Proceedings of ICAR 2003
The 11th International Conference on Advanced Robotics
Coimbra, Portugal, June 30 - July 3, 2003

Stereo Visual Servoing with Oriented Blobs

E. Cervera

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

F. Berry, P. Martinet

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

Abstract

In this paper, points and orientations are used in a stereo visual servoing scheme to guide a robot end-effector in a positioning task. Visual features are obtained from segmented images of objects, or blobs. From each blob, its center of gravity and the orientation of its major axis of inertia is computed. Stereo data and camera calibration allows to estimate 3D position and orientation, and to compute their interaction matrices, or Jacobians. The presented approach is validated with real experiments, using a coarsely calibrated stereo rig, in a eye-in-hand configuration, and 2D or 3D features. The target object is a pair of pliers, a real life object, and the presented positioning task is targeted to perform a future grasping task.

1 Introduction

Visual servoing has been traditionally applied to points [7] though the ability to use any geometrical feature has been thoroughly established [4]. Though successful approaches using lines have been reported [5, 6, 1], they are still rare in the literature with respect to those devoted to point features, one possible reason being the difficulty posed by line representations.

The authors have previously used the center of gravity of a color-segmented object [8, 2, 3]. In this paper, an extension of this work is presented using *oriented blobs*: the center of gravity and the orientation of the major axis of inertia of the segmented region.

Stereo vision is used to recover the depth of the features. 3D coordinates of points can be recovered from calibrated stereo images, and 3D orientation of lines is estimated in a similar way.

The paper is organized as follows: Section 2 describes what an oriented blob is, and how its data is first processed, for later computing the interaction matrix (Section 3) associated to either 2D or 3D features. The approach is validated by experimental results in Section 4, and finally there is the conclusion and possible extensions (Section 5). Though image-based visual servoing with points and lines was formalized in [4], the main contribution of this paper is the integration of points and line orientations obtained from blobs, not physical points nor lines.

2 Oriented blobs representation

An object can be segmented from an image by its color, pattern, motion, or, if stereo is used, depth. In any case, what is obtained is a 2D region in the plane, a *blob*, where some statistics can be computed, namely the center of gravity and the major axis of inertia, among others. A blob is *oriented* if this axis can be computed reliably.

In this work, the center of gravity is used as if it were a *physical* point, and the orientation of the inertia axis is used as if it were a *physical* line. One should be cautious, though, about the 3D shape of the object. The approach should work best if the object is symmetric along its axis so the interpretation is valid regardless the point of view.

Let (u, v, ϕ) be the features extracted from the image. Image coordinates can be obtained from pixel coordinates if the camera intrinsic parameters are known (see Fig. 1).

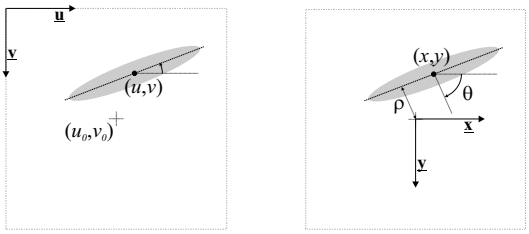

Figure 1: *Relationships between raw image data (left) and image coordinates and line parameters (right) of an oriented blob.*

Thus, image coordinates are computed as:

$$x = \frac{u - u_0}{F_u} \quad (1)$$

$$y = \frac{v - v_0}{F_v} \quad (2)$$

The angle needs to be converted to the correct frame convention:

$$\theta = \frac{\pi}{2} - \phi \quad (3)$$

The line parameter ρ can be readily obtained. Though being redundant and not utilized in the feature vector, it is needed to compute the interaction matrix of θ. It is given by:

$$\rho = x\cos\theta + y\sin\theta \quad (4)$$

The ambiguity of the line representation (ρ, θ) is overcome by fixing the sign of ρ and modulating to 2π the angle difference, as stated in [4]. The process is done in the same way for both the left and right cameras.

3 Computing the interaction matrix

In this section the interaction matrix or Jacobian is computed. First, it is computed for 2D features, namely (u, v, θ), the position and orientation of a blob. Second, the result is presented for 3D features, i.e. the position and orientation of the blob in 3D space estimated from stereo images. Finally, the extension to multiple blobs is shown, and how the interaction matrix is used in the control law is recalled.

3.1 Using 2D features

The first proposed scheme uses a feature vector containing 2D stereo image data, namely

$$\underline{\mathbf{s}} = (u_L, v_L, \theta_L, u_r, v_R, \theta_R)^T$$

However, 3D information is needed to compute the interaction matrix. In the case of the 2D point, its depth is needed. For the 2D line orientation, one of the planes containing the line, not passing through the origin, is needed.

Though 3D information can be recovered from other sources (object models, range data), in this work stereo vision information is used to compute 3D data. Fig. 2 (top of next page) depicts the configuration of a stereo vision system.

In the following, a simplified stereo configuration is assumed: both camera frames are equally oriented (parallel optical axis), focals are equal for both cameras too, b is the distance between their optical centers, and the end-effector frame is located in the middle of the camera frames, with the same orientation. Depth of points is readily obtained from stereo disparity:

$$Z_L = Z_R = \frac{bF_u}{u_L - u_R} \quad (5)$$

The interpretation plane can be recovered from one point and line orientation. However, this plane contains the origin too, thus it cannot be used in the interaction matrix. The solution consists of using the interpretation plane of the left image in the interaction matrix of the right image features, and vice versa. Let us compute the normal vector corresponding to the left image plane:

$$\underline{\mathbf{n}}_L = (x_L, y_L, 1)^T \times (-\sin\theta_L, \cos\theta_L, 0)^T \quad (6)$$

As a result, the coefficients of the interpretation plane π_{1L} equation are:

$$a_L = -\cos\theta_L \quad (7)$$
$$b_L = -\sin\theta_L \quad (8)$$
$$c_L = \rho_L \quad (9)$$

Since the plane contains both the image points and the 3D points, the fourth coefficient of the plane π_{1L}equation can be computed. As mentioned before, these values are used in the interaction matrix of the right image features, thus the substituted 3D point coordinates (Eq. 5) should be located in the right camera frame. There is no frame mismatch since both frames are identically oriented in our setup, thus vectors are invariant in both frames:

$$d_L = -(a_L x_R Z_R + b_L y_R Z_R + c_L Z_R) \quad (10)$$

The interaction matrix for the 2D point and line orientation of the right image is shown in Eq. 11 (next page, below Fig. 2) where

$$\lambda_{\theta_R} = (a_L \sin\theta_R - b_L \frac{F_u}{F_v}\cos\theta_R)/d_L \quad (12)$$

The complete interaction matrix is made of the matrices computed for both left and right image features. These matrices should be properly combined [8], by using the screw transformations from the end-effector to the left and right cameras frame respectively:

$$\mathbf{L} = \begin{pmatrix} \mathbf{L}_L \mathbf{M}_e^L \\ \mathbf{L}_R \mathbf{M}_e^R \end{pmatrix} \quad (13)$$

where the matrices $\mathbf{M}_e^L$ and $\mathbf{M}_e^R$ are the transformations of the screw between the left and right camera frames respectively and the end-effector frame:

$$\mathbf{M}_e^i = \begin{pmatrix} \mathbf{R}_e^i & [\underline{\mathbf{t}}_e^i]_\times \mathbf{R}_e^i \\ \mathbf{O}_3 & \mathbf{R}_e^i \end{pmatrix} \quad (14)$$

3.2 Using 3D features

The depth of a point (Z-coordinate) was computed in (5). The rest of the 3D coordinates can easily be computed, e.g. in the left camera frame:

$$X_L = x_L Z_L \quad (15)$$
$$Y_L = y_L Z_L \quad (16)$$

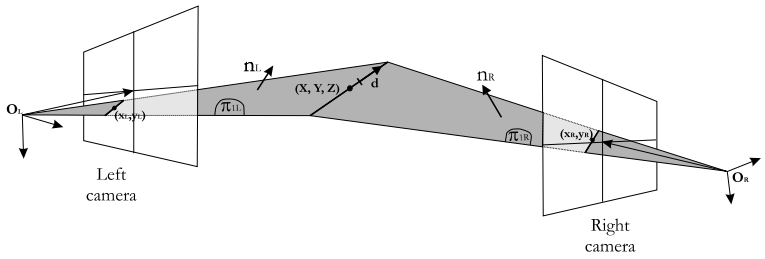

Figure 2: *Configuration of a stereo vision system.*

$$\mathbf{L}_R = \begin{pmatrix} -\frac{F_u}{Z_R} & 0 & \frac{u_R}{Z_R} & \frac{u_R v_R}{F_v} & -F_u - \frac{u_L^2}{F_u} & \frac{v_L F_u}{F_v} \\ 0 & -\frac{F_v}{Z_R} & \frac{v_R}{Z_R} & F_v + \frac{v_L^2}{F_v} & -\frac{u_R v_R}{F_u} & \frac{u_L F_v}{F_u} \\ \lambda_{\theta_R} \cos\theta_R & \frac{\lambda_{\theta_R} F_v \sin\theta_R}{F_u} & -\frac{\lambda_{\theta_R} \rho_R}{F_u} & -\frac{\rho_R \cos\theta_R}{F_v} & -\frac{\rho_R \sin\theta_R}{F_u} & -\frac{F_u^2 \cos^2\theta_R + F_v^2 \sin^2\theta_R}{F_u F_v} \end{pmatrix} \tag{11}$$

where X_L, Y_L and Z_L is the estimated 3D coordinates expressed in left frame (resp. in right frame). These coordinates are written $\underline{\mathbf{p}} = (X, Y, Z)^T$ in the end-effector frame and is directly obtained with the help of the extrinsic camera parameters. This point is one half of the feature vector, the other half being the vector of the 3D line corresponding to the orientation of the object. Such orientation can be recovered without much trouble: since this line is the intersection of both left and right planes, its orientation (let us denote it as $\underline{\mathbf{d}}$) is computed as the normalized cross product of the normal vectors (6) of both planes:

$$\underline{\mathbf{d}} = \frac{\underline{\mathbf{n}}_L \times \underline{\mathbf{n}}_R}{\|\underline{\mathbf{n}}_L \times \underline{\mathbf{n}}_R\|} \tag{17}$$

The feature vector is thus $\underline{\mathbf{s}} = (\underline{\mathbf{p}}^T, \underline{\mathbf{d}}^T)^T$ and its corresponding interaction matrix is:

$$\mathbf{L} = \begin{pmatrix} -\mathbf{I}_3 & [\underline{\mathbf{p}}]_\times \\ \mathbf{O}_3 & [\underline{\mathbf{d}}]_\times \end{pmatrix} \tag{18}$$

3.3 Case of several blobs

One single blob is not sufficient to create a *virtual link*, i.e. to control the 6 d.o.f. of the manipulator. Intuitively, it can be seen that there is a rotation around the line corresponding to the axis of inertia which does not affect the scene.

Several blobs can be easily managed by stacking the interaction matrix corresponding to each blob, to create a single matrix:

$$\mathbf{L} = \begin{pmatrix} \vdots \\ \mathbf{L}_i \\ \vdots \end{pmatrix} \tag{19}$$

where $\mathbf{L}_i$ is computed from stereo images of blob i either by (13) or (18) depending on whether 2D or 3D image features are used.

3.4 Control law

A basic manipulation tasks consists in a motion between two locations. Image features recorded on the destination position ($\underline{\mathbf{s}}^*$) are used to regulate the control loop, [4], with a simple proportional control law:

$$\underline{\mathbf{T}} = -\lambda \mathbf{L}^+ (\underline{\mathbf{s}} - \underline{\mathbf{s}}^*) \tag{20}$$

where $\underline{\mathbf{T}} = (\underline{\mathbf{v}}^T, \underline{\boldsymbol{\omega}}^T)^T$ is the kinematic screw applied to the end-effector and $\mathbf{L}^+$ is the pseudo-inverse of the interaction matrix.

4 Application

In this section, the presented framework is applied to the special configuration of a pliers object. This configuration appears in the image as two segmented blobs, where the center of gravity and the angle of the major axis of inertia are computed, on each image of the stereo pair.

Figure 3 shows the configuration of system. A Mitsubishi PA-10 manipulator holds a stereo rig made of two miniature color cameras, in a eye-in-hand configuration.

The vision system consists of two Cognachrome boards, which deliver color segmentation data (center of gravity, axis of inertia) at 30 Hz. These are off-the-shelf rather inexpensive systems, well suited for visual servoing, which have been used in the authors' previous works [8, 2, 3].

The workplace is depicted in Fig. 4, where the pliers lie on a black floor and the robot is observing the object. Segmented regions are shown in Fig. 5 for both

left and right images. Pliers are orange-colored, thus the blobs corresponding to both arms are segmented based on color information. Blobs are not symmetric along their inertia axes, but the system is expected to be robust against minor deviations.

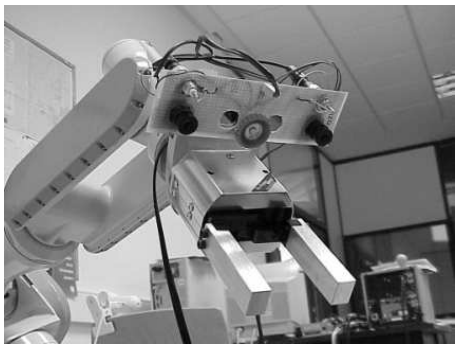

Figure 3: *Robot end-effector, with stereo rig.*

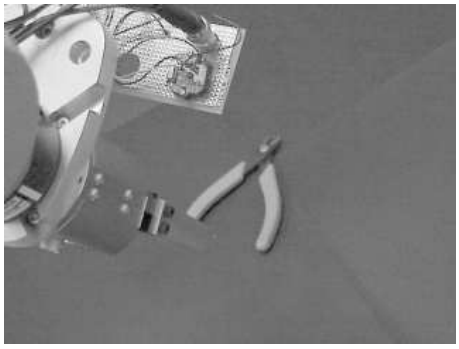

Figure 4: *Workplace containing the target object (pliers) and the end-effector.*

Depth can be recovered from stereo images, as well as the planes containing the lines. In this work a coarse camera calibration is assumed, i.e. the parameters are roughly known, usually up to their nominal values, but no explicit calibration procedure has been undertaken. The target is to test the robustness of visual servoing with the proposed features.

The task consists of moving the manipulator from its initial position to a desired one, e.g. as a prior location to grasping the object. In the following results, the desired position is obtained when the pliers are centered with respect to the end-effector frame, vertically oriented, in a normal plane with respect to z-axis. The rotation between initial and final positions is quite significant, amounting to approximately 45 degrees.

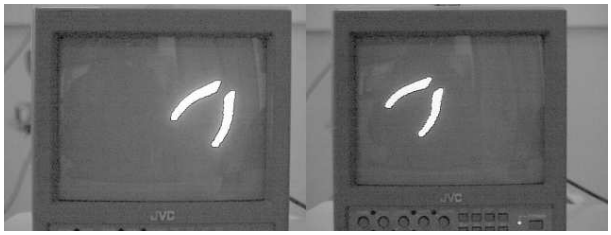

Figure 5: *Segmented images of the pliers, as seen by the left and right cameras.*

4.1 Using 2D features

The feature vector is made of the 2D pixel coordinates and 2D line orientation extracted from each of the two blobs (let them be numbered 1 and 2), on the left and right images. The feature vector $\underline{\mathbf{s}}$ is thus:

$$(u_{1L}, v_{1L}, \theta_{1L}, u_{2L}, v_{2L}, \theta_{2L}, u_{1R}, v_{1R}, \theta_{1R}, u_{2R}, v_{2R}, \theta_{2R})^T \tag{21}$$

The only parameter which remains to be fixed is the gain λ in the control law (20), which was set to 0.1, after several trials. In the experiments, the gain determined heavily the dynamic behavior of the system, and it was kept relatively low to avoid the robot going out of its joint limits.

Figure 6 depicts the task behavior. Top plots show feature errors, for pixel coordinates and line orientations. Linear and angular velocities are plotted in the middle. At the bottom, there is the image trajectory at the left (both images are represented in the same figure) and the trajectory of the end-effector in 3D space at the right.

4.2 Using 3D features

The feature vector consists of estimated 3D point coordinates, and estimated orientations:

$$\underline{\mathbf{s}} = (\underline{\mathbf{p}}_1^T, \underline{\mathbf{d}}_1^T, \underline{\mathbf{p}}_2^T, \underline{\mathbf{d}}_2^T)^T \tag{22}$$

These 3D estimations should be expressed in the end-effector frame, a step easily achieved with the help of the camera extrinsic parameters.

The choice of the gain λ posed even more problems than before. Due to the different nature of point and orientation features, a different gain was chosen for translation and rotation velocities. This is easily achieved replacing the scalar λ in (20) by a diagonal matrix Λ, where the top three diagonal values were set to 0.025, and the rest of the values were set to 0.0025.

Figure 7 depicts the task behavior. Top plots show again the feature errors, which now refer to 3D coordinates of the points and 3D orientation of the lines. Pixel coordinates and line orientations errors follow, just to compare with the 2D approach. Linear and angular velocities are plotted just below, and at the bottom, there is again the image trajectory and the trajectory of the end-effector in 3D space.

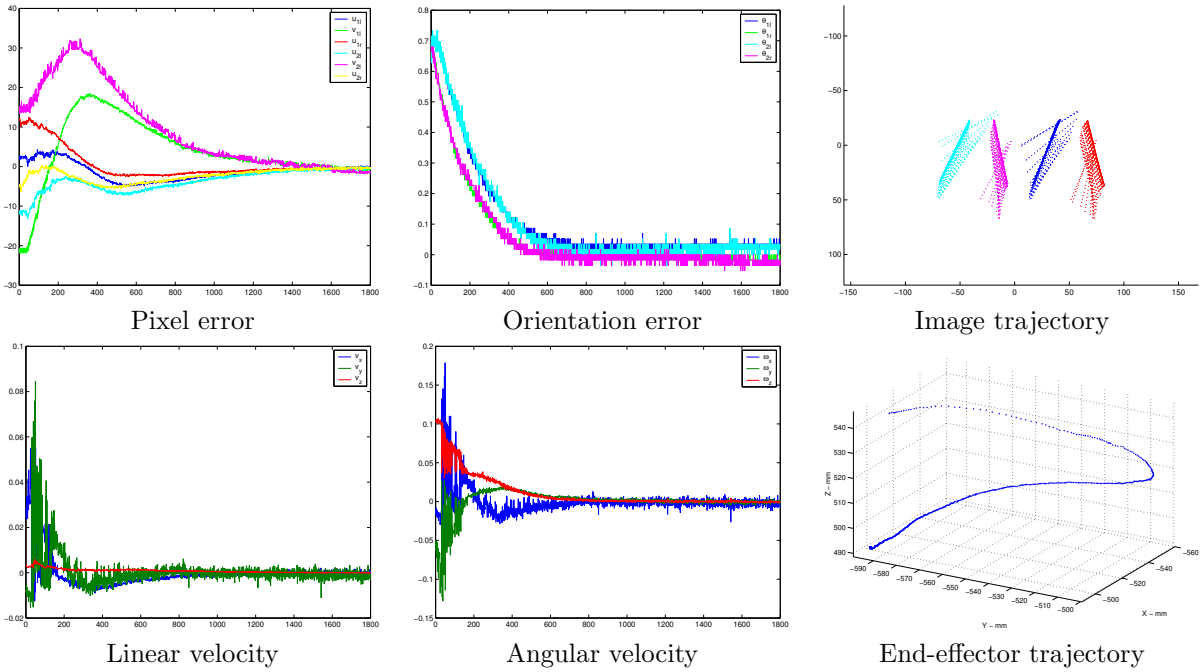

Figure 6: *Task behavior when using 2D control features.*

4.3 Discussion

Image trajectories are better when using 2D features, but this is not surprising since the control loop provides a exponential decrease of feature errors. With 3D features, this decrease does not necessarily correspond to straight image trajectories.

Though convergence is achieved with either 2D or 3D features, a better dynamic behavior appears in the first case. This is due to the coupling between rotation and translation. Though this coupling is present in both cases, its influence seems to be greater in the 3D case. The choice of different gains for linear and angular velocities provided a better behavior, but how to obtain the optimal gain values remains to be solved.

Another drawback of the 3D approach is the noisy estimated 3D values, specially the orientation of the line. Though such noise has a heavy influence in the velocities, the image error (pixel and orientation) at convergence is not much worse than in the 2D case. However, in our implementation the sensor features are noisy. In fact, the spatial discretization of the features provided by the vision system are not precise enough (i.e orientation $\pm 1^o$).

5 Conclusion

A visual servoing approach using one point and the orientation of a blob has been presented. A stereo rig mounted in a eye-in-hand configuration is used. The target object consisted of a pair of pliers which the robot observed in a motion prior to grasping it, and both 2D and 3D features were compared.

Though convegence is achieved, the trajectory of the camera is not as smooth as desired, and control gain must be kept small to ensure convergence. Further experiments are needed to evaluate the approach, e.g. by adding more features since two blobs is the minimum required, but additional ones would probably add robustness to the task.

Experimentals results show the superiority of using 2D features in face of 3D ones. Convergence is smoother, and the control gain is greater, thus providing a faster motion. However, further testing is needed to thoroughly evaluate the influence of noise, calibration and coupling in the trajectory and convergence of the end-effector.

Future extensions include further connection with the grasping phase, though an external camera is needed to keep the target object in the field of view as the end-effector approaches to it. The presented approach is suitable to external cameras too, e.g. mounting the stereo rig on a pan-tilt head.

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.

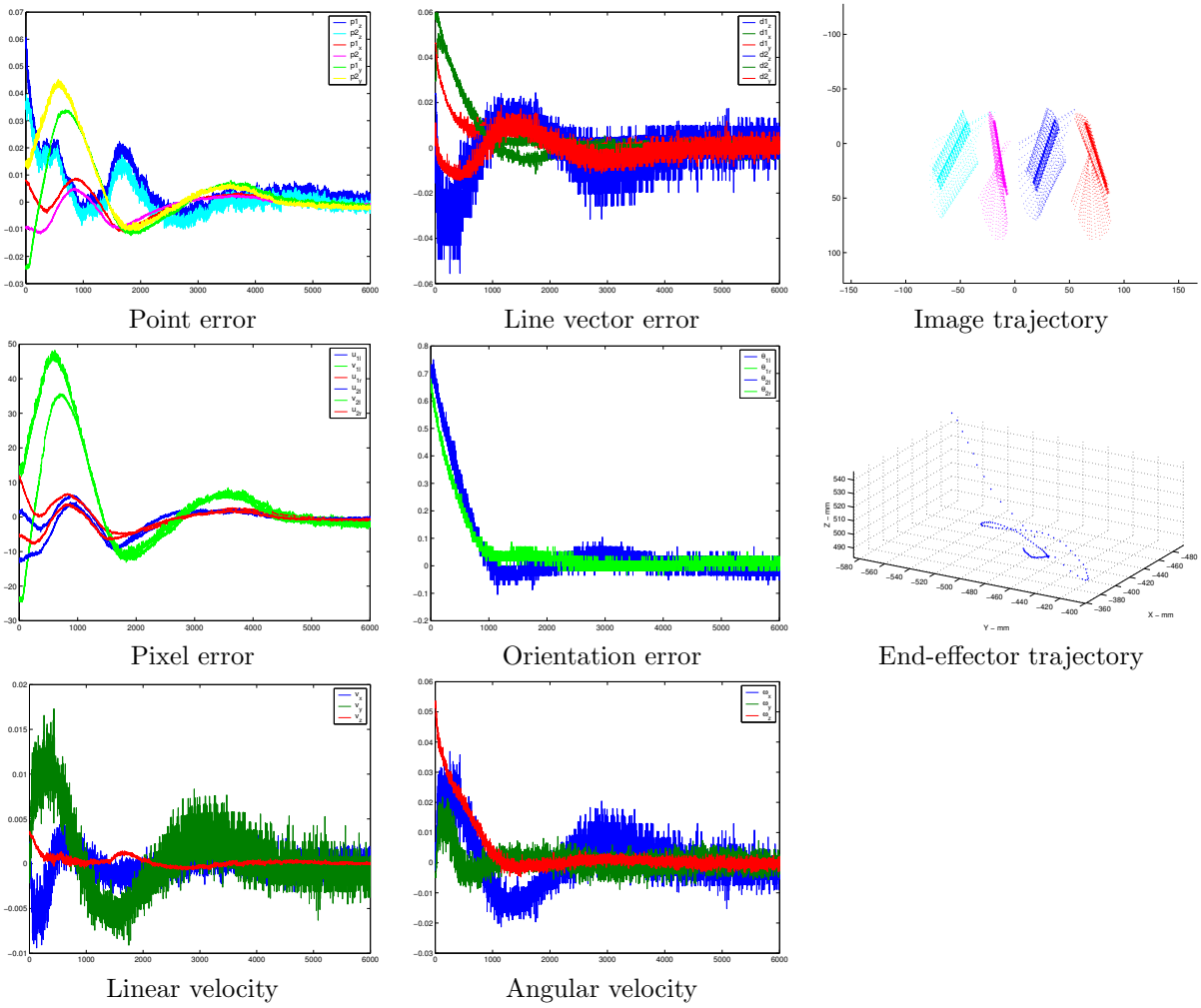

Figure 7: *Task behavior when using 3D control features.*

References

[1] N. Andreff, B. Espiau, and R. Horaud. Visual servoing from lines. In *Proc. IEEE Int. Conf. on Robotics and Automation*, 2000.

[2] E. Cervera, F. Berry, and P. Martinet. Stereo visual servoing with a single point: a comparative study. In *Proc. of the Int. Conf. on Advanced Robotics*, pages 213–218, Budapest, Hungary, 2001.

[3] E. Cervera, F. Berry, and P. Martinet. Is 3D useful in stereo visual servoing? In *Proc. of the Int. Conf. on Robotics and Automation*, page To appear, Washington, USA, 11-15 May 2002.

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

[5] F.Chaumette. Visual servoing using image features defined upon geometrical primitives. In *Proceedings of the 33rd IEEE Conference on Decision and Control*, volume 4, pages 3782–3787, Florida, USA, 1994. CDC'94.

[6] G. Hager. A modular system for robust positioning using feedback from stereo vision. *IEEE Transactions on Robotics and Automation*, 13(4):582–595, August 1997.

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

[8] P. Martinet and E. Cervera. Stacking jacobians properly in stereo visual servoing. In *Proc. of the Int. Conf. on Robotics and Automation*, pages 717–722, Seoul, Korea, 21-26 May 2001.