ELSEVIER

Robotics and Autonomous Systems 16 (1995) 29-38

Robotics and Autonomous Systems

Control of camera motions from the planning of image contours

Carlo Colombo[1], Eckhard Kruse[2], Paolo Dario

ARTS Lab, Scuola Superiore Sant'Anna, Via Carducci, 40, 56127 Pisa, Italy

Abstract

A robotic system for camera positioning with respect to fixed planar objects is proposed, whose control is based on planning shape transformations of the objects' *silhouette* in the image. The control equations are linearized by means of an affine camera projection model, and even more simplified by maintaining fixation onto the objects during camera motions. No accurate camera calibration is required, and image contour-based visual analysis provides the system with a robust sensory feedback. The implementation with an eye-in-hand robotic setup is described and discussed. Applications are in the domains of purposive navigation and vision-based grasping.

Keywords: Robot vision; Visual servoing; Motion planning; Active contours

1. Introduction

A major issue of research in robotics is the development of complex tasks as the result of a close cooperation between environment perception (*exterocepsis*) and motor action. The cooperation is often structured as a feedback loop, where robot actuators are controlled on the basis of both the desired task, and a set of parameters describing the current state of the environment. The more complex the representation—or, in ultimate analysis, the sensory processing—the slower the motor response to an environmental stimulus, and the higher the risk of system instability. Therefore, toward the design of a robust, real-time sensorimotor control strategy, care should be taken so as to keep the representation set as small as possible, by regarding perception in the context of the task the system is engaged in, and not in a separate way [4].

The use of visual sensors in the exteroceptive feedback loop (*visual servoing*) appears to be a natural choice in the design of complex robot tasks. Yet, the large amount of data to be processed makes it particularly difficult to achieve real-time performance using visual servoing. The paradigm of *active vision* provides a way to cope with such difficulty, by combining task-oriented visuomotor strategies with anthropomorphic visual mechanisms such as space-variant sensing, selective attention and fixation control [13].

The task of *relative positioning*, i.e. establishing or maintaining with a robot a given spatial configuration with respect to an object in the environment—is of special interest for its applications in several research fields in robotics, such as navigation (correcting the robot position with respect to a landmark, wall-following) and sensory-guided grasping (placing an end-effector in the work space). Such a task involves the control of six Cartesian degrees of free-

[1] Corresponding author.
Tel: +39-(0)50-883207;
Fax: +39-(0)50-883215;
E-mail: Columbus@shamash.sssup.it

[2] Visiting scientist from the Technical University of Braunschweig, Germany, with a fellowship of the EC "Erasmus" Program.

SSDI 0921-8890(95)00031-3

dom which, in a visual servoing system, can be associated without loss of generality to the parameters of the transformation between a camera-centered and an object-centered frame. A single control loop strategy for visual relative positioning has been recently proposed by Espiau et al. [7], in which the feedback error is measured in the image instead of in space. The method is based on tracking a suitable number of simple geometrical primitives—say, points or lines—in the image plane, and solving for robot motions on the basis of a desired 2D trajectory of such primitives. The method is more robust than 3D approaches in the presence of system inaccuracies (camera calibration, kinematic modeling). However, it has a rather high computational complexity and, being an absolute metric approach, it requires a full camera calibration. In a recent paper, Grosso et al. [8] demonstrated the advantages of constraining the visual system to actively maintain fixation onto an environmental point while moving. In fact, in this case, the feedback loop can be fully decoupled yielding a significant complexity reduction, and it is also possible to adopt a relative metric approach for relative positioning, thus avoiding to accurately calibrate the camera.

Despite the significant advances of the recent past, much work has still to be done concerning both robustness and speed performance in visual servoing systems. For instance, visual analysis through the tracking of points/lines in the image plane, when possible, is usually not reliable and/or not fast enough for many robotic tasks [6].

In this paper, a visual servoing system that integrates active vision principles and simple control strategies for the design of robust relative positioning tasks is presented, and its implementation with an eye-in-hand robotic setup is described. Control of camera motion is based on contour planning and servoing in the image plane, and enjoys the beneficial effects of maintaining fixation onto the object. An affine camera model [11] is adopted, which allows the tracking of planar objects in the environment, and a low-cost extraction of visual parameters from raw image data. Using image contours for representation and tracking allows the system to deal with generic object shapes, thus leading to interesting robotic applications like positioning with respect to natural landmarks for navigation, and manipulation tasks. Due to the intrinsic redundancy in the measurements, visual analysis based on image contours provides also a highly reliable sensory feedback information.

The paper is organized as follows. In Section 2 the theoretical aspects of the approach are discussed, namely the geometry and kinematics of camera-environment interaction, and the proposed strategy for contour planning in the image plane. The eye-in-hand implementation of the system is then described in Section 3, where a special emphasis is put on control aspects and on outlining practical problems encountered. Some concluding remarks are offered in Section 4. The paper concludes with an Appendix devoted to the use of image contours for representation and visual analysis.

2. Theory of operation

The execution of a camera positioning task with respect to a fixed object in the visual environment is sketched in Fig. 1 (left). The object is represented in terms of its *silhouette*, or image contour, and it is assumed to be quasi-planar—a computer screen, a panel on the wall, a book, a cookie box, etc. A reference image contour, or *template*, corresponding to a reference spatial configuration (pose, distance) between camera and object to be reached, is also stored in the system memory. Positioning is accomplished by generating a camera motion which allows the initial object contour to be transformed gradually, *in the image*, so as to attain the shape of the template (Fig. 1 (*right*)).

2.1. Geometry and kinematics

In order to describe the geometrical interaction between camera and object, we fix a frame $\{X, Y, Z\}$ to the camera, with Z the depth axis. (As the camera will move in the environment, it is evident that all geometrical entities and relationships referred to the camera-centered frame are time-dependent. However, for the sake of simplicity we will often avoid to mention explicitly the independent variable t in the equations.) We express the planarity constraint for the object as

$$Z(X,Y) = pX + qY + c, \tag{1}$$

where the depth gradient components p and q encode the instantaneous orientation of the object plane's normal unit vector with respect to the depth axis. As-

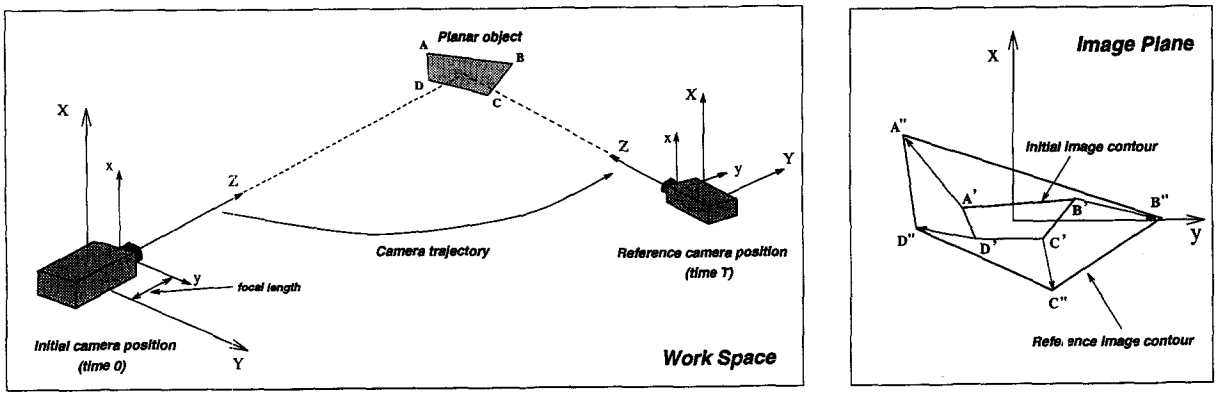


Fig. 1. Left: geometry of camera-object interaction. Right: affine-related image contours.

suming the transversal dimensions of the object to be "small" (say, by one order of magnitude) with respect to its average depth d, the pinhole projection of a world point $(X, Y, Z)^{\mathrm{T}}$ onto the corresponding image point $\boldsymbol{x} = (x, y)^{\mathrm{T}}$ is well approximated by an affine (linear) transformation [11]. (This camera model is a generalization of the orthographic projection.) The planarity condition, together with the affine camera geometry assumption, ensures that *any two contour views of the same object are linearly related.* In particular, let the initial (time $t = 0$) contour and the reference template (to be reached at time $t = T$) be denoted by the two image point sets $\{\boldsymbol{x}_0\}$ and $\{\boldsymbol{x}_{\mathrm{T}}\}$, respectively. Then, the affine transformation "in the large" between *omologous points* of these contours—that is, points in the two contours which are image projections of the same world point—can be expressed as

$$\boldsymbol{x}_{\mathrm{T}} = \boldsymbol{x}_{\mathrm{T}}^{\mathrm{B}} + \mathcal{A}_{\mathrm{T}}(\boldsymbol{x}_o - \boldsymbol{x}_o^{\mathrm{B}}), \tag{2}$$

with $\mathcal{A}_{\mathrm{T}}$ a 2×2 matrix, and B indicating the centroid of the contours.

We can also imagine to obtain the reference template from the initial contour through a sequence of affine transformations "in the small", such that the speed of each contour point is a linear function of image coordinates:

$$\dot{\boldsymbol{x}} = \dot{\boldsymbol{x}}^{\mathrm{B}} + \mathcal{M}_t(\boldsymbol{x} - \boldsymbol{x}^{\mathrm{B}}). \tag{3}$$

(We use the notation $\mathcal{M}_t$ to emphasize that this 2×2 matrix is constant with respect to space, depending on time only.) Notice that the affine transformation in the plane (3) has in general six degrees of freedom. However, the two degrees of freedom encoded in the term $\dot{\boldsymbol{x}}^{\mathrm{B}}$ and corresponding to a *rigid translation* of the contour, are suppressed by constraining the camera to fixate an object point while moving, as shown in Fig. 1. The *fixation constraint* can be formulated, in our case, in terms of temporal invariance of the contour centroid:

$$\boldsymbol{x}^{\mathrm{B}} = \boldsymbol{0} \quad \forall t \in [0, T], \tag{4}$$

which implies also that $d = c$. The remaining four contour degrees of freedom, encoded in $\mathcal{M}_t$, are related instead to *changes in shape* for the contour, and are conveniently expressed in terms of the *differential invariants* of speed: divergence (div), curl, and two components of deformation (def1,def2). Each invariant results from a simple linear combination of the elements of $\mathcal{M}_t$ (see Appendix), and has a direct interpretation in terms of elementary image contour deformations: the divergence accounts for changes in area, the curl for rigid rotations and the deformation components with expansions/compressions along mutually perpendicular axes, without changes in area [6].

Let us now focus on how camera movements in space, described by three translations $(V_x, V_y, V_z)^{\mathrm{T}}$ and three rotations $(\Omega_x, \Omega_y, \Omega_z)^{\mathrm{T}}$ about the camera coordinate axes, affect contour transformations in the image in the case of fixation. First of all, notice that the fixation condition induces a constraint between camera translations and rotations relative to the camera axes parallel to the image plane. In fact, referring again to Fig. 1 (left), it is clear that in order to maintain fixation, any camera translation parallel to the image plane has to be compensated by a suitable rotation of the camera around the depth axis, and precisely:

$$(\Omega_x, \Omega_y)^{\mathrm{T}} = d^{-1}(V_y, -V_x)^{\mathrm{T}}. \tag{5}$$

As a consequence, in the presence of fixation, *not only the contour degrees of freedom in the image, but also the camera degrees of freedom in space are reduced from six to four.* The dependence of image speed invariants on 3D sensor motions is well known in the literature [10,3]. In the special case of an affine camera fixating a planar object while moving, the following relations hold

$$\mathrm{div} = 2d^{-1}V_z + pV_x + qV_y\,, \tag{6}$$

$$\mathrm{curl} = -2\Omega_z - qV_x + pV_y\,, \tag{7}$$

$$\begin{pmatrix} \mathrm{def1} \\ \mathrm{def2} \end{pmatrix} = d^{-1}\begin{bmatrix} p & -q \\ q & p \end{bmatrix}\begin{pmatrix} V_x \\ V_y \end{pmatrix}. \tag{8}$$

Thus, camera translations along the depth axis affect only area changes for the contour (Eq. (6)), and similarly camera rotations about the depth axis determine only rigid contour rotations (Eq. (7)). Finally, according to Eq. (8) there is a one-to-one correspondence between contour deformations and camera translations parallel to the image plane; notice that this is the only case in which the *relative orientation* between camera and object can change [8].

2.2. Planning and servoing

In Fig. 2 a continuous differential servo loop scheme for visual control of camera motions based on image contour planning and tracking is shown. The scheme does not refer to any particular robotic setup, and considers as commanded speeds the camera rototranslations in the work space (see Section 3 for joint space control using an eye-in-hand setup). A *task planner* generates a sequence of affine transformations of the initial contour aimed at obtaining the desired template with the method of Eq. (3). As fixation is maintained (Eqs. (4) and (5)), the planner generates, at each control step, only a desired change in shape for the contour $\mathcal{M}_t^{\mathrm{plan}}$ corresponding one-to-one to a set of desired differential invariants $\{\mathrm{div}, \mathrm{curl}, \mathrm{def1}, \mathrm{def2}\}^{\mathrm{plan}}$. An *integral feedback* strategy is also employed, which compensates for various modeling and estimation errors based on the analysis of current visual data obtained by tracking the object contour from frame to frame. In this way, an estimate of the current centroid can be used for fulfilling the fixation task, while the current shape of the contour can be compared with the desired one (see Appendix) to yield a suitable error term $\mathcal{M}_t^{\mathrm{servo}}$ to be added to the desired contour evolution $\mathcal{M}_t^{\mathrm{plan}}$. The four invariants obtained from the combination of planning and servoing provide the current desired camera speed by simple inversion of Eqs. (6)–(8). This results in a decoupled scheme which makes it possible to have an independent control of rotation, distance and orientation. Notice that in order to use the control equations, an estimate of the orientation and distance parameters p, q and d is required. The accuracy of this estimate is not critical though, since the feedback closure is at the image (2D), and not at the work space (3D) level: a rough estimate can suffice, even if it can somewhat delay the convergence of control. Such a rough estimate can be obtained from an *initial guess* $\hat{p}_0$, $\hat{q}_0$ and $\hat{d}_0$ provided at start-up, by predicting and filtering the p, q and d parameters by means of a simplified version of the evolution equations for a moving plane (see e.g. [9]). By taking into account the affine camera approximation and the fixation constraint (5), and assuming that the *slant* of the plane be not too large, we obtain

$$\begin{cases} \dot{d} \approx & -V_z \\ \dot{p} \approx & -d^{-1}V_x + q\Omega_z \\ \dot{q} \approx & -d^{-1}V_y - p\Omega_z\,. \end{cases} \tag{9}$$

Let us now go back to the off-line planning of a desired contour shape evolution $\mathcal{M}_t^{\mathrm{plan}}$. Such evolution will be based on the shape of the initial and reference contours (endpoint conditions) and, in ultimate analysis, on a desired affine transformation "in the large", $\mathcal{A}_{\mathrm{T}}$, obtained at system start-up by a least squares comparison of these two contours (see also Appendix) once that at least two omologous contour points are known. In order to obtain smooth (i.e. with gradual velocities and accelerations and beneficial effects also on contour tracking and visual analysis) camera motions, we let the desired contour speed $\dot{\boldsymbol{x}}^{\mathrm{plan}}$ be a *quintic* polynomial in t, and impose that both velocity and acceleration be zero at $t = 0$ and $t = T$. By taking into account the fixation constraint (4), we write the affine mapping between the current desired contour $\{\boldsymbol{x}^{\mathrm{plan}}\}$ and the initial contour $\{\boldsymbol{x}_0\}$ as

$$\boldsymbol{x}^{\mathrm{plan}} = \mathcal{A}\,\boldsymbol{x}_0. \tag{10}$$

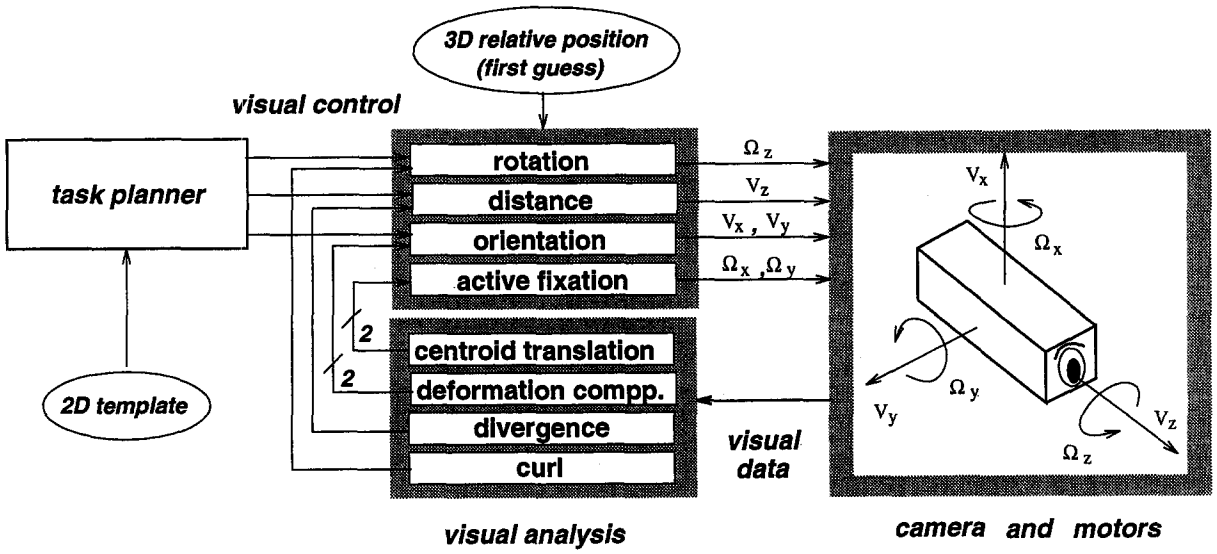


Fig. 2. The control architecture of the system.

Then, differentiating Eq. (10) with respect to time and recalling Eqs. (2 and (3), we easily obtain the following desired motion matrix:

$$\mathcal{M}_t^{\text{plan}} = \frac{\alpha}{T}\frac{\mathcal{A}_T + (\beta\Gamma - 1)\mathcal{I}_2}{\beta^2\Gamma + \beta\Delta + 1}, \qquad (11)$$

where $\Gamma = \det\mathcal{A}_T - \text{tr}\mathcal{A}_T + 1$, $\Delta = \text{tr}\mathcal{A}_T - 2$ and $\mathcal{I}_2$ (the 2×2 identity matrix) are constants, while $\alpha = 30\tau^2(1-\tau)^2$ and $\beta = \tau^3(6\tau^2 - 15\tau + 10) \in [0,1]$ depend on the *normalized task time* $\tau = (t/T) \in [0,1]$. (Using a *cubic* polynomial approximation with zero velocity at the endpoints would lead to a similar result: $\alpha = 6\tau(1-\tau)$ and $\beta = \tau^2(3-2\tau) \in [0,1]$, with faster convergence at the expense of a less smooth trajectory.) Let us illustrate now the characteristics of the planning strategy by referring to the positioning example of Fig. 3, which shows the initial and reference views of a book upon a table. (Notice that, due to the small dimensions of the book, the corresponding contours are affine-related even if perspective effects in the image periphery are clearly visible—the table's sides are not parallel due either to a large slant of the plane (left) or to a small distance (right).) Fig. 4 shows the behavior of the desired 2D invariants produced by the task planner (quintic approximation). These are converted into the 3D camera motion commands of Fig. 5 and, as a result, in changes of relative position. Notice how all the computed quantities vary gracefully, due to the polynomial planning (all curves depart from and terminate to zero with zero first derivative). A characteristic of the proposed planning technique is that motion commands may also invert their sign during positioning (see for instance the V_z curve). Thus for the example given, the reference configuration is reached by first going away from the object, and then approaching it (the changes of contour area are first negative and then positive, as shows the divergence curve), after having smoothly rotated (see the curl and Ω_z curves) and changed orientation (deformations curves).

3. Implementation aspects

The system was implemented and tested using the eye-in-hand setup of Fig. 6. This includes a low-cost camera equipped with a Themis TSVME 630 frame grabber and mounted on the end-effector of a six degrees of freedom GT robot arm, and a Sun SPARCstation 4 for high-level control and user interface. Two Themis TSVME 120 boards control the frame grabber and the manipulator on the VME bus, and communicate with the Sun via two serial lines at 38400 baud—a quite low transmission rate. In order to bypass this communication bottleneck, some preprocessing is carried out directly at the VME boards level,

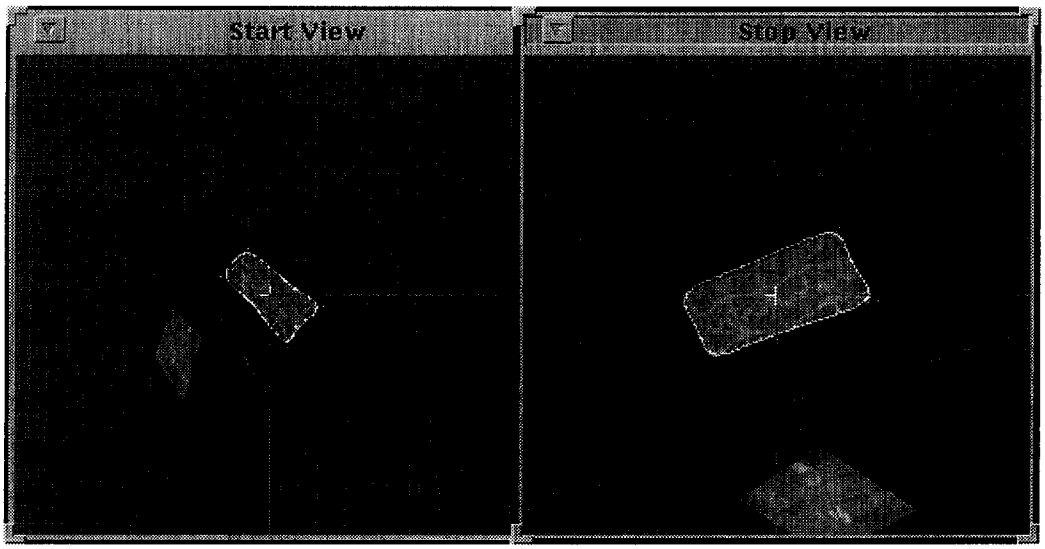


Fig. 3. Left: initial object view. Right: reference object view.

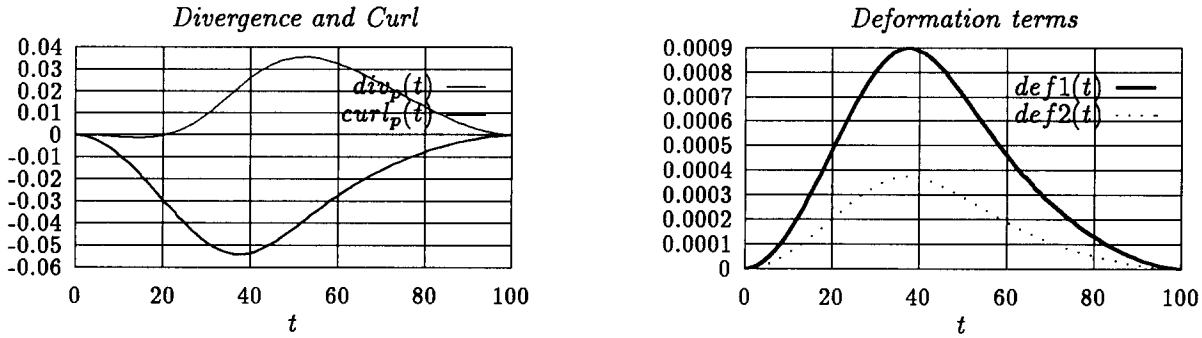


Fig. 4. Differential invariants. Left: desired divergence and curl. Right: desired components of deformation.

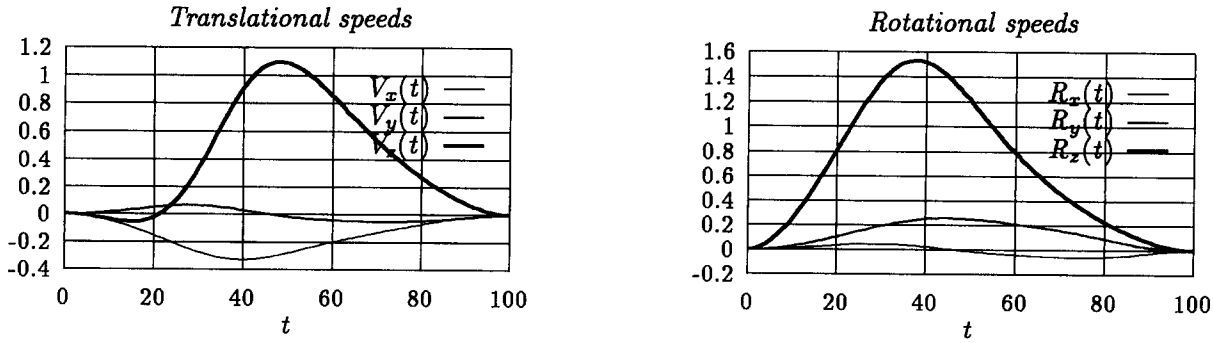


Fig. 5. 3D speeds. Left: translational speeds. Right: rotational speeds.

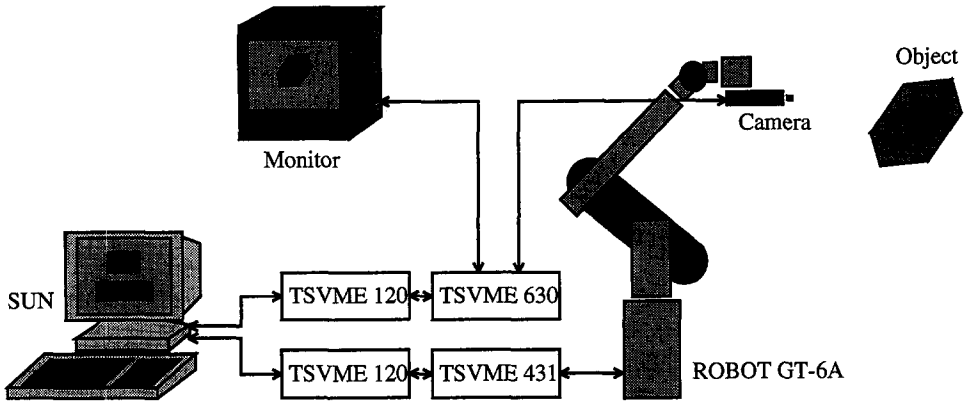


Fig. 6. System implementation.

thus allowing us to drastically reduce the amount of data being transmitted. This proves to be of particular importance for visual computations. In the present implementation, the algorithm proposed recently in [2] was used for the *active tracking* of image contours, which features a B-spline contour representation (see also Appendix), a Kalman filter scheme to add robustness to the tracking behavior, and allows the contour to deform only in an affine fashion, which is exactly our case. Due to the impossibility of transmitting to the Sun the complete 256×256 one-byte pixels image produced by the frame grabber—about 16 seconds would be needed!—the search of brightness gradient maxima around the current contour required by the algorithm is performed on the frame grabber board, and only the changes relative to the current contour are transmitted. The further processing for contour tracking, including the Kalman filtering, takes place on the Sun instead and, at the end of one cycle, the new positions of the control points are returned. Representing B-splines using from around 10 to 20 control points proved to provide enough visual accuracy while keeping computational complexity low.

3.1. The system at work

Before executing the assigned positioning task, a two-step *system initialization procedure* is run in order to define the object to be tracked and to estimate its current distance and orientation. In the first step, the robot is placed so that the object be entirely in the field of view of the camera, with its centroid approximately in the optical center. A number of image points in the surroundings of the object are marked by hand, and used as the control points of the B-spline contour, which then automatically locks onto the intensity edges of the object. (Such a manual procedure could also be easily avoided, e.g. by letting the contour "explode" from the center of the image until it "locks" on the object's *silhouette*.) Notice that, since the shape of the contour is entirely defined by the set of control points, the amount of data to be handled is very small. In the second initialization step, a first guess for the distance and orientation parameters (see also Section 2) is obtained as the result of a "calibration" procedure (Fig. 7). This consists in executing with the robot three translational movements along the current camera frame axes—while always maintaining fixation by keeping the object in the center of the image—and measuring the resulting area changes of the contour by estimating the divergence as shown in the Appendix:

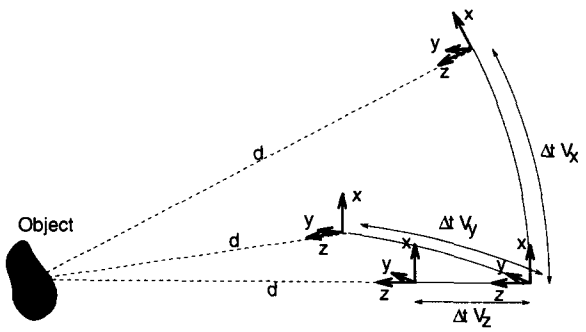


Fig. 7. The "calibration" movements.

$$\begin{cases} \hat{d}_0 = 2V_z/\widehat{\mathrm{div}}, & V_x = V_y = 0, \\ \hat{p}_0 = \hat{d}_0\,\widehat{\mathrm{div}}/V_x, & V_y = V_z = 0, \\ \hat{q}_0 = \hat{d}_0\,\widehat{\mathrm{div}}/V_y, & V_x = V_z = 0. \end{cases} \tag{12}$$

(Alternative 3D parameters estimation schemes could be adopted e.g. using a *stereo head-in-hand* system.) As one can see, the initialization procedure is quite simple and requires virtually no knowledge about camera settings.

At run time, two different kinds of tasks are supported by the system, namely *tracking*—that is, maintaining a given spatial configuration between the object and the camera—and *positioning*—or moving towards a final relative configuration, given in terms of an affine transformation of the contour. The control scheme is quite similar in the two cases. The only difference is that the positioning task is preceded by the additional off-line planning of the smooth affine mapping sequence described in Section 2, while the tracking task is a special case of positioning, characterized by only one differential transformation.

As the control scheme is continuous differential in nature, with desired commands given in terms of Cartesian velocities, a *resolved motion rate control* technique is used for the control of the manipulator. Such a technique has a low computational cost, since it avoids to compute the inverse kinematics of the manipulator, using instead the inverse Jacobian matrix $\mathcal{K}(\boldsymbol{q})$ to transform the desired Cartesian velocities $\dot{\boldsymbol{X}} \doteq (V_x, V_y, V_z, \Omega_x, \Omega_y, \Omega_z)^{\mathrm{T}}$ into manipulator joint velocities $\dot{\boldsymbol{q}} = (\dot{q}_1, \dot{q}_2, \dot{q}_3, \dot{q}_4, \dot{q}_5, \dot{q}_6)^{\mathrm{T}}$. Explicitly, we have

$$\mathcal{K}(\boldsymbol{q}) = \mathcal{J}^{-1}(\boldsymbol{q}), \qquad \dot{\boldsymbol{X}} = \mathcal{J}(\boldsymbol{q})\,\dot{\boldsymbol{q}}, \tag{13}$$

where the Jacobian $\mathcal{J}(\boldsymbol{q})$ depends on the current joint configuration of the manipulator, $\boldsymbol{q}$, and is easily calculated at each control step using standard algorithms [12]. However, notice that in singular manipulator configurations the Jacobian is not invertible; what we actually do compute is a *generalized inverse* of the Jacobian, $\mathcal{K}^{\dagger}(\boldsymbol{q})$, given by

$$\mathcal{K}^{\dagger}(\boldsymbol{q}) = \mathcal{J}^{\mathrm{T}}(\boldsymbol{q})[\mathcal{J}(\boldsymbol{q})\,\mathcal{J}^{\mathrm{T}}(\boldsymbol{q}) + \lambda \mathcal{I}_6]^{-1}, \tag{14}$$

where $\mathcal{I}_6$ is the 6×6 identity, and λ is calculated on the basis of the smallest eigenvalue of the Jacobian [5]. (Of course this generalized inverse coincides with $\mathcal{K}(\boldsymbol{q})$ far from singularities.)

3.2. System characteristics

As a typical performance, the reference configuration is reached with an error of within 5 degrees for relative orientation and a few millimeters for relative distance. The response of the system is also satisfactory, with a cycle time of about 100 msec. A relatively fast and robust system behavior was thus obtained, even without the use of dedicated hardware, thanks to a careful design of the system processing and communication software modules.

We conclude the description of the system developed, by discussing the main difficulties encountered during its implementation and the solutions which were or could be adopted to bypass them.

It is clear that most of the limitations of the current version of the system come from the slow hardware with which it is built. As already mentioned, the connection via the two serial lines was the most serious problem, and it is no doubt that a smaller cycle time could be obtained by substituting it with faster links.

Concerning the software aspects, the main share of processing time is devoted to contour tracking. The complexity of Kalman filtered active contours is dominated by $O(N^2)$ terms, with N the number of control points, so to speed up computations a good practice is to reduce the number of control points to the minimum possible. For instance for a rectangular object, twelve control points placed in triplets near the corners provided good results. However, for an object with a more complicated outline a higher number of control points is crucial to guarantee that the contour remain safely attached to it.

Another problem, connected with the mechanical response of the arm, was to move the robot smoothly since every jolting can cause the active contour to get "distracted". By adjusting the parameters of the Kalman filter, the active contour was given a higher inertia, so as to practically ignore short jolts. Yet, using that possibility the active contour becomes also less reactive, so that the maximum camera speed has to be reduced. Using a camera with a short focal length provides further improvements, because the same angular movement causes less change in the image. Yet there remains the drawback that also the precision of the measurements, which depend on image data, is reduced. With both the robot and the active contour having their own inertias, attention has to be paid to

oscillations. To avoid resonance, the feedback signals have to be damped appropriately. A careful choice of parameters allowed to provide the system with slightly overdamped characteristics.

4. Conclusions

A system for robot relative positioning with respect to planar objects has been proposed, whose control is based on the planning and tracking of image contours. Dramatic simplifications are obtained from active fixation, and the explicit assumption of an affine camera model allows the design of a decoupled control architecture. The system does not require accurate camera calibration. Some of the most relevant features of the system arise from the visual features used, namely B-spline, affinely deformable curves. These can adapt to generic object shapes, and appear to be adequate in applications like positioning with respect to natural landmarks in navigation, and in manipulation tasks. Moreover, visual analysis based on image contours is compact and provides also a robust sensory feedback, due to the intrinsic redundancy in the measurements.

We have also presented and discussed the main features of a system implementation with an eye-in-hand robot configuration provided with some off-the-shelf hardware. A satisfactory performance of the system was obtained both in terms of distance/orientation error and cycle time, thus witnessing the importance of suitable processing and communication software.

Future work will address the problem of generalizing the system to non-planar and moving objects, and extending the proposed method to full perspective images.

Acknowledgements

The authors wish to thank Dr. A.M. Sabatini and Dr. B. Allotta for the useful discussions in the theoretical development of this work.

Appendix A. Affine differential contour transformations

In this Appendix we show how to describe and compute affine differential transformations of contour shape. These are required both for generating the proper 2D feedback signal (see Section 2) and for obtaining a first guess of the 3D orientation and distance parameters between the camera and the object plane (see Section 3). Let $\delta\mathcal{A}$ be one of such transformations "in the small" between the image contours $\{\boldsymbol{x}_1\}$ and $\{\boldsymbol{x}_2\}$:

$$\boldsymbol{x}_2 = \boldsymbol{x}_2^{\mathrm{B}} + \delta\mathcal{A}\,(\boldsymbol{x}_1 - \boldsymbol{x}_1^{\mathrm{B}}). \tag{A.1}$$

Due to the differential nature of the transformation, we can use Eq. (3) and the finite differences approximation to derivatives to relate the two contours to conclude that

$$\begin{pmatrix} \mu_{11} & \mu_{12} \\ \mu_{21} & \mu_{22} \end{pmatrix} \doteq \mathcal{M} \approx \delta\mathcal{A} - \mathcal{I}, \tag{A.2}$$

and derive the differential invariants as [6]

$$\begin{cases} \mathrm{div} = \mu_{11} + \mu_{22}, \\ \mathrm{curl} = \mu_{21} - \mu_{12}, \\ \mathrm{def1} = \mu_{11} - \mu_{22}, \\ \mathrm{def2} = \mu_{21} + \mu_{12}. \end{cases} \tag{A.3}$$

Thus, in order to compute the invariants, it is sufficient to estimate the matrix $\delta\mathcal{A}$ as follows.

Image contours are represented using a quadratic B-spline interpolation:

$$\boldsymbol{x}(s) = \sum_{i=1}^{N} f_i(s)\,\boldsymbol{x}^i, \tag{A.4}$$

where s is arc length, $f_i(s)$ ($\sum_{i=1}^{N} f_i(s) \equiv 1$) are the interpolating functions and the $\boldsymbol{x}^i$ are the N control points of the B-spline [1]. Note that, since every point on the B-spline is a linear combination of the control points, Eq. (A.1) holds true also for control points:

$$\boldsymbol{x}_2^i = \boldsymbol{x}_2^{\mathrm{B}} + \delta\mathcal{A}\,(\boldsymbol{x}_1^i - \boldsymbol{x}_1^{\mathrm{B}}), \quad i \in \{1, \ldots, N\}. \tag{A.5}$$

A *least squares* estimation of the elements of $\delta\mathcal{A}$ can thus be carried out by using only the control points. Explicitly, the unknown matrix entries are obtained by minimizing the quadratic error

$$\varepsilon(\delta\mathcal{A}) = \sum_{i=1}^{N} \| (x_2^i - \hat{x}_2^{\mathrm{B}}) - \delta\mathcal{A}(x_1^i - \hat{x}_1^{\mathrm{B}}) \|^2, \quad \text{(A.6)}$$

where $\| \ \|$ is the usual vector norm, and the centroid is evaluated simply as the average of the control points:

$$\hat{x}^{\mathrm{B}} = \frac{1}{N} \sum_{i=1}^{N} x^i. \quad \text{(A.7)}$$

References

[1] R.H. Bartels, J.C. Beatty and B.A. Barsky, *An Introdution to Splines for Use in Computer Graphics and Geometric Modeling* (Morgan Kaufmann, Los Altos, CA, 1987).

[2] A. Blake, R. Curwen and A. Zisserman, A framework for spatiotemporal control in the tracking of visual contours, *Int. J. Computer Vision* 11 (2) (1993) 127–145.

[3] P. Bouthemy and E. François, Motion segmentation and qualitative dynamic scene analysis from an image sequence, *Int. J. Computer Vision* 10 (2) (1993) 157–182.

[4] R. Brooks, A robust layered control system for a mobile robot, *IEEE J. Robotics and Automation* (1986) 14–23.

[5] S. Chiaverini, Estimate of the two smallest singular values of the Jacobian matrix: Application to damped least-square inverse kinematics, *J. Robotics Systems* 10 (8) (1993) 991–1008.

[6] R. Cipolla and A. Blake, Surface orientation and time to contact from image divergence and deformation, *Proc. 2nd Eur. Conf. on Computer Vision*, S. Margherita Ligure, Italy (1992) 187–202.

[7] B. Espiau, F. Chaumette and P. Rives, A new approach to visual servoing in robotics, *IEEE Trans. Robotics and Automation* 8 (3) (1992) 313–326.

[8] E. Grosso and D.H. Ballard, Head-centered orientation strategies in animate vision, *Proc. 4th Int. Conf. on Computer Vision*, Berlin, Germany (1993) 395–402.

[9] K. Kanatani, Tracing planar surface motion from a projection without knowing the correspondence, *Computer Vision, Graphics, and Image Processing*, 29 (1985) 1–12.

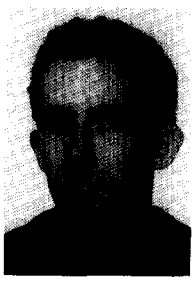
[10] J.J. Koenderink, Optic flow, *Vision Research*, 26 (1) (1986) 161–180.

[11] J.L. Mundy and A. Zisserman, Projective geometry for machine vision, in: J.L. Mundy and A. Zisserman, eds., *Geometric Invariance in Computer Vision* (MIT Press, Cambridge, MA, 1992).

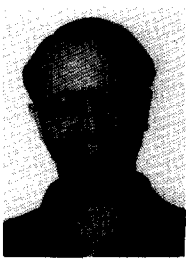
[12] R.P. Paul, *Robot Manipulators: Mathematics, Programming and Control* (Mit Press, Cambridge, MA, 1981).

[13] M.J. Swain and M.A. Stricker, Promising directions in active vision, *Int. J. Computer Vision* 11 (2) (1993) 109–126. Written by the attendees of the NSF Active Vision Workshop, University of Chicago (August 5–7, 1991).

Carlo Colombo received the *Laurea* degree in Electronic Engineering from the University of Florence, Italy, in 1992. He is currently a Ph.D. student in Robotics at the Scuola Superiore Sant'Anna, Pisa, Italy. From September to December 1994 he was a visiting scientist at the National Polytechnic Institute of Grenoble, France, supported by a grant from the EC "Human Capital and Mobility" network SMART. His research interests include pattern recognition, human and robot vision, sensorimotor coordination, machine learning and autonomous robotics. Carlo Colombo is a student member of the IEEE and IAPR societies, and a member of the Organizing Committee of the 3rd International Symposium on Intelligent Robotic Systems SIRS'95, Pisa, Italy.



Eckhard Kruse received in April 1995 the *Diplom* in Informatics from the Technical University of Braunschweig, Germany, where he has begun working towards his Ph.D. From October 1993 to May 1994 he was an exchange student with the EC "Erasmus" Program at the ARTS Lab of the Scuola Superiore Sant'Anna, Pisa, Italy, working on stereo vision and vision-based robot control. His current research interests are in the fields of robot path planning and active vision.



Paolo Dario is an Associate Professor of Biomedical Engineering at the Scuola Superiore Sant'Anna, Pisa, Italy. He earned a Dr. Eng. Degree in Mechanical Engineering from the University of Pisa. He was Visiting Professor at Brown University (Providence, RI), at the Academia Sinica (Bejing, China), at the Ecole Federal Politechnique of Lausanne (Switzerland), and was awarded a Fellowship from the Japanese Government as a Foreign Expert in Robotics.

He teaches a course on Biomedical Technologies and Robotics at the Scuola Superiore Sant'Anna and a course on Mechatronics at the School of Engineering of the University of Pisa. He is also director of the ARTS Lab of the Scuola Superiore Sant'Anna. His main research interests are in the fields of sensors and actuators for biomedical and robotic applications, and of intelligent robotic systems and microsystems.

Prof. Dario is also the scientific responsible for a number of national and european projects within the ESPRIT, BRITE-EURAM, TIDE, SPRINT, "Human Capital and Mobility," and "Erasmus" Programs.

Prof. Dario is a member of the Scientific Board of the Italian National Program on Robotics and of the Italian Society of Robotics. In addition Prof. Dario is the Chairman of the Technical Committee on Micro Robots and Cellular Robots of the IEEE Society of Robotics and Automation.