Improving Image-Based Visual Servoing with Three-Dimensional Features

E. Cervera
A. P. del Pobil

Robotic Intelligence Laboratory
Jaume-I University
12071 Castelló, Spain
ecervera@icc.uji.es
pobil@icc.uji.es

F. Berry
P. Martinet

LASMEA - GRAVIR
Blaise Pascal University of Clermont-Ferrand
63177 Aubière - Cedex, France
berry@lasmea.univ-bpclermont.fr
martinet@lasmea.univ-bpclermont.fr

Abstract

Neither of the classical visual servoing approaches, position-based and image-based, are completely satisfactory. In position-based visual servoing the trajectory of the robot is well stated, but the approach suffers mainly from the image features going out of the visual field of the cameras. On the other hand, image-based visual servoing has been found generally satisfactory and robust in the presence of camera and hand–eye calibration errors. However, in some cases, singularities and local minima may arise, and the robot can go further from its joint limits. This paper is a step towards the synthesis of both approaches with their particular advantages, i.e., the trajectory of the camera motion is predictable and the image features remain in the field of view of the camera. The basis is the introduction of three-dimensional information in the feature vector. Point depth and object pose produce useful behavior in the control of the camera. Using the task-function approach, we demonstrate the relationship between the velocity screw of the camera and the current and desired poses of the object in the camera frame. Camera calibration is assumed, at least coarsely. Experimental results on real robotic platforms illustrate the presented approach.

KEY WORDS—hybrid, depth, pose, object model, trajectories

The International Journal of Robotics Research
Vol. 22, No. 10–11, October–November 2003, pp. 821-839,

1. Introduction

This work aims to improve the behavior of image-based visual servoing. As pointed out in Chaumette (1998) and Hutchinson, Hager, and Corke (1996), in some cases, convergence and stability problems may occur. Singularities in the Jacobian or interaction matrix, or local minima, can spoil down the servoing task. Position-based visual servoing is not free of drawbacks either; image features are not controlled, thus the target may go out from the field of view of the camera. Generally, it is very important to study the behavior of the control loop along the followed trajectory regarding all of these drawbacks (singularities, local minima, keeping the target in the field of view of the camera). During the last decade, many works have been done in this way, in order to solve such problems.

In image-based control approach, the ideal case is to find a particular visual feature where the interaction matrix has no local minima nor singularities, and where the exponential decrease of the corresponding error function involves a straight three-dimensional (3D) trajectory between the initial and final camera poses. Most of time people use the interaction matrix computed at the equilibrium; this may help to avoid singularities and the need for 3D depth estimation, but it affects the trajectory of the camera.

Previous works using image point features present an explicit trajectory generation expressed in image space (Berry, Martinet, and Gallice 1997, 1999). In this case, the motion is controlled only in such image space. Furthermore, they all rely on metric knowledge. Other alternatives for improving image-based visual servoing have been proposed, such

as combining the regulation of the vision task with the minimization of a secondary cost function, to avoid joint limits and kinematic singularities (Marchand, Chaumette, and Rizzo 1996). In the case of complex objects, several works based on geometrical features (Berry, Martinet, and Gallice 2000) or an automatic selection of visual features (Janabi-Sharifi and Wilson 1997) coupled to an SSD optical flow technique (Papanikolopoulos 1995) have been developed. Another method for tracking complex objects based on the estimation of the two-dimensional (2D) object image motion along with the computation of its 3D pose is presented in Marchand et al. (1999).

More recent improvements have been presented in Corke and Hutchinson (2000) and Mezouar and Chaumette (2001). The authors use potential fields in image space in order to keep the object in the field of view of the camera. In addition, Mezouar (2001) has extended the local property of convergence in the whole operating space by proposing an image trajectory generation and control under constraint (in order to avoid singularities, keep the object in the field of view of the camera, and obtain 3D straight line behavior of the camera pose). Hashimoto and Noritsugu (2000) have used intermediate reference images obtained by interpolation. One of the main problems of such techniques is to assume that the motion between each generated images is compatible and smooth enough for a robot. In particular, if the robot is a nonholomous one, then the generated images have to take into account such a constraint. Another choice is to use a robust control technique such as robust quadratic stabilization (Tarbouriech and Soueres 2000) or dissipation theory (Maruyama, Fujita, and Kanitani 1999). Generally, these techniques are very complex and time-consuming, and the result is a conservative system with very low performances.

In the position-based control approach (Wilson, Williams Hulls, and Bell 1996; Martinet, Gallice, and Khadraoui 1996), most of representations avoid the problems of local minima or singularities of the corresponding interaction matrices, and very often a straight 3D line between the initial and final embedded camera pose is obtained. Unfortunately, the problem of the features going out of the camera view is not directly solved. Zanne, Morel, and Piestan (2000) have used sliding model control theory to design a 3D vision-based controller that is robust to bounded parametric estimation errors. In Martinet and Gallice (1999) and Thuilot et al. (2002), new 3D orientation features are used in the control loop with a nonlinear approach, in order to keep the target in the field of view.

A mixed 2D–3D approach can be used in order to take advantage of both approaches. This is done in the so-called $2\frac{1}{2}$D visual servoing (Malis, Chaumette, and Boudet 1999; Chaumette and Malis 2000). This approach consists of combining visual features obtained from the image, and features expressed in the Euclidean space. The 3D information can be retrieved either by a pose estimation algorithm (if a CAD model of the target is known), or by a projective reconstruction, obtained from the current and desired images. In either case, the rotation $\mathbf{R}$ of the camera and a depth ratio need to be computed. In Morel et al. (1999) the size of the shape in the image space is taken into account in order to keep the object in the field of view of the camera. An ellipsis is defined which includes all the features used in the reconstruction algorithm, and a control law is designed to keep such ellipsis within the image bounds. Another natural way to mix 2D and 3D is to use a binocular system. Since pioneer works by Maru et al. (1993), binocular vision has often been used to provide better robustness to calibration errors. Grosso et al. (1996) have compared a continuous measure of the end-effector motion field with the actual position of the target. Lamiroy, Puget, and Horaud (2000a) have described a number of geometric tools to obtain a very robust visual stereo servoing platform.

The approach proposed in this paper defines alternative feature vectors combining 2D and 3D information. It should be noted that 3D information (depth) has been required in the classic image-based approach for computing the interaction matrix. Since this information is available, it makes sense to use it not only in the interaction matrix but in other phases of the servoing scheme.

Using the *task-function* framework (Samson, Le Borgne, and Espiau 1991; Espiau, Chaumette, and Rives 1992), the velocity screw of the camera can be obtained from the task error and the pseudo-inverse of the interaction matrix. We will demonstrate that, for small changes of orientation, the computed screw depends only on the current and desired poses of the object in the camera frame, and consequently there are no singularities nor local minima in the trajectory of the camera.

As a further step, the use of 3D coordinates of points is explored, which turns out to be a particular case of the previous one. Nonetheless, stronger results for the velocity screw are obtained for some particular geometric configurations of the target object. Although 3D visual features have been studied before (see, for example, Martinet, Gallice, and Khadraoui 1996), a new approach is proposed and analytical results are presented for the velocity screw based on a set of feature points.

Finally, object pose is used in the feature vector, as a natural extension of 3D points. This approach is shown to best combine the advantages of both position-based and image-based visual servoing schemes, since the trajectory of the object in the camera view is obtained, and the trajectory of the camera in space can be estimated.

In Section 2 pixel information is used in combination with depth, showing the resulting interaction matrix and the velocity screw of the camera. Theoretical results are presented for a stereo system where depth is estimated from pixel disparity. The same approach leads to the use of 3D point coordinates in the feature vector. Next, in Section 3 we present a new approach using 3D pose information of the object (position and orientation) as a control feature. Theoretical results for

the interaction matrix, the velocity screw, and the trajectories are shown. Experimental results are described in Section 4. Finally, in Section 5 we draw some conclusions and open issues. At the end of the paper, an Appendix is provided with the complete mathematical derivations of the results presented in the paper.

2. Image Points and Depth

The main advantage of image-based visual servoing is the direct utilization of image features, without the need to compute the 3D pose of the target. However, in the original approach, depth information corresponding to each image point must be introduced in the interaction matrix, as well as the camera calibration parameters. In fact, if this information is known, the 3D coordinates of the points can be readily obtained, and 3D image features can be used.

In image-based visual servoing, a feature vector $\mathbf{s}$ has to reach a desired value $\mathbf{s}^*$. Usually, $\mathbf{s}$ is composed of the image coordinates of several points of the object. The key issue in visual servoing is to find the relationship between the derivative of the feature vector and the velocity screw of the camera

$$\mathbf{v} = \begin{pmatrix} \boldsymbol{\upsilon} \\ \boldsymbol{\omega} \end{pmatrix}:$$

$$\dot{\mathbf{s}} = \mathbf{L}_s \mathbf{v}. \tag{1}$$

Here $\mathbf{L}_s$ is the *interaction matrix* or *Jacobian matrix*.

If the feature vector $\mathbf{s}_i$ is composed of the image coordinates $(u_i, v_i)^{\mathrm{t}}$ of a single 3D point $\mathbf{p}_i$ $(X_i, Y_i, Z_i)^{\mathrm{t}}$, then the interaction matrix is

$$\mathbf{L}_{s_i}(\mathbf{s}_i, Z_i, \mathbf{A}) = \begin{pmatrix} \alpha_u & \alpha_{uv} \\ 0 & \alpha_v \end{pmatrix} \begin{pmatrix} -\frac{1}{Z_i} & 0 & \frac{x_i}{Z_i} & x_i y_i & -(1+x_i^2) & y_i \\ 0 & -\frac{1}{Z_i} & \frac{y_i}{Z_i} & (1+y_i^2) & -x_i y_i & -x_i \end{pmatrix}. \tag{2}$$

Here, Z_i is the z-coordinate of $\mathbf{p}_i$, $\mathbf{A}$ is the matrix of the camera intrinsic parameters (see, for example, Faugeras (1993) for more details)

$$\mathbf{A} = \begin{pmatrix} \alpha_u & \alpha_{uv} & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3}$$

and $(x_i, y_i)^{\mathrm{t}}$ are obtained by perspective projection. When several image points are used, the interaction matrix is obtained by simply stacking the matrices for each elementary point.

In the following, 2D information (pixels) is combined with 3D data (depth) in order to improve the behavior of the servoing task. In Section 2.1 we present a feature vector combining pixels and depth (Z-coordinate) of points, and we describe the properties of the interaction matrix, the velocity screw, and the resulting motion of the object in the camera frame. In Section 2.2 we present an alternative formulation of the feature vector using stereo disparity, and show how the same interesting properties are obtained.

2.1. Pixels and Depth

A requirement in image-based visual servoing with points is the value of Z_i for each feature point, in order to compute the interaction matrix. Although some approaches only use the value of the depth at the equilibrium, a better behavior is obtained if it is estimated at each iteration. In this case, depth can be included in the feature vector (Cervera and Martinet 1999a), as presented below, where the interaction matrix and the velocity screw are derived.

2.1.1. Feature Vector

Assuming a camera model without distortion, the relationship between the pixel coordinates $(u_i, v_i)^{\mathrm{t}}$ and the image coordinates $(x_i, y_i)^{\mathrm{t}}$ is linear.

Since the image coordinates are obtained from the perspective projection of the coordinates of each point $\mathbf{p}_i$ in the camera frame, the following relationship is readily obtained:

$$\mathbf{s}_i = \begin{pmatrix} u_i Z_i \\ v_i Z_i \\ Z_i \end{pmatrix} = \mathbf{A}\mathbf{p}_i. \tag{4}$$

The advantage over pure 2D visual servoing is that the feature vector is linearly dependent on the 3D coordinates. The nonlinearity due to perspective projection has been removed. This fact will lead to interesting dynamic properties of the system, as shown in the following. Of course, we assume that Z_i can be estimated, but this assumption is present in 2D visual servoing, since such values are needed in the interaction matrix.

2.1.2. Interaction Matrix

Using the well-known relationships between the velocity of point i of the object and the velocity screw of the camera, the interaction matrix $\mathbf{L}_s$ is

$$\mathbf{L}_s = \begin{pmatrix} \vdots & \\ -\mathbf{A} & \mathbf{A}\left[\mathbf{p}_i\right]_\times \\ \vdots & \end{pmatrix} \tag{5}$$

$$= \begin{pmatrix} \vdots & \\ -\mathbf{A} & \mathbf{A}\left[\mathbf{A}^{-1}\mathbf{s}_i\right]_\times \\ \vdots & \end{pmatrix} \tag{6}$$

where $\mathbf{A}^{-1}$ always exists, since neither α_u nor α_v are null.

Usually, the pseudo-inverse of the interaction matrix is computed numerically at each iteration of the control algorithm. However, it may be derived symbolically, as we show in the next section. For that purpose, the expression of the matrix in eq. (5) is utilized.

We do not intend to use the symbolic result in the control loop, but it will be useful to calculate the velocity screw of the camera.

We will demonstrate that, for small rotations of the camera, such a velocity screw can be approximated by a compact expression which *only depends on the current and desired poses of the object* in the camera frame.

2.1.3. Pseudo-Inverse of the Interaction Matrix

In general, matrix $\mathbf{L}_s$ is not square, thus it is not invertible. However, a pseudo-inverse matrix $\mathbf{L}_s^+$ can be calculated as

$$\mathbf{L}_s^+ = \left(\mathbf{L}_s^{\mathrm{t}}\mathbf{L}_s\right)^{-1}\mathbf{L}_s^{\mathrm{t}} \tag{7}$$

so that $\mathbf{L}_s^+\mathbf{L}_s = \mathbf{I}$.

As derived in the Appendix, the resulting pseudo-inverse matrix is

$$\mathbf{L}_s^+ = \begin{pmatrix} \cdots & -\frac{1}{n}(\mathbf{A}^{\mathrm{t}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{t}} + \left[\mathbf{p}\right]_\times \mathbf{T}_i & \cdots \\ & \mathbf{T}_i & \end{pmatrix} \tag{8}$$

where

$$\mathbf{T}_i = \mathbf{R}\mathbf{M}^{-1}\left[{}^b\mathbf{p}_i\right]_\times \mathbf{R}^{\mathrm{t}}\mathbf{A}^{\mathrm{t}}. \tag{9}$$

Such an expression involves the camera parameters $\mathbf{A}$, the pose of the object in the camera frame, expressed by $\mathbf{p}$ and $\mathbf{R}$, and the 3D model of the object, given by the relative coordinates of each point with respect to the origin of the object $\left[{}^b\mathbf{p}_i\right]_\times$.

There are no obvious conclusions from this expression; in fact, the usual choice is to compute numerically the pseudo-inverse in the control loop. However, it allows the computation of the velocity screw, with some interesting results.

2.1.4. Velocity Screw

The velocity screw is calculated by means of the classical task function approach (Samson, Le Borgne, and Espiau 1991; Espiau, Chaumette, and Rives 1992). The complete derivation is shown in the Appendix. Although no simple expression is obtained in the general case, if the rotation between the current and desired pose of the object θ is small, then the velocity screw is

$$\mathbf{v} \approx -\lambda \begin{pmatrix} (\mathbf{p}^* - \mathbf{p}) + \left[\mathbf{p}\right]_\times \mathbf{R}\mathbf{u}\theta \\ \mathbf{R}\mathbf{u}\theta \end{pmatrix} \tag{10}$$

i.e., it only depends on the pose of the object, no longer depending on the particular 3D model of such an object. We could say that this approach is *invariant* to the 3D characteristics of the object.

In addition, the trajectory of the camera is not a straight line. This could be surprising, since this approach uses 3D information, and *traditional* position-based visual servoing is meant to produce such a straight trajectory. The reason is that the feature vector consists of the pose of the object (as obtained indirectly from the 3D coordinates of its points), not the pose of the camera.

2.1.5. Motion of the Object in the Camera Frame

Since the motion of the camera cannot be easily described, let us calculate the velocity of the center of gravity of the object $\dot{\mathbf{p}}$, with respect to the camera frame. Given the velocity screw of the camera in the general case (45):

$$\begin{aligned} \dot{\mathbf{p}} &= -\lambda\left(-(\mathbf{p}^* - \mathbf{p}) - \left[\mathbf{p}\right]_\times \mathbf{R}\mathbf{W} + \left[\mathbf{p}\right]_\times \mathbf{R}\mathbf{W}\right) \\ &= \lambda(\mathbf{p}^* - \mathbf{p}). \end{aligned} \tag{11}$$

Thus, the center of gravity of the object moves along a *straight-line trajectory* from its initial to its final position in the camera frame. Consequently, the object is most likely to remain in the camera field of view during the whole task.

Let us remark that this expression is an equality, not an approximation. Thus, the center of gravity of the object describes a straight-line trajectory for *any rotation* between the current and desired poses, small or large. Consequently, our approach does not suffer as much as traditional position-based ones from the problem of getting the object out of the field of view of the camera.

Of course, the method relies on the estimation of the depth of points, and the camera focal parameters, which are needed for the interaction matrix. The better the estimation, the more likely the object will remain in the field of view. Unfortunately, it is not possible to derive a quantitative measure of such probability, as in other position-based approaches.

2.2. Using Disparity for Depth

The presented approach can be directly applied to a stereo system, where depth is estimated from pixel disparity (Cervera, Berry, and Martinet 2002). For simplicity, a parallel stereo configuration, with equal focal lengths is assumed. Thus, disparity values can be used in the feature vector, defined as

$$\mathbf{s}_i = \begin{pmatrix} \dfrac{u_{il} + u_{ir}}{u_{il} - u_{ir}} \\ \dfrac{v_{il} + v_{ir}}{u_{il} - u_{ir}} \\ \dfrac{1}{u_{il} - u_{ir}} \end{pmatrix} \tag{12}$$

where subscripts l and r refer to the left and right camera pixels, respectively.

It can be shown that this feature vector results from a linear combination of the coordinates of the corresponding 3D point:

$$\mathbf{s}_i = \mathbf{A}\mathbf{p}_i . \tag{13}$$

The matrix $\mathbf{A}$ depends on the camera parameters as before, and also the camera baseline b, i.e., the distance between left and right cameras:

$$\mathbf{A} = \begin{pmatrix} \frac{2}{b} & \frac{2\alpha_{uv}}{b\alpha_u} & \frac{2u_0}{b\alpha_u} \\ 0 & \frac{2\alpha_v}{b\alpha_u} & \frac{2v_0}{b\alpha_u} \\ 0 & 0 & \frac{1}{b\alpha_u} \end{pmatrix} . \tag{14}$$

The resulting interaction matrix is defined as previously

$$\mathbf{L}_s = \begin{pmatrix} \vdots & \\ -\mathbf{A} & \mathbf{A}\left[\mathbf{A}^{-1}\mathbf{s}_i\right]_\times \\ \vdots & \end{pmatrix} \tag{15}$$

which certainly exists since matrix $\mathbf{A}$ has an inverse too, since none of its elements is null.

In this way, 3D coordinates need not be estimated, and the interaction matrix is kept linear with respect to $\mathbf{s}$.

Previous theoretical results still hold. Although the velocity screw (10) is valid for small angles only, the trajectory of the center of gravity of the set of points (11) still translates along a straight path during the task.

2.3. Using 3D Point Coordinates

In addition to the previous results, theoretical developments can be further obtained if the feature vector is composed of the 3D coordinates of the points of the object, i.e., $\mathbf{s}_i = \mathbf{p}_i$ (Martinet, Gallice, and Khadraoui 1996).

We should note that no additional information is needed for the computation of such coordinates, which are easily obtained from pixels and depth estimates, assuming the camera intrinsic calibration parameters also.

2.3.1. Interaction Matrix

Instead of repeating the development, all of the previous theoretical results can be used by means of simply replacing $\mathbf{A}$ by the identity matrix. Thus, the new interaction matrix is

$$\mathbf{L}_s = \begin{pmatrix} \vdots & \\ -\mathbf{I} & \left[\mathbf{p}_i\right]_\times \\ \vdots & \end{pmatrix} \tag{16}$$

and its pseudo-inverse is

$$\mathbf{L}_s^+ = \begin{pmatrix} \cdots & -\frac{1}{n}\mathbf{I} + \left[\mathbf{p}\right]_\times \mathbf{T}_i & \cdots \\ & \mathbf{T}_i & \end{pmatrix} \tag{17}$$

where

$$\mathbf{T}_i = \mathbf{R}\mathbf{M}^{-1}\left[{}^b\mathbf{p}_i\right]_\times \mathbf{R}^{\mathrm{t}} \tag{18}$$

$$\mathbf{M} = \sum_{i=1}^{n}\left[{}^b\mathbf{p}_i\right]_\times^2 . \tag{19}$$

Let us note that these expressions do not depend on the camera intrinsic parameters (although they are used in the estimation of 3D points from pixel data). In addition, matrix $\mathbf{M}$ is computed from the 3D model of the object, not depending on its absolute spatial position nor orientation.

2.3.2. Velocity Screw

Besides obtaining the same approximation of the velocity screw of the camera, *an exact computation of the velocity screw for any rotation* can be obtained if $\mathbf{M}$ is a diagonal matrix (some objects for $\mathbf{M}$ as a diagonal matrix are, for example, the tetrahedron, the square and the cube).

As derived in the Appendix, the velocity screw of the camera is then

$$\mathbf{v} = -\lambda \begin{pmatrix} (\mathbf{p}^* - \mathbf{p}) + \left[\mathbf{p}\right]_\times \mathbf{R}\mathbf{u}\sin\theta \\ \mathbf{R}\mathbf{u}\sin\theta \end{pmatrix} \tag{20}$$

which is valid for all $\mathbf{u}\theta$.

The motion of the camera is always the same regardless of the 3D model of the object. An intuitive explanation is that the property of $\mathbf{M}$ being symmetric produces some compensation among the motion of the points, leading to a camera motion which only depends on the pose of the object, not its particular points.

As shown in eq. (11), the object will follow a straight-line path in the camera frame. However, care must be taken if $\frac{\pi}{2} < \theta \leq \pi$, since $\sin\theta$ decreases. Consequently, the camera *accelerates* during the first phase of the motion until $\theta = \frac{\pi}{2}$ then decelerates until convergence. This may cause unexpected behavior if the initial velocity of the camera is near to the maximum limit in order to improve convergence times.

Finally, if $\theta = \pi$ (180°) then the camera does not rotate at all, which will produce inconsistent results if the translational motion *does not change the orientation of the object with respect to the camera*, e.g., if the object is positioned along the straight line which joins the current and desired camera positions. Since the object is somehow symmetric, a 180° rotation may leave the object unchanged, thus not producing any rotational motion of the camera.

3. Object Pose

In the previous section, it has been theoretically shown how 3D information improves the behavior of image-based visual servoing. Effectively, it allows us to compute the velocity screw and predict the motion of the object in the camera frame.

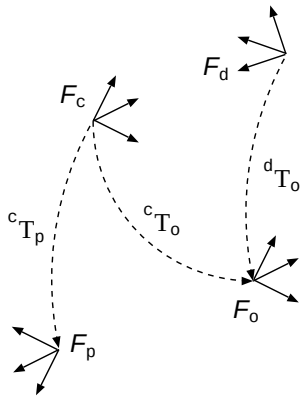


Fig. 1. Frames defined in the visual servoing task.

In this section, the feature vector consists of the pose of the object (Cervera and Martinet 1999b). This is genuine 3D information, but the approach differs from classic position-based visual servoing in that the error is not measured in the camera frame, but in a new frame defined in the following.

As a result, further advantages can be obtained, as the complete knowledge of the trajectories of the object in the camera frame, and the camera in the space.

3.1. Task Frames

Let $\mathcal{F}_o$ be the coordinate frame attached to the observed object, and let $\mathcal{F}_c$ and $\mathcal{F}_d$ be the coordinate frames associated to the current and desired camera poses, respectively, as depicted in Figure 1. Let ${}^c\mathbf{T}_o$ and ${}^d\mathbf{T}_o$ be the transformations between the camera and object frames, for the current and desired camera poses, respectively. Let us define a new frame $\mathcal{F}_p$, rigidly attached to the current camera frame, as

$$ {}^c\mathbf{T}_p = {}^d\mathbf{T}_o; \tag{21} $$

that is, the transformation between the current camera frame and the defined one is the same as that of the desired camera frame and the object frame.

The key is to define the error vector in the new frame, instead of defining it in the camera frame as usual. The pose of the object is then

$$ \begin{aligned} {}^p\mathbf{T}_o &= \left({}^c\mathbf{T}_p\right)^{-1} {}^c\mathbf{T}_o \\ &= \left({}^d\mathbf{T}_o\right)^{-1} {}^c\mathbf{T}_o \end{aligned} \tag{22} $$

which can be calculated since ${}^d\mathbf{T}_o$ is given, and ${}^c\mathbf{T}_o$ is reconstructed from the current image and the model of the object.

Let

$$ {}^p\mathbf{T}_o = \begin{pmatrix} {}^p\mathbf{R}_o & {}^p\mathbf{p}_o \\ 0 & 1 \end{pmatrix} \tag{23} $$

be the decomposition of the transformation matrix in its rotation and translation components. Then let us define the feature vector as

$$ \mathbf{s} = \begin{pmatrix} {}^p\mathbf{p}_o \\ \mathbf{u}\theta \end{pmatrix} \tag{24} $$

where $\mathbf{u}$ and θ are the axis and angle of rotation corresponding to matrix ${}^p\mathbf{R}_o$. The error vector is

$$ \mathbf{e} = \mathbf{s} - \mathbf{s}^* \tag{25} $$

where $\mathbf{s}^*$ is the desired feature vector.

The interesting thing about this new frame is that, when the camera achieves the desired pose, frames $\mathcal{F}_p$ and $\mathcal{F}_o$ are coincident, thus $\mathbf{s}^* = \mathbf{0}$. In the next sections, we will demonstrate how this property allows the computation of the exact trajectory of the object in the camera frame, and the trajectory of the camera in the world frame.

3.2. Trajectory of the Object in the Camera Frame

The importance of such a trajectory ${}^c\mathbf{p}_o(t)$ relies on the fact that it determines whether the object remains within the vision field of the camera or not.

As shown in the Appendix,

$$ {}^c\mathbf{p}_o(t) = \left({}^c\mathbf{p}_o(0) - {}^d\mathbf{p}_o\right) \mathrm{e}^{-\lambda t} + {}^d\mathbf{p}_o \tag{26} $$

which is simply a straight-line trajectory from the initial position ${}^c\mathbf{p}_o(0)$ to the desired one ${}^d\mathbf{p}_o$, the exponential factor only affecting the velocity along such a line.

The orientation of the object with respect to the camera frame is also derived, giving

$$ {}^c\mathbf{R}_o(t) = {}^d\mathbf{R}_o\mathbf{R}\left(\mathbf{u}\theta(0)\mathrm{e}^{-\lambda t}\right) \tag{27} $$

which is a composition of the constant rotation ${}^d\mathbf{R}_o$ and the variable rotation around the constant axis $\mathbf{u}$, as given in eq. (60).

As when using 3D point features, the center of the object describes a straight path from its initial position to the desired one, in the camera frame. Since both positions are within the camera field of view, all the intermediate ones are within this field too. However, there is no guarantee that the rest of the points of the object will remain within the field of view, although a partial result about this is presented in the next section.

Theoretical demonstrations of robustness and stability are also hard to obtain due to the iterative nature of pose estimation algorithms.

3.3. Image Point Trajectories

In the proposed framework, image features (points) are not directly controlled. However, since the trajectory of the object in the camera frame is known, it is possible to calculate

the trajectory of each point in the camera frame and even its trajectory in the image plane.

If the angle of rotation θ between the current and desired poses of the object with respect to the camera is small, then we will demonstrate that each object point follows a straight-line trajectory in 3D space from its initial to its final position (see the Appendix). Thus, the projection of each point describes a straight line from its initial to its final position in the image plane too.

The trajectory of a point i, which is rigidly attached to the object, is then

$$ {}^c\mathbf{p}_i(t) \approx {}^d\mathbf{p}_i + \left({}^c\mathbf{p}_i(0) - {}^d\mathbf{p}_i\right) \mathrm{e}^{-\lambda t} \tag{28} $$

which is effectively a straight line from its initial to its final position in the camera frame. Consequently, its projection is a straight line joining its initial and final positions in the image plane.

From the point of view of the image, such behavior is the same as that produced by classic image-based visual servoing, where the task error is defined in terms of image points. In this last approach, image points are constrained to follow straight lines even in the presence of high orientation variations, thus frequently causing inadequate camera motions (Chaumette 1998).

3.4. Advantages of the Proposed Coordinate Frame

The key advantage of the proposed framework is that the trajectory of the object in the camera frame is known, and the object is likely to remain within the camera field (in the absence of high calibration and/or model errors).

However, we might wonder about the need to introduce a new coordinate frame $\mathcal{F}_p$ instead of using directly the current camera frame $\mathcal{F}_c$. Effectively, if the error vector is defined in $\mathcal{F}_c$, then the trajectory of the object is exactly the same as given in eq. (26).

The difference between choosing either of the two frames concerns only the orientation of the object. If the error vector is defined in the camera frame, then $\mathbf{s}^*$ is no longer null, and

$$ \mathbf{e} = \begin{pmatrix} {}^c\mathbf{p}_o - {}^d\mathbf{p}_o \\ \mathbf{u}_c\theta_c - \mathbf{u}_d\theta_d \end{pmatrix} \tag{29} $$

where $\mathbf{u}_c\theta_c$ and $\mathbf{u}_d\theta_d$ are, respectively, the axes and angles which correspond to the rotation matrices ${}^c\mathbf{R}_o(0)$ and ${}^d\mathbf{R}_o$.

Solving the differential equation leads to

$$ \mathbf{u}\theta(t) = \mathbf{u}_c\theta_c \mathrm{e}^{-\lambda t} + (1 - \mathrm{e}^{-\lambda t})\mathbf{u}_d\theta_d. \tag{30} $$

The problem now is that, in the general case, the orientation of the axis of rotation changes, since it is a linear combination of the two not-necessarily-parallel vectors, $\mathbf{u}_c$ and $\mathbf{u}_d$. Only if these vectors are parallel or either of them is null, then the axis of rotation is constant.

This is clearly a disadvantage of the camera frame, which motivates the introduction of frame $\mathcal{F}_p$. A constant rotation axis not only produces simpler trajectories, but it also allows the determination of the trajectories of object points, as described before, and an easier calculation of the velocity screw, as shown in the next section.

3.5. Velocity Screw

The velocity screw in the camera frame, using the task function approach (Samson, Le Borgne, and Espiau 1991)—see the Appendix—is

$$ {}^c\mathbf{v} = -\lambda \begin{pmatrix} -{}^c\mathbf{p}_o + {}^d\mathbf{p}_o - \left[{}^c\mathbf{p}_o\right]_\times {}^d\mathbf{R}_o\mathbf{u}\theta \\ -{}^d\mathbf{R}_o\mathbf{u}\theta \end{pmatrix} \tag{31} $$

where $\mathbf{u}$ and θ are the axis and angle which correspond to the rotation matrix

$$ \mathbf{R}(\mathbf{u}\theta) = ({}^d\mathbf{R}_o)^{-1}{}^c\mathbf{R}_o; \tag{32} $$

that is, the pure rotation between the initial and desired orientations of the object with respect to the camera.

The computation of the control law consists of three steps, as follows.

1. Obtain ${}^c\mathbf{p}_o$ and ${}^c\mathbf{R}_o$ from the current image and the model of the observed object with a reconstruction algorithm; e.g., a fast algorithm is proposed in Dementhon and Davis (1995).
2. Calculate $\mathbf{u}\theta$ as shown in eq. (32).
3. Compute the velocity screw in the camera frame ${}^c\mathbf{v}$ as given by eq. (31). This screw can be readily transformed to the end-effector or base frame and, via the Jacobian robot, to joint velocities.

As explained in the literature (Samson, Le Borgne, and Espiau 1991; Chaumette 1998), the system is globally asymptotically stable if $\widehat{\mathbf{L}}_s^{-1}\mathbf{L}_s > 0$ where $\widehat{\mathbf{L}}_s^{-1}$ is an estimation based on the current measurements. This condition is surely met since, *in the absence of calibration errors*, $\widehat{\mathbf{L}}_s^{-1}\mathbf{L}_s = \mathbf{I}_6$.

3.6. Trajectory of the Camera

Since the object is supposed static, then

$$ \mathbf{p}_c(t) = \mathbf{R}_o{}^o\mathbf{p}_c(t) + \mathbf{p}_o \tag{33} $$

where $\mathbf{p}_o$ and $\mathbf{R}_o$ are, respectively, the position and orientation of the object with respect to an absolute frame.

It follows immediately that

$$ \mathbf{p}_c(t) = -\mathbf{R}_o{}^o\mathbf{R}_c(t){}^c\mathbf{p}_o(t) + \mathbf{p}_o \tag{34} $$

which allows us to calculate the trajectory of the camera in terms of the trajectory of the object in the camera frame and the pose of the object on the absolute frame.

Since the trajectory of the object in the camera frame is described by eqs. (26) and (27), the trajectory of the camera can be completely expressed in terms of the task parameters $({}^c\mathbf{T}_o(0), {}^d\mathbf{T}_o, \mathbf{T}_o)$

$$\begin{aligned} \mathbf{p}_c(t) &= -\mathbf{R}_c(t)\left(\left({}^c\mathbf{p}_o(0) - {}^d\mathbf{p}_o\right)\mathrm{e}^{-\lambda t} + {}^d\mathbf{p}_o\right) + \mathbf{p}_o \\ \mathbf{R}_c(t) &= \mathbf{R}_o\mathbf{R}(\mathbf{u}\theta\mathrm{e}^{-\lambda t})({}^d\mathbf{R}_o)^{-1} \end{aligned} \tag{35}$$

where $\mathbf{u}$ and θ are the axis and the angle which correspond to the rotation matrix $({}^c\mathbf{R}_o(0))^{-1}{}^d\mathbf{R}_o$.

4. Experimental Results

The presented approaches have been tested on real robotic platforms. Eye-in-hand configurations have been used, both in mono and stereo cases. The monocular platform is a Cartesian manipulator, and the stereo one is a Mitsubishi PA-10 arm, although controlled in the Cartesian space.

4.1. Pixels and Depth

This scheme has been implemented on a robotic platform with six degrees of freedom and an eye-in-hand configuration. 2D visual features are extracted at video rate (25 Hz) and 3D features at 12.5 Hz. The target object is composed of four points which define a tetrahedron. 3D coordinates are obtained from the pose of the object, which is extracted from the images and an internal model with the algorithm of Dementhon and Davis (1995).

Initial and desired poses of the camera are shown in Table 1. As explained in Malis, Chaumette, and Boudet (1999), this is a difficult task since the displacement is important and the object moves towards the border of the image.

Figure 2 depicts the trajectories of the object in the image plane, when three different features are used in the control loop: pure 2D points, 2D points combined with depth, and pure 3D points. Since the involved rotation is high, the points of the object do not follow straight lines from the initial to the final positions. Although trajectories are more curved in the proposed approaches than in image-based visual servoing, the object remains in the field of view of the camera during the complete servoing task.

Camera trajectories in 3D space are depicted in Figure 3. The trajectory of the camera when using depth or 3D points is closer to a straight path than when using the classic image-based scheme. As a consequence, the manipulator is better kept from reaching its joint limits.

Table 1. Initial and Desired Poses of the Camera

Pose	Translation (mm)	Rotation $\mathbf{u}\theta$ (deg)
Initial	0 0 −500	0 0 0
Desired	−225 249 −408	7 37 −70

4.2. Pixels and Stereo Disparity

The stereo visual servoing platform consists of a Mitsubishi PA-10 arm, controlled in the Cartesian space. Attached to the end-effector of the arm is a stereo rig with two miniature CMOS color cameras, linked to two video boards which deliver the visual features at video rate (30 Hz).

We give the estimation of the parameters (intrinsic and extrinsic) of both cameras, as used in the experiments: F_u is 300; F_v is 450; and b is 118 mm.

The target object consists of four co-planar points located at the vertices of an 11 cm square and the fifth point is located at the center of the square.

The velocity screw is computed from the pseudo-inverse of the interaction matrix (Espiau, Chaumette, and Rives 1992)

$$\mathbf{v} = -\lambda\mathbf{L}^{+}(\mathbf{s} - \mathbf{s}^{*}) \tag{36}$$

with λ set to 0.5 in all the experiments.

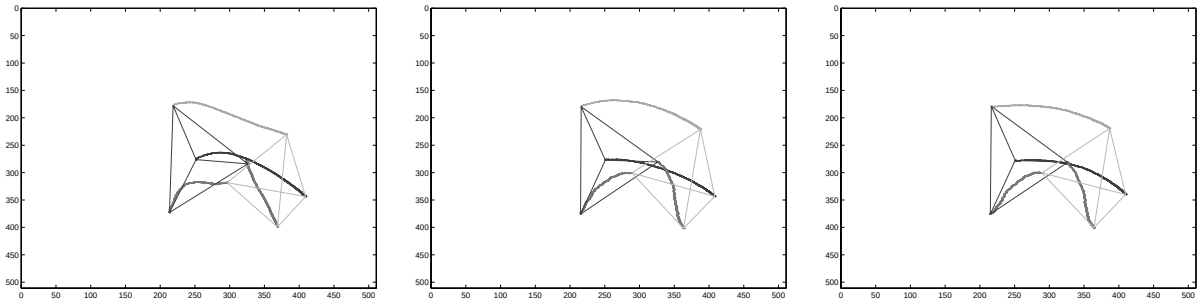
Image measurements are noisy, since the experiments are carried out in a standard office environment, without any special illumination. As a result, there is an almost-uniform noise whose magnitude is ± 1 for u_i^l and u_i^r, and ± 2 for v_i^l and v_i^r. Additionally, pixel coordinates are quantified to a resolution of 200×200.

Experimental results are depicted in Figures 4, 5, and 6. Each figure consists of a set of plots (from top to bottom): the image trajectories of the points, the errors of the visual features, and the velocity screw. Convergence is better when 3D information is used, either stereo disparity, or estimated coordinates.

Figure 7 depicts better this advantage, by showing the 3D trajectory of the end-effector. Both approaches using 3D information in the feature vector accomplish a better trajectory than the pure 2D image-based approach, although using stereo features too.

Convergence to the desired images is always achieved, but quality is worse with the stereo 2D features. As pointed out by Lamiroy et al. (2000b), the stereo interaction matrix is largely overconstrained, and the control data $\mathbf{s}$ and $\mathbf{s}^*$ are redundant. However, this is not sufficient to explain the curvy trajectory of the end-effector (bottom of Figure 7), which almost leads out of the range of robot joints.

Such a trajectory is neither caused by too high a gain; with $\lambda = 0.1$ a smoother but similar trajectory is obtained, as depicted in Figure 8. This problem has not been addressed before since very few experiments with image-based stereo visual servoing have been carried out *with cameras mounted on the end-effector*. To our knowledge, only Maru et al. (1993) have worked with this setup, but their tasks involved rather small



(a) (b) (c)

Fig. 2. Object trajectory in the image: (a) 2D points, (b) 2D/depth, and (c) 3D points.

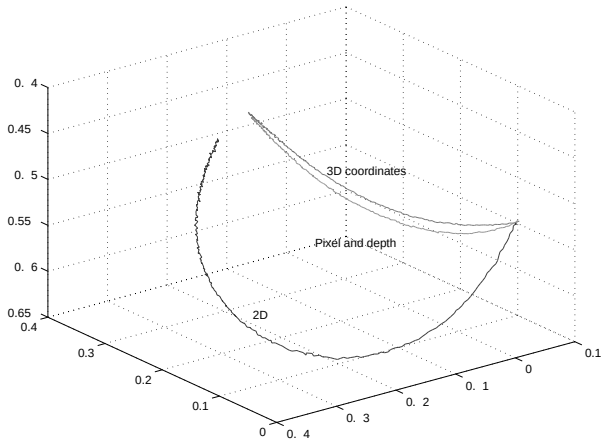


Fig. 3. Trajectories of the camera.

rotations $(\phi, \theta, \psi) = (10, 10, 10)$ (degrees). In our manipulation task, the rotation between the initial and destination poses is $(\phi, \theta, \psi) = (72, 57, 50)$ (degrees). Translational distance is 250 mm, as opposed to 173 mm in Maru et al. (1993).

Approaches based on 3D features work better due to the linearity of the interaction matrix. As shown theoretically, not only the image points but the center of gravity of 3D points translates along a straight line. As a result, the trajectory of the end-effector frame is closer to a straight line too, even with large rotations between frames.

4.3. Object Pose

This scheme has been implemented on an eye-in-hand monocular configuration. The target object is composed of four points which define a tetrahedron. The pose of the object is extracted from the images and an internal model with the algorithm of Dementhon and Davis (1995). Initial and desired poses of the camera are shown in Table 2.

The proposed control law has been experimented together with position-based and image-based ones. However, the classic approaches do not converge and simulated results are provided. The position-based approach failed due to the object going out of the camera field of view (see Figure 9). On the other hand, in both image-based visual servoing and our approach, the object remained always in the camera field of view. However, image-based visual servoing failed due to the imposed camera trajectory, which pushed the robot farther from its physical joint limits.

Table 2. Initial and Desired Poses of the Camera

Pose	Translation	Rotation
Pose	(mm)	$\mathbf{u}\theta$ (deg)
Initial	0 0 −500	0 0 0
Desired	−304 251 −366	−6 26 −50

Figure 10 depicts the trajectory of the camera in 3D space. In position-based visual servoing, the trajectory is ideally a straight line, but convergence is not achieved, as explained before; thus only the result from simulation is shown. In the image-based approach, the trajectory goes farther from the joint limits, thus it fails too. In the figure, the complete

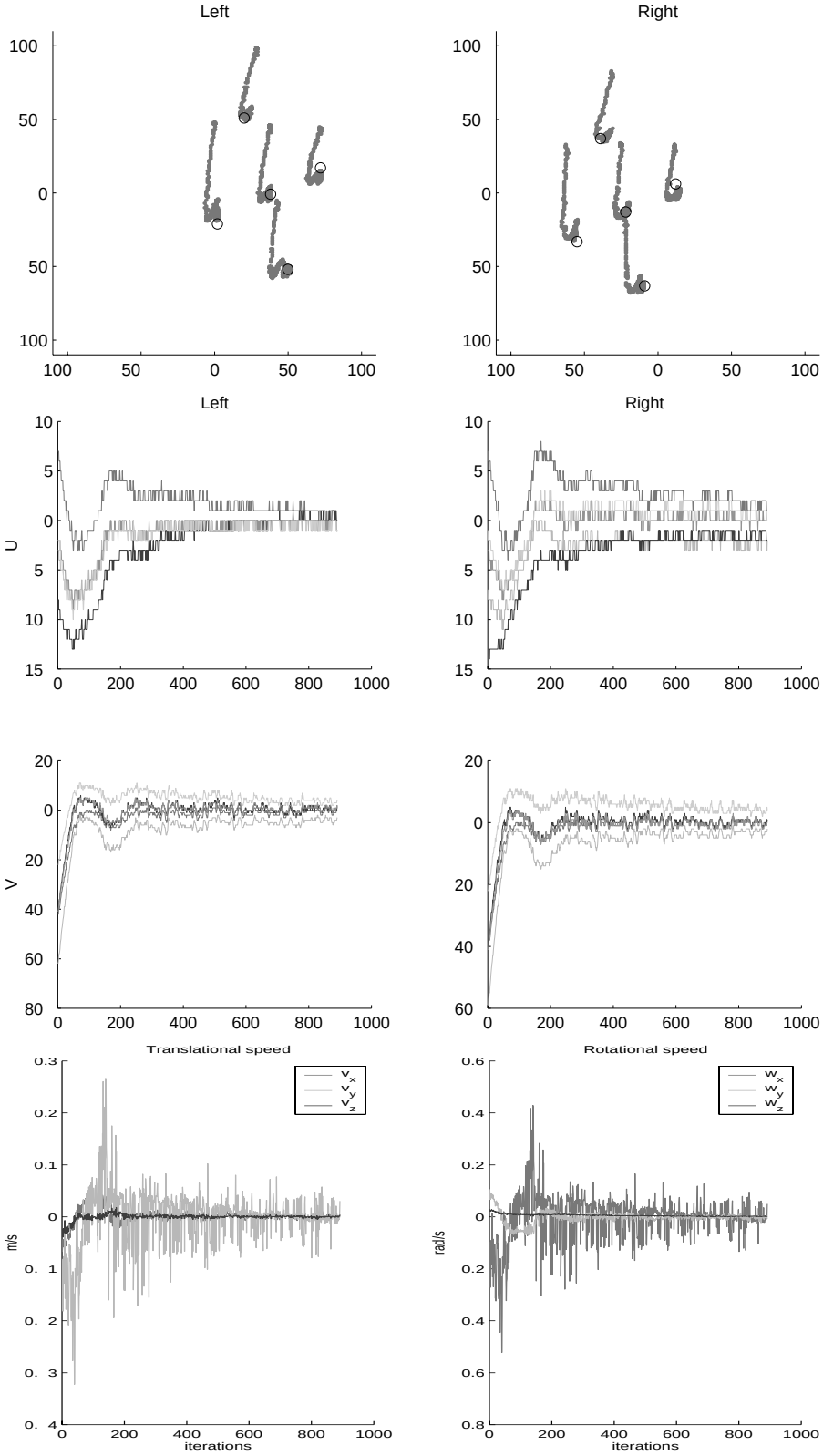


Fig. 4. Stereo 2D points: (from top to bottom) image trajectories, pixel errors (u, v), and velocity screw.

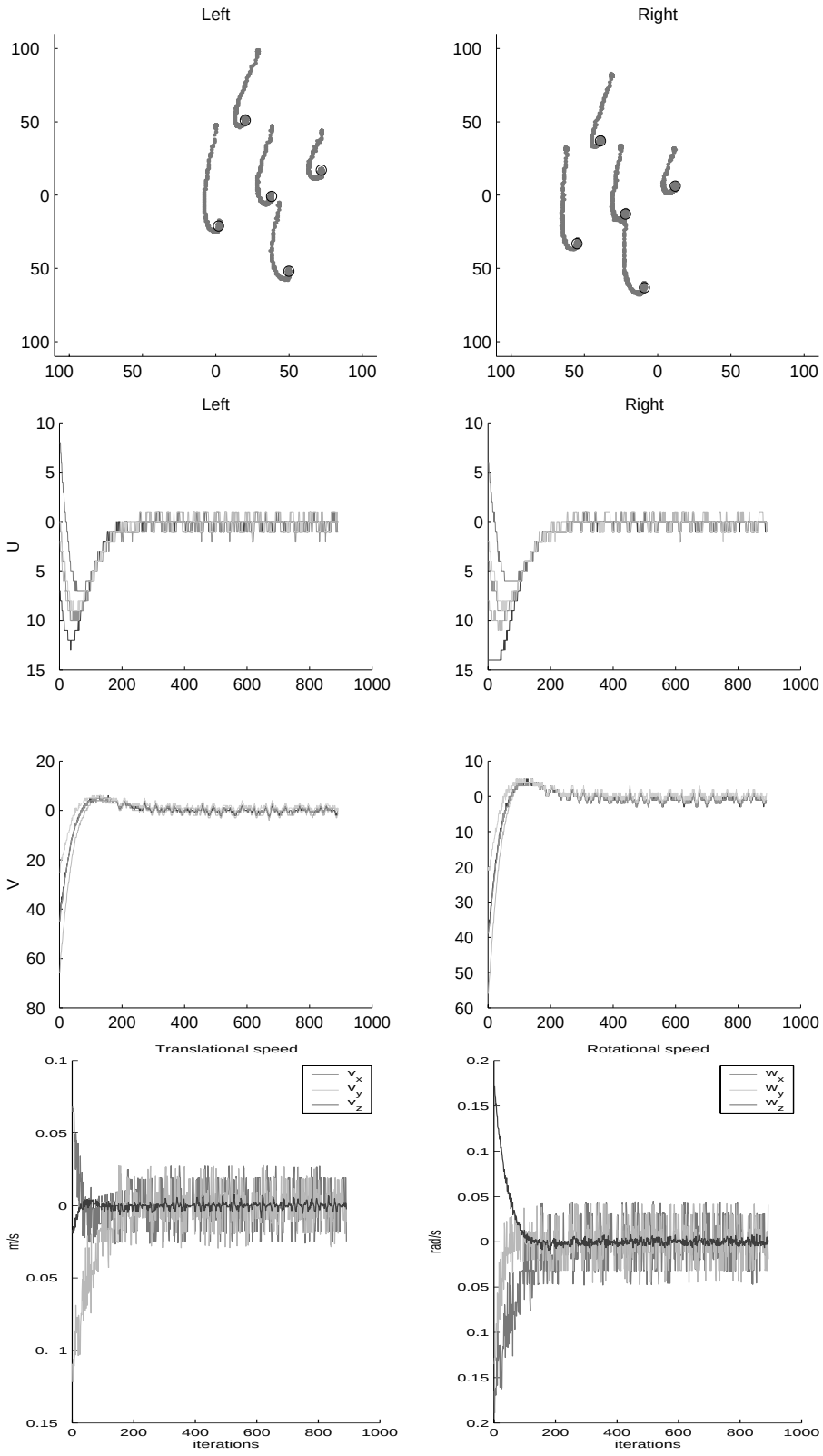


Fig. 5. Estimated 3D points: (from top to bottom) image trajectories, pixel errors (u, v), and velocity screw.

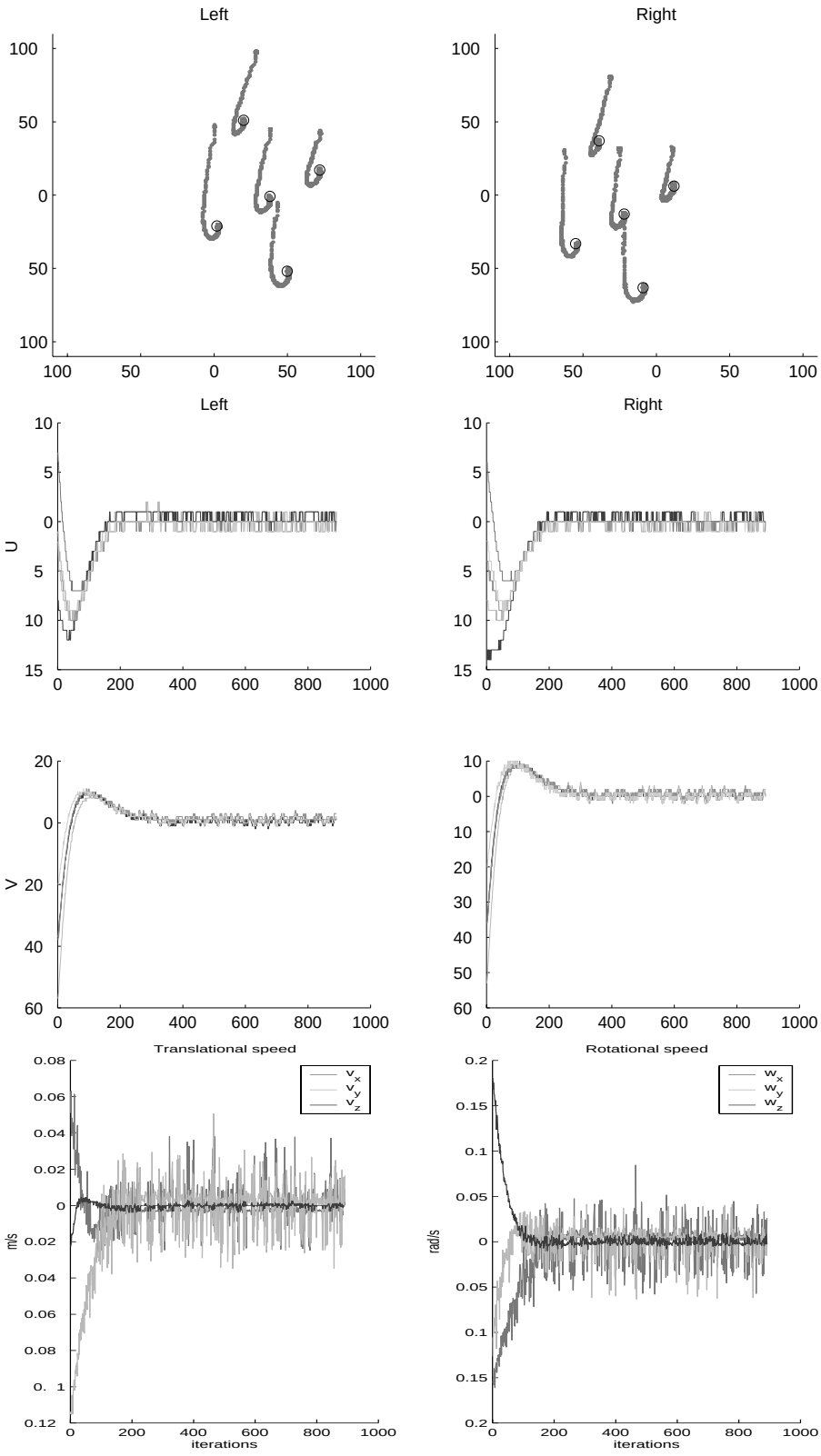
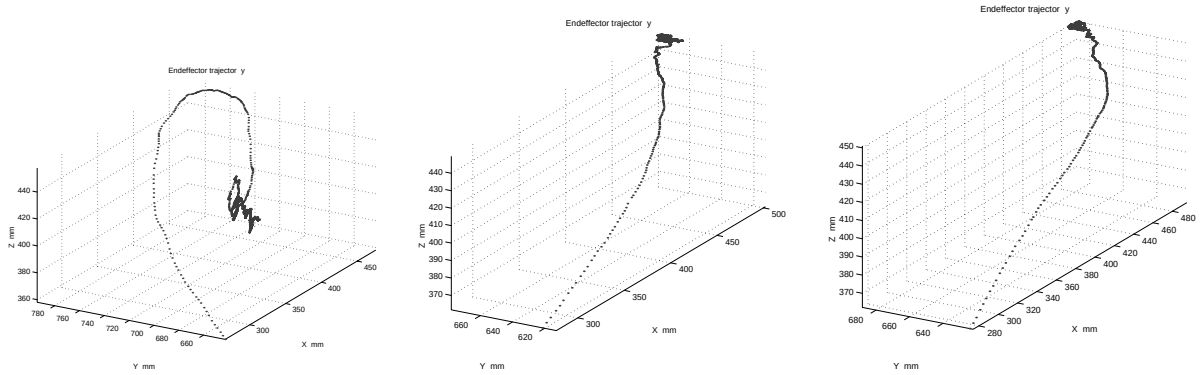


Fig. 6. Stereo 2D and disparity: (from top to bottom) image trajectories, pixel errors (u, v), and velocity screw.



(a) (b) (c)

Fig. 7. End-effector trajectory in 3D space: (a) Stereo 2D points, (b) 3D points, and (c) 2D points/disparity.

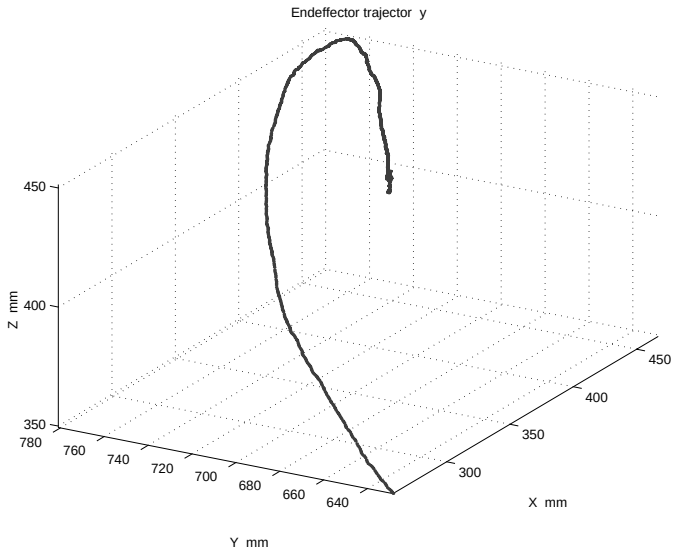
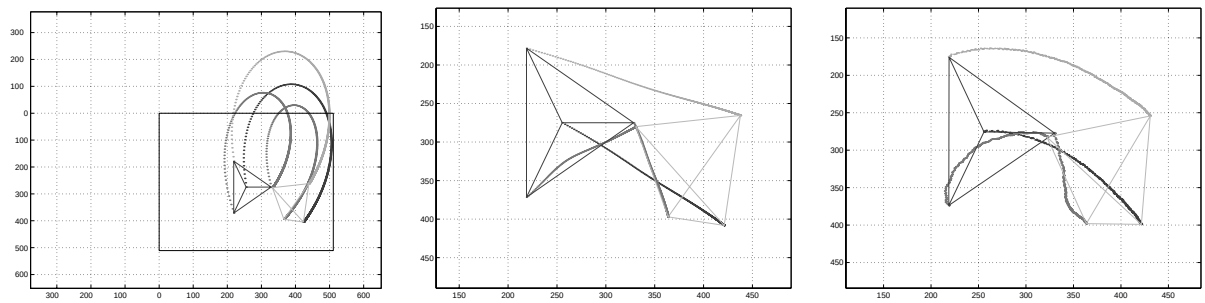


Fig. 8. Trajectory of the end-effector, with stereo 2D features, and $\lambda = 0.1$.



(a) (b) (c)

Fig. 9. Object trajectory in the image: (a) Position-based (simulation), (b) image-based (simulation), and (c) object pose (real).

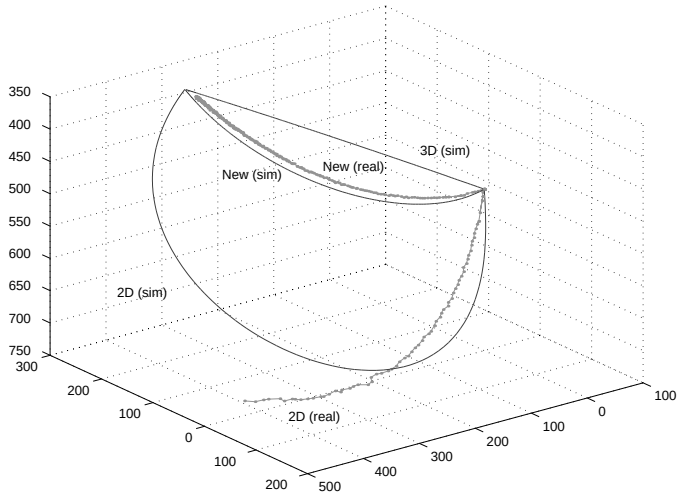


Fig. 10. Trajectories of the camera with 2D image-based (simulation and real—failure), 3D position-based (simulation only) and the proposed visual servoing approaches (simulation and real—successful).

trajectory is obtained from simulation, and the real one stops with the joint range error. Finally, with the proposed model the trajectory is near to the straight path, and convergence to the desired pose is achieved (both in simulation and real experiment).

The differences between real and simulated trajectories are due to the calibration errors of the real platform which cannot be completely captured in the simulations.

5. Summary and Conclusions

The use of 3D features in image-based visual servoing has been proposed. This extension does not require additional information, i.e., only the pixel coordinates and the estimated depth of each point are used, as in classical 2D visual servoing.

Such new features result in a linear control law, from which the computed screw can be obtained, resulting in a motion of the object along a straight path in the camera frame; thus, the object is most likely to remain in the field of view of the camera during the visual servoing task.

The proposed scheme is applied to a stereo visual servoing system, where disparity replaces the point depth in the feature vector. Theoretical results show the similarities between this and the previous approach.

We also propose the use of 3D coordinates of the points of the object, obtained from the image features, the estimated depth and the camera parameters. We demonstrate that it is a particular case of the first proposed scheme. Nevertheless, an exact value of the velocity screw is obtained, provided that an additional geometric property of the target object is met.

As a further step, a visual servoing model using object pose has been presented, which inherits the advantages from position-based and image-based approaches, namely the camera trajectory is predictable and the image features remain in the camera field of view. In addition, convergence is achieved for any configuration of the camera and object provided that the pose can be correctly reconstructed from the object image and model.

The proposed approaches exhibit a better behavior than the classic image-based one, since the trajectories of the camera in the absolute frame are less elongated when the change of orientation is high. Thus, the risk of the robot going out of its joint limits is lowered.

Besides proper camera calibration, the main requirement of the proposed approach is the acquisition of 3D information either by a pose estimation algorithm (assuming a CAD model of the target), or by a stereo visual system. A projective reconstruction is used in $2\frac{1}{2}$D visual servoing, which does not have these requirements, at the expense of being less robust with respect to image measurement errors, and more computationally expensive.

Experimental results on a robotic platform show the feasibility of the proposed scheme in a real-world environment. Future work includes the study of the efficiency and robustness of the proposed schemes when compared to the classic image-based and position-based visual servoing approaches, particularly in face of camera calibration errors. The influence of object geometry needs to be studied, since some control properties depend on its symmetry, although the final target is to find features to work without object models.

Appendix

Pseudo-Inverse of the Interaction Matrix for Image Points and Depth

First, let us calculate the product of the interaction matrix by its transpose

$$\mathbf{L}_s^{\mathrm{t}}\mathbf{L}_s = \begin{pmatrix} \cdots & -\mathbf{A}^{\mathrm{t}} & \cdots \\ & -\left[\mathbf{p}_i\right]_\times \mathbf{A}^{\mathrm{t}} & \end{pmatrix} \begin{pmatrix} \vdots & \\ -\mathbf{A} & \mathbf{A}\left[\mathbf{p}_i\right]_\times \\ \vdots & \end{pmatrix} = \begin{pmatrix} n\mathbf{A}^{\mathrm{t}}\mathbf{A} & -n\mathbf{A}^{\mathrm{t}}\mathbf{A}\left[\mathbf{p}\right]_\times \\ n\left[\mathbf{p}\right]_\times \mathbf{A}^{\mathrm{t}}\mathbf{A} & -n\left[\mathbf{p}\right]_\times \mathbf{A}^{\mathrm{t}}\mathbf{A}\left[\mathbf{p}\right]_\times - \mathbf{S} \end{pmatrix} \quad (37)$$

where

$$\mathbf{S} = \mathbf{R}\mathbf{M}\mathbf{R}^{\mathrm{t}} \quad (38)$$

$$\mathbf{M} = \sum_{i=1}^{n} \left[{}^b\mathbf{p}_i\right]_\times \mathbf{R}^{\mathrm{t}}\mathbf{A}^{\mathrm{t}}\mathbf{A}\mathbf{R}\left[{}^b\mathbf{p}_i\right]_\times . \quad (39)$$

The inverse matrix of the above product is

$$\left(\mathbf{L}_s^{\mathrm{t}}\mathbf{L}_s\right)^{-1} = \begin{pmatrix} \frac{1}{n}(\mathbf{A}^{\mathrm{t}}\mathbf{A})^{-1} + \left[\mathbf{p}\right]_\times \mathbf{S}^{-1}\left[\mathbf{p}\right]_\times & -\left[\mathbf{p}\right]_\times \mathbf{S}^{-1} \\ \mathbf{S}^{-1}\left[\mathbf{p}\right]_\times & -\mathbf{S}^{-1} \end{pmatrix} \quad (40)$$

which exists as long as $\mathbf{M}$ has an inverse, since $\mathbf{S}^{-1} = \mathbf{R}\mathbf{M}^{-1}\mathbf{R}^{\mathrm{t}}$.

Finally, the pseudo-inverse matrix is

$$\mathbf{L}_s^{+} = \begin{pmatrix} \cdots & -\frac{1}{n}(\mathbf{A}^{\mathrm{t}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{t}} + \left[\mathbf{p}\right]_\times \mathbf{T}_i & \cdots \\ & \mathbf{T}_i & \end{pmatrix} \quad (41)$$

where

$$\mathbf{T}_i = \mathbf{R}\mathbf{M}^{-1}\left[{}^b\mathbf{p}_i\right]_\times \mathbf{R}^{\mathrm{t}}\mathbf{A}^{\mathrm{t}}. \quad (42)$$

Velocity Screw for Image Points and Depth

Prior to the calculation of the velocity screw, we must define the error vector $\mathbf{e}$ between the current feature vector s and the desired one $\mathbf{s}^*$:

$$\mathbf{e} = \mathbf{s} - \mathbf{s}^* = \begin{pmatrix} \vdots \\ \mathbf{A}\left(\mathbf{p}_i - \mathbf{p}_i^*\right) \\ \vdots \end{pmatrix} = \begin{pmatrix} \vdots \\ \mathbf{A}\left((\mathbf{p} - \mathbf{p}^*) + (\mathbf{R} - \mathbf{R}^*){}^b\mathbf{p}_i\right) \\ \vdots \end{pmatrix}. \quad (43)$$

The velocity screw is then calculated by means of the task-function approach (Samson, Le Borgne, and Espiau 1991; Espiau, Chaumette, and Rives 1992):

$$\mathbf{v} = -\lambda \mathbf{L}_s^{+}\mathbf{e}. \quad (44)$$

From the above definitions,

$$\mathbf{v} = -\lambda \begin{pmatrix} -\frac{1}{n}\sum_{i=1}^{n}(\mathbf{p} - \mathbf{p}^*) \\ \mathbf{0} \end{pmatrix} - \lambda \begin{pmatrix} \left[\mathbf{p}\right]_\times \sum_{i=1}^{n} \mathbf{T}_i \mathbf{A}(\mathbf{R} - \mathbf{R}^*){}^b\mathbf{p}_i \\ \sum_{i=1}^{n} \mathbf{T}_i\, mA(\mathbf{R} - \mathbf{R}^*){}^b\mathbf{p}_i \end{pmatrix} = -\lambda \begin{pmatrix} (\mathbf{p}^* - \mathbf{p}) + \left[\mathbf{p}\right]_\times \mathbf{R}\mathbf{W} \\ \mathbf{R}\mathbf{W} \end{pmatrix} \quad (45)$$

where

$$\mathbf{W} = \sum_{i=1}^{n} \mathbf{M}^{-1}\left[{}^b\mathbf{p}_i\right]_\times \mathbf{R}^{\mathrm{t}}\mathbf{A}^{\mathrm{t}}\mathbf{A}(\mathbf{R} - \mathbf{R}^*){}^b\mathbf{p}_i. \quad (46)$$

According to the properties of rotation matrices, the above difference is

$$\mathbf{R} - \mathbf{R}^* = -\mathbf{R}\left[\mathbf{u}\right]_\times^2 (1 - \cos\theta) - \mathbf{R}\left[\mathbf{u}\right]_\times \sin\theta \quad (47)$$

where $\mathbf{u}$ and θ are the axis and the angle which correspond to the rotation matrix product $\mathbf{R}^{\mathrm{t}}\mathbf{R}^*$.

If θ is small, we can take the approximations $\sin\theta \approx \theta$ and $\cos\theta \approx 1$. Then,

$$\mathbf{W} \approx \sum_{i=1}^{n} \mathbf{M}^{-1}\left[{}^b\mathbf{p}_i\right]_\times \mathbf{R}^{\mathrm{t}}\mathbf{A}^{\mathrm{t}}\mathbf{A}\left(-\mathbf{R}\left[\mathbf{u}\right]_\times \theta\right){}^b\mathbf{p}_i \approx \mathbf{M}^{-1}\sum_{i=1}^{n}\left[{}^b\mathbf{p}_i\right]_\times \mathbf{R}^{\mathrm{t}}\mathbf{A}^{\mathrm{t}}\mathbf{A}\mathbf{R}\left[{}^b\mathbf{p}_i\right]_\times \mathbf{u}\theta \quad (48)$$

which, according to the definition of $\mathbf{M}$ in eq. (39), leads to

$$\mathbf{W} \approx \mathbf{u}\theta. \quad (49)$$

Thus, the velocity screw is

$$\mathbf{v} \approx -\lambda \begin{pmatrix} (\mathbf{p}^* - \mathbf{p}) + \left[\mathbf{p}\right]_\times \mathbf{R}\mathbf{u}\theta \\ \mathbf{R}\mathbf{u}\theta \end{pmatrix}. \quad (50)$$

Velocity Screw for 3D Points

The velocity screw of the camera can be approximated in the same way as when using image points and depth. Nonetheless, *an exact computation of the velocity screw for any rotation* can be obtained if $\mathbf{M}$ is a diagonal matrix (some objects for $\mathbf{M}$ as a diagonal matrix are, for example, the tetrahedron, the square and the cube).

From the properties of skew-symmetric matrices,

$$\mathbf{M} = \sum_{i=1}^{n} {}^b\mathbf{p}_i\, {}^b\mathbf{p}_i{}^{\mathrm{t}} - \sum_{i=1}^{n} {}^b\mathbf{p}_i{}^{\mathrm{t}}\, {}^b\mathbf{p}_i \mathbf{I}. \quad (51)$$

Since both $\mathbf{M}$ and the second sum are diagonal, then the first sum is diagonal too, i.e.,

$$\sum_{i=1}^{n} {}^b\mathbf{p}_i\, {}^b\mathbf{p}_i{}^{\mathrm{t}} = \alpha\mathbf{I} \quad (52)$$

where α is a positive scalar.

Then, from eqs. (46), (47) and (52)

$$\begin{aligned}\mathbf{W} &= -\mathbf{M}^{-1}\sum_{i=1}^{n}\left[{}^{b}\mathbf{p}_i\right]_\times \left[\mathbf{u}\right]_\times^2 {}^{b}\mathbf{p}_i(1-\cos\theta) \\ &\quad + \mathbf{M}^{-1}\sum_{i=1}^{n}\left[{}^{b}\mathbf{p}_i\right]_\times^2 \mathbf{u}\sin\theta \\ &= -\mathbf{M}^{-1}\sum_{i=1}^{n}\left[{}^{b}\mathbf{p}_i\right]_\times \mathbf{u}\mathbf{u}^{\mathrm{t}}{}^{b}\mathbf{p}_i(1-\cos\theta) + \mathbf{u}\sin\theta \\ &= \mathbf{M}^{-1}\left[\mathbf{u}\right]_\times \sum_{i=1}^{n} {}^{b}\mathbf{p}_i {}^{b}\mathbf{p}_i^{\mathrm{t}}\mathbf{u}(1-\cos\theta) + \mathbf{u}\sin\theta \\ &= \alpha\mathbf{M}^{-1}\left[\mathbf{u}\right]_\times \mathbf{u}(1-\cos\theta) + \mathbf{u}\sin\theta \\ &= \mathbf{u}\sin\theta. \end{aligned} \tag{53}$$

The velocity screw of the camera is

$$\mathbf{v} = -\lambda\begin{pmatrix} (\mathbf{p}^* - \mathbf{p}) + \left[\mathbf{p}\right]_\times \mathbf{R}\mathbf{u}\sin\theta \\ \mathbf{R}\mathbf{u}\sin\theta \end{pmatrix} \tag{54}$$

which is valid for all $\mathbf{u}\theta$.

Trajectory of the Object for Object Pose

Let us compute the trajectory of the object in the camera frame $({}^{c}\mathbf{p}_o(t), {}^{c}\mathbf{R}_o(t))$. To begin with, we need to know the temporal evolution of the feature vector, defined by the pose of the object in the new frame $\mathcal{F}_p$, as presented in Section 3.1.

In this case, when the camera achieves the desired pose, frames $\mathcal{F}_p$ and $\mathcal{F}_o$ are coincident, thus $\mathbf{s}^* = \mathbf{0}$ and

$$\begin{aligned}\mathbf{e} &= \mathbf{s} \\ &= \begin{pmatrix} {}^{p}\mathbf{p}_o \\ \mathbf{u}\theta \end{pmatrix}. \end{aligned} \tag{55}$$

If an exponential decoupled convergence of the task function is imposed then

$$\dot{\mathbf{e}} = -\lambda\mathbf{e} \tag{56}$$

which is a vectorial differential equation whose solution is

$$\mathbf{e}(t) = \mathbf{e}(0)\mathrm{e}^{-\lambda t}. \tag{57}$$

Following the definition of the error,

$$\begin{aligned} {}^{p}\mathbf{p}_o(t) &= {}^{p}\mathbf{p}_o(0)\mathrm{e}^{-\lambda t} \\ \mathbf{u}\theta(t) &= \mathbf{u}\theta(0)\mathrm{e}^{-\lambda t} \end{aligned} \tag{58}$$

where

$$\begin{aligned} {}^{p}\mathbf{p}_o(0) &= {}^{p}\mathbf{R}_c\,{}^{c}\mathbf{p}_o(0) + {}^{p}\mathbf{p}_c \\ &= ({}^{d}\mathbf{R}_o)^{-1}{}^{c}\mathbf{p}_o(0) + {}^{o}\mathbf{p}_d \\ &= ({}^{d}\mathbf{R}_o)^{-1}{}^{c}\mathbf{p}_o(0) - ({}^{d}\mathbf{R}_o)^{-1}{}^{d}\mathbf{p}_o \\ &= ({}^{d}\mathbf{R}_o)^{-1}\left({}^{c}\mathbf{p}_o(0) - {}^{d}\mathbf{p}_o\right) \end{aligned} \tag{59}$$

and where $\mathbf{u}$ and $\theta(0)$ are the axis and the angle which correspond to the rotation matrix

$$\begin{aligned} {}^{p}\mathbf{R}_o(0) &= ({}^{c}\mathbf{R}_p)^{-1}{}^{c}\mathbf{R}_o(0) \\ &= ({}^{d}\mathbf{R}_o)^{-1}{}^{c}\mathbf{R}_o(0). \end{aligned} \tag{60}$$

Now, let us proceed to compute the pose of the object in the camera frame ${}^{c}\mathbf{p}_o(t)$

$$\begin{aligned} {}^{c}\mathbf{p}_o(t) &= {}^{c}\mathbf{R}_p\,{}^{p}\mathbf{p}_o(t) + {}^{c}\mathbf{p}_p \\ &= {}^{c}\mathbf{R}_p\,{}^{p}\mathbf{p}_o(0)\mathrm{e}^{-\lambda t} + {}^{c}\mathbf{p}_p \end{aligned} \tag{61}$$

where it should be noted that both ${}^{c}\mathbf{R}_p$ and ${}^{c}\mathbf{p}_p$ are constant since the frame $\mathcal{F}_p$ is rigidly attached to the camera frame.

When $t = 0$,

$${}^{p}\mathbf{p}_o(0) = {}^{p}\mathbf{R}_c\left({}^{c}\mathbf{p}_o(0) - {}^{c}\mathbf{p}_p\right). \tag{62}$$

Then,

$$\begin{aligned} {}^{c}\mathbf{p}_o(t) &= {}^{c}\mathbf{R}_p\,{}^{p}\mathbf{R}_c\left({}^{c}\mathbf{p}_o(0) - {}^{c}\mathbf{p}_p\right)\mathrm{e}^{-\lambda t} + {}^{c}\mathbf{p}_p \\ &= \left({}^{c}\mathbf{p}_o(0) - {}^{d}\mathbf{p}_o\right)\mathrm{e}^{-\lambda t} + {}^{d}\mathbf{p}_o \end{aligned} \tag{63}$$

which is simply a straight-line trajectory from the initial position ${}^{c}\mathbf{p}_o(0)$ to the desired one ${}^{d}\mathbf{p}_o$, the exponential factor only affecting the velocity along such a line.

The orientation of the object with respect to the camera frame can also be obtained. From eq. (58)

$${}^{p}\mathbf{R}_o(t) = \mathbf{R}\left(\mathbf{u}\theta(0)\mathrm{e}^{-\lambda t}\right) \tag{64}$$

and, by changing the coordinate frame,

$$\begin{aligned} {}^{c}\mathbf{R}_o(t) &= {}^{c}\mathbf{R}_p\,{}^{p}\mathbf{R}_o(t) \\ &= {}^{d}\mathbf{R}_o\mathbf{R}\left(\mathbf{u}\theta(0)\mathrm{e}^{-\lambda t}\right), \end{aligned} \tag{65}$$

which is a composition of the constant rotation ${}^{d}\mathbf{R}_o$ and the variable rotation around the constant axis $\mathbf{u}$, as given in eq. (60).

Image Point Trajectories for Object Pose

The trajectory of a point i, which is rigidly attached to the object, is

$${}^{c}\mathbf{p}_i(t) = {}^{c}\mathbf{R}_o(t)\,{}^{o}\mathbf{p}_i + {}^{c}\mathbf{p}_o(t) \tag{66}$$

where ${}^{o}\mathbf{p}_i$ is the position of such a point in the object frame. Since the point is rigidly attached, this position is constant.

From eqs. (63) and (65)

$$\begin{aligned} {}^{c}\mathbf{p}_i(t) &= {}^{d}\mathbf{R}_o\mathbf{R}(\mathbf{u}\theta(0)\mathrm{e}^{-\lambda t}){}^{o}\mathbf{p}_i \\ &\quad + ({}^{c}\mathbf{p}_o(0) - {}^{d}\mathbf{p}_o)\mathrm{e}^{-\lambda t} + {}^{d}\mathbf{p}_o. \end{aligned} \tag{67}$$

A rotation matrix with axis $\mathbf{u}$ and angle θ can be expressed as

$$\mathbf{R}(\mathbf{u}\theta) = \mathbf{I}_3 + \left[\mathbf{u}\right]_\times^2(1-\cos\theta) + \left[\mathbf{u}\right]_\times\sin\theta \tag{68}$$

which, if θ is small, leads to

$$\mathbf{R}(\mathbf{u}\theta) \approx \mathbf{I}_3 + [\mathbf{u}]_\times \theta. \tag{69}$$

Assuming such an approximation, eq. (27) becomes

$${}^c\mathbf{R}_o(t) \approx {}^d\mathbf{R}_o + {}^d\mathbf{R}_o\,[\mathbf{u}]_\times\,\theta(0)\mathrm{e}^{-\lambda t}. \tag{70}$$

Thus, the trajectory of the object point is

$$\begin{aligned} {}^c\mathbf{p}_i(t) &\approx {}^d\mathbf{R}_o\left(\mathbf{I}_3 + [\mathbf{u}]_\times\,\theta(0)\mathrm{e}^{-\lambda t}\right){}^o\mathbf{p}_i \\ &\quad + ({}^c\mathbf{p}_o(0) - {}^d\mathbf{p}_o)\mathrm{e}^{-\lambda t} + {}^d\mathbf{p}_o \\ &\approx {}^d\mathbf{R}_o\,{}^o\mathbf{p}_i + {}^d\mathbf{p}_o \\ &\quad + \left(({}^c\mathbf{R}_o(0) - {}^d\mathbf{R}_o){}^o\mathbf{p}_i + {}^c\mathbf{p}_o(0) - {}^d\mathbf{p}_o\right)\mathrm{e}^{-\lambda t} \\ &\approx {}^d\mathbf{p}_i + \left({}^c\mathbf{p}_i(0) - {}^d\mathbf{p}_i\right)\mathrm{e}^{-\lambda t}. \end{aligned} \tag{71}$$

Velocity Screw for Object Pose

Let us calculate the control law which achieves the desired exponential decoupled convergence, as imposed in eq. (56). Using the task-function approach (Samson, Le Borgne, and Espiau 1991), the velocity screw is

$$\mathbf{v} = -\lambda\mathbf{L}_s^{-1}\mathbf{e} \tag{72}$$

where $\mathbf{L}_s$ is the *interaction matrix* (also known as the *image interaction* matrix) which relates the derivatives of the feature vector and the velocity screw:

$$\dot{\mathbf{s}} = \mathbf{L}_s\mathbf{v}. \tag{73}$$

The interaction matrix for the defined feature vector is

$$\mathbf{L}_s = \begin{pmatrix} -\mathbf{I}_3 & [{}^p\mathbf{p}_o]_\times \\ \mathbf{0} & -\mathbf{L}_\omega \end{pmatrix} \tag{74}$$

where $\mathbf{L}_\omega$ is the submatrix which relates the rotational components:

$$\frac{\mathrm{d}(\mathbf{u}\theta)}{\mathrm{d}t} = -\mathbf{L}_\omega\boldsymbol{\omega}. \tag{75}$$

As presented in Malis, Chaumette, and Boudet (1999), this matrix is

$$\mathbf{L}_\omega = \mathbf{I} - \frac{\theta}{2}[\mathbf{u}]_\times + \left(1 - \frac{\mathrm{sinc}(\theta)}{\mathrm{sinc}^2(\frac{\theta}{2})}\right)[\mathbf{u}]_\times^2 \tag{76}$$

where the function $\mathrm{sinc}(\theta)$, called *sinus cardinal*, is defined as

$$\mathrm{sinc}(\theta) = \frac{\sin\theta}{\theta}. \tag{77}$$

The inverse of the interaction matrix is

$$\mathbf{L}_s^{-1} = \begin{pmatrix} -\mathbf{I}_3 & -[{}^p\mathbf{p}_o]_\times\mathbf{L}_\omega^{-1} \\ \mathbf{0} & -\mathbf{L}_\omega^{-1} \end{pmatrix} \tag{78}$$

where

$$\mathbf{L}_\omega^{-1} = \mathbf{I} + \frac{\theta}{2}\mathrm{sinc}^2\left(\frac{\theta}{2}\right)[\mathbf{u}]_\times + (1 - \mathrm{sinc}(\theta))\,[\mathbf{u}]_\times^2. \tag{79}$$

Originally, the error vector has been defined in frame $\mathcal{F}_p$, thus the velocity screw is calculated in the same coordinate frame:

$$\begin{aligned} {}^p\mathbf{v} &= -\lambda\begin{pmatrix} -\mathbf{I}_3 & -[{}^p\mathbf{p}_o]_\times\mathbf{L}_\omega^{-1} \\ \mathbf{0} & -\mathbf{L}_\omega^{-1} \end{pmatrix}\begin{pmatrix} {}^p\mathbf{p}_o \\ \mathbf{u}\theta \end{pmatrix} \\ &= -\lambda\begin{pmatrix} -{}^p\mathbf{p}_o - [{}^p\mathbf{p}_o]_\times\mathbf{L}_\omega^{-1}\mathbf{u}\theta \\ -\mathbf{L}_\omega^{-1}\mathbf{u}\theta \end{pmatrix}. \end{aligned} \tag{80}$$

Since $\mathbf{u}\mathbf{u} = 0$ and after eq. (79),

$$\mathbf{L}_\omega^{-1}\mathbf{u}\theta = \mathbf{u}\theta \tag{81}$$

and, consequently,

$${}^p\mathbf{v} = -\lambda\begin{pmatrix} -{}^p\mathbf{p}_o - [{}^p\mathbf{p}_o]_\times\mathbf{u}\theta \\ -\mathbf{u}\theta \end{pmatrix}. \tag{82}$$

We should note that the precedent simplification cannot be applied in the camera frame $\mathcal{F}_c$, as indicated in Section 3.4 since, in the general case, $[\mathbf{u}_c]_\times\,\mathbf{u}_d$ is not null.

Let us calculate now the velocity screw in the camera frame

$$\begin{aligned} {}^c\mathbf{v} &= -\lambda\begin{pmatrix} {}^c\mathbf{R}_p\left(-{}^p\mathbf{p}_o - [{}^p\mathbf{p}_o]_\times\mathbf{u}\theta\right) - [{}^c\mathbf{p}_p]_\times\,{}^c\mathbf{R}_p\mathbf{u}\theta \\ -{}^c\mathbf{R}_p\mathbf{u}\theta \end{pmatrix} \\ &= -\lambda\begin{pmatrix} -{}^c\mathbf{p}_o + {}^c\mathbf{p}_p - [{}^c\mathbf{p}_o - {}^c\mathbf{p}_p + {}^c\mathbf{p}_p]_\times\,{}^c\mathbf{R}_p\mathbf{u}\theta \\ -{}^c\mathbf{R}_p\mathbf{u}\theta \end{pmatrix} \\ &= -\lambda\begin{pmatrix} -{}^c\mathbf{p}_o + {}^d\mathbf{p}_o - [{}^c\mathbf{p}_o]_\times\,{}^d\mathbf{R}_o\mathbf{u}\theta \\ -{}^d\mathbf{R}_o\mathbf{u}\theta \end{pmatrix} \end{aligned} \tag{83}$$

where $\mathbf{u}$ and θ are the axis and angle which correspond to the rotation matrix

$$\begin{aligned} \mathbf{R}(\mathbf{u}\theta) &= {}^p\mathbf{R}_c\,{}^c\mathbf{R}_o \\ &= ({}^d\mathbf{R}_o)^{-1}{}^c\mathbf{R}_o. \end{aligned} \tag{84}$$

Acknowledgments

Support for this research is provided in part by the Spanish Ministry of Science and Technology under projects DPI2001-3801 and HF2001-0112, and by the Generalitat Valenciana under project CTIDIA/2002/195. The authors gratefully acknowledge this support.

References

Berry, F., Martinet, P., and Gallice, G. 1997. Trajectory generation by visual servoing. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, 7–11 September, Grenoble, France, Vol. 2, pp. 1065–1071.

Berry, F., Martinet, P., and Gallice, J. 1999. Visual feedback in camera motion generation: Experimental results. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, 17–21 October, Kyongju, Korea, Vol. 1, pp. 513–518.

Berry, F., Martinet, P., and Gallice, J. 2000. Real time visual servoing around a complex object. *IEICE Transactions on Information and Systems*, Special Issue on Machine Vision Applications, E83-D(7):1358–1368.

Cervera, E., and Martinet, P. 1999a. Combining pixel and depth information in image-based visual servoing. In *Proceedings of the International Conference on Advanced Robotics*, 25–27 October, Tokyo, Japan, Vol. 1, pp. 445–450.

Cervera, E., and Martinet, P. 1999b. Visual servoing with indirect image control and a predictable camera trajectory. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, 17–21 October, Kyongju, Korea, Vol. 1, pp. 381–386.

Cervera, E., Berry, F., and Martinet, P. 2002. Image-based stereo visual servoing: 2D vs 3D features. In *Proceedings of the International Conference on Robotics and Automation*, 11–15 May, Washington, DC, pp. 456–461.

Chaumette, F. 1998. Potential problems of stability and convergence in image-based and position-based visual servoing. In G. Hager, D. Kriegman, and A. Morse, eds., *The Confluence of Vision and Control*. Springer Verlag, Berlin.

Chaumette, F., and Malis, E. 2000. 2 1/2 D visual servoing: a possible solution to improve image-based and position-based visual servoings. In *Proceedings of the IEEE International Conference on Robotics and Automation*, April, San Francisco, CA, Vol. 1, pp. 630–635.

Corke, P., and Hutchinson, S. 2000. A new hybrid image-based visual servo control scheme. In *Proceedings of the IEEE International Conference on Decision and Control*, December, Sidney, Australia, Vol. 3, pp. 2521–2526.

Dementhon, D. F., and Davis, L. S. 1995. Model-based object pose in 25 lines of code. *International Journal of Computer Vision* 15(1/2):123–141.

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

Faugeras, O. 1993. *Three-Dimensional Computer Vision: A Geometric Viewpoint*. MIT Press, Cambridge, MA.

Grosso, E., Meta, M., Andrea, A., and Sandini, G. 1996. Robust visual servoing in 3-d reaching tasks. *IEEE Transactions on Robotics and Automation* 12(5):732–742.

Hashimoto, K., and Noritsugu, T. 2000. Potential problems and switching control for visual servoing. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, November, Takamatsu, Japan, Vol. 1, pp. 423–428.

Hutchinson, S., Hager, G. D., and Corke, P. I. 1996. A tutorial on visual servo control. *IEEE Transactions on Robotics and Automation* 12(5):651–670.

Janabi-Sharifi, F., and Wilson, W. J. 1997. Automatic selection of image features for visual servoing. *IEEE Journal of Robotics and Automation* 5(3):404–417.

Lamiroy, B., Puget, C., and Horaud, R. 2000a. What metric stereo can do for visual servoing. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1–3 November, Takamatsu, Japan.

Lamiroy, B., Espiau, B., Andreff, N., and Horaud, R. 2000b. Controlling robots with two cameras: How to do it properly. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 24–28 April, San Francisco, CA, pp. 2100–2105.

Malis, E., Chaumette, F., and Boudet, S. 1999. 2 1/2 D visual servoing. *IEEE Transactions on Robotics and Automation* 15(2):238–250.

Marchand, E., Chaumette, F., and Rizzo, A. 1996. Using the task function approach to avoid joint limits and kinematics singularities in visual servoing. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, November, Osaka, Japan, Vol. 3, pp. 1083–1090.

Marchand, E., Bouthemy, P., Chaumette, F., and Moreau, V. 1999. Robust real-time visual tracking using a 2D–3D model-based approach. In *Proceedings of the IEEE International Conference on Computer Vision*, September, Kerkira, Greece, Vol. 1, pp. 262–268.

Martinet, P., and Gallice, J. 1999. Position-based visual servoing using a nonlinear approach. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, October, Kyongju, Korea, Vol. 1, pp. 531–536.

Martinet, P., Gallice, J., and Khadraoui, D. 1996. Vision-based control law using 3D visual features. In *World Automation Congress, WAC'96, Robotics and Manufacturing Systems*, May, Montpellier, France, pp. 497–502.

Maru, N., Kase, H., Yamada, S., Nishikawa, A., and Miyazaki, F. 1993. Manipulator control by visual servoing with stereo vision. In *Proceedings IROS'93*, Yokohama, Japan, pp. 1866–1870.

Maruyama, A., Fujita, M., and Kanitani, K. 1999. Full 3D visual feedback control of robotic manipulators: A dissipation theoretical approach. In *Proceedings of the IEEE International Conference on Advanced Robotics*, October, Tokyo, Japan, pp. 439–444.

Mezouar, Y. November 2001. *Planification de trajectoires pour l'asservissement visuel*. Ph.D. Thesis, IRISA/INRIA-Rennes, Rennes, France.

Mezouar, Y., and Chaumette, F. 2001. Design and tracking of desirable image trajectories under mechanics and visibility constraints. In *Proceedings of the IEEE International Conference on Robotics and Automation*, May, Seoul, Korea, Vol. 1, pp. 731–736.

Morel, G., Leibezeit, T., Szewczyk, J., Boudet, S., and Pot, J.

2000. Explicit incorporation of 2D constraints in vision-based control of robot manipulators. In P. Corke and J. Trevelyan, eds., *International Symposium on Experimental Robotics*, Lecture Notes in Control and Information Sciences Vol. 250. Springer Verlag, Berlin, pp 99–108.

Papanikolopoulos, N. P. 1995. Selection of features and evaluation of visual measurements during robotic visual servoing tasks. *Journal of Intelligent and Robotic Systems* 13:279–304.

Samson, C., Le Borgne, M., and Espiau, B. 1991. *Robot Control: the Task Function Approach*, Oxford Engineering Science Series Vol. 22. Clarendon Press, Oxford.

Tarbouriech, S., and Soueres, P. 2000. Advanced control strategy for the visual servoing scheme. In *Proceedings of the IFAC International Symposium on Robot Control*, September, Vienna, Austria, Vol. 2, pp. 457–462.

Thuilot, B., Martinet, P., Cordesses, L., and Gallice, J. 2002. Position-based visual servoing: Keeping the object in the field of vision. In *Proceedings of the IEEE International Conference on Robotics and Automation*, May, Washington, DC, pp. 1624–1629.

Wilson, W. J., Williams Hulls, C. C., and Bell, G. S. 1996. Relative end-effector control using Cartesian position-based visual servoing. *IEEE Transactions on Robotics and Automation* 12(5):684–696.

Zanne, P., Morel, G., and Piestan, F. 2000. Robust vision-based 3D trajectory tracking using sliding model control. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 24–28 April, San Francisco, CA, Vol. 3, pp. 2088–2093.