UNIVERSITÀ
DEGLI STUDI
FIRENZE

DOTTORATO DI RICERCA IN
INGEGNERIA INDUSTRIALE E DELL'AFFIDABILITÀ

CICLO XXVIII

COORDINATORE Prof. Maurizio De Lucia
REFERENTE Prof. Mario Tucci

Progettazione e testing di un software modulare di bordo e di gestione per AUV orientato alla portabilità

Settore Scientifico Disciplinare ING-IND/13

Dottorando
Dott. Niccolò Monni

Tutor
Prof. Benedetto Allotta

Coordinatore
Prof. Maurizio De Lucia

Anni 2013/2015

Indice

1 Introduzione **4**

1.1 Contesto 4

1.2 Robot Operating System 7

2 Fondamenti teorici **10**

2.1 Sistemi di riferimento 10

2.2 Cinematica 12

2.3 Dinamica 13

2.3.1 Dinamica del corpo rigido 14

2.3.2 Effetti idrodinamici 15

2.4 Filtro di Kalman 19

2.4.1 Filtro di Kalman Esteso 20

3 Software **23**

3.1 Gestione dispositivi di bordo 26

3.2 Gestione di missione e controllo 27

3.2.1 Mission Supervisor 27

3.2.2 Navigation 37

3.2.3 Control 42

3.3 Pianificazione ed esecuzione delle missioni 44

3.4 Gestione delle emergenze 47
3.4.1 Identificazione 48
3.4.2 Contromisure 49
3.4.3 Configurazione 50
3.5 Percezione 52
3.6 Comunicazione 53
3.6.1 Scambio di file 55
3.7 Interfacce 56
3.7.1 Navigazione e testing 56
3.7.2 Pianificazione e controllo 59
3.8 Simulazioni 60

4 Applicazioni **61**
4.1 Tifone 61
4.2 MARTA 66
4.2.1 Caratteristiche principali 67
4.2.2 Risultati 71
4.3 FeelHippo 77
4.3.1 Caratteristiche meccaniche 77
4.3.2 Architettura elettronica 79
4.4 Cooperazione 80
4.4.1 Localizzazione acustica 80

5 Conclusioni **89**

A Listati **91**
A.1 Reference Generator 91
A.2 Tracker 92
A.3 Finder 94

A.4 Filter 96
A.5 Watcher 99
A.6 PerceptionAlgorithm 100

Capitolo 1

Introduzione

Questa tesi raccoglie le esperienze e il lavoro che ho svolto nel campo della robotica sottomarina durante i tre anni del mio dottorato di ricerca, presso il Laboratorio di Modellazione dinamica e Meccatronica (MDM Lab) del DIEF (Dipartimento di Ingegneria Industriale - Firenze) dell'Università degli Studi di Firenze, tra il 2013 e il 2015.

1.1 Contesto

Durante questo periodo, il gruppo di ricerca di MDM Lab è stato attivamente coinvolto nella progettazione, costruzione e controllo di UUV (Unmanned Underwater Vehicles), veicoli sottomarini senza un pilota a bordo, nell'ambito di diversi progetti. Gli UUV sono classificati in due categorie principali:

- **ROV (Remotely Operated Vehicle) -** il pilota è un essere umano che guida il veicolo con una console di controllo posta su una nave di supporto. Il ROV è sempre collegato alla nave attraverso un cavo ombelicale che ha la duplice funzione di fornire la potenza richiesta e di assicurare uno scambio di dati continuo con la superficie.

- **AUV (Autonomous Underwater Vehicle) -** in questo caso i veicoli sono completamente autonomi e non richiedono l'intervento di un operatore per il movimento e il controllo dei payload. Nessun cavo è necessario affinché un AUV possa eseguire una missione anche lontano dal luogo di lancio, con l'unico vincolo di conservare energia sufficiente (di solito immagazzinata nelle batterie poste a bordo) per raggiungere il punto di recupero.

I progetti in cui è stato coinvolto MDM Lab nel corso degli anni 2012-2015 hanno riguardato entrambe le tipologie di UUV. In particolare il gruppo ha iniziato ad affrontare la ricerca nel campo della robotica subacquea con l'inizio del progetto THESAURUS (TecnicHe per l'Esplorazione Sottomarina Archeologica mediante l'Utilizzo di Robot aUtonomi a Sciami), progetto iniziato nel marzo 2011 e finanziato dalla Regione Toscana. All'interno di THESAURUS (concluso con successo alla fine dell'estate 2013) MDM Lab è stato partner del consorzio coordinato dal Centro Enrico Piaggio dell'Università di Pisa. L'obiettivo del progetto THESAURUS era quello di sviluppare metodologie scientifiche multidisciplinari utili a individuare, censire e documentare manufatti e relitti subacquei di valore archeologico, storico-artistico ed etnoantropologico. Uno dei mezzi da sviluppare nell'ambito del progetto THESAURUS era un piccolo team di AUV, equipaggiati con sensori ottici e acustici, capaci di cooperare per uno scopo comune sfruttando anche la capacità di comunicare tramite dispositivi acustici. All'interno del progetto il compito di MDM Lab includeva il design, lo sviluppo e la costruzione degli AUV del team in aggiunta all'implementazione del software di Guida, Navigazione e Controllo (GNC), in grado di assicurare ai veicoli capacità di viaggio autonomo in condizioni di sicurezza lungo una traiettoria desiderata.

Figura 1.1: Logo del progetto THESAURUS

Durante l'ultimo anno del progetto THESAURUS, MDM Lab ha cominciato a lavorare sul progetto Europeo ARROWS (ARchaeological RObot systems for the World's Seas) come partner coordinatore, progetto concluso durante l'estate del 2015. L'obiettivo del progetto ARROWS era molto simile a quello di THESAURUS, con alcune innovazioni come la capacità di una ri-pianificazione autonoma e dinamica delle missioni basata su un modello ontologico del mondo, che potesse essere aggiornato dalle informazioni ottenute dai veicoli coinvolti in tempo reale e ri-condiviso. Il ruolo di MDM Lab all'interno del progetto riguardava il design, lo sviluppo e la costruzione di un nuovo AUV con forti caratteristiche di modularità, e l'implementazione di un nuovo sistema GNC. Questo nuovo veicolo è stato integrato all'interno di un team eterogeneo di AUV già esistenti (appartenenti agli altri partner del progetto) o sviluppati da zero all'interno di ARROWS.

Figura 1.2: Logo del progetto ARROWS

Durante il periodo di sovrapposizione dei due progetti (fine di THESAURUS e inizio di ARROWS), che coincide con l'inizio della mia attività di

dottorato, avendo a disposizione per la prima volta più AUV di tipo diverso, si è palesata la necessità di un unico software di bordo capace di adattarsi ad ogni veicolo, permettere la condivisione di informazioni ed essere facilmente integrabile con le applicazioni sviluppate dagli altri partner. Si è quindi deciso di generalizzare ulteriormente il problema progettando un software capace di poter essere adattato a un qualunque UUV dotato di un computer di bordo e a qualunque tipologia di utilizzo, facendo della modularità e della flessibilità le sue caratteristiche principali. Questo ambizioso progetto è l'attività principale svolta all'interno di questo dottorato di ricerca; il software è stato creato in parallelo alla costruzione degli AUV MARTA e FeelHippo, assecondandone le esigenze e, nelle fasi iniziali, è stato testato principalmente sull'AUV TifOne (veicoli che verranno descritti nel capitolo 4). Lo sviluppo ha richiesto più di due anni, al termine dei quali si è giunti ad una versione stabile e funzionante applicata su tutti i veicoli citati.

Tutto il pacchetto si basa su ROS, software open source che ha semplificato molto il lavoro e che rappresenta ormai uno standard nella robotica di alto livello.

1.2 Robot Operating System

Figura 1.3: Logo di ROS

ROS è un framework open source per lo sviluppo di applicazioni robotiche. Nato nel 2007 all'interno dello *Stanford Artificial Intelligence Labora-*

tory, il laboratorio di intelligenza artificiale presso la Stanford University, si è rapidamente diffuso sia a livello universitario che industriale, diventando di fatto uno standard per le applicazioni robotiche. Fornisce i servizi tipici dei comuni sistemi operativi quali astrazioni hardware, controllo di device, scambio di messaggi tra processi, usufruendo del sistema operativo host per funzionalità di basso livello come lo scheduling dei processi, l'accesso al disco e la comunicazione di rete.

ROS si basa su un'architettura a grafo, chiamato *ROS Computational Graph*, simile ad una rete peer-to-peer, nella quale ogni processo in esecuzione può avviare uno scambio di informazioni con gli altri. I principali elementi di tale architettura sono:

- *Nodi*: ovvero un qualsiasi processo in grado di svolgere un'attività computazionale. La suddivisione dell'intero sistema di controllo in moduli software, ciascuno dei quali adibito ad una particolare funzione, permette lo sviluppo e una fase di debug più semplice, nonché una maggior riusabilità del codice. Particolare attenzione viene rivolta al *Master Node,* il processo principale, che fornisce servizi di name registration e lookup, identificando e tenendo traccia di tutti i nodi in esecuzione, che in tal modo si possono individuare e instaurare una comunicazione. Gestisce inoltre il *Parameter Server*, un database, accessibile in lettura e scrittura da tutti i nodi, usato per la memorizzazione di parametri operativi;

- *Messaggi*: I flussi di informazioni scambiati fra i vari nodi vengono organizzati mediante l'utilizzo di messaggi. Tali entità sono delle semplici strutture, definibili a seconda delle diverse esigenze di progetto e

dei vari nodi, e costruite tramite la combinazione dei principali tipi di primitivi (float, integer, boolean, ...);

- *Topics*: La comunicazione è regolata secondo il principio di *publish/-subscribe* e tramite l'uso di topic, ovvero identificatori che ne identificano il contenuto. Un nodo invia un messaggio pubblicandolo su una particolare topic a cui si sottoscrivono i nodi interessati a tale contenuto; secondo tale principio viene accentua la modularità del sistema, in quanto i nodi che pubblicano non sono a conoscenza dei nodi che leggono e viceversa;

- *Servizi*: Permettono una comunicazione di tipo request/reply più diretta rispetto alla trasmissione molti-a-molti offerta dalle topic, e definita da un nome identificante il servizio e da una coppia di messaggi. Un nodo può inviare richiesta del servizio tramite un opportuno messaggio di request al processo che offre tale servizio, che invia la risposta mediante un messaggio di reply.

L'automatizzazione del sistema, di importanza vitale nelle applicazioni robotiche, è resa possibile tramite lo strumento *roslaunch* che interpreta appositi file XML al cui interno sono inseriti i nodi da avviare e gli eventuali parametri da caricare sul Parameter Server. Un ulteriore strumento offerto da ROS è il *rosbag,* un particolare nodo che si occupa della memorizzazione, in particolari file di tipo *bag,* di tutti i messaggi scambiati all'interno del Computation Graph. Tale strumento risulta molto utile sia in fase di debug, per controllare il corretto funzionamento del sistema, sia durante la normale vita operativa dell'apparato per una successiva elaborazione dei dati registrati.

Capitolo 2

Fondamenti teorici

In questo capitolo verranno introdotte le notazioni usate nel seguito della trattazione, e verrà analizzato il modello del veicolo con particolare attenzione alle equazioni cinematiche e dinamiche.

2.1 Sistemi di riferimento

Nel seguito viene fatto uso di due sistemi di riferimento:

- Body-Fixed Frame $< b >$, terna di riferimento solidale al veicolo, che ha origine O_b nel baricentro del veicolo e gli assi X_b, Y_b e Z_b coincidenti con i suoi assi principali di inerzia. Generalmente tali assi sono definiti prendendo X_b come asse longitudinale, diretto da poppa verso prua, Y_b come asse trasversale diretto verso tribordo (asse destro del veicolo guardando verso prua) e Z_b come asse verticale diretto verso il basso;

- Earth-Fixed Frame $< n >$, terna di riferimento inerziale, con origine O_n situata sul livello del mare e con assi allineati secondo il sistema di riferimento NED (North East Down), ovvero con X_n diretto verso nord, Y_n diretto verso est e Z_n diretto verso il basso.

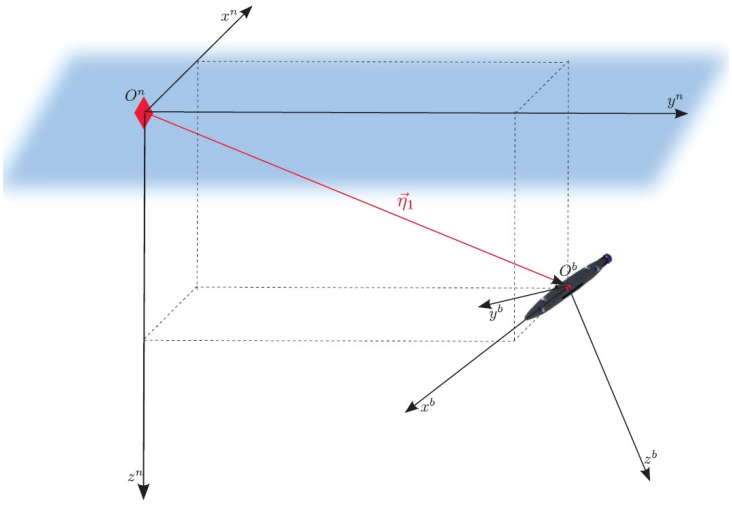


Figura 2.1: Sistemi di riferimento

Generalmente durante lo studio del moto di un veicolo sottomarino si preferisce indicare la posizione e l'orientazione del veicolo tramite le coordinate in terna fissa, mentre le velocità lineari ed angolari in terna body. In particolare, le varie quantità sono definite in accordo alla notazione SNAME (Society of Naval Architects and Marine Engineers) e vengono riportate in 2.1

$$\vec{\eta} = \begin{bmatrix} \vec{\eta}_1^T & \vec{\eta}_2^T \end{bmatrix}^T \qquad \vec{\eta}_1 = \begin{bmatrix} x & y & z \end{bmatrix}^T \qquad \vec{\eta}_2 = \begin{bmatrix} \phi & \theta & \psi \end{bmatrix}^T$$

$$\vec{v} = \begin{bmatrix} \vec{v}_1^T & \vec{v}_2^T \end{bmatrix}^T \qquad \vec{v}_1 = \begin{bmatrix} u & v & w \end{bmatrix}^T \qquad \vec{v}_2 = \begin{bmatrix} p & q & r \end{bmatrix}^T$$

$$\vec{\tau} = \begin{bmatrix} \vec{\tau}_1^T & \vec{\tau}_2^T \end{bmatrix}^T \qquad \vec{\tau}_1 = \begin{bmatrix} X & Y & Z \end{bmatrix}^T \qquad \vec{\tau}_2 = \begin{bmatrix} K & M & N \end{bmatrix}^T \tag{2.1}$$

dove $\vec{\eta}$ denota il vettore di posizione $\vec{\eta}_1$ ed orientazione $\vec{\eta}_2$, in coordinate in terna fissa, $\vec{v}$ indica il vettore delle velocità lineari $\vec{v}_1$ ed angolari $\vec{v}_2$ espresse in terna body e $\vec{\tau}$ è usato per descrivere le forze $\vec{\tau}_1$, ed i momenti $\vec{\tau}_2$, agenti

sul veicolo, in terna body. I movimenti lungo le varie direzioni sono definiti:

- *surge,* traslazione lungo x;
- *sway,* traslazione lungo y;
- *heave*, traslazione lungo z;
- *roll,* rotazione attorno all'asse x;
- *pitch,* rotazione attorno all'asse y;
- *yaw,* rotazione attorno all'asse y.

2.2 Cinematica

L'orientazione della terna body rispetto alla terna fissa viene espressa mediante la terna di angoli di Eulero di tipo RPY (Roll ϕ, Pitch θ, Yaw ψ), per tale motivo è facile dimostrare la relazione riguardante il moto di traslazione del veicolo

$$\dot{\vec{\eta}}_1 = J_1\left(\vec{\eta}_2\right)\vec{v}_1 \tag{2.2}$$

dove $J_1\left(\vec{\eta}_2\right)$ è la matrice di rotazione dipendente dagli angoli di Eulero, ottenuta mediante composizione di tre successive rotazioni definite rispetto la terna fissa, definita in 2.3

$$\begin{aligned} J_1\left(\vec{\eta}_2\right) &= R_z(\phi)R_y(\theta)R_x(\psi) = \\ &= \begin{bmatrix} c_\psi c_\theta & -s_\psi c_\phi + c_\psi s_\theta s_\phi & s_\psi s_\phi + c_\psi c_\phi s_\theta \\ s_\psi c_\theta & c_\psi c_\phi + s_\phi s_\theta s_\phi & -c_\psi s_\phi + s_\theta s_\psi c_\phi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{bmatrix} \end{aligned} \tag{2.3}$$

nella quale è stato fatto uso della notazione $c_\alpha = \cos\alpha$ e $s_\alpha = \sin\alpha$. Da tale relazione è possibile inoltre identificare la relazione inversa

$$\vec{v}_1 = J_1^{-1}\left(\vec{\eta}_2\right)\dot{\vec{\eta}}_1 = J_1^T\left(\vec{\eta}_2\right)\dot{\vec{\eta}}_1 \tag{2.4}$$

Per quanto riguarda la velocità angolare $\vec{v}_2$ e il vettore delle derivate degli angoli RPY $\dot{\vec{\eta}}_2$, vale la relazione

$$\dot{\vec{\eta}}_2 = J_2\left(\vec{\eta}_2\right)\vec{v}_2 \tag{2.5}$$

dove $J_2\left(\vec{\eta}_2\right)$ è definita come

$$J\left(\vec{\eta}_2\right) = \begin{bmatrix} 1 & s\phi t\theta & c\phi t\theta \\ 0 & c\phi & -s\phi \\ 0 & s\phi/c\theta & c\phi/c\theta \end{bmatrix} \tag{2.6}$$

nella quale è stata introdotta la relazione $t\alpha = \tan\alpha$. Anche in questo caso è possibile definire la relazione inversa

$$\vec{v}_2 = J_2^{-1}\left(\vec{\eta}_2\right)\dot{\vec{\eta}}_2 = \begin{bmatrix} 1 & 0 & -s\theta \\ 0 & c\phi & c\theta s\phi \\ 0 & -s\phi & c\theta c\phi \end{bmatrix}\dot{\vec{\eta}}_2 \tag{2.7}$$

Sintetizzando i risultati ottenuti, l'espressioni cinematiche possono essere espresse con la seguente forma compatta

$$\begin{bmatrix} \dot{\vec{\eta}}_1 \\ \dot{\vec{\eta}}_2 \end{bmatrix} = \begin{bmatrix} J_1\left(\vec{\eta}_2\right) & O_{3x3} \\ O_{3x3} & J_2\left(\vec{\eta}_2\right) \end{bmatrix}\begin{bmatrix} \vec{v}_1 \\ \vec{v}_2 \end{bmatrix} \quad \Longleftrightarrow \quad \dot{\vec{\eta}} = J\left(\vec{\eta}\right)\vec{v} \tag{2.8}$$

dalla quale si può ricavare

$$\ddot{\vec{\eta}} = J\left(\vec{\eta}\right)\dot{\vec{v}} + \dot{J}\left(\vec{\eta}\right) \quad \Longleftrightarrow \quad \dot{\vec{v}} = J^{-1}\left(\vec{\eta}\right)\left[\ddot{\vec{\eta}} - \dot{J}\left(\vec{\eta}\right)J^{-1}\left(\vec{\eta}\right)\dot{\vec{\eta}}\right] \tag{2.9}$$

2.3 Dinamica

Nel seguente paragrafo verrà descritta l'equazione non lineare, riportata in (2.10), che descrive il moto di un corpo rigido con 6 gradi di libertà nello spazio

$$M\dot{\vec{v}} + C(\vec{v})\vec{v} + D(\vec{v})\vec{v} + \vec{g}(\vec{\eta}) = \vec{\tau} - \vec{\tau}_d \tag{2.10}$$

dove:

- M, matrice di inerzia;
- $C(\vec{v})$, matrice dei termini centripeti e di accelerazione di Coriolis;
- $D(\vec{v})$, matrice di Damping, contenente gli effetti viscosi di smorzamento idrodinamico;
- $\vec{g}(\vec{\eta})$, vettore della forza di gravità e di galleggiamento;
- $\vec{\tau}$, vettore delle forze e dei momenti applicati al veicolo;
- $\vec{\tau}_d$, vettore dei disturbi ambientali, come correnti marine, onde o venti.

Data la natura dell'ambiente operativo, tale relazione deve tener conto degli effetti di interazione che avvengono tra il veicolo ed il fluido in cui è immerso. Per tale motivo la matrice di inerzia e la matrice dei fenomeni centripeti e di Coriolis sono costituite dalla somma di due contributi

$$\begin{aligned} M &= M_{RB} + M_A \\ C &= C_{RB} + C_A \end{aligned} \tag{2.11}$$

2.3.1 Dinamica del corpo rigido

Usando la terna di riferimento Body-Fixed Frame, riportata nei paragrafi precedenti, nella quale l'origine O_b coincide con il centro di massa del veicolo e gli assi lungo i suoi assi di inerzia, e indicando con m la massa del veicolo e con I_{ii} il momento di inerzia rispetto al generico asse i, si possono definire le seguenti matrici:

- M_{RB} è la matrice di massa legata al moto del corpo rigido nello spazio. La matrice assume la seguente forma

$$M_{RB} = \begin{bmatrix} m & 0 & 0 & & & \\ 0 & m & 0 & & O_{3x3} & \\ 0 & 0 & m & & & \\ & & & I_{xx} & 0 & 0 \\ & O_{3x3} & & 0 & I_{yy} & 0 \\ & & & 0 & 0 & I_{zz} \end{bmatrix} \tag{2.12}$$

- C_{RB} è la matrice dei fenomeni centripeti e di Coriolis riguardanti direttamente il corpo rigido. Sotto le ipotesi fatte si ottiene la matrice diagonale a blocchi

$$C_{RB} = \begin{bmatrix} 0 & -mr & mq & & & \\ mr & 0 & -mp & & O_{3x3} & \\ -mq & mp & 0 & & & \\ & & & 0 & I_{zz}r & -I_{yy}q \\ & O_{3x3} & & -I_{zz}r & 0 & I_{xx}p \\ & & & I_{yy}q & -I_{xx}p & 0 \end{bmatrix} \tag{2.13}$$

2.3.2 Effetti idrodinamici

In questo paragrafo saranno esaminati gli effetti idrodinamici su un corpo rigido in movimento in un fluido. Gli effetti sono principalmente di due tipologie:

- Massa ed inerzia aggiunte: contributo legato all'inerzia del fluido che aderisce alla superficie del veicolo;

- Effetti viscosi: contributo realizzato dalle forze dissipative di Drag e Lift dovute alla viscosità del fluido sul veicolo in movimento.

Massa ed inerzia aggiunte

Quando un corpo rigido si muove in un fluido si deve considerare che la massa aggiunta del fluido che aderisce alla superficie del corpo stesso (condizione di aderenza). Tale effetto non può essere trascurato data l'elevata densità dell'acqua, quindi sarà necessaria una forza aggiuntiva per accelerare veicolo e fluido circostante. A tale scopo si introducono la matrice di massa ed inerzia aggiunte M_A e la matrice degli effetti centrifughi e di Coriolis aggiunti C_A. Defininendo la forza di massa aggiunta X_A lungo l'asse x e causata da un'accelerazione $\dot{u}$ nella stessa direzione

$$X_A = X_{\dot{u}}\dot{u} \qquad dove \qquad X_{\dot{u}} = \frac{\partial X}{\partial \dot{u}} \tag{2.14}$$

e sotto le ipotesi che il veicolo sia totalmente immerso, che presenti tre assi di simmetria e che si muova con velocità non elevate, si possono definire le seguenti matrici:

- M_A è la matrice diagonale di massa aggiunta

$$M_A = \begin{bmatrix} X_{\dot{u}} & 0 & 0 & & & \\ 0 & Y_{\dot{v}} & 0 & & O_{3x3} & \\ 0 & 0 & Z_{\dot{w}} & & & \\ & & & K_{\dot{p}} & 0 & 0 \\ & O_{3x3} & & 0 & M_{\dot{q}} & 0 \\ & & & 0 & 0 & N_{\dot{r}} \end{bmatrix} \tag{2.15}$$

- C_A è la matrice dei fenomeni centripeti e di Coriolis aggiunti

$$C_A = \begin{bmatrix} & & & 0 & -Z_{\dot{w}}w & Y_{\dot{v}}v \\ & O_{3x3} & & Z_{\dot{w}}w & 0 & -X_{\dot{u}}u \\ & & & -Y_{\dot{v}}v & X_{\dot{u}}u & 0 \\ 0 & -Z_{\dot{w}}w & Y_{\dot{v}}v & 0 & -N_{\dot{r}}r & M_{\dot{q}}q \\ Z_{\dot{w}}w & 0 & -X_{\dot{u}}u & N_{\dot{r}}r & 0 & -K_{\dot{p}}p \\ -Y_{\dot{v}}v & X_{\dot{u}}u & 0 & -M_{\dot{q}}q & K_{\dot{p}}p & 0 \end{bmatrix} \tag{2.16}$$

Effetti viscosi

Il fenomeno dello smorzamento idrodinamico modellato all'interno della matice di Damping $D(\vec{v})$, avviene ogni volta che un corpo solido si muove all'interno di un fluido viscoso. Viene causato dai seguenti contributi:

- forza di resistenza di attrito viscoso dovuta alle forze che si scambiano il corpo ed il fluido in moto relativo tra di loro. Se infatti il fluido è viscoso e se, per esempio, consideriamo il corpo in moto ed il fluido fermo, le particelle di fluido a contatto con il corpo dovranno essere in moto con il corpo (condizione di aderenza). Il corpo quindi eserciterà sulle particelle di fluido più prossime ad esso una azione accelerante. Per il principio di azione e reazione il fluido quindi eserciterà sul corpo una azione frenante;

- forza di resistenza di forma dovuta anch'essa alla viscosità del fluido, ma attraverso il meccanismo della separazione delle linee di flusso. A causa della viscosità il fluido perde energia aggirando il corpo e ciò genera la separazione delle linee di flusso che, a sua volta, formerà una zona detta zona di ricircolazione nella regione posteriore. Tale fenomeno genera

la resistenza di forma e si evidenzia maggiormente nei corpi cosiddetti tozzi, dove cioè le dimensioni perpendicolari al moto sono consistenti rispetto alle altre.

Per un veicolo sottomarino a 6 DOF che si muove ad elevate velocità tale matrice risulta fortemente non lineare ed accoppiata. Tuttavia, sotto le ipotesi di velocità di avanzamento non elevata e supponendo che il veicolo abbia tre piani di simmetria, la matrice $D(\vec{v})$ risulta diagonale e definita

$$\begin{aligned} D(\vec{v}) \;=\; & -diag\left\{ \begin{array}{cccccc} X_u & Y_v & Z_w & K_p & M_q & N_r \end{array} \right\} + \\ & -diag\left\{ \begin{array}{cccccc} X_{u|u|}|u| & Y_{v|v|}|v| & Z_{w|w|}|w| & K_{p|p|}|p| & M_{q|q|}|q| & N_{r|r|}|r| \end{array} \right\} \end{aligned}$$

Effetti di gravità e di galleggiamento

Il vettore $\vec{g}(\vec{\eta})$ modella gli effetti della forza di gravità $\vec{f_G}^b$ che agisce su centro di massa del veicolo

$$\vec{r_G}^b = \left[\begin{array}{ccc} x_G & y_G & z_G \end{array} \right]^T \tag{2.17}$$

e della forza di galleggiamento $\vec{f_B}^b$ che agisce su centro di galleggiamento del veicolo

$$\vec{r_B}^b = \left[\begin{array}{ccc} x_B & y_B & z_B \end{array} \right]^T \tag{2.18}$$

Definito il vettore dell'accelerazione gravitazionale, espresso in Earth-Fixed Frame,

$$\vec{g}^n = \left[\begin{array}{cccc} 0 & 0 & 9,81 \end{array} \right]^T \mathrm{m/s^2} \tag{2.19}$$

il modulo della forza peso W e della forza di galleggiamento B sono date da

$$W = m\,||\vec{g}^n|| \qquad B = \rho\, V_{RB}\,||\vec{g}^n|| \tag{2.20}$$

dove m è la massa e V_{RB} il volume immerso del veicolo e ρ è la densità del fluido. Le due forze, espresse in Body-Fixed Frame, si possono esprimere

come

$$\begin{aligned} \vec{f_G}^{\,b}(\vec{\eta}_2) &= J_1^{-1}(\vec{\eta}_2)\, m\, \vec{g}^n \\ \vec{f_B}^{\,b}(\vec{\eta}_2) &= -J_1^{-1}(\vec{\eta}_2)\, \rho\, V_{RB}\, \vec{g}^n \end{aligned} \tag{2.21}$$

A partire da tali risultati, il vettore $\vec{g}(\vec{\eta})$ può essere scritto come

$$\vec{g}(\vec{\eta}) = -\begin{bmatrix} \vec{f_G}^{\,b} + \vec{f_B}^{\,b} \\ \vec{r_B}^{\,b} \wedge \vec{f_B}^{\,b} \end{bmatrix} \tag{2.22}$$

esplicitando i risultati

$$\vec{g}(\vec{\eta}) = -\begin{bmatrix} (W-B)\, s\theta \\ -(W-B)\, c\theta\, s\phi \\ -(W-B)\, c\theta\, c\phi \\ y_B\, B\, c\theta c\phi - z_B\, B\, c\theta s\phi \\ -z_B\, B\, s\theta - x_B\, B\, c\theta c\phi \\ y_B\, B\, c\theta s\phi - y_B\, B\, s\theta \end{bmatrix} \tag{2.23}$$

2.4 Filtro di Kalman

Si riporta un breve richiamo alle equazioni del filtro di Kalman che verranno usate nei capitoli successivi. Il filtro di Kalman è un efficiente filtro ricorsivo che valuta lo stato di un sistema dinamico a partire da una serie di misure soggette a rumore. Si consideri il sistema dinamico lineare a tempo discreto riportato in (2.24), soggetto a rumore di processo w_k e a rumore di misura v_k.

$$\begin{cases} \vec{x}_{k+1} = A_k \vec{x}_k + B_k \vec{u}_k + \vec{w}_k \\ \vec{y}_k = C_k \vec{x}_k + \vec{v}_k \end{cases} \tag{2.24}$$

Il rumore di processo $\vec{w}_k$ ed il rumore di misura $\vec{v}_k$ sono incorrelati nel tempo, congiuntamente gaussiani a media nulla, con matrici di covarianza

$$\begin{aligned} Q_k &= Var(\vec{w}_k) \\ R_k &= Var(\vec{v}_k) \end{aligned} \tag{2.25}$$

Si suppone che le matrici A_k, B_k, C_k siano note e che lo stato iniziale $\vec{x}_0$, non noto, sia modellabile come una variabile aleatoria gaussiana con media e covarianza note. Sotto queste ipotesi esiste una soluzione ricorsiva al problema di stima dello stato, ottima dal punto di vista statistico, data dal filtro di Kalman. Le equazioni del filtro di Kalman sono

- Predizione ad un passo

$$\begin{aligned} \hat{\vec{x}}_{k+1|k} &= A_k\, \hat{\vec{x}}_{k|k} + B_k\, \vec{u}_k \\ P_{k+1|k} &= A_k\, P_{k|k}\, A_k^T + Q_k \end{aligned} \tag{2.26}$$

  dove $\hat{\vec{x}}_{k+1|k}$ è la stima dello stato e $P_{k+1|k}$ è l'errore quadratico medio, entrambi relativi all'istante $k+1$ basati sulle osservazioni fino all'istante k;

- Aggiornamento della stima

$$\begin{aligned} \hat{\vec{x}}_{k+1|k+1} &= \hat{\vec{x}}_{k+1|k} + L_{k+1}\left(\vec{y}_{k+1} - C_k\, \hat{\vec{x}}_{k+1|k}\right) \\ L_{k+1} &= P_{k+1|k} C_k^T (C_k\, P_{k+1|k}\, C_k^T + R_{k+1})^{-1} \\ P_{k+1|k+1} &= P_{k+1|k} - P_{k+1|k}\, C_k^T (C_k\, P_{k+1|k}\, C_k^T + R_k)^{-1} C_k\, P_{k+1|k} = \\ &= P_{k+1|k}(I - C_k^T L_{k+1}^T) \end{aligned} \tag{2.27}$$

  dove L_k è il guadagno del filtro di Kalman all'istante $k+1$.

Le equazioni vengono inizializzate con $\hat{\vec{x}}_{0|0}$ e $P_{0|0}$, media e covarianza a priori della variabile aleatoria $\vec{x}_0$.

2.4.1 Filtro di Kalman Esteso

Nel caso generale di sistema dinamico non lineare riportato in (2.28), non è possibile ottenere una soluzione ottima al problema del filtraggio e si deve ricorrere a soluzioni approssimate. Il metodo di linearizzazione, sul quale si

basa il filtro di Kalman Esteso (EKF), è una delle tecniche più usate per trovare una soluzione approssimata al problema del filtraggio.

$$\begin{cases} \vec{x}_{k+1} = f_k\left(\vec{x}_k, \vec{u}_k\right) + \vec{w}_k \\ \vec{y}_k = h_k\left(\vec{x}_k, \vec{u}_k\right) + \vec{v}_k \end{cases} \tag{2.28}$$

L'idea è quella di approssimare le funzioni non lineari $f(\cdot)$ e $h(\cdot)$ mediante uno sviluppo in serie di Taylor al primo ordine nell'intorno della stima corrente. In questo modo il sistema (2.28) diventa lineare e ipotizzando che la densità di probabilità condizionata sia Gaussiana, si può applicare il filtro di Kalman al sistema linearizzato, calcolando media e covarianza.

Si definiscono le matrici Jacobiane

$$\begin{cases} A_k = \left.\frac{\partial f(\vec{x},\vec{u})}{\partial \vec{x}}\right|_{\vec{x}=\hat{\vec{x}}_{k|k},\vec{u}=\vec{u}_k} \\ C_k = \left.\frac{\partial h(\vec{x},\vec{u})}{\partial \vec{x}}\right|_{\vec{x}=\hat{\vec{x}}_{k|k-1},\vec{u}=\vec{u}_k} \end{cases} \tag{2.29}$$

e si approssimano al primo ordine le suddette funzioni

$$\begin{cases} f\left(k, \vec{x}_k\right) = f_k\left(\hat{\vec{x}}_{k|k}, \vec{u}_k\right) + A_k\left(\vec{x}_k - \hat{\vec{x}}_{k|k}\right) \\ h\left(k, \vec{x}_k\right) = h_k\left(\hat{\vec{x}}_{k|k}, \vec{u}_k\right) + C_k\left(\vec{x}_k - \hat{\vec{x}}_{k|k-1}\right) \end{cases} \tag{2.30}$$

Utilizzando tali approssimazioni, il sistema assume una forma lineare per cui si può applicare il filtro di Kalman, ottenendo le seguenti ricorsioni:

- Predizione ad un passo

$$\begin{aligned} \hat{\vec{x}}_{k+1|k} &= f_k\left(\hat{\vec{x}}_{k|k}, \vec{u}_k\right) \\ P_{k+1|k} &= A_k\, P_{k|k}\, A_k^T + Q_k \end{aligned} \tag{2.31}$$

- Correzione della stima

$$\begin{aligned}
\hat{\vec{y}}_{k+1|k} &= h_k\left(\hat{\vec{x}}_{k|k}, \vec{u}_k\right) \\
L_{k+1} &= P_{k+1|k} C_k^T (C_k\, P_{k+1|k}\, C_k^T + R_{k+1})^{-1} \\
\hat{\vec{x}}_{k+1|k+1} &= \hat{\vec{x}}_{k+1|k} + L_{k+1}\left(\vec{y}_{k+1} - \hat{\vec{y}}_{k+1|k}\right) \\
P_{k+1|k+1} &= P_{k+1|k} - P_{k+1|k}\, C_k^T (C_k\, P_{k+1|k}\, C_k^T + R_k)^{-1} C_k\, P_{k+1|k}
\end{aligned} \tag{2.32}$$

Si osserva che nel filtro di Kalman esteso, oltre agli errori di processo e di misura, sono presenti errori di linearizzazione, che non possono essere quantificati e di cui non si tiene conto nel valutare la covarianza dell'errore di stima.

Capitolo 3

Software

Il software di un sottomarino autonomo deve essere in grado di svolgere diversi compiti: ad esempio comunicare con diversi dispositivi, determinare lo stato del veicolo oppure prendere decisioni in base all'ambiente circostante.

Un possibile approccio alla programmazione potrebbe essere quello di creare un applicativo multi-thread capace di gestire internamente tutte queste problematiche, in questo modo si garantirebbe una maggiore efficienza di calcolo ed un occupazione di memoria ridotta. Purtroppo gli svantaggi di questo sistema sono svariati: innanzitutto un applicativo di questo tipo risulterebbe molto complesso e richiederebbe un codice dalla struttura molto ben articolata per risultare leggibile e facilmente modificabile, inoltre il lavoro in team viene reso più complesso dal fatto di avere un unico codice da sviluppare. Tra l'altro si deve anche considerare che in ambiente subacqueo la dinamica è relativamente lenta e ai sistemi di controllo non è quindi richiesto di lavorare a frequenze elevate. Inoltre i moderni elaboratori hanno dimensioni molto ridotte e mettono a disposizione risorse decisamente sovradimensionate rispetto ai requisiti del controllo.

L'alternativa proposta in questo lavoro consiste nel suddividere tutto il

software di bordo in tanti piccoli applicativi ciascuno con la propria funzione. Il sistema diviene in questo modo "modulare" in quanto a seconda della configurazione hardware o dalla volontà del programmatore, questi "pacchetti" possono essere aggiunti, tolti o sostituiti in modo molto semplice e senza richiedere una nuova compilazione.

L'utilizzo del framework ROS semplifica enormemente l'organizzazione del codice e la comunicazione inter-processo che avviene tramite socket dedicati, generati automaticamente dal sistema. In questo modo si ottiene una vera e propria rete di "nodi" capaci di interagire tra loro e di svolgere tutte le funzioni assegnate in modo autonomo. Questa modularità si traduce anche nella possibilità di adattare, in maniera piuttosto semplice, lo stesso software a più veicoli con caratteristiche simili per far compiere, ad esempio, lo stesso tipo di missione a due AUV con hardware completamente diverso.

Un altro concetto perseguito nello svolgimento di questo lavoro di tesi è la "flessibilità", cioè la capacità del software di adattarsi in maniera semplice ad una qualsiasi situazione. L'obiettivo è quello di poter modificare il modo in cui il veicolo affronta la missione assegnata, oppure aggiungere funzionalità alla stessa senza dover operare modifiche "importanti" al software e possibilmente senza dover ricompilare. A questo scopo è stato creato un insieme di nodi strutturali che rappresentano il "core" del sistema, cioè la parte che gestisce la navigazione e di cui non si può fare a meno. È stato fatto in modo che questa parte sia interamente configurabile da file esterni in modo da non avere mai bisogno di modificarne il codice. A fianco di questo ci sono altri nodi che svolgono compiti singoli e possono essere creati o modificati dallo sviluppatore secondo dei template precostituiti, questo aspetto verrà approfondito successivamente.

Modularità e flessibilità combinate insieme rispondono all'obiettivo prin-

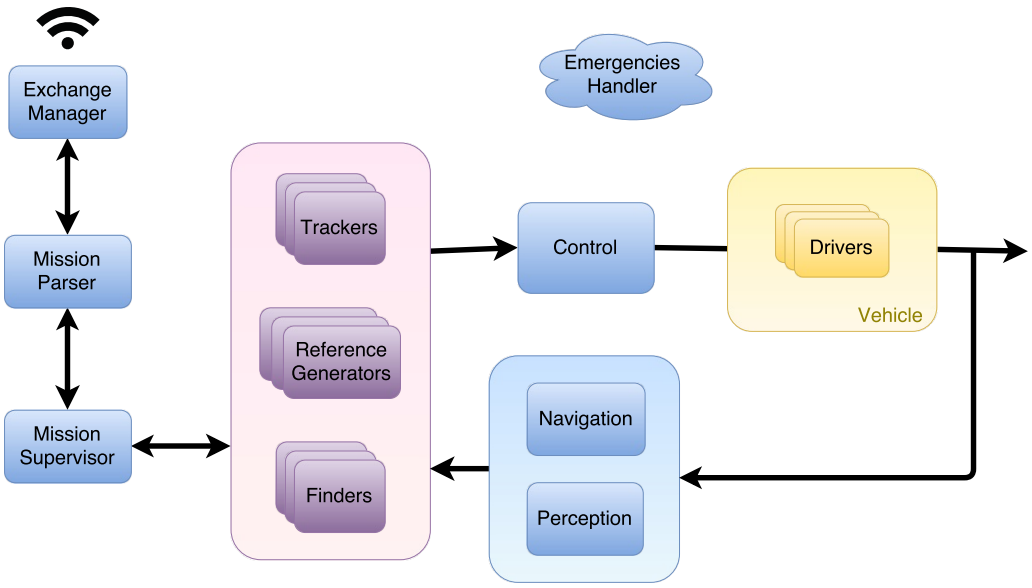


Figura 3.1: Schema software

cipale di questo lavoro di tesi: la "portabilità". Dando allo sviluppatore infinite possibilità di adattamento a esigenze diverse.

Nel grafo in figura 3.1 è rappresentata la struttura del software sviluppato sotto forma di anello di controllo per dare un idea del flusso di dati. Questo schema non è comunque vincolante: ogni nodo ha sempre accesso ai dati scambiati da tutti gli altri nodi. Questi sono suddivisi nel seguente modo:

Drivers - Nodi che fungono da interfaccia diretta con il veicolo,

Trackers, Refgens e Finders - Nodi dedicati allo svoglimento del comportamento assegnato,

Navigation - Nodo responsbile della stima dello stato del veicolo,

Control - Nodo che mette a disposizione dei controllori PID su ogni grado di libertà del veicolo,

Mission Supervisor - Nodo che decide in che modo eseguire ogni comportamento,

Mission Parser - Nodo che permette di eseguire un'intera missione descitta in un file,

Emergencies Handler - Nodo che identifica situazioni di criticità e prende provvedimenti di emergenza,

Perception - Nodo che elabora dati provenienti da sensori capaci di dare informazioni sull'ambiente esterno,

Exchange Manager - Nodo che permette la comunicazione con il resto del mondo.

Tutto il codice è stato sviluppato nei linguaggi C++ e Python e funziona in ambienti GNU/Linux.

3.1 Gestione dispositivi di bordo

In ambito sottomarino ci sono pochissimi standard. Ogni dispositivo utilizza il proprio protocollo di comunicazione e la propria interfaccia. Per questa ragione, per ogni veicolo, è necessario sviluppare il software che dialoga con i dispositivi di bordo. Il pacchetto "drivers" contiene quindi diversi nodi, uno per dispositivo utilizzato, sviluppati per i veicoli che sono stati testati durante la sperimentazione. Lo sviluppatore ha la libertà di creare i propri nodi ROS a piacere, seguendo delle semplici convenzioni:

- Utilizzare il "server di parametri" di ROS per le personalizzazioni
- Utilizzare una nomenclatura predefinita per i parametri

- Utilizzare i tipi di messaggi predefiniti
- Utilizzare una nomenclatura predefinita per i nomi delle topic

Dato che la grande maggioranza di sensori e attuatori utilizza il protocollo seriale, per semplificare il lavoro è stata creata una libreria di comunicazione seriale che consente di leggere e scrivere sulle porte di sistema in maniera semplificata.

È possibile fare una distinzione tra tre categorie di dispositivi: sensori e attuatori per la navigazione, che sono vitali per il veicolo e rimangono sempre attivi; strumenti di comunicazione, ad esempio modem acustici o radio; e "payload", ovvero strumenti atti a raccogliere dati sull'ambiente circostante che possono essere attivati o disattivati da file di missione.

3.2 Gestione di missione e controllo

Questa è la parte che consente al veicolo di eseguire una serie di operazioni impartite dal livello superiore. Se il veicolo è dotato di più elaboratori questa parte di software viene eseguito solitamente sul computer più stabile, perché i nodi descritti in questa sezione insieme ai drivers di navigazione, sono vitali per il corretto funzionamento del sottomarino.

3.2.1 Mission Supervisor

Mission Supervisor è il nodo che gestisce le operazioni elementari del veicolo, funge da nodo di smistamento di tutte i parametri di un dato comportamento e determina quando passare da un task ad un altro. Questo viene fatto attraverso la gestione diretta di altre tre tipologie di nodi: i "Reference Generators", i "Trackers" e i "Finders".

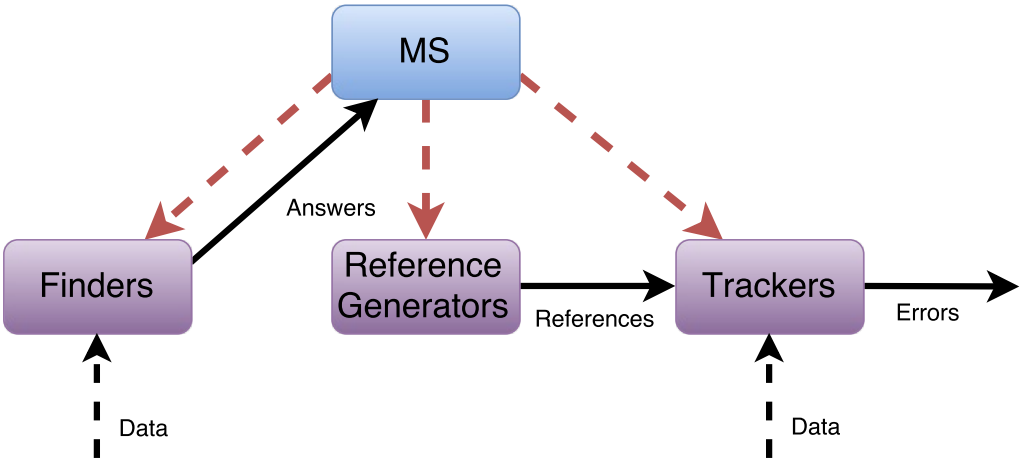


Figura 3.2: Funzionamento del nodo Mission Supervisor

Come vedremo successivamente Mission Supervisor (MS) ha a disposizione tre set di nodi, il suo compito è abilitare o disabilitare quelli necessari all'esecuzione del task corrente trasferendogli i parametri necessari al loro funzionamento. Grazie a questa scelta si ottiene un nodo di complessità elevata ma che una volta realizzato non viene più modificato, ed un set di nodi molto semplici che hanno il vantaggio di poter essere riutilizzati in task diversi. In figura 3.2 è riportato lo schema di funzionamento di MS e dei nodi collegati. L'attivazione di questi ultimi avviene in contemporanea all'arrivo di un nuovo ordine: i Reference Generators pubblicano i riferimenti che il controllo deve inseguire, i Trackers leggono questi riferimenti e, confrontandoli con lo stato del veicolo, generano gli errori di inseguimento, i Finders monitorano lo stato e pubblicano una risposta booleana che MS interpreta al fine di determinare se il task corrente è concluso. Il nodo funziona operando i seguenti passaggi consecutivi:

1. Riceve dall'alto livello l'ordine di eseguire un comportamento,

2. Controlla che lo stato del veicolo sia adeguato per iniziare,
3. Leggendo un file di configurazione ottiene i dettagli del comportamento,
4. Attiva i Reference Generators, i Trackers e i Finders necessari,
5. Utilizza le risposte dei Finders per passare da un Task ad un altro,
6. A comportamento compiuto comunica l'evento all'alto livello e si mette in attesa di una nuovo ordine.

Il file di configurazione di MS è scritto in YAML, un formato per la serializzazione di dati di semplice comprensione da parte dell'essere umano, questo utilizza l'indentazione e dei caratteri speciali quali ":", "-" e "|", per ordinare i dati in strutture. Questo file serve per "mappare" tutte le tipologie di comportamenti che il veicolo può compiere. Ognuno di questi comportamenti contiene la lista di task elementari che lo compongono, in ogni task sono definiti i nodi da utilizzare, i parametri da impostare e le regole per passare al task successivo.

Per comprenderne meglio il funzionamento si riporta una parte di un file di configurazione di MS:

```
---
- name: waypoint_depth
  tasks:
   - name: goToDepth
     finders:
      - name: depth
        manualUid: "depth0"
        params:
```

```
9           - type: msgparam
10            params: ["MP:lla_pos/reference"]
11          - type: function
12            name: constrain
13            params: ["MP:pos_tolerance/z", 0.5, 10.0]
14      referenceGenerators:
15       - name: rpy
16         params:
17          - type: const
18            params: [0, 0]
19          - type: topic
20            params:
21            - "/nav_status::NavStatus|Euler::orientation.yaw"
22          - type: const
23            params: [0]
24       - name: llhp
25         params:
26          - type: topic
27            params:
28            - "/nav_status::NavStatus|Position::position.lat"
29            - "/nav_status::NavStatus|Position::position.lon"
30       - name: depth
31         params:
32          - type: msgparam
33            params: ["MP:lla_pos/reference", 0.3, 100.0]
34      trackers: [rpy, ll, depth]
35      jumpRules:
```

```
36        - rule: "_depth0"
37          behavior: waypoint_depth
38          submission: goToWP
39          priority: 1
40  ---
```

Questo esempio mostra un singolo task di nome “goToDepth” appartenente al comportamento “waypoint_depth”. Nel corpo di ogni task ci sono 5 campi: il primo è il nome che serve per identificarlo all’interno della struttura, dopodiché ci sono tre campi contenenti la lista dei nodi da attivare suddivisi per categoria ed infine le “jumpRules”.

Per quanto riguarda i Finders ed i Reference Generators è necessario specificare nome e parametri, i primi in aggiunta richiedono anche un ID univoco (manualUid) perché è consentito attivarne più di uno dello stesso tipo in contemporanea. I parametri possono essere di quattro tipi:

const - Valori costanti specificati direttamente nel file

msgparam - Parametri da ottenere dall’ordine ricevuto che contiene le informazioni del comportamento, ad esempio

```
1  ---
2  - type: msgparam
3    params: ["MP:lla_pos/reference"]
4  ---
```

topic - Parametri da ottenere da una topic a cui MS è registrato, bisogna specificare il tipo di messaggio, il campo desiderato ed eventualmente il sotto-campo, ad esempio

```
---
- type: topic
  params:
  - "/nav_status::NavStatus|Position::position.lat"
  - "/nav_status::NavStatus|Position::position.lon"
---
```

funtion - Che permette di utilizzare semplici funzioni definite in MS, ad esempio "constrain" confina il valore in un determinato intervallo

```
---
- type: function
  name: constrain
  params: ["MP:pos_tolerance/z", 0.5, 10.0]
---
```

oppure "getHeadingTo" calcola l'angolo di heading tra un punto ed un altro:

```
---
- type: function
  name: getHeadingTo
  params:
  - "MP:pre_lla_pos/latitude"
  - "MP:pre_lla_pos/longitude"
  - "MP:lla_pos/latitude"
  - "MP:lla_pos/longitude"
---
```

Per quanto riguarda i Trackers invece si ha una semplice lista, in quanto non richiedono parametri per il loro funzionamento.

In jumpRules sono contenute le regole che definiscono le condizioni di superamento del task, possono essere più di una e nel campo "rule" viene espressa una funzione logica che utilizza le risposte dei Finders, quando una condizione risulta vera MS passa al comportamento e task corrispondente. Nel caso si abbiano più regole soddisfatte nello stesso momento MS scegli quale utilizzare in base al campo "priority". In questo esempio, più articolato del precedente, sono definite due regole: nella prima devono essere soddisfatte due condizioni contemporaneamente, "rpy0" e "depth0", nella seconda soltanto "timer0" con una priorità più alta.

```
---
jumpRules:
- rule: "_rpy0&&_depth0"
  mission: pfline_altitude
  submission: goToAltitude
  priority: 2
- rule: "_timer0"
  mission: go_to_surface
  submission: hp
  priority: 1
---
```

Questo modo di organizzare e pianificare i comportamenti risulta molto versatile offrendo molte possibilità di personalizzazione, allo stesso tempo però modificare questo file può risultare macchinoso e ripetitivo. Per evitare che eventuali errori di sintassi vadano a inficiare il corretto funzionamento del veicolo è stato programmato in MS un check del file al suo avvio, in modo da bloccare l'esecuzione in presenza di errori. In più, per rendere la scrittura di

questo file più agevole, è stata sviluppata una utility che permette di leggere e scrivere la configurazione attraverso una comoda interfaccia grafica (Figura 3.3).

Reference Generators

In generale, nei sistemi di controllo, l'obiettivo principale è quello di condurre lo stato del sistema (o una sua parte) ad un valore desiderato, in genere variabile nel tempo. Questo segnale viene chiamato "riferimento" ed il suo studio è importante perché influisce sulle caratteristiche dell'uscita del sistema di controllo. I Reference Generators (RG) sono i nodi che si occupano di calcolare, istante per istante, i riferimenti di cui il controllo ha bisogno.

In tutto il sistema mette a disposizione dodici gradi di libertà su cui operare il controllo in modo indipendente: le posizione sui tre assi x, y e z, le rotazioni intorno agli stessi assi e le corrispettive velocità lineari e angolari. Ovviamente non tutti i gradi di libertà sono sempre attivi, è compito di MS scegliere per ogni task i RG e i trackers da utilizzare, e di conseguenza i gradi di libertà sui quali operare il controllo.

Per ovvie ragioni non possono essere contemporaneamente attivi due RG che generino riferimenti per lo stesso grado di libertà, ma possono coesistere per essere attivati in task differenti. È inoltre permesso che un solo RG generi più di un riferimento e che siano attivi allo stesso tempo più di un RG con riferimenti di tipo diverso.

Ogni nodo di questo tipo deve estendere una classe base che gestisce automaticamente l'attivazione/disattivazione e mette a disposizione i parametri forniti da MS (Listato A.1).

Figura 3.3: Utility per la creazione del file di configurazione di MS

Trackers

I nodi di tipo Tracker confrontano lo stato del veicolo con i riferimenti ottenuti dai RG per calcolare l'errore di inseguimento e la sua derivata discreta rispetto al tempo. Da sottolineare che gli errori di posizione e velocità devono essere sempre espressi in terna body, o meglio per le posizioni l'errore va inteso come distanza tra l'origine della terna solidale e il riferimento, mentre per le velocità ci si riferisce a quella del veicolo in terna fissa espressa però in terna body.

Anche in questo caso i nodi vengono creati estendendo una classe base che si occupa di gestire l'attivazione/disattivazione dello stesso (Listato A.2), per questo tipo di nodi non sono previsti parametri.

Finders

Il compito dei nodi Finders è molto semplice: verificano se una condizione è vera o falsa. Anche questi nodi vengono attivati o disattivati da MS attraverso un messaggio e, una volta attivati, ad intervalli di tempo regolari inviano un altro messaggio con una risposta booleana.

L'unica complicazione è dovuta al fatto che più di un Finder dello stesso tipo può essere attivo con parametri diversi, questo è utile ad esempio nel caso dei timer: si può avere la necessità di avviare due o più timer in tempi diversi ed agire in maniera diversa ciascuno. Il problema è stato risolto utilizzando una lista dinamica di thread che vengono costruiti o distrutti all'occorrenza, si identificano grazie ad un ID unico assegnato manualmente nel file di configurazione di MS. Il codice della classe base da reimplementare viene riportato in appendice (Listato A.3).

3.2.2 Navigation

Nelle applicazioni sottomarine il problema della "localizzazione" è di vitale importanza: l'acqua è un ottimo schermo per le onde elettromagnetiche, e quindi, sotto la superficie, è impensabile l'utilizzo di dispositivi come ad esempio il GPS o il radar che semplificano notevolmente il problema negli altri ambiti. L'unico vantaggio che fornisce l'acqua rispetto all'aria consiste in una migliore trasmissione delle onde acustiche, e la maggior parte dei dispositivi che vengono utilizzati in questo campo sono quindi di tipo acustico. Dato l'alto livello tecnologico richiesto e data la relativamente bassa diffusione, questi dispositivi risultano spesso molto costosi. Si trovano anche in commercio strumenti assemblati che promettono, grazie all'alto grado di raffinatezza e integrazione, di fornire una stima di posizione affidabile senza bisogno di ulteriore elaborazione di dati e senza la conoscenza delle caratteristiche del veicolo su cui vengono montati. Ovviamente però questa "comodità" va di pari passo con dei costi che spesso non sono congrui al target del progetto, risulta spesso necessario utilizzare algoritmi più o meno elaborati, che permettano di stimare lo stato del veicolo con precisione adeguate all'applicazione.

Al nodo in cui vengono implementati questi algoritmi è stato dato nome "Navigation", cioè la parte di software che legge i sensori di navigazione e genera una stima di posa del veicolo. La volontà di creare un software orientato alla ricerca e allo sviluppo ha portato a fare uno sforzo ulteriore per rendere questo nodo un "contenitore" di algoritmi di stima che possono essere ideati e implementati dallo sviluppatore. Anche in questo caso si fa uso di una classe che presenta le funzioni base da implementare nel proprio filtro, il codice viene riportato in appendice (Listato A.4).

Di seguito si descrivono i principali sensori di navigazione che solitamente

vengono usati sugli AUV.

Accelerometri Un accelerometro misura l'accelerazione del corpo rigido su cui è montato solidarmente, a cui è sommato l'opposto dell'accelerazione di gravità ($\vec{g} = \begin{bmatrix} 0 & 0 & 9,81 \end{bmatrix}^T$ m/s nel sistema di riferimento fisso) espressa nel sistema di riferimento solidale. L'equazione caratteristica può essere espressa come segue:

$$\vec{a}^{\,meas} = J_1^T \left(\ddot{\vec{\eta}}_1 - \vec{g} \right) + \vec{b}_a + \vec{\delta}_a \tag{3.1}$$

con $\vec{b}_a$ e $\vec{\delta}_a$ rispettivamente il bias e il rumore di misura (Gaussiano a media nulla).

Giroscopi Per quanto riguarda i giroscopi, invece, misurano la velocità angolare del veicolo nel sistema di riferimento solidale e la loro equazione caretteristica è:

$$\vec{\nu_2}^{meas} = \vec{\nu_2} + \vec{b}_{\nu_2} + \vec{\delta}_{\nu_2} \tag{3.2}$$

con $\vec{b}_{\nu_2}$ e $\vec{\delta}_{\nu_2}$ rispettivamente il bias e e il rumore di misura (Gaussiano a media nulla). Se il bias dell'accelerometro è più o meno stabile, dopo un periodo iniziale transitorio, quello del giroscopio non lo è, quindi deve essere stimato online per ridurre al minimo l'errore che ne deriva. I giroscopi più precisi sono fatti in fibra ottica e prendono il nome di FOG (Fiber Optics Gyroscope). Il funzionamento è basato sull'effetto Sagnac presente nell'interferometro di Sagnac. Se da una parte la loro maggior precisione si traduce in un ridotto rumore e un bias più stabile, dall'altro lato nell'output del sensore viene introdotto un contributo dovuto alla rotazione terrestre che deve essere compensata.

Le equazioni 3.1 e 3.2 sottolineano che un sistema di navigazione basata solo su sensori inerziali è intrinsecamente soggetto a errori di deriva illimitati

causati dai bias dei componenti che, anche se identificati e calibrati, non possono essere completamente compensati. In particolare la seguente tabella riassume questo aspetto:

Errori dovuti al giroscopio	Errori dovuti all'accelerometro
$\delta\vec{v} = \frac{1}{2}\vec{e}_g t^2$	$\delta\vec{v} = \vec{e}_a t$
$\delta\vec{p} = \frac{1}{6}\vec{e}_g t^3$	$\delta\vec{p} = \frac{1}{2}\vec{e}_a t^2$

(3.3)

dove $\delta\vec{v}$ e $\delta\vec{p}$ rappresentano rispettivamente la velocità e gli errori di posizione stimati e $\vec{e}_a$ e $\vec{e}_g$ sono gli errori residui sulle misure.

Magnetometri Questo sensore fornisce una misura del campo magnetico locale espressa in terna solidale. Definendo il campo magnetico terreste locale in terna fissa come un vettore costante $\vec{m}^{NED} = \begin{bmatrix} m_x^{NED} & m_y^{NED} & m_z^{NED} \end{bmatrix}^T$, l'equazione caratteristica del sensore, in assenza di disturbi, è:

$$\vec{m}^{meas} = C_{si}\left(J_1(\vec{\eta}_2)\vec{m}^{NED}\right) + \vec{b}_m + \vec{\delta}_m \tag{3.4}$$

Anche in questo caso il termine $\vec{\delta}_m$ è l'errore di misura (Gaussiano a media nulla). Il termine $\vec{b}_m$ è il bias costante sulla misura ma, per il magnetometro, non è generalmente introdotto dal sensore stesso ma dipende dai così detti "Hard Iron Effects", distorsioni create da oggetti che producono un campo magnetico. La matrice C_{si} è usata per modellare i "Soft Iron Effects", comunemente causati da metalli come nichel e ferro, che introducono distorsioni, deflessioni e alterazione nel campo magnetico esistente che viene allungato e schiacciato in base a che direzione il campo agisce sul sensore. Questi disturbi possono essere identificati e compensati da una fase di taratura preliminare dedicata.

Sensore di profondità I sensori usati per misurare la profondità nei veicoli sottomarini sono sensori di pressione. La pressione locale viene convertita in una misura di profondità in accordo alla relazione idrostatica:

$$p - p_{atm} = \rho g d \qquad \Rightarrow \qquad d = \frac{p - p_{atm}}{\rho g} \tag{3.5}$$

dove:

- p è la pressione locale misurata dal sensore;
- p_{atm} è la pressione atmosferica locale sulla superficie;
- g è la norma dell'accelerazione di gravità;
- d è l'altezza della colonna d'acqua sopra lo strumento, quindi la profondità.

La misura di pressione p^{meas} è caratterizzata dalla seguente equazione:

$$p^{meas} = p + b_p + \delta_p \tag{3.6}$$

con b_p e δ_p rispettivamente il bias e il rumore dello strumento (Gaussiano a media nulla); l'equazione caratteristica della misura di profondità risulta:

$$z^{meas} = z + \delta_z \tag{3.7}$$

con δ_z errore di misura (Gaussiano a media nulla); non c'è da considerare un bias in questa misura di profondità perché è ottenuta grazie ad una differenza di due pressioni (locale e atmosferica) ottenute dallo stesso sensore.

Doppler Velocity Log Un DVL (Doppler Velocity Log) è un sensore di velocità basato sull'effetto Doppler che interessa un segnale acustico trasmesso da un oggetto in movimento. In particolare, il DVL viene comunemente

montato sulla parte inferiore dei veicoli sottomarini diretto verso il fondale. Il dispositivo trasmette un segnare acustico e, analizzando l'effetto Doppler presente sull'eco che ritorna dal fondale, calcola la misura di velocità del veicolo (in tre dimensioni) rispetto al sistema di riferimento fisso, ma espresso il terna solidale. L'equazione caratteristica che descrive le misure fornite è la seguente:

$$\vec{\nu_1}^{meas} = \vec{\nu_1} + \vec{b}_{\nu_1} + \vec{\delta}_{\nu_1} \tag{3.8}$$

con $\vec{b}_{\nu_1}$ e $\vec{\delta}_{\nu_1}$ rispettivamente il bias e il rumore di misura (Gaussiano a media nulla). La particolarità del DVL è che il suo errore di misura $\vec{\delta}_{\nu_2}$ è caratterizzato da una covarianza non costante, dipendente dalla norma della velocità misurata.

Global Positioning System Il GPS, anche se non funziona sott'acqua, è equipaggiato su tutti i veicoli subacquei moderni. È solitamente utilizzato per l'inizializzazione dei filtri di navigazione e per resettare l'errore quando il veicolo raggiunge la superficie. La posizione misurata è fornita rispetto al sistema di coordinate geografiche in latitudine e longitudine; per poter esplicitare le misure ottenute all'interno di un filtro di navigazione risulta necessario convertirle per determinare le corrispondenti coordinate in terna fissa. Esistono delle funzioni standard per operare questa conversione. Nominando "$LL2NE$" la funzione generale usata per la conversione e $\vec{ll}^{\,meas}$ la misura data dal GPS, l'equazione caratteristica del sensore può essere espressa come:

$$\begin{bmatrix} x \\ y \end{bmatrix} = LL2NE\left(\vec{ll}^{\,meas} + \vec{\delta}^{\,ll}, \vec{O}^{n}_{ll},\right) \tag{3.9}$$

dove $\vec{\delta}^{\,ll}$ è il rumore di misura (Gaussiano a media nulla). A causa della non-linearità della funzione $LL2NE$, la risultante misura di rumore su $\begin{bmatrix} x & y \end{bmatrix}^T$

non è caratterizzata dalle stesse proprietà di $\vec{\delta}^{ll}$.

3.2.3 Control

Control è il nodo centrale del controllo, all'interno vi sono implementati 12 controllori di tipo PID (Proportional Integrative Derivative) indipendenti sui seguenti gradi di libertà:

- 3 posizioni lungo gli assi x^b, y^b e z^b della terna solidale al veicolo,
- 3 rotazioni del veicolo espresse secondo gli angoli di eulero ϕ, θ e ψ,
- 3 velocità lineari del veicolo espresse in terna body u, v e w,
- 3 velocità angolari p, q e r intorno agli assi della terna solidale.

PID Si introducono le caratteristiche di un semplice controllore PID per un generico sistema: si consideri un generico sistema P(s) da controllare attraverso un blocco controllore C(s) come mostrato in figura 3.4.

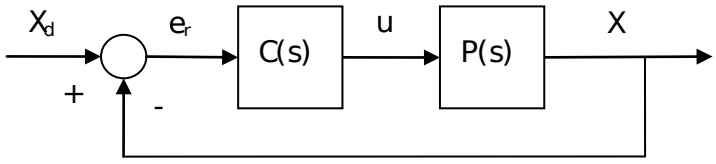


Figura 3.4: Schema di controllo in retroazione

La funzione di trasferimento di un controllore PID si presenta in generale sotto la forma:

$$K_P + \frac{K_I}{s} + K_D s = \frac{K_D s^2 + K_P s + K_I}{s} \tag{3.10}$$

La variabile e_r in figura 3.4 rappresenta l'errore tra l'uscita desiderata e l'uscita effettiva del sistema; tale segnale è a sua volta l'ingresso del controllore PID, il quale genera un segnale di uscita u composto dalle tre componenti, proporzionale, integrale e derivativa:

$$\vec{u} = K_P\vec{e_r} + K_D\dot{\vec{e}_r} + K_I\int_0^t \vec{e_r}(T)dT \tag{3.11}$$

Il contributo proporzionale ha l'effetto di migliorare la prontezza della risposta e di ridurre, ma non di eliminare, l'errore a regime permanente; la parte integrale influisce sull'errore a regime permanente eliminandolo, ma peggiora la risposta transitoria; l'azione derivativa ha invece l'effetto di aumentare la stabilità del sistema migliorando la risposta in regime transitorio.

Coordinate La particolare scelta di operare il controllo di posizione in coordinate "body" è dovuta al fatto che, non essendo la dinamica del veicolo isotropa (sopratutto nei veicoli di forma affusolata), si ha un comportamento diverso lungo le direzioni della terna fissa a seconda della sua orientazione attuale; scegliendo invece di operare il controllo in terna solidale al veicolo, rispetto agli assi di questa terna, il comportamento dinamico è sempre il medesimo, cosa fondamentale per la scelta dei parametri dei controllori PID. Questa scelta da luogo però una piccola complicazione: molto spesso risulta comodo esprimere i riferimenti in terna fissa, ad esempio il riferimento di profondità o di latitudine e longitudine, in questi casi si opera un cambio di coordinate (da terna fissa a body) con lo svantaggio però che un segnale di riferimento in coordinate fisse interessa più controllori in coordinate solidali. Nel caso ci siano più contributi agenti sugli stessi assi (cosa plausibile dopo il cambio di coordinate) i contributi vengono sommati, a patto che siano

linearmente indipendenti. La verifica di indipendenza viene effettuata nel seguente modo: per ogni coppia $\vec{e}_1$, $\vec{e}_2$ deve valere la seguente relazione:

$$\|\vec{e}_1 \times \vec{e}_2\| \leq \epsilon \tag{3.12}$$

dove ϵ è una tolleranza da scegliere in base alla precisione desiderata.

I guadagni dei controllori pid vengono ottenuti attraverso il server dei parametri di ROS e possono essere modificati mentre il sistema è in esecuzione. Ogni volta che uno o più parametri vengono modificati il controllo viene resettato.

Compensazione di gravità e galleggiamento In aggiunta ai controllori PID, se si ha una buona conoscenza delle caratteristiche del veicolo, questo nodo permette di generare un segnale di controllo che compensa le forze di gravità e galleggiamento definite in 2.10 come $\vec{g}(\vec{\eta})$. Di conseguenza le azioni generate dal controllo diventano:

$$\tau = \tau_{PID} - \vec{g}(\vec{\eta}) \tag{3.13}$$

3.3 Pianificazione ed esecuzione delle missioni

Per "Missione" si intende una successione di comportamenti che il veicolo deve effettuare con successo. Al momento le capacità decisionali affidate al sistema riguardano solo lo svolgimento dei singoli comportamenti, non il passaggio dall'uno all'altro che avviene secondo una semplice sequenza. Questa sequenza viene definita dall'utilizzatore per mezzo di un file di missione e il nodo che si occupa di interpretare questo file prende il nome di "Mission Parser".

Analogamente al file di configurazione di MS, anche quello di missione è in formato yaml e se ne riporta un esempio:

```
mission:
  - name: waypoint_depth
    devices: {}
    lla_pos: {latitude: 42.99571, longitude: 10.49856, reference: 0}
    pos_tolerance: {x: 3, y: 3, z: 0.3}
    ori_tolerance: {x: 0.5, y: 0.5, z: 0.25}
    reference: {x: 0.7, y: 0, z: 0}
    thrust: 0
  - name: lawn_mower_altitude
    devices: {sss: 1, tv: 1}
    lla_pos: {latitude: 42.99564, longitude: 10.49870, reference: 3}
    length: 100
    width: 50
    spacing: 10
    direction: 90
    pos_tolerance: {x: 5, y: 5, z: 0.5}
    ori_tolerance: {x: 0.5, y: 0.5, z: 0.25}
    reference: {x: 0.5, y: 0, z: 0}
    thrust: 0
  - name: waypoint_altitude
    devices: {sss: 0, tv: 0}
    lla_pos: {latitude: 42.99619, longitude: 10.49786, reference: 3}
    pos_tolerance: {x: 5, y: 5, z: 0.5}
    ori_tolerance: {x: 0.5, y: 0.5, z: 0.25}
    reference: {x: 0.7, y: 0, z: 0}
    thrust: 0
  - name: waypoint_depth
```

```
28      devices: {}
29      lla_pos: {latitude: 42.99617, longitude: 10.4978, reference: 0}
30      pos_tolerance: {x: 5, y: 5, z: 0.5}
31      ori_tolerance: {x: 0.5, y: 0.5, z: 0.25}
32      reference: {x: 0.7, y: 0, z: 0}
33      thrust: 0
```

All'interno di questo file è presente un vettore "mission" i cui elementi corrispondono a comportamenti da eseguire in sequenza per portare a termine la missione. Tutti i dati sono espressi sotto forma di "dizionari", cioè strutture di dati in cui ad ogni valore è associata un etichetta detta "key" siano essi numeri, stringhe o dizionari stessi. In ogni elemento deve essere presente la chiave "name", necessario al nodo per riconoscere il tipo di comportamento da utilizzare. A seconda del nome specificato Mission Parser può operare in due modi:

- **Trasparente** - in cui il nodo non effettua alcuna operazione se non quella di trasferire direttamente le informazioni nel messaggio di richiesta a MS.
- **Interpretato** - in cui i dati vengono elaborati da una funzione definita nel codice del nodo, generando uno o più comportamenti diversi da far compiere al veicolo.

Nel primo caso sia il nome che i dati forniti devono corrispondere ad un comportamento definito nel file di configurazione di MS, in particolare il campo "name" deve coincidere al nome di un comportamento e i dati all'interno del messaggio sono descritti concatenando il nome dei campi nei quali sono contenuti con un simbolo "/" (ad esempio *lla_pos/latitude*). Questa funzione sarebbe già sufficiente a svolgere gran parte delle operazioni, richiede però

uno sforzo notevole da parte dell'utente per la stesura di file che possono essere anche molto lunghi e ripetitivi; inoltre, dato che il più delle volte la missione viene trasferita al veicolo tramite il canale radio, è bene ridurre il numero di bytes da trasmettere senza però perdere informazioni importanti. Per questo è stata implementata in Mission Parser la possibilità di inserire funzioni speciali che consentono di interpretare i dati presenti nel file di missione generando automaticamente dei comportamenti da eseguire in sequenza, come se fossero stati scritti a mano nel file stesso. Un esempio molto chiaro è quello dove si vuol far compiere al veicolo un percorso dato da un pattern predefinito (nel file esempio precedente un "lawn_mower", cioè una greca), scrivere punto per punto la traiettoria significherebbe ripetere molte volte le stesse righe con coordinate diverse, in questo modo invece con poche informazioni si generano automaticamente tutti i punti del percorso.

A Mission Parser è affidato un ulteriore compito: attivare o disattivare i dispositivi hardware non vitali. Ci possono essere vari motivi per cui fare questa operazione, ad esempio se un sensore richiede parecchia potenza si potrebbero ottimizzare i consumi spegnendolo quando non è necessario, oppure ci possono essere dei dispositivi che interferiscono tra loro (molto comune utilizzando sensori acustici) e si vogliono utilizzare in momenti differenti. Per fare questo basta inserire nel campo "devices" il nome identificativo del dispositivo e un booleano che corrisponde allo stato di attivazione, il nodo provvede a comunicare la variazione dello stato ai nodi drivers interessati.

3.4 Gestione delle emergenze

In ambiente subacqueo evitare gli incidenti e reagire tempestivamente ai malfunzionamenti è di vitale importanza: la scarsa visibilità e la difficoltà di in-

tervento da parte dell'uomo, soprattutto in caso di veicoli autonomi, rendono proibitivi il ritrovamento e recupero di un veicolo disperso, considerando inoltre i costi medi della componentistica di bordo, si giunge alla conclusione che va riservata un'attenzione particolare al tema dell'identificazione e gestione delle emergenze. Inoltre in alcune applicazioni come ad esempio la ricerca archeologica, la salvaguardia dei siti d'interesse diventa addirittura prioritaria rispetto al veicolo stesso.

Il nodo "Emergency Handler" si occupa quindi di verificare il corretto funzionamento del veicolo monitorando gli altri nodi e gestendo le situazioni critiche in modo da minimizzare i rischi per il veicolo e per l'ambiente circostante.

3.4.1 Identificazione

Il primo passo consiste nel monitorare costantemente il sistema alla ricerca di eventuali problematiche che possono venire rivelate in tre metodi diversi:

- controllando se alcuni messaggi vengono pubblicati con regolarità per rilevare blocchi o crash dei nodi interessati; Emergency Handler si sottoscrive alle topic con il solo scopo di verificarne la pubblicazione e, per ognuna, crea un countdown da resettare ad ogni nuova ricezione del messaggio, quando il conteggio finisce viene inviato un messaggio di warning e se il problema persiste scatta la contromisura

- monitorando alcuni dati del sistema che, per il corretto funzionamento, devono rimanere entro un determinato range, ad esempio il livello delle batterie oppure la distanza da un ostacolo frontale; se un parametro rimane fuori dalle proprie soglie per più di un tempo limite scatta la contromisura corrispondente

- ricevendo segnalazioni di emergenza da altri nodi: è stato infatti creato un sistema che permette ad ogni nodo, in ogni momento, di segnalare dei problemi a Emergency Handler, mettendolo in grado di prendere le adeguate contromisure.

3.4.2 Contromisure

Dopo aver individuato la presenza di una o più emergenze il nodo deve decidere quale contromisura adottare per far fronte al problema in modo più efficiente. Sono state definite quattro tipologie di contromisura ordinate da un indice di priorità in modo che, nel caso si verifichino due o più emergenze concorrenti, il nodo sia comunque in grado di prendere una decisione univoca. Le contromisure, ordinate per priorità crescente, sono:

1. **Controlled surface** - il veicolo raggiunge la superficie in maniera controllata e mantiene la direzione corrente; è l'unica ad essere reversibile anche senza l'intervento dell'operatore, se il motivo per cui è scattata cessa di esistere viene annullata anche la contromisura

2. **Surface** - il veicolo utilizza la spinta delle sue eliche verticali, se disponibili, per riemergere rapidamente; viene bypassato il controllo intervenendo direttamente sul driver dei motori

3. **Motor OFF** - i motori vengono spenti e il veicolo, essendo positivo in acqua, riemerge lentamente

4. **Drop** - se disponibili, oltre allo spegnimento dei motori, vengono rilasciate le zavorre di sicurezza per una riemersione più rapida

In ogni caso al verificarsi di un emergenza, se il veicolo sta effettuando una missione questa viene interrotta e può essere ripresa se lo stato torna

alla normalità. Alla contromisura viene sempre associata una stringa di testo contenente una sintetica descrizione delle cause che l'hanno generata, in modo che l'utente, anche da remoto, possa avere una situazione chiara di cosa sta succedendo a bordo ed intervenire di conseguenza.

3.4.3 Configurazione

Ovviamente tutte le funzioni che sono state descritte possono essere, in linea con gli obiettivi di questo lavoro, configurate dall'utente. Come nei casi precedenti è stato scelto il formato YAML per la semplicità di utilizzo e di lettura.

La configurazione di Emergency Handler è divisa in due file. Il primo contiene tutti i controlli temporizzati, in altre parole questo file descrive quali sono le topic che il nodo deve monitorare costantemente. Il primo tipo di timer controlla solamente che una topic venga pubblicata senza interruzioni prolungate:

```
---
- device: "imu"
  enable: 1
  code: "a001"
  alert_timeout: 5.0
  emergency_timeout: 10.0
  emergency_countermeasure: "SURFACE"
  topic_name: "drivers/imu"
---
```

in questo esempio viene verificata la pubblicazione continuativa della topic "drivers/imu" che, se interrotta per più di 10 secondi, genera la contromisura

"Surface". La sintassi è simile nell'altro caso, in cui si verifica che un dato non esca da un interfallo scelto:

```
---
- device: altitude_limit
  enable: 1
  code: "c004"
  upper_limit: 100.0
  lower_limit: 1.5
  alert_timeout: 3.0
  emergency_timeout: 5.0
  emergency_countermeasure: "SURFACE"
  topic_name: "drivers/altitude"
---
```

ad ognuno di questi elementi è associato un oggetto instaziato nel codice del nodo, che deve essere di tipo "watcher" (listato A.5).

Il secondo file di configurazione, invece, rappresenta una mappa di tutte le segnalazioni che possono pervenire dagli altri nodi. Ad esempio:

```
---
- node: "elmo"
  code:
    - name: c1
      countermeasure: "CONTROLLED_SURFACE"
      explanation: "BATTERY LEVEL critical"
- node: "ioboard"
  code:
   - name: c1
     countermeasure: "CONTROLLED_SURFACE"
```

```
11      explanation: "AWASH detected by on-board sensors"
12  ---
```

devono essere specificati il nodo di origine della segnalazione, il tipo di contromisura da adottare e una spiegazione sintetica del problema.

3.5 Percezione

Fin adesso si sono consideratoìi i dispositivi di tipo payload come oggetti presenti a bordo veicolo che consentono, una volta attivati, di raccogliere dati utili all'utente in modo da essere elaborati successivamente. Anche se questo è vero per la maggioranza delle situazioni, in generale è possibile utilizzare le informazioni sull'ambiente esterno per influenzare la missione, ad esempio con il riconoscimento di un oggetto di interesse o di un ostacolo. Il nodo "Perception" si occupa di fornire questo tipo di informazioni elaborando i dati forniti dai nodi drivers operando esternamente a questi ultimi, in modo da massimizzare la modularità. Analogamente a "Navigation" si tratta di un contenitore di algoritmi definiti dallo sviluppatore, solo che in questo caso è possibile utilizzarne anche più di uno in contemporanea.

La scelta di concentrare in un solo nodo l'elaborazione dei dati sull'ambiente esterno, leggermente in controtendenza con il resto del lavoro, è dovuta alla volontà di concentrare in una sola uscita tutti i risultati offerti da queste analisi in modo che siano utilizzabili indipendentemente dalla loro provenienza. Si prenda come esempio la localizzazione di un ostacolo: al software che controlla l'andamento della missione non interessa il modo in cui è stata ottenuta questa informazione, ma soltanto la posizione e le dimensioni dell'ostacolo stesso in modo da prendere le decisioni adeguate. Tutte le informazioni ottenute dagli algoritmi implementati in Perception vengono

quindi impacchettate e pubblicate su una topic dedicata in modo da essere disponibili in ogni momento per tutti gli altri nodi, in particolare per Finders, Trackers e, nel caso si volessero condividere con gli altri veicoli, anche Exchange Manager.

Ovviamente, per ottimizzare la richiesta di risorse (che sono generalmente elevate in questi casi), ogni algoritmo implementato nel nodo può essere attivato e disattivato, a seconda delle esigenze, da file di missione attraverso il campo "perception", in maniera analoga a quanto già descritto per i drivers. Per aggiungere un nuovo algoritmo di elaborazione dati, lo sviluppatore deve estendere la classe PerceptionAlgorithm (Listato A.6), che gestisce automaticamente la formattazione dell'output e l'attivazione, e assegnare un identificativo in modo che corrisponda a quello specificato nel file di missione.

3.6 Comunicazione

La comunicazione con l'esterno è un aspetto chiave nei veicoli sottomarini principalmente per due motivi: in primo luogo, quando il veicolo opera in immersione, i riscontri visivi da parte dell'uomo sono pressochè assenti, per cui è importante avere un continuo scambio di dati che permetta di monitorare la missione. Il secondo motivo è dovuto alla difficoltà nell'utilizzare i comuni mezzi di comunicazione sott'acqua, anche con le più moderne tecnologie non è ancora possibile, in questo ambiente, utilizzare le onde elettromagnetiche sott'acqua per scambiare dati. Si deve quindi ricorrere, anche in questo caso, a strumenti acustici. Questi ultimi però hanno il difetto di avere bande di trasmissione molto ridotte e una scarsa affidabilità a causa di riflessioni sulla superficie e su eventuali ostacoli. Le soluzioni adottate più frequentemente sono le seguenti:

- Comunicazione di tipo WLAN per un veloce scambio dati in superficie a distanze ridotte
- Comunicazione ad onde radio di frequenza ridotta per la comunicazione in superficie a grande distanza
- Comunicazione tramite modem acustici quando il veicolo è in immersione

Il primo metodo consente di accedere direttamente al computer del veicolo e non è necessario del software aggiuntivo, mentre negli altri due casi la comunicazione consente unicamente di scambiare dati "grezzi" che devono essere generati e interpretati dal software di gestione del veicolo.

Exchange Manager (EM) è il nodo che assolve quest'ultimo compito, facendo da tramite tra i nodi drivers di comunicazione e il sistema stesso. L'idea alla base consiste nell'utilizzare i nodi drivers in maniera "trasparente" di modo che, inviando un messaggio su una topic specifica di "outbox", questo stesso messaggio sia recapitato su una topic di "inbox" di un'altra utenza, connessa alla stessa rete indipendentemente dal canale di comunicazione.

Ridurre al minimo le dimensioni dei dati da scambiare è di fondamentale importanza, soprattutto nel caso di canale acustico, perché la velocità di trasmissione è bassa e c'è un alta percentuale di packet-loss. Per questa ragione si utilizza il formato binario, che risulta difficilmente leggibile dall'utente, ma minimizza la dimensione dei pacchetti; ogni messaggio ha la seguente struttura:

Header (2 bytes) - identifica l'inizio del messaggio, è identico per ogni messaggio scambiato

Length (1 byte) - lunghezza del campo payload in bytes (può variare da 0 a 255)

Command (1 byte) - identifica il tipo di messaggio

Payload (N bytes) - corpo del messaggio

Checksum (1 byte) - carattere di controllo per la verifica di integrità

Una mappatura degli identificativi di messaggio, definiti all'interno del codice di EM, permette di associare il contenuto del campo "Payload" alle informazioni di interesse.

3.6.1 Scambio di file

EM prevede anche il trasferimento di interi file, proprietà pensata in particolare per caricare a bordo i file di missione. Questi vengono compressi e suddivisi in piccole parti (chunk), in modo da poter essere inviati sfruttando il protocollo descritto precedentemente. Il procedimento prevede l'esecuzione in sequenza dei seguenti passi:

- [mittente → ricevente] Richiesta di trasferimento
- [mittente ← ricevente] Conferma di trasferimento
- [mittente → ricevente] Richiesta di invio chunk (ripetuto per tutti i chunk)
- [mittente ← ricevente] Conferma di chunk ricevuto
- [mittente → ricevente] Trasferimento completato
- [mittente ← ricevente] Conferma di trasferimento completato

nel caso il mittente non riceva, per qualsiasi motivo, una delle conferme entro un tempo predefinito, questo ripete il passo precedente per un certo numero di volte (anch'esso predefinito) aldilà del quale l'intera operazione di trasferimento fallisce.

3.7 Interfacce

Le attività in mare, teatro consono per i veicoli subaquei, non sempre sono confortevoli per chi opera con questi sistemi. Le avverse condizioni meteo, o il semplice dondolio di un imbarcazione risultano spesso fastidiosi per chi deve rimanere concentrato davanti allo schermo di un computer. È necessario semplificare al massimo questo tipo di operazioni ed evitare di operare attraverso linee di comando, utilizzando invece applicativi grafici che diano riscontri visivi immediati. Inoltre tali strumenti automatizzano una serie di operazioni che, condotte a mano, richiederebbero parecchio tempo. Sono state sviluppate due diverse interfacce per due diverse necessità: una che permette un controllo diretto sul veicolo e una per la pianificazione ed il monitoraggio delle missioni.

3.7.1 Navigazione e testing

Questa GUI (Graphical User Interface), grazie all'integrazione tra il sistema ROS e il framework QT [QT,], consente di operare in maniera diretta sul veicolo e richiede una connessione stabile di tipo LAN, ottenibile tramite cavo o WiFi (WLAN). Viene utilizzata principalmente per due scopi: guidare il veicolo come un ROV attraverso un comune joystick, oppure monitorare il veicolo per operazioni di test, in modo da avere tutti i dati sotto controllo in ogni momento. In Figura 3.5 è riportata la schermata principale. Le principali funzionalità dell'interfaccia sono:

- Visualizzazione di pitch, yaw, profondià e velocità sia grafica che testuale
- Visualizzazione della profondità del fondale

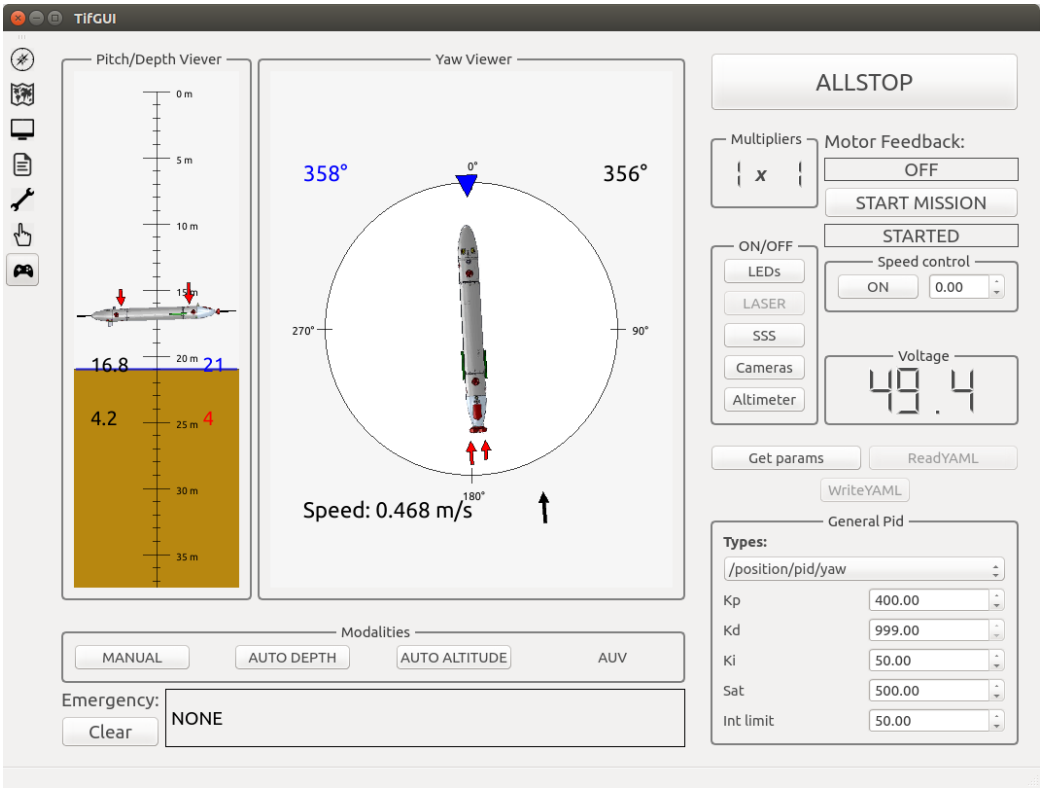


Figura 3.5: Interfaccia grafica di navigazione e testing

- Visualizzazione delle spinte dei motori
- Impostazione della modalità di navigazione (ROV, AUV)
- Lettura e scrittura dei parametri dei controllori
- Visualizzazione delle emergenze
- Comando di interruzione dell'alimentazione dei propulsori
- Monitoraggio della tensione delle batterie
- Visualizzazione di posizione su mappa geografica
- Visualizzazione grafica dei RG, Finders e Trackers attivi con relativi paramentri (Figura 3.6)

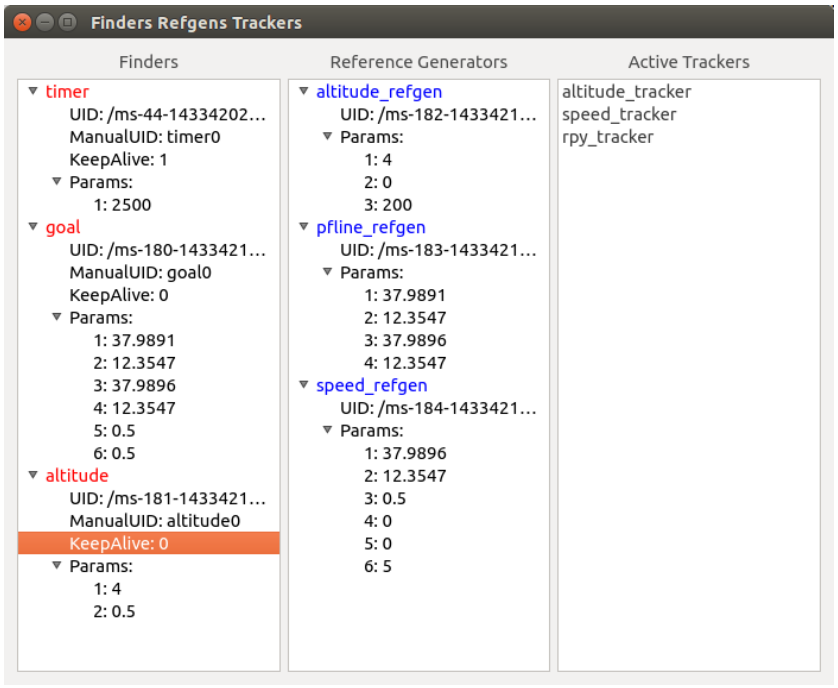


Figura 3.6: Tool per il monitoraggio di Finders, Reference Generators e Trackers

3.7.2 Pianificazione e controllo

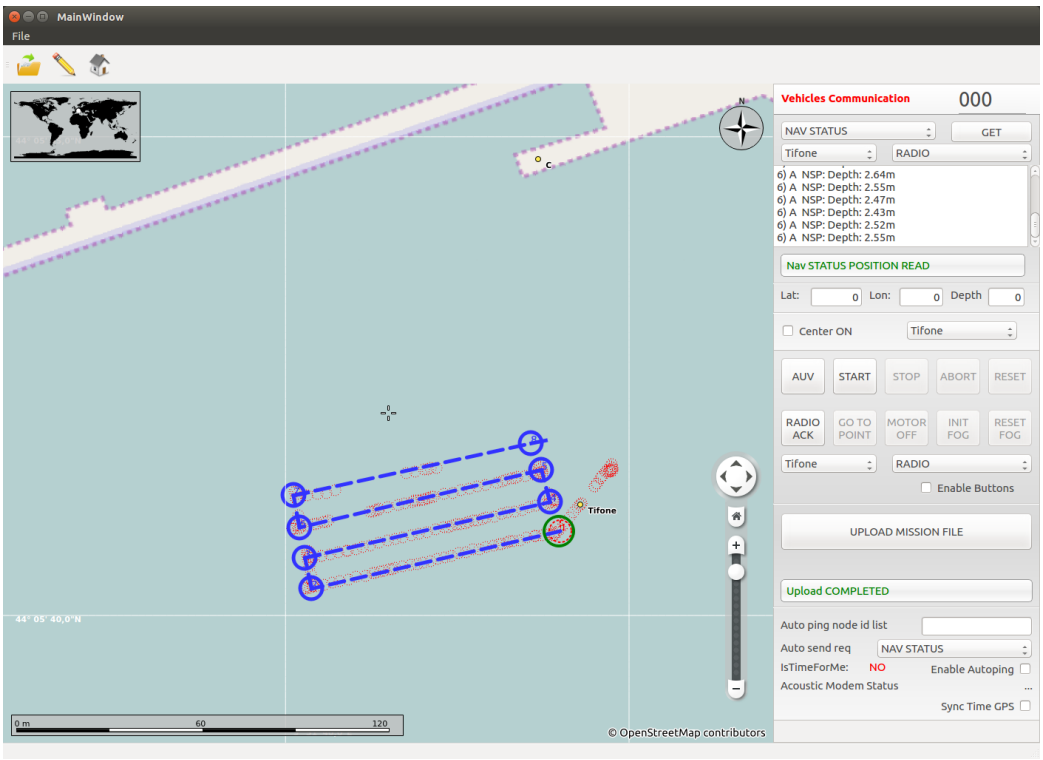


Figura 3.7: Interfaccia grafica di pianificazione e controllo

La seconda interfaccia proposta è stata realizzata per permettere all'utente di impartire comandi di base a veicoli, ricevere informazioni vitali, generare ed inviare i file di missione.

In figura 3.7 è riportato uno screenshot della schermata, la componente principale è una mappa geografica basata sul sistema cartografico Open Street Map [OpenStreetMap,] (scaricabile da internet e utilizzabile offline) in cui è possibile tracciare il percorso desiderato e visualizzare la cronologia di posizioni assunte da ogni veicolo operativo. Sulla destra è presente un pannello di controllo che consente di visualizzare tutti i messaggi ricevuti,

di impartire comandi e di inviare il file di missione generato sulla base del percorso disegnato.

Questo tool costituisce una vera e propria postazione di comando per gestire gli AUV in operatività, comunicando attraverso modem radio o acustico a seconda se il destinatario è in superficie o immerso. Per il suo funzionamento si appoggia ai nodi descritti in precedenza come Exchange Manager ed i drivers necessari ai dispositivi utilizzati.

3.8 Simulazioni

In fase di sviluppo è bene che ogni modifica apportata al sistema, prima di essere utilizzata sul campo, venga testata in un abiente di simulazione. Per praticità, è stato introdotto all'interno del sistema un nodo, chiamato "auv_sim", in grado di prendere il posto della parte hardware del veicolo. Avviando il sistema in modalità simulazione, questo nodo prende il posto dei nodi drivers e, simulandone il comportamento, genera i segnali di uscita dei sensori basandosi su un modello matematico. Per quanto riguarda la navigazione, è stato implementato il modello completo di un veicolo sottomarino come in 2.10:

$$M\dot{\vec{\nu}} + C\left(\vec{\nu}\right)\vec{\nu} + D\left(\vec{\nu}\right)\vec{\nu} + \vec{g}\left(\vec{\eta}\right) = \vec{\tau} \tag{3.14}$$

In questo modo si chiude il loop di controllo per avere un idea di come si comporterà realmente il veicolo.

Capitolo 4

Applicazioni

In questo capitolo si cerca di dimostrare la validità del software proposto mostrando i risultati ottenuti su tre piattaforme dalle caratteristiche molto diverse. Gli AUV costruiti all'intermo del laboratorio MDMLab nel corso degli ultimi 4 anni, sono parte integrante di questo lavoro di tesi in quanto lo sviluppo e il testing del software è andato di pari passo alle esigenze di questi veicoli.

Di seguito viene fornita una breve descrizione delle caratteristiche principali ed alcuni dei risulati più significativi ottenuti attraverso il loro utilizzo.

4.1 Tifone

Tifone è un AUV di classe media le cui caratteristiche sono comparable a quelle di altri AUV esistenti. Considerando le sue dimensioni (lunghezza di circa 3600 mm, diametro esterno di 350 mm, peso di 130-180 kg a seconda del payload caricato) e le performance richieste (profondità raggiungibile di circa 300 m, almeno 8 ore di autonomia e una velocità massima di 5-6 nodi), il veicolo piò essere comparato al più piccolo Remus 100 [Packard et al., 2013]

e al più grande Remus 600 [Stokey et al., 2005]. Attualmente sono state costruite due versioni del Tifone, caratterizzate da diversi sensori e payload. Questi sono stati chiamati rispettivamente TifOne e TifTu. Nelle figure 4.1 e 4.2 sono riportati i disegni CAD e la versione finale costruita del primo.

Figura 4.1: Disegno CAD del veicolo TIfone

Figura 4.2: Versione finale ralizzata

L'AUV Tifone ha cinque gradi di libertà (DOF) controllati attivamente (traslazioni longitudinali, laterali e verticali, rotazioni di beccheggio e imbardata), grazie a sei eliche (due in direzione laterale, due verticali e due longitudinali dedicate alla propulsione). Il corretto posizionamento e l'allineamento dei centri di gravità e galleggiamento assicurano la stabità del veicolo rispetto al moto di rollio.

Il sistma è stato disegnato dividendo i sottosistemi di bordo in due categorie:

- *sistemi vitali*: tutta la navigazione, comunicazione e i componenti legati alla sicurezze e funzionalità del veicolo sono controllati da un PC-104 industriale, chiamato PC Vitale
- *payload*: tutti i sensori e le funzioni relative ai vari payload sono gestiti da uno o più PC a alte prestazioni.

Il set di sensori di navigazione disponibili sui due veicoli di tipo Tifone sono:

- **Unità di misura inerziale (IMU) Xsens MTi -** dispositivo che include un giroscopio 3D, un accelerometro 3D e un magnetometro 3D che forniscono i dati di misura alla frequenza massima di 100 Hz (figura 4.3)

Sensor performance	Rate of turn	Acceleration	Magnetic field
Dimensions	3 axes	3 axes	3 axes
Full Scale (standard)	± 300 deg/s	± 50 m/s²	± 750 mGauss
Linearity	0.1% of FS	0.2% of FS	0.2% of FS
Bias stability[5]	20 deg/h	0.02 m/s²	0.1 mGauss
Scale Factor stability[5]	-	0.03%	0.5%
Noise	0.05 deg/s/√Hz	0.002 m/s²/√Hz	0.5 mGauss
Alignment error	0.1 deg	0.1 deg	0.1 deg
Bandwidth	40 Hz	30 Hz	10 Hz
Max update rate	512 Hz	512 Hz	512 Hz

Figura 4.3: Xsens IMU - Datasheet

- **Giroscopio a fibra ottica (FOG) KVH DSP 1750 -** Giroscopio digitale a singolo asse molto accurato basato sull'effetto Sagnac in un interferometro di Sagnac, è usato per misurare la velocità angolare intorno al suo asse verticale, per ottenere stime affidabili dell'angolo di imbardata, anche in presenza di disturbi magnetici (figura 4.4)

Number of Axes	Single-Axis (Digital)	
Variant	Standard Rate	High Rate
Input Rate *(max.)*	±490°/s	±1000°/s
Bias Instability *(constant temperature, short-term, in-run)*	0.1°/hr, 1σ (max), ≤0.05°/hr, 1σ (typical)	
Bias Temp. Sensitivity *(full temp.)*	≤1°/hr, 1σ (max), 0.7°/hr (typical)	
Bias Offset *(max. at 25° C)*	Unshielded: ±10°/hr Magnetically shielded: ±2°/hr	
Scale Factor *(nominal)*	1 ±0.2%	
Scale Factor Non-linearity *(at 25° C)*	≤50 ppm, 1σ	
Scale Factor Temp. Sensitivity	≤200 ppm, 1σ	
Angle Random Walk	≤0.013°/√hr ≤0.8°/hr/√Hz	
Bandwidth (-3 dB)	≥440 Hz (±4%)	

Figura 4.4: KVH DSP 1750 FOG - Datasheet

- **Doppler Velocity Log (DVL) Teledyne Explorer -** sensore che misura la velocità lineare, in terna body, del veicolo rispetto al fondale o alla colonna d'acqua soprastante. Inoltre, se riesce a raggiungere il fondale, è anche capace di misurarne la distanza come un altimetro (figura 4.5)

Maximum Altitude[1,3]	66m
Minimum Altitude	0.5m (0.25m optionally)
Velocity Range[2]	±17.0 m/s
Long Term Accuracy[4]	±0.5% ± 0.2cm/s
Long Term Accuracy[5]	±1.15% ± 0.2cm/s
Precision @ 1m/s[6]	±1.0cm/s
Precision @ 3m/s[6]	±1.9cm/s
Precision @ 5m/s[6]	±2.8 cm/s
Resolution	0.1cm/s (default), 0.001cm/s (selectable
Ping Rate	12Hz max

Figura 4.5: Teledyne Explorer DVL - Datasheet

- **Sensore di profondità STS DTM -** sensore di pressione digitale usato per ricavare la profondità del veicolo (figura 4.6)

	> 25 ... 600, (1), (2)
Overpressure	3 x FS (≤ 850 / ≤ 1500 bar)
Burst pressure	> 850 / ≤ 1500 bar
Accuracy, (3), (± % FS)	≤ 0.1
Thermal shift, (± % FS/°C)	
Zero point 0...70°C	≤ 0.015
Zero point -25...85°C	≤ 0.02
Span 0...70°C	≤ 0.015
Span -25...85°C	≤ 0.02
Long term stability, (4)	< 0.1% FS / < 0.2% FS

(1) Titanium available ≤ 400 bar (burst pressure > 550 bar)
(2) Overpressure and burst pressure 1500 bar (stainless steel) optional

Figura 4.6: STS DTM Depth sensor - Datasheet

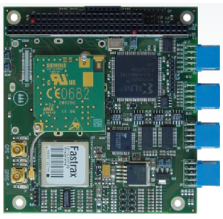
- **Ultra Short Baseline (USBL) Evologics S2CR** 18/34 **USBL -** sistema di localizzazione acustica compatibile con i modem acustici della classe di veicoli Tifone utilizzati per la comunicazione e la cooperazione. TifTu ospita a bordo il trasduttore USBL capace di localizzare gli altri veicoli, TifOne ospita invece un modem capace di essere localizzato (figura 4.7)

- **Modulo GPS per PC-104 COM-1289 -** ricevitore GPS integrato in un modulo di espansione per il PC Vitale

La tabella 4.1 riassume il set di sensori di navigazione che sono disponibili in ognuno dei due AUV di classe Tifone.

GENERAL	OPERATING DEPTH	Delrin	200 m
		Aluminium Alloy	1000 m
		Stainless Steel	2000 m
		Titanium	2000 m
	OPERATING RANGE		3500 m
	FREQUENCY BAND		18 - 34 kHz
	TRANSDUCER BEAM PATTERN		horizontally omnidirectional
USBL	SLANT RANGE ACCURACY [1]		0.01 m
	BEARING RESOLUTION		0.1 degrees
	NOMINAL SNR		10 dB

Figura 4.7: Evologics S2CR 18/34 USBL - Datasheet



Main GPS Features

- Fastrax iTrax03 12-channel low power GPS receiver
- L1 frequency, C/A code (SPS)
- Update rate: 1fix/s (user configurable up to 3 Hz)
- Accuracy:
  - Position: 1.2m (CEP95)
  - Velocity: 0.1m/s
  - Time: 20ns rms (static mode)
- Reacquisition: 100ms (typical)
- Tracking sensitivity: -156 dBm
- Support for +3.3 active & passive external antennas
- Protocols: NMEA-0183 and proprietary iTalk™ binary protocol

Figura 4.8: PC-104 GPS Module COM-1289 - Datasheet

4.2 MARTA

Il veicolo MARTA (MARine Tool for Archeology) è composto da diversi moduli, ognuno dedicato a un particolare compito (propulsione, sensori payload,

Sensore	TifOne	TifTu
Xsens IMU	✓	✓
KVH FOG	✓	×
Teledyne DVL	✓	×
STS DTM	✓	✓
Evologics USBL	Modem localizzabile	Trasduttore localizzatore
PC-104 GPS	✓	✓

Tabella 4.1: Set di sensori per TifOne e TifTu

alimentazione, ecc...). In questo modo MARTA AUV può essere modificato in accordo alla tipologia di missione da eseguire. Questo veicolo può essere messo a mare anche da una piccola imbarcazione, è modulare e ha una lunghezza totale di circa $3,5$ m (dipende dalla configurazione), un diametro esterno di 180 mm e un peso in aria di circa 80 kg. Analogamente al veicolo Tifone, MARTA ha cinque gradi di libertà completamente controllabili per mezzo di sei attuatori (motori elettrici più eliche): due posteriori, due laterali e due verticali. Il sistema di propulsione è ridondante: il veicolo non è equipaggiato soltanto con eliche, la traslazione verticale e l'angolo di beccheggio possono essere controllate anche per mezzo di due moduli di galleggiamento (posizionati a prua e a poppa); in questo modo il veicolo può essere controllato anche vicino al fondale, senza mettere a rischio le eliche ed evitando di muovere la sabbia che può creare problemi nell'acquisizione ottica e acustica. In figura 4.9 è mostrato il disegno 3D definitivo di MARTA AUV, mentre in figura 4.10 la sua realizzazione.

4.2.1 Caratteristiche principali

Riassumendo, le caratteristiche principali sono:

- Profondità raggiungibile: 150 m

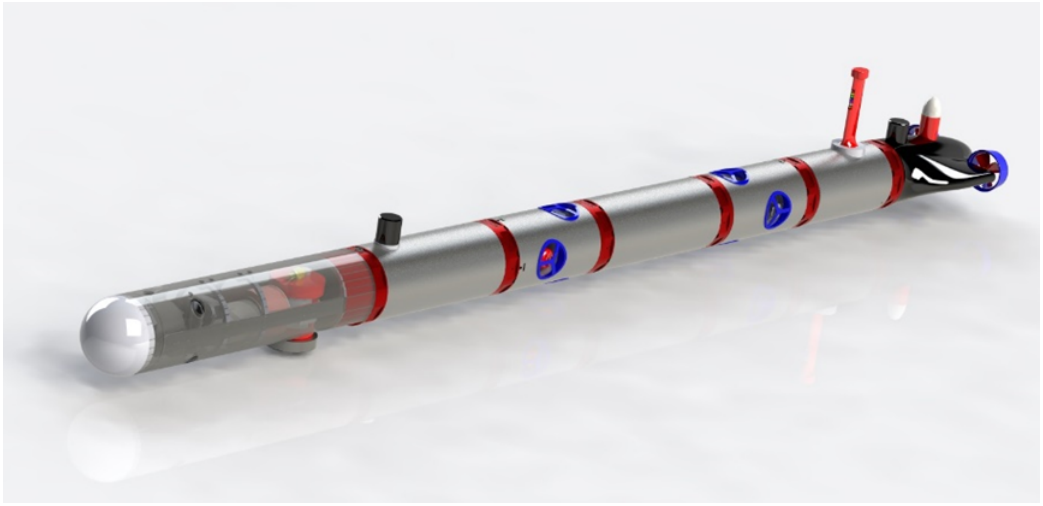


Figura 4.9: Disegno CAD di MARTA

Figura 4.10: Marta AUV

- Velocità massima: 4 nodi
- Autonomia: circa 4 ore
- Peso: 80 kg
- Modularità: vengono scelti diversi moduli a seconda della missione da compiere
- Capacità di Hovering: il veicolo, sia in superficie che sott'acqua, è capace di mantenere la sua posizione.

I moduli di MARTA sono costruiti in Al Anticorodal e ospitano i seguenti componenti:

- due moduli di galleggiamento
- due computer vitali ODROID-XU
- due diversi modem acustici (Evologics e Applicon)
- un sensore di pressione SensorTechincs
- una piattaroma inerziale (IMU) Xsens MTi-G-700 GPS
- un giroscopio a fibra ottica (FOG) a singolo asse KVH DSP-1760
- un DVL NavQuest 600 Micro
- un radio modem RF Solutions
- sei batterie LiPo da sei celle a 22.2 V MaxAmps
- payload ottici e acustici

La modularità di MARTA è assicurata anche da un punto di vista elettrico. Le interfacce elettriche di ogni modulo sono standard in entrambi i lati per permettere ad ogni modulo di poter occupare una qualsiasi posizione all'interno del veicolo, e per poter essere aggiunte o rimosse a seconda delle necessità della missione. In particolare, in ogni interfaccia, sono presenti i due cavi di alimentazione dotati di un connettore standard che evita potenziali inversioni di polarità. Lo scambio di dati tra ogni modulo è assicurato da un cavo ethernet e uno switch presenti in ogni interfaccia. Infine, è presente un canale di connessione CAN-BUS in ogni modulo, assicurando la possibilità di controllare gli attuatori dal PC di poppa o di prua, ovunque essi si trovino.

Per quanto riguarda la movimentazione del veicolo, come visibile in figura 4.11, la propulsione di default di MARTA è composta da sei motori (due principali a poppa, due laterali e due verticali) usati per controllare tutti i gradi di libertà del veicolo, ad eccezione del rollio.

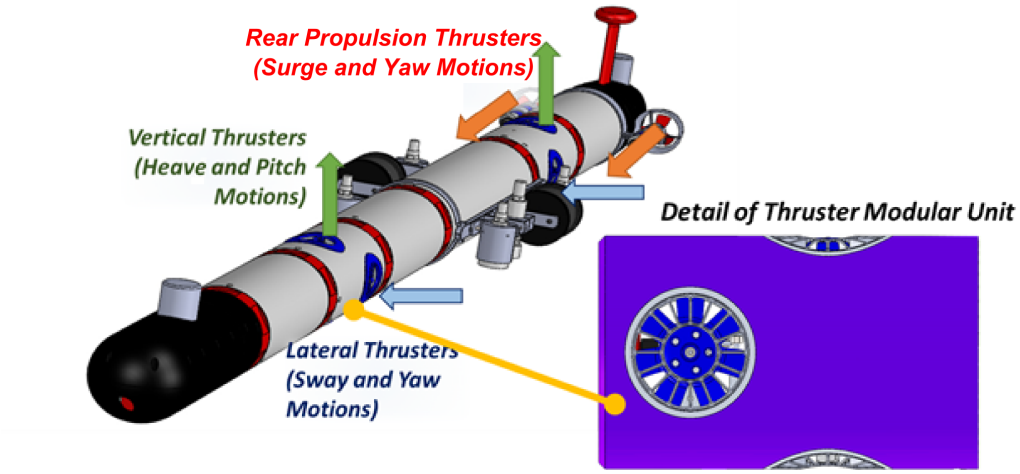


Figura 4.11: MARTA, divisione in moduli e sistema di propulsione

Per quanto riguarda i sensori payload, MARTA è capace di ospitare dispositivi specifici per le missioni archeologiche:

- Payload acustico: un forward-looking sonar 2D Teledyne BlueView M900 è montato sulla prua

- Payload ottico: un modulo di MARTA è dedicato a ospitare il dispositivi ottici (una coppia di telecamere Basler Ace per la stereo vistion, una lama laser e quattro illuminatori a LED) inoltre all'interno del modulo è presente un elaboratore ad alte prestazioni SBC Commel LS-378 per l'acquisizione e l'elaborazione dei dati.

4.2.2 Risultati

Il veicolo MARTA è diventato operativo nel luglio 2015 con la conclusione del progetto ARROWS. Tra i risultati più interessanti ottenuti con questo veicolo si sottolinea la validazione di un metodo di stima di orientazione basato su piattaforma inerziale e FOG [Allotta et al., 2015b].

I sensori coinvolti in questa attività sono: IMU Xsens MTi-G-700 e FOG KVH DSP-1760 montato col suo asse sensibile lungo l'asse verticale del veicolo, coincidente con quello verticale dell'IMU. Per prima cosa viene effettuate un giro completo intorno all'asse verticale; i dati acquisiti dal sensore in questa fase sono usati per eseguire l'algoritmo di calibrazione. Conseguentemente, MARTA viene ruotata in modo da rimanere approssimativamente sul piano orizzontale; le prestazioni dell'algoritmo di stima proposto, che utilizza i dati calibrati secondo la fase precedente, vengono valutate in assenza di fonti di disturbo esterno considerabili. Si confrontano i risultati ottenuti con la stima interna della piattaforma inerziale Xsens.

La figura 4.12 riporta le tre componenti del vettore $\vec{\eta}_{2diff} = \vec{\eta}_{2filter} - \vec{\eta}_{2Xsens}$ ottenuti durante il primo test. Il grafico mostra che la differenza tra rollio e beccheggio è piccola durante tutto il tempo necessario a com-

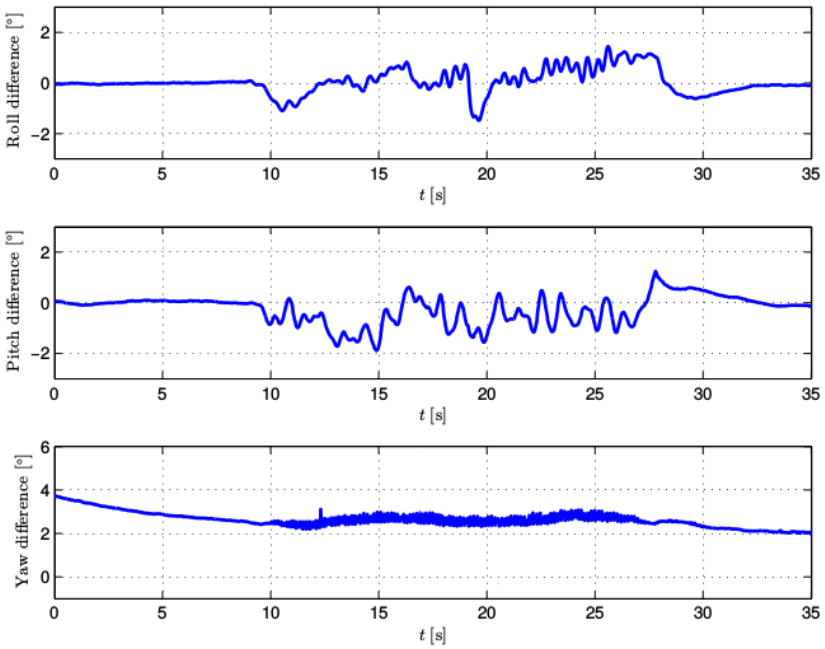


Figura 4.12: $\vec{\eta}_{2diff}$ ottenuta durante il primo test

pletare il cerchio, portando alla deduzione che le prestazioni dei due filtri riguardo queste due variabili sono comparabili in assenza di disturbi magnetici. L'imbardata differisce di circa 3° per tutta la durata; questo valore corrisponde all'angolo di declinazione magnetica locale (in questo caso Toscana, Italia), dovuto alla discrepanza tra il Nord geografico e quello magnetico. Questo angolo, localmente costante e calcolabile tramite accurati modelli[NGDC-NOOA, 2014], è internamente compensato dall'algoritmo interno dell'Xsens. Quindi, alla luce di questa considerazione, i due algoritmi sembrano essere comparabili anche per questo grado di libertà, in assenza di disturbi magnetici esterni.

Il valore dell'angolo di imbardata per gli algoritmi considerati è visibile in figura 4.13, evidenziando quanto questi si sovrappongano durante tutto il test.

Il secondo test mostra come l'algoritmo interno dell'Xsens può portare a

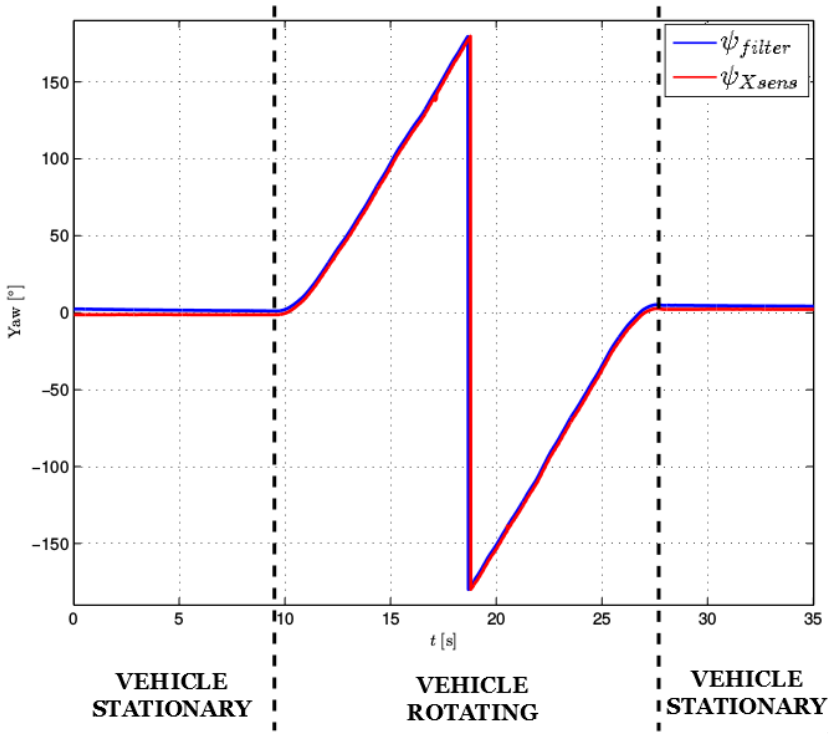


Figura 4.13: Valori di imbardata ottenuti nel primo test

funzionamenti non accettabili in presenza di disturbi magnetici (siano essi magneti o oggetti ferromagnetici) vicini al sensore, portando l'uscita a divergere dalla orientazione corretta. L'algoritmo proposto, invece, garantisce una corretta stima anche in presenza di questi disturbi. Questo test è composto dai seguenti passi: senza effettuare una calibrazione del magnetometro il veicolo è all'inizio mantenuto stazionario fino alla convergenza del filtro, successivamente viene introdotta artificialmente una fonte di disturbo magnetico (dopo 17 secondi) avvicinando una pezzo di metallo vicino al veicolo; dopo un certo tempo viene effettuata una rotazione di circa 40°, e l'oggetto di metallo viene finalmente rimosso.

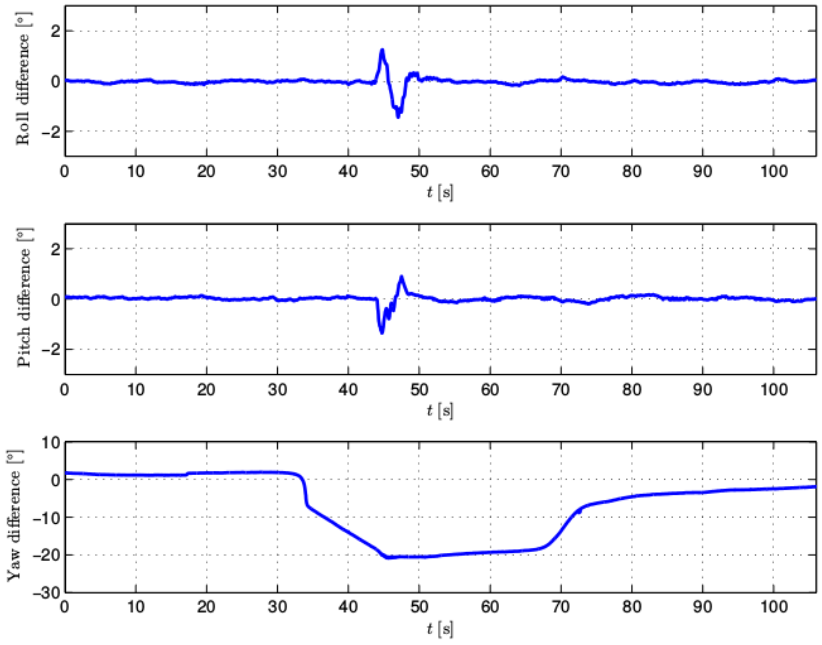


Figura 4.14: $\vec{\eta}_{2diff}$ ottenuta durante il secondo test

Come per il test precedente, in figura 4.14 si riporta la differenza tra le uscite dei due algoritmi; mentre φ_{filter} e ϑ_{filter} rimangono vicini al relativi valori stimati per mezzo del software Xsens, lo stesso non succede per la stima dell'imbardata, che presenta una differenza rilevante, fino a 20°.

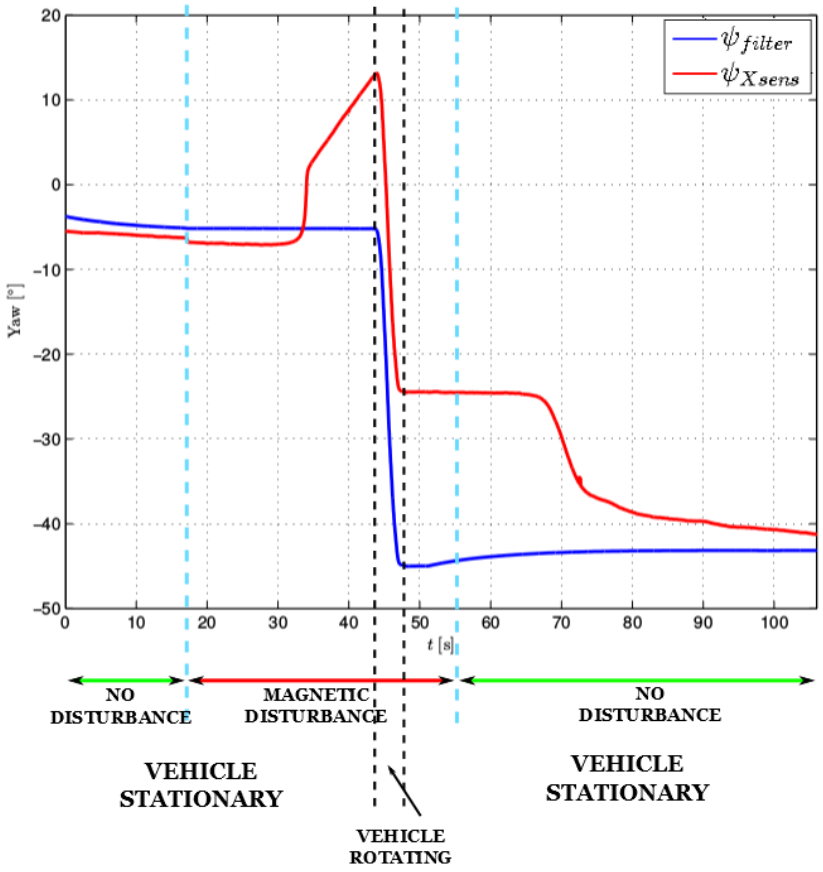


Figura 4.15: Valori di imbardata ottenuti nel secondo test

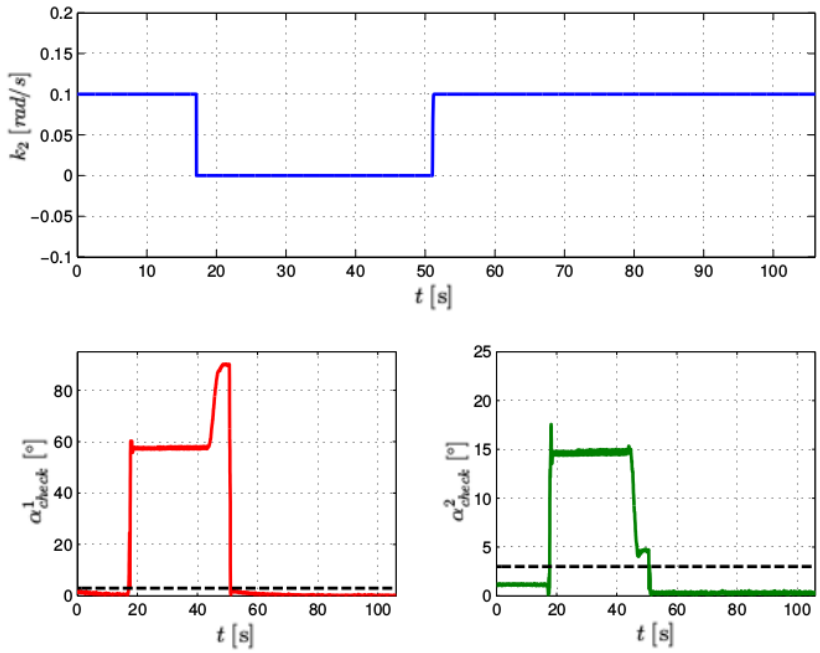


Figura 4.16: k_2 e angoli di controllo durante il secondo test

Le figure 4.15 e 4.16 mostrano che, quando il pezzo di metallo inizia a disturbare il sensore, gli angoli di controllo assumono un valore molto alto e il guadagno viene prontamente ridotto a zero. L'algoritmo dell'Xsens, invece non riconosce il disturbo, causando una deviazione della stima di imbardata anche se il veicolo viene mantenuto stazionario. Quando la fonte di disturbo viene rimossa dopo la rotazione, gli angoli di controllo crollano al disotto della soglia e il guadagno riacquista il valore iniziale. Osservando la figura 4.15, è possibile dedurre l'andamento del filtro: quando k_2 è a zero, la stima di imbardata viene calcolata solamente integrando le letture del FOG. Quando il veicolo è fermo l'angolo di imbardata mostra un drift dovuto all'integrazione del rumore del giroscopio; comunque, data la precisione del dispositivo, la deviazione nel breve termine è trascurabile. Questo è supportato dal fatto che, dopo che il guadagno viene incrementato di nuovo, la correzione applicata dal magnetometro non modifica sostanzialmente la stima corrente.

Nel caso dell'Xsens, quando il disturbo viene aggiunto, si registra un drift significativo. Altri test simili sono stati effettuati per valutare l'affidabilità del metodo proposto; confermano tutti i risultati ottenuti dall'analisi appena descritta.

Per sintetizzare, l'AUV MARTA è stato utilizzato per validare un algoritmo di stima di orientazione dimostrando attraverso un attività sperimentale che l'integrazione dei due dispositivi supera i limiti dei sistemi basati soltanto su sensori MEMS. Questo algoritmo può essere utilizzato su tutti i veicoli dotati dei due sensori citati.

4.3 FeelHippo

FeelHippo è un AUV a basso costo sviluppato dal team di studenti dell'Università di Firenze che ha partecipato alle competizioni internazionali SAUC-E 2013 e Eurathlon 2015. Il suo design è orientato a fronteggiare i compiti proposti dalle regole delle competizioni ma, allo stesso tempo, in modo che sia semplice integrare funzionalità aggiuntive in futuro.

4.3.1 Caratteristiche meccaniche

Il corpo centrale di FeelHippo (figura 4.17) è un tubo di plexiglass che contiene la strumentazione, eccetto per le batterie che sono alloggiate nei due tubi sottostanti. Le altre parti, come i propulsori, il sensore di pressione e le luci LED, risiedono all'esterno della parte stagna e sono collegate alla struttura da specifici supporti in plastica (ABS) realizzati per mezzo di una stampante 3D per prototipazione rapida dell'MDMLab.

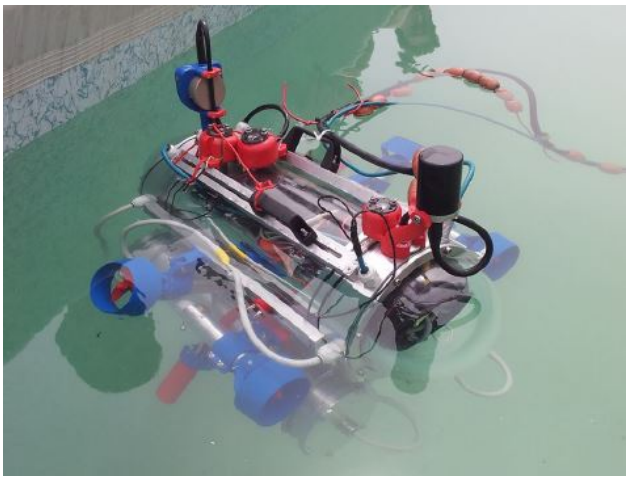


Figura 4.17: AUV FeelHippo nella piscina dell'MDMLab

Il moto di FeelHippo è sotto-attuato ma sufficiente per i classici compiti di monitoraggio ed è dato da 4 eliche [Carlton, 2007a] che permettono di controllare quattro gradi di libertà del veicolo:

- due di queste sono montate a V, per controllare la profondità e il movimento laterale
- le altre due sono orizzontali, montate ai due lati del veicolo in modo da controllare il moto longitudinale e l'angolo di imbardata.

Per quanto riguarda gli altri dispositivi, nella parte anteriore ci sono un sensore di tipo echo-sounder e tre LED capaci di illuminare l'ambiente subacqueo; questa soluzione consente a una telecamera frontale di ottenere buone immagini. Nella parte posteriore del veicolo ci sono, invece, il sensore di profondità, una luce stroboscopica (per segnalazione, scelta obbligata da regolamento delle competizioni) e un altra luce LED puntata verso il fondale, in modo da illuminare la zona inquadrata da una seconda telecamera puntata verso il basso. Accanto alla luce stroboscopica è presente il trasduttore di un modem acustico per la comunicazione.

Dentro al tubo di plexiglass la strumentazione è alloggiata su due ripiani dello stesso materiale. Sul piano superiore sono posizionati i seguenti dispositivi: la piattaforma inerziale Xsens IMU, un display LCD, una scheda elettronica Micro Maestro per il controllo dei motori e un unità di calcolo. Nella parte sottostante ci sono, invece: l'elettronica del modem acustico, un convertitore DC-DC (24V-12V), gli azionamenti dei motori e la telecamera puntata verso il basso. Le quattro batterie LiPo sono alloggiate all'interno dei tubi per abbassare il più possibile il baricentro e per facilitarne l'estrazione per la ricarica. La struttura dell'AUV garantisce:

- facilità di assemblaggio
- semplice accesso ai componenti interni
- soluzioni sicure grazie all'alimentazione a 24 V
- buona stabilità a rollio
- peso ridotto (circa 30 kg)
- basso costo

4.3.2 Architettura elettronica

Il veicolo è equipaggiato con quattro batterie LiPo a sei celle la cui tensione nominale è di 22.2 V collegate in parallelo. Ognuna può fornire una corrente di 8 Ah per un totale di circa 710 Wh. Tutti i moduli a bordo sono alimentati dall'alimentazione principale, direttamente o tramite appropriati convertitori DC-DC. I livelli di tensione sono due, il principale a 24 V e quello derivato a 12 V. Ogni computer a bordo è dedicato ad un particolare task, e tra loro sono interfacciati tramite connessione Ethernet grazie all'utilizzo del middleware

ROS. L'AUV ha quattro motori elettrici di tipo brushless (Maxon Motors). Il modello scelto è da 100 W e non prevede sensori di velocità. Nonostante le sue piccole dimensioni e il peso ridotto, questo motore genere una coppia piuttosto alta grazie a un riduttore epicicloidale (riduzione 3.8 : 1).

4.4 Cooperazione

Al momento non è ancora stata integrata nel software una logica di cooperazione vera e propria tra veicoli. Al livello attuale il sistema mette a disposizione un metodo di comunicazione efficiente basato su slot temporali ma, al momento, viene sfruttato solamente per scambiare informazioni con la stazione di base a scopo di monitoraggio.

In questo ambito sono comunque stati eseguiti importanti test riguardo la localizzazione subaquea per mezzo di strumentazione acustica, in collaborazione con i centri CSSN (Centro di Supporto e Sperimentazione Navale) e CMRE (Centre for Maritime Research & Experimentation) di La Spezia.

4.4.1 Localizzazione acustica

Questa ricerca è stata basata sull'uso dell'AUV TifOne, una boa dotata di un modem acustico USBL (Evologics S2CR 18/34 USBL) e il sistema LOON (Littoral Ocean Observatory Network, un network di quattro modem acustici, di cui uno USBL, montati sul fondale marino in posizioni note) del CMRE.

Questa attività si vogliono confrontare le prestazioni dell'USBL fisso appartenente al sistema LOON e quello mobile, montato su una boa. Le tipologie di test effettuati sono due:

- Localizzazione dei modem fissi e del veicolo TifOne (in missione subacquea) sfruttando la posizione nota del modem USBL utilizzato

- Auto-localizzazione del modem USBL utilizzato mediante la conoscenza della posizione dei modem fissi e, in contemporanea, localizzazione del veicolo TifOne in missione subacquea

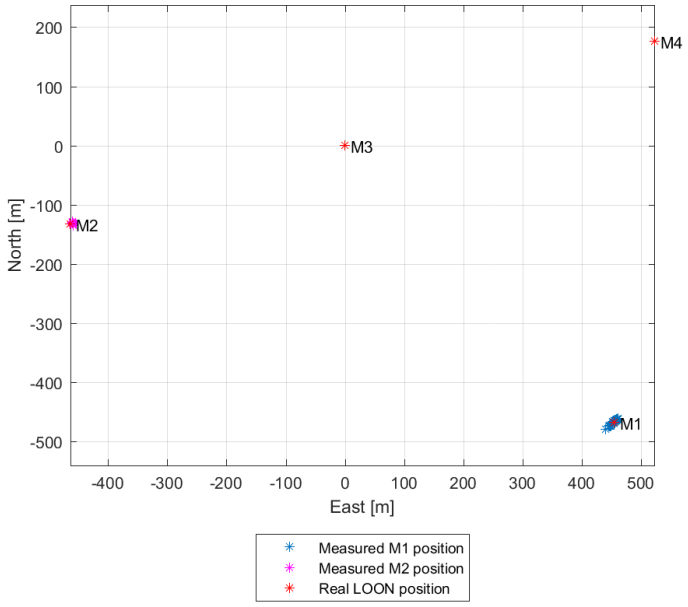


Figura 4.18: Distribuzione della posizione di M1 e M2 utilizzando M3-USBL

USBL fisso La figura 4.18 dimostra chiaramente le ottime prestazioni dell'USBL fisso nella localizzazione dei modem M1 e M2. L'errore di posizione è riportato in figura 4.19, ottenuta comparando le misure e la stima ottenuta da un filtro di Kalman Esteso con modello di posizione costante. I benefici nelle prestazioni utilizzando il filtro EKF sono molto piccoli se comparati alla distanza tra M3 e gli altri modem.

Per valutare le capacità di auto-localizzazione di questo USBL, una campagna di acquisizione dati è stata dedicata al confronto tra la localizzazione

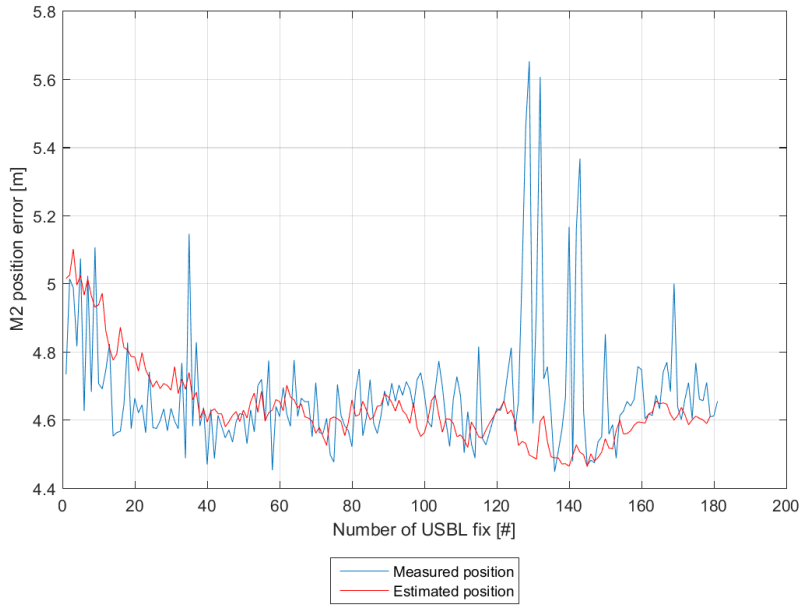


Figura 4.19: Errore di posizione misurata e stimata di M2

acustica (rispetto agli altri modem del LOON) di M3 e la sua posizione conosciuta.

In figura 4.20 gli asterischi neri rappresentano la distribuzione del processo di auto-localizzazione acustica di M3. Gli asterischi blu e magenta sono le posizioni di M1 e M2 rispettivamente, calcolate basandosi sulla posizione stimata di M3. L'errore risultante è quindi una somma di contributi dovuta alle due fasi. In figura 4.21 invece è riportato l'errore del processo di auto-localizzazione di M3. L'errore risulta molto contenuto se confrontato con le distanze relative tra i modem.

Purtroppo, il modem M3 non è mai riuscito ad ottenere la posizione del veicolo TifOne, probabilmente a causa del posizionamento del modem sul veicolo nella parte superiore, il secondo test non risulta quindi completo.

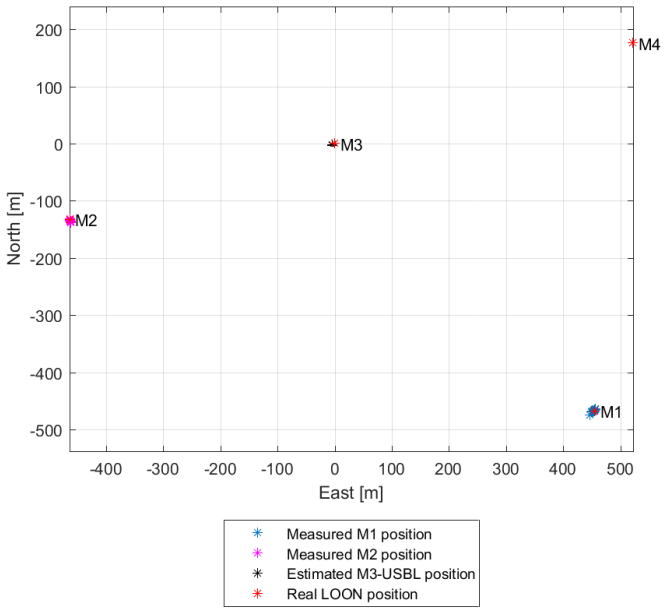


Figura 4.20: Distribuzione della posizione di M1 e M2 utilizzando M3-USBL (con auto-localizzazione)

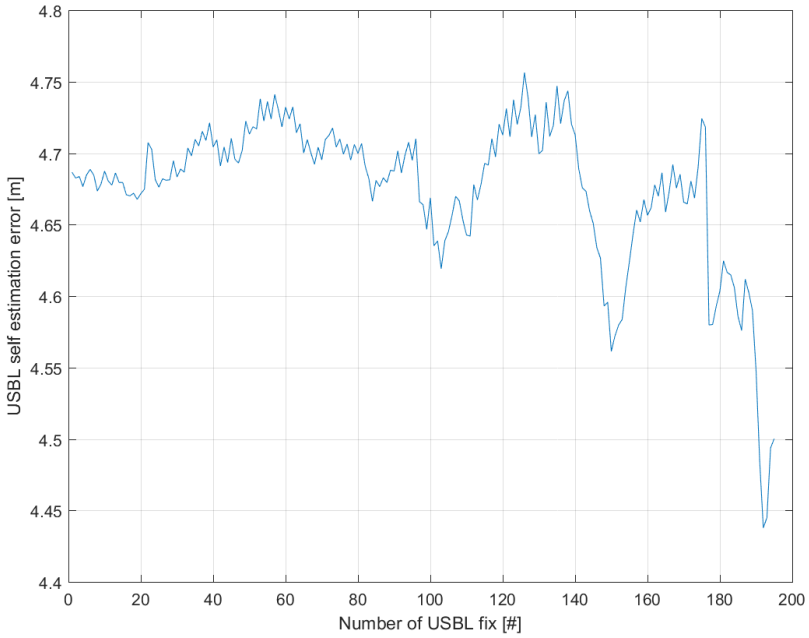


Figura 4.21: Errore di posizione nell'auto-localizzazione di M3

USBL mobile Si riportano adesso i dati relativi agli stessi test effettuati però con l'USBL mobile montato sulla boa. In questo caso, sia la posizione che l'orientazione del modem non sono fissi e vengono misurati per mezzo di una piattaforma inerziale Xsens MTi-G-700 GPS. L'errore di posizione quindi sarà affetto, oltre all'errore di misura dell'USBL, anche ai contributi dovuti all'utilizzo di questo sensore.

Il primo test consiste nella localizzazione dei modem fissi e di Tifone sfruttando il GPS del sensore a bordo della boa. Le figure 4.22 e 4.23 rappresentano la distribuzione di misure della posizione di M2, M3 e TifOne; sono presenti anche degli outliers dovuti probabilmente a delle riflessioni acustiche. Per quanto riguarda TifOne, viene riportato anche una stima della posizione ottenuta tramite un EKF basato su un semplice modello cinematico. Nel grafico in figura 4.24 è rappresentato l'errore di misura e di stima per

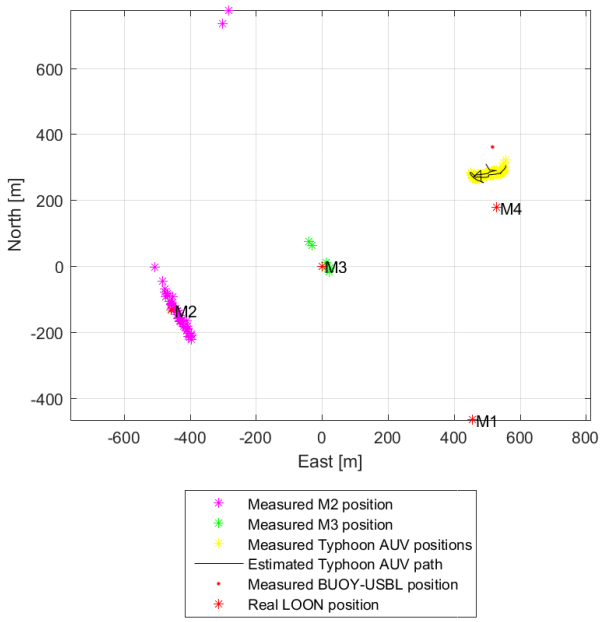


Figura 4.22: Distribuzione della posizione di M2, M3 e Tifone utilizzando la Boa

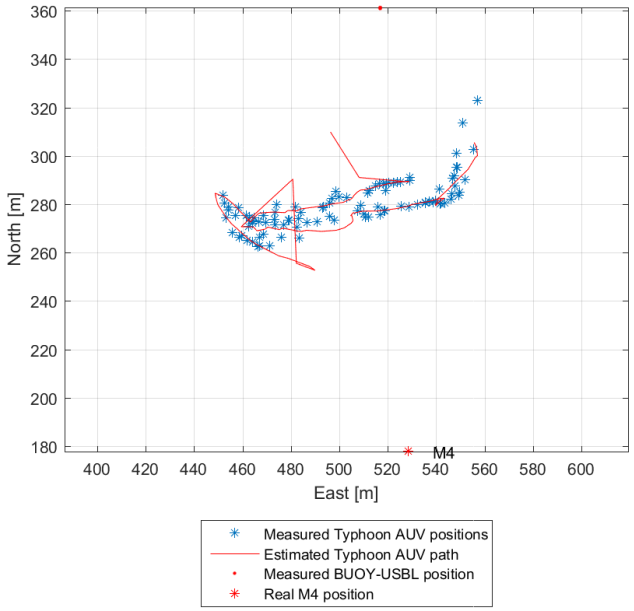


Figura 4.23: Dettaglio della posizione di TifOne utilizzando la boa

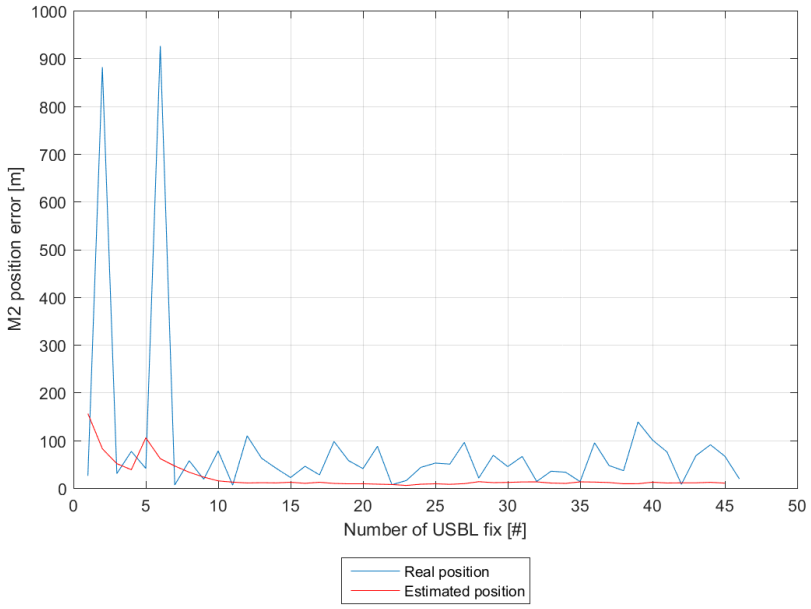


Figura 4.24: Errore di posizione misurata e stimata di M2

il modem M2. Da questi grafici è facile notare quanto peggiorino le misure di posizione basandosi su un USBL mobile con piattaforma inerziale esterna rispetto all'USBL fisso, il filtraggio dei dati attraverso filtri di Kalman consente di ridurre notevolmente l'errore.

Infine si mostrano i grafici relativi all'auto-localizzazione della boa rispetto ai modem fissi. Come nel caso precedente, in figura 4.25 vengono mostrate le posizioni dei modem fissi e di TifOne basandosi sulla posizione stimata della boa. In figura 4.26 viene riportato l'errore del processo di auto-localizzazione dell'USBL montato sulla boa confrontato con il GPS; in questo caso l'errore risulta particolarmente limitato se confrontato con le misure ottenute, questo fatto è dovuto al modo in cui vengono processati i dati all'interno del filtro di Kalman: considerando pesi diversi per distanza e heading si ha che, utilizzando più modem per localizzarsi, il filtro operi in

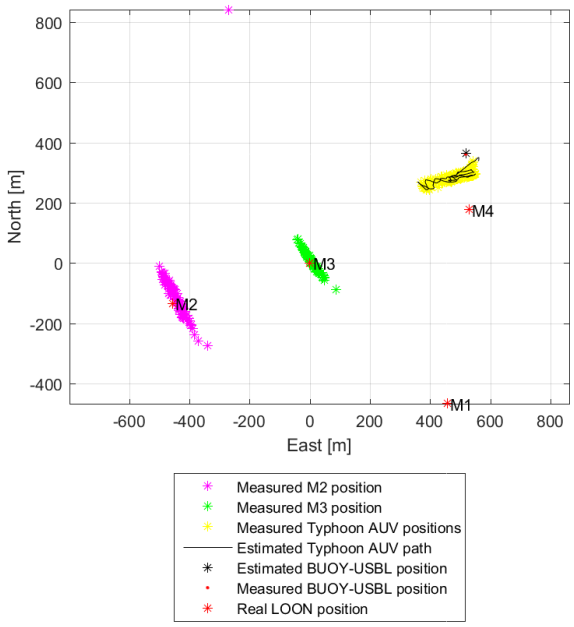


Figura 4.25: Distribuzione della posizione di M2, M3 e TifOne utilizzando la boa (con auto-localizzazione)

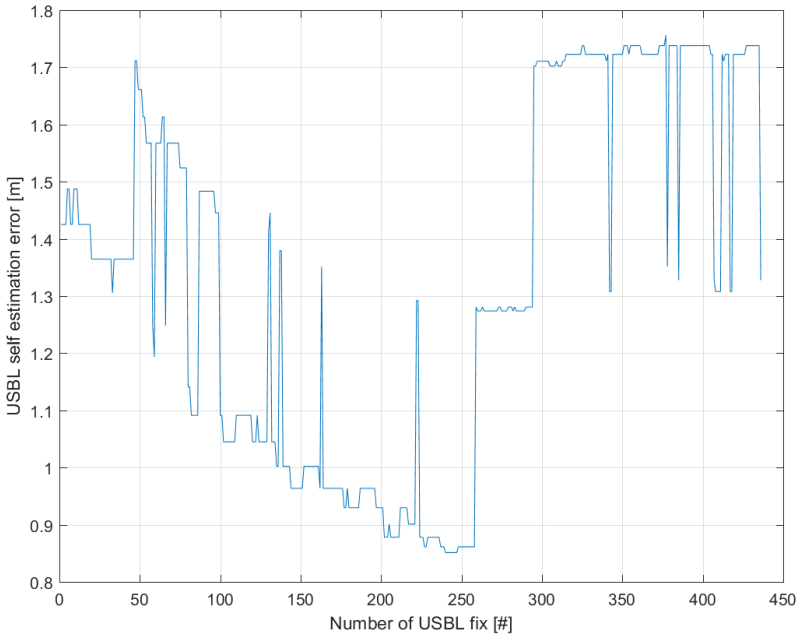


Figura 4.26: Errore di posizione nell'auto-localizzazione della boa

modo simile a una triangolazione, per cui le misure ridondanti consentono di migliorare molto la stima.

Capitolo 5

Conclusioni

In questa tesi sono stati descritti la progettazione, realizzazione ed i test di un sistema software capace di gestire le funzionalità degli UUV (Unmanned Underwater Vehicles). Il lavoro in questione si inserisce in un contesto di ricerca portato avanti dal Laboratorio di Modellazione Dinamica e Meccatronica (MDM Lab) del Dipartimento di Ingegneria industriale dell'Università degli Studi di Firenze negli ultimi anni. In particolare è stato parte integrante di due progetti: il progetto regionale Toscano THESAURUS e il progetto Europeo FP7 ARROWS, entrambi animati dall'obiettivo di portare la tecnologia degli AUV dagli attuali campi di utilizzo (es. Oil&Gas e militare) sino al vasto campo dell'archeologia subacquea, in modo da fornire nuove tecnologie per un più efficiente sviluppo di questa importante ambito culturale.

Gli obiettivi posti per questo lavoro però, non si sono limitati a soddisfare le esigenze di questi progetti, ma cercano di gettare le basi per una piattaforma software di più largo impiego. Grazie alle caratteristiche di modularità e portabilità, il sistema sviluppato è stato creato in modo da poter essere riadattato a tutti i tipi di UUV, ed i test effettuati su tre diversi veicoli (TifOne, MARTA e FeelHippo) lo dimostrano. Gli AUV in questione sono

attualmente a disposizione di MDM Lab e il loro funzionamento è basato sul software descritto.

Le possibili applicazioni di questo lavoro sono quindi molteplici. Innanzitutto in ambito di ricerca, date le caratteristiche citate, rappresenta una base per l'implementazione di nuovi algoritmi di navigazione e controllo, oppure di utilizzo della stumentazione di bordo (sia online che in post-process). Inoltre il sistema è applicabile all'utilizzo in molti ambiti, come l'archeologia, campo per cui è stato progettato, e rappresenta una possibile alternativa agli attuali prodotti commerciali, che hanno la caratteristica di essere chiusi e limitati al veicolo per cui sono stati progettati.

Tra i più interessanti sviluppi futuri si indica l'implementazione di una logica di interazione tra veicoli e di operatività in sciami, caratteristica che non è stata affrontata all'interno di questa tesi e oggetto di una intensa attività di ricerca in ambito internazionale.

Appendice A

Listati

A.1 Reference Generator

```
1  class ReferenceGeneratorNode : public Node {
2  protected:
3    TargetParams localParameters;
4  private:
5    boost::mutex mtx_;
6    ros::Subscriber sEnabledReferenceGenerators;
7    ros::Time activationTime;
8
9  public:
10   boost::function<void()> parameterUpdateCB;
11   boost::function<void()> disabledCB;
12   bool isActive;
13   /**
14     * \brief Public constructor
15     */
16   ReferenceGeneratorNode(int argc, char ** argv, int frequency,
17                  std::string name, std::string name_space);
```

```
18
19    ~ReferenceGeneratorNode();
20
21    /**
22      * \brief This method is a callback for
23      * enabled_reference_generators topic.
24      * This topic provide a list of reference
25      * generators enabled at this time.
26      * @param msg
27      */
28    void checkEnabledReferenceGenerators(const Targets::ConstPtr& msg);
29
30    /**
31      * \brief This method provides the main loop,
32      * calling periodically the given function
33      * @param callback reference of the function which has to be called
34      */
35    void nodeLoop(boost::function<void()> callback);
36  };
```

A.2 Tracker

```
1   class TrackerNode: public Node{
2   private:
3
4     ros::Subscriber sEnabledTrackers;
5     ros::Publisher errorPublisher;
6     ros::Time activationTime;
7     ros::Time lastSensorTime;
8     ros::Duration lastDt;
9     bool dt_used;
```

```
10
11  protected:
12    boost::mutex mtx_;
13    bool sensor_ready, target_ready;
14
15  public:
16    bool isActive;
17
18    /**
19      * \brief Public constructor
20      */
21    TrackerNode(int argc, char ** argv, int frequency,
22              std::string name, std::string name_space);
23
24    ~TrackerNode();
25
26    void publishErrors(ErrorsTimed errors);
27
28    /**
29      * \brief This function provide a callback for enabledTrakers topic.
30      * This topic provide a list of trackers enabled at this time.
31      * @param msg
32      */
33    void checkEnabledTrackers(const Trackers::ConstPtr& msg);
34
35    /**
36    * \brief Main loop
37    * This method provides the main loop,
38    * calling periodically the given function
39    * @param callback reference of the function which has to be called
40    */
41    void nodeLoop(boost::function<void()> callback);
42
```

```
43    /**
44      * \brief Obtain the elapsed time between the last two readings
45      * @return delta time
46      */
47    double getDt();
48
49    /**
50      * \brief Update delta time calculation
51      * This should be done after each reading
52      */
53    void updateDt();
54  };
```

A.3 Finder

```
1   /**
2    * This class contains all the informations needed by an active finder
3    * instance, including the running thread pointer
4    */
5   class finder_info{
6   public:
7     FinderInfo info;
8     boost::thread * thptr;
9     bool run;
10
11    /**
12      * Public constructor
13      * @param th is the pointer to the thread which loops
14      *           until the finder is enabled
15      * @param param is structure which incudes all the parameters
16      *              needed by this finder instance
```

```
17      */
18    finder_info(boost::thread *th, FinderInfo &param){
19        info = param;
20        thptr = th;
21        run = true;
22    }
23  };
24
25  typedef std::map<std::string, finder_info> map_thread_t;
26
27  class FinderNode : public Node{
28  private:
29    map_thread_t active_thread;
30  protected:
31    boost::mutex mtx_;
32    bool sensor_ready;
33
34  public:
35    bool is_active;
36    bool new_data_avaiable;
37    ros::Subscriber topicSubscriber;
38    ros::Subscriber activeFindersSub;
39    ros::Publisher finderAnsPub;
40
41    /**
42      * Public constructor
43      * setup ROS subscriptions
44      */
45    FinderNode(int argc, char ** argv, int frequency, std::string name,
46            std::string name_space="finders", std::string service_name="");
47
48    ~FinderNode();
49
```

```
50    /**
51      * ROS callback, it creates a thread for each finder enabled
52      * and put it in the "active_thread" map
53      */
54    void checkEnabledFinders(const Finders::ConstPtr& msg);
55
56    /**
57      * Method to be implemented in the child class
58      * there should be an infinite loop which publish the answer
59      */
60    virtual void create_thread(FinderInfo info)=0;
61  };
```

A.4 Filter

```
1   class filter {
2   public:
3     // Filter initialization
4     virtual void initialize() = 0;
5
6     /** Update the filter status
7       *
8       * @return true if imu and depth status are ok
9       */
10    virtual bool update(int n = 0, ...) = 0;
11
12    /** Prediction of the filter status
13      *
14      * @return true if succeded
15      */
16    virtual bool prediction(int n = 0, ...) = 0;
```

```
17
18      /** Fill the NavStatus structure with status data
19        *
20        * @return the message ready to be sent
21        */
22      virtual NavStatus fill_status() = 0;
23
24      // Re-initialize the filter
25      virtual void reset() = 0;
26
27      // Filter state [ eta ; eta_dot]
28      Matrix<double, Dynamic, 1> state;
29
30      // State covariance matrix
31      Matrix<double, Dynamic, Dynamic> covariance;
32
33      // Error covariance matrix
34      Matrix<double, Dynamic, Dynamic> matrixQ;
35
36      // DVL measures and covariance matrix
37      Matrix<double, 3, 1> dvl;
38      Matrix<double, 3, 3> matrixR_dvl;
39
40      // IMU measures and covariance matrix
41      ImuData imu;
42      ImuCov matrixR_imu;
43
44      // depth measures and variance
45      double depth;
46      double cov_depth;
47
48      // Thrusters speed
49      Matrix<double, 6, 1> rpm;
```

```
50
51    // GPS measures and covariance matrix
52    Matrix<double, 2, 1> gps;
53    Matrix<double, 2, 2> matrixR_gps_ned;
54
55    // USBL measures and covariance matrix
56    Matrix<double, 3, 1> usbl;
57    Matrix<double, 3, 3> matrixR_usbl;
58
59    // Origin of the reference frame
60    Matrix<double, 3, 1> origin;
61
62    ros::Time dvl_time; // Store the time of last valid measure for DVL
63    ros::Time imu_time; // Store the time of last valid measure for IMU
64    ros::Time depth_time; // Store the time of last valid measure for Depth sensor
65    ros::Time gps_time; // Store the time of last valid measure for GPS
66    ros::Time filter_time; // Time of the last iteration of the filter
67
68    bool flag_dvl; // DVL ON/OFF flag
69    bool flag_imu; // Imu ON/OFF flag
70    bool flag_depth; // Depth ON/OFF flag
71    bool flag_gps; // GPS ON/OFF flag
72    bool flag_usbl; // USBL ON/OFF flag
73
74    // initialization flags
75    bool imu_initialized;
76    bool gps_initialized;
77    bool depth_initialized;
78
79    bool initialized; // Status of the filter, if initialized the position is avaible
80 };
```

A.5 Watcher

```
/** structure containing every parameter needed
 * by a Watcher
 */
struct timeouts {
  string device;
  bool enable;
  double alert_timeout;
  double emergency_timeout;
  countermeasure_code alert_countermeasure;
  countermeasure_code emergency_countermeasure;
  string topic_name;
  string enable_topic_name;
  double upper_limit;
  double lower_limit;
  double depth_threshold;
};

template <class T>
class Watcher {
public:
  /** Constructor that set the topic subrscription,
    * start the timer and start a new thread containing
    * the main loop
    */
  Watcher(ros::NodeHandle * n, timeouts timeout);

  /** Stop the thread
    */
  virtual ~Watcher();

```

```
31    /** Method containing the main loop where the timer is updated
32      * and the conditions are checked
33      */
34    void loop();
35
36    /** ROS callback that reset the timer
37      */
38    void callback(T msg);
39
40    boost::thread * _thr;
41    ros::Time timer;
42    ros::NodeHandle * _node_handle;
43    timeouts _timeout;
44    bool nav_status_initialized_in_node;
45    bool condition;
46    ros::Subscriber subscriber;
47  };
```

A.6 PerceptionAlgorithm

```
1   namespace auvPerception {
2     // Status coding
3     enum {
4       STATE_ERROR = -1,
5       STATE_UNINITIALIZED = 0,
6       STATE_INITIALIZED = 1,
7       STATE_RUNNING = 2,
8       STATE_COMPLETED = 3
9     };
10
11    class PerceptionAlgorithm {
```

```
12    private:
13      Json::Value params;
14      std::string name;
15      int _status;
16    public:
17      PerceptionAlgorithm();
18
19      // Get the current status
20      int getStatus();
21
22      // Check if the process is completed
23      bool isCompleted();
24
25      // Check if the algorithm is initialized
26      bool isInitialized();
27
28      // Retrieve device info params from the server
29      void getDeviceInfo(std::string device_name);
30
31      // Do something for initialize the algorithm
32      virtual bool initialize(std::string device_name)=0;
33
34      // Read the params needed from the server
35      virtual bool readParams()=0;
36
37      // Run the algorithm
38      virtual bool run(void* data);
39
40      // Check if something has been detected
41      bool hasDetected();
42
43      // Get the output formatted as JSON
44      std::string getDetectionAsJsonString();
```

```
45    protected:
46      void setStatus(int status);
47      Json::Value detections;
48
49    };
50  }
```

Bibliografia

[Allotta, 2014] Allotta, B. (2014). http://www.arrowsproject.eu. *Official Site of the European ARROWS Project*.

[Allotta et al., 2014a] Allotta, B., Bartolini, F., Caiti, A., Costanzi, R., Di Corato, F., Fenucci, D., Gelli, J., Guerrini, P., Monni, N., Munafó, A., Natalini, M., Pugi, L., Ridolfi, A., and Potter, J. (2014a). Typhoon at CommsNet 2013: experimental experience on auv navigation and localization. In *Proceedings of the 19th IFAC World Congress*, Cape Town, South Africa.

[Allotta et al., 2014b] Allotta, B., Bartolini, F., Costanzi, R., Monni, N., Pugi, L., and Ridolfi, A. (2014b). Preliminary design and fast prototyping of an autonomous underwater vehicle propulsion system. *Proceedings of the Institution of Mechanical Engineers - Part M, Journal of Engineering for the Maritime Environment*, DOI 10.1177/1475090213514040.

[Allotta and Caiti, 2014] Allotta, B. and Caiti, A. (2014). http://thesaurus.isti.cnr.it. *Official Site of the Italian THESAURUS Project*.

[Allotta et al., 2015a] Allotta, B., Caiti, A., Costanzi, R., Fanelli, F., Fenucci, D., Meli, E., and Ridolfi, A. (2015a). Unscented kalman filtering for

autonomous underwater navigation. In *VI International Conference on Computational Methods in Marine Engineering*.

[Allotta et al., 2015b] Allotta, B., Costanzi, R., Fanelli, F., Monni, N., and Ridolfi, A. (2015b). Single axis fog aided attitude estimation algorithm for mobile robots. *Mechatronics*, 30:158–173.

[Allotta et al., 2014c] Allotta, B., Costanzi, R., Meli, E., Pugi, L., Ridolfi, A., and Vettori, G. (2014c). Cooperative localization of a team of auvs by a tetrahedral configuration. *Robotics and Autonomous Systems*, 62:1228–1237.

[Allotta et al., 2012] Allotta, B., Costanzi, R., Monni, N., Pugi, L., Ridolfi, A., and Vettori, G. (2012). Design and simulation of an autonomous underwater vehicle. In *Proceedings of the European Congress on Computational Methods in Applied Sciences and Engineering, ECCOMAS 2012*, Vienna, Austria.

[Antonelli, 2006] Antonelli, G. (2006). *Underwater Robots*. Elsevier, Amsterdam, Holland.

[Arrichiello et al., 2011] Arrichiello, F., Antonelli, G., Aguiar, A. P., and Pascoal, A. (2011). Observability metric for the relative localization of auvs based on range and depth measurements: Theory and experiments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, San Francisco, CA, USA.

[Bahr et al., 2009a] Bahr, A., Leonard, J., and Fallon, M. (2009a). Cooperative localization for autonomous underwater vehicles. *The International Journal Robotics Research*, 28:714–728.

[Bahr et al., 2009b] Bahr, A., Walter, M., and Leonard, J. (2009b). Consistent cooperative localization. In *Proceedings of the IEEE International Conference of Robotics and Automation (ICRA)*, Kobe, Japan.

[Bar-Shalom et al., 2001] Bar-Shalom, Y., Li, X. R., and Kirubarajan, T., editors (2001). *Estimation with Applications to Tracking and Navigation*. Wiley and Sons, New York, NY, USA.

[Barisic et al., 2012] Barisic, M., Vasilijevic, A., and Nad, D. (2012). Sigma-point unscented kalman filter used for auv navigation. In *20th Mediterranean Conference on Control & Automation (MED)*, Barcelona, Spain.

[Barshan and Durrant-Whyte, 1995] Barshan, B. and Durrant-Whyte, H. F. (1995). Inertial navigation systems for mobile robots. *IEEE Transactions on Robotics and Automation*, 11:328–342.

[Batista et al., 2011] Batista, P., Silvestre, C., and Oliveira, P. (2011). Single range aided navigation and source localization: observability and filter design. *Systems and Control Letters*, 60:665–673.

[Breivik and Fossen, 2005] Breivik, M. and Fossen, T. I. (2005). Guidance-based path following for autonomous underwater vehicles. In *Proceedings of the IEEE OCEANS 2005*, Washington D.C.,USA.

[Caffaz et al., 2010] Caffaz, A., Caiti, A., Casalino, G., and Turetta, A. (2010). The hybrid glider/auv folaga. *IEEE Robotics and Automation Magazine*, 17:31–40.

[Caiti et al., 2013] Caiti, A., Calabrò, V., Fabbri, T., Fenucci, D., and Munafò, A. (2013). Underwater communication and distributed localization

of auv teams. In *Proceedings of IEEE OCEANS 2013*, San Diego, CA, USA.

[Carlton, 2007a] Carlton, J., editor (2007a). *Marine propellers and propulsion*. Elsevier, Amsterdam, Holland.

[Carlton, 2007b] Carlton, J. (2007b). *Marine Propellers and Propulsion, 2nd edition*. Springer-Verlag, Heidelberg, Germany.

[Costanzi, 2015] Costanzi, R. (2015). *Navigation Systems for Unmanned Underwater Vehicles*. PhD thesis, University of Florence.

[Di Corato et al., 2014] Di Corato, F., Fenucci, D., Caiti, A., Costanzi, R., Monni, N., Pugi, L., Ridolfi, A., and Allotta, B. (2014). Toward underwater acoustic-based simultaneous localization and mapping. experimental results with the typhoon auv at commsnet13 sea trial. In *Proceedings of OCEANS'14 MTS/IEEE St. John's Oceans*.

[Eustice et al., 2011] Eustice, R., Whitcomb, L., Singh, H., and Grund, M. (2011). Experimental results in synchronous-clock one-way-travel time acoustic navigation for autonomous underwater vehicles. In *IEEE International Conference on Robotics and Automation (ICRA)*, Rome, Italy.

[Evensen, 2009] Evensen, G., editor (2009). *Data Assimilation*. Springer Verlag, Heidelberg, Germany.

[Fitzgibbon et al., 1999] Fitzgibbon, A., Pilu, M., and Fisher, R. (1999). Direct least square fitting of ellipses. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 21.

[Fjellstad and Fossen, 1994] Fjellstad, O. and Fossen, T. I. (1994). Position and attitude tracking of auv's: A quaternion feedback approach. *IEEE Journal of Oceanic Engineering*, 19:512–518.

[Fossen, 1994] Fossen, T. (1994). *Guidance and Control of Ocean Vehicles*. John Wiley and Sons, London, UK.

[Gao and Chitre, 2010] Gao, R. and Chitre, M. (2010). Cooperative positioning using range-only measurements between two auvs. In *Proceedings of IEEE OCEANS 2010*, Sidney, Australia.

[Gebre-Egziabhern, 2004] Gebre-Egziabhern, D. (2004). Design and performance analysis of a low-cost aided dead reckoning navigator. In *Ph.D. Thesis*, Standford University, CA, USA.

[Halíř and Flusser, 1998] Halíř, R. and Flusser, J. (1998). Numerically stable direct least square fitting of ellipses. In *WSCG International Conference in Central Europe on Computer Graphics and Visualization*.

[Julier and Uhlmann, 1997] Julier, S. and Uhlmann, J. (1997). A new extension of the kalman filter to nonlinear systems. *IEEE Transactions on Automatic Control*, pages 182–193.

[Julier and Uhlmann, 2004] Julier, S. and Uhlmann, J. (2004). Unscented filtering and nonlinear estimaion. *Proceedings of the IEEE*, 92:401–422.

[Kalman, 1960] Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82:35–45.

[Kinsey et al., 2014] Kinsey, J., Yang, Q., and Howland, J. (2014). Nonlinear dynamic model-based state estimators for underwater navigation of remo-

tely operated vehicles. *Control Systems Technology, IEEE Transactions on*, 10.1109/TCST.2013.2293958.

[Larsen, 2000] Larsen, M. (2000). Synthetic long baseline navigation of underwater vehicles. In *Proceedings of MTS/IEEE OCEANS 2000*, Providence, RI, USA.

[Lee et al., 2007] Lee, P. M., Jun, B. H., Kim, K., Lee, J., Aoki, T., and Hyakudome, T. (2007). Simulation of an inertial acoustic navigation system with range aiding for an autonomous underwater vehicle. *IEEE Journal of Oceanic Engineering*, 32:327–345.

[Mackenzie, 1981] Mackenzie, K. (1981). Nine-term equation for the sound speed in the oceans. *J. Acoust. Soc. Am.*, 70:807–812.

[Mahony et al., 2008] Mahony, R., Hamel, T., and Pflimlinn, J. M. (2008). Nonlinear complementary filters on the special orthogonal group. *IEEE Trans. Automat. Control*, 53:1203–1218.

[Marco and Healey, 2001] Marco, D. and Healey, A. (2001). Command, control, and navigation experimental results with the nps aries auv. *IEEE Journal of Oceanic Engineering*, 26:466–476.

[Marins et al., 2001] Marins, J., Yun, X., Bachmann, E., McGhee, R., and Zyda, M. (2001). An extended kalman filter for quaternion-based orientation estimation using marg sensors. In *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, Maui, HI.

[Morgado et al., 2010] Morgado, M., Batista, P., Oliveira, P., and Silvestre, C. (2010). Position usbl/dvl sensor-based navigation filter in the presence

of unknown ocean currents. In *49th IEEE Conference on Decision and Control (CDC)*, Atlanta, GA, USA.

[Morgado et al., 2014] Morgado, M., Oliveira, P., Silvestre, C., and Vasconcelos, J. F. (2014). Embedded vehicle dynamics aiding for usbl/ins underwater navigation system. *Control Systems Technology, IEEE Transactions on*, 10.1109/TCST.2013.2245133.

[NGDC-NOOA, 2014] NGDC-NOOA (2014). www.ngdc.noaa.gov. *NOAA National Geophysical Data Center - Official Website*.

[OpenStreetMap,] OpenStreetMap. `www.openstreetmap.org`.

[Packard et al., 2013] Packard, G., Kukulya, A., Austin, T., Dennett, M., Littlefield, R., Packard, G., Purcell, M., Stokey, R., and Skomal, G. (2013). Continuous autonomous tracking and imaging of white sharks and basking sharks using a remus-100 auv. In *Proceedings of the IEEE OCEANS 2013*, San Diego, CA, USA.

[Parlangeli et al., 2012] Parlangeli, G., Pedone, P., and Indiveri, G. (2012). Relative pose observability analysis for 3d nonholonomic vehicles based on range measurements only. In *Proceedings of the 9th IFAC Conference on Manoeuvring and Control of Marine Craft*, Arenzano, Italy.

[Petres et al., 2007] Petres, C., Pailhas, Y., Patron, P., Petillot, Y., Evans, J., and Lane, D. (2007). Path planning for autonomous underwater vehicles. *IEEE Transactions on Robotics*, 23:331–341.

[QT,] QT. `www.qt.io`.

[Rigby et al., 2006] Rigby, P., Pizarro, O., and Williams, S. (2006). Towards geo-referenced auv navigation through fusion of usbl and dvl mea-

surements. In *Proceedings of MTS/IEEE OCEANS 2006*, Boston, MA, USA.

[Ristic et al., 2004] Ristic, B., Arulampalam, S., and Gordon, N., editors (2004). *Beyond the Kalman Filter*. Artech House Publishers, Boston, MA, USA.

[ROS,] ROS. The robot operating system. `http://www.ros.org/`.

[Sayed, 2009] Sayed, A. H., editor (2009). *Adaptive Filters*. Wiley and Sons, Hoboken, NJ, USA.

[Shin, 2005] Shin, E. (2005). Estimation techniques for low-cost inertial navigation. In *M.Sc. Thesis*, Department of Geomatics Engineering, University of Calgary, Canada.

[Siciliano and Khatib, 2001] Siciliano, B. and Khatib, O., editors (2001). *Handbook of Robotics*. Springer Handbooks, Stanford, CA, USA.

[Singh et al., 2006] Singh, S., Grund, M., Bingham, B., Eustice, R., Singh, H., and Freitag, L. (2006). Underwater acoustic navigation with the whoi micro-modem. In *Proceedings MTS/IEEE Conference and Exhibition OCEANS 2006*, Boston, MA, USA.

[Stokey et al., 2005] Stokey, R., Roup, A., von Alt, C., Allen, B., Forrester, N., Austin, T., Goldsborough, R., Purcell, M., Jaffre, F., Packard, G., and Kukulya, A. (2005). Development of the remus 600 autonomous underwater vehicle. In *Proceedings of the IEEE OCEANS 2005*, Washington D.C.,USA.

[Thrun et al., 2005] Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic Robotics*. The MIT Press.

[UniWisconsin, 2014] UniWisconsin (2014). http://www.uwgb.edu. *University of Wisconsin - Green Bay Web Site*.

[VectorNav-Technologies, 2014] VectorNav-Technologies (2014). http://www.vectornav.com. *Official website of VectorNav Technologies*.

[Wan and Merwe, 2001] Wan, E. A. and Merwe, R. V. D. (2001). *The Unscented Kalman Flter*. S. Haykin Edition, New York, NY, USA.

[Webster et al., 2009] Webster, S., Eustice, R., Singh, H., and Whitcomb, L. (2009). Preliminary deep water results in single-beacon one-way travel-time acoustic navigation for underwater vehicles. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, St. Louis, MO, USA.

[Wolfram-MathWorld, 2014] Wolfram-MathWorld (2014). http://www.mathworld.wolfram.com. *Wolfram MathWorld - Official Website*.

[Yun et al., 1999] Yun, X., Bachmann, E., McGhee, R., Whalen, R., Roberts, R., Knapp, R., Healey, A., and Zyda, M. (1999). Testing and evaluation of an integrated gps/ins system for small auv navigation. *IEEE Journal of Oceanic Engineering*, 24:396–404.

[Zhao and Gao, 2004] Zhao, L. and Gao, W. (2004). The experimental study on gps/ins/dvl integration for auv. In *Position Location and Navigation Symposium (PLANS)*, Monterey, California.