util_Rot_List_2D_CS pts_rel c s))
              y1 (cadar (Util_Rot_List_2D_CS pts_rel c s)))

        (if (> (abs x1) 1e-6)
          (progn
            (setq A_val (/ y1 (* x1 x1)))
            (if (> (abs A_val) 1e-9)
              (progn
                (setq p_dist (/ 1.0 (* 4.0 (abs A_val)))
                      F_loc (if (> A_val 0) (list 0.0 p_dist 0.0) (list 0.0 (- p_dist)
0.0))
                      F_wcs (Vet_Sum V (list (car (Util_Rot_2D_Inv_CS F_loc c s))
(cadr (Util_Rot_2D_Inv_CS F_loc c s)) 0.0)))
                (Draw_ConPar_VeFu_Compl V F_wcs (list P1 P2 V))
                (setq num_found (1+ num_found)))))))

      (setq last_run_ents (Util_Collect_Ents e_marker))
```

```
            (princ (strcat "\nDisegnate " (itoa num_found) " parabole uniche valide.")))))

      (if (and (= opt_g "off") guide_ents)
          (foreach e guide_ents (if (entget e) (entdel e))))
      (princ "\nComando terminato."))
    (princ "\nInput annullato."))
  (Restore_Conic_App old_os old_pd old_ce)
  (princ))
)
;;-------------------------------------------------------------------------------------------
;;-------------------------------------------------------------------------------------------

;;-------------------------------------------------------------------------------------------
;;-------------------------------------------------------------------------------------------
(progn
;;-------------------------------------------------------------------------------------------
(defun PAR2PFAPI_Helper_UI_Prompt (opt_gf opt_gv opt_m)
    (initget "Fuochi Vertici Modo Uscita")
    (strcat "\nF:" opt_gf " V:" opt_gv " | M:" opt_m " | Punto o [Fuochi/Vertici/Modo/Uscita] <
     Uscita>: ")
  )
;;-------------------------------------------------------------------------------------------
(defun PAR2PFAPI_Modulo_Pi (ang_val) (setq ang_val (rem ang_val pi))  (if (< ang_val 0.0)
     (+ ang_val pi) ang_val))
;;-------------------------------------------------------------------------------------------
(defun PAR2PFAPI_Fractal_Around_Param_Safe (param_center step_base / step_idx params_local)
    (setq params_local nil
          step_idx 1)
    (while (<= step_idx 16)
           (setq params_local (cons (+ param_center (* step_idx (/ step_base 16.0)))
     params_local)
                 params_local (cons (- param_center (* step_idx (/ step_base 16.0)))
     params_local)
                 step_idx (1+ step_idx)))
    (setq step_idx 1)
    (while (<= step_idx 8)
           (setq params_local (cons (+ param_center (* 1.0 step_base)
     (* step_idx (/ step_base 8.0))) params_local)
                 params_local (cons (- param_center (* 1.0 step_base)
     (* step_idx (/ step_base 8.0))) params_local)
                 step_idx (1+ step_idx)))
    (setq step_idx 1)
    (while (<= step_idx 4)
           (setq params_local (cons (+ param_center (* 2.0 step_base)
```

```
     (* step_idx (/ step_base 4.0))) params_local)
                params_local (cons (- param_center (* 2.0 step_base)
     (* step_idx (/ step_base 4.0))) params_local)
                step_idx (1+ step_idx)))
    params_local)
;;-------------------------------------------------------------------------------------
(defun PAR2PFAPI_Calc_F_Linear (pt_F P1 P2 u_x u_y n_x n_y const_num const_den / dx1 dy1 dx2 dy
     2 x_norm1 x_norm2)
    (setq dx1     (- (car  P1) (car  pt_F))
          dy1     (- (cadr P1) (cadr pt_F))
          dx2     (- (car  P2) (car  pt_F))
          dy2     (- (cadr P2) (cadr pt_F))
          x_norm1 (+ (* dx1 n_x) (* dy1 n_y))
          x_norm2 (+ (* dx2 n_x) (* dy2 n_y)))
    (/ (* const_num (+ x_norm1 x_norm2)) const_den))
;;-------------------------------------------------------------------------------------
(defun PAR2PFAPI_Engine_Proprio (P1 P2 P3 opt_e / MAX_DIST hd_steps step_base ang_A ang_B final
     _params curr_idx sing_params clean_params prev_p branches_V curr_branch_V branches_F curr_
     branch_F prev_den ang_rad u_x u_y n_x n_y dx1 dy1 dx2 dy2 x1 y1 x2 y2 den A_val C_val f_di
     st pt_V pt_F min_dist)
  (setq MAX_DIST     (* (distance P1 P2) 100.0 opt_e)
        hd_steps     7200.0
        step_base    (/ pi hd_steps)
        ang_A        (PAR2PFAPI_Modulo_Pi (angle P1 P2))
        ang_B        (PAR2PFAPI_Modulo_Pi (angle P3 (Vet_Mid P1 P2)))
        sing_params  (list ang_A ang_B)
        final_params nil
        curr_idx     0)p
  (while (<= curr_idx hd_steps)
         (setq final_params (cons (* curr_idx step_base) final_params)
               curr_idx (1+ curr_idx)))
  (foreach sp sing_params
          (setq final_params (append final_params (PAR2PFAPI_Fractal_Around_Param_Safe sp step_
     base))))
  (setq final_params (vl-sort final_params '<)
        clean_params nil
        prev_p      -1e99)
  (foreach p final_params
           (if (> (- p prev_p) 1e-9) (setq clean_params (cons p clean_params) prev_p p)))
  (setq final_params   (reverse clean_params)
        branches_V     nil
        curr_branch_V  nil
        branches_F     nil
        curr_branch_F  nil
        prev_den       nil)
```

```
  (foreach ang_rad final_params
          (setq u_x (cos ang_rad)
               u_y (sin ang_rad)
               n_x (- u_y) n_y u_x
                dx1 (- (car  P1) (car  P3))
               dy1 (- (cadr P1) (cadr P3))
                dx2 (- (car  P2) (car  P3))
               dy2 (- (cadr P2) (cadr P3))
                x1  (+ (* dx1 n_x) (* dy1 n_y))
               y1  (+ (* dx1 u_x) (* dy1 u_y))
                x2  (+ (* dx2 n_x) (* dy2 n_y))
               y2  (+ (* dx2 u_x) (* dy2 u_y))
                den (- (* x1 x1) (* x2 x2)))
          (if (and prev_den (< (* prev_den den) 0.0))
              (progn (if curr_branch_V (setq branches_V (cons (reverse curr_branch_V)
     branches_V) curr_branch_V nil))
                     (if curr_branch_F (setq branches_F (cons (reverse curr_branch_F)
     branches_F) curr_branch_F nil))))
          (if (> (abs den) 1e-9)
              (progn  (setq A_val (/ (- y1 y2) den))
                      (if (> (abs A_val) 1e-9)
                          (progn (setq C_val   (- y1 (* A_val x1 x1))
                                       f_dist   (/ 1.0 (* 4.0 A_val))
                                       pt_V     (list (+ (car P3) (* C_val u_x)) (+ (cadr P3)
     (* C_val u_y)) 0.0)
                                       pt_F     (list (+ (car pt_V) (* f_dist u_x))
     (+ (cadr pt_V) (* f_dist u_y)) 0.0)
                                       min_dist (min (distance pt_V P1) (distance pt_V P2)
     (abs C_val)))
                                 (if (< min_dist MAX_DIST)
                                     (progn (setq curr_branch_V (cons (list pt_V 'PARABOLA ang
     _rad f_dist pt_V pt_F) curr_branch_V))
                                            (setq curr_branch_F (cons (list pt_F 'PARABOLA ang
     _rad f_dist pt_V pt_F) curr_branch_F)))
                                     (progn (if curr_branch_V (setq branches_V (cons (reverse
     curr_branch_V) branches_V) curr_branch_V nil))
                                            (if curr_branch_F (setq branches_F (cons (reverse
     curr_branch_F) branches_F) curr_branch_F nil)))))))))
          (setq prev_den den))
  (if curr_branch_V (setq branches_V (cons (reverse curr_branch_V) branches_V)))
  (if curr_branch_F (setq branches_F (cons (reverse curr_branch_F) branches_F)))
  (list (apply 'append branches_V) (apply 'append branches_F) nil))
;;-------------------------------------------------------------------------------------
(defun PAR2PFAPI_Engine_Improprio (P1 P2 vec_dir opt_e / C_pt a_val c_dist F_hyp V_hyp e_mark e
     nts_iperbole obj_spline len_tot step curr_d pt_F branch_F branch_V u_x u_y n_x n_y delta_P
```

```
 12_x delta_P12_y const_den const_num f_dist pt_V ang_rad payload_F payload_V vla_list)
(setq C_pt   (Vet_Mid P1 P2)
     c_dist (distance C_pt P1)
      F_hyp  P1
     a_val  (* c_dist (abs (cos (- (angle C_pt P1) (angle '(0 0 0) vec_dir))))))
(if (> a_val 1e-6)
    (progn (setq V_hyp       (Vet_Sum C_pt (Vet_Sca (Vet_Uni (Vet_Sub F_hyp C_pt)) a_val))
                 u_x         (car  vec_dir)
               u_y         (cadr vec_dir)
               n_x         (- u_y)
               n_y         u_x
                 ang_rad     (angle '(0 0 0) vec_dir)
                 delta_P12_x (- (car  P1) (car  P2))
               delta_P12_y (- (cadr P1) (cadr P2))
                 const_den   (* 4.0 (+ (* delta_P12_x u_x) (* delta_P12_y u_y)))
                 const_num          (+ (* delta_P12_x n_x) (* delta_P12_y n_y))
                 e_mark      (entlast))
           (Draw_ConIpe_CeVeFu_Light C_pt V_hyp F_hyp (list V_hyp (Vet_Sum C_pt (Vet_Sca ve
 c_dir (* a_val opt_e 15.0)))))
           (setq ents_iperbole (Util_Collect_Ents e_mark)
 branch_F nil branch_V nil vla_list nil)
           (foreach ent ents_iperbole
                   (if (= (cdr (assoc 0 (entget ent))) "SPLINE")
                       (progn (setq obj_spline (vlax-ename->vla-object ent)
                                    vla_list   (cons obj_spline vla_list)
                                    len_tot    (vlax-curve-getDistAtParam obj_spline (vlax-
 curve-getEndParam obj_spline))
                                    step       (/ len_tot 2048.0)
                              curr_d     0.0)
                             (vla-put-visible obj_spline :vlax-false)
                             (while (<= curr_d len_tot)
                                    (setq pt_F      (vlax-curve-getPointAtDist obj_spline
 curr_d)
                                          f_dist    (PAR2PFAPI_Calc_F_Linear pt_F P1 P2 u_
 x u_y n_x n_y const_num const_den)
                                          pt_V      (list (- (car pt_F) (* f_dist u_x))
 (- (cadr pt_F) (* f_dist u_y)) 0.0)
                                          payload_V (list pt_V 'PARABOLA ang_rad f_dist pt
 _V pt_F)
                                          payload_F (list pt_F 'PARABOLA ang_rad f_dist pt
 _V pt_F)
                                          branch_V  (cons payload_V branch_V)
                                    branch_F  (cons payload_F branch_F)
                                          curr_d    (+ curr_d step))))))
           (list (reverse branch_V) (reverse branch_F) vla_list))
```

```
        (progn (list nil nil nil))))
;;-------------------------------------------------------------------------------------
(defun PAR2PFAPI_Universal_Loop (P1 P2 flat_map_V ents_V flat_map_F ents_F f_type vla_list P3_r
     ef vec_ref / loop_main opt_gf opt_gv opt_m p_input last_run_ents best_p res_v res_f)
    (setq loop_main T opt_gv "ON" opt_gf "OFF" opt_m "Sostituisci" last_run_ents nil)
    (Util_Update_Guides_Vis "ON" ents_V) (Util_Update_Guides_Vis "OFF" ents_F)
    (while loop_main
           (setvar "OSMODE" 0)
           (setq p_input (getpoint (PAR2PFAPI_Helper_UI_Prompt opt_gf opt_gv opt_m)))
           (cond ((null p_input)        (setq loop_main nil))
                 ((= p_input "Uscita")  (setq loop_main nil))
                  ((= p_input "Fuochi")  (setq opt_gf    (if (= opt_gf "ON") "OFF" "ON"))
                                         (Util_Update_Guides_Vis opt_gf ents_F))
                  ((= p_input "Vertici") (setq opt_gv    (if (= opt_gv "ON") "OFF" "ON"))
                                         (Util_Update_Guides_Vis opt_gv ents_V))
                  ((= p_input "Modo")    (setq opt_m     (if (= opt_m "Sostituisci")
     "Aggiungi" "Sostituisci") last_run_ents nil))
                  ((listp p_input)       (setq best_p nil)
                                         (cond ((and (= opt_gv "ON")  (= opt_gf "OFF"))
     (setq best_p (PAR2PFAPI_Refine_Locus p_input P1 P2 flat_map_V f_type vla_list P3_ref vec_r
     ef "V")))
                                               ((and (= opt_gv "OFF") (= opt_gf "ON"))
     (setq best_p (PAR2PFAPI_Refine_Locus p_input P1 P2 flat_map_F f_type vla_list P3_ref vec_r
     ef "F")))
                                               ((and (= opt_gv "ON")  (= opt_gf "ON"))
     (setq res_v  (PAR2PFAPI_Refine_Locus p_input P1 P2 flat_map_V f_type vla_list P3_ref vec_r
     ef "V")
                                                                                          re
     s_f  (PAR2PFAPI_Refine_Locus p_input P1 P2 flat_map_F f_type vla_list P3_ref vec_ref "F"))
                                                                                (if (< (
     distance p_input (car res_v)) (distance p_input (car res_f)))
                                                                                   (setq best_p
res_v)
                                                                                   (setq best_p
res_f))))
                                         (if best_p (setq last_run_ents (PAR2PFAPI_Draw_Engine P
     1 P2 best_p opt_m last_run_ents)) (princ "\n[!] Attiva un luogo.")))))
    (if (= opt_gv "OFF") (Util_Del_Ents_List ents_V)) (if (= opt_gf "OFF")
     (Util_Del_Ents_List ents_F))
    (foreach vla_obj vla_list (vla-delete vla_obj)))
;;-------------------------------------------------------------------------------------
(defun PAR2PFAPI_Draw_Engine (P1 P2 payload opt_m last_ents / e_marker pt_V pt_F)
    (if (and (= opt_m "Sostituisci") last_ents)
        (foreach e last_ents (if (entget e) (entdel e))))
    (setq e_marker (entlast)
```

```
          pt_V     (nth 4 payload)
          pt_F     (nth 5 payload))
    (Draw_ConPar_VeFu_Compl pt_V pt_F (list P1 P2 pt_V))
    (Safe_Collect_Ents e_marker))
;;-------------------------------------------------------------------------------------
  (defun ssadd_list (ent_list / ss e)
    (setq ss (ssadd))
    (foreach e ent_list (if (entget e) (ssadd e ss)))
    ss)
;;-------------------------------------------------------------------------------------
(defun PAR2PFAPI_Libero_Runner (P1 P2 / loop_active V pts_rel Step Ang deg2rad prev_err curr_er
     r min_a max_a mid_a mid_err Solutions is_dup c s x1 y1 A_val p_dist F_loc F_wcs num_found
     opt_g opt_m guide_ents last_run_ents e_marker)
    (setq opt_g         "off"
          opt_m         "Sostituisci"
          guide_ents    nil
          last_run_ents nil
          deg2rad       (/ pi 180.0)
           guide_ents    (Helper_Update_Dima_Par2pv "No" guide_ents P1 P2)
           loop_active   T)
    (while loop_active
           (setvar "OSMODE" 512) (initget "Guide Modo Uscita")
           (setq V (getpoint (strcat "\nGUIDE: " opt_g " - MODO: " opt_m " | Specifica il Verti
     ce (V) o [Guide/Modo/Uscita] <Uscita>: ")))
           (setvar "OSMODE" 0)
           (cond  ((null V)       (setq loop_active   nil))
                ((= V "Uscita") (setq loop_active   nil))
                  ((= V "Guide")  (setq opt_g         (if (= opt_g "on") "off" "on")
                                   guide_ents    (Helper_Update_Dima_Par2pv (if (= opt_g "on") "Si"
"No") guide_ents P1 P2)))
                  ((= V "Modo")   (setq opt_m         (if (= opt_m "Sostituisci")
     "Aggiungi" "Sostituisci")
                                   last_run_ents nil))
                  ((listp V)      (if (and (= opt_m "Sostituisci") last_run_ents)
                                  (foreach e last_run_ents (if (entget e) (entdel e))))
                                  (setq e_marker  (entlast)
                                   pts_rel   (list (Vet_Sub P1 V) (Vet_Sub P2 V))
                                   Step      0.5
                                   Ang       0.0
                                   Solutions nil
                                   prev_err  (Geom_Eval_Par_Error pts_rel 1.0 0.0))
                                  (repeat 360
                                         (setq Ang      (+ Ang Step)
                                          curr_err (Geom_Eval_Par_Error pts_rel (cos (* Ang
deg2rad)) (sin (* Ang deg2rad))))
```

```
                                    (if (< (* prev_err curr_err) 0)
                                        (progn (setq min_a  (- Ang Step)
                                                     max_a  Ang)
                                                (repeat 20
                                                 (setq mid_a   (* 0.5 (+ min_a max_a))
                                                       mid_err (Geom_Eval_Par_Error pts_rel (cos
(* mid_a deg2rad))
     (sin (* mid_a deg2rad))))
                                                   (if (< (* mid_err prev_err) 0)
                                                  (setq max_a   mid_a)
                                                  (setq min_a    mid_a
                                                        prev_err mid_err)))
                                                (setq is_dup nil)
                                               (foreach s_val Solutions
                                                       (if (< (abs (- mid_a s_val)) 1.0)
                                                        (setq is_dup T)))
                                                (if (not is_dup)
                                               (setq Solutions (cons mid_a Solutions)))))
                                    (setq prev_err curr_err))
                                (setq num_found 0)
                                (foreach sol Solutions
                                         (setq c  (cos (* sol deg2rad))
                                         s  (sin (* sol deg2rad))
                                         x1 (caar  (Util_Rot_List_2D_CS pts_rel c s))
                                         y1 (cadar (Util_Rot_List_2D_CS pts_rel c s)))
                                         (if (> (abs x1) 1e-6)
                                             (progn (setq A_val (/ y1 (* x1 x1)))
                                                    (if (> (abs A_val) 1e-9)
                                                        (progn (setq p_dist (/ 1.0 (* 4.0 (ab
     s A_val)))
                                                               F_loc  (if (> A_val 0) (list 0.0
p_dist 0.0) (list 0.0 (- p_dist) 0.0))
                                                                    F_wcs  (Vet_Sum V (list
     (car (Util_Rot_2D_Inv_CS F_loc c s)) (cadr (Util_Rot_2D_Inv_CS F_loc c s)) 0.0)))
                                                                 (Draw_ConPar_VeFu_Compl V F_wc
     s (list P1 P2 V))
                                                        (setq num_found (1+ num_found)))))))
                                (setq last_run_ents (Util_Collect_Ents e_marker))
                               (princ (strcat "\n[OK] Trovate " (itoa num_found) " parabole."))))))
    (if (and (= opt_g "off") guide_ents)
        (foreach e guide_ents (if (entget e) (entdel e)))))
;;-------------------------------------------------------------------------------------
(defun PAR2PFAPI_Get_Top3_Indices (pt_mouse flat_map / i d candidates)
  (setq i 0 candidates nil)
  (foreach p flat_map
```

```
          (setq d          (distance pt_mouse (car p))
                candidates (cons (list d i) candidates)
                i          (1+ i)))
  (setq candidates (vl-sort candidates '(lambda (a b) (< (car a) (car b)))))
  (list (nth 0 candidates) (nth 1 candidates) (nth 2 candidates)))
;;-------------------------------------------------------------------------------
(defun PAR2PFAPI_Refine_Locus_Ultra (pt_mouse P1 P2 map_data f_type vla_list P3_ref vec_ref act
    ive_loc / D12 top3 branches_to_scan best_payload best_dist_m idx_val idx low_idx high_idx
    ang_start ang_end step_count j cur_a u_x u_y n_x n_y dx1 dy1 dx2 dy2 x1 y1 x2 y2 den A_val
    C_val f_dist pt_V pt_F payload d_m vla_obj pt_anchor dist_p dist_step curr_d cur_pt_F delt
    a_P12_x delta_P12_y const_den const_num branch_payload found_better)
  (setq D12              (distance P1 P2)
        top3             (PAR2PFAPI_Get_Top3_Indices pt_mouse map_data)
        branches_to_scan  nil
        best_payload     (nth (cadr (car top3)) map_data)
        best_dist_m      (caar top3)
        found_better      nil)
  (foreach i_data top3
          (if (and i_data (setq idx_val (cadr i_data)))
              (if (not (vl-some '(lambda (x) (or (= x idx_val) (= x (1- idx_val))
    (= x (1+ idx_val)))) branches_to_scan))
              (setq branches_to_scan (cons idx_val branches_to_scan)))))

  (cond ((= f_type "Proprio") (foreach idx branches_to_scan
                                      (setq low_idx    (max 0 (1- idx))
                                            high_idx   (min (1- (length map_data)) (1+ idx))
                                            ang_start  (nth 2 (nth low_idx map_data))
                                            ang_end    (nth 2 (nth high_idx map_data))
                                            step_count  30
                                       j           0)
                                      (repeat (1+ step_count)
                                              (setq cur_a (+ ang_start (* j (/ (- ang_end ang_
    start) (float step_count))))
                                                    u_x   (cos cur_a)
                                              u_y   (sin cur_a)
                                              n_x   (- u_y)
                                              n_y    u_x
                                                    dx1   (- (car  P1) (car  P3_ref))
                                              dy1   (- (cadr P1) (cadr P3_ref))
                                                    dx2   (- (car  P2) (car  P3_ref))
                                              dy2   (- (cadr P2) (cadr P3_ref))
                                                    x1    (+ (* dx1 n_x) (* dy1 n_y))
                                              y1    (+ (* dx1 u_x) (* dy1 u_y))
                                                    x2    (+ (* dx2 n_x) (* dy2 n_y))
                                              y2    (+ (* dx2 u_x) (* dy2 u_y))
```

```
                                        den   (- (* x1 x1)   (* x2 x2)))
                                    (if (> (abs den) 1e-9)
                                        (progn (setq A_val (/ (- y1 y2) den))
                                               (if (> (abs A_val) 1e-12)
                                                   (progn (setq C_val   (- y1 (* A_v
al x1 x1))
                                                                f_dist  (/ 1.0 (* 4.
0 A_val))
                                                                pt_V    (list (+ (ca
r P3_ref) (* C_val u_x))   (+ (cadr P3_ref) (* C_val u_y))  0.0)
                                                                pt_F    (list (+ (ca
r pt_V)   (* f_dist u_x))  (+ (cadr pt_V)   (* f_dist u_y)) 0.0)
                                                                payload (list (if (=
active_loc "V") pt_V pt_F) 'PARABOLA cur_a f_dist pt_V pt_F)
                                                                d_m     (distance pt
_mouse (car payload)))
                                                          (if (< d_m best_dist_m)
                                                              (setq best_dist_m   d_
m
                                                                    best_payload  payload
                                                                    found_better  T))))))
                                    (setq j (1+ j)))))
   ((= f_type "Improprio") (setq u_x         (car vec_ref)
                                 u_y         (cadr vec_ref)
                                 n_x         (- u_y)
                                 n_y         u_x
                                 delta_P12_x (- (car  P1) (car  P2))
                                 delta_P12_y (- (cadr P1) (cadr P2))
                                 const_den   (* 4.0 (+ (* delta_P12_x u_x)
(* delta_P12_y u_y)))
                                 const_num          (+ (* delta_P12_x n_x)
(* delta_P12_y n_y)))
                           (foreach vla_obj vla_list
                                    (foreach idx branches_to_scan
                                             (setq branch_payload (nth idx map_data)
                                                   pt_anchor      (if (= active_loc "V")
                                                                      (nth 5 branch_payloa
d)
                                                                      (vlax-curve-getClose
stPointTo vla_obj pt_mouse))
                                                   dist_p         (vlax-curve-getDistAtPoi
nt vla_obj pt_anchor))
                                             (if dist_p
                                                 (progn  (setq dist_step (/ (/ D12 50.0)
30.0) j -15)
```

```
                                                        (while (<= j 15)
                                                               (setq curr_d (+ dist_p (*
    j dist_step)))
                                                               (if (and (>= curr_d 0)
    (<= curr_d (vlax-curve-getDistAtParam vla_obj (vlax-curve-getEndParam vla_obj))))
                                                                   (setq cur_pt_F (vlax-c
    urve-getPointAtDist vla_obj curr_d)
                                                                         f_dist   (PAR2PF
    API_Calc_F_Linear cur_pt_F P1 P2 u_x u_y n_x n_y const_num const_den)
                                                                         pt_V     (list (
    - (car cur_pt_F) (* f_dist u_x)) (- (cadr cur_pt_F) (* f_dist u_y)) 0.0)
                                                                         payload  (list (
    if (= active_loc "V") pt_V cur_pt_F) 'PARABOLA (angle '(0 0 0) vec_ref)
    f_dist pt_V cur_pt_F)
                                                                         d_m      (distan
    ce pt_mouse (car payload)))
                                                                   (setq cur_pt_F nil))
                                                               (if (and cur_pt_F (< d_m b
    est_dist_m))
                                                                   (setq best_dist_m d_m
    best_payload payload found_better T))
                                                               (setq j (1+ j)))))))))
  (if found_better
      (princ (strcat "\n[LUSNAP] Raffinato a dist: " (rtos best_dist_m 2 4)))
      (princ "\n[LUSNAP] Nodo discreto ottimale."))
  best_payload)
;;-------------------------------------------------------------------------------------
(defun PAR2PFAPI_Universal_Loop (P1 P2 flat_map_V ents_V flat_map_F ents_F f_type vla_list P3_r
    ef vec_ref / loop_main opt_gf opt_gv opt_m p_input last_run_ents best_p res_v res_f)
  (setq loop_main     T
        opt_gv        "ON"
        opt_gf        "OFF"
        opt_m         "Sostituisci"
        last_run_ents nil)
  (Util_Update_Guides_Vis "ON"  ents_V)
  (Util_Update_Guides_Vis "OFF" ents_F)
  (while loop_main
         (setvar "OSMODE" 0)
         (setq p_input (getpoint (PAR2PFAPI_Helper_UI_Prompt opt_gf opt_gv opt_m)))
         (cond ((null p_input)        (setq loop_main nil))
               ((= p_input "Uscita")  (setq loop_main nil))
               ((= p_input "Fuochi")  (setq opt_gf    (if (= opt_gf "ON") "OFF" "ON"))
                                      (Util_Update_Guides_Vis opt_gf ents_F))
               ((= p_input "Vertici") (setq opt_gv    (if (= opt_gv "ON") "OFF" "ON"))
                                      (Util_Update_Guides_Vis opt_gv ents_V))
```

```
            ((= p_input "Modo")    (setq opt_m     (if (= opt_m "Sostituisci")
     "Aggiungi" "Sostituisci") last_run_ents nil))
            ((listp p_input)       (setq best_p nil)
                                   (cond ((and (= opt_gv "ON") (= opt_gf "OFF"))
     (setq best_p (PAR2PFAPI_Refine_Locus_Ultra p_input P1 P2 flat_map_V f_type vla_list P3_ref
     vec_ref "V")))
                                         ((and (= opt_gv "OFF") (= opt_gf "ON"))
     (setq best_p (PAR2PFAPI_Refine_Locus_Ultra p_input P1 P2 flat_map_F f_type vla_list P3_ref
     vec_ref "F")))
                                         ((and (= opt_gv "ON") (= opt_gf "ON"))
     (setq res_v  (PAR2PFAPI_Refine_Locus_Ultra p_input P1 P2 flat_map_V f_type vla_list P3_ref
     vec_ref "V")
                                                                                    res_
     f  (PAR2PFAPI_Refine_Locus_Ultra p_input P1 P2 flat_map_F f_type vla_list P3_ref vec_ref "
     F"))
                                                                                (if (< (di
     stance p_input (car res_v)) (distance p_input (car res_f)))
                                                                            (setq best_p res_v)
(setq best_p res_f))))
                                    (if best_p
                                   (setq last_run_ents (PAR2PFAPI_Draw_Engine P1 P2 best_p opt_m
last_run_ents))
                                   (princ "\n[!] Attiva un luogo.")))))
  (if (= opt_gv "OFF")
      (Util_Del_Ents_List ents_V))
  (if (= opt_gf "OFF")
      (Util_Del_Ents_List ents_F))
  (foreach vla_obj vla_list (vla-delete vla_obj)))
;;-------------------------------------------------------------------------------------------
(defun c:par2PFAPI (/ pts old_os old_pd old_ce P1 P2 f_choice res flat_map_V flat_map_F e_mark
     ents_V ents_F t_start t_end elapsed P3_val vec_val D1 D2 vla_list P_mid)
    (mapcar 'set '(pts old_os old_pd old_ce)
     (Init_Conic_App "LABORATORIO PARABOLA 2P (API)" 2 "\nSeleziona 2 punti base: " 32))
    (if pts
      (progn
        (mapcar 'set '(P1 P2) pts)
        (Draw_Line_2Pu_R P1 P2 *LaCo_Poligoni_coniche*)
        (initget "Proprio Improprio Libero")
        (setq f_choice (if (setq f_choice (getkword "\nFascio: [Proprio/Improprio/Libero] <Libe
     ro>: ")) f_choice "Libero"))
        (cond
          ((= f_choice "Libero")      (PAR2PFAPI_Libero_Runner P1 P2))

          ((or (= f_choice "Proprio")
               (= f_choice "Improprio"))
```

```
        (setvar "OSMODE" (if old_os old_os 0))
        (if (= f_choice "Proprio")
            (if (setq P3_val (getpoint "\nSpecifica P3 (Perno di rotazione): "))
                (progn (Draw_Point_R P3_val *LaCo_Punti_scelti*)
                       (Draw_Line_2Pu_R P1 P3_val *LaCo_Costruzioni*)
                       (Draw_Line_2Pu_R P2 P3_val *LaCo_Costruzioni*)
                       (setq t_start (* 86400.0 (getvar "DATE"))
                             res     (PAR2PFAPI_Engine_Proprio P1 P2 P3_val 1.0))))
            (progn  (setq P_mid (Vet_Mid P1 P2))
                    (if (setq D2 (getpoint P_mid "\nDefinisci direzione asse (origine in Mid
    P1P2): "))
                        (progn (setq vec_val (Vet_Uni (Vet_Sub D2 P_mid)))
                               (Draw_Line_2Pu_R P1 D2 *LaCo_Costruzioni*)
                               (Draw_Line_2Pu_R P2 D2 *LaCo_Costruzioni*)
                               (setq t_start (* 86400.0 (getvar "DATE"))
                                     res     (PAR2PFAPI_Engine_Improprio P1 P2 vec_val 1.0))
    ))))
        (if (and res (car res))
            (progn (setq t_end (* 86400.0 (getvar "DATE")) elapsed (- t_end t_start))
                   (princ (strcat " Mappe OK (" (rtos elapsed 2 3) "s)"))
                   (setq flat_map_V (car res)
                      flat_map_F (cadr res)
                      vla_list   (nth 2 res))
                   (setq e_mark (entlast))
                 (setvar "CECOLOR" "5")
                 (foreach p flat_map_V
                         (Draw_Point_R (car p) *LaCo_Luoghi_geometrici*))
                   (setq ents_V (Util_Collect_Ents e_mark))
                 (if ents_V
                    (command "_.-GROUP" "_Create" "*" "LV" (ssadd_list ents_V) ""))
                   (setq e_mark (entlast))
                 (setvar "CECOLOR" "3")
                 (foreach p flat_map_F
                         (Draw_Point_R (car p) *LaCo_Luoghi_geometrici*))
                   (setq ents_F (Util_Collect_Ents e_mark))
                 (if ents_F
                    (command "_.-GROUP" "_Create" "*" "LF" (ssadd_list ents_F) ""))
                   (setvar "CECOLOR" "BYLAYER")
                 (PAR2PFAPI_Universal_Loop P1 P2 flat_map_V ents_V flat_map_F ents_F f_choice
vla_list P
    3_val vec_val)))))))
    (Restore_Conic_App old_os old_pd old_ce))
)
;;-------------------------------------------------------------------------------
```

```
;;-------------------------------------------------------------------------------------

;;-------------------------------------------------------------------------------------
;;-------------------------------------------------------------------------------------
(progn
;;-------------------------------------------------------------------------------------
(defun PAR2PFAPI_Helper_Nearest_Control_Pt (pt_click P1 P2 P3 / d1 d2 d3 d_min)
  (setq d1 (distance pt_click P1) d2 (distance pt_click P2) d3 (distance pt_click P3)
        d_min (min d1 d2 d3))
  (cond ((= d_min d1) 'P1) ((= d_min d2) 'P2) (T 'P3)))
;;-------------------------------------------------------------------------------------
(defun PAR2PFAPI_Draw_Skeleton_Rea (P1 P2 P3 f_type / e_list m_pt)
  (setq e_list (list
                 (Draw_Point_R P1 *LaCo_Punti_scelti*)
                 (Draw_Point_R P2 *LaCo_Punti_scelti*)
                 (Draw_Point_R P3 *LaCo_Punti_scelti*)
                 (Draw_Line_2Pu_R P1 P2 *LaCo_Poligoni_coniche*)
                 (Draw_Line_2Pu_R P1 P3 *LaCo_Costruzioni*)
                 (Draw_Line_2Pu_R P2 P3 *LaCo_Costruzioni*)
               ))
  (if (= f_type "Improprio")
      (progn
        (setq m_pt (Vet_Mid P1 P2))
        (setq e_list (cons (Draw_Line_2Pu_R m_pt P3 *LaCo_Costruzioni*) e_list))))
  (vl-remove nil e_list))
;;-------------------------------------------------------------------------------------
(defun PAR2PFAPI_Draw_Locus_Entities_Rea (res_map opt_gv opt_gf / e_mark ents_V ents_F)
  (setq ents_V nil ents_F nil)
  (if (and (= opt_gv "ON") (car res_map))
      (progn
        (setq e_mark (entlast))
        (setvar "CECOLOR" "5")
        (foreach p (car res_map) (Draw_Point_R (car p) *LaCo_Luoghi_geometrici*))
        (setq ents_V (Util_Collect_Ents e_mark))))
  (if (and (= opt_gf "ON") (cadr res_map))
      (progn
        (setq e_mark (entlast))
        (setvar "CECOLOR" "3")
        (foreach p (cadr res_map) (Draw_Point_R (car p) *LaCo_Luoghi_geometrici*))
        (setq ents_F (Util_Collect_Ents e_mark))))
  (setvar "CECOLOR" "BYLAYER")
  (list ents_V ents_F))
;;-------------------------------------------------------------------------------------
(defun c:par2PFAPIRea (/ P1 P2 P3 f_type loop opt_gv opt_gf pt_in pt_final res_map ents_locus e
     nts_skel v_rel vec_dir near_tag old_os p_old tag_str ents_V ents_F)
```

```
(setq old_os (getvar "OSMODE"))
(setvar "CMDECHO" 0) (setvar "OSMODE" 32)

(princ "\n--- CONFIGURAZIONE LABORATORIO REA (V. 1.7) ---")

(if (and (setq P1 (getpoint "\nDefinisci P1: "))
         (setq P2 (getpoint P1 "\nDefinisci P2: ")))
  (progn
    (setq ents_skel (list (Draw_Line_2Pu_R P1 P2 *LaCo_Poligoni_coniche*)
                          (Draw_Point_R P1 *LaCo_Punti_scelti*)
                          (Draw_Point_R P2 *LaCo_Punti_scelti*)))
    (command "_.redraw")

    (initget "Proprio Improprio")
    (setq f_type (getkword "\nTipo fascio [Proprio/Improprio] <Proprio>: "))
    (if (not f_type) (setq f_type "Proprio"))

    (setvar "OSMODE" 0)
    (setq P3 (getpoint (Vet_Mid P1 P2) (if (= f_type "Proprio")
   "\nPerno rotazione P3: " "\nDirezione asse P3: ")))

    (setq loop T opt_gv "ON" opt_gf "ON" ents_locus nil)
    (setq v_rel (Vet_Sub P3 (Vet_Mid P1 P2)))

    (while loop
      (if ents_locus (Util_Del_Ents_List (append (car ents_locus) (cadr ents_locus))))
      (if ents_skel (Util_Del_Ents_List ents_skel))

      (setq ents_skel (PAR2PFAPI_Draw_Skeleton_Rea P1 P2 P3 f_type))

      (setq res_map
        (if (= f_type "Proprio")
            (PAR2PFAPI_Engine_Proprio P1 P2 P3 1.0)
            (progn
              (setq vec_dir (Vet_Uni (Vet_Sub P3 (Vet_Mid P1 P2))))
              (PAR2PFAPI_Engine_Improprio P1 P2 vec_dir 1.0))))

      (setq ents_locus (PAR2PFAPI_Draw_Locus_Entities_Rea res_map opt_gv opt_gf))
      (setq ents_V (car ents_locus) ents_F (cadr ents_locus))

      (if ents_V (command "_.-GROUP" "_Create" "*" "LV" (ssadd_list ents_V) ""))
      (if ents_F (command "_.-GROUP" "_Create" "*" "LF" (ssadd_list ents_F) ""))

      (command "_.redraw")
```

```
        (initget "Fuochi Vertici Uscita")
        (setq pt_in (getpoint (strcat "\n[REA-" f_type "] F:" opt_gf " V:" opt_gv " | Seleziona
     P1/P2/P3 o [Fuochi/Vertici/Uscita] <Uscita>: ")))

        (cond
          ((or (null pt_in) (= pt_in "Uscita")) (setq loop nil))
          ((= pt_in "Fuochi")  (setq opt_gf (if (= opt_gf "ON") "OFF" "ON")))
          ((= pt_in "Vertici") (setq opt_gv (if (= opt_gv "ON") "OFF" "ON")))
          ((listp pt_in)
           (setq near_tag (PAR2PFAPI_Helper_Nearest_Control_Pt pt_in P1 P2 P3))
           (setq p_old (eval near_tag))
           (setq tag_str (vl-princ-to-string near_tag))

           (princ (strcat "\nSposto " tag_str ". Specifica nuova posizione: "))
           (setq pt_final (getpoint p_old))

           (if pt_final
             (progn
               (cond
                 ((or (= near_tag 'P1) (= near_tag 'P2))
                  (if (= near_tag 'P1) (setq P1 pt_final) (setq P2 pt_final))
                  (setq P3 (Vet_Sum (Vet_Mid P1 P2) v_rel)))
                 ((= near_tag 'P3)
                  (setq P3 pt_final)
                  (setq v_rel (Vet_Sub P3 (Vet_Mid P1 P2)))))
               (princ (strcat "\n[REA] " tag_str " aggiornato.")))
             (princ "\n[!] Operazione annullata."))
           )
          )
        )

       (princ "\n[SYSTEM] Consolidamento finale...")
       (PAR2PFAPI_Draw_Skeleton_Rea P1 P2 P3 f_type)

       (if (and (= f_type "Improprio") (nth 2 res_map)) (foreach obj (nth 2 res_map)
     (vla-delete obj)))
    )
  )

  (setvar "OSMODE" old_os)
  (princ "\n[OK] Laboratorio REA terminato. Dati raggruppati.") (princ))
)
;;-------------------------------------------------------------------------------------
;;-------------------------------------------------------------------------------------
```

```
;;-------------------------------------------------------------------------------
;;-------------------------------------------------------------------------------
(progn
;;-------------------------------------------------------------------------------
(defun ELL3P_Helper_UI_Prompt (opt_gc opt_gf opt_gv opt_t opt_m)
  (if (= opt_gv "ON")
      (progn
        (initget "Centri Fuochi Vertici Target Modo Uscita")
        (strcat "\nC:" opt_gc " F:" opt_gf " V:" opt_gv " T:" opt_t " | M:" opt_m " | Punto o [
     Centri/Fuochi/Vertici/Target/Modo/Uscita] <Uscita>: "))
      (progn
        (initget "Centri Fuochi Vertici Modo Uscita")
        (strcat "\nC:" opt_gc " F:" opt_gf " V:" opt_gv " | M:" opt_m " | Punto o [Centri/Fuoch
     i/Vertici/Modo/Uscita] <Uscita>: "))))
;;-------------------------------------------------------------------------------
(defun ELL3P_Master_Engine (pts mode fixed_val / P1 pts_rel G_bary check_dist local_os K_HD ste
     p list_C map_C list_F map_F list_V map_V i ang_rad ratios_to_check current_ratio c s pts_l
     oc solver_res C_loc a_val b_val C_wcs_rel C_wcs V_maj_wcs V_min_wcs c_foc F1 F2 V1 V2 g_da
     ta e_start guide_C guide_F guide_V)
  (setq P1 (car pts))
  (setq G_bary     (Pu_Baricentro2D_LisPu pts)
        check_dist (* 100.0 (apply 'max (mapcar '(lambda (p) (distance p G_bary)) pts)))
        local_os   (getvar "OSMODE"))

  (cond ((= mode 1)  (setq K_HD 7200.0  step (/ 0.95  K_HD)))
        ((= mode 2)  (setq K_HD 3600.0  step (/ 180.0 K_HD))))

  (setvar "OSMODE" 0)
  (setq list_C nil map_C nil list_F nil map_F nil list_V nil map_V nil i 0)

  (repeat (fix (1+ K_HD))
    (cond
      ((= mode 1)
       (setq ang_rad fixed_val
             ratios_to_check (list (+ 0.05 (* i step)))))
      ((= mode 2)
       (setq ang_rad (* i step (/ pi 180.0))
             ratios_to_check (list fixed_val))))

    (setq c (cos ang_rad) s (sin ang_rad)
          pts_rel (mapcar '(lambda (p) (list (- (car p) (car P1)) (- (cadr p) (cadr P1)) 0.0))
     pts)
          pts_loc (Util_Rot_List_2D_CS pts_rel c s))

    (foreach current_ratio ratios_to_check
```

```
    (setq solver_res (Math_Solve_Ell_3P_Ratio pts_loc current_ratio))
    (if (and solver_res (listp solver_res))
      (progn
        (setq C_loc (car solver_res)
              a_val (cadr solver_res)
              b_val (* a_val current_ratio)
              C_wcs_rel (Util_Rot_2D_Inv_CS C_loc c s)
              C_wcs (list (+ (car C_wcs_rel) (car P1)) (+ (cadr C_wcs_rel) (cadr P1)) 0.0)
              V_maj_wcs (list c s 0.0)
              V_min_wcs (list (- s) c 0.0)
              c_foc (sqrt (abs (- (* a_val a_val) (* b_val b_val))))
              g_data (list 'ELLISSE current_ratio ang_rad C_wcs V_maj_wcs a_val b_val V_min_w
   cs c_foc))

        (if (< (distance C_wcs G_bary) check_dist)
            (setq list_C (cons C_wcs list_C) map_C (cons (cons C_wcs g_data) map_C)))

        (setq F1 (Vet_Sum C_wcs (Vet_Sca V_maj_wcs c_foc))
              F2 (Vet_Sum C_wcs (Vet_Sca V_maj_wcs (- c_foc))))
        (if (< (distance F1 G_bary) check_dist) (setq list_F (cons F1 list_F)
   map_F (cons (cons F1 g_data) map_F)))
        (if (< (distance F2 G_bary) check_dist) (setq list_F (cons F2 list_F)
   map_F (cons (cons F2 g_data) map_F)))

        (setq V1 (Vet_Sum C_wcs (Vet_Sca V_maj_wcs a_val))
              V2 (Vet_Sum C_wcs (Vet_Sca V_maj_wcs (- a_val))))
        (if (< (distance V1 G_bary) check_dist) (setq list_V (cons V1 list_V)
   map_V (cons (cons V1 g_data) map_V)))
        (if (< (distance V2 G_bary) check_dist) (setq list_V (cons V2 list_V)
   map_V (cons (cons V2 g_data) map_V))))))
  (setq i (1+ i)))

(setq e_start (entlast))
(setvar "CECOLOR" (itoa *ELL3P_Col_C*))
   (foreach pt list_C (Draw_Point_R pt *LaCo_Luoghi_geometrici*))
(setq guide_C (Safe_Collect_Ents e_start))
   (if guide_C (Util_Make_Anonymous_Group guide_C "Locus_Centri"))

(setq e_start (entlast))
(setvar "CECOLOR" (itoa *ELL3P_Col_F*))
   (foreach pt list_F (Draw_Point_R pt *LaCo_Luoghi_geometrici*))
(setq guide_F (Safe_Collect_Ents e_start))
   (if guide_F (Util_Make_Anonymous_Group guide_F "Locus_Fuochi"))

(setq e_start (entlast))
```

```
  (setvar "CECOLOR" (itoa *ELL3P_Col_V*))
     (foreach pt list_V (Draw_Point_R pt *LaCo_Luoghi_geometrici*))
  (setq guide_V (Safe_Collect_Ents e_start))
     (if guide_V (Util_Make_Anonymous_Group guide_V "Locus_Vertici"))

  (setvar "CECOLOR" "BYLAYER")
  (setvar "OSMODE" local_os)
  (list map_C guide_C map_F guide_F map_V guide_V))
;;-------------------------------------------------------------------------------------
(defun ELL3P_Eval_Param_Distance (pts mode fixed_val param target_type pt_mouse / P1 c s pts_re
     l pts_loc solver_res a_val b_val C_wcs_rel C_wcs V_maj_wcs V_min_wcs c_foc F1 F2 V1 V2 g_d
     ata min_d ang_rad cur_ratio)
  (setq P1 (car pts))
  (if (= mode 2)
      (setq ang_rad param        cur_ratio fixed_val)
      (setq ang_rad fixed_val    cur_ratio param))

  (setq c (cos ang_rad) s (sin ang_rad))
  (setq pts_rel (mapcar '(lambda (p) (list (- (car p) (car P1)) (- (cadr p) (cadr P1)) 0.0))
     pts)
        pts_loc (Util_Rot_List_2D_CS pts_rel c s)
        solver_res (Math_Solve_Ell_3P_Ratio pts_loc cur_ratio))

  (if (and solver_res (listp solver_res))
      (progn
        (setq a_val (cadr solver_res) b_val (* a_val cur_ratio)
              C_wcs_rel (Util_Rot_2D_Inv_CS (car solver_res) c s)
              C_wcs (list (+ (car C_wcs_rel) (car P1)) (+ (cadr C_wcs_rel) (cadr P1)) 0.0)
              V_maj_wcs (list c s 0.0)
              V_min_wcs (list (- s) c 0.0)
              c_foc (sqrt (abs (- (* a_val a_val) (* b_val b_val))))
              g_data (list 'ELLISSE cur_ratio ang_rad C_wcs V_maj_wcs a_val b_val V_min_wcs c_f
     oc)
              min_d 1e99)

        (cond  ((= target_type "Centri")      (setq min_d (distance C_wcs pt_mouse)))
               ((= target_type "Fuochi")
     (setq F1    (Vet_Sum C_wcs (Vet_Sca V_maj_wcs c_foc))
                                              F2    (Vet_Sum C_wcs (Vet_Sca V_maj_wcs (-
     c_foc)))
                                              min_d (min (distance F1 pt_mouse)
     (distance F2 pt_mouse))))
               ((= target_type "Vertici")
     (setq V1    (Vet_Sum C_wcs (Vet_Sca V_maj_wcs a_val))
                                              V2    (Vet_Sum C_wcs (Vet_Sca V_maj_wcs (-
```

```
    a_val)))
                                                    min_d (min (distance V1 pt_mouse)
    (distance V2 pt_mouse)))))
       (list min_d (cons pt_mouse g_data)))))
;;-------------------------------------------------------------------------------------
(defun ELL3P_Refine_Locus (pt_mouse pts map_data mode fixed_val target_type / min1 min2 item1 i
    tem2 d val1 val2 step i cur_val res best_res min_d)
  (setq min1 1e99 min2 1e99 item1 nil item2 nil)
  (foreach item map_data
    (setq d (distance pt_mouse (car item)))
    (cond
      ((< d min1) (setq min2 min1 item2 item1) (setq min1 d item1 item))
      ((< d min2) (setq min2 d item2 item))))

  (if (and item1 item2)
      (progn
        (setq val1 (if (= mode 2) (nth 2 (cdr item1)) (nth 1 (cdr item1)))
              val2 (if (= mode 2) (nth 2 (cdr item2)) (nth 1 (cdr item2)))
              step (/ (- val2 val1) 15.0)
              i 0 min_d 1e99 best_res nil)

        (while (<= i 15)
          (setq cur_val (+ val1 (* i step))
                res (ELL3P_Eval_Param_Distance pts mode fixed_val cur_val target_type pt_mouse)
    )
          (if (and res (< (car res) min_d))
              (setq min_d (car res) best_res res))
          (setq i (1+ i)))
        best_res)
      nil))
;;-------------------------------------------------------------------------------------
(defun ELL3P_Math_Error_Func (pts th current_ratio target_pt target_type / c s P1 pts_rel pts_l
    oc solver_res C_wcs_rel C_wcs a_val b_val c_foc d target_dist)
  (setq P1 (car pts) c (cos th) s (sin th)
        pts_rel (mapcar '(lambda (p) (list (- (car p) (car P1)) (- (cadr p) (cadr P1)) 0.0))
    pts)
        pts_loc (Util_Rot_List_2D_CS pts_rel c s)
        solver_res (Math_Solve_Ell_3P_Ratio pts_loc current_ratio))
  (if (and solver_res (listp solver_res))
      (progn
        (setq a_val (cadr solver_res) b_val (* a_val current_ratio)
              C_wcs_rel (Util_Rot_2D_Inv_CS (car solver_res) c s)
              C_wcs (list (+ (car C_wcs_rel) (car P1)) (+ (cadr C_wcs_rel) (cadr P1)) 0.0)
              d (distance C_wcs target_pt)
              c_foc (sqrt (abs (- (* a_val a_val) (* b_val b_val)))))
```

```
        (cond ((= target_type "V_maj") (setq target_dist a_val))
              ((= target_type "V_min") (setq target_dist b_val))
              ((= target_type "Fuoco") (setq target_dist c_foc)))
        (abs (- d target_dist)))
      nil))
;;-------------------------------------------------------------------------------------
(defun ELL3P_Eval_Cands (pts cur_ratio ang_rad target_type / P1 c s pts_rel pts_loc solver_res
     a_val b_val C_wcs_rel C_wcs V_maj_wcs V_min_wcs c_foc F1 F2 V1 V2 V3 V4 g_data cands)
  (setq P1 (car pts) c (cos ang_rad) s (sin ang_rad))
  (setq pts_rel (mapcar '(lambda (p) (list (- (car p) (car P1)) (- (cadr p) (cadr P1)) 0.0))
     pts)
        pts_loc (Util_Rot_List_2D_CS pts_rel c s)
        solver_res (Math_Solve_Ell_3P_Ratio pts_loc cur_ratio))

  (if (and solver_res (listp solver_res))
      (progn
        (setq a_val     (cadr solver_res) b_val (* a_val cur_ratio)
              C_wcs_rel (Util_Rot_2D_Inv_CS (car solver_res) c s)
              C_wcs     (list (+ (car C_wcs_rel) (car P1)) (+ (cadr C_wcs_rel) (cadr P1)) 0.0)
              V_maj_wcs (list c s 0.0)
              V_min_wcs (list (- s) c 0.0)
              c_foc     (sqrt (abs (- (* a_val a_val) (* b_val b_val))))
              g_data    (list 'ELLISSE cur_ratio ang_rad C_wcs V_maj_wcs a_val b_val V_min_wcs
     c_foc))

        (cond  ((= target_type "Fuochi")
     (setq F1    (Vet_Sum C_wcs (Vet_Sca V_maj_wcs c_foc))
                                             F2    (Vet_Sum C_wcs (Vet_Sca V_maj_wcs (- c_f
     oc)))
                                             cands (list (cons F1 g_data)
     (cons F2 g_data))))
               ((= target_type "Vertici")
     (setq V1    (Vet_Sum C_wcs (Vet_Sca V_maj_wcs a_val))
                                             V2    (Vet_Sum C_wcs (Vet_Sca V_maj_wcs (- a_v
     al)))
                                             V3    (Vet_Sum C_wcs (Vet_Sca V_min_wcs b_val)
     )
                                             V4    (Vet_Sum C_wcs (Vet_Sca V_min_wcs (- b_v
     al)))
                                             cands (list (cons V1 g_data) (cons V2 g_data)
     (cons V3 g_data) (cons V4 g_data)))))
        cands)))
;;-------------------------------------------------------------------------------------
(defun ELL3P_Solve_Exact_Node_Algebraic (p_target pts mode fixed_val active_node_type / target_
     types t_type solutions limit step i param err_list e_val prev_e curr_e next_e local_minima
```

```
   min_param best_err p_fine fine_step j e_fine cands_pack)
(setq target_types (cond ((= active_node_type "Vertici") (list "V_maj" "V_min"))
                         ((= active_node_type "Fuochi")  (list "Fuoco")))
      solutions nil)

(if (= mode 1)
    (setq limit 1.0 step (/ 1.0 720.0))
    (setq limit pi  step (/ pi 720.0)))

(foreach t_type target_types
  (setq i 0 err_list nil)
  (while (<= i 720)
    (setq param (if (= mode 1) (+ 0.05 (* i step)) (* i step)))
    (setq e_val (if (= mode 1)
                    (ELL3P_Math_Error_Func pts fixed_val param p_target t_type)
                    (ELL3P_Math_Error_Func pts param fixed_val p_target t_type)))
    (setq err_list (append err_list (list (cons param e_val))))
    (setq i (1+ i)))

  (setq i 1 local_minima nil)
  (while (< i 720)
    (setq prev_e (cdr (nth (1- i) err_list))
          curr_e (cdr (nth i err_list))
          next_e (cdr (nth (1+ i) err_list)))

    (if (and prev_e curr_e next_e)
        (if (and (< curr_e prev_e) (< curr_e next_e) (< curr_e 50.0))
            (setq local_minima (cons (car (nth i err_list)) local_minima))))
    (setq i (1+ i)))

  (foreach min_param local_minima
    (setq fine_step (/ (* step 2.0) 1000.0)
          j 0
          p_fine (- min_param step)
          best_err 1e99
          best_param nil)

    (while (<= j 1000)
      (if (or (and (= mode 1) (> p_fine 0.001) (< p_fine 1.0))
              (= mode 2))
          (progn
            (setq e_fine (if (= mode 1)
                             (ELL3P_Math_Error_Func pts fixed_val p_fine p_target t_type)
                             (ELL3P_Math_Error_Func pts p_fine fixed_val p_target t_type)))
            (if (and e_fine (< e_fine best_err))
```

```
                    (setq best_err e_fine best_param p_fine))))
          (setq p_fine (+ p_fine fine_step) j (1+ j)))

        (if (and best_param (< best_err 1e-4))
            (progn
              (setq cands_pack (if (= mode 1)
                                   (ELL3P_Eval_Cands pts best_param fixed_val active_node_type)
                                   (ELL3P_Eval_Cands pts fixed_val best_param active_node_type)))
              (foreach cand cands_pack
                (if (< (distance (car cand) p_target) 1e-3)
                    (setq solutions (cons cand solutions))))))))
  solutions)
;;-------------------------------------------------------------------------------------------
(defun ELL3P_Precalc_Exact_Nodes (pts mode fixed_val node_type / map_list sols)
  (setq map_list nil)
  (foreach p pts
    (setq sols (ELL3P_Solve_Exact_Node_Algebraic p pts mode fixed_val node_type))
    (setq map_list (cons (cons p sols) map_list)))
  map_list)
;;-------------------------------------------------------------------------------------------
(defun ELL3P_Draw_Engine (geo_packet opt_m last_ents / current_ratio Center V_dir a b V_min c_f
     oc A1 A2 B1 B2 F1 F2 e_marker)
  (if (and (= opt_m "Sostituisci") last_ents) (foreach e last_ents (if (entget e) (entdel e))))
  (setq e_marker      (entlast)
        current_ratio (nth 2 geo_packet)
        Center        (nth 4 geo_packet)
        V_dir         (nth 5 geo_packet)
        a             (nth 6 geo_packet)
        b             (nth 7 geo_packet)
        V_min         (nth 8 geo_packet)
        c_foc         (nth 9 geo_packet)

        A1            (Vet_Sum Center (Vet_Sca V_dir a))
        A2            (Vet_Sum Center (Vet_Sca V_dir (- a)))
        B1            (Vet_Sum Center (Vet_Sca V_min b))
        B2            (Vet_Sum Center (Vet_Sca V_min (- b)))
        F1            (Vet_Sum Center (Vet_Sca V_dir c_foc))
        F2            (Vet_Sum Center (Vet_Sca V_dir (- c_foc))))
  (Draw_ConEll_CeAx_Foci Center A1 A2 B1 B2 F1 F2 current_ratio)
  (Safe_Collect_Ents e_marker))
;;-------------------------------------------------------------------------------------------
(defun ELL3P_Get_Data (app_title / data pts)
  (setq data (Init_Conic_App app_title 3 "\nSeleziona 3 punti: " 0))
  (setq pts (car data))
  (if pts
```

```
    (progn  (Draw_Triangolo_Lis3Pu        pts *LaCo_Poligoni*)
            (Draw_Triangolo_Ortico_Lis3Pu pts *LaCo_Costruzioni*)))
  data)
;;-------------------------------------------------------------------------------------
(defun ELL3P_Get_Constraints (/ m_choice p_dir1 p_dir2 f_val mode_idx)
  (initget "Rapporto Direzione")
  (setq m_choice (if (setq m_choice (getkword "\nVincolo: [Rapporto/Direzione] <Rapporto>: "))
     m_choice "Rapporto"))
  (cond
    ((= m_choice "Direzione")
     (setq p_dir1 (getpoint "\nDirezione asse Maggiore (Pt1): ")
           p_dir2 (getpoint p_dir1 "\nDirezione asse (Pt2): ")
           f_val (angle p_dir1 p_dir2)
           mode_idx 1))
    ((= m_choice "Rapporto")
     (setq f_val (if (setq f_val (getreal "\nRapporto b/a fisso <0.5>: ")) f_val 0.5)
           mode_idx 2)
     (if (> f_val 1.0) (setq f_val (/ 1.0 f_val)))))
  (list mode_idx f_val))
;;-------------------------------------------------------------------------------------
(defun ELL3P_Universal_Loop (pts mode_idx f_val map_C ents_C map_F ents_F map_V ents_V exact_V_
     map exact_F_map / loop_main opt_gc opt_gf opt_gv opt_t opt_m p5 last_run_ents res_C res_F
     res_V global_min_d best_global nearest_base_pt exact_solutions min_d d active_node_type ex
     act_map last_target_node sol_index)
  (setq loop_main T opt_gc "ON" opt_gf "OFF" opt_gv "OFF" opt_t "Luogo" opt_m "Sostituisci" las
     t_run_ents nil)
  (setq last_target_node nil sol_index 0)

  (Util_Update_Guides_Vis "ON"  ents_C)
  (Util_Update_Guides_Vis "OFF" ents_F)
  (Util_Update_Guides_Vis "OFF" ents_V)

  (while loop_main
    (setvar "OSMODE" 0)
    (setq p5 (getpoint (ELL3P_Helper_UI_Prompt opt_gc opt_gf opt_gv opt_t opt_m)))

    (cond
      ((or (null p5) (= p5 "Uscita")) (setq loop_main nil))

      ((= p5 "Centri")  (setq opt_gc (if (= opt_gc "ON") "OFF" "ON"))
     (Util_Update_Guides_Vis opt_gc ents_C))
      ((= p5 "Fuochi")  (setq opt_gf (if (= opt_gf "ON") "OFF" "ON"))
     (Util_Update_Guides_Vis opt_gf ents_F))
      ((= p5 "Vertici")
       (setq opt_gv (if (= opt_gv "ON") "OFF" "ON"))
```

```
  (if (= opt_gv "OFF") (setq opt_t "Luogo"))
  (Util_Update_Guides_Vis opt_gv ents_V))

 ((= p5 "Target")  (setq opt_t  (if (= opt_t "Luogo") "Nodo" "Luogo")))
 ((= p5 "Modo")    (setq opt_m  (if (= opt_m "Sostituisci") "Aggiungi" "Sostituisci")
last_run_ents nil))

 ((listp p5)
  (if (= opt_t "Luogo")
      (progn
        (setq res_C nil res_F nil res_V nil global_min_d 1e99 best_global nil)

        (if (= opt_gc "ON")
(setq res_C (ELL3P_Refine_Locus p5 pts map_C mode_idx f_val "Centri")))
        (if (= opt_gf "ON")
(setq res_F (ELL3P_Refine_Locus p5 pts map_F mode_idx f_val "Fuochi")))
        (if (= opt_gv "ON")
(setq res_V (ELL3P_Refine_Locus p5 pts map_V mode_idx f_val "Vertici")))

        (if (and res_C (< (car res_C) global_min_d)) (setq global_min_d (car res_C)
best_global (cadr res_C)))
        (if (and res_F (< (car res_F) global_min_d)) (setq global_min_d (car res_F)
best_global (cadr res_F)))
        (if (and res_V (< (car res_V) global_min_d)) (setq global_min_d (car res_V)
best_global (cadr res_V)))

        (if best_global
            (setq last_run_ents (ELL3P_Draw_Engine best_global opt_m last_run_ents))
            (princ "\n[!] Nessun Luogo Geometrico acceso.")))

      (progn
        (setq nearest_base_pt nil min_d 1e99)
        (foreach p pts (setq d (distance p5 p)) (if (< d min_d)
(setq min_d d nearest_base_pt p)))

        (setq active_node_type nil exact_map nil)
        (if (= opt_gv "ON") (setq active_node_type "Vertici" exact_map exact_V_map))
        (if (= opt_gf "ON") (setq active_node_type "Fuochi"  exact_map exact_F_map))

        (if (not active_node_type)
            (princ "\n[!] Attiva i Vertici o i Fuochi per cercare il Nodo Esatto.")
            (progn
              (setq exact_solutions (cdr (assoc nearest_base_pt exact_map)))
              (if exact_solutions
                  (progn
```

```
                        (if (and last_target_node (equal nearest_base_pt last_target_node 1e-4
    ))
                            (progn
                              (setq sol_index (1+ sol_index))
                              (if (>= sol_index (length exact_solutions)) (setq sol_index 0)))
                            (progn
                              (setq last_target_node nearest_base_pt)
                              (setq sol_index 0)))

                        (setq last_geo (nth sol_index exact_solutions)
                              last_run_ents (ELL3P_Draw_Engine last_geo opt_m last_run_ents))
                        (princ (strcat "\n[Target Nodo] Visualizzata soluzione " (itoa (1+ sol
    _index)) " di " (itoa (length exact_solutions)))))
                      (princ "\n[!] Nessuna soluzione analitica convergente per questo Nodo.")
    ))))))
   )
  )

  (if (= opt_gc "OFF") (Util_Del_Ents_List ents_C))
  (if (= opt_gf "OFF") (Util_Del_Ents_List ents_F))
  (if (= opt_gv "OFF") (Util_Del_Ents_List ents_V)))
;;-------------------------------------------------------------------------------------
(defun c:ell3PRD (/ pts old_os old_pd old_ce mode_idx f_val master_res map_C ents_C map_F ents_
    F map_V ents_V exact_V_map exact_F_map t_start t_end elapsed)
  (setq *ELL3P_Col_C* 1
        *ELL3P_Col_F* 3
        *ELL3P_Col_V* 5)

  (mapcar 'set '(pts old_os old_pd old_ce)
     (ELL3P_Get_Data "LABORATORIO UNIVERSALE ELLISSE 3P"))
  (if pts
    (progn
      (mapcar 'set '(mode_idx f_val) (ELL3P_Get_Constraints))

      (setq t_start (* 86400.0 (getvar "DATE")))

      (princ "\nCalcolo Master Locus (Single-Pass) in corso...")
      (setq master_res (ELL3P_Master_Engine pts mode_idx f_val))
      (mapcar 'set '(map_C ents_C map_F ents_F map_V ents_V) master_res)

      (princ "\nPre-calcolo dei Nodi Esatti (Vertici e Fuochi) in corso...")
      (setq exact_V_map (ELL3P_Precalc_Exact_Nodes pts mode_idx f_val "Vertici")
            exact_F_map (ELL3P_Precalc_Exact_Nodes pts mode_idx f_val "Fuochi"))

      (setq t_end (* 86400.0 (getvar "DATE")) elapsed (- t_end t_start))
```

```
      (princ (strcat "\n[TIMER] Motore geometrico e Nodi pre-calcolati in " (rtos elapsed 2 3)
     " secondi."))

      (ELL3P_Universal_Loop pts mode_idx f_val map_C ents_C map_F ents_F map_V ents_V exact_V_m
     ap exact_F_map)))
  (Restore_Conic_App old_os old_pd old_ce))
)
;;-------------------------------------------------------------------------------------------
;;-------------------------------------------------------------------------------------------

;;-------------------------------------------------------------------------------------------
;;-------------------------------------------------------------------------------------------
(progn
;;-------------------------------------------------------------------------------------------
(defun IPE3P_Helper_UI_Prompt (opt_gc opt_gf opt_gv opt_t opt_m opt_e)
  (if (= opt_gv "ON")
      (progn
        (initget "Centri Fuochi Vertici Target Modo Estendi Accorcia Uscita")
        (strcat "\nC:" opt_gc " F:" opt_gf " V:" opt_gv " T:" opt_t " | M:" opt_m " | E:" (rtos
     opt_e 2 1)
     " | Punto o [Centri/Fuochi/Vertici/Target/Modo/Estendi/Accorcia/Uscita] <Uscita>: "))
      (progn
        (initget "Centri Fuochi Vertici Modo Estendi Accorcia Uscita")
        (strcat "\nC:" opt_gc " F:" opt_gf " V:" opt_gv " | M:" opt_m " | E:" (rtos opt_e 2 1)
     " | Punto o [Centri/Fuochi/Vertici/Modo/Estendi/Accorcia/Uscita] <Uscita>: "))))
;;-------------------------------------------------------------------------------------------
(defun IPE3P_Find_Absolute_Nearest_Data (pt_mouse map_data / min_dist best_item best_idx curr_i
     dx curr_dist)
  (setq min_dist 1e99 best_item nil best_idx -1 curr_idx 0)
  (foreach item map_data
    (if (< (setq curr_dist (distance pt_mouse (car item))) min_dist)
        (setq min_dist curr_dist best_item item best_idx curr_idx))
    (setq curr_idx (1+ curr_idx)))
  (list best_item best_idx min_dist))
;;-------------------------------------------------------------------------------------------
(defun IPE3P_Modulo_Pi (ang_val)
  (setq ang_val (rem ang_val pi))
  (if (< ang_val 0.0) (+ ang_val pi) ang_val))
;;-------------------------------------------------------------------------------------------
(defun IPE3P_Helper_Orthogonal_Bisectors (ang_1 ang_2 / bisect_1 bisect_2)
  (setq bisect_1 (IPE3P_Modulo_Pi (/ (+ ang_1 ang_2) 2.0))
        bisect_2 (IPE3P_Modulo_Pi (+ bisect_1 (/ pi 2.0))))
  (list bisect_1 bisect_2))
;;-------------------------------------------------------------------------------------------
(defun IPE3P_Helper_Calc_Angles_Singularity (pt_A pt_B pt_opp ratio_val / pt_proj dist_height d
```

```
     ist_shift vec_dir pt_S1 pt_S2 ang_line1)
  (setq pt_proj      (Math_Proiezione_1Pu_su_2Pu pt_opp pt_A pt_B)
        dist_height  (distance pt_opp pt_proj)
        dist_shift   (* dist_height (/ (abs (- 1.0 (* ratio_val ratio_val)))
     (* 2.0 ratio_val)))
        vec_dir      (Vet_Uni (Vet_Sub pt_B pt_A))
        pt_S1        (Vet_Sum pt_proj (Vet_Sca vec_dir dist_shift))
        pt_S2        (Vet_Sum pt_proj (Vet_Sca vec_dir (- dist_shift)))
        ang_line1    (angle pt_A pt_B))
  (if (equal dist_shift 0.0 1e-6)
              (IPE3P_Helper_Orthogonal_Bisectors ang_line1 (angle pt_proj pt_opp))
      (append (IPE3P_Helper_Orthogonal_Bisectors ang_line1 (angle pt_S1   pt_opp))
              (IPE3P_Helper_Orthogonal_Bisectors ang_line1 (angle pt_S2   pt_opp)))))
;;-------------------------------------------------------------------------------------
(defun IPE3P_Calc_Exact_Singularity_Params (lisPu ratio_val / Pu1 Pu2 Pu3)
  (mapcar 'set '(Pu1 Pu2 Pu3) lisPu)
  (append (IPE3P_Helper_Calc_Angles_Singularity Pu1 Pu2 Pu3 ratio_val)
          (IPE3P_Helper_Calc_Angles_Singularity Pu2 Pu3 Pu1 ratio_val)
          (IPE3P_Helper_Calc_Angles_Singularity Pu3 Pu1 Pu2 ratio_val)))
;;-------------------------------------------------------------------------------------
(defun IPE3P_Helper_Fractal_Band (param_center step_base max_steps base_mult / step_idx band_li
     st offset)
  (setq step_idx 1
       band_list nil)
  (while (<= step_idx max_steps)
         (setq offset (+ (* base_mult step_base) (* step_idx (/ step_base (float max_steps)))))
         (setq band_list (cons (+ param_center offset) band_list)
               band_list (cons (- param_center offset) band_list)
               step_idx  (1+ step_idx)))
  band_list)
;;-------------------------------------------------------------------------------------
(defun IPE3P_Fractal_Around_Param_Safe (param_center step_base)
  (append (IPE3P_Helper_Fractal_Band  param_center step_base 16 0.0)
          (IPE3P_Helper_Fractal_Band  param_center step_base  8 1.0)
          (IPE3P_Helper_Fractal_Band  param_center step_base  4 2.0)
          (IPE3P_Helper_Fractal_Band  param_center step_base  2 3.0)))
;;-------------------------------------------------------------------------------------
(defun IPE3P_Helper_Draw_And_Group_Locus (pt_list col_val layer_name group_name / e_start guide
     _ents)
  (if pt_list
      (progn
        (setq e_start (entlast))
        (setvar "CECOLOR" (itoa col_val))
        (foreach pt pt_list (Draw_Point_R pt layer_name))
        (setq guide_ents (Safe_Collect_Ents e_start))
```

```
        (if guide_ents (Util_Make_Anonymous_Group guide_ents group_name))
        guide_ents)))
;;-------------------------------------------------------------------------------
(defun IPE3P_Master_Engine_Pure (pts_list mode forced_val / pt_1 pt_2 pt_3 pts_rel cos_val sin_
     val pts_loc solver_res center_loc a_val b_val center_wcs center_wcs_rel v_maj_wcs c_foc pt
     _f1 pt_f2 pt_v1 pt_v2 G_bary check_dist list_C map_C list_F map_F list_V map_V start_val e
     nd_val hd_steps step_base curr_idx final_params sing_params clean_params prev_p ang_rad cu
     r_ratio g_data guide_C guide_F guide_V)
  (mapcar 'set '(pt_1 pt_2 pt_3) pts_list)
  (setq G_bary     (Pu_Baricentro2D_LisPu pts_list)
        check_dist (* 100.0 (apply 'max (mapcar '(lambda (p) (distance p G_bary)) pts_list))))

  (setq start_val (if (= mode 1) 0.05 0.0)
        end_val   (if (= mode 1) 19.95 pi)
        hd_steps  3600.0
        step_base (/ (- end_val start_val) hd_steps))

  (setvar "OSMODE" 0)

  (setq final_params nil curr_idx 0)
  (while (<= curr_idx hd_steps)
    (setq final_params (cons (+ start_val (* curr_idx step_base)) final_params)
     curr_idx (1+ curr_idx)))

  (if (= mode 2)
      (progn
        (setq sing_params (IPE3P_Calc_Exact_Singularity_Params pts_list forced_val))
        (foreach sp sing_params
          (setq final_params (append final_params (IPE3P_Fractal_Around_Param_Safe sp step_base
     ))))))

  (setq final_params (vl-sort final_params '<) clean_params nil prev_p -1e99)
  (foreach p final_params
    (if (> (- p prev_p) 1e-9)
        (progn (setq clean_params (cons p clean_params) prev_p p))))
  (setq final_params (reverse clean_params))

  (setq list_C nil map_C nil list_F nil map_F nil list_V nil map_V nil)
  (foreach curr_val final_params
    (if (= mode 1)
        (setq ang_rad forced_val     cur_ratio curr_val)
        (setq ang_rad curr_val       cur_ratio forced_val))

    (setq cos_val (cos ang_rad) sin_val (sin ang_rad)
          pts_rel (mapcar '(lambda (p) (list (- (car p) (car pt_1)) (- (cadr p) (cadr pt_1))
```

```
   0.0)) pts_list)
        pts_loc (Util_Rot_List_2D_CS pts_rel cos_val sin_val)
        solver_res (Math_Solve_Ipe_3P_Ratio pts_loc cur_ratio))

  (if (and solver_res (listp solver_res))
    (progn
      (setq center_loc (car solver_res)
            a_val (cadr solver_res)
            b_val (* a_val cur_ratio)
            center_wcs_rel (Util_Rot_2D_Inv_CS center_loc cos_val sin_val)
            center_wcs (list (+ (car center_wcs_rel) (car pt_1)) (+ (cadr center_wcs_rel)
   (cadr pt_1)) 0.0)
            v_maj_wcs (list cos_val sin_val 0.0)
            c_foc (sqrt (+ (* a_val a_val) (* b_val b_val)))
            g_data (list 'IPERBOLE cur_ratio ang_rad center_wcs v_maj_wcs a_val b_val c_foc))

      (if (< (distance center_wcs G_bary) check_dist)
          (setq list_C (cons center_wcs list_C) map_C (cons (cons center_wcs g_data) map_C)))

      (setq pt_f1 (Vet_Sum center_wcs (Vet_Sca v_maj_wcs c_foc))
            pt_f2 (Vet_Sum center_wcs (Vet_Sca v_maj_wcs (- c_foc))))
      (if (< (distance pt_f1 G_bary) check_dist) (setq list_F (cons pt_f1 list_F)
   map_F (cons (cons pt_f1 g_data) map_F)))
      (if (< (distance pt_f2 G_bary) check_dist) (setq list_F (cons pt_f2 list_F)
   map_F (cons (cons pt_f2 g_data) map_F)))

      (setq pt_v1 (Vet_Sum center_wcs (Vet_Sca v_maj_wcs a_val))
            pt_v2 (Vet_Sum center_wcs (Vet_Sca v_maj_wcs (- a_val))))
      (if (< (distance pt_v1 G_bary) check_dist) (setq list_V (cons pt_v1 list_V)
   map_V (cons (cons pt_v1 g_data) map_V)))
      (if (< (distance pt_v2 G_bary) check_dist) (setq list_V (cons pt_v2 list_V)
   map_V (cons (cons pt_v2 g_data) map_V))))))

(setq list_C (reverse list_C) map_C (reverse map_C)
      list_F (reverse list_F) map_F (reverse map_F)
      list_V (reverse list_V) map_V (reverse map_V))

(setq guide_C (IPE3P_Helper_Draw_And_Group_Locus list_C *IPE3P_Col_C* *LaCo_Luoghi_geometrici
   * "Locus_Centri"))
(setq guide_F (IPE3P_Helper_Draw_And_Group_Locus list_F *IPE3P_Col_F* *LaCo_Luoghi_geometrici
   * "Locus_Fuochi"))
(setq guide_V (IPE3P_Helper_Draw_And_Group_Locus list_V *IPE3P_Col_V* *LaCo_Luoghi_geometrici
   * "Locus_Vertici"))

(setvar "CECOLOR" "BYLAYER")
```

```
  (list map_C guide_C map_F guide_F map_V guide_V))
;;-------------------------------------------------------------------------------------
(defun IPE3P_Eval_Cands (pts_list cur_ratio ang_rad target_type / pt_1 cos_val sin_val pts_rel
     pts_loc solver_res a_val b_val center_wcs_rel center_wcs v_maj_wcs c_foc pt_f1 pt_f2 pt_v1
     pt_v2 g_data cands)
  (setq pt_1 (car pts_list) cos_val (cos ang_rad) sin_val (sin ang_rad))

  (setq pts_rel (mapcar '(lambda (p) (list (- (car p) (car pt_1)) (- (cadr p) (cadr pt_1))
     0.0)) pts_list)
        pts_loc (Util_Rot_List_2D_CS pts_rel cos_val sin_val)
        solver_res (Math_Solve_Ipe_3P_Ratio pts_loc cur_ratio))

  (if (and solver_res (listp solver_res))
      (progn
        (setq a_val (cadr solver_res) b_val (* a_val cur_ratio)
              center_wcs_rel (Util_Rot_2D_Inv_CS (car solver_res) cos_val sin_val)
              center_wcs (list (+ (car center_wcs_rel) (car pt_1)) (+ (cadr center_wcs_rel)
     (cadr pt_1)) 0.0)
              v_maj_wcs (list cos_val sin_val 0.0)
              c_foc (sqrt (+ (* a_val a_val) (* b_val b_val)))
              g_data (list 'IPERBOLE cur_ratio ang_rad center_wcs v_maj_wcs a_val b_val c_foc))

        (cond
          ((= target_type "Centri")  (setq cands (list (cons center_wcs g_data))))
          ((= target_type "Fuochi")
           (setq pt_f1 (Vet_Sum center_wcs (Vet_Sca v_maj_wcs c_foc))
                 pt_f2 (Vet_Sum center_wcs (Vet_Sca v_maj_wcs (- c_foc)))
                 cands (list (cons pt_f1 g_data) (cons pt_f2 g_data))))
          ((= target_type "Vertici")
           (setq pt_v1 (Vet_Sum center_wcs (Vet_Sca v_maj_wcs a_val))
                 pt_v2 (Vet_Sum center_wcs (Vet_Sca v_maj_wcs (- a_val)))
                 cands (list (cons pt_v1 g_data) (cons pt_v2 g_data)))))
        cands)
      nil))
;;-------------------------------------------------------------------------------------
(defun IPE3P_Refine_Locus_Pure (pt_mouse pts_list pure_map mode target_type / king_data best_it
     em best_idx min_d_before min_d_after step_idx max_idx dist_val arm_length step_master delt
     a_span step_micro val_cur cands pt_cand dist_king dist_mouse min_dist_mouse best_res king_
     ratio king_angle)
  (setq king_data (IPE3P_Find_Absolute_Nearest_Data pt_mouse pure_map))
  (setq best_item (car king_data) best_idx (cadr king_data))

  (if best_item
      (progn
        (setq king_ratio (nth 1 (cdr best_item))
```

```
      king_angle (nth 2 (cdr best_item)))

(setq min_d_before 1e99 min_d_after 1e99)
(setq step_idx (max 0 (- best_idx 4)))
(while (< step_idx best_idx)
  (setq dist_val (distance (car best_item) (car (nth step_idx pure_map))))
  (if (< dist_val min_d_before) (setq min_d_before dist_val))
  (setq step_idx (1+ step_idx)))
(setq step_idx (1+ best_idx) max_idx (length pure_map))
(while (and (< step_idx (+ best_idx 5)) (< step_idx max_idx))
  (setq dist_val (distance (car best_item) (car (nth step_idx pure_map))))
  (if (< dist_val min_d_after) (setq min_d_after dist_val))
  (setq step_idx (1+ step_idx)))

(setq arm_length (max (if (= min_d_before 1e99) 0.0 min_d_before)
                      (if (= min_d_after  1e99) 0.0 min_d_after)))
(if (= arm_length 0.0) (setq arm_length 1.0))

(setq step_master (if (= mode 1) (/ 20.0 7200.0) (/ pi 3600.0)))
(setq delta_span (* step_master 2.5))
(setq step_micro (/ (* 2.0 delta_span) 25.0))

(setq val_cur (if (= mode 2) (- king_angle delta_span) (- king_ratio delta_span))
      step_idx 0
      min_dist_mouse 1e99
      best_res nil)

(while (<= step_idx 25)
  (if (= mode 1)
      (setq cands (IPE3P_Eval_Cands pts_list val_cur king_angle target_type))
      (setq cands (IPE3P_Eval_Cands pts_list king_ratio val_cur target_type)))

  (foreach cand_pack cands
    (setq pt_cand (car cand_pack)
          dist_king (distance pt_cand (car best_item)))

    (if (<= dist_king arm_length)
        (progn
          (setq dist_mouse (distance pt_cand pt_mouse))
          (if (< dist_mouse min_dist_mouse)
              (setq min_dist_mouse dist_mouse best_res cand_pack)))))
  (setq val_cur (+ val_cur step_micro) step_idx (1+ step_idx)))

(if best_res
    (list min_dist_mouse best_res)
```

```
            (list (distance pt_mouse (car best_item)) best_item)))
      nil))
;;-------------------------------------------------------------------------------------
(defun IPE3P_Get_Best_Refined_Among_Lists (pt_mouse pts_list lists_of_maps mode target_type / g
     lobal_min_d best_global res)
  (setq global_min_d 1e99 best_global nil)
  (foreach pure_map lists_of_maps
    (if pure_map
      (progn
        (setq res (IPE3P_Refine_Locus_Pure pt_mouse pts_list pure_map mode target_type))
        (if (and res (< (car res) global_min_d))
            (setq global_min_d (car res) best_global (cadr res))))))
  (if best_global (list global_min_d best_global) nil))
;;-------------------------------------------------------------------------------------
(defun IPE3P_Math_Error_Func (pts_list angle_val current_ratio target_pt target_type / cos_val
     sin_val pt_1 pts_rel pts_loc solver_res center_wcs_rel center_wcs a_val b_val c_foc dist_v
     al target_dist)
  (setq pt_1 (car pts_list) cos_val (cos angle_val) sin_val (sin angle_val)
        pts_rel (mapcar '(lambda (p) (list (- (car p) (car pt_1)) (- (cadr p) (cadr pt_1))
     0.0)) pts_list)
        pts_loc (Util_Rot_List_2D_CS pts_rel cos_val sin_val)
        solver_res (Math_Solve_Ipe_3P_Ratio pts_loc current_ratio))

  (if (and solver_res (listp solver_res))
      (progn
        (setq a_val (cadr solver_res) b_val (* a_val current_ratio)
              center_wcs_rel (Util_Rot_2D_Inv_CS (car solver_res) cos_val sin_val)
              center_wcs (list (+ (car center_wcs_rel) (car pt_1)) (+ (cadr center_wcs_rel)
     (cadr pt_1)) 0.0)
              dist_val (distance center_wcs target_pt)
              c_foc (sqrt (+ (* a_val a_val) (* b_val b_val))))

        (cond ((= target_type "V_maj") (setq target_dist a_val))
              ((= target_type "Fuoco") (setq target_dist c_foc))
              ((= target_type "Centro")(setq target_dist 0.0)))
        (abs (- dist_val target_dist)))
      nil))
;;-------------------------------------------------------------------------------------
(defun IPE3P_Solve_Exact_Node_Algebraic (p_target pts_list mode fixed_val active_node_type / t_
     type solutions limit step_val step_idx param_val err_list e_val prev_e curr_e next_e local
     _minima min_param best_err p_fine fine_step j_idx e_fine cands_pack)
  (setq t_type (cond ((= active_node_type "Vertici") "V_maj")
                     ((= active_node_type "Fuochi")  "Fuoco")
                     ((= active_node_type "Centro")  "Centro"))
        solutions nil)
```

```
(if (= mode 1)
    (setq limit 20.0 step_val (/ 20.0 720.0))
    (setq limit pi   step_val (/ pi 720.0)))

(setq step_idx 0 err_list nil)
(while (<= step_idx 720)
  (setq param_val (* step_idx step_val))
  (setq e_val (if (= mode 1)
                 (IPE3P_Math_Error_Func pts_list fixed_val param_val p_target t_type)
                 (IPE3P_Math_Error_Func pts_list param_val fixed_val p_target t_type)))
  (setq err_list (append err_list (list (cons param_val e_val))))
  (setq step_idx (1+ step_idx)))

(setq step_idx 1 local_minima nil)
(while (< step_idx 720)
  (setq prev_e (cdr (nth (1- step_idx) err_list))
        curr_e (cdr (nth step_idx err_list))
        next_e (cdr (nth (1+ step_idx) err_list)))

  (if (and prev_e curr_e next_e)
      (if (and (< curr_e prev_e) (< curr_e next_e) (< curr_e 50.0))
          (setq local_minima (cons (car (nth step_idx err_list)) local_minima))))
  (setq step_idx (1+ step_idx)))

(foreach min_param local_minima
  (setq fine_step (/ (* step_val 2.0) 1000.0)
        j_idx 0
        p_fine (- min_param step_val)
        best_err 1e99
        best_param nil)

  (while (<= j_idx 1000)
    (setq e_fine (if (= mode 1)
                    (IPE3P_Math_Error_Func pts_list fixed_val p_fine p_target t_type)
                    (IPE3P_Math_Error_Func pts_list p_fine fixed_val p_target t_type)))
    (if (and e_fine (< e_fine best_err))
        (setq best_err e_fine best_param p_fine))
    (setq p_fine (+ p_fine fine_step) j_idx (1+ j_idx)))

  (if (and best_param (< best_err 1e-4))
      (progn
        (setq cands_pack (if (= mode 1)
                             (IPE3P_Eval_Cands pts_list best_param fixed_val active_node_type
   )
```

```
                              (IPE3P_Eval_Cands pts_list fixed_val best_param active_node_type
    )))
         (foreach cand cands_pack
           (if (< (distance (car cand) p_target) 1e-3)
               (setq solutions (cons cand solutions)))))))
  solutions)
;;-------------------------------------------------------------------------------------------
(defun IPE3P_Precalc_Exact_Nodes (pts_list mode fixed_val node_type / map_list sols)
  (setq map_list nil)
  (foreach pt pts_list
    (setq sols (IPE3P_Solve_Exact_Node_Algebraic pt pts_list mode fixed_val node_type))
    (setq map_list (cons (cons pt sols) map_list)))
  map_list)
;;-------------------------------------------------------------------------------------------
(defun IPE3P_Draw_Engine (pts_tri geo_packet opt_m opt_e last_ents / center_pt vec_dir a_val b_
    val c_foc pt_v1 pt_f1 e_marker max_proj x_limit dummy_pt)
  (if (and (= opt_m "Sostituisci") last_ents) (foreach e last_ents (if (entget e) (entdel e))))
  (setq e_marker  (entlast)
        center_pt (nth 4 geo_packet)
        vec_dir   (nth 5 geo_packet)
        a_val     (nth 6 geo_packet)
        b_val     (nth 7 geo_packet)
        c_foc     (nth 8 geo_packet)
        pt_v1     (Vet_Sum center_pt (Vet_Sca vec_dir a_val))
        pt_f1     (Vet_Sum center_pt (Vet_Sca vec_dir c_foc)))

  (setq max_proj (get-max-abs-projection-on-axis pts_tri center_pt vec_dir))
  (setq x_limit (* (max max_proj (* a_val 1.2)) opt_e))
  (setq dummy_pt (Vet_Sum center_pt (Vet_Sca vec_dir x_limit)))

  (Draw_ConIpe_CeVeFu_Compl_Free center_pt pt_v1 pt_f1 (append pts_tri (list dummy_pt)))
  (Safe_Collect_Ents e_marker))
;;-------------------------------------------------------------------------------------------
(defun IPE3P_Get_Data (app_title / data pts_list)
  (setq data (Init_Conic_App app_title 3 "\nSeleziona 3 punti: " 0))
  (setq pts_list (car data))
  (if pts_list
    (progn  (Draw_Triangolo_Lis3Pu        pts_list *LaCo_Poligoni*)
            (Draw_Triangolo_Ortico_Lis3Pu pts_list *LaCo_Costruzioni*)))
  data)
;;-------------------------------------------------------------------------------------------
(defun IPE3P_Get_Constraints (/ mode_choice p_dir1 p_dir2 force_val mode_idx)
  (initget "Rapporto Direzione")
  (setq mode_choice (if (setq mode_choice (getkword "\nVincolo: [Rapporto/Direzione] <Rapporto>
     : ")) mode_choice "Rapporto"))
```

```
  (cond
    ((= mode_choice "Direzione")
     (setq p_dir1 (getpoint "\nDirezione asse Reale (Pt1): ")
           p_dir2 (getpoint p_dir1 "\nDirezione asse (Pt2): ")
           force_val (angle p_dir1 p_dir2)
           mode_idx 1))
    ((= mode_choice "Rapporto")
     (setq force_val (if (setq force_val (getreal "\nRapporto b/a <1.0>: ")) force_val 1.0)
           mode_idx 2)))
  (list mode_idx force_val))
;;-------------------------------------------------------------------------------------
(defun IPE3P_Universal_Loop (pts_list mode_idx f_val m_C_base e_C_base m_C_inv e_C_inv m_F_base
     e_F_base m_F_inv e_F_inv m_V_base e_V_base m_V_inv e_V_inv exact_V_map exact_F_map / loop_
     main opt_gc opt_gf opt_gv opt_t opt_m opt_e pt_click last_run_ents last_geo res_C res_F re
     s_V global_min_d best_global nearest_base_pt min_d curr_dist active_node_type exact_map ex
     act_solutions last_target_node sol_index)
  (setq loop_main T opt_gc "ON" opt_gf "OFF" opt_gv "OFF" opt_t "Luogo" opt_m "Sostituisci" opt
     _e 1.0 last_run_ents nil last_geo nil)
  (setq last_target_node nil sol_index 0)

  (Util_Update_Guides_Vis "ON"  (append e_C_base e_C_inv))
  (Util_Update_Guides_Vis "OFF" (append e_F_base e_F_inv))
  (Util_Update_Guides_Vis "OFF" (append e_V_base e_V_inv))

  (while loop_main
    (setvar "OSMODE" 0)
    (setq pt_click (getpoint (IPE3P_Helper_UI_Prompt opt_gc opt_gf opt_gv opt_t opt_m opt_e)))

    (cond
      ((or (null pt_click) (= pt_click "Uscita")) (setq loop_main nil))

      ((= pt_click "Centri")  (setq opt_gc (if (= opt_gc "ON") "OFF" "ON"))
     (Util_Update_Guides_Vis opt_gc (append e_C_base e_C_inv)))
      ((= pt_click "Fuochi")  (setq opt_gf (if (= opt_gf "ON") "OFF" "ON"))
     (Util_Update_Guides_Vis opt_gf (append e_F_base e_F_inv)))
      ((= pt_click "Vertici")
       (setq opt_gv (if (= opt_gv "ON") "OFF" "ON"))
       (if (= opt_gv "OFF") (setq opt_t "Luogo"))
       (Util_Update_Guides_Vis opt_gv (append e_V_base e_V_inv)))

      ((= pt_click "Target")  (setq opt_t  (if (= opt_t "Luogo") "Nodo" "Luogo")))
      ((= pt_click "Modo")    (setq opt_m  (if (= opt_m "Sostituisci")
     "Aggiungi" "Sostituisci") last_run_ents nil))

      ((= pt_click "Estendi")
```

```
  (setq opt_e (+ opt_e 0.5))
  (if (and last_geo (= opt_m "Sostituisci"))
      (setq last_run_ents (IPE3P_Draw_Engine pts_list last_geo opt_m opt_e last_run_ents))
))

 ((= pt_click "Accorcia")
  (setq opt_e (max 0.5 (- opt_e 0.5)))
  (if (and last_geo (= opt_m "Sostituisci"))
      (setq last_run_ents (IPE3P_Draw_Engine pts_list last_geo opt_m opt_e last_run_ents))
))

 ((listp pt_click)
  (if (= opt_t "Luogo")
      (progn
        (setq res_C nil res_F nil res_V nil global_min_d 1e99 best_global nil)

        (if (= opt_gc "ON")
(setq res_C (IPE3P_Get_Best_Refined_Among_Lists pt_click pts_list (list m_C_base m_C_inv)
mode_idx "Centri")))
        (if (= opt_gf "ON")
(setq res_F (IPE3P_Get_Best_Refined_Among_Lists pt_click pts_list (list m_F_base m_F_inv)
mode_idx "Fuochi")))
        (if (= opt_gv "ON")
(setq res_V (IPE3P_Get_Best_Refined_Among_Lists pt_click pts_list (list m_V_base m_V_inv)
mode_idx "Vertici")))

        (if (and res_C (< (car res_C) global_min_d)) (setq global_min_d (car res_C)
best_global (cadr res_C)))
        (if (and res_F (< (car res_F) global_min_d)) (setq global_min_d (car res_F)
best_global (cadr res_F)))
        (if (and res_V (< (car res_V) global_min_d)) (setq global_min_d (car res_V)
best_global (cadr res_V)))

        (if best_global
            (setq last_geo best_global
                  last_run_ents (IPE3P_Draw_Engine pts_list best_global opt_m opt_e last_r
un_ents))
            (princ "\n[!] Nessun Luogo Geometrico acceso.")))

      (progn
        (setq nearest_base_pt nil min_d 1e99)
        (foreach p pts_list (setq curr_dist (distance pt_click p)) (if (< curr_dist min_d)
(setq min_d curr_dist nearest_base_pt p)))

        (setq active_node_type nil exact_map nil)
```

```
            (if (= opt_gv "ON") (setq active_node_type "Vertici" exact_map exact_V_map))
            (if (= opt_gf "ON") (setq active_node_type "Fuochi"  exact_map exact_F_map))

            (if (not active_node_type)
                (princ "\n[!] Attiva i Vertici o i Fuochi per cercare il Nodo Esatto.")
                (progn
                  (setq exact_solutions (cdr (assoc nearest_base_pt exact_map)))
                  (if exact_solutions
                      (progn
                        (if (and last_target_node (equal nearest_base_pt last_target_node 1e-4
    ))
                            (progn
                              (setq sol_index (1+ sol_index))
                              (if (>= sol_index (length exact_solutions)) (setq sol_index 0)))
                            (progn
                              (setq last_target_node nearest_base_pt)
                              (setq sol_index 0)))

                        (setq last_geo (nth sol_index exact_solutions)
                              last_run_ents (IPE3P_Draw_Engine pts_list last_geo opt_m opt_e l
    ast_run_ents))
                        (princ (strcat "\n[Target Nodo] Visualizzata soluzione " (itoa (1+ sol
    _index)) " di " (itoa (length exact_solutions)))))
                      (princ "\n[!] Nessuna soluzione convergente per questo Nodo."))))))
    )
  )

  (if (= opt_gc "OFF") (Util_Del_Ents_List (append e_C_base e_C_inv)))
  (if (= opt_gf "OFF") (Util_Del_Ents_List (append e_F_base e_F_inv)))
  (if (= opt_gv "OFF") (Util_Del_Ents_List (append e_V_base e_V_inv)))
)
;;-------------------------------------------------------------------------------------
(defun c:ipe3PRD (/ *IPE3P_Col_C* *IPE3P_Col_F* *IPE3P_Col_V* pts_list old_os old_pd old_ce mod
    e_idx f_val res_base res_inv exact_V_map exact_F_map t_start t_end elapsed)
  (setq *IPE3P_Col_C* 1
        *IPE3P_Col_F* 3
        *IPE3P_Col_V* 5)

  (mapcar 'set '(pts_list old_os old_pd old_ce)
     (IPE3P_Get_Data "LABORATORIO UNIVERSALE IPERBOLE 3P"))
  (if pts_list
    (progn
      (mapcar 'set '(mode_idx f_val) (IPE3P_Get_Constraints))

      (setq t_start (* 86400.0 (getvar "DATE")))
```

```
      (princ "\nCalcolo Famiglia Base (Single-Pass & Deterministic Pre-Mapping) in corso...")
      (setq res_base (IPE3P_Master_Engine_Pure pts_list mode_idx f_val))

      (setq res_inv (list nil nil nil nil nil nil))
      (if (and (= mode_idx 2) (not (equal f_val 1.0 1e-6)))
          (progn
            (princ "\nCalcolo Famiglia Inversa (Single-Pass & Deterministic Pre-Mapping)
    in corso...")
            (setq res_inv (IPE3P_Master_Engine_Pure pts_list mode_idx (/ 1.0 f_val)))))

      (princ "\nPre-calcolo dei Nodi Esatti (Vertici e Fuochi) in corso...")
      (setq exact_V_map (IPE3P_Precalc_Exact_Nodes pts_list mode_idx f_val "Vertici")
            exact_F_map (IPE3P_Precalc_Exact_Nodes pts_list mode_idx f_val "Fuochi"))

      (setq t_end (* 86400.0 (getvar "DATE")) elapsed (- t_end t_start))
      (princ (strcat "\n[TIMER] Motore geometrico e Nodi pre-calcolati in " (rtos elapsed 2 3)
    " secondi."))

      (IPE3P_Universal_Loop pts_list mode_idx f_val
                            (nth 0 res_base) (nth 1 res_base) (nth 0 res_inv) (nth 1 res_inv)
                            (nth 2 res_base) (nth 3 res_base) (nth 2 res_inv) (nth 3 res_inv)
                            (nth 4 res_base) (nth 5 res_base) (nth 4 res_inv) (nth 5 res_inv)
                            exact_V_map exact_F_map)))
  (Restore_Conic_App old_os old_pd old_ce) (princ)
)
)
;;-----------------------------------------------------------------------------------------
;;-----------------------------------------------------------------------------------------

;;-----------------------------------------------------------------------------------------
;;-----------------------------------------------------------------------------------------
(progn
;;-----------------------------------------------------------------------------------------
(defun PAR3P_Calc_Par_V_2P (pt_v pt_1 pt_2 / x1 y1 x2 y2 step_val idx roots th1 th2 f1 f2 th_mi
    d f_mid best_root best_a cos_th sin_th u1_best v1_best v2_best focal_dist F_wcs)
  (setq x1 (- (car pt_1) (car pt_v)) y1 (- (cadr pt_1) (cadr pt_v))
        x2 (- (car pt_2) (car pt_v)) y2 (- (cadr pt_2) (cadr pt_v)))
  (defun F_func (th / c s u1 v1 u2 v2)
    (setq c (cos th) s (sin th)
          u1 (+ (* x1 c) (* y1 s)) v1 (+ (* -1.0 x1 s) (* y1 c))
          u2 (+ (* x2 c) (* y2 s)) v2 (+ (* -1.0 x2 s) (* y2 c)))
    (- (* v1 u2 u2) (* v2 u1 u1)))

  (setq step_val (/ pi 180.0) idx 0 roots nil)
```

```
  (while (< idx 180)
    (setq th1 (* idx step_val) th2 (* (1+ idx) step_val) f1 (F_func th1) f2 (F_func th2))
    (if (<= (* f1 f2) 0.0)
        (progn
          (repeat 50
            (setq th_mid (/ (+ th1 th2) 2.0) f_mid (F_func th_mid))
            (if (<= (* f1 f_mid) 0.0) (setq th2 th_mid f2 f_mid) (setq th1 th_mid f1 f_mid)))
          (setq roots (cons th_mid roots))))
    (setq idx (1+ idx)))

  (setq best_root nil best_a nil)
  (foreach th roots
    (setq cos_th (cos th) sin_th (sin th)
          u1_best (+ (* x1 cos_th) (* y1 sin_th))
          v1_best (+ (* -1.0 x1 sin_th) (* y1 cos_th))
          v2_best (+ (* -1.0 x2 sin_th) (* y2 cos_th)))
    (if (and (> (* v1_best v2_best) 1e-9) (> (abs u1_best) 1e-9))
        (setq best_root th best_a (/ v1_best (* u1_best u1_best)))))

  (if (and best_root best_a)
      (progn
        (setq focal_dist (/ 1.0 (* 4.0 best_a)))
        (setq F_wcs (list (- (car pt_v) (* focal_dist (sin best_root))) (+ (cadr pt_v)
     (* focal_dist (cos best_root))) 0.0))
        (list pt_v F_wcs))
      nil))
;;-------------------------------------------------------------------------------------------
(defun PAR3P_Precalc_Exact_Nodes (pts / map_list p1 p2 p3)
  (setq p1 (nth 0 pts) p2 (nth 1 pts) p3 (nth 2 pts))
  (list (cons p1 (PAR3P_Calc_Par_V_2P p1 p2 p3))
        (cons p2 (PAR3P_Calc_Par_V_2P p2 p1 p3))
        (cons p3 (PAR3P_Calc_Par_V_2P p3 p1 p2))))
;;-------------------------------------------------------------------------------------------
(defun PAR3P_Master_Par_3P (pts dist_limit / raw_data branches_V branches_F map_V map_F ename_V
     ename_F old_ce e_start guide_V guide_F)
  (setq map_V nil map_F nil old_ce (getvar "CECOLOR"))
  (setq raw_data (Geom_Scan_3P_Parabolas_Adaptive pts 180.0 1.5))
  (if (and raw_data (listp raw_data))
    (progn
      (setq e_start (entlast)) (setvar "CECOLOR" (itoa *PAR3P_Col_V*))
      (setq branches_V (Util_Process_Branches_Data raw_data dist_limit))
      (foreach b branches_V (if (> (length b) 3)
        (if (setq ename_V (Draw_Curve_Segment_Spline (mapcar 'car b) *LaCo_Luoghi_geometrici*))
            (setq map_V (cons (cons ename_V b) map_V)))))
      (setq guide_V (Safe_Collect_Ents e_start))
```

```
    (if guide_V (Util_Make_Anonymous_Group guide_V "Locus_Vertici"))
     (setq e_start (entlast)) (setvar "CECOLOR" (itoa *PAR3P_Col_F*))
     (setq branches_F (Util_Process_Branches_Data_SmartStitch raw_data dist_limit))
     (foreach b branches_F (if (> (length b) 3)
       (if (setq ename_F (Draw_Curve_Segment_Spline (mapcar 'cadr b)
    *LaCo_Luoghi_geometrici*))
             (setq map_F (cons (cons ename_F b) map_F)))))
     (setq guide_F (Safe_Collect_Ents e_start))
    (if guide_F (Util_Make_Anonymous_Group guide_F "Locus_Fuochi"))))
  (setvar "CECOLOR" old_ce)
  (list guide_V map_V guide_F map_F))
;;-----------------------------------------------------------------------------------------
(defun PAR3P_Helper_UI_Prompt (opt_gf opt_gv opt_t opt_m)
  (initget "Fuochi Vertici Target Modo Uscita")
  (strcat "\nF:" opt_gf " V:" opt_gv " T:" opt_t " | M:" opt_m " | Punto o [Fuochi/Vertici/Targ
     et/Modo/Uscita] <Uscita>: "))
;;-----------------------------------------------------------------------------------------
(defun PAR3P_Geom_Get_Closest_Guide_Target (pt_in guide_list min_d best_c pt_proj / cur_p d)
  (foreach c guide_list
    (setq cur_p (vlax-curve-getClosestPointTo (vlax-ename->vla-object c) pt_in)
          d     (distance pt_in cur_p))
    (if (< d min_d)
      (setq min_d d
            best_c c
            pt_proj cur_p)))
  (list min_d best_c pt_proj))
;;-----------------------------------------------------------------------------------------
(defun PAR3P_Universal_Loop (pts guide_V map_V guide_F map_F tri_scale exact_map / loop opt_gv
     opt_gf opt_t opt_m pt_in last_ents e_mark best_c min_d cur_p d pt_proj branch ref_data np
     sol)
  (setq loop      T
        opt_gv    "ON"
        opt_gf    "OFF"
        opt_t     "Luogo"
        opt_m     "Sostituisci"
        last_ents nil)
  (Util_Update_Guides_Vis "ON"  guide_V)
  (Util_Update_Guides_Vis "OFF" guide_F)
  (while loop
         (setvar "OSMODE" 512)
         (setq pt_in (getpoint (PAR3P_Helper_UI_Prompt opt_gf opt_gv opt_t opt_m)))
         (setvar "OSMODE" 0)
         (cond ((null pt_in)         (setq loop nil))
               ((= pt_in "Uscita")   (setq loop nil))
               ((= pt_in "Fuochi")   (setq opt_gf (if (= opt_gf "ON") "OFF" "ON"))
```

```
                                (Util_Update_Guides_Vis opt_gf guide_F))
          ((= pt_in "Vertici")  (setq opt_gv (if (= opt_gv "ON") "OFF" "ON"))
                                (Util_Update_Guides_Vis opt_gv guide_V))
          ((= pt_in "Target")   (setq opt_t (if (= opt_t "Luogo") "Nodo" "Luogo")))
          ((= pt_in "Modo")     (setq opt_m (if (= opt_m "Sostituisci")
"Aggiungi" "Sostituisci") last_ents nil))
          ((listp pt_in)        (if (and (= opt_m "Sostituisci") last_ents)
                                    (foreach e last_ents (if (entget e) (entdel e))))
                                (setq e_mark (entlast))
                                (if (= opt_t "Nodo")
                                    (progn (setq np  (car (vl-sort pts '(lambda (a b)
(< (distance pt_in a) (distance pt_in b))))))
                                           (setq sol (cdr (assoc np exact_map)))
                                           (if sol
                                               (progn (Draw_Point_R np *LaCo_Costruzioni*)
                                                      (Draw_ConPar_Data_3P pts sol)
                                                      (setq last_ents (Safe_Collect_Ents e
_mark)))
                                               (progn (princ "\n[ERR] Nodo non risolvibile
."))))
                                    (progn (setq best_c nil min_d 1e99 pt_proj nil)
                                           (if (= opt_gv "ON")
(mapcar 'set '(min_d best_c pt_proj)
(PAR3P_Geom_Get_Closest_Guide_Target pt_in guide_V min_d best_c pt_proj)))
                                           (if (= opt_gf "ON")
(mapcar 'set '(min_d best_c pt_proj)
(PAR3P_Geom_Get_Closest_Guide_Target pt_in guide_F min_d best_c pt_proj)))
                                           (if (and best_c (< min_d (/ tri_scale 3.0)))
                                               (progn (Draw_Point_R pt_proj *LaCo_Costruzi
oni*)
                                                      (setq branch   (if (member best_c gu
ide_V) (cdr (assoc best_c map_V)) (cdr (assoc best_c map_F))))
                                                      (setq ref_data (if (member best_c gu
ide_V)
                                                                         (Geom_Refine_Par_
Topological        pt_proj (vlax-ename->vla-object best_c) branch pts)
                                                                         (Geom_Refine_Par_
Topological_Fuochi pt_proj (vlax-ename->vla-object best_c) branch pts)))
                                                      (if ref_data
                                                          (progn (Draw_ConPar_Data_3P pts
ref_data)
                                                                 (setq last_ents (Safe_Col
lect_Ents e_mark)))
                                                          (progn (princ "\n[ERR] Affinamen
to fallito."))))
```

```
                                                    (progn (princ "\n[WARN] Clicca vicino alle
     guide!")))))))))
  (if (= opt_gv "OFF") (Util_Del_Ents_List guide_V))
  (if (= opt_gf "OFF") (Util_Del_Ents_List guide_F)))
;;-------------------------------------------------------------------------------------------
(defun PAR3P_Master_Reactive_Universal_3P (titolo / pts old_os old_pd old_ce loop opt_gv opt_gf
     pt_in pt_old pt_new tri dist_lim master_res gV gF old_ce2)
à (mapcar 'set '(pts old_os old_pd old_ce)
     (Init_Conic_App titolo 3 "\nSeleziona 3 punti base: " 0))
à (if (and pts (= (length pts) 3))
à à (progn (setq opt_gv "ON" opt_gf "OFF" loop T)
à à à (defun update_locus (res / m_res) (setq tri (Geom_MinMax_Dist_LisPu pts nil 2)
     dist_lim (* tri 5.0) m_res (PAR3P_Master_Par_3P pts dist_lim) gV (car m_res)
     gF (nth 2 m_res)) (Util_Update_Guides_Vis opt_gv gV) (Util_Update_Guides_Vis opt_gf gF))
à à à (update_locus 4.0)
à à à (while loop
à à à à (setvar "OSMODE" 8) (initget "Fuochi Vertici Uscita")à
à à à à (setq pt_in (getpoint (strcat "\nF:" opt_gf " V:" opt_gv " | Muovi punto o [Fuochi/Vert
     ici/Uscita] <Uscita per HD>: "))) (setvar "OSMODE" 0)
à à à à (condà
à à à à à ((or (null pt_in) (= pt_in "Uscita")) (setq loop nil))
à à à à à ((= pt_in "Vertici") (setq opt_gv (if (= opt_gv "ON") "OFF" "ON"))
     (Util_Update_Guides_Vis opt_gv gV))
à à à à à ((= pt_in "Fuochi") (setq opt_gf (if (= opt_gf "ON") "OFF" "ON"))
     (Util_Update_Guides_Vis opt_gf gF))
à à à à à ((listp pt_in) (setq pt_old (car (vl-sort pts '(lambda (a b) (< (distance pt_in a)
     (distance pt_in b))))))
à à à à à  (if (setq pt_new (getpoint pt_old "\nPosizione: "))
à à à à à à à  (progn (Util_Del_Ents_At_Pt pt_old *LaCo_Punti_scelti*)
     (Draw_Point_R pt_new *LaCo_Punti_scelti*)
à à à à à à à à  (setq pts (mapcar '(lambda (p) (if (< (distance p pt_old) 1e-6) pt_new p))
     pts))
à à à à à à à à  (Util_Del_Ents_List (append gV gF)) (update_locus 4.0))))))
à à à (princ "\n[SYSTEM] Ricalcolo HD...") (Util_Del_Ents_List (append gV gF))
     (update_locus 1.0)))
à (Restore_Conic_App old_os old_pd old_ce) (princ))
;;-------------------------------------------------------------------------------------------
(defun c:par3P (/ pts old_os old_pd old_ce tri dist_lim master_res exact_map *PAR3P_Col_V* *PAR
     3P_Col_F* t_s t_e)
  (setq *PAR3P_Col_V* 5 *PAR3P_Col_F* 3)
  (mapcar 'set '(pts old_os old_pd old_ce)
     (Init_Conic_App "LABORATORIO PARABOLA 3P" 3 "\nSeleziona 3 punti: " 32))
  (if pts
    (progn (Draw_Triangolo_Lis3Pu pts *LaCo_Poligoni*)
      (setq tri (Geom_MinMax_Dist_LisPu pts nil 2) dist_lim (* tri 5.0)
```

```
      t_s (* 86400.0 (getvar "DATE")))
       (princ "\nCalcolo Master e Nodi...")
       (setq master_res (PAR3P_Master_Par_3P pts dist_lim)
      exact_map (PAR3P_Precalc_Exact_Nodes pts))
       (princ (strcat "\n[TIMER] Concluso in " (rtos (- (* 86400.0 (getvar "DATE")) t_s) 2 3)
      "s."))
       (PAR3P_Universal_Loop pts (car master_res) (cadr master_res) (nth 2 master_res)
      (nth 3 master_res) tri exact_map)))
  (Restore_Conic_App old_os old_pd old_ce) (princ))
;;-------------------------------------------------------------------------------------------
(defun c:par3PRea (/ *PAR3P_Col_V* *PAR3P_Col_F*)
  (setq *PAR3P_Col_V* 5 *PAR3P_Col_F* 3)
  (PAR3P_Master_Reactive_Universal_3P "PARABOLA 3 PUNTI (EDITING REATTIVO)"))
)
;;-------------------------------------------------------------------------------------------
;;-------------------------------------------------------------------------------------------

;;-------------------------------------------------------------------------------------------
;;-------------------------------------------------------------------------------------------
(progn
;;-------------------------------------------------------------------------------------------
(defun Geom_3Pts_Core_Proprio (pts center_pt / p1 p2 p3 result sym_pts)
  (mapcar 'set '(p1 p2 p3) pts)
  (Draw_Point_R center_pt *LaCo_Punti_scelti*)
  (princ "\n[SYSTEM] Analisi Algebrica (Matrice 3x3)...")
  (if (setq result (Draw_ConCen_3P_Compl center_pt p1 p2 p3   pts))
      (progn
        (setq sym_pts (mapcar '(lambda (p) (Geom_1Pu_Simm_PuC p center_pt)) pts))
        (foreach p sym_pts (Draw_Point_R p *LaCo_Costruzioni*))
        (princ " -> OK. Soluzione Algebrica trovata.")
        T)
      (progn (princ "\n[ERRORE] Configurazione geometrica impossibile.") nil)))
;;-------------------------------------------------------------------------------------------
(defun Geom_3Pts_Core_Improprio (pts Pd1 Pd2 / best_set geom_data Ve Fu U_ax)
  (Draw_Lis_Point_R (list Pd1 Pd2) *LaCo_Punti_scelti*)
  (Draw_Line_2Pu_R Pd1 Pd2 *LaCo_Costruzioni*)
  (princ "\n[SYSTEM] Calcola Parabola analitica...")
  (setq best_set (append pts (Calcola_Parabola pts (list Pd1 Pd2))))
  (if (and best_set (= (length best_set) 5))
      (progn
        (if (setq geom_data (Solve_Par_Geometry best_set Pd1 Pd2))
            (progn
              (mapcar 'set '(Ve Fu) geom_data)
              (setq U_ax (Vet_Uni (Vet_Sub Fu Ve)))
              (foreach P pts
```

```
                (Draw_Point_R (Geom_1Pu_Simm_PuC P (Vet_Sum Ve (Vet_Sca U_ax (Vet_Dot (Vet_Sub
     P Ve) U_ax)))) *LaCo_Costruzioni*))))
        (Draw_Con5P_Nor_Compl best_set *LaCo_Tracciati* *LaCo_Costruzioni* nil)
        T)
      (progn (princ "\n[ERRORE] Configurazione punti non valida.") nil)))
;;-----------------------------------------------------------------------------------------
(defun Geom_3Pts_Update_Guides (pts opt_g opt_c old_guides / P1 P2 P3 m12 m23 m31 d_max ext1 ex
     t2 ext3 new_guides)
  (if old_guides (foreach e old_guides (if (entget e) (entdel e))))
  (setq new_guides nil)
  (if (and (= opt_g "ON") (= opt_c "Proprio"))
    (progn
      (mapcar 'set '(P1 P2 P3) pts)
      (setq m12 (Math_PuntoMedio_2Pu P1 P2)
            m23 (Math_PuntoMedio_2Pu P2 P3)
            m31 (Math_PuntoMedio_2Pu P3 P1)
            d_max (* (max (distance P1 P2) (distance P2 P3) (distance P3 P1)) 20.0)
            ext1 (Geom_2Pu_Simm_PuC m12 (Vet_Uni (Vet_Sub m31 m12)) d_max)
            ext2 (Geom_2Pu_Simm_PuC m12 (Vet_Uni (Vet_Sub m23 m12)) d_max)
            ext3 (Geom_2Pu_Simm_PuC m23 (Vet_Uni (Vet_Sub m31 m23)) d_max))
      (setq new_guides (list (Draw_Line_2Pu_R (car ext1) (cadr ext1) *LaCo_Luoghi_geometrici*)
                             (Draw_Line_2Pu_R (car ext2) (cadr ext2) *LaCo_Luoghi_geometrici*)
                             (Draw_Line_2Pu_R (car ext3) (cadr ext3)
     *LaCo_Luoghi_geometrici*)))
      (princ "\n[INFO] Guide mediali aggiornate.")))
  new_guides)
;;-----------------------------------------------------------------------------------------
(defun Geom_3Pts_Master_Engine (pts old_os / loop_main opt_g opt_m opt_c guide_ents last_run_en
     ts pt_input Pd2 e_marker)
  (setq loop_main    T
        opt_g        "ON"
        opt_m        "Sostituisci"
        opt_c        "Proprio"
        guide_ents   nil
        last_run_ents nil)

  (setq guide_ents (Geom_3Pts_Update_Guides pts opt_g opt_c guide_ents))

  (while loop_main
    (setvar "OSMODE" (if old_os old_os 0))
    (initget "Centro Guide Modo Fine")
    (setq pt_input (getpoint (if (= opt_c "Proprio")
     (strcat "\nC:Proprio | G:"   opt_g " | M:" opt_m " | Centro o [Centro/Guide/Modo/Fine] <Fi
     ne>: ")
                                                    (strcat "\nC:Improprio | M:" opt_m " | 1à
```

```
     Punto Direzione asse o [Centro/Modo/Fine] <Fine>: "))))
    (setvar "OSMODE" 0)
    (cond  ((null pt_input)          (setq loop_main nil))
           ((= pt_input "Fine")      (setq loop_main nil))
           ((= pt_input "Centro")    (setq opt_c (if (= opt_c "Proprio")
     "Improprio" "Proprio"))
                                     (setq guide_ents (Geom_3Pts_Update_Guides pts opt_g opt_c
     guide_ents)))
           ((= pt_input "Guide")     (setq opt_g (if (= opt_g "ON") "OFF" "ON"))
                                     (setq guide_ents (Geom_3Pts_Update_Guides pts opt_g opt_c
     guide_ents)))
           ((= pt_input "Modo")      (setq opt_m (if (= opt_m "Sostituisci")
     "Aggiungi" "Sostituisci"))
                                     (if (= opt_m "Aggiungi") (setq last_run_ents nil)))
           ((listp pt_input)         (if (= opt_c "Proprio")
                                         (progn (if (and (= opt_m "Sostituisci") last_run_ents)
                                                    (foreach e last_run_ents (if (entget e)
     (entdel e))))
                                                (setq e_marker (entlast))
                                                (Geom_3Pts_Core_Proprio pts pt_input)
                                                (if (= opt_m "Sostituisci")
                                                    (setq last_run_ents (Util_Collect_Ents e_ma
     rker))
                                                    (setq last_run_ents nil)))
                                         (progn (if (setq Pd2 (getpoint pt_input "\n2à Punto Di
     rezione asse: "))
                                                    (progn  (if (and (= opt_m "Sostituisci")
     last_run_ents)
                                                                (foreach e last_run_ents (if (e
     ntget e) (entdel e))))
                                                            (setq e_marker (entlast))
                                                            (Geom_3Pts_Core_Improprio pts pt_in
     put Pd2)
                                                            (if (= opt_m "Sostituisci")
                                                                (setq last_run_ents (Util_Colle
     ct_Ents e_marker))
                                                                (setq last_run_ents nil)))
                                                    (progn (princ "\n[WARN] Secondo punto manca
     nte. Conica precedente mantenuta."))))))))
  )
  (if guide_ents (foreach e guide_ents (if (entget e) (entdel e)))))
;;-------------------------------------------------------------------------------------
(defun c:con3pc (/ pts old_os old_pd old_ce)
  (mapcar 'set '(pts old_os old_pd old_ce)
     (Init_Conic_App "CON3PC (Master Loop: 3 Punti Fissi)" 3 "\nSeleziona i 3 punti FISSI del f
```

```
     ascio: " 32))
  (if pts
    (progn
      (Draw_Triangolo_Lis3Pu pts *LaCo_Poligoni*)
      (setvar "OSMODE" 0)
      (Geom_3Pts_Master_Engine pts old_os)
      (princ "\nSessione Master terminata."))
    (princ "\nInput annullato."))
  (Restore_Conic_App old_os old_pd old_ce)
  (princ))
)
;;-------------------------------------------------------------------------------------
;;-------------------------------------------------------------------------------------

;;-------------------------------------------------------------------------------------
;;-------------------------------------------------------------------------------------
(progn
;;-------------------------------------------------------------------------------------
;;-------------------------------------------------------------------------------------

;;-------------------------------------------------------------------------------------
;;-------------------------------------------------------------------------------------
(progn
;;-------------------------------------------------------------------------------------
(defun Geom_Master_Draw_Framework (class P_diag / tipo pts_ord)
  (mapcar 'set '(tipo pts_ord) class)
  (cond
    ((member tipo '(CONCAVO_ORTOCENTRICO CONCAVO_GENERICO))
     (Handler_Concavo_4P            pts_ord               *LaCo_Poligoni* *LaCo_Costruzioni*))
    ((member tipo '(TRAPEZIO_ISOSCELE TRAPEZIO_SCALENO))
     (Handler_Trapezio_4P           pts_ord      P_diag   *LaCo_Poligoni* *LaCo_Costruzioni*))
    ((member tipo '(QUADRANGOLO_CICLICO QUADRANGOLO_GENERICO))
     (Handler_Convesso_4P           pts_ord               *LaCo_Poligoni* *LaCo_Costruzioni*))
    ((member tipo '(QUADRATO RETTANGOLO ROMBO PARALLELOGRAMMA))
     (Handler_Simmetria_Centrale_4P pts_ord tipo P_diag   *LaCo_Poligoni* *LaCo_Costruzioni*))
    ((member tipo '(DEGENERE_PUNTO_4 DEGENERE_SEGMENTO_3_1
                    DEGENERE_SEGMENTO_2_2 DEGENERE_TRIANGOLO_2_1_1
                    DEGENERE_ALLINEATI_4 DEGENERE_ALLINEATI_3_1))
     (Handler_Degenere_4P           pts_ord tipo          *LaCo_Poligoni* *LaCo_Costruzioni*)))
     )
;;-------------------------------------------------------------------------------------
(defun Handler_Concavo_4P (pts_ord layPol layCos / P1 P2 P3 P4 i1 i2 i3)
  (mapcar 'set '(P1 P2 P3 P4) pts_ord)
  (mapcar 'set '(i1 i2 i3)       (list (inters P1 P4 P2 P3 nil) (inters P2 P4 P1 P3 nil)
     (inters P3 P4 P1 P2 nil)))
```

```
  (Draw_Lis_Line_Lis_Coppie_Pu_R (list (list P1 P2) (list P2 P3) (list P3 P1) (list P1 P4)
     (list P2 P4) (list P3 P4)) layPol)
  (Draw_Lis_Line_Lis_Coppie_Pu_R (list (list P4 i1) (list P4 i2) (list P4 i3))
     layCos)
  (Draw_Lis_Point_R              (list i1 i2 i3)
     layCos))
;;-------------------------------------------------------------------------------------
(defun Handler_Trapezio_4P (pts_ord I layPol layCos / P1 P2 P3 P4 int1 int2 F_v L_conv1 L_conv2
     )
  (mapcar 'set '(P1 P2 P3 P4) pts_ord)
  (setq int1 (inters P1 P2 P3 P4 nil)
        int2 (inters P2 P3 P1 P4 nil))
  (if int1 (setq F_v int1   L_conv1 (list P1 P2)     L_conv2 (list P3 P4))
           (setq F_v int2   L_conv1 (list P2 P3)     L_conv2 (list P1 P4)))
  (Draw_Lis_Line_Lis_Coppie_Pu_R (list (list P1 P2) (list P2 P3) (list P3 P4) (list P4 P1))
     layPol)
  (Draw_Lis_Line_Lis_Coppie_Pu_R (list (list P1 P3) (list P2 P4))
     layCos)
  (Draw_Lis_Line_Lis_Coppie_Pu_R (list (Geom_MinMax_Dist_LisPu L_conv1 F_v 3)
                                       (Geom_MinMax_Dist_LisPu L_conv2 F_v 3))
     layCos)
  (Draw_Lis_Point_R              (list F_v I)        layCos))
;;-------------------------------------------------------------------------------------
(defun Handler_Convesso_4P (pts_ord layPol layCos / P1 P2 P3 P4 F1 F2)
  (mapcar 'set '(P1 P2 P3 P4) pts_ord)
  (setq F1 (inters P1 P2 P3 P4 nil)
        F2 (inters P2 P3 P1 P4 nil))
  (Draw_Lis_Line_Lis_Coppie_Pu_R (list (list P1 P2) (list P2 P3) (list P3 P4) (list P4 P1))
     layPol)
  (Draw_Lis_Line_Lis_Coppie_Pu_R (list (list P1 P3) (list P2 P4))
     layCos)
  (Draw_Lis_Line_Lis_Coppie_Pu_R (list (Geom_MinMax_Dist_LisPu (list P1 P2) F1 3)
                                       (Geom_MinMax_Dist_LisPu (list P3 P4) F1 3)
                                       (Geom_MinMax_Dist_LisPu (list P2 P3) F2 3)
                                       (Geom_MinMax_Dist_LisPu (list P1 P4) F2 3))
     layCos)
  (Draw_Lis_Point_R              (list F1 F2)                                    layCos))
;;-------------------------------------------------------------------------------------
(defun Handler_Simmetria_Centrale_4P (pts_ord tipo_f I layPol layCos / P1 P2 P3 P4)
  (princ (strcat "\n[INFO] Simmetria Centrale (" (vl-symbol-name tipo_f) ") rilevata."))
  (mapcar 'set '(P1 P2 P3 P4) pts_ord)
  (Draw_Lis_Line_Lis_Coppie_Pu_R  (list (list P1 P2) (list P2 P3) (list P3 P4) (list P4 P1))
     layPol)
  (Draw_Lis_Line_Lis_Coppie_Pu_R  (list (list P1 P3) (list P2 P4))
     layCos)
```

```
  (if I (Draw_Point_R I layCos)))
;;-------------------------------------------------------------------------------------
(defun Handler_Degenere_4P (pts_ord tipo_f layPol layCos)
  (princ (strcat "\n[ATTENZIONE] Configurazione " (vl-symbol-name tipo_f) " intercettata."))
  (princ "\n[DRAW] Impossibile generare l'impalcatura standard. Vengono tracciati solo i punti
     base.")
  (Draw_Lis_Point_R  pst_ord layCos))
)
;;-------------------------------------------------------------------------------------
(defun Geom_ConCen_4PC (cen pts / best_err best_res best_coeffs i j k pA pB pC coeffs A B C err
     _curr pt_test x y val_test)
  (setq best_err 1.0e99 best_res nil best_coeffs nil i 0)
  (while (< i 2)
    (setq j (+ i 1))
    (while (< j 3)
      (setq k (+ j 1))
      (while (< k 4)
        (setq pA (nth i pts)
              pB (nth j pts)
              pC (nth k pts))
        (setq coeffs (Alg3x3_Calcola_Coeffs cen pA pB pC))
        (if coeffs
          (progn
            (mapcar 'set '(A B C) coeffs)
            (setq err_curr 0.0)
            (foreach idx '(0 1 2 3)
              (if (and (/= idx i) (/= idx j) (/= idx k))
                (progn
                  (setq pt_test (mapcar '- (nth idx pts) cen))
                  (mapcar 'set '(x y) pt_test)
                  (setq val_test (+ (* A x x) (* B x y) (* C y y))
                        err_curr (abs (- val_test 1.0))))))
            (if (< err_curr best_err)
                (setq best_err err_curr best_res (list pA pB pC) best_coeffs coeffs))))
        (setq k (1+ k)))
      (setq j (1+ j)))
    (setq i (1+ i)))
  (if (and best_res (< best_err 1.0))
      (list best_err best_res best_coeffs)
      nil))
;;-------------------------------------------------------------------------------------
(defun Draw_ConCen_4PC (cen pts lay / math_data err res)
  (princ "\n[SYSTEM] Avvio Torneo 3-su-4 (Centro Fisso)...")
  (setq math_data (Geom_ConCen_4PC cen pts))
  (if math_data
```

```
      (progn
        (setq err (car math_data) res (cadr math_data))
        (princ (strcat "\n[DECISIONE] Vincitore finale con residuo: " (rtos err 2 16)))
        (Draw_ConCen_3P_Compl cen (car res) (cadr res) (caddr res) pts))
      (princ "\n[ERRORE] Geometria impossibile per il centro dato."))
  (princ))
;;-------------------------------------------------------------------------------------------
(defun Math_Extract_Geom_Data_From_Coeffs (K PuC m_dist / A B C D E F det h k F_prime delta A_n
     ew C_new a_n b_n ang_rad C_wcs V_dir a_w b_w)
  (mapcar 'set '(A B C D E F) K)
  (setq det (- (* 4.0 A C) (* B B)))
  (if (not (equal det 0.0 1e-12))
    (progn
      (setq h       (/ (- (* B E) (* 2.0 C D)) det)
            k       (/ (- (* B D) (* 2.0 A E)) det)
            F_prime (+ (* A h h) (* B h k) (* C k k) (* D h) (* E k) F))
      (if (not (equal F_prime 0.0 1e-12))
        (progn
          (setq delta   (sqrt (+ (Math_Sqr (- A C)) (Math_Sqr B)))
                A_new   (* 0.5 (+ A C delta))
                C_new   (* 0.5 (- (+ A C) delta))
                ang_rad (if (equal A C 1e-12) (/ pi 4.0) (* 0.5 (atan B (- A C)))))
          (if (> (/ (- F_prime) A_new) 0)
              (setq a_n (sqrt (/ (- F_prime) A_new))    b_n (sqrt (abs (/ (- F_prime) C_new))))
              (setq a_n (sqrt (/ (- F_prime) C_new))    b_n (sqrt (abs (/ (- F_prime) A_new)))
                    ang_rad (+ ang_rad (/ pi 2.0))))
          (setq C_wcs (Vet_Sum PuC (Vet_Sca (list h k 0.0) m_dist))
                V_dir (list (cos ang_rad) (sin ang_rad) 0.0)
                a_w (* a_n m_dist)
                b_w (* b_n m_dist))
          (list C_wcs V_dir a_w b_w))))))
;;-------------------------------------------------------------------------------------------
(defun Geom_Format_Map_Data (v_list f_type g_data)  (mapcar '(lambda (v)
     (append (list v f_type) g_data)) v_list))
;;-------------------------------------------------------------------------------------------
(defun Geom_Conic_4Pf_Engine (P1 P2 P3 P4 Cg geo_packet opt_m opt_e last_ents / f_type Center V
     _dir a b V1 F1 p_list max_p x_limit dummy_pt ang_rad e_marker)

  (if (and (= opt_m "Sostituisci") last_ents)
    (foreach e last_ents (if (entget e) (entdel e))))

  (setq e_marker (entlast)
        f_type   (cadr geo_packet))

  (cond
```

```
    ((eq f_type 'PARABOLA)
     (setq ang_rad (caddr geo_packet))
     (Draw_Conic_4P_AxisDir P1 P2 P3 P4 Cg (polar Cg ang_rad 1000.0)
     *LaCo_Tracciati* *LaCo_Costruzioni* *LaCo_Punti_scelti* nil))

    ((eq f_type 'ELLISSE)
     (mapcar 'set '(Center V_dir a b) (cddr geo_packet))
     (setq V1 (Vet_Sum Center (Vet_Sca V_dir a))
           F1 (Vet_Sum Center (Vet_Sca V_dir (sqrt (abs (- (* a a) (* b b)))))))
     (Draw_ConEll_CeVe_CoVe Center V1 F1 (list P1 P2 P3 P4)))

    ((eq f_type 'IPERBOLE)
     (mapcar 'set '(Center V_dir a b) (cddr geo_packet))
     (setq V1       (Vet_Sum Center (Vet_Sca V_dir a))
           F1       (Vet_Sum Center (Vet_Sca V_dir (Pitag_Ip a b)))
           p_list   (list P1 P2 P3 P4)
           max_p    (get-max-abs-projection-on-axis p_list Center V_dir)
           x_limit  (* (max max_p (* a 1.2)) opt_e)
           dummy_pt (Vet_Sum Center (Vet_Sca V_dir x_limit)))
     (Draw_ConIpe_CeVeFu_Compl_Free Center V1 F1 (append p_list (list dummy_pt)))))

  (Safe_Collect_Ents e_marker))
;;-------------------------------------------------------------------------------------
(defun Geom_Conic_4Pv_Engine (P1 P2 P3 P4 Cg geo_packet opt_m opt_e last_ents / f_type Center V
     _dir a b V1 F1 p_list max_p x_limit dummy_pt ang_rad e_marker)

  (if (and (= opt_m "Sostituisci") last_ents) (foreach e last_ents (if (entget e) (entdel e))))

  (setq e_marker (entlast)
        f_type   (cadr geo_packet))

  (cond  ((eq f_type 'PARABOLA)  (setq ang_rad (caddr geo_packet))
                                 (Draw_Conic_4P_AxisDir P1 P2 P3 P4 Cg (polar Cg ang_rad 1000.0
     ) *LaCo_Tracciati* *LaCo_Costruzioni* *LaCo_Punti_scelti* nil))

         ((eq f_type 'ELLISSE)   (mapcar 'set  '(Center V_dir a b) (cddr geo_packet))
                                 (setq V1       (Vet_Sum Center (Vet_Sca V_dir a))
                                       F1       (Vet_Sum Center (Vet_Sca V_dir (sqrt (abs (- (*
     a a) (* b b)))))))
                                 (Draw_ConEll_CeVe_CoVe Center V1 F1 (list P1 P2 P3 P4)))

         ((eq f_type 'IPERBOLE)  (mapcar 'set  '(Center V_dir a b) (cddr geo_packet))
                                 (setq V1       (Vet_Sum Center (Vet_Sca V_dir a))
                                       F1       (Vet_Sum Center (Vet_Sca V_dir (Pitag_Ip a b)))
                                       p_list   (list P1 P2 P3 P4)
```

```
                                        max_p    (get-max-abs-projection-on-axis p_list Center V
    _dir)
                                        x_limit  (* (max max_p (* a 1.2)) opt_e)
                                        dummy_pt (Vet_Sum Center (Vet_Sca V_dir x_limit)))
                                   (Draw_ConIpe_CeVeFu_Compl_Free Center V1 F1 (append p_list (li
    st dummy_pt)))))
  (Safe_Collect_Ents e_marker))
;;-------------------------------------------------------------------------------------------
(defun Math_ConIpeEqu_Coeffs (cen pts / pt1 pt2 pt3 cx cy p1x p1y p2x p2y p3x p3y u1 v1 u2 v2 u
    3 v3 det1 det2 det3 abs1 abs2 abs3 max_det best_A best_B)

  (mapcar 'set '(pt1 pt2 pt3) pts)
  (setq cx (car cen) cy (cadr cen))

  (setq p1x (- (car pt1) cx)  p1y (- (cadr pt1) cy)
        p2x (- (car pt2) cx)  p2y (- (cadr pt2) cy)
        p3x (- (car pt3) cx)  p3y (- (cadr pt3) cy))

  (setq u1 (- (* p1x p1x) (* p1y p1y))  v1 (* p1x p1y)
        u2 (- (* p2x p2x) (* p2y p2y))  v2 (* p2x p2y)
        u3 (- (* p3x p3x) (* p3y p3y))  v3 (* p3x p3y))

  (setq det1 (- (* u1 v2) (* u2 v1))  abs1 (abs det1)
        det2 (- (* u2 v3) (* u3 v2))  abs2 (abs det2)
        det3 (- (* u3 v1) (* u1 v3))  abs3 (abs det3))

  (setq max_det -1.0)
  (if (> abs1 max_det) (setq max_det abs1  best_A (/ (- v2 v1) det1)  best_B (/ (- u1 u2)
    det1)))
  (if (> abs2 max_det) (setq max_det abs2  best_A (/ (- v3 v2) det2)  best_B (/ (- u2 u3)
    det2)))
  (if (> abs3 max_det) (setq max_det abs3  best_A (/ (- v1 v3) det3)  best_B (/ (- u3 u1)
    det3)))

  (if (> max_det 1e-10) (list best_A best_B (- best_A) 0.0 0.0 -1.0)))
;;-------------------------------------------------------------------------------------------
(defun CON4P_Execute_Conic_Action (geo_data mode P1 P2 P3 P4 Cg opt_m opt_e last_run_ents / e_m
    arker)
  (if (eq (cadr geo_data) 'PARABOLA)
      (progn
        (if (and (= opt_m "Sostituisci") last_run_ents) (foreach e last_run_ents (if (entget e)
    (entdel e))))
        (setq e_marker (entlast))
        (CON4P_Geom_Draw_Target_Parabola P1 P2 P3 P4 (car geo_data) mode)
        (CON4P_Local_Collect e_marker))
```

```
      (if (= mode "FOCI")
          (Geom_Conic_4Pf_Engine P1 P2 P3 P4 Cg geo_data opt_m opt_e last_run_ents)
          (Geom_Conic_4Pv_Engine P1 P2 P3 P4 Cg geo_data opt_m opt_e last_run_ents))))
;;-------------------------------------------------------------------------------------
(defun CON4P_Execute_Center_Conic (action_data opt_m opt_e last_run_ents P1 P2 P3 P4 pts_ord Cg
     / e_marker pt_proj_act D_ref c_norm p_norm math_data K_full g_data det f_type V_dir a b tm
     p g_data_wcs)
  (if (and (= opt_m "Sostituisci") last_run_ents) (foreach e last_run_ents (if (entget e)
     (entdel e))))
  (setq e_marker (entlast) pt_proj_act (nth 2 action_data))
  (Draw_Point_R pt_proj_act *LaCo_Centri*)

  (setq D_ref (distance P1 P3)) (if (<= D_ref 0.0) (setq D_ref 1.0))
  (setq c_norm (mapcar '(lambda (c) (/ c D_ref)) pt_proj_act) p_norm (mapcar '(lambda (p)
     (mapcar '(lambda (c) (/ c D_ref)) p)) pts_ord))

  (if (setq math_data (Geom_ConCen_4PC c_norm p_norm))
      (progn
        (setq K_full (list (nth 0 (caddr math_data)) (nth 1 (caddr math_data))
     (nth 2 (caddr math_data)) 0.0 0.0 -1.0)
     g_data (Math_Extract_Geom_Data_From_Coeffs K_full c_norm 1.0))
        (if g_data
          (progn
            (setq det    (- (* 4.0 (nth 0 K_full) (nth 2 K_full)) (* (nth 1 K_full)
     (nth 1 K_full)))
                  f_type (if (> det 0) 'ELLISSE 'IPERBOLE))
            (setq V_dir  (nth 1 g_data)
                  a      (* (nth 2 g_data) D_ref)
                  b      (* (nth 3 g_data) D_ref))
            (if (and (eq f_type 'ELLISSE) (< a b))
                (setq V_dir (list (- (cadr V_dir)) (car V_dir) 0.0)
                      tmp   a
                      a     b
                      b     tmp))
            (setq g_data_wcs (list pt_proj_act V_dir a b))
            (Geom_Conic_4Pf_Engine P1 P2 P3 P4 Cg (cons nil (cons f_type g_data_wcs))
     "Aggiungi" opt_e nil)))))
  (CON4P_Local_Collect e_marker))
;;-------------------------------------------------------------------------------------
(defun CON4P_Refresh_Last_Conic (last_geo opt_m opt_e last_run_ents P1 P2 P3 P4 pts_ord Cg pts_
     map_f)
  (if (eq (car last_geo) 'PUNTO)
      (CON4P_Execute_Center_Conic last_geo opt_m opt_e last_run_ents P1 P2 P3 P4 pts_ord Cg)
      (CON4P_Execute_Conic_Action last_geo (if (member last_geo pts_map_f) "FOCI" "VERTICES")
     P1 P2 P3 P4 Cg opt_m opt_e last_run_ents)))
```

```
;;-------------------------------------------------------------------------------------
(defun CON4P_Evaluate_LUSNAP_Target (p5 par_data_map meta_map pts_ord mode D_ref / best_para d_
     para micro_res d_micro best_micro)
  (setq best_para (CON4P_Geom_Find_Nearest_Data p5 par_data_map)
        d_para    (if best_para (distance p5 (car best_para)) 1e99))
  (setq micro_res (CON4P_Geom_MicroScan p5 meta_map pts_ord mode D_ref)
        d_micro   (car micro_res)
        best_micro (cadr micro_res))
  (if (< d_para d_micro)
      (list d_para best_para)
      (list d_micro best_micro)))
;;-------------------------------------------------------------------------------------
(defun CON4P_Identify_Node_Target (p5 P1 P2 P3 P4 node_solutions last_node_idx sol_idx opt_m op
     t_e last_run_ents Cg / min_d curr_node_idx d_list d curr_sols last_geo)
  (setq min_d 1e99 curr_node_idx -1 d_list (list P1 P2 P3 P4))
  (foreach n '(0 1 2 3)
    (if (< (setq d (distance p5 (nth n d_list))) min_d) (setq min_d d curr_node_idx n)))
  (if (= curr_node_idx last_node_idx)
      (setq sol_idx (1+ sol_idx))
      (setq sol_idx 0 last_node_idx curr_node_idx))
  (setq curr_sols (nth curr_node_idx node_solutions))
  (if curr_sols
      (progn
        (setq last_geo (nth (rem sol_idx (length curr_sols)) curr_sols))
        (Draw_Point_R (car last_geo) *LaCo_Punti_scelti*)
        (setq last_run_ents (CON4P_Execute_Conic_Action last_geo "VERTICES" P1 P2 P3 P4 Cg opt_
     m opt_e last_run_ents)))
      (princ "\n[!] Nessuna soluzione analitica per questo nodo."))
  (list sol_idx last_node_idx last_geo last_run_ents))
;;-------------------------------------------------------------------------------------
(defun CON4P_Identify_Locus_Target (p5 opt_gc opt_gf opt_gv locus_vla_list par_data_f meta_map_
     f par_data_v meta_map_v pts_ord P1 P2 P3 P4 Cg opt_m opt_e last_run_ents / t_click_start d
     _c d_f d_v best_c best_f best_v pt_proj min_d winner_type last_geo t_click_end)
  (setq t_click_start (getvar "MILLISECS"))
  (setq d_c 1e99 d_f 1e99 d_v 1e99 best_c nil best_f nil best_v nil)

  (if (and (= opt_gc "ON")
     (setq pt_proj (CON4P_Geom_Get_Projected_Center p5 locus_vla_list nil)))
                          (setq d_c    (distance p5 pt_proj)
                                best_c (list 'PUNTO p5 pt_proj)))
  (if (= opt_gf "ON") (mapcar 'set '(d_f best_f)
     (CON4P_Evaluate_LUSNAP_Target p5 (cadr par_data_f)
     meta_map_f pts_ord "FOCI"     (distance P1 P3))))
  (if (= opt_gv "ON") (mapcar 'set '(d_v best_v)
     (CON4P_Evaluate_LUSNAP_Target p5 (cadr par_data_v)
```

```
    meta_map_v pts_ord "VERTICES" (distance P1 P3))))

  (setq min_d (min d_c d_f d_v) winner_type "")
  (cond
    ((= min_d 1e99) (princ "\n[!] Tutti i LUSNAP attivi sono vuoti o spenti."))
    ((= min_d d_v)  (setq last_geo best_v winner_type "VERTICES") (Draw_Point_R (car best_v)
     *LaCo_Punti_scelti*)
     (setq last_run_ents (CON4P_Execute_Conic_Action best_v "VERTICES" P1 P2 P3 P4 Cg opt_m opt
     _e last_run_ents)))
    ((= min_d d_f)  (setq last_geo best_f winner_type "FOCI")     (Draw_Point_R (car best_f)
     *LaCo_Punti_scelti*)
     (setq last_run_ents (CON4P_Execute_Conic_Action best_f "FOCI" P1 P2 P3 P4 Cg opt_m opt_e l
     ast_run_ents)))
    ((= min_d d_c)  (setq last_geo best_c winner_type "CENTRI")
     (setq last_run_ents (CON4P_Execute_Center_Conic best_c opt_m opt_e last_run_ents P1 P2 P3
     P4 pts_ord Cg)))
  )

  (setq t_click_end (getvar "MILLISECS"))
  (if (/= winner_type "")
      (princ (strcat "\n[DEBUG] Vincitore: " winner_type " | Tempo: " (itoa (- t_click_end t_cl
     ick_start)) " ms | Distanza: " (rtos min_d 2 6))))
  (list last_geo last_run_ents))
;;-------------------------------------------------------------------------------------
(defun Math_Solve_Cubic_Real (A B_coeff C D / a_norm b_norm c_norm Q R D_isc theta sqrt_Q S_val
     T_val root1 root2 root3)
  (if (< (abs A) 1e-9)
    (if (< (abs B_coeff) 1e-9)
      (if (< (abs C) 1e-9) nil (list (/ (- D) C)))
      (progn
        (setq D_isc (- (* C C) (* 4.0 B_coeff D)))
        (cond
          ((< D_isc -1e-9) nil)
          ((<= (abs D_isc) 1e-9) (list (/ (- C) (* 2.0 B_coeff))))
          (T (list (/ (+ (- C) (sqrt D_isc)) (* 2.0 B_coeff))
                   (/ (- (- C) (sqrt D_isc)) (* 2.0 B_coeff)))))))
    (progn
      (setq a_norm (/ B_coeff A)
            b_norm (/ C A)
            c_norm (/ D A))
      (setq Q (/ (- (* 3.0 b_norm) (* a_norm a_norm)) 9.0)
            R (/ (- (* 9.0 a_norm b_norm) (+ (* 27.0 c_norm) (* 2.0 a_norm a_norm a_norm)))
     54.0)
            D_isc (+ (* Q Q Q) (* R R)))
```

```
      (cond
        ((<= D_isc 1e-9)
         (setq sqrt_Q (sqrt (abs Q))
               theta  (Math_Acos (max -1.0 (min 1.0 (/ R (* sqrt_Q sqrt_Q sqrt_Q))))))
         (setq root1 (- (* 2.0 sqrt_Q (cos (/ theta 3.0))) (/ a_norm 3.0))
               root2 (- (* 2.0 sqrt_Q (cos (/ (+ theta (* 2.0 pi)) 3.0))) (/ a_norm 3.0))
               root3 (- (* 2.0 sqrt_Q (cos (/ (+ theta (* 4.0 pi)) 3.0))) (/ a_norm 3.0)))
         (list root1 root2 root3)
        )
        (T
         (setq S_val (if (>= (+ R (sqrt D_isc)) 0.0)
                         (expt (+ R (sqrt D_isc)) (/ 1.0 3.0))
                         (- (expt (abs (+ R (sqrt D_isc))) (/ 1.0 3.0)))))
         (setq T_val (if (>= (- R (sqrt D_isc)) 0.0)
                         (expt (- R (sqrt D_isc)) (/ 1.0 3.0))
                         (- (expt (abs (- R (sqrt D_isc))) (/ 1.0 3.0)))))
         (list (- (+ S_val T_val) (/ a_norm 3.0)))  )  )  )  )  )
;;-------------------------------------------------------------------------------------
(defun Geom_Eval_Matrix_Parallelism (M P / a b c d_coeff e N_x N_y Det V_x_num V_y_num x0 y0)
  (setq a (nth 0 M) b (nth 1 M) c (nth 2 M) d_coeff (nth 3 M) e (nth 4 M) x0 (car P)
     y0 (cadr P))
  (setq Det (- (* 4.0 a c) (* b b)))
  (setq N_x (+ (* 2.0 a x0) (* b y0) d_coeff)
        N_y (+ (* b x0) (* 2.0 c y0) e))
  (setq V_x_num (- (* x0 Det) (- (* b e) (* 2.0 c d_coeff)))
        V_y_num (- (* y0 Det) (- (* b d_coeff) (* 2.0 a e))))
  (- (* V_x_num N_y) (* V_y_num N_x)))
;;-------------------------------------------------------------------------------------
(defun Geom_Precalc_Node_Conics (P1 P2 P3 P4 / PuC_Ref P1n P2n P3n P4n K1 K2 node_list_abs mast
     er_solution_list curr_node_abs curr_node_loc K_plus K_minus F_plus F_minus coeff_A coeff_B
     coeff_C coeff_D lambda_roots node_valid_sols K_lam Geom_Data f_type det_norm max_k geo_pac
     ket)
  (setq PuC_Ref (Pu_Baricentro2D_LisPu (list P1 P2 P3 P4)))

  (setq P1n (Vet_Sub P1 PuC_Ref) P2n (Vet_Sub P2 PuC_Ref) P3n (Vet_Sub P3 PuC_Ref)
     P4n (Vet_Sub P4 PuC_Ref))
  (setq K1 (Math_Multiply_Lines_To_Conic (Math_Line_General_Eq P1n P2n)
     (Math_Line_General_Eq P3n P4n))
        K2 (Math_Multiply_Lines_To_Conic (Math_Line_General_Eq P1n P3n)
     (Math_Line_General_Eq P2n P4n)))

  (setq node_list_abs (list P1 P2 P3 P4) master_solution_list nil)

  (foreach curr_node_abs node_list_abs
    (setq node_valid_sols nil)
```

```
    (setq curr_node_loc (Vet_Sub curr_node_abs PuC_Ref))

    (setq coeff_D (Geom_Eval_Matrix_Parallelism K1 curr_node_loc)
          coeff_A (Geom_Eval_Matrix_Parallelism K2 curr_node_loc)
          K_plus  (Math_Combine_Conics K1 K2  1.0)
          K_minus (Math_Combine_Conics K1 K2 -1.0)
          F_plus  (Geom_Eval_Matrix_Parallelism K_plus  curr_node_loc)
          F_minus (Geom_Eval_Matrix_Parallelism K_minus curr_node_loc)
          coeff_B (- (/ (+ F_plus F_minus) 2.0) coeff_D)
          coeff_C (- (/ (- F_plus F_minus) 2.0) coeff_A))

    (setq lambda_roots (Math_Solve_Cubic_Real coeff_A coeff_B coeff_C coeff_D))

    (if lambda_roots
      (foreach lam lambda_roots
        (setq K_lam (Math_Combine_Conics K1 K2 lam))
        (setq max_k (max (abs (nth 0 K_lam)) (abs (nth 1 K_lam)) (abs (nth 2 K_lam))))

        (if (> max_k 1e-12)
            (setq det_norm (/ (- (* 4.0 (nth 0 K_lam) (nth 2 K_lam)) (* (nth 1 K_lam)
     (nth 1 K_lam))) (* max_k max_k)))
            (setq det_norm 0.0))

        (if (not (equal det_norm 0.0 1e-6))
          (progn
            (setq Geom_Data (Math_Extract_Geom_Data_From_Coeffs K_lam PuC_Ref 1.0))
            (if Geom_Data
              (progn
                (setq f_type (if (> det_norm 0) 'ELLISSE 'IPERBOLE))
                (if (and (eq f_type 'ELLISSE) (< (nth 2 Geom_Data) (nth 3 Geom_Data)))
                    (setq Geom_Data (list (nth 0 Geom_Data)
                                          (list (- (cadr (nth 1 Geom_Data)))
     (car (nth 1 Geom_Data)) 0.0)
                                          (nth 3 Geom_Data)
                                          (nth 2 Geom_Data))))
                (setq geo_packet (cons curr_node_abs (cons f_type Geom_Data)))
                (setq node_valid_sols (append node_valid_sols (list geo_packet)))  )  )  )  )
     )  )
    (setq master_solution_list (append master_solution_list (list node_valid_sols))))
  master_solution_list)
;;-------------------------------------------------------------------------------------
(defun Math_Classify_Forma_Lis4Pu (pts)  (apply 'Math_Classify_Forma_4Pu pts))
;;-------------------------------------------------------------------------------------
(defun Geom_Master_Analysis_4P (pts / tipo pts_ord)
  (mapcar 'set '(tipo pts_ord) (Math_Classify_Forma_Lis4Pu pts))
```

```
  (list tipo pts_ord (Pu_Baricentro2D_LisPu pts_ord) (Math_Inters_Quaterna pts_ord 3)))
;;-------------------------------------------------------------------------------------
(defun CON4P_Geom_Master_Exit_Handler (opt_gc opt_gr guide_ents_c guide_ents_r group_name)
  (if (and (= opt_gc "OFF") guide_ents_c) (foreach e guide_ents_c (if (entget e) (entdel e))))
  (if (and (= opt_gr "OFF") guide_ents_r) (foreach e guide_ents_r (if (entget e) (entdel e))))
  (princ (strcat "\nLaboratorio " group_name " concluso."))
  (princ))
;;-------------------------------------------------------------------------------------
(defun CON4P_Geom_4P_Master_Setup (titolo snap_mode / pts old_os old_pd old_ce tipo pts_ord Cg
     P_diag P1 P2 P3 P4 ref_pt layPol layCos)
  (mapcar 'set '(pts old_os old_pd old_ce)
     (Init_Conic_App titolo 4 "\nSeleziona 4 punti: " snap_mode))
  (if pts
      (progn (mapcar 'set '(tipo pts_ord Cg P_diag) (Geom_Master_Analysis_4P pts))
             (mapcar 'set '(P1 P2 P3 P4) pts_ord)
             (setq ref_pt (if P_diag P_diag Cg) layPol (Util_SafeLayer "Coniche - Poligoni")
     layCos (Util_SafeLayer "Coniche - Costruzioni"))
             (Geom_Master_Draw_Framework (list tipo pts_ord) P_diag)
             (list pts old_os old_pd old_ce tipo pts_ord P1 P2 P3 P4 Cg P_diag ref_pt))))
;;-------------------------------------------------------------------------------------
(defun CON4P_Get_Config_String (tipo)
  (cond ((wcmatch (vl-symbol-name tipo) "CONCAVO*")  "CC")
        ((wcmatch (vl-symbol-name tipo) "TRAPEZIO*") "TR")
        (T "CV")))
;;-------------------------------------------------------------------------------------
(defun CON4P_Geom_Format_Trapezoid_Locus (P1 P2 P3 P4 old_vla_list / cfg ext_t v12 v34 M1 M2 F1
     ID P_M pts_line dir sorted_pts d_tot p_ext1 p_ext2 pts_all i lay_locus new_vla_list return
     _list e_p5p6)
  (setq cfg (CON4P_Get_Global_Config) ext_t (cdr (assoc 'ext_trap cfg)))
  (foreach obj old_vla_list (if (= (type obj) 'ENAME) (entdel obj) (vla-delete obj)))
  (setq lay_locus (if *LaCo_Luoghi_geometrici* *LaCo_Luoghi_geometrici* "Coniche - Luoghi geome
     trici"))
  (setq v12 (Vet_Sub P2 P1) v34 (Vet_Sub P4 P3))
  (if (P_Are_Vectors_Parallel_2D v12 v34 1e-5)
    (setq M1 (Math_PuntoMedio_2Pu P1 P2) M2 (Math_PuntoMedio_2Pu P3 P4)
     F1 (inters P1 P4 P2 P3 nil))
    (setq M1 (Math_PuntoMedio_2Pu P1 P4) M2 (Math_PuntoMedio_2Pu P2 P3)
     F1 (inters P1 P2 P3 P4 nil)))
  (setq ID (inters P1 P3 P2 P4 nil) P_M (Math_PuntoMedio_2Pu M1 M2))
  (setq pts_line (vl-remove nil (list F1 M1 ID P_M M2)))
  (setq dir (angle M1 M2))
  (setq sorted_pts (vl-sort pts_line '(lambda (a b) (< (* (distance M1 a) (cos (- (angle M1 a)
     dir))) (* (distance M1 b) (cos (- (angle M1 b) dir)))))))
  (setq d_tot (distance (car sorted_pts) (last sorted_pts)))
  (setq p_ext1 (polar (car sorted_pts) (angle (cadr sorted_pts) (car sorted_pts))
```

```
    (* ext_t d_tot)))
  (setq p_ext2 (polar (last sorted_pts) (angle (nth (- (length sorted_pts) 2) sorted_pts)
    (last sorted_pts)) (* ext_t d_tot)))
  (setq pts_all (append (list p_ext1) sorted_pts (list p_ext2))
    new_vla_list nil return_list nil i 0)
  (entmake (list '(0 . "LINE")
    (cons 8 (if *LaCo_Costruzioni* *LaCo_Costruzioni* "Coniche - Costruzioni")) (cons 10 M1)
    (cons 11 M2)))
  (setq e_p5p6 (entlast) return_list (list e_p5p6))
  (while (< i (1- (length pts_all)))
    (entmake (list '(0 . "LINE") (cons 8 lay_locus) (cons 10 (nth i pts_all))
     (cons 11 (nth (1+ i) pts_all))))
    (setq new_vla_list (cons (vlax-ename->vla-object (entlast)) new_vla_list)
     return_list (cons (entlast) return_list))
    (setq i (1+ i)))
  (cons (reverse new_vla_list) (reverse return_list)))
;;-----------------------------------------------------------------------------------------
(defun CON4P_Capture_Locus_Entities (ent_start lay_target / ename new_objs ent_data obj)
  (setq new_objs nil)
  (if ent_start
      (setq ename (entnext ent_start))
      (setq ename (entnext)))
  (while ename
    (setq ent_data (entget ename))
    (if (and (= (strcase (cdr (assoc 8 ent_data))) (strcase lay_target))
             (member (cdr (assoc 0 ent_data)) '("LINE" "XLINE" "SPLINE" "ELLIPSE" "CIRCLE")))
        (progn
          (setq obj (vlax-ename->vla-object ename))
          (setq new_objs (cons obj new_objs))))
    (setq ename (entnext ename)))
  (reverse new_objs))
;;-----------------------------------------------------------------------------------------
(defun Math_Circonferenza_Eulero_4Pu (Pu1 Pu2 Pu3 Pu4 / Pu5 Pu6 Pu7 Pu8 cir1 cir2 cir3 Ce1 ra1
     Ce2 ra2 Ce3 ra3)
  (if (P_Concavo_Ortocentrico_4Pu Pu1 Pu2 Pu3 Pu4)
      (progn (mapcar 'set '(Pu5 Pu6 Pu7 Pu8) (Geom_Ordina_Concavo_4Pu Pu1 Pu2 Pu3 Pu4))
             (setq cir1 (apply 'Math_Circonferenza_3Pu (Math_lisPuntoMedio_lis2Pu (list (list P
     u5 Pu6) (list Pu6 Pu7) (list Pu7 Pu5))))
                   cir2 (apply 'Math_Circonferenza_3Pu (Math_lisPuntoMedio_lis2Pu (list (list P
     u5 Pu8) (list Pu6 Pu8) (list Pu7 Pu8))))
                   cir3 (apply 'Math_Circonferenza_3Pu (Math_Proiezione_lisTerne  (list (list P
     u5 Pu6 Pu7) (list Pu6 Pu5 Pu7) (list Pu7 Pu5 Pu6)))))

             (if (and cir1 cir2 cir3)
                 (progn  (mapcar 'set '(Ce1 ra1) cir1)
```

```
                        (mapcar 'set '(Ce2 ra2) cir2)
                        (mapcar 'set '(Ce3 ra3) cir3)
                        (list (Pu_Baricentro2D_LisPu (list Ce1 Ce2 Ce3))
                              (Util_Media_lisNum (list ra1 ra2 ra3))))
                (progn (princ "\n[ERR] Errore nel calcolo delle terne di Eulero.") nil)))
      (progn (princ "\n[ERR] I punti non formano un sistema ortocentrico.")
             nil)))
;;-------------------------------------------------------------------------------------
(defun CON4P_Geom_Draw_Locus_And_Capture (pts_raw classification / cfg tol_g ext_c ext_v ext_t
     tipo pts_ord e_start p0 p1 p2 p3 m1 m2 v_dir pts_mid pts_ext eulero_data)
  (setq cfg   (CON4P_Get_Global_Config)
        tol_g (cdr (assoc 'tol_geom cfg))
        ext_c (cdr (assoc 'ext_conc cfg))
        ext_v (cdr (assoc 'ext_conv cfg))
        ext_t (cdr (assoc 'ext_trap cfg)))

  (mapcar 'set '(tipo pts_ord) classification)
  (princ (strcat "\n[ANALISI] Configurazione: " (vl-symbol-name tipo)))

  (setq e_start (entlast))

  (cond
    ((member tipo '(QUADRATO RETTANGOLO ROMBO PARALLELOGRAMMA))
     (princ " -> Luogo: PUNTO UNICO.")
     (Draw_Point_R (Pu_Baricentro2D_LisPu pts_ord) *LaCo_Luoghi_geometrici*))

    ((member tipo '(TRAPEZIO_ISOSCELE TRAPEZIO_SCALENO))
     (princ " -> Luogo: RETTA.")
     (setq v_dir (Geom_Check_Trapezoid_Direction pts_ord))
     (mapcar 'set '(p0 p1 p2 p3) pts_ord)
     (if (P_Are_Vectors_Parallel_2D (mapcar '- p1 p0) v_dir tol_g)
         (setq m1 (Math_PuntoMedio_2Pu p0 p1) m2 (Math_PuntoMedio_2Pu p2 p3))
         (setq m1 (Math_PuntoMedio_2Pu p0 p3) m2 (Math_PuntoMedio_2Pu p1 p2)))
     (setq pts_ext (Geom_2Pu_Simm_PuC (Vet_Mid m1 m2) (Vet_Uni (mapcar '- m2 m1))
     (* (distance m1 m2) ext_t)))
     (Draw_Line_2Pu_R (car pts_ext) (cadr pts_ext) *LaCo_Luoghi_geometrici*))

    ((eq tipo 'CONCAVO_ORTOCENTRICO)
     (princ " -> Luogo: CERCHIO DI EULERO.")
     (setq eulero_data (apply 'Math_Circonferenza_Eulero_4Pu pts_raw))
     (if eulero_data (Draw_Circle_R (car eulero_data) (cadr eulero_data)
     *LaCo_Luoghi_geometrici*)))

    ((eq tipo 'CONCAVO_GENERICO)
     (princ " -> Luogo: ELLISSE 9 PUNTI.")
```

```
     (setq pts_mid (vl-remove nil (apply 'Math_ConicaCentri_9Pu pts_raw))
           pts_ext (Util_Scale_Pts_From_Centroid pts_raw ext_c))
     (Draw_Con5P_Nor_Light (list (car pts_mid) (cadr pts_mid) (caddr pts_mid) (cadddr pts_mid)
     (nth 4 pts_mid)) pts_ext))

    ((member tipo '(QUADRANGOLO_CICLICO QUADRANGOLO_GENERICO))
     (princ " -> Luogo: IPERBOLE 9 PUNTI.")
     (setq pts_mid (vl-remove nil (apply 'Math_ConicaCentri_9Pu pts_raw))
           pts_ext (Util_Scale_Pts_From_Centroid pts_raw ext_v))
     (Draw_Con5P_Nor_Light (list (car pts_mid) (cadr pts_mid) (caddr pts_mid) (cadddr pts_mid)
     (nth 4 pts_mid)) pts_ext)))
  (CON4P_Capture_Locus_Entities e_start *LaCo_Luoghi_geometrici*))
;;-------------------------------------------------------------------------------------------
(defun CON4P_Local_Collect (marker / e lst)
  (setq lst nil e (if marker marker (entnext)))
  (while (setq e (entnext e)) (if (entget e) (setq lst (cons e lst))))
  (reverse lst))
;;-------------------------------------------------------------------------------------------
(defun CON4P_Draw_Points_With_Colors (pt_lists colors lay / old_cecolor res)
  (setq old_cecolor (getvar "CECOLOR") res nil)
  (setq res (mapcar '(lambda (pts col) (if pts (progn (setvar "CECOLOR" col)
     (Draw_Lis_Point_R pts lay)))) pt_lists colors))
  (setvar "CECOLOR" old_cecolor)
  (apply 'append res))
;;-------------------------------------------------------------------------------------------
(defun CON4P_Geom_Get_Projected_Center (pt_ucs vla_list extend / pt_wcs best_pt_wcs min_dist pr
     oj_pt_wcs dist)
  (setq pt_wcs (trans pt_ucs 1 0) best_pt_wcs nil min_dist 1e99)
  (foreach obj vla_list
    (if (not (vl-catch-all-error-p (setq proj_pt_wcs (vl-catch-all-apply 'vlax-curve-getClosest
     PointTo (list obj pt_wcs extend)))))
        (if proj_pt_wcs
            (progn
              (setq dist (distance pt_wcs proj_pt_wcs))
              (if (< dist min_dist) (setq min_dist dist best_pt_wcs proj_pt_wcs))))))
  (if best_pt_wcs (trans best_pt_wcs 0 1) nil))
;;-------------------------------------------------------------------------------------------
(defun CON4P_Geom_Find_Nearest_Data (pt map_list / best_item min_dist d)
  (if (null map_list)
    nil
    (progn
      (setq min_dist 1e99 best_item nil)
      (foreach item map_list
        (setq d (distance pt (car item)))
        (if (< d min_dist) (setq min_dist d best_item item)))
```

```
      best_item)))
;;-------------------------------------------------------------------------------------
(defun CON4P_Geom_Calc_Micro_Param_List (obj p_center step_span ref_len phys_step / st1 en1 p_m
     in p_max pt_min pt_max dist_phys num_steps step_val param_list i curr_p)
  (setq st1   (vlax-curve-getStartParam obj)
        en1   (vlax-curve-getEndParam obj)
        p_min (max st1 (- p_center step_span))
        p_max (min en1 (+ p_center step_span)))

  (setq pt_min (vl-catch-all-apply 'vlax-curve-getPointAtParam (list obj p_min))
        pt_max (vl-catch-all-apply 'vlax-curve-getPointAtParam (list obj p_max)))

  (setq dist_phys (if (not (or (vl-catch-all-error-p pt_min) (vl-catch-all-error-p pt_max)))
                      (distance pt_min pt_max)
                      (* ref_len 0.1)))

  (setq num_steps  (min 200 (max 5 (fix (/ dist_phys phys_step))))
        step_val   (/ (- p_max p_min) (float num_steps))
        param_list nil
        i          0)

  (while (<= i num_steps)
    (setq curr_p (+ p_min (* i step_val)))
    (if (and (> curr_p (+ st1 1e-9)) (< curr_p (- en1 1e-9)))
        (setq param_list (cons curr_p param_list)))
    (setq i (1+ i)))
  (cons p_center param_list))
;;-------------------------------------------------------------------------------------
(defun CON4P_Geom_Eval_Micro_Param (obj param pts_ord mode ref_len p5 / center conic_data f_typ
     e V_dir a b g_data_wcs pt_list map_data d_check local_best_d local_best_geo)
  (setq local_best_d 1e99 local_best_geo nil)
  (if (not (vl-catch-all-error-p (setq center (vl-catch-all-apply 'vlax-curve-getPointAtParam (
     list obj param)))))
    (if (setq conic_data (CON4P_Helper_Solve_Normalized_Conic center pts_ord ref_len 'CENTERED)
     )
      (progn
        (mapcar 'set '(f_type V_dir a b) conic_data)
        (setq g_data_wcs (list center V_dir a b))

        (cond
          ((and (= mode "FOCI")     (eq f_type 'ELLISSE))
     (setq pt_list (Geom_2Pu_Simm_PuC center V_dir (sqrt (abs (- (* a a) (* b b)))))))
          ((and (= mode "FOCI")     (eq f_type 'IPERBOLE))
     (setq pt_list (Geom_2Pu_Simm_PuC center V_dir (Pitag_Ip a b))))
          ((and (= mode "VERTICES") (eq f_type 'ELLISSE))
```

```
     (setq pt_list (append (Geom_2Pu_Simm_PuC center V_dir a)
                                                                          (Geom_2Pu_Sim
     m_PuC center (list (- (cadr V_dir)) (car V_dir) 0.0) b))))
          ((and (= mode "VERTICES") (eq f_type 'IPERBOLE))
     (setq pt_list (Geom_2Pu_Simm_PuC center V_dir a))))
        (setq map_data (Geom_Format_Map_Data pt_list f_type g_data_wcs))

        (mapcar '(lambda (pt item)
                   (if (< (setq d_check (distance p5 pt)) local_best_d)
                       (setq local_best_d d_check local_best_geo item)))
                pt_list map_data))))
  (if local_best_geo
      (list local_best_d local_best_geo)
      nil))
;;-------------------------------------------------------------------------------------
(defun CON4P_Geom_MicroScan (p5 meta_list pts_ord mode ref_len / cfg phys_step best1 best2 d1 d
     2 d_curr item obj1 obj2 p1 p2 s1 s2 contig best_d best_geo
                                                          pt_min pt_max dist_phys num_steps
step_val param_li
     st i scan_batches curr_obj eval_res
                                                          exact_point payload_list final_geo
                                                             Cg x1 y1 x2 y2 M11 M22 M12 D_g
     ram alpha beta A0 B0 C0 dA dB dC norm0 normD K_val th_min th_max
                                                          theta t_val A B C K_full g_data det
f_type V_dir a
     b tmp curr_pts map_data d_check)
  (if (or (null meta_list) (< (length meta_list) 2))
      (list 1e99 nil)
      (progn
        (setq cfg       (CON4P_Get_Global_Config)
              phys_step (* ref_len (cdr (assoc 'micro_step_factor cfg))))

        (setq d1 1e99 d2 1e99 best1 nil best2 nil)
        (foreach item meta_list
          (setq d_curr (distance p5 (car item)))
          (cond  ((< d_curr d1) (setq best2 best1    d2 d1      best1 item     d1 d_curr))
                 ((< d_curr d2) (setq best2 item     d2 d_curr))))
        (setq obj1 (cadr   best1)   p1 (caddr  best1)   s1 (cadddr best1)
              obj2 (cadr   best2)   p2 (caddr  best2)   s2 (cadddr best2)
              contig (and (eq obj1 obj2) (<= (abs (- p1 p2)) (* 1.5 s1)))
              best_d 1e99
              best_geo nil)

        (if (null obj1)
            (progn
```

```
         (setq Cg (Math_PuntoMedio_2Pu (nth 0 pts_ord) (nth 2 pts_ord)))
         (mapcar 'set '(x1 y1) (mapcar '(lambda (c) (/ c ref_len))
(Vet_Sub (nth 0 pts_ord) Cg)))
         (mapcar 'set '(x2 y2) (mapcar '(lambda (c) (/ c ref_len))
(Vet_Sub (nth 1 pts_ord) Cg)))
         (setq M11 (+ (* x1 x1 x1 x1) (* x1 x1 y1 y1) (* y1 y1 y1 y1))
               M22 (+ (* x2 x2 x2 x2) (* x2 x2 y2 y2) (* y2 y2 y2 y2))
               M12 (+ (* x1 x1 x2 x2) (* x1 y1 x2 y2) (* y1 y1 y2 y2))
               D_gram (- (* M11 M22) (* M12 M12))
               alpha (/ (- M22 M12) D_gram) beta (/ (- M11 M12) D_gram)
               A0 (+ (* alpha x1 x1) (* beta x2 x2))
               B0 (+ (* alpha x1 y1) (* beta x2 y2))
               C0 (+ (* alpha y1 y1) (* beta y2 y2))
               dA (* y1 y2) dB (- (+ (* x1 y2) (* x2 y1))) dC (* x1 x2)
               norm0 (sqrt (+ (* A0 A0) (* B0 B0) (* C0 C0)))
               normD (sqrt (+ (* dA dA) (* dB dB) (* dC dC)))
               K_val (if (> normD 1e-9) (/ norm0 normD) 1.0))

         (setq th_min (min p1 p2)
             th_max (max p1 p2))
         (if (< (- th_max th_min) 1e-6)
             (setq th_min (- p1 s1) th_max (+ p1 s1)))

         (setq num_steps 100
               step_val  (/ (- th_max th_min) (float num_steps))
               i 0)

         (while (<= i num_steps)
           (setq theta (+ th_min (* i step_val))
                 t_val (* (/ (sin theta) (cos theta)) K_val)
                 A (+ A0 (* t_val dA)) B (+ B0 (* t_val dB)) C (+ C0 (* t_val dC))
                 K_full (list A B C 0.0 0.0 -1.0)
                 g_data (Math_Extract_Geom_Data_From_Coeffs K_full '(0.0 0.0 0.0) 1.0))
           (if g_data
             (progn
               (setq det (- (* 4.0 A C) (* B B))
                     f_type (if (> det 0) 'ELLISSE 'IPERBOLE)
                     V_dir (nth 1 g_data) a (* (nth 2 g_data) ref_len) b (* (nth 3 g_data)
ref_len))
               (if (and (eq f_type 'ELLISSE) (< a b))
                   (setq V_dir (list (- (cadr V_dir)) (car V_dir) 0.0) tmp a a b b tmp))

               (setq curr_pts nil)
               (cond ((and (= mode "FOCI")     (eq f_type 'ELLISSE))
(setq curr_pts (Geom_2Pu_Simm_PuC Cg V_dir (sqrt (abs (- (* a a) (* b b)))))))
```

```
                    ((and (= mode "FOCI")     (eq f_type 'IPERBOLE))
(setq curr_pts (Geom_2Pu_Simm_PuC Cg V_dir (Pitag_Ip a b))))
                    ((and (= mode "VERTICES") (eq f_type 'ELLISSE))
(setq curr_pts (append (Geom_2Pu_Simm_PuC Cg V_dir a)

(Geom_2Pu_Simm_PuC Cg (list (- (cadr V_dir)) (car V_dir) 0.0) b))))
                    ((and (= mode "VERTICES") (eq f_type 'IPERBOLE))
(setq curr_pts (Geom_2Pu_Simm_PuC Cg V_dir a))))

               (if curr_pts
                 (progn
                   (setq map_data (Geom_Format_Map_Data curr_pts f_type (list Cg V_dir a b
)))
                   (mapcar '(lambda (pt item)
                              (if (< (setq d_check (distance p5 pt)) best_d)
                                  (setq best_d d_check best_geo item)))
                           curr_pts map_data)))))
           (setq i (1+ i)))
         (list best_d best_geo))
       (progn
         (if contig
             (progn (setq pt_min    (vlax-curve-getPointAtParam obj1 p1)
                          pt_max    (vlax-curve-getPointAtParam obj1 p2)
                          dist_phys (if (and pt_min pt_max) (distance pt_min pt_max)
(* ref_len 0.1))
                          num_steps (min 200 (max 5 (fix (/ dist_phys phys_step))))
                          step_val  (/ (- p2 p1) (float num_steps))
                          param_list nil i 0)
                    (while (<= i num_steps) (setq param_list (cons (+ p1 (* i step_val))
param_list) i (1+ i)))
                    (setq scan_batches (list (cons obj1 param_list))))
             (setq scan_batches (list (cons obj1 (CON4P_Geom_Calc_Micro_Param_List obj1 p1
s1 ref_len phys_step))
                                      (cons obj2 (CON4P_Geom_Calc_Micro_Param_List obj2 p2
s2 ref_len phys_step)))))
         (foreach batch scan_batches
           (setq curr_obj (car batch))
           (foreach p (cdr batch)
             (setq eval_res (CON4P_Geom_Eval_Micro_Param curr_obj p pts_ord mode ref_len p
5))
             (if (and eval_res (< (car eval_res) best_d))
                 (setq best_d (car eval_res) best_geo (cadr eval_res)))))
         (list best_d best_geo))))))
;;-------------------------------------------------------------------------------
(defun CON4P_Geom_Draw_Target_Parabola (P1 P2 P3 P4 target_pt mode / cfg tol_m tol_g Angles_Deg
```

```
    V_Norm z_orig pts_o Cg dx dy ang_pca c_pre s_pre D_ref pts_norm c s pts_rot x1 x2 x3 x4 y1
    y2 y3 y4 sq_x1 sq_x2 sq_x3 sq_x4 D_det ka kb kc max_err err Xv_n Yv_n p_dist_n Xv Yv p_dis
    t pts_temp V_wcs F_wcs min_d d best_V best_F l_ents)
  (setq cfg (CON4P_Get_Global_Config) tol_m (cdr (assoc 'tol_math cfg))
    tol_g (cdr (assoc 'tol_geom cfg)))
  (setq Angles_Deg (Geom_Find_Parabolic_Angles P1 P2 P3 P4)
       V_Norm     (Vet_Uni (Vet_Cro (Vet_Sub P2 P1) (Vet_Sub P3 P1)))
       z_orig     (caddr (trans P1 0 V_Norm))
       pts_o      (Util_Trans_Pts_List (list P1 P2 P3 P4) 0 V_Norm)
       Cg         (Pu_Baricentro2D_LisPu pts_o)
       pts_o      (Util_Translate_Pts_Relative pts_o Cg) min_d 1e99 best_V nil best_F nil)
  (mapcar 'set '(dx dy) (Util_Get_Pts_BBox_Size pts_o))
  (setq ang_pca  (if (> dy dx) (/ pi 2.0) (angle '(0 0) (Vet_Sub (nth 2 pts_o) (nth 0 pts_o))))
       c_pre    (cos ang_pca)
       s_pre    (sin ang_pca)
       pts_o    (Util_Rot_List_2D_CS pts_o c_pre s_pre)
        D_ref    (distance P1 P3)
        D_ref    (if (< D_ref 1e-6) 1.0 D_ref)
        pts_norm (mapcar '(lambda (p) (mapcar '(lambda (c) (/ c D_ref)) p)) pts_o))

  (if Angles_Deg
    (foreach Theta_Deg Angles_Deg
      (if Theta_Deg
        (progn
          (setq c       (cos (* Theta_Deg (/ pi 180.0)))
              s       (sin (* Theta_Deg (/ pi 180.0)))
              pts_rot (Util_Rot_List_2D_CS pts_norm c s))
          (mapcar 'set '(x1 x2 x3 x4)             (Math_Extract_Coord_Pow pts_rot 0 1))
         (mapcar 'set '(y1 y2 y3 y4)             (Math_Extract_Coord_Pow pts_rot 1 1))
         (mapcar 'set '(sq_x1 sq_x2 sq_x3 sq_x4) (Math_Extract_Coord_Pow pts_rot 0 2))
          (setq D_det (Math_Cyclic_Det sq_x1 sq_x2 sq_x3 x1 x2 x3))
          (if (> (abs D_det) tol_m)
            (progn
              (setq ka      (/ (Math_Cyclic_Det y1 y2 y3 x1 x2 x3) D_det)
                  kb      (/ (Math_Cyclic_Det sq_x1 sq_x2 sq_x3 y1 y2 y3) D_det)
                  kc      (/ (+ (* sq_x1 (Math_Diff_Prod x2 y3 x3 y2))
    (* sq_x2 (Math_Diff_Prod x3 y1 x1 y3)) (* sq_x3 (Math_Diff_Prod x1 y2 x2 y1))) D_det)
                  max_err 0.0)
              (foreach p_check pts_rot
                    (setq err (abs (- (+ (Math_Sum_Prod ka (Math_Sqr (car p_check)) kb (car
p_check)) kc)
    (cadr p_check))))
                    (if (> err max_err) (setq max_err err)))
              (if (< max_err 1e-2)
                (progn
```

```
                (setq Xv_n     (/ (- kb) (* 2.0 ka))
                   Yv_n     (- kc (/ (* kb kb) (* 4.0 ka)))
                   p_dist_n (/ 1.0 (* 4.0 ka)))
                (setq Xv (* Xv_n D_ref) Yv (* Yv_n D_ref) p_dist (* p_dist_n D_ref))
                (setq pts_temp (Util_Rot_List_2D_Inv_CS (Util_Rot_List_2D_Inv_CS (list (list
    Xv Yv) (list Xv (+ Yv p_dist))) c s) c_pre s_pre)
    V_wcs (trans (Vet_Sum (list (car (nth 0 pts_temp)) (cadr (nth 0 pts_temp)) z_orig) Cg)
    V_Norm 0) F_wcs (trans (Vet_Sum (list (car (nth 1 pts_temp)) (cadr (nth 1 pts_temp))
    z_orig) Cg) V_Norm 0))
                (setq d (if (= mode "FOCI") (distance target_pt F_wcs)
    (distance target_pt V_wcs)))
                (if (< d min_d) (setq min_d d best_V V_wcs best_F F_wcs))))))))))
  (if (and best_V best_F (< min_d (* D_ref 0.01)))
      (setq l_ents (Draw_ConPar_VeFu_Compl best_V best_F (list P1 P2 P3 P4)))
      (princ "\n[DRAW] Errore: Nessuna parabola corrisponde al bersaglio."))
  l_ents)
;;-------------------------------------------------------------------------------------
(defun CON4P_Geom_Update_MetaMap (pts_list vla_obj p_val s_val map_entry current_map)
  (foreach p pts_list
    (setq current_map (cons (list p vla_obj p_val s_val map_entry) current_map)))
  current_map)
;;-------------------------------------------------------------------------------------
(defun CON4P_Geom_Classify_And_Normalize (K_full g_data D_ref / det f_type V_dir a b)
  (setq det    (- (* 4.0 (nth 0 K_full) (nth 2 K_full)) (* (nth 1 K_full) (nth 1 K_full)))
        f_type (if (> det 0) 'ELLISSE 'IPERBOLE)
        V_dir  (nth 1 g_data)
        a      (* (nth 2 g_data) D_ref)
        b      (* (nth 3 g_data) D_ref))
  (if (and (eq f_type 'ELLISSE) (< a b))
      (list f_type (list (- (cadr V_dir)) (car V_dir) 0.0) b a)
      (list f_type V_dir a b)))
;;-------------------------------------------------------------------------------------
(defun CON4P_Process_Single_VLA_Point (vla_obj param_val pts_ord mode ref_len step_val / center
     conic_data f_type V_dir a b g_data_wcs pts_asse_a pts_asse_b curr_pts c_dist)
  (if (not (vl-catch-all-error-p (setq center (vl-catch-all-apply 'vlax-curve-getPointAtParam (
     list vla_obj param_val)))))
    (if (setq conic_data (CON4P_Helper_Solve_Normalized_Conic center pts_ord ref_len 'CENTERED)
     )
      (progn
        (setq valid_centers_count (1+ valid_centers_count))
        (mapcar 'set '(f_type V_dir a b) conic_data)
        (setq g_data_wcs (list center V_dir a b) curr_pts nil)

        (cond
          ((and (= mode "FOCI")     (eq f_type 'ELLISSE))
```

```
     (setq c_dist      (sqrt (abs (- (* a a) (* b b))))
                                                                curr_pts    (Geom_2Pu_Sim
     m_PuC center V_dir c_dist)
                                                                pts_1       (append curr_
     pts pts_1)))
          ((and (= mode "FOCI")     (eq f_type 'IPERBOLE))
     (setq c_dist      (Pitag_Ip a b)
                                                                curr_pts    (Geom_2Pu_Sim
     m_PuC center V_dir c_dist)
                                                                pts_2       (append curr_
     pts pts_2)))
          ((and (= mode "VERTICES") (eq f_type 'ELLISSE))
     (setq pts_asse_a  (Geom_2Pu_Simm_PuC center V_dir a)
                                                                pts_asse_b  (Geom_2Pu_Sim
     m_PuC center (list (- (cadr V_dir)) (car V_dir) 0.0) b)
                                                                pts_1       (append pts_a
     sse_a pts_1)
                                                                pts_4
                                                           (append pts_asse_b pts_4)
                                                                curr_pts    (append pts_a
     sse_a pts_asse_b)))
          ((and (= mode "VERTICES") (eq f_type 'IPERBOLE))
     (setq curr_pts    (Geom_2Pu_Simm_PuC center V_dir a)
                                                                pts_2       (append curr_
     pts pts_2))))

        (if curr_pts
            (setq pts_map  (cons (Geom_Format_Map_Data curr_pts f_type g_data_wcs) pts_map)
                  meta_map (CON4P_Geom_Update_MetaMap curr_pts vla_obj param_val step_val (car
     pts_map) meta_map)))))))
;;-------------------------------------------------------------------------------------------
(defun CON4P_Elabora_Ramo_Partizionato (obj ramo scan_mod pts_ord mode D_ref / cfg mult_e i seg
     _start seg_end current_scan_mod step_p)
  (setq cfg (CON4P_Get_Global_Config) mult_e (cdr (assoc 'mult_ext cfg)))
  (setq i 0)
  (while (< i (1- (length ramo)))
    (setq seg_start        (car (nth i ramo))
          seg_end          (car (nth (1+ i) ramo))
          current_scan_mod (if (or (= i 0) (= i (- (length ramo) 2))) (* mult_e scan_mod)
     (float scan_mod))
          step_p           (/ (- seg_end seg_start) current_scan_mod))
    (CON4P_Process_Segment_Matrioska obj seg_start seg_end step_p pts_ord mode D_ref)
    (setq i (1+ i))))
;;-------------------------------------------------------------------------------------------
(defun CON4P_Geom_Scan_Parallelogram_Locus (pts_ord mode D_ref /
```

```
                                                                CFG_STEP_BASE CFG_LIMIT_ANG
                                                                CFG_THR_1 CFG_DIV_1 CFG_THR_2
   CFG_DIV_2
                                                                CFG_THR_3 CFG_DIV_3 CFG_THR_4
   CFG_DIV_4
                                                                Cg P1 P2 P3 x1 y1 x2 y2 M11 M
   22 M12
                                                                D_gram alpha beta A0 B0 C0 dA
   dB dC norm0 normD K Qa Qb Qc
                                                                sol_t th1 th2 t3 th3 theta t_
   val A B C K_full g_data
                                                                det f_type V_dir a b pts_1 pt
   s_2 pts_4 pts_map meta_map
                                                                c_dist curr_pts step_base ste
   p_cur valid_centers_count pts_asse_a pts_asse_b tmp dist_th pi_half)

(setq CFG_STEP_BASE  0.001
      CFG_LIMIT_ANG  1.57079
      CFG_THR_1  (* CFG_STEP_BASE 10.0)    CFG_DIV_1    4.0
      CFG_THR_2  (* CFG_STEP_BASE 2.0)     CFG_DIV_2   10.0
      CFG_THR_3  (* CFG_STEP_BASE 0.2)     CFG_DIV_3   25.0
      CFG_THR_4  (* CFG_STEP_BASE 0.02)    CFG_DIV_4  100.0
)

(mapcar 'set '(P1 P2 P3) pts_ord)
(setq Cg (Math_PuntoMedio_2Pu P1 P3))

(mapcar 'set '(x1 y1) (mapcar '(lambda (c) (/ c D_ref)) (Vet_Sub P1 Cg)))
(mapcar 'set '(x2 y2) (mapcar '(lambda (c) (/ c D_ref)) (Vet_Sub P2 Cg)))

(setq M11     (+ (* x1 x1 x1 x1) (* x1 x1 y1 y1) (* y1 y1 y1 y1))
      M22     (+ (* x2 x2 x2 x2) (* x2 x2 y2 y2) (* y2 y2 y2 y2))
      M12     (+ (* x1 x1 x2 x2) (* x1 y1 x2 y2) (* y1 y1 y2 y2))
      D_gram  (- (* M11 M22) (* M12 M12))
      alpha   (/ (- M22 M12) D_gram)
      beta    (/ (- M11 M12) D_gram)
      A0      (+ (* alpha x1 x1) (* beta x2 x2))
      B0      (+ (* alpha x1 y1) (* beta x2 y2))
      C0      (+ (* alpha y1 y1) (* beta y2 y2))
      dA      (* y1 y2)
      dB      (- (+ (* x1 y2) (* x2 y1)))
      dC      (* x1 x2)
      norm0   (sqrt (+ (* A0 A0) (* B0 B0) (* C0 C0)))
      normD   (sqrt (+ (* dA dA) (* dB dB) (* dC dC)))
      K       (if (> normD 1e-9) (/ norm0 normD) 1.0)
```

```
      Qa      (- (* 4.0 dA dC) (* dB dB))
      Qb      (- (+ (* 4.0 A0 dC) (* 4.0 C0 dA)) (* 2.0 B0 dB))
      Qc      (- (* 4.0 A0 C0) (* B0 B0))
      sol_t   (Math_Solve_Quadratic Qa Qb Qc)
      th1     (if (car  sol_t) (atan (car  sol_t) K)                        999.0)
      th2     (if (cadr sol_t) (atan (cadr sol_t) K)                        999.0)
      th3     (if (> (abs (- dA dC)) 1e-9) (atan (/ (- C0 A0) (- dA dC)) K) 999.0))

(setq pts_1    nil
     pts_2    nil
     pts_4    nil
     pts_map  nil
     meta_map nil
     valid_centers_count 0)

(setq theta     (- CFG_LIMIT_ANG))
(setq pi_half   (* pi 0.5))
(setq step_base CFG_STEP_BASE)

(while (<= theta CFG_LIMIT_ANG)

  (setq dist_th (min (abs (- theta th1))
                     (abs (- theta th2))
                     (abs (- theta th3))
                     (abs (- theta pi_half))
                     (abs (+ theta pi_half))))

  (setq step_cur (cond ((< dist_th CFG_THR_4) (/ step_base CFG_DIV_4))
                       ((< dist_th CFG_THR_3) (/ step_base CFG_DIV_3))
                       ((< dist_th CFG_THR_2) (/ step_base CFG_DIV_2))
                       ((< dist_th CFG_THR_1) (/ step_base CFG_DIV_1))
                       (T                     step_base)))

  (setq t_val  (* (/ (sin theta) (cos theta)) K))
  (setq A      (+ A0 (* t_val dA))
        B      (+ B0 (* t_val dB))
        C      (+ C0 (* t_val dC)))
  (setq K_full (list A B C 0.0 0.0 -1.0)
        g_data (Math_Extract_Geom_Data_From_Coeffs K_full '(0.0 0.0 0.0) 1.0))

  (if g_data
      (progn
        (setq det    (- (* 4.0 A C) (* B B))
              f_type (if (> det 0) 'ELLISSE 'IPERBOLE)
              V_dir  (nth 1 g_data)
```

```
          a      (* (nth 2 g_data) D_ref)
          b      (* (nth 3 g_data) D_ref))

     (if (and (eq f_type 'ELLISSE) (< a b))
         (setq V_dir (list (- (cadr V_dir)) (car V_dir) 0.0) tmp a a b b tmp))

     (setq curr_pts nil)
     (cond  ((and (= mode "FOCI") (eq f_type 'ELLISSE))
   (setq c_dist     (sqrt (abs (- (* a a) (* b b))))
                                                                  curr_pts   (Geom_2Pu_S
   imm_PuC Cg V_dir c_dist)
                                                                  pts_1      (append cur
   r_pts pts_1)))
            ((and (= mode "FOCI") (eq f_type 'IPERBOLE))
   (setq c_dist     (Pitag_Ip a b)
                                                                  curr_pts   (Geom_2Pu_S
   imm_PuC Cg V_dir c_dist)
                                                                  pts_2      (append cur
   r_pts pts_2)))
            ((and (= mode "VERTICES") (eq f_type 'ELLISSE))
   (setq pts_asse_a (Geom_2Pu_Simm_PuC Cg V_dir a)
                                                                  pts_asse_b (Geom_2Pu_S
   imm_PuC Cg (list (- (cadr V_dir)) (car V_dir) 0.0) b)
                                                                  pts_1      (append pts
   _asse_a pts_1)
                                                                  pts_4      (append pts
   _asse_b pts_4)
                                                                  curr_pts   (append pts
   _asse_a pts_asse_b)))
            ((and (= mode "VERTICES") (eq f_type 'IPERBOLE))
   (setq curr_pts   (Geom_2Pu_Simm_PuC Cg V_dir a)
                                                                  pts_2      (append cur
   r_pts pts_2))))

     (if curr_pts
         (progn
           (setq pts_map             (cons (Geom_Format_Map_Data curr_pts f_type (list Cg
   V_dir a b)) pts_map))
           (setq meta_map            (CON4P_Geom_Update_MetaMap curr_pts nil theta step_cu
   r (car pts_map) meta_map))
           (setq valid_centers_count (1+ valid_centers_count))))))
  (setq theta (+ theta step_cur)))
(list (reverse pts_1) (reverse pts_2) nil (reverse pts_4) (apply 'append (reverse pts_map))
   valid_centers_count (reverse meta_map)))
;;-------------------------------------------------------------------------------
```

```
(defun CON4P_Draw_Orthogonal_Locus (pts_ord mode Cg / cfg ext_t col_el col_ip P1 P2 P3 P4 M12 M
     23 M34 M41 L1_ext1 L1_ext2 L2_ext1 L2_ext2 old_ce res)
  (setq cfg   (CON4P_Get_Global_Config)
        ext_t (cdr (assoc 'ext_trap cfg))
        old_ce (getvar "CECOLOR"))

  (if (= mode "FOCI")
      (setq col_el (cdr (assoc 'col_ElFu   cfg))
           col_ip (cdr (assoc 'col_IpFu   cfg)))
      (setq col_el (cdr (assoc 'col_ElVeMa cfg))
           col_ip (cdr (assoc 'col_IpVe   cfg))))

  (mapcar 'set '(P1 P2 P3 P4)      pts_ord)
  (mapcar 'set '(M12 M23 M34 M41)  (Math_lisPuntoMedio_lis2Pu (list (list P1 P2) (list P2 P3)
     (list P3 P4) (list P4 P1))))

  (setq L1_ext1 (polar M12 (angle M34 M12) (* (distance M12 M34) ext_t))
        L1_ext2 (polar M34 (angle M12 M34) (* (distance M12 M34) ext_t))
        L2_ext1 (polar M23 (angle M41 M23) (* (distance M23 M41) ext_t))
        L2_ext2 (polar M41 (angle M23 M41) (* (distance M23 M41) ext_t)))

  (setvar "CECOLOR" col_ip) (Draw_Line_2Pu_R M12 M34     *LaCo_Luoghi_geometrici*)
     (setq res (cons (entlast) res))
  (setvar "CECOLOR" col_el) (Draw_Line_2Pu_R M12 L1_ext1 *LaCo_Luoghi_geometrici*)
     (setq res (cons (entlast) res))
  (setvar "CECOLOR" col_el) (Draw_Line_2Pu_R M34 L1_ext2 *LaCo_Luoghi_geometrici*)
     (setq res (cons (entlast) res))
  (setvar "CECOLOR" col_ip) (Draw_Line_2Pu_R M23 M41     *LaCo_Luoghi_geometrici*)
     (setq res (cons (entlast) res))
  (setvar "CECOLOR" col_el) (Draw_Line_2Pu_R M23 L2_ext1 *LaCo_Luoghi_geometrici*)
     (setq res (cons (entlast) res))
  (setvar "CECOLOR" col_el) (Draw_Line_2Pu_R M41 L2_ext2 *LaCo_Luoghi_geometrici*)
     (setq res (cons (entlast) res))
  (setvar "CECOLOR" old_ce)
  res)
;;-------------------------------------------------------------------------------------
(defun CON4P_Solve_Orthogonal_Conic (Cg P_rect P_axis mode f_target_type / v_ax v_perp dist_c d
     x dy dx2 dy2 L2 ax2 ay2 f_type out_v out_a out_b c2 sol s1 s2 P_rel)
  (setq v_ax   (Vet_Uni (Vet_Sub P_axis Cg))
        v_perp (list (- (cadr v_ax)) (car v_ax) 0.0)
        dist_c (distance Cg P_axis)
        P_rel  (Vet_Sub P_rect Cg)
        dx     (abs (Vet_Dot P_rel v_ax))
        dy     (abs (Vet_Dot P_rel v_perp))
        L2     (* dist_c dist_c)
```

```
        dx2    (* dx dx)
        dy2    (* dy dy))
  (cond
    ((= mode "V")
     (if (> L2 dx2)
         (progn
           (setq ax2 L2
                 ay2 (/ (* dy2 ax2) (- ax2 dx2))
                 f_type 'ELLISSE)
           (if (>= ax2 ay2)
               (setq out_v v_ax   out_a (sqrt ax2) out_b (sqrt ay2))
               (setq out_v v_perp out_a (sqrt ay2) out_b (sqrt ax2))))
         (progn
           (setq ax2 L2
                 ay2 (/ (* dy2 ax2) (- dx2 ax2))
                 f_type 'IPERBOLE
                 out_v v_ax out_a (sqrt ax2) out_b (sqrt ay2)))))
    ((= mode "F")
     (setq c2 L2)
     (setq sol (Math_Solve_Quadratic 1.0 (- (+ c2 dx2 dy2)) (* c2 dx2)))
     (if (and (listp sol) (= (length sol) 2))
         (progn
           (setq s1 (car sol) s2 (cadr sol))
           (if (eq f_target_type 'ELLISSE)
               (setq ax2 (max s1 s2) ay2 (abs (- ax2 c2)) f_type 'ELLISSE)
               (setq ax2 (min s1 s2) ay2 (abs (- c2 ax2)) f_type 'IPERBOLE))
           (setq out_v v_ax out_a (sqrt ax2) out_b (sqrt ay2))))))
  (if f_type
      (list f_type out_v out_a out_b)
      nil))
;;-------------------------------------------------------------------------------------------
(defun Math_Solve_Quadratic (a b c / delta res)
  (setq delta (- (* b b) (* 4.0 a c)))
  (setq res
    (cond  ((< delta 0.0) nil)
           ((equal delta 0.0 1e-12) (list (/ (- b) (* 2.0 a))))
           (t (list (/ (+ (- b) (sqrt delta)) (* 2.0 a)) (/ (- (- b) (sqrt delta))
     (* 2.0 a))))))
  res)
;;-------------------------------------------------------------------------------------------
(defun CON4P_Geom_Conic_4Pv_Engine_Local (P1 P2 P3 P4 Cg geo_packet opt_m opt_e last_ents / f_t
     ype V_dir a b V1 F1 p_list e_marker max_proj target_dist dummy_pt)
  (if (and (= opt_m "Sostituisci") last_ents)
      (foreach e (if (listp last_ents) last_ents (list last_ents)) (if (and e (entget e))
     (entdel e))))
```

```
  (setq e_marker (entlast)
        f_type   (nth 1 geo_packet)
        V_dir    (nth 3 geo_packet)
        a        (nth 4 geo_packet)
        b        (nth 5 geo_packet)
        p_list   (list P1 P2 P3 P4))

  (cond ((eq f_type 'ELLISSE)  (setq V1 (Vet_Sum Cg (Vet_Sca V_dir a))
                                     F1 (Vet_Sum Cg (Vet_Sca V_dir (sqrt (abs (- (* a a)
     (* b b)))))))
                               (Draw_ConEll_CeVe_CoVe Cg V1 F1 p_list))

        ((eq f_type 'IPERBOLE) (setq V1 (Vet_Sum Cg (Vet_Sca V_dir a))
                                     F1 (Vet_Sum Cg (Vet_Sca V_dir (sqrt (+ (* a a)
     (* b b))))))
                               (setq max_proj 0.0)
                               (foreach p p_list
                                        (setq max_proj (max max_proj (abs (Vet_Dot (Vet_Sub p C
     g) V_dir)))))
                               (setq target_dist (* max_proj (max 1.0 opt_e)))
                               (setq dummy_pt    (Vet_Sum Cg (Vet_Sca V_dir target_dist)))
                               (Draw_ConIpe_CeVeFu_Compl_Free Cg V1 F1 (append p_list (list dum
     my_pt)))))
  (Safe_Collect_Ents e_marker))
;;-------------------------------------------------------------------------------------
(defun CON4P_Refresh_Orthogonal_Graph (P1 P2 P3 P4 Cg g_data opt_m opt_e last_ents f_target_typ
     e / v_dir a c_pt new_g_data pack)
  (setq v_dir (nth 1 g_data)
        a     (nth 2 g_data)
        c_pt  (Vet_Sum Cg (Vet_Sca v_dir a)))

  (if (setq new_g_data (CON4P_Solve_Orthogonal_Conic Cg P1 c_pt "V" f_target_type))
      (progn
        (setq pack (list nil (nth 0 new_g_data) Cg (nth 1 new_g_data) (nth 2 new_g_data)
     (nth 3 new_g_data)))
        (CON4P_Geom_Conic_4Pv_Engine_Local P1 P2 P3 P4 Cg pack opt_m opt_e last_ents))
      last_ents))
;;-------------------------------------------------------------------------------------
(defun CON4P_Identify_Orthogonal_Target (p5 pts_ord Cg mode opt_m opt_e last_ents f_target_type
     / P1 P2 P3 P4 M12 M34 M23 M41 p_proj1 p_proj2 best_p g_data pack new_ents)

  (mapcar 'set '(P1 P2 P3 P4)      pts_ord)
  (mapcar 'set '(M12 M23 M34 M41)  (Math_lisPuntoMedio_lis2Pu (list (list P1 P2) (list P2 P3)
     (list P3 P4) (list P4 P1))))
```

```
  (setq p_proj1 (Math_Proiezione_1Pu_su_2Pu p5 M12 M34)
        p_proj2 (Math_Proiezione_1Pu_su_2Pu p5 M23 M41)
        best_p (if (< (distance p5 p_proj1) (distance p5 p_proj2)) p_proj1 p_proj2))

  (if (setq g_data (CON4P_Solve_Orthogonal_Conic Cg (car pts_ord) best_p (if (= mode "FOCI")
     "F" "V") f_target_type))
      (progn
        (setq pack (list nil (nth 0 g_data) Cg (nth 1 g_data) (nth 2 g_data) (nth 3 g_data)))
        (setq new_ents (CON4P_Geom_Conic_4Pv_Engine_Local (nth 0 pts_ord) (nth 1 pts_ord)
     (nth 2 pts_ord) (nth 3 pts_ord) Cg pack opt_m opt_e last_ents))
        (append g_data (list new_ents)))))
;;-------------------------------------------------------------------------------------------
(defun CON4P_Helper_Pack_And_Draw (scan_res par_data colors lay / pts_1 pts_2 dummy_nil pts_4 p
     ts_map valid_centers meta_map pts_parabola guide_ents)
  (mapcar 'set '(pts_1 pts_2 dummy_nil pts_4 pts_map valid_centers meta_map) scan_res)
  (setq pts_parabola (if par_data (car par_data) nil))
  (setq guide_ents (CON4P_Draw_Points_With_Colors (list pts_1 pts_2 pts_parabola pts_4)
     colors lay))
  (list guide_ents pts_map meta_map par_data))
;;-------------------------------------------------------------------------------------------
(defun CON4P_Helper_Auto_Matrioska (obj st en div_mod pts_ord mode D_ref / step_p)
  (if (and st en)
      (progn
        (setq step_p (/ (abs (- en st)) (float div_mod)))
        (CON4P_Process_Segment_Matrioska obj st en step_p pts_ord mode D_ref))))
;;-------------------------------------------------------------------------------------------
(defun CON4P_Helper_Solve_Normalized_Conic (center pts_ord D_ref solver_type / center_norm pts_
     ord_norm math_data K_full g_data)
  (if (<= D_ref 0.0) (setq D_ref 1.0))
  (setq center_norm  (mapcar '(lambda (c) (/ c D_ref)) center)
        pts_ord_norm (mapcar '(lambda (p) (mapcar '(lambda (c) (/ c D_ref)) p)) pts_ord))

  (cond ((= solver_type 'CENTERED)
     (if (setq math_data (Geom_ConCen_4PC center_norm pts_ord_norm))
                                        (setq K_full (append (caddr math_data)
     '(0.0 0.0 -1.0)))))
        ((= solver_type 'EQUILATERAL)
     (setq K_full (Math_ConIpeEqu_Coeffs center_norm pts_ord_norm))))

  (if K_full
      (progn (setq g_data (Math_Extract_Geom_Data_From_Coeffs K_full center_norm 1.0))
             (if g_data (CON4P_Geom_Classify_And_Normalize K_full g_data D_ref) nil))))
;;-------------------------------------------------------------------------------------------
(defun CON4P_Geom_Scan_Orthocentric_Circle (pts_ord locus_vla_list scan_mod mode D_ref / eulero
```

```
   _vla pts_1 pts_2 pts_4 pts_map meta_map valid_centers_count circle_data E_cen E_rad P0 P1
   P2 P3 ID1 ID2 ID3 id_list ang_list all_angles a_start a_end stp center conic_data V_dir a
   b g_data_wcs c_dist curr_pts step_val obj_test mapped_param i clean_angles prev_a)
(setq pts_1 nil pts_2 nil pts_4 nil pts_map nil meta_map nil valid_centers_count 0 eulero_vla
   nil)

(foreach obj locus_vla_list
  (setq obj_test (if (= (type obj) 'ENAME) (vlax-ename->vla-object obj) obj))
  (if (not (vl-catch-all-error-p (vl-catch-all-apply 'vlax-curve-getEndParam (list obj_test))
   )) (setq eulero_vla obj_test)))

(if eulero_vla
  (progn
    (setq circle_data (entget (vlax-vla-object->ename eulero_vla))
   E_cen (cdr (assoc 10 circle_data)) E_rad (cdr (assoc 40 circle_data)))
    (setq P0 (nth 0 pts_ord) P1 (nth 1 pts_ord) P2 (nth 2 pts_ord) P3 (nth 3 pts_ord))
    (setq ID1 (inters P0 P1 P2 P3 nil) ID2 (inters P0 P2 P1 P3 nil)
   ID3 (inters P0 P3 P1 P2 nil))
    (setq id_list (vl-remove nil (list ID1 ID2 ID3)))
    (setq ang_list (vl-sort (mapcar '(lambda (p) (angle E_cen p)) id_list) '<))

    (if (>= (length ang_list) 3)
        (progn
          (setq all_angles nil i 0)
          (while (< i 3)
            (setq a_start (nth i ang_list) a_end (if (= i 2) (car ang_list) (nth (1+ i)
   ang_list)))
            (if (< (- a_end a_start) 0.0) (setq a_end (+ a_end (* 2.0 pi))))
            (setq stp (/ (- a_end a_start) (* 4.0 scan_mod)))
            (setq all_angles (append all_angles (CON4P_Get_Matrioska_Angles a_start a_end stp
   )))
            (setq i (1+ i))))
        (progn
          (setq stp (/ (* 2.0 pi) (* 4.0 scan_mod)))
          (setq all_angles (CON4P_Get_Matrioska_Angles 0.0 (* 2.0 pi) stp))))

    (setq all_angles (vl-sort all_angles '<) clean_angles nil prev_a -1e99)
    (foreach a all_angles (if (> (- a prev_a) 1e-9) (setq clean_angles (cons a clean_angles)
   prev_a a)))
    (setq all_angles (reverse clean_angles))

    (setq step_val (/ (* 2.0 pi) (* 4.0 scan_mod)))
    (foreach ang all_angles
      (setq center (polar E_cen ang E_rad))
      (if (setq conic_data (CON4P_Helper_Solve_Normalized_Conic center pts_ord D_ref 'EQUILAT
```

```
    ERAL))
         (progn
           (setq valid_centers_count (1+ valid_centers_count))
           (mapcar 'set '(f_type V_dir a b) conic_data)
           (setq g_data_wcs (list center V_dir a b) curr_pts nil)

           (cond
             ((= mode "FOCI")
              (setq c_dist (Pitag_Ip a b) curr_pts (Geom_2Pu_Simm_PuC center V_dir c_dist)
    pts_2 (append curr_pts pts_2)))
             ((= mode "VERTICES")
              (setq curr_pts (Geom_2Pu_Simm_PuC center V_dir a)
    pts_2 (append curr_pts pts_2))))

           (if curr_pts
               (progn
                 (setq mapped_param (if (>= ang (* 2.0 pi)) (rem ang (* 2.0 pi)) ang))
                 (setq pts_map (cons (Geom_Format_Map_Data curr_pts 'IPERBOLE g_data_wcs)
    pts_map))
                 (setq meta_map (CON4P_Geom_Update_MetaMap curr_pts eulero_vla mapped_param st
    ep_val (car pts_map) meta_map))))
         )
       )
     )
   )
 )
 (list (reverse pts_1) (reverse pts_2) nil (reverse pts_4) (apply 'append (reverse pts_map))
    valid_centers_count (reverse meta_map)))
;;-------------------------------------------------------------------------------------------
(defun Geom_Get_Par_Vertices_List (P1 P2 P3 P4 tipo_f /
                                   Angles_Deg V_Norm z_orig pts_o Cg dx dy ang_pca c_pre s
    _pre c s pts_rot
                                   x1 x2 x3 x4 y1 y2 y3 y4 sq_x1 sq_x2 sq_x3 sq_x4 D ka kb
    kc Xv Yv
                                   X_fin Y_fin V_wcs l_vert p_map processed_angles is_geom
    _twin)
 (setq l_vert nil p_map nil)

 (if (not (wcmatch (vl-symbol-name tipo_f) "CONCAVO*,PARALLELOGRAMMA*"))
   (progn
     (setq Angles_Deg (Geom_Find_Parabolic_Angles P1 P2 P3 P4))

     (if Angles_Deg
       (progn
         (setq V_Norm  (Vet_Uni (Vet_Cro (Vet_Sub P2 P1) (Vet_Sub P3 P1)))
```

```
            z_orig  (caddr (trans P1 0 V_Norm))
            pts_o   (Util_Trans_Pts_List (list P1 P2 P3 P4) 0 V_Norm)
            Cg      (Pu_Baricentro2D_LisPu pts_o)
            pts_o   (Util_Translate_Pts_Relative pts_o Cg))

     (mapcar 'set '(dx dy) (Util_Get_Pts_BBox_Size pts_o))

     (setq ang_pca (if (> dy dx) (/ pi 2.0) (angle '(0 0) (Vet_Sub (nth 2 pts_o)
(nth 0 pts_o))))
            c_pre   (cos ang_pca)
            s_pre   (sin ang_pca)
            pts_o   (Util_Rot_List_2D_CS pts_o c_pre s_pre)
            processed_angles nil)

     (foreach Theta_Deg Angles_Deg
       (setq c (cos (* Theta_Deg (/ pi 180.0)))
             s (sin (* Theta_Deg (/ pi 180.0)))
             pts_rot (Util_Rot_List_2D_CS pts_o c s))

       (mapcar 'set '(x1 x2 x3 x4)             (Math_Extract_Coord_Pow pts_rot 0 1))
       (mapcar 'set '(y1 y2 y3 y4)             (Math_Extract_Coord_Pow pts_rot 1 1))
       (mapcar 'set '(sq_x1 sq_x2 sq_x3 sq_x4) (Math_Extract_Coord_Pow pts_rot 0 2))

       (setq D (Math_Cyclic_Det sq_x1 sq_x2 sq_x3 x1 x2 x3))

       (if (> (abs D) 1e-12)
         (progn
           (setq ka (/ (Math_Cyclic_Det y1 y2 y3 x1 x2 x3) D)
                 kb (/ (Math_Cyclic_Det sq_x1 sq_x2 sq_x3 y1 y2 y3) D)
                 kc (/ (+ (* sq_x1 (Math_Diff_Prod x2 y3 x3 y2))
                          (* sq_x2 (Math_Diff_Prod x3 y1 x1 y3))
                          (* sq_x3 (Math_Diff_Prod x1 y2 x2 y1))) D))
           (setq is_geom_twin (vl-some '(lambda (pa) (< (abs (- (abs (- pa Theta_Deg))
180.0)) 2.0)) processed_angles))
           (if (not is_geom_twin)
               (progn  (setq processed_angles (cons Theta_Deg processed_angles)
                             Xv (/ (- kb) (* 2.0 ka))
                             Yv (- kc (/ (* kb kb) (* 4.0 ka))))
               (mapcar 'set '(X_fin Y_fin)
(car (Util_Rot_List_2D_Inv_CS (Util_Rot_List_2D_Inv_CS (list (list Xv Yv)) c s)
c_pre s_pre)))
               (setq V_wcs  (trans (list (+ X_fin (car Cg)) (+ Y_fin (cadr Cg)) z_orig)
V_Norm 0)
                     l_vert (cons V_wcs l_vert)
                     p_map  (cons (list V_wcs 'PARABOLA (* Theta_Deg (/ pi 180.0)))
```

```
    p_map)))))))))))
  (list (reverse l_vert) p_map))
;;-------------------------------------------------------------------------------------
(defun Geom_Get_Par_Foci_List (P1 P2 P3 P4 tipo_f / Angles_Deg V_Norm z_orig pts_o Cg dx dy ang
    _pca c_pre s_pre c s pts_rot x1 x2 x3 x4 y1 y2 y3 y4 sq_x1 sq_x2 sq_x3 sq_x4 D ka kb kc Xv
    Yv p_dist pts_temp F_wcs l_foci p_map processed_angles is_geom_twin)
  (setq l_foci nil p_map nil)
  (if (not (wcmatch (vl-symbol-name tipo_f) "CONCAVO*,PARALLELOGRAMMA*"))
    (progn
      (setq Angles_Deg (Geom_Find_Parabolic_Angles P1 P2 P3 P4))
      (if Angles_Deg
        (progn
          (setq V_Norm     (Vet_Uni (Vet_Cro (Vet_Sub P2 P1) (Vet_Sub P3 P1)))
                z_orig     (caddr (trans P1 0 V_Norm))
                pts_o      (Util_Trans_Pts_List (list P1 P2 P3 P4) 0 V_Norm)
                Cg         (Pu_Baricentro2D_LisPu pts_o)
                pts_o      (Util_Translate_Pts_Relative pts_o Cg))
          (mapcar 'set '(dx dy) (Util_Get_Pts_BBox_Size pts_o))
          (setq ang_pca (if (> dy dx) (/ pi 2.0) (angle '(0 0) (Vet_Sub (nth 2 pts_o)
    (nth 0 pts_o))))
                c_pre   (cos ang_pca) s_pre (sin ang_pca)
                pts_o   (Util_Rot_List_2D_CS pts_o c_pre s_pre)
                processed_angles nil)
          (foreach Theta_Deg Angles_Deg
            (setq c (cos (* Theta_Deg (/ pi 180.0))) s (sin (* Theta_Deg (/ pi 180.0)))
                  pts_rot (Util_Rot_List_2D_CS pts_o c s))
            (mapcar 'set '(x1 x2 x3 x4)             (Math_Extract_Coord_Pow pts_rot 0 1))
            (mapcar 'set '(y1 y2 y3 y4)             (Math_Extract_Coord_Pow pts_rot 1 1))
            (mapcar 'set '(sq_x1 sq_x2 sq_x3 sq_x4) (Math_Extract_Coord_Pow pts_rot 0 2))
            (setq D (Math_Cyclic_Det sq_x1 sq_x2 sq_x3 x1 x2 x3))
            (if (> (abs D) 1e-12)
              (progn
                (setq ka (/ (Math_Cyclic_Det y1 y2 y3 x1 x2 x3) D)
                      kb (/ (Math_Cyclic_Det sq_x1 sq_x2 sq_x3 y1 y2 y3) D)
                      kc (/ (+ (* sq_x1 (Math_Diff_Prod x2 y3 x3 y2))
                               (* sq_x2 (Math_Diff_Prod x3 y1 x1 y3))
                               (* sq_x3 (Math_Diff_Prod x1 y2 x2 y1))) D))

                (setq is_geom_twin (vl-some '(lambda (pa) (< (abs (- (abs (- pa Theta_Deg))
    180.0)) 2.0)) processed_angles))
                (if (not is_geom_twin)
                  (progn
                    (setq processed_angles (cons Theta_Deg processed_angles)
                          Xv (/ (- kb) (* 2.0 ka)) Yv (- kc (/ (* kb kb) (* 4.0 ka)))
    p_dist (/ 1.0 (* 4.0 ka))
```

```
                        pts_temp (Util_Rot_List_2D_Inv_CS (Util_Rot_List_2D_Inv_CS (list (lis
    t Xv Yv) (list Xv (+ Yv p_dist))) c s) c_pre s_pre)
                        F_wcs (trans (Vet_Sum (list (car (nth 1 pts_temp))
    (cadr (nth 1 pts_temp)) z_orig) Cg) V_Norm 0)
                        l_foci (cons F_wcs l_foci)
                        p_map (cons (list F_wcs 'PARABOLA (* Theta_Deg (/ pi 180.0))) p_map)
    )   )  )  )  )  )  )  )  )  )
  (list (reverse l_foci) p_map))
;;-------------------------------------------------------------------------------------
(defun CON4P_Core_Matrioska_Loop (base_val dir k_start k_step divisors_list L_init first_diviso
    r cfg_iter param_list / L iter k local_st sub_step j)
  (setq L L_init iter 0)
  (while (< iter cfg_iter)
    (setq k k_start)
    (foreach div divisors_list
      (setq local_st (+ base_val (* dir (* k L)))
            sub_step (/ L div)
            j 1)
      (while (<= j (fix div))
        (setq param_list (cons (+ local_st (* j sub_step)) param_list))
        (setq j (1+ j)))
      (setq k (+ k k_step)))
    (setq L (/ L first_divisor) iter (1+ iter)))
  param_list)
;;-------------------------------------------------------------------------------------
(defun CON4P_Core_Matrioska_Engine (st en stp / cfg_iter cfg_divs cfg_len first_divisor divisor
    s_start divisors_end num_steps step_val param_list i clean_list prev_p p)

  (setq cfg_iter 2)
  (setq cfg_divs '(nil 8 8 8 8  7 7 7 7
                   6 6 6 6  5 5 5 5
                   4 4 4 4  3 3 3 3))

  (setq cfg_len        (length cfg_divs)
        first_divisor  (float  cfg_len)
        divisors_start (mapcar 'float (cons first_divisor (cdr cfg_divs)))
        divisors_end   (reverse divisors_start)
        num_steps      (max 1 (fix (/ (- en st) stp)))
        step_val       (/ (- en st) (float num_steps))
        param_list     nil)

  (if (>= num_steps (* 2 cfg_len))     (progn
            (setq param_list (CON4P_Core_Matrioska_Loop st 1 0 1 divisors_start step_val first
    _divisor cfg_iter param_list))
            (setq param_list (cons st param_list))
```

```
            (setq i cfg_len)
            (while (<= i (- num_steps cfg_len))
                   (setq param_list (cons (+ st (* i step_val)) param_list))
                   (setq i (1+ i)))
            (setq param_list (CON4P_Core_Matrioska_Loop en -1 cfg_len -1 divisors_end step_val
    first_divisor cfg_iter param_list))
            (setq param_list (cons en param_list)))
     (progn
            (setq i 0)
            (while (<= i num_steps) (setq param_list (cons (+ st (* i step_val)) param_list))
    (setq i (1+ i)))))

  (setq param_list (vl-sort param_list '<) clean_list nil prev_p -1e99)
  (foreach p param_list
           (if (> (- p prev_p) 1e-9)
               (setq clean_list (cons p clean_list) prev_p p)))
           (reverse clean_list))
;;-------------------------------------------------------------------------------------------
(defun CON4P_Process_Segment_Matrioska (obj st en stp pts_ord mode D_ref / param_list pt_3d p)

  (setq param_list (CON4P_Core_Matrioska_Engine st en stp))
  (foreach p param_list
    (CON4P_Process_Single_VLA_Point obj p pts_ord mode D_ref stp)))
;;-------------------------------------------------------------------------------------------
(defun CON4P_Get_Matrioska_Angles (st en stp)  (CON4P_Core_Matrioska_Engine st en stp))
;;-------------------------------------------------------------------------------------------
(defun CON4P_Geom_Search_Grid_Ell_Param (curve_obj st_param en_param pts_ord D_ref tol /
  CFG_DIVS CFG_ITERS get_ratio pt_catch a b iter step_p i curr_p curr_r max_r best_p new_a new_
    b)

  (princ "\n[DEBUG-GRID] Avvio motore a griglia per CEPC...")
  (setq CFG_DIVS  16.0
        CFG_ITERS 4)
;;-------------------------------------------------------------------------------------------
  (defun get_ratio (param / pt conic_data semi_a semi_b)
    (setq pt_catch (vl-catch-all-apply 'vlax-curve-getPointAtParam (list curve_obj param)))
    (if (not (vl-catch-all-error-p pt_catch))
      (progn
        (setq pt pt_catch)
        (setq conic_data (CON4P_Helper_Solve_Normalized_Conic pt pts_ord D_ref 'CENTERED))
        (if conic_data
          (progn
            (setq semi_a (nth 2 conic_data) semi_b (nth 3 conic_data))
            (if (and (> semi_a 0) (> semi_b 0))
              (/ (min semi_a semi_b) (max semi_a semi_b))
```

```
            -1.0))
         -1.0))
      -1.0)
  )

  (setq a st_param
        b en_param
        iter 0
        best_p a)

  (while (< iter CFG_ITERS)
    (princ (strcat "\n[DEBUG-GRID]   Iterazione " (itoa (1+ iter)) "/" (itoa CFG_ITERS)
     " - Zoom Range: [" (rtos a 2 6) " - " (rtos b 2 6) "]"))
    (setq step_p (/ (- b a) CFG_DIVS)
          i 0
          max_r -2.0)

    (while (<= i CFG_DIVS)
      (setq curr_p (+ a (* i step_p))
            curr_r (get_ratio curr_p))
      (if (> curr_r max_r)
        (setq max_r curr_r
              best_p curr_p))
      (setq i (1+ i)))

    (setq new_a (max st_param (- best_p step_p))
          new_b (min en_param (+ best_p step_p)))
    (setq a new_a
          b new_b
          iter (1+ iter)))

  (princ (strcat "\n[DEBUG-GRID] OK! Parametro Ottimale CEPC = " (rtos best_p 2 6)))
  best_p
)
;;-------------------------------------------------------------------------------------
(defun CON4P_Draw_ConIpe_Equil (pts_ord / tipo class_data sol_data f_type Center V_dir a b V1 c
    F1)
  (vl-load-com)

  (setq class_data (Math_Classify_Forma_Lis4Pu pts_ord)
        tipo       (car class_data))

  (princ (strcat "\n[ON-FLY] Analisi topologia per Iperbole Equilatera: " (vl-symbol-name tipo)
    "..."))
```

```
  (cond
    ((eq tipo 'CONCAVO_ORTOCENTRICO)
     (princ "\n[STOP] Quadrangolo Ortocentrico: esistono infinite iperboli equilatere (locus =
     Cerchio Eulero). Impossibile determinare soluzione univoca.")
     nil)

    ((wcmatch (vl-symbol-name tipo) "QUADRATO*,RETTANGOLO*,ROMBO*,PARALLELOGRAMMA*")
     (princ (strcat "\n[STOP] Topologia " (vl-symbol-name tipo)
     " a simmetria centrale: la conica algebrica diretta non à un'iperbole equilatera."))
     nil)

    ((wcmatch (vl-symbol-name tipo) "DEGENERE*")
     (princ "\n[STOP] Configurazione degenere.")
     nil)

    (T
     (princ " Procedo al calcolo algebrico.")

     (setq sol_data (CON4P_Geom_Solve_Unique_Equilateral_Ipe pts_ord))

     (if sol_data
         (progn
           (mapcar 'set '(f_type Center V_dir a b) sol_data)

           (setq V1 (Vet_Sum Center (Vet_Sca V_dir a)))

           (setq c  (Pitag_Ip a b)
                 F1 (Vet_Sum Center (Vet_Sca V_dir c)))

           (princ "\n[DRAW] Disegno dell'Iperbole Equilatera unica in corso...")

           (Draw_ConIpe_CeVeFu_Compl Center V1 F1 pts_ord)
         )
         (progn
           (princ "\n[ERRORE] Errore imprevisto nel calcolo algebrico (matrice singolare o inst
     abilità numerica).")
           nil
         )
     )
    )
  )
)
;;-------------------------------------------------------------------------------------
(defun CON4P_Draw_ConEll_Special (pts_ord meta_map / cepc_entry g_data Center V_dir a b V1 F1)
  (princ "\n[CEPC] Recupero Ellisse Pseudo-Circolare marchiata dal Radar...")
```

```
  (if (setq cepc_entry (vl-some '(lambda (m) (if (eq (cadr (nth 4 m)) 'CEPC) m nil)) meta_map))
      (progn
        (setq g_data (nth 4 cepc_entry))
        (mapcar 'set '(Center V_dir a b) (cddr g_data))
        (setq V1 (Vet_Sum Center (Vet_Sca V_dir a))
              F1 (Vet_Sum Center (Vet_Sca V_dir (sqrt (abs (- (* a a) (* b b)))))))
        (princ (strcat "\n[DRAW] Tracciamento Ellisse Ottimale (Rapporto b/a=" (rtos (/ (min a
    b) (max a b)) 2 4) ")."))
        (Draw_ConEll_CeVe_CoVe Center V1 F1 pts_ord)
      )
      (progn
        (princ "\n[WARN] Nessun centro marchiato 'CEPC' trovato nella mappa. La topologia corre
    nte potrebbe non prevederlo.")
        nil
      )
  )
)
;;-------------------------------------------------------------------------------------------
(defun CON4P_Get_Global_Config ()
  (list (cons 'scan_base 128)
        (cons 'tol_math 1e-9)
        (cons 'tol_geom 1e-3)
        (cons 'ext_trap 16.0)
        (cons 'ext_conv 2.0)
        (cons 'ext_conc 1.5)
        (cons 'mult_ext 3.0)
        (cons 'micro_step_factor 0.001)
        (cons 'col_ElFu  "14")
        (cons 'col_ElVeMa  "1")
        (cons 'col_ElVeMi "40")
        (cons 'col_IpFu    "5")
        (cons 'col_IpVe  "140")
        (cons 'col_PaFu   "84")
        (cons 'col_PaVe    "3")))
;;-------------------------------------------------------------------------------------------
(defun CON4P_Geom_Solve_Unique_Equilateral_Ipe (pts_ord / Cg D_ref pts_norm K1 K2 A1 C1 A2 C2 s
    um1 sum2 lam_val K_eq Geom_Data max_k det_norm actual_center)
  (setq Cg    (Pu_Baricentro2D_LisPu pts_ord)
        D_ref (distance (nth 0 pts_ord) (nth 2 pts_ord)))
  (if (<= D_ref 0.0) (setq D_ref 1.0))

  (setq pts_norm (mapcar '(lambda (p) (mapcar '(lambda (c) (/ c D_ref)) (Vet_Sub p Cg)))
    pts_ord))

  (setq K1 (Math_Multiply_Lines_To_Conic (Math_Line_General_Eq (nth 0 pts_norm)
```

```
    (nth 1 pts_norm))
                                        (Math_Line_General_Eq (nth 2 pts_norm)
    (nth 3 pts_norm)))
       K2 (Math_Multiply_Lines_To_Conic (Math_Line_General_Eq (nth 0 pts_norm)
    (nth 2 pts_norm))
                                        (Math_Line_General_Eq (nth 1 pts_norm)
    (nth 3 pts_norm))))

  (setq A1 (nth 0 K1) C1 (nth 2 K1) A2 (nth 0 K2) C2 (nth 2 K2) sum1 (+ A1 C1) sum2 (+ A2 C2))

  (cond
    ((and (< (abs sum1) 1e-9) (< (abs sum2) 1e-9)) nil)
    ((< (abs sum2) 1e-9) (setq K_eq K2))
    (T (setq lam_val (- (/ sum1 sum2)) K_eq (Math_Combine_Conics K1 K2 lam_val))))

  (if K_eq
      (progn
        (setq Geom_Data (Math_Extract_Geom_Data_From_Coeffs K_eq Cg D_ref))
        (if Geom_Data
            (progn
              (setq max_k (max (abs (nth 0 K_eq)) (abs (nth 1 K_eq)) (abs (nth 2 K_eq)))
                    det_norm (/ (- (* 4.0 (nth 0 K_eq) (nth 2 K_eq)) (* (nth 1 K_eq)
    (nth 1 K_eq))) (* max_k max_k)))
              (if (< det_norm -1e-6)
                  (progn
                    (setq actual_center (nth 0 Geom_Data))
                    (list actual_center 'IPERBOLE actual_center (nth 1 Geom_Data)
    (nth 2 Geom_Data) (nth 3 Geom_Data))
                  )
                  nil))))))
;;-------------------------------------------------------------------------------------------
(defun CON4P_Helper_UI_Prompt (menu_type tipo state_list / opt_gc opt_gf opt_gv opt_t opt_e opt
    _m opt_f_type)
  (mapcar 'set '(opt_gc opt_gf opt_gv opt_t opt_e opt_m opt_f_type) state_list)
  (cond
    ((eq menu_type 'ORTHO)
     (initget "Fuochi Vertici Estendi Accorcia Modo Uscita")
     (strcat "\n[" (vl-symbol-name tipo) "] "
             (if (= opt_gv "ON") "VERTICI" (strcat "FUOCHI_" (if (eq opt_f_type 'ELLISSE)
    "E" "I")))
             " | E:" (rtos opt_e 2 1)
             " | M:" opt_m
             " | Click o [Fuochi/Vertici/Estendi/Accorcia/Modo/Uscita] <Uscita>: "))

    ((eq menu_type 'SPECIALI)
```

```
     (initget "Ellisse Iperbole Parabola Modo Generiche")
     (strcat "\n[" (vl-symbol-name tipo) "] *** SPECIALI ***"
             " | M:" opt_m
             " | Scegli [Ellisse/Iperbole/Parabola/Modo/Generiche] <Generiche>: "))

    (T
     (initget "Centri Fuochi Vertici Estendi Accorcia Target Speciali Modo Uscita")
     (strcat "\nC:" opt_gc " F:" opt_gf " V:" opt_gv " T:" opt_t
             " | E:" (rtos opt_e 2 1)
             " | M:" opt_m
             " | Punto o [Centri/Fuochi/Vertici/Estendi/Accorcia/Target/Speciali/Modo/Uscita] <
     Uscita>: "))
  )
)
;;-------------------------------------------------------------------------------------
(defun CON4P_Geom_Scan_VLA_Centers (pts_ord locus_vla_list scan_mod mode / D_ref pts_1 pts_2 pt
     s_4 pts_map meta_map valid_centers_count is_convex_generic is_Ell_closed M12 M23 M34 M41 M
     D1 MD2 F1 F2 ID points_list obj_1 obj_2 list_1 list_2 pt p_name pt_proj_1 pt_proj_2 pt_pro
     j_A pt_proj_B d1 d2 ramo_A ramo_B list_A list_B obj_A obj_B ID1 ID2 ID3 p_list pA pB pC en
     dP stP i pt_item step_p current_scan_mod seg_start seg_end param_cepc clean_A clean_B prev
     _p e_temp_line vla_temp_line param_cepc_temp pt_cepc_wcs proj_cepc dist_cepc param_local_c
     epc is_cepc_here raw_E_F raw_F_G obj_E_F obj_F_G pt_E pt_G cepc_packet conic_data f_type V
     _dir a b)

  (setq D_ref (distance (nth 0 pts_ord) (nth 2 pts_ord)))
  (if (<= D_ref 0.0) (setq D_ref 1.0))

  (setq pts_1 nil pts_2 nil pts_4 nil pts_map nil meta_map nil valid_centers_count 0 pt_cepc_wc
     s nil cepc_packet nil)

  (setq is_convex_generic (and (= (length locus_vla_list) 2)
     (= (cdr (assoc 0 (entget (vlax-vla-object->ename (car locus_vla_list))))) "SPLINE")))
  (setq is_Ell_closed (and (= (length locus_vla_list) 1)
     (vlax-curve-isClosed (car locus_vla_list))))

  (cond
    (is_convex_generic
      (princ "\n[DEBUG-SCAN] Inizio blocco Quadrangolo Convesso (Iperbole a rami separati)...")
      (mapcar 'set '(M12 M23 M34 M41 MD1 MD2 F2 ID F1) (apply 'Math_ConicaCentri_9Pu pts_ord))
      (setq points_list (mapcar '(lambda (sym) (cons (vl-symbol-name sym) (eval sym)))
     '(F1 F2 ID M12 M23 M34 M41)))
      (setq obj_1  (car locus_vla_list) obj_2  (cadr locus_vla_list) list_1 nil list_2 nil)

      (foreach item points_list
        (setq p_name (car item) pt (cdr item))
```

```
   (if pt
     (progn
       (setq pt_proj_1 (vlax-curve-getClosestPointTo obj_1 pt)
             pt_proj_2 (vlax-curve-getClosestPointTo obj_2 pt)
             d1        (distance pt pt_proj_1)
             d2        (distance pt pt_proj_2))
       (if (< d1 d2)
           (setq list_1 (cons (list (vlax-curve-getParamAtPoint obj_1 pt_proj_1) p_name)
list_1))
           (setq list_2 (cons (list (vlax-curve-getParamAtPoint obj_2 pt_proj_2) p_name)
list_2))))))

 (if (>= (length list_1) (length list_2))
     (setq list_A list_1 obj_A obj_1 list_B list_2 obj_B obj_2)
     (setq list_A list_2 obj_A obj_2 list_B list_1 obj_B obj_1))

 (foreach pt_item (list (cons "MD1" MD1) (cons "MD2" MD2))
   (setq pt (cdr pt_item))
   (if pt
     (progn
       (setq pt_proj_A (vlax-curve-getClosestPointTo obj_A pt)
             pt_proj_B (vlax-curve-getClosestPointTo obj_B pt))
       (if (< (distance pt pt_proj_B) (distance pt pt_proj_A))
           (setq list_B (cons (list (vlax-curve-getParamAtPoint obj_B pt_proj_B)
(car pt_item)) list_B))))))

 (setq param_cepc (CON4P_Geom_Search_Grid_Ell_Param obj_B (vlax-curve-getStartParam obj_B)
(vlax-curve-getEndParam obj_B) pts_ord D_ref 1e-8))
 (setq pt_cepc_wcs (vlax-curve-getPointAtParam obj_B param_cepc))
 (setq list_B (cons (list param_cepc "CEPC") list_B))

 (setq list_A (vl-sort list_A '(lambda (a b) (< (car a) (car b))))
       list_B (vl-sort list_B '(lambda (a b) (< (car a) (car b)))))

 (setq clean_A nil prev_p -1e99)
 (foreach item list_A
   (if (> (- (car item) prev_p) 1e-7)
       (setq clean_A (cons item clean_A) prev_p (car item))))
 (setq list_A (reverse clean_A))

 (setq clean_B nil prev_p -1e99)
 (foreach item list_B
   (if (> (- (car item) prev_p) 1e-7)
       (setq clean_B (cons item clean_B) prev_p (car item))))
 (setq list_B (reverse clean_B))
```

```
  (if (>= (length list_A) 5)
      (progn
        (setq ramo_A (append (list (list (vlax-curve-getStartParam obj_A) "START"))
 list_A (list (list (vlax-curve-getEndParam obj_A) "END")))
              ramo_B (append (list (list (vlax-curve-getStartParam obj_B) "START"))
 list_B (list (list (vlax-curve-getEndParam obj_B) "END"))))
        (CON4P_Elabora_Ramo_Partizionato obj_A ramo_A scan_mod pts_ord mode D_ref)
        (CON4P_Elabora_Ramo_Partizionato obj_B ramo_B scan_mod pts_ord mode D_ref))
      (progn
        (CON4P_Helper_Auto_Matrioska obj_A (vlax-curve-getStartParam obj_A)
 (vlax-curve-getEndParam obj_A) (* 2.0 scan_mod) pts_ord mode D_ref)
        (CON4P_Helper_Auto_Matrioska obj_B (vlax-curve-getStartParam obj_B)
 (vlax-curve-getEndParam obj_B) (* 2.0 scan_mod) pts_ord mode D_ref))))

(is_Ell_closed
  (setq obj_1 (if (= (type (car locus_vla_list)) 'ENAME)
 (vlax-ename->vla-object (car locus_vla_list)) (car locus_vla_list))
        ID1 (inters (nth 0 pts_ord) (nth 1 pts_ord) (nth 2 pts_ord) (nth 3 pts_ord) nil)
        ID2 (inters (nth 0 pts_ord) (nth 2 pts_ord) (nth 1 pts_ord) (nth 3 pts_ord) nil)
        ID3 (inters (nth 0 pts_ord) (nth 3 pts_ord) (nth 1 pts_ord) (nth 2 pts_ord) nil))

  (setq p_list nil)
  (foreach pt (vl-remove nil (list ID1 ID2 ID3))
    (if (not (vl-catch-all-error-p (setq pt_proj_1 (vlax-curve-getClosestPointTo obj_1 pt))
 ))
        (setq p_list (cons (vlax-curve-getParamAtPoint obj_1 pt_proj_1) p_list))))

  (setq p_list (vl-sort p_list '<))
  (if (= (length p_list) 3)
      (progn
        (setq pA (nth 0 p_list) pB (nth 1 p_list) pC (nth 2 p_list)
              stP (vlax-curve-getStartParam obj_1) endP (vlax-curve-getEndParam obj_1))
        (CON4P_Helper_Auto_Matrioska obj_1 pA pB (* 4.0 scan_mod) pts_ord mode D_ref)
        (CON4P_Helper_Auto_Matrioska obj_1 pB pC (* 4.0 scan_mod) pts_ord mode D_ref)
        (setq step_p (/ (+ (- endP pC) (- pA stP)) (* 4.0 scan_mod)))
        (CON4P_Process_Segment_Matrioska obj_1 pC endP step_p pts_ord mode D_ref)
        (CON4P_Process_Segment_Matrioska obj_1 stP pA  step_p pts_ord mode D_ref))
      (CON4P_Helper_Auto_Matrioska obj_1 (vlax-curve-getStartParam obj_1)
 (vlax-curve-getEndParam obj_1) (* 4.0 scan_mod) pts_ord mode D_ref)))

(T
  (setq i 0)
  (if (>= (length locus_vla_list) 6)
    (progn
```

```
     (setq raw_E_F (nth 4 locus_vla_list)
           raw_F_G (nth 5 locus_vla_list)
           obj_E_F (if (= (type raw_E_F) 'ENAME) (vlax-ename->vla-object raw_E_F) raw_E_F)
           obj_F_G (if (= (type raw_F_G) 'ENAME) (vlax-ename->vla-object raw_F_G) raw_F_G)
           pt_E    (vlax-curve-getStartPoint obj_E_F)
           pt_G    (vlax-curve-getEndPoint obj_F_G))
     (entmake (list '(0 . "LINE") (cons 10 pt_E) (cons 11 pt_G)))
     (setq e_temp_line   (entlast)
           vla_temp_line (vlax-ename->vla-object e_temp_line))
     (setq param_cepc_temp (CON4P_Geom_Search_Grid_Ell_Param vla_temp_line 0.0 (vlax-curve
-getEndParam vla_temp_line) pts_ord D_ref 1e-8))
     (setq pt_cepc_wcs     (vlax-curve-getPointAtParam vla_temp_line param_cepc_temp))
     (entdel e_temp_line)))

 (foreach obj locus_vla_list
   (setq obj_1 (if (= (type obj) 'ENAME) (vlax-ename->vla-object obj) obj))
   (if (not (vl-catch-all-error-p (vl-catch-all-apply 'vlax-curve-getEndParam (list obj_1)
)))
     (progn
       (setq seg_start (vlax-curve-getStartParam obj_1)
             seg_end   (vlax-curve-getEndParam obj_1)
             current_scan_mod (if (or (= i 0) (= i (1- (length locus_vla_list))))
(* 2.0 scan_mod) (float scan_mod)))

       (if (and seg_start seg_end)
         (progn
           (setq is_cepc_here nil)
           (if pt_cepc_wcs
             (progn
               (setq proj_cepc (vlax-curve-getClosestPointTo obj_1 pt_cepc_wcs)
                     dist_cepc (distance pt_cepc_wcs proj_cepc))
               (if (< dist_cepc 1e-5)
                 (progn
                   (setq param_local_cepc (vlax-curve-getParamAtPoint obj_1 proj_cepc)
                         is_cepc_here     (and (> param_local_cepc (+ seg_start 1e-6))
(< param_local_cepc (- seg_end 1e-6))))))))

           (if is_cepc_here
             (progn
               (CON4P_Helper_Auto_Matrioska obj_1 seg_start param_local_cepc current_scan_
mod pts_ord mode D_ref)
               (CON4P_Helper_Auto_Matrioska obj_1 param_local_cepc seg_end current_scan_mo
d pts_ord mode D_ref))
             (CON4P_Helper_Auto_Matrioska obj_1 seg_start seg_end current_scan_mod pts_ord
mode D_ref))))))
```

```
        (setq i (1+ i)))))

  (if pt_cepc_wcs
    (if (setq conic_data (CON4P_Helper_Solve_Normalized_Conic pt_cepc_wcs pts_ord D_ref 'CENTER
     ED))
      (progn
        (mapcar 'set '(f_type V_dir a b) conic_data)
        (if (eq f_type 'ELLISSE)
          (setq cepc_packet (list pt_cepc_wcs 'ELLISSE pt_cepc_wcs V_dir a b))))))
  (list (reverse pts_1) (reverse pts_2) nil (reverse pts_4) (apply 'append (reverse pts_map))
     valid_centers_count (reverse meta_map) cepc_packet))
;;-------------------------------------------------------------------------------------
(defun c:CON4P (/ cfg scan_mod col_foci col_vertices setup_data pts old_os old_pd old_ce tipo p
     ts_ord
                 P1 P2 P3 P4 Cg P_diag ref_pt config_str trap_data e_start_c locus_vla_list gui
     de_ents_c
                 pts_map_f meta_map_f par_data_f guide_ents_f pts_map_v meta_map_v par_data_v g
     uide_ents_v
                 node_solutions opt_gc opt_gf opt_gv opt_t opt_m opt_e loop_main loop_speciali
     p_spec par_idx p5 last_run_ents last_geo
                 t_start t_end elapsed mins secs valid_centers last_node_idx sol_idx run_intera
     ctive_loop e_marker
                 opt_f_type last_g_data raw_scan_f raw_scan_v cepc_data equil_data par_sols geo
     _data)
  (setq cfg          (CON4P_Get_Global_Config)
        scan_mod     (cdr (assoc 'scan_base cfg))
        col_foci     (list (cdr (assoc 'col_ElFu cfg))   (cdr (assoc 'col_IpFu cfg))
     (cdr (assoc 'col_PaFu cfg)))
        col_vertices (list (cdr (assoc 'col_ElVeMa cfg)) (cdr (assoc 'col_IpVe cfg))
     (cdr (assoc 'col_PaVe cfg))   (cdr (assoc 'col_ElVeMi cfg))))

  (if (setq setup_data (CON4P_Geom_4P_Master_Setup "LABORATORIO 4P - UNIFICATO DEFINITIVO v.18.
     4" 0))
    (progn
      (setq t_start (* 86400.0 (getvar "DATE")) run_interactive_loop nil)
      (mapcar 'set '(pts old_os old_pd old_ce tipo pts_ord P1 P2 P3 P4 Cg P_diag ref_pt)
     setup_data)

      (cond
        ((wcmatch (vl-symbol-name tipo) "DEGENERE*")
     (princ "\n[!] Configurazione Degenere intercettata. L'analisi à stata interrotta in attesa
     di implementazione."))

        ((member tipo '(QUADRATO RETTANGOLO))
     (princ "\n[CALCOLO] Motore Ortogonale attivato per Rettangolo/Quadrato...")
```

```
    (setq opt_gf "ON" opt_gv "OFF" opt_f_type 'ELLISSE opt_m "Sostituisci" opt_e 1.0 loop_
main T last_run_ents nil last_g_data nil)
    (setq guide_ents_c (CON4P_Draw_Orthogonal_Locus pts_ord "FOCI" Cg))
    (while loop_main
      (setvar "OSMODE" 0)
      (setq p5 (getpoint (CON4P_Helper_UI_Prompt 'ORTHO tipo (list nil nil opt_gv nil opt_
e opt_m opt_f_type))))
      (cond
        ((or (null p5) (= p5 "Uscita")) (setq loop_main nil))
        ((= p5 "Fuochi")
         (if (= opt_gf "ON") (setq opt_f_type (if (eq opt_f_type 'ELLISSE)
'IPERBOLE 'ELLISSE)) (setq opt_gf "ON" opt_gv "OFF" opt_f_type 'ELLISSE))
         (if guide_ents_c (foreach e guide_ents_c (if (and e (entget e)) (entdel e))))
         (setq guide_ents_c (CON4P_Draw_Orthogonal_Locus pts_ord "FOCI" Cg))
         (if last_g_data (setq last_run_ents (CON4P_Refresh_Orthogonal_Graph P1 P2 P3 P4 C
g last_g_data opt_m opt_e last_run_ents opt_f_type))))
        ((= p5 "Vertici")
         (setq opt_gv "ON" opt_gf "OFF")
         (if guide_ents_c (foreach e guide_ents_c (if (and e (entget e)) (entdel e))))
         (setq guide_ents_c (CON4P_Draw_Orthogonal_Locus pts_ord "VERTICES" Cg)))
        ((or (= p5 "Estendi") (= p5 "Accorcia"))
         (setq opt_e (if (= p5 "Estendi") (+ opt_e 0.1) (max 1.0 (- opt_e 0.1))))
         (if last_g_data (setq last_run_ents (CON4P_Refresh_Orthogonal_Graph P1 P2 P3 P4 C
g last_g_data opt_m opt_e last_run_ents opt_f_type))))
        ((= p5 "Modo") (setq opt_m (if (= opt_m "Sostituisci") "Aggiungi" "Sostituisci")))
        ((listp p5)
         (setq last_g_data (CON4P_Identify_Orthogonal_Target p5 pts_ord Cg (if (= opt_gf "
ON") "FOCI" "VERTICES") opt_m opt_e last_run_ents opt_f_type))
         (if last_g_data (setq last_run_ents (last last_g_data))))))
    (if guide_ents_c (foreach e guide_ents_c (if (and e (entget e)) (entdel e)))))

   ((member tipo '(ROMBO PARALLELOGRAMMA))
     (princ "\n[CALCOLO] Motore Discreto Parallelogramma: Generazione Adattiva in corso...
")
     (setq e_start_c      (entlast)
locus_vla_list (CON4P_Geom_Draw_Locus_And_Capture pts_ord (list tipo pts_ord))
guide_ents_c (CON4P_Local_Collect e_start_c))
     (mapcar 'set '(guide_ents_f pts_map_f meta_map_f par_data_f)
(CON4P_Helper_Pack_And_Draw (CON4P_Geom_Scan_Parallelogram_Locus pts_ord "FOCI" (distance
P1 P3)) nil col_foci *LaCo_Luoghi_geometrici*))
     (setq node_solutions (Geom_Precalc_Node_Conics P1 P2 P3 P4))
     (mapcar 'set '(guide_ents_v pts_map_v meta_map_v par_data_v)
(CON4P_Helper_Pack_And_Draw (CON4P_Geom_Scan_Parallelogram_Locus pts_ord "VERTICES" (dista
nce P1 P3)) nil col_vertices *LaCo_Luoghi_geometrici*))
     (setq cepc_data nil equil_data nil par_sols (cadr par_data_v)
```

```
run_interactive_loop T))

   ((eq tipo 'CONCAVO_ORTOCENTRICO)
     (princ "\n[CALCOLO] Motore Ortocentrico: Estrazione diretta Iperboli Equilatere in co
rso...")
     (setq e_start_c      (entlast)
locus_vla_list (CON4P_Geom_Draw_Locus_And_Capture pts_ord (list tipo pts_ord))
guide_ents_c (CON4P_Local_Collect e_start_c))
     (mapcar 'set '(guide_ents_f pts_map_f meta_map_f par_data_f)
(CON4P_Helper_Pack_And_Draw (CON4P_Geom_Scan_Orthocentric_Circle pts_ord locus_vla_list sc
an_mod "FOCI" (distance P1 P3)) nil col_foci *LaCo_Luoghi_geometrici*))
     (setq node_solutions (Geom_Precalc_Node_Conics P1 P2 P3 P4))
     (mapcar 'set '(guide_ents_v pts_map_v meta_map_v par_data_v)
(CON4P_Helper_Pack_And_Draw (CON4P_Geom_Scan_Orthocentric_Circle pts_ord locus_vla_list sc
an_mod "VERTICES" (distance P1 P3)) nil col_vertices *LaCo_Luoghi_geometrici*))
     (setq cepc_data nil equil_data nil par_sols (cadr par_data_v)
run_interactive_loop T))

   ((eq tipo 'CONCAVO_GENERICO)
     (princ "\n[CALCOLO] Motore Unificato (Concavo Generico): Generazione Impalcatura e Pa
rtizionatore in corso...")
     (setq e_start_c      (entlast)
locus_vla_list (CON4P_Geom_Draw_Locus_And_Capture pts_ord (list tipo pts_ord))
guide_ents_c (CON4P_Local_Collect e_start_c))
     (setq raw_scan_f     (CON4P_Geom_Scan_VLA_Centers pts_ord locus_vla_list scan_mod "FO
CI")
           raw_scan_v     (CON4P_Geom_Scan_VLA_Centers pts_ord locus_vla_list scan_mod "VE
RTICES")
           par_data_f     (Geom_Get_Par_Foci_List      P1 P2 P3 P4 tipo)
           par_data_v     (Geom_Get_Par_Vertices_List P1 P2 P3 P4 tipo)
           node_solutions (Geom_Precalc_Node_Conics        P1 P2 P3 P4)
           cepc_data      (nth 7 raw_scan_v)
           equil_data     (CON4P_Geom_Solve_Unique_Equilateral_Ipe pts_ord)
           par_sols       (cadr par_data_v))
     (mapcar 'set '(guide_ents_f pts_map_f meta_map_f par_data_f)
(CON4P_Helper_Pack_And_Draw raw_scan_f par_data_f col_foci     *LaCo_Luoghi_geometrici*))
     (mapcar 'set '(guide_ents_v pts_map_v meta_map_v par_data_v)
(CON4P_Helper_Pack_And_Draw raw_scan_v par_data_v col_vertices *LaCo_Luoghi_geometrici*))
     (setq run_interactive_loop T))

   ((member tipo '(QUADRANGOLO_CICLICO QUADRANGOLO_GENERICO))
     (princ "\n[CALCOLO] Motore Unificato (Convessi): Generazione Impalcatura e Partiziona
tore in corso...")
     (setq e_start_c      (entlast)
locus_vla_list (CON4P_Geom_Draw_Locus_And_Capture pts_ord (list tipo pts_ord))
```

```
guide_ents_c (CON4P_Local_Collect e_start_c))
     (setq raw_scan_f     (CON4P_Geom_Scan_VLA_Centers pts_ord locus_vla_list scan_mod "FO
CI")
           raw_scan_v     (CON4P_Geom_Scan_VLA_Centers pts_ord locus_vla_list scan_mod "VE
RTICES")
           par_data_f     (Geom_Get_Par_Foci_List      P1 P2 P3 P4 tipo)
           par_data_v     (Geom_Get_Par_Vertices_List P1 P2 P3 P4 tipo)
           node_solutions (Geom_Precalc_Node_Conics P1 P2 P3 P4)
           cepc_data      (nth 7 raw_scan_v)
           equil_data     (CON4P_Geom_Solve_Unique_Equilateral_Ipe pts_ord)
           par_sols       (cadr par_data_v))
     (mapcar 'set '(guide_ents_f pts_map_f meta_map_f par_data_f)
(CON4P_Helper_Pack_And_Draw raw_scan_f par_data_f col_foci     *LaCo_Luoghi_geometrici*))
     (mapcar 'set '(guide_ents_v pts_map_v meta_map_v par_data_v)
(CON4P_Helper_Pack_And_Draw raw_scan_v par_data_v col_vertices *LaCo_Luoghi_geometrici*))
     (setq run_interactive_loop T))

   ((member tipo '(TRAPEZIO_ISOSCELE TRAPEZIO_SCALENO))
     (princ "\n[CALCOLO] Motore Unificato (Trapezi): Generazione Impalcatura e Partizionat
ore in corso...")
     (setq e_start_c      (entlast)
locus_vla_list (CON4P_Geom_Draw_Locus_And_Capture pts_ord (list tipo pts_ord))
           trap_data      (CON4P_Geom_Format_Trapezoid_Locus P1 P2 P3 P4 locus_vla_list)
           locus_vla_list (car trap_data) guide_ents_c (CON4P_Local_Collect e_start_c))
     (setq raw_scan_f     (CON4P_Geom_Scan_VLA_Centers pts_ord locus_vla_list scan_mod "FO
CI")
           raw_scan_v     (CON4P_Geom_Scan_VLA_Centers pts_ord locus_vla_list scan_mod "VE
RTICES")
           par_data_f     (Geom_Get_Par_Foci_List      P1 P2 P3 P4 tipo)
           par_data_v     (Geom_Get_Par_Vertices_List P1 P2 P3 P4 tipo)
           node_solutions (Geom_Precalc_Node_Conics P1 P2 P3 P4)
           cepc_data      (nth 7 raw_scan_v)
           equil_data     (CON4P_Geom_Solve_Unique_Equilateral_Ipe pts_ord)
           par_sols       (cadr par_data_v))
     (mapcar 'set '(guide_ents_f pts_map_f meta_map_f par_data_f)
(CON4P_Helper_Pack_And_Draw raw_scan_f par_data_f col_foci     *LaCo_Luoghi_geometrici*))
     (mapcar 'set '(guide_ents_v pts_map_v meta_map_v par_data_v)
(CON4P_Helper_Pack_And_Draw raw_scan_v par_data_v col_vertices *LaCo_Luoghi_geometrici*))
     (setq run_interactive_loop T))

   (T (princ "\n[!] Errore critico: Topologia non riconosciuta dal Master Unificato.")))

 (if run_interactive_loop
   (progn
     (if guide_ents_f (Util_Make_Anonymous_Group guide_ents_f "LUSNAP_Fuochi"))
```

```
     (if guide_ents_v (Util_Make_Anonymous_Group guide_ents_v "LUSNAP_Vertici"))

     (setq t_end (* 86400.0 (getvar "DATE")) elapsed (- t_end t_start)
mins (fix (/ elapsed 60)) secs (fix (rem elapsed 60))
config_str (CON4P_Get_Config_String tipo))
     (princ (strcat "\n[" config_str "] Fine Calcoli: " (itoa mins) "m " (itoa secs) "s"))
     (setq opt_gc "ON" opt_gf "ON" opt_gv "ON" opt_t "Luogo" opt_m "Sostituisci" opt_e 1.0
last_run_ents nil last_geo nil loop_main T last_node_idx -1 sol_idx 0)

     (while loop_main
       (setvar "OSMODE" 0)
       (setq p5 (getpoint (CON4P_Helper_UI_Prompt 'MAIN tipo (list opt_gc opt_gf opt_gv op
t_t opt_e opt_m nil))))
       (cond
         ((null p5)         (setq loop_main nil))
         ((= p5 "Uscita")   (setq loop_main nil))
         ((= p5 "Centri")   (setq opt_gc (if (= opt_gc "ON") "OFF" "ON"))
(Util_Update_Guides_Vis opt_gc guide_ents_c))
         ((= p5 "Fuochi")   (setq opt_gf (if (= opt_gf "ON") "OFF" "ON"))
(Util_Update_Guides_Vis opt_gf guide_ents_f))
         ((= p5 "Vertici")  (setq opt_gv (if (= opt_gv "ON") "OFF" "ON"))
(Util_Update_Guides_Vis opt_gv guide_ents_v))
         ((= p5 "Target")   (setq opt_t  (if (= opt_t "Luogo") "Nodo" "Luogo")))
         ((= p5 "Modo")     (setq opt_m  (if (= opt_m "Sostituisci")
"Aggiungi" "Sostituisci")))
         ((= p5 "Estendi")  (setq opt_e (+ opt_e 0.5))
(if (and last_geo (= opt_m "Sostituisci"))
(setq last_run_ents (CON4P_Refresh_Last_Conic last_geo opt_m opt_e last_run_ents P1 P2 P3
P4 pts_ord Cg pts_map_f))))
         ((= p5 "Accorcia") (setq opt_e (max 1.0 (- opt_e 0.5)))
(if (and last_geo (= opt_m "Sostituisci"))
(setq last_run_ents (CON4P_Refresh_Last_Conic last_geo opt_m opt_e last_run_ents P1 P2 P3
P4 pts_ord Cg pts_map_f))))

         ((= p5 "Speciali")
          (setq loop_speciali T par_idx 0)
          (while loop_speciali
            (setq p_spec (getkword (CON4P_Helper_UI_Prompt 'SPECIALI tipo (list nil nil ni
l nil nil opt_m nil))))
            (cond
              ((or (null p_spec) (= p_spec "Generiche"))
               (setq loop_speciali nil))

              ((= p_spec "Modo")
               (setq opt_m (if (= opt_m "Sostituisci") "Aggiungi" "Sostituisci")))
```

```
               ((= p_spec "Ellisse")
                (if cepc_data
                    (progn
                      (setq last_run_ents (CON4P_Execute_Conic_Action cepc_data "VERTICES"
   P1 P2 P3 P4 Cg opt_m opt_e last_run_ents))
                      (setq last_geo cepc_data)
                      (princ "\n[DRAW] Ellisse Pseudo-Circolare tracciata con successo."))
                    (princ "\n[WARN] Nessuna Ellisse Speciale disponibile per questa config
   urazione.")))

               ((= p_spec "Iperbole")
                (if equil_data
                    (progn
                      (setq last_run_ents (CON4P_Execute_Conic_Action equil_data "VERTICES"
   P1 P2 P3 P4 Cg opt_m opt_e last_run_ents))
                      (setq last_geo equil_data)
                      (princ "\n[DRAW] Iperbole Equilatera tracciata con successo."))
                    (princ "\n[WARN] Nessuna Iperbole Equilatera disponibile per questa con
   figurazione.")))

               ((= p_spec "Parabola")
                (if par_sols
                    (progn
                      (setq geo_data (nth par_idx par_sols))
                      (setq last_run_ents (CON4P_Execute_Conic_Action geo_data "VERTICES" P
   1 P2 P3 P4 Cg opt_m opt_e last_run_ents))
                      (setq last_geo geo_data)
                      (setq par_idx (rem (1+ par_idx) (length par_sols))))
                    (princ "\n[WARN] Nessuna parabola disponibile per questa configurazione
   .")))
             )))

          ((and (listp p5) (= opt_t "Nodo"))
           (mapcar 'set '(sol_idx last_node_idx last_geo last_run_ents)
   (CON4P_Identify_Node_Target p5 P1 P2 P3 P4 node_solutions last_node_idx sol_idx opt_m opt_
   e last_run_ents Cg)))
          ((and (listp p5) (= opt_t "Luogo"))
           (mapcar 'set '(last_geo last_run_ents)
   (CON4P_Identify_Locus_Target p5 opt_gc opt_gf opt_gv locus_vla_list par_data_f meta_map_f
   par_data_v meta_map_v pts_ord P1 P2 P3 P4 Cg opt_m opt_e last_run_ents)))))
      (CON4P_Geom_Master_Exit_Handler opt_gc opt_gv guide_ents_c (append guide_ents_f guide
   _ents_v) "LUSNAP Unificato"))))
  (princ "\nInput annullato."))
(Restore_Conic_App old_os old_pd old_ce))
```

```
)
;;-------------------------------------------------------------------------------------
;;-------------------------------------------------------------------------------------

;;-------------------------------------------------------------------------------------
;;-------------------------------------------------------------------------------------
(progn
;;-------------------------------------------------------------------------------------
(defun Draw_Guide_Parabolas_4P (pts / tipo pOrd P1 P2 P3 P4 d_max ext1 ext2 ext3 ext4)
  (mapcar 'set '(tipo pOrd) (apply 'Math_Classify_Forma_4Pu pts))

  (mapcar 'set '(P1 P2 P3 P4) pOrd)

  (setq d_max (* (max (distance P1 P2) (distance P2 P3) (distance P3 P4) (distance P4 P1))
     20.0))

  (cond
    ((member tipo '(QUADRATO RETTANGOLO ROMBO PARALLELOGRAMMA))
     (princ "\n[GUIDA] Lati paralleli: i confini parabolici degenerano in 2 coppie di rette.")
     (setq ext1 (Geom_2Pu_Simm_PuC (Vet_Mid P1 P2) (Vet_Uni (Vet_Sub P2 P1)) d_max)
           ext2 (Geom_2Pu_Simm_PuC (Vet_Mid P3 P4) (Vet_Uni (Vet_Sub P4 P3)) d_max)
           ext3 (Geom_2Pu_Simm_PuC (Vet_Mid P2 P3) (Vet_Uni (Vet_Sub P3 P2)) d_max)
           ext4 (Geom_2Pu_Simm_PuC (Vet_Mid P4 P1) (Vet_Uni (Vet_Sub P1 P4)) d_max))
     (Draw_Line_2Pu_R (car ext1) (cadr ext1) *LaCo_Luoghi_geometrici*)
     (Draw_Line_2Pu_R (car ext2) (cadr ext2) *LaCo_Luoghi_geometrici*)
     (Draw_Line_2Pu_R (car ext3) (cadr ext3) *LaCo_Luoghi_geometrici*)
     (Draw_Line_2Pu_R (car ext4) (cadr ext4) *LaCo_Luoghi_geometrici*))

    ((member tipo '(TRAPEZIO_ISOSCELE TRAPEZIO_SCALENO))
     (princ "\n[GUIDA] Trapezio: 1 parabola vera + 1 degenere (coppia rette parallele).")
     (Draw_ConPar_4P_Light P1 P3 P2 P4)
     (if (P_Paral_2Re P1 P2 P3 P4)
         (setq ext1 (Geom_2Pu_Simm_PuC (Vet_Mid P1 P2) (Vet_Uni (Vet_Sub P2 P1)) d_max)
               ext2 (Geom_2Pu_Simm_PuC (Vet_Mid P3 P4) (Vet_Uni (Vet_Sub P4 P3)) d_max))
         (setq ext1 (Geom_2Pu_Simm_PuC (Vet_Mid P2 P3) (Vet_Uni (Vet_Sub P3 P2)) d_max)
               ext2 (Geom_2Pu_Simm_PuC (Vet_Mid P4 P1) (Vet_Uni (Vet_Sub P1 P4)) d_max)))
     (Draw_Line_2Pu_R (car ext1) (cadr ext1) *LaCo_Luoghi_geometrici*)
     (Draw_Line_2Pu_R (car ext2) (cadr ext2) *LaCo_Luoghi_geometrici*))

    ((wcmatch (vl-symbol-name tipo) "CONCAVO*")
     (princ "\n[GUIDA] Configurazione concava: dominanza iperbolica totale."))
    ((wcmatch (vl-symbol-name tipo) "DEGENERE*")     (princ "\n[GUIDA] Punti degeneri."))
    (T                                               (princ "\n[GUIDA] Tracciamento parabole di
     confine...")
                                                     (Draw_ConPar_4P_Light P1 P3 P2 P4)))
```

```
  (princ)
)
;;-------------------------------------------------------------------------------------
(defun Geom_Draw_Quadrangolo_Smart (pts lay / p1 p2 p3 p4 pts_ord e_mark)
  (if (= (length pts) 4)
    (progn
      (setq e_mark (entlast))
      (mapcar 'set '(p1 p2 p3 p4) pts)
      (cond
        ((P_Conve_4Pu p1 p2 p3 p4)
         (if (setq pts_ord (Geom_Ordina_CCW_4Pu p1 p2 p3 p4))
             (Draw_Quadrangolo_Convesso_Lis4Pu pts_ord lay)))
        ((P_Conca_4Pu p1 p2 p3 p4)
         (if (setq pts_ord (Geom_Ordina_Concavo_4Pu p1 p2 p3 p4))
             (Draw_Quadrangolo_Concavo_Lis4Pu pts_ord lay))))
      (Util_Collect_Ents e_mark))))
;;-------------------------------------------------------------------------------------
(defun Geom_5P_Update_Guides (pts opt_g old_guides / e_start e_curr new_guides)
  (if old_guides (foreach e old_guides (if (entget e) (entdel e))))
  (setq new_guides nil)
  (if (and (= opt_g "ON") pts)
    (progn
      (setq e_start (entlast))
      (Draw_Guide_Parabolas_4P pts)
      (setq e_curr (entnext e_start))
      (while e_curr
        (setq new_guides (cons e_curr new_guides)
              e_curr (entnext e_curr)))
      (princ "\n[INFO] Guide paraboliche aggiornate.")))
  new_guides)
;;-------------------------------------------------------------------------------------
(defun c:con5p (/ pts old_os old_pd old_ce opt_g opt_m guide_ents quad_ents last_run_ents loop_
     main p5 pts_ext e_marker)
  (mapcar 'set '(pts old_os old_pd old_ce)
     (Init_Conic_App "CONICA 5 PUNTI (MODO LABORATORIO)" 4 "\nSeleziona i primi 4 punti base: "
     32))
  (if pts
    (progn
      (setvar "OSMODE" 0) (setvar "CMDECHO" 0)

      (setq quad_ents (Geom_Draw_Quadrangolo_Smart pts *LaCo_Poligoni*))
      (setq opt_g "ON" opt_m "Sostituisci" guide_ents nil last_run_ents nil loop_main T)
      (setq guide_ents (Geom_5P_Update_Guides pts opt_g guide_ents))

      (while loop_main
```

```
      (setvar "OSMODE" (if old_os old_os 0))
      (initget "Guide Modo Fine")
      (setq p5 (getpoint (strcat "\nG:" opt_g " | M:" opt_m " | 5à punto o [Guide/Modo/Fine]
    <Fine>: ")))
      (setvar "OSMODE" 0)

      (cond  ((or (null p5) (= p5 "Fine")) (setq loop_main nil))
             ((= p5 "Guide")               (setq opt_g (if (= opt_g "ON") "OFF" "ON"))
                                           (setq guide_ents (Geom_5P_Update_Guides pts opt_g
    guide_ents)))
             ((= p5 "Modo")                (setq opt_m (if (= opt_m "Sostituisci")
    "Aggiungi" "Sostituisci"))
                                           (if (= opt_m "Aggiungi")
    (setq last_run_ents nil)))
             ((listp p5)                   (if (and (= opt_m "Sostituisci") last_run_ents)
                                           (foreach e last_run_ents (if (entget e)
    (entdel e))))
                                           (setq e_marker (entlast))
                                           (Draw_Point_R p5 *LaCo_Punti_scelti*)
                                           (setq pts_ext (Util_Scale_Pts_From_Centroid pts 5.
    0))
                                           (Draw_Con5P_Nor_Compl (append pts (list p5))
    *LaCo_Tracciati* *LaCo_Costruzioni* pts_ext)
                                           (if (= opt_m "Sostituisci")
                                               (setq last_run_ents (Util_Collect_Ents e_marke
    r))
                                               (setq last_run_ents nil))
                                           (princ (strcat "\n[SYSTEM] Conica generata in moda
    lità " opt_m "."))))))
       (if guide_ents (foreach e guide_ents (if (entget e) (entdel e))))
       (princ "\nComando terminato.")))
  (Restore_Conic_App old_os old_pd old_ce))
;;-------------------------------------------------------------------------------------------
(defun c:con5pRea (/ pts old_os old_pd old_ce opt_g guide_ents loop_main pt_old pt_new anchors
    pts_ext e_marker current_conic_ents quad_ents min_d d pt_input)
  (mapcar 'set '(pts old_os old_pd old_ce)
    (Init_Conic_App "EDITING REATTIVO 5 PUNTI" 4 "\nSeleziona i 4 punti base temporanei: " 32)
    )

  (if (and pts (= (length pts) 4))
    (progn
      (setvar "OSMODE" 0) (setvar "CMDECHO" 0)
      (setq opt_g "ON" guide_ents nil loop_main T anchors pts)

      (setq quad_ents (Geom_Draw_Quadrangolo_Smart anchors *LaCo_Poligoni*))
```

```
 (setq guide_ents (Geom_5P_Update_Guides anchors opt_g guide_ents))

 (setq pt_new nil)
 (while (and loop_main (null pt_new))
        (setvar "OSMODE" (if old_os old_os 0))
        (initget "Guide Fine")
        (setvar "ERRNO" 0)
        (setq pt_new (getpoint (strcat "\nG:" opt_g " | Specifica 5à punto o [Guide/Fine]
<Fine>: ")))
        (setvar "OSMODE" 0)
        (cond  ((or (= (getvar "ERRNO") 52) (null pt_new) (= pt_new "Fine"))
(setq loop_main nil pt_new nil))
               ((= pt_new "Guide")
(setq opt_g      (if (= opt_g "ON") "OFF" "ON") pt_new nil)
                                                                        (setq guide_e
nts (Geom_5P_Update_Guides anchors opt_g guide_ents)))
               ((listp pt_new)
(setq pts        (append pts (list pt_new)))
                                                                        (Draw_Point_R
pt_new *LaCo_Punti_scelti*))
               (t                                                       (princ "\n[WA
RN] Specifica la posizione del 5à punto.") (setq pt_new nil))))

 (if (and loop_main (listp pt_new))
   (progn
     (setq e_marker (entlast))
     (setq pts_ext (Util_Scale_Pts_From_Centroid pts 5.0))
     (Draw_Con5P_Nor_Compl pts *LaCo_Tracciati* *LaCo_Costruzioni* pts_ext)
     (setq current_conic_ents (Util_Collect_Ents e_marker))

     (while loop_main
       (setvar "ERRNO" 0)
       (setvar "OSMODE" 8)
       (initget "Guide Fine")
       (setq pt_input (getpoint (strcat "\nG:" opt_g " | Seleziona vertice da muovere o [G
uide/Fine] <Fine>: ")))
       (setvar "OSMODE" 0)

       (cond
         ((or (= (getvar "ERRNO") 52) (null pt_input) (= pt_input "Fine"))
          (setq loop_main nil))

         ((= pt_input "Guide")
          (setq opt_g (if (= opt_g "ON") "OFF" "ON"))
          (setq guide_ents (Geom_5P_Update_Guides anchors opt_g guide_ents)))
```

```
          ((listp pt_input)
           (setq pt_old nil min_d 1e99)
           (foreach p pts (if (< (setq d (distance pt_input p)) min_d)
    (setq min_d d pt_old p)))

           (setq anchors (vl-remove-if '(lambda (p) (< (distance p pt_old) 1e-6)) pts))

           (if quad_ents (Util_Del_Ents_List quad_ents))
           (setq quad_ents (Geom_Draw_Quadrangolo_Smart anchors *LaCo_Poligoni*))
           (setq guide_ents (Geom_5P_Update_Guides anchors opt_g guide_ents))

           (setvar "OSMODE" (if old_os old_os 0))
           (setq pt_new (getpoint pt_old "\nNuova posizione: "))

           (if pt_new
             (progn
               (if current_conic_ents (foreach e current_conic_ents (if (entget e)
    (entdel e))))
               (Util_Del_Ents_At_Pt pt_old *LaCo_Punti_scelti*)

               (Draw_Point_R pt_new *LaCo_Punti_scelti*)
               (setq pts (mapcar '(lambda (p) (if (< (distance p pt_old) 1e-6) pt_new p))
    pts))

               (setq e_marker (entlast))
               (setq pts_ext (Util_Scale_Pts_From_Centroid pts 5.0))
               (Draw_Con5P_Nor_Compl pts *LaCo_Tracciati* *LaCo_Costruzioni* pts_ext)
               (setq current_conic_ents (Util_Collect_Ents e_marker)))
             (princ "\nSpostamento annullato. Riferimenti mantenuti.")))))
     (if guide_ents (foreach e guide_ents (if (entget e) (entdel e))))
     (if quad_ents (Util_Del_Ents_List quad_ents))))
  (Restore_Conic_App old_os old_pd old_ce))
)
;;-------------------------------------------------------------------------------------
;;-------------------------------------------------------------------------------------
;;-------------------------------------------------------------------------------------
;;-------------------------------------------------------------------------------------
```