Collineation estimation from two unmatched views of an unknown planar contour for visual servoing

G. Chesi*, E. Malis# and R. Cipolla#

*Dipartimento di Ingegneria dell'Informazione - Università di Siena
Via Roma, 56 - 53100 Siena - Italy

#Department of Engineering
University of Cambridge
Cambridge CB2 1PZ, UK

Abstract

In this paper we describe a method to compute the collineation matrix between two unmatched images of an unknown planar contour described using a B-spline snake. The two images of the contour are matched and the collineation matrix is used to servo a camera mounted on the robot end-effector using a 2 1/2 D visual servoing technique. The experimental results, obtained using common planar objects, show that our method give very good results and allow the robot end-effector to be positioned with a great precision.

1 Introduction

The visual servoing scheme of robot manipulators can be divided in three steps. In the first off-line learning step, the reference image of the object corresponding to a desired position of the robot is acquired and some image features are extracted. In general, objects are represented by free-form curves, i.e., arbitrary space curves of the type found in practice. A curve is usually described as a set of chained points. The reference image can be obtained using the *teaching by showing technique* (moving the robot in the desired position with respect to the object) or using a CAD model of the object. In the second off-line step, after the robot and/or the object have been moved, the problem is to find the point-to-point correspondence from two images taken at two different arbitrary robot positions. Finally, in the third on-line step, the robot is commanded so that the current features reach their desired position in the image.

Indeed, the second step is crucial for any visual servoing technique, using one or more cameras. Geometric curve matching is a difficult problem in computer vision. In general, the problem can be solved partially by finding a geometric invariant for the curve. Geometric invariants are shape measures that remain constant under change between viewpoints [2] [4]. Indeed, the invariant measured on the object should be constant in all images and thus the matching process has to deal only with the variations between different objects, rather than those produced by viewing a single object. The main methods for curve matching are based on finding invariants to the transformation linking two images of the curve.

Geometric invariants have been studied extensively [10] [5] [6]. However, existing invariants suffer from occlusion and image noise. To cope with these problems, semi-local integral invariants were proposed in [8]. They showed that it is possible to define invariants semi-locally with a lower order of derivatives and hence less sensitive to noise. Although semi-local integral invariants reduce the order of derivatives required, it is still high in the general affine case. A quasi-invariant parametrisation was introduced in [9] which make it possible to use second order derivatives instead of fourth and fifth.

Not only derivatives of high order are difficult to calculate since sensitive to noise, but some curves do not allow the computation of derivatives up to second order (e.g. polygonal curves). Hence, for these contours, it is not possible to use differential or semi local-integral invariants, while it would be possible to use invariants based on features like corners that, however, are not present in smooth curves. Our method allows us to deal with both previous cases. It is based on the hypothesis that the object has already been recognised (by hand in the presented experiments) and that occlusions can occur only during the servoing and not in the matching step. Together with computing the correspondences between the points of the two contours, the homography matrix between the two views is estimated in order to use the visual servoing method proposed in [3]. This approach is called 2 1/2 D visual servoing since the input is expressed in part in the 3D Cartesian space and in part in the 2D image space. More precisely, it is based on the camera displacement estimation from the homography matrix (the rotation and the scaled translation of the camera) between the current and desired views of an object. Using such a scheme, where the rotational control loop can be decoupled from the translational one, the convergence can be ensured in all the task space (i.e. for any initial camera position).

The paper is organised as follows. In the first section, we present the algorithm for matching two views of a contour and, at the same time, for finding the collineation between them. In the second section, we show how to use the collineation to servo the robot with the 2 1/2 D visual servoing technique. In the third section, experimental results are illustrated to demonstrate the validity of our method. Finally, in the conclusion we discuss its possible improvements.

2 Homography estimation

This section describes how estimating the collineation $\mathbf{G}$ existing between the two views of one curve without "a priori" knowing the matching between the points. Let be $\mathbf{S}^*, \mathbf{S} \in \mathbb{R}^{n\times 2}$ the matrices containing respectively the desired B-spline snake and the initial B-spline snake, that is $\mathbf{S}^* = [\mathbf{s}_x^*, \mathbf{s}_y^*]$ and $\mathbf{S} = [\mathbf{s}_x, \mathbf{s}_y]$ (where the vectors $\mathbf{s}_x$ and $\mathbf{s}_y$ contains the coordinates of n points equally distributed on the B-spline snakes). In the first subsection, we analyse the algorithm for determining an estimate of the collineation matrix $\mathbf{G}$. Indeed, after the collineation matrix has been estimated, the homography matrix can be easily computed up to a scalar factor using the matrix of the camera internal parameters $\mathbf{A}$:

$$\mathbf{H} = \mathbf{A}\mathbf{G}\mathbf{A}^{-1} \tag{1}$$

In the second subsection, we show the results obtained with our method.

2.1 The algorithm

The algorithm is made up of two main parts. The first, the correspondence finder, determines the correspondence index q^* between the desired B-spline snake and the initial B-spline snake, that is the index that locates a generic point of $\mathbf{S}^*$ on $\mathbf{S}$. In fact, the two B-spline snakes are built by starting from different points on the curve and there is no a priori correspondence. The second, the collineation finder, calculates the matrix $\mathbf{G}$ taking into account the correspondence found in the first stage. Since the collineation matrix is defined up to a scalar factor, we can set, without loss of generality, $\mathbf{G}(3,3) = 1$. Moreover, we decompose the matrix as

$$\mathbf{G} = \begin{bmatrix} \mathbf{G}^w \\ g_7, g_8, 1 \end{bmatrix} \tag{2}$$

where $\mathbf{G}^w \in \mathbb{R}^{2\times 3}$ is the weak perspective sub-matrix.
Consider first the correspondence finder. We define an arc-length coordinate q in order to compute the matching of the starting point on the B-spline snakes. For each integer $q \in [1, n]$ we determine the correspondence error ϵ_q with the following algorithm:

1) find the least-squares solution $\widehat{\mathbf{G}}_q^w = \mathrm{argmin}_{\mathbf{G}_q^w} \sum_{j=1}^n ||\mathbf{b}_j||^2$ subject to the linear constraint

$$\begin{bmatrix} \mathbf{s}_x^*(j+q) \\ \mathbf{s}_y^*(j+q) \end{bmatrix} = \mathbf{G}_q^w \begin{bmatrix} \mathbf{s}_x(j) \\ \mathbf{s}_y(j) \\ 1 \end{bmatrix} + \mathbf{b}_j \quad \forall 1 \le j \le n \tag{3}$$

where $\mathbf{b}_j \in \mathbb{R}^2$ is the error. An index $j+q$ greater than the length n of the B-spline snake indicates the point number $((j+q-1)\ mod\ n) + 1$, as if the B-spline snake was periodic. In this step, we estimate $\widehat{\mathbf{G}}_q^w$ which depends on the unknown q.

2) compute the reprojected snake $\widehat{\mathbf{S}}^*$ using the estimated weak perspective matrix $\widehat{\mathbf{G}}_q^w$, i.e.

$$\begin{bmatrix} \widehat{\mathbf{s}}_x^*(j) \\ \widehat{\mathbf{s}}_y^*(j) \end{bmatrix} = \widehat{\mathbf{G}}_q^w \begin{bmatrix} \mathbf{s}_x(j) \\ \mathbf{s}_y(j) \\ 1 \end{bmatrix} \quad \forall 1 \le j \le n \tag{4}$$

3) redistribute the points of $\widehat{\mathbf{S}}^*$ equally on the B-spline snake and compute again $\widehat{\mathbf{G}}_q^w$ as in *1)* using the reprojected snake $\widehat{\mathbf{S}}^*$ instead of $\mathbf{S}^*$. The correspondence error is:

$$\epsilon_q = \sqrt{\frac{1}{n}\sum_{j=1}^{n} ||\mathbf{b}_j||^2} \tag{5}$$

The correspondence coordinate q^* is defined as $q^* = \mathrm{argmin}_q\ \epsilon_q$ (i.e. the value of q which minimises the error ϵ_q).

Now, let's see the collineation finder. First of all, we set $\widehat{\mathbf{G}}^w = \mathbf{G}_{q^*}^w$. We find the collineation $\mathbf{G}$ in this way:

1) set $k = 1$ and compute the reprojected snake $\hat{\mathbf{S}}^*$ using the estimated weak perspective matrix $\widehat{\mathbf{G}}^w$ and equally distribute the points of $\widehat{\mathbf{S}}^*$ on the B-spline snake;

2) estimate the two unknowns $[g_7, g_8]$ by solving $[\widehat{g}_7, \widehat{g}_8] = \operatorname{argmin}_{g_7,g_8} \sum_{j=1}^{n} ||\mathbf{b}_j||^2$, subject to:

$$\begin{bmatrix} \widehat{\mathbf{s}}^*_x(j+q^*) \\ \widehat{\mathbf{s}}^*_y(j+q^*) \end{bmatrix} = \frac{\widehat{\mathbf{G}}^w \begin{bmatrix} \mathbf{s}_x(j) \\ \mathbf{s}_y(j) \\ 1 \end{bmatrix}}{[g_7, g_8, 1] \begin{bmatrix} \mathbf{s}_x(j) \\ \mathbf{s}_y(j) \\ 1 \end{bmatrix}} + \mathbf{b}_j \quad \forall 1 \le j \le n \tag{6}$$

3) estimate again $\widehat{\mathbf{G}}^w = \operatorname{argmin}_{\mathbf{G}^w} \sum_{j=1}^{n} ||\mathbf{b}_j||^2$, subject to the constraint (linear in $\mathbf{G}^w$):

$$\begin{bmatrix} \widehat{\mathbf{s}}^*_x(j+q^*) \\ \widehat{\mathbf{s}}^*_y(j+q^*) \end{bmatrix} = \frac{\mathbf{G}^w \begin{bmatrix} \mathbf{s}_x(j) \\ \mathbf{s}_y(j) \\ 1 \end{bmatrix}}{[\widehat{g}_7, \widehat{g}_8, 1] \begin{bmatrix} \mathbf{s}_x(j) \\ \mathbf{s}_y(j) \\ 1 \end{bmatrix}} + \mathbf{b}_j \quad \forall 1 \le j \le n \tag{7}$$

4) set $e_k = \sqrt{\frac{1}{n} \sum_{j=1}^{n} ||\mathbf{b}_j||^2}$ and the collineation matrix to $\widehat{\mathbf{G}} = \begin{bmatrix} \widehat{\mathbf{G}}^w \\ \widehat{g}_7, \widehat{g}_8, 1 \end{bmatrix}$;

5) compute the reprojected snake $\widehat{\mathbf{S}}^*$ using the estimated matrix $\widehat{\mathbf{G}}$, i.e.:

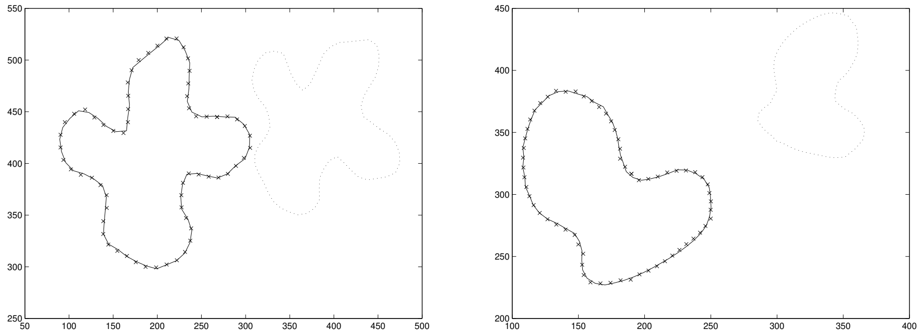
$$\begin{bmatrix} \widehat{\mathbf{s}}^*_x(j) \\ \widehat{\mathbf{s}}^*_y(j) \\ 1 \end{bmatrix} = \frac{\widehat{\mathbf{G}} \begin{bmatrix} \mathbf{s}_x(j) \\ \mathbf{s}_y(j) \\ 1 \end{bmatrix}}{[\widehat{g}_7, \widehat{g}_8, 1] \begin{bmatrix} \mathbf{s}_x(j) \\ \mathbf{s}_y(j) \\ 1 \end{bmatrix}} \quad \forall 1 \le j \le n \tag{8}$$

6) equally distribute the points of $\widehat{\mathbf{S}}^*$ on the B-spline snake, increases k (if $k < k_{MAX}$) and go to *2)*;

The best collineation matrix $\mathbf{G}$ minimise the error e_k. The second part of the algorithm depends on the value q^* found in the first part. Under weak perspective transformations the value will be accurate but not under full perspective transformations. To cope with this problem the algorithm can be iterated using in the first part the parameters $[\widehat{g}_7, \widehat{g}_8]$ estimated in the second part. Obviously, the curves for which we wish to compute the collineation cannot have symmetries (e.g. rotational symmetries). In fact, for these curves, the correspondence is ambiguous if there are not a priori hypothesises.

2.2 Examples

In the figures below we can see the results of the algorithm described in the last subsection. In each image the dotted curve represents the initial view of the object, the solid curve is the desired view. The crosses represent the initial curve projected using the collineation $\mathbf{G}$. Figure 2.2(a) shows the case of translation and rotation around optical axis. The rotations around the other axes are small. Figure 2.2(b) shows a more general case, with big rotations around all the axes. In both case the estimated collineation, obtained with $n = 64$, is in good agreement with the image of the curve.



(a) translation and simple rotation

(b) general translation and rotation

Figure 1: Results using general contours

3 Visual Servoing

3.1 Homography matrix decomposition

After matrix $\mathbf{H}$ is computed, $\mathbf{R}$, $\mathbf{t}/d^*$ and $\mathbf{n}^*$ can be estimated which provides a partial pose estimation. More precisely, $\mathbf{H}$ can be written [1]:

$$\mathbf{H} = \mathbf{R} + \frac{\mathbf{t}}{d^*}\mathbf{n}^{*T} \tag{9}$$

where $\mathbf{R}$ and $\mathbf{t}$ are the rotational matrix and the translational vector between the current camera frame F and the desired camera frame F^* respectively, $\mathbf{n}^*$ is the unit vector normal to the target plane π expressed in F^* and d^* is the distance between the origin of F^* and π. Unfortunately, in the most general case, we have two different solutions. As the target is planar, the indetermination is eliminated by choosing the solution which is such that the vector $\mathbf{n}^*$ is as co-linear as possible with the desired orientation of the camera optical axis. Let us notice that the structure of the reference plane can be directly reconstructed from the homography matrix. For example, the ratio ρ between the Z coordinate of a 3D reference point lying on π and d^* will be used further:

$$\rho = \frac{Z}{d^*} = \frac{\det(\mathbf{H})}{\mathbf{n}^T\mathbf{m}} \tag{10}$$

where $\mathbf{m}$ is the vector containing the metric image coordinates of the point.

3.2 Choosing the control vector

In order to control the camera orientation, we use of course the 3D estimated rotation $\mathbf{R}$ between $\mathcal{F}$ and $\mathcal{F}^*$ (that has to reach the identity matrix). Let $\mathbf{u}$ be the rotation axis and θ the rotation angle obtained from $\mathbf{R}$. The time variation of $\mathbf{u}\theta$ can be expressed as a function of the camera velocity screw $\mathbf{v} = \left[\begin{array}{cc} \boldsymbol{v}^T & \boldsymbol{\omega}^T \end{array}\right]^T$ (where the three dimensional vectors $\boldsymbol{v}$ and $\boldsymbol{\omega}$ are the translational and rotational camera velocity) under the following form:

$$\frac{d(\mathbf{u}\theta)}{dt} = \left[\begin{array}{cc} \mathbf{0} & \mathbf{L}_w \end{array}\right] \mathbf{v} \tag{11}$$

where $\mathbf{L}_w$, given in [3], is such that ${\mathbf{L}_w}^{-1}\mathbf{u}\theta = \mathbf{u}\theta$. The control of the camera orientation is thus decoupled from the control of its position since the former is directly available from the obtained partial pose.

The position of the camera can be controlled in the image space and in the Cartesian space at the same time. Consider a point $\mathcal{P}$ (called the reference point) of the observed object. The time derivative of its coordinates $\mathbf{x}$, expressed in the current camera frame, can be written as:

$$\dot{\mathbf{x}} = \left[\begin{array}{cc} -\mathbf{I}_3 & [\mathbf{x}]_\times \end{array}\right] \mathbf{v} \tag{12}$$

Let us define the *extended image coordinates* $\mathbf{m}_e$ as follows:

$$\mathbf{m}_e = \left[\begin{array}{ccc} x & y & z \end{array}\right]^T = \left[\begin{array}{ccc} \frac{X}{Z} & \frac{Y}{Z} & \log(Z) \end{array}\right]^T \tag{13}$$

where $z = \log(Z)$ is a supplementary normalised coordinate. The time derivative of the extended image coordinates can be written as:

$$\dot{\mathbf{m}}_e = \left[\begin{array}{c} \dot{x} \\ \dot{y} \\ \dot{z} \end{array}\right] = \left[\begin{array}{c} \frac{\dot{X}}{Z} - \frac{X}{Z}\frac{\dot{Z}}{Z} \\ \frac{\dot{Y}}{Z} - \frac{Y}{Z}\frac{\dot{Z}}{Z} \\ \frac{\dot{Z}}{Z} \end{array}\right] = \frac{1}{Z}\left[\begin{array}{ccc} 1 & 0 & -\frac{X}{Z} \\ 0 & 1 & -\frac{Y}{Z} \\ 0 & 0 & 1 \end{array}\right]\left[\begin{array}{c} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{array}\right] = -\frac{1}{Z}\mathbf{L}_v(\mathbf{m})\dot{\mathbf{x}} \tag{14}$$

where $\mathbf{L}_v(\mathbf{m})$ is an upper triangular matrix given by:

$$\mathbf{L}_v(\mathbf{m}) = \left[\begin{array}{ccc} -1 & 0 & x \\ 0 & -1 & y \\ 0 & 0 & -1 \end{array}\right] \tag{15}$$

Then, using equation (12) with equation (14), we finally obtain (since $\mathbf{x} = Z\mathbf{m}$):

$$\dot{\mathbf{m}}_e = \left[\begin{array}{cc} \frac{1}{Z}\mathbf{L}_v(\mathbf{m}) & \mathbf{L}_{v\omega}(\mathbf{m}) \end{array}\right] \mathbf{v} \tag{16}$$

where:

$$\mathbf{L}_{v\omega}(\mathbf{m}) = \mathbf{L}_v(\mathbf{m})[\mathbf{m}]_\times = \left[\begin{array}{ccc} xy & -(1+x^2) & y \\ (1+y^2) & -xy & -x \\ -y & x & 0 \end{array}\right] \tag{17}$$

3.3 The control law

A general positioning task controlling the 6 camera d.o.f. can be described as the regulation to zero of a task function [7]. In the case of 2 1/2 D visual servoing, the positioning task controlling the 6 camera d.o.f. can be described as the regulation to zero of the following task function:

$$\mathbf{e} = \left[\begin{array}{cc} \mathbf{m}_e^T - \mathbf{m}_e^{T*} & \mathbf{u}^T\theta \end{array} \right]^T \tag{18}$$

where the first two components of $\mathbf{m}_e - \mathbf{m}_e^*$ are directly computed from the current and desired images, and its last component, equal to $\log(Z/Z^*)$, is estimated using equation (10). The exponential convergence of $\mathbf{m}_e$ toward $\mathbf{m}_e^*$ and $\mathbf{u}\theta$ toward 0 can be obtained by imposing $\dot{\mathbf{e}} = -\lambda \mathbf{e}$ (where λ tunes the convergence rate). If the target is assumed to be motionless the corresponding control law is given by:

$$\mathbf{v} = -\lambda \widehat{\mathbf{L}}^{-1} \widehat{\mathbf{e}} \tag{19}$$

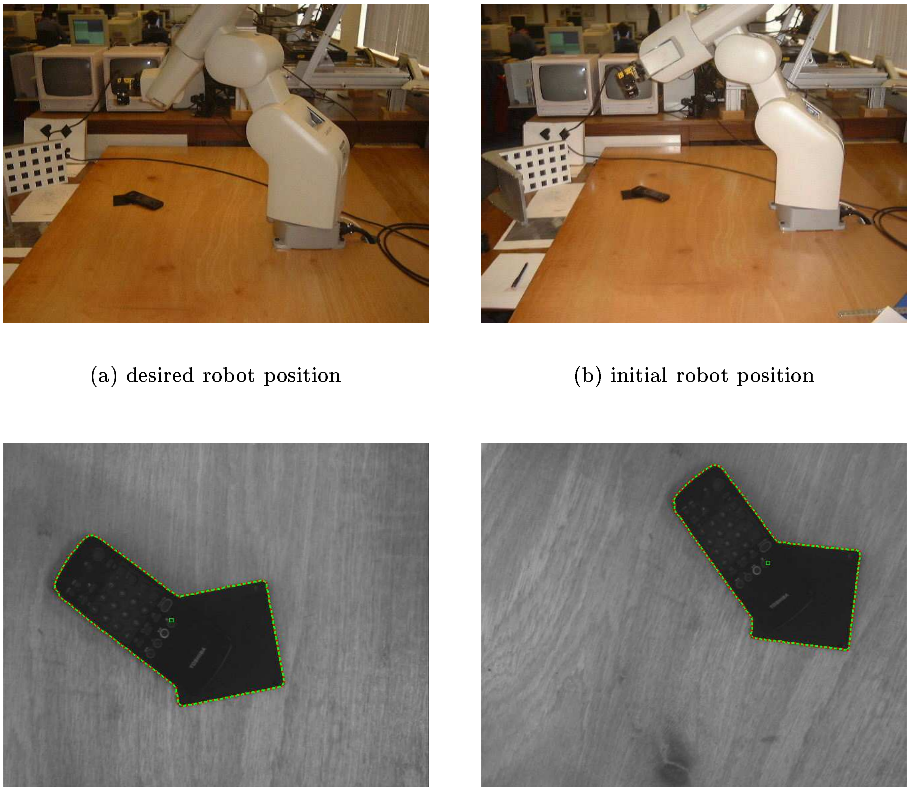
where $\mathbf{v}$ is the camera velocity sent to the robot controller, $\widehat{\mathbf{L}}$ is an approximation of the interaction matrix related to the time variation of the task function $\mathbf{e}$, and $\widehat{\mathbf{e}}$ is the estimated task function. In our case, we have:

$$\mathbf{v} = -\lambda \left[\begin{array}{cc} \widehat{Z}\widehat{\mathbf{L}}_v^{-1} & -\widehat{Z}\widehat{\mathbf{L}}_v^{-1}\widehat{\mathbf{L}}_{v\omega}^{-1}\widehat{\mathbf{L}}_\omega \\ \mathbf{0} & \mathbf{I}_3 \end{array} \right] \left[\begin{array}{c} \widehat{\mathbf{m}}_e - \widehat{\mathbf{m}}_e^* \\ \widehat{\mathbf{u}}\widehat{\theta} \end{array} \right] \tag{20}$$

$\mathbf{L}$ is an upper block triangular matrix. If the point $\mathbf{m}_e$ lies on π, then $Z = \rho\ d^*$ (see equation (10)) and the distance d^* is the only unknown parameter. An approximate value has thus to be chosen during the off-line learning stage. However, this value has not to be precisely determined (by hand in the following experiments) since it has a small influence on the stability of the system. More precisely, it influences the time-to-convergence of the translational velocity and the amplitude of the possible tracking error due to a wrong compensation of the rotational motion. As far as the tracking error is concerned, it is proportional to the rotational velocity and thus disappears when the camera is correctly oriented. Let us remark that the rotational control loop is decoupled from the translational one. A such decoupled system allows to obtain the convergence in all the task space if exact model and perfect measurements are assumed. Furthermore and contrarily to 2D and 3D visual servoings, it is possible to obtain the necessary and sufficient conditions for local asymptotic stability, and sufficient conditions for global asymptotic stability in presence of camera calibration errors (see [3] for more details).

4 Experimental Results

For our experiments, we use a Mitsubishi robot RV-E2 Movemaster with 6 d.o.f. Consider the initial image shown in Figure 2(c), constituted by a floppy disk and a remote control. The selected B-spline snake represents the initial curve. Figure 2(a) shows the initial robot position. We wish to move the robot from this position to the desired position showed in Figure 2(b), corresponding at the object view of Figure 2(d), where the B-spline snake indicates the desired curve. After the homography estimation procedure and visual servoing we obtained the view of the object shown in Figure 3. The desired and final B-spline snakes are shown to be in good agreement. The visual servoing stopped when the error between corresponding points is less than 0.5 pixel. Figure 4 shows the behaviour of the extended image coordinates and orientation of the camera and its translational and rotational velocity (i.e. the control law). The control law is stable and the error converge to zero but not exponentially since the system is coarsely calibrated and the distance between the camera and the target was set to 40 cm while its real value was 60 cm. The convergence of the visual servoing demonstrates that the initial matching was good in spite of the noise and the general camera displacement.



(a) desired robot position

(b) initial robot position

(c) desired image

(d) initial image

Figure 2: initial and desired views of the object

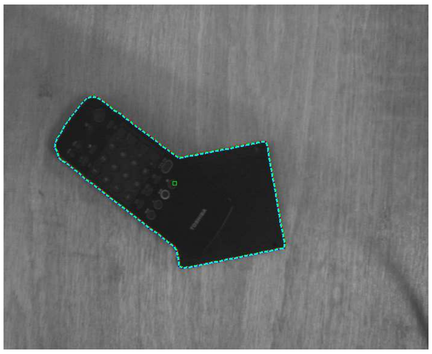
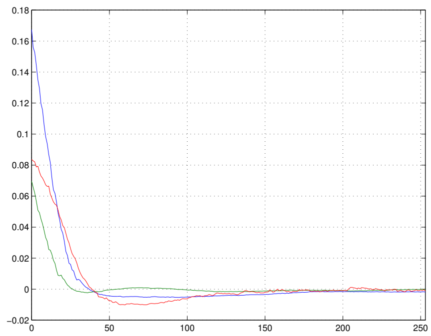
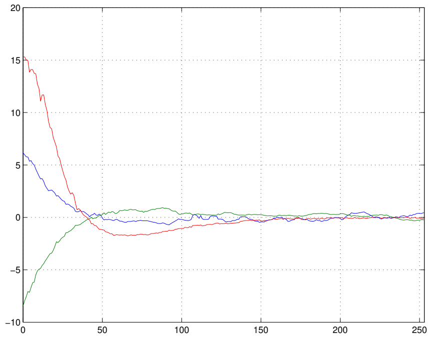


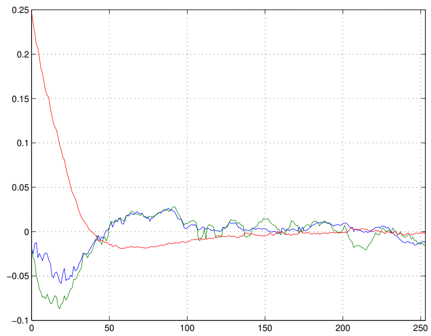
Figure 3: final view.



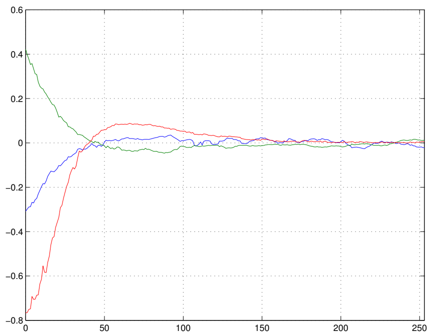
(a) extended image coordinates



(b) rotation $\mathbf{u}\theta$ (deg)



(c) translational velocity (cm/s)



(d) rotational velocity (deg/s)

Figure 4: experimental results with a general camera displacement

5 Conclusion

In this paper we have presented a method for matching and computing the homography matrix between two views of a planar contour. Our methods provides very good results even when geometric invariants cannot be used. The experimental results confirm that recovering the camera motion from an estimated homography matrix gives good results. The homography matrix estimation can be useful in all applications where scaled 3D reconstruction is useful. In our case, the motion parameters extracted from the homography matrix have been used to perform 2 1/2 D visual servoing. This method does not need any 3D model of the target, nor a precise camera calibration and presents very interesting decoupling and stability properties. Future work will be devoted to the extending the method for matching complex images even in case of occlusions.

Acknowledgements

This work was supported by an EC (ESPRIT) grant no. LTR26247 (VIGOR). We would thank Tom Drummond for providing the B-Spline snake software.

References

[1] O. Faugeras and F. Lustman. Motion and structure from motion in a piecewise planar environment. *International Journal of Pattern Recognition and Artificial Intelligence*, 2(3):485–508, 1988.

[2] D. Forsyth, J. Mundy, A. Zisserman, C. Coelho, A. Heller, and C. Rothwell. Invariant descriptors for 3-d object recognition and pose. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(10):971–991, October 1991.

[3] E. Malis, F. Chaumette, and S. Boudet. 2 1/2 d visual servoing. *IEEE Trans. on Robotics and Automation*, 15(2):234–246, April 1999.

[4] J. Mundy and A. Zisserman. *Geometric invariance in computer vision.* MIT Press, 1992.

[5] T. Reiss. *Recognizing planar object using invariant image features.* (LNCS 676), Springer Verlag, 1993.

[6] C. Rothwell. *Object recognition through invariant indexing.* Oxford Science Publications, 1995.

[7] C. Samson, M. Le Borgne, and B. Espiau. *Robot Control: the Task Function Approach*, volume 22 of *Oxford Engineering Science Series.* Clarendon Press, Oxford, Royaume Uni, 1991.

[8] J. Sato and R. Cipolla. Affine integral invariants and matching of curves. In *International Conference on Pattern Recognition*, volume 1, pages 915–919, Vienna, Austria, 1996.

[9] J. Sato and R. Cipolla. Quasi-invariant parametrisations and matching of curves in images. *International Journal of Computer Vision*, 28(2):117–136, June/July 1998.

[10] G. Taubin and D. Cooper. *Object recognition based on moment (or algebraic) invariants*, volume In Geometric Invariance in Computer vision. MIT Press, 1992.