IMPLEMENTATION OF A HOMOGRAPHY-BASED VISUAL SERVO CONTROL USING A QUATERNION FORMULATION

T. Koenig and G. N. De Souza
ViGIR Laboratory, University of Missouri, 349 Eng. Building West, Columbia-MO, U.S.A.
post@tikey.de, DeSouzaG@missouri.edu

Keywords: Image-base servo, visual servoing, quaternion.

Abstract: In this paper, we present the implementation of a homography-based visual servo controller as introduced in (Hu et al., 2006). In contrast to other visual servo controllers, this formulation uses a quaternion representation of the rotation. By doing so, potential singularities introduced by the rotational matrix representation can be avoided, which is usually a very desirable property in, for example, aerospace applications such as for visual control of satellites, helicopters, etc.
The movement of the camera and the image processing were performed using a simulation of the real environment. This testing environment was developed in Matlab-Simulink and it allowed us to test the controller regardeless of the mechanism in which the camera was moved and the underlying controller that was needed for this movement. The final controller was tested using yet another simulation program provided by Kawasaki Japan for the UX150 industrial robot. The setup for testing and the results of the simulations are presented in this paper.

1 INTRODUCTION

Any control system using visual-sensory feedback loops falls into one of four categories. These categories, or approaches to visual servoing, are derived from choices made regarding two criterias: the coordinate space of the error function, and the hierarchical structure of the control system. These choices will determine whether the system is a *position-based* or an *image-based* system, as well as if it is a *dynamic look-and-move* or a *direct visual servo* (Hutchinson et al., 1996).

For various reasons including simplicity of design, most systems developed to date fall into the position-based, dynamic look-and-move category (DeSouza and Kak, 2004). In this paper however, we describe an image-based, dynamic look-and-move visual servoing system. Another difference between our approach and other more popular choices in the literature is in the use of a quaternion representation, which eliminates the potential singularities introduced by a rotational matrix representation (Hu et al., 2006).

We based the development of our controller on the ideas introduced in (Hu et al., 2006), which requires the assumption that a target object has four coplanar and non-colinear feature points denoted by O_i, where $i = 1 \ldots 4$. The plane defined by those 4 feature points is denoted by Π. Moreover, two coordinate frames must be defined: $\mathcal{F}(t)$ and ${}^*\mathcal{F}$, where $\mathcal{F}(t)$ is affixed to the moving camera and ${}^*\mathcal{F}$ represents the desired position of the camera. Figure 1 depicts the above concepts as well as the vectors ${}^\circ\bar{m}_i(t), {}^*\bar{m}_i \in \mathbb{R}^3$ representing the position of each of the four feature points with respect to the corresponding coordinate frames. That is:

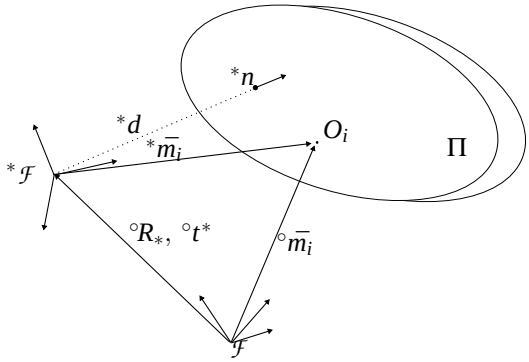

Figure 1: Relationships between the frames and the plane.

$$ {}^\circ\bar{m}_i = [{}^\circ x_i \ \ {}^\circ y_i \ \ {}^\circ z_i]^T \qquad i = 1 \ldots 4 \qquad (1) $$

$$ {}^*\bar{m}_i = [{}^* x_i \ \ {}^* y_i \ \ {}^* z_i]^T \qquad i = 1 \ldots 4 \qquad (2) $$

The relationship between these two sets of vectors can

be expressed as

$$^{\circ}\bar{m_i} = {}^{\circ}t^{*} + {}^{\circ}R_{*}\,{}^{*}\bar{m_i} \tag{3}$$

where $^{\circ}t^{*}(t)$ is the translation between the two frames, and $^{\circ}R_{*}(t)$ is the rotation matrix which brings $^{*}\mathcal{F}$ onto $\mathcal{F}$.

Intuitively, the control objective can be regarded as the task of moving the robot so that $^{\circ}\bar{m_i}(t)$ equals $^{*}\bar{m_i}$ $\forall i$ as $t \to \inf$. However, an image-based visual servoing system is not expected to calculate the real coordinates of these feature points. Instead, it can only extract the image coordinates of those same points. That is, the coordinates $^{\circ}p_i$ and $^{*}p_i$ of the projection of the feature points onto the image plane, given by: $^{\circ}p_i = A\,{}^{\circ}m_i$ and $^{*}p_i = A\,{}^{*}m_i$, where A is the matrix of the intrinsic parameters of the camera. So, the real control objective becomes that of moving the robot so that $^{\circ}p_i$ equals $^{*}p_i$.

This idea will be further detailed in the following section.

2 DESIGN OF THE CONTROLLER

As mentioned above, the control objective is to regulate the camera to a desired position relative to the target object. In order to achieve this control objective the image coordinates at the desired position have to be known. This can be done by taking an image of the target object at the desired position and extracting the feature points using an image processing algorithm. Once a picture is taken and the image coordinates are extracted, those coordinates can be stored for future reference. It is assumed that the motion of the camera is unconstrained and the linear and angular velocities of the camera can be controlled independently. Furthermore the camera has to be calibrated, i. e. the intrinsic parameters of the camera A must be known.

As we mentioned earlier, in the Euclidean space the control objective can be expressed as:

$$^{\circ}R_{*}(t) \to I_3 \qquad \text{as} \qquad t \to \inf \tag{4}$$

$$||\,^{\circ}t^{*}(t)|| \to 0 \qquad \text{as} \qquad t \to \inf \tag{5}$$

and the translation regulation error $e(t) \in \mathbb{R}^3$ can be defined using the extended normalized coordinates as:

$$\begin{aligned} e &= {}^{\circ}m_e - {}^{*}m_e \\ &= \left[\frac{^{\circ}x_i}{^{\circ}z_i} - \frac{^{*}x_i}{^{*}z_i} \;\; \frac{^{\circ}y_i}{^{\circ}z_i} - \frac{^{*}y_i}{^{*}z_i} \;\; \ln\left(\frac{^{\circ}z_i}{^{*}z_i}\right)\right]^{T} \end{aligned} \tag{6}$$

The translation regulation objective can then be quantified as the desire to regulate $e(t)$ in the sense that

$$||e(t)|| \to 0 \qquad \text{as} \qquad t \to \inf\,. \tag{7}$$

It can be easily verified that if (7) is satisfied, the extended normalized coordinates will approach the desired extended normalized coordinates, i. e.

$$^{\circ}m_i(t) \to {}^{*}m_i(t) \text{ and } {}^{\circ}z_i(t) \to {}^{*}z_i(t) \tag{8}$$

as $t \to \inf$. Moreover, if (7) and (8) are satisfied, (5) is also satisfied.

Similarly, the rotation regulation objective in (4) can be expressed in terms of its quaternion vector $q = [q_0 \;\; \tilde{q}\;]^T$, $\tilde{q} = [q_1 \;\; q_2 \;\; q_3]^T$ (Chou and Kamel, 1991) by:

$$||\tilde{q}(t)|| \to 0 \qquad \text{as} \qquad t \to \inf\,. \tag{9}$$

In that case, if (7) and (9) are satisfied, the control objective stated in (4) is also satisfied.

For such translational and rotational control objectives, it was shown in (Hu et al., 2006) that the closed-loop error system is given by:

$$\dot{q_0} = \frac{1}{2}\tilde{q}^T K_\omega \left(I_3 - \tilde{q}^{\times}\right)^{-1}\tilde{q} \tag{10}$$

$$\dot{\tilde{q}} = -\frac{1}{2}K_\omega \left(q_0 I_3 - \tilde{q}^{\times}\right)\left(I_3 - \tilde{q}^{\times}\right)^{-1}\tilde{q} \tag{11}$$

$$^{*}z_i\dot{e} = -K_v e + \tilde{z}_i L_\omega \omega_c \tag{12}$$

and the control inputs by:

$$\omega_c = -K_\omega \left(I_3 - \tilde{q}^{\times}\right)^{-1}\tilde{q} \tag{13}$$

$$v_c = \frac{1}{\alpha_i}L_v^{-1}\left(K_v e + {}^{*}\hat{z}_i L_\omega \omega_c\right) \tag{14}$$

where $^{*}\hat{z}_i = e^T L_\omega \omega_c$ is an estimation for the unknown $^{*}z_i$; $\tilde{q}^{\times}$ is the anti-symmetric matrix representation of the vector $\tilde{q}$; L_v, L_ω are the linear and angular Jacobian-like matrices; $K_\omega, K_v \in \mathcal{R}^{3\times 3}$ are diagonal matrices of positive constant control gains; and the estimation error $\tilde{z}(t) \in \mathcal{R}$ is defined as $\tilde{z}_i = {}^{*}z_i - {}^{*}\hat{z}_i$.

A proof of stability for the controller above can be found in (Hu et al., 2006).

3 IMPLEMENTATION

In this work the task of controlling a robot with a visual-servoing algorithm was divided into four major parts:

1. Capturing images using a video camera.
2. Processing the images to get the coordinates of the feature points.
3. Calculating the input variables, i. e. the velocities of the robot endeffector.
4. Moving the robot according to the given input variables.

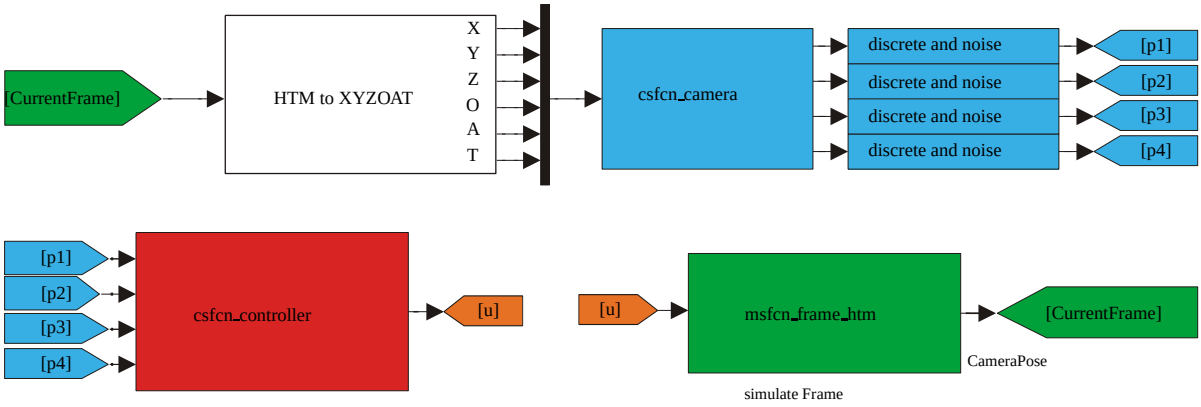

Figure 2: The Simulink-Model.

The hardware available for this task consisted of a Kawasaki industrial robot with a camera mounted on its endeffector and vision-sensor network for capturing and processing images from the camera.

The controller was implemented as a C++ class, in order to guarantee the future reusablility of the code in different scenarios (as it will be better explained below). Initially, the controller requires the camera intrinsic parameters and the desired image coordinates of the four feature points. Next it extracts the current image coordinates and then it calculates the linear and angular velocities of the endeffector, i. e. the input variables of the controller.

In order to safely test the controller, the real pair robot/camera was replaced by two different simulators. The first simulator, which was implemented in MATLABSIMULINK, simulates an arbitrary motion of the camera in space. The camera is represented by a coordinate frame as it is briefly described later. With this simulator it is possible to move the camera according to the exact given velocities.

In all testing scenarios performed, the camera simulator needed to output realistic image coordinates of the feature points. To achieve that, the simulator relied on a very accurate calibration procedure (Hirsh et al., 2001) as well as exact coordinates of the feature points in space with respect to the camera. Given that, the simulator could then return the image coordinates of the feature points at each time instant t. The camera simulator was also implemented as a C++ class.

The second simulator is a program provided by Kawasaki Japan. This simulator can execute the exact same software as the real robot and therefore it allowed for the testing of the code used to move the real robot. This code is responsible for performing the forward and inverse kinematics, as well as the dynamics of the robot.

The basic structure of the simulink model can be seen in Figure 2.

In this work, we will not report the resultis from the tests with the real robot. So, in order to demonstrate the system in a more realistic setting, noise was added to the image processing algorithm and a time discretization of the image acquisition was introduced to simulate the camera.

3.1 Describing the Pose and Velocity of Objects

The position and orientation (pose) of a rigid object in space can be described by the pose of an attached coordinate frame. There are several possible notations to represent the pose of a target coordinate frame with respect to a reference one, including the homogeneous transformation matrix, Euler Angles, etc. (Saeed, 2001) and (Spong and Vidyasagar, 1989). Since we were using the Kawasaki robot and simulator, we adopted the XYZOAT notation as defined by Kawasaki. In that system, the pose of a frame $\mathcal{F}$ with respect to a reference frame ${}^{*}\mathcal{F}$ is described by three translational and three rotational parameters. That is, the cartesian coordinates X, Y, and Z, plus the Orientation, Approach, and Tool angles in the vector form:

$$x = \begin{bmatrix} x & y & z & \phi & \theta & \psi \end{bmatrix}^T$$

This notation is equivalent to the homogeneous transformation matrix:

$$H = \begin{bmatrix} C\phi C\theta C\psi - S\theta S\psi & -C\phi C\theta S\psi - S\phi C\psi & C\phi S\theta & x \\ S\phi C\theta C\psi + C\theta S\psi & -S\phi C\theta S\psi + C\phi C\psi & S\phi S\theta & y \\ -S\theta C\psi & S\theta S\psi & C\theta & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (15)$$

where $S\alpha = \sin(\alpha)$ and $C\alpha = \cos(\alpha)$ and the angles ϕ, θ, and ψ correspond, respectively, to O, A, and T and represent:

- Rotation of ϕ about the $\bar{a}$-axis (z-axis of the moving frame) followed by
- Rotation of θ about the $\bar{o}$-axis (y-axis of the moving frame) followed by
- Rotation of ψ about the $\bar{a}$-axis (z-axis of the moving frame).

4 RESULTS

The controller was tested in three different scenarios: pure linear movement; pure angular movement; and combined movement. In all these cases, the camera intrinsic parameters were:

$$A = \begin{bmatrix} 122.5 & -3.7737 & 100 \\ 0 & 122.6763 & 100 \\ 0 & 0 & 1 \end{bmatrix} \quad (16)$$

The four feature points were arranged in a square around the origin of the reference frame and had coordinates:

$$\begin{aligned} {}^{w}c_1 &= [-5 \;\; -5 \;\; 0]^T \\ {}^{w}c_2 &= [5 \;\; 5 \;\; 0]^T \\ {}^{w}c_3 &= [5 \;\; -5 \;\; 0]^T \\ {}^{w}c_4 &= [-5 \;\; 5 \;\; 0]^T \end{aligned} \quad (17)$$

The desired pose of the camera was the same for all the simulations, only the start poses differ. The desired pose of the camera was 20 units above the target object, exactly in the middle of the four feature points. The camera was facing straight towards the target object, i. e. its z-axis was perpendicular to the xy-plane and pointing out. The x-axis of the camera was antiparallel to the x-axis of the reference frame and the y-axis parallel to the y-axis of the reference frame. This pose can be described by:

$${}^{*}x_w = \begin{bmatrix} 0 & 0 & 20 & 0^\circ & 180^\circ & 0^\circ \end{bmatrix}^T \quad (18)$$

The control gains used for the controller were:

$$K_v = \begin{bmatrix} 25 & 0 & 0 \\ 0 & 25 & 0 \\ 0 & 0 & 25 \end{bmatrix} \quad (19)$$

$$K_\omega = \begin{bmatrix} 1.5 & 0 & 0 \\ 0 & 1.5 & 0 \\ 0 & 0 & 1.5 \end{bmatrix} \quad (20)$$

As mentioned in Section 3, we simulated both the noisy and the discrete aspects of a real camera. That is, we added noise to the image coordinates to simulate a typical accuracy of 0.5 pixels within a random error of 2 pixels in any direction. These values were obtained experimentally using real images and a previously developed feature extraction algorithm.

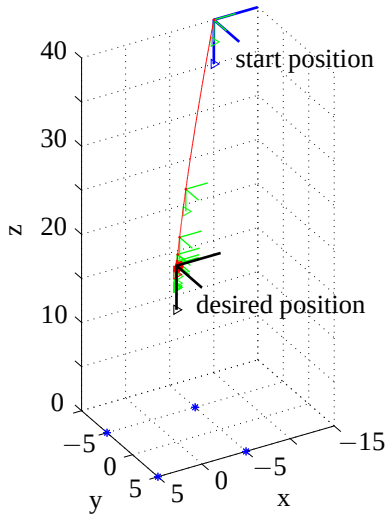

Figure 3: Linear motion simulation in euclidean space.

4.1 Linear Motion

In this simulation the camera did not rotate, i. e. the orientation of the camera in the start pose was the same as in the desired pose. The camera was simply moved 20 units along the z-axis of the reference frame and -10 units along the x- and the y-axis of the reference frame. The start pose was given by

$${}^{\circ}x_w = \begin{bmatrix} -10 & -10 & 40 & 0^\circ & 180^\circ & 0^\circ \end{bmatrix}^T \quad (21)$$

Figure 3 shows the pose of the camera at ten time instants. The z-axis of the camera – the direction in which the camera is "looking" – is marked with a triangle in the figure. The four points on the target object, lying in the xy-plane, are marked with a star. In Figure 4 the image coordinates of the four points are shown. The image coordinate at the start pose is marked with a circle, the image coordinate at the desired pose with a star.

In Figure 5 the control inputs are shown. The first part shows the linear velocities, the second part the angular velocities of the camera.

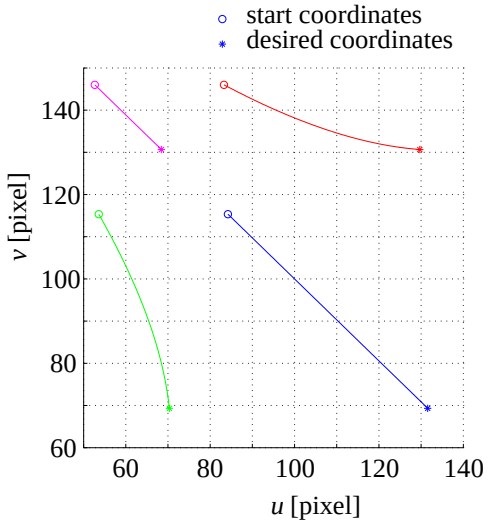

Figure 4: Coordinates of the feature points in image space for the linear motion.

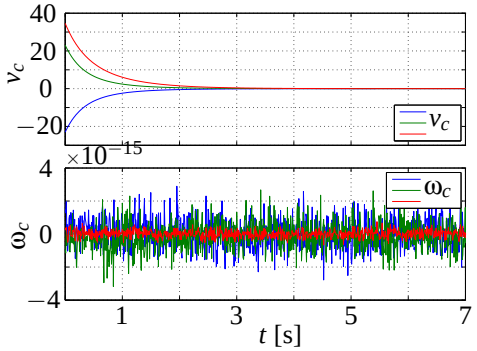

Figure 5: Control input for the linear motion.

4.2 Angular Motion

In this simulation the start position of the camera was identical to the desired position, only the orientation differed. The start pose is given by:

$$^{\circ}x_w = \begin{bmatrix} 0 & 0 & 20 & 45^{\circ} & 150^{\circ} & 5^{\circ} \end{bmatrix}^T \quad (22)$$

As in Section 4.1 the figures 6, 7 and 8 show the movement of the camera in euclidean space, the image coordinates of the four feature points and the control variables.

4.3 Coupled Motion

In this simulation the camera could perform any generic movement, i. e. both the position and the orientation at the beginning differ from the desired pose of the camera. The start pose is given by:

$$^{\circ}x_w = \begin{bmatrix} 10 & -10 & 40 & 90^{\circ} & 140^{\circ} & 10^{\circ} \end{bmatrix}^T \quad (23)$$

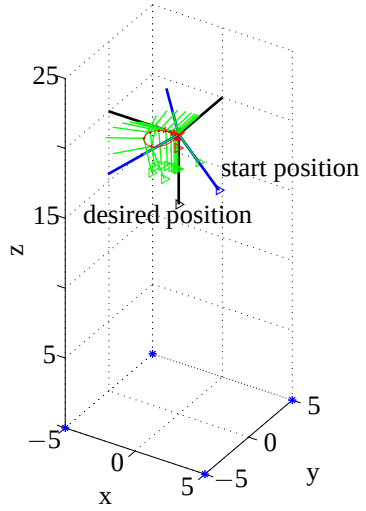

Figure 6: Angular motion simulation in euclidean space.

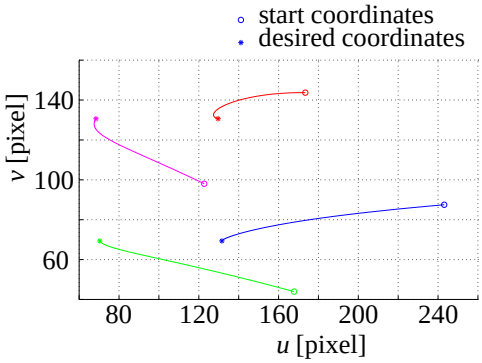

Figure 7: Coordinates of the target points in image space for the angular motion.

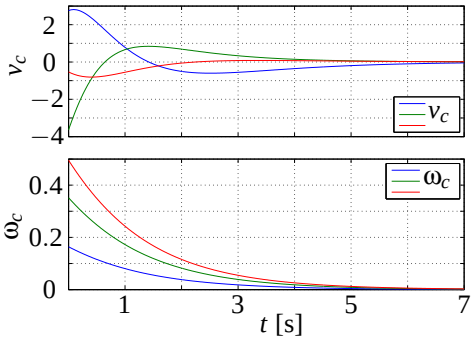

Figure 8: Control input for the angular motion.

As in Section 4.1 the figures 9, 10 and 11 show the movement of the camera in euclidean space, the image coordinates of the four feature points and the control variables.

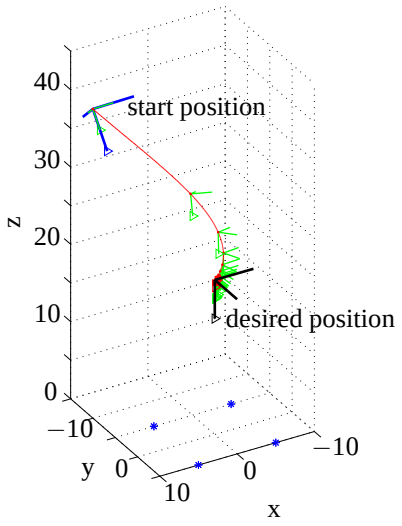

Figure 9: Coupled motion simulation in euclidean space.

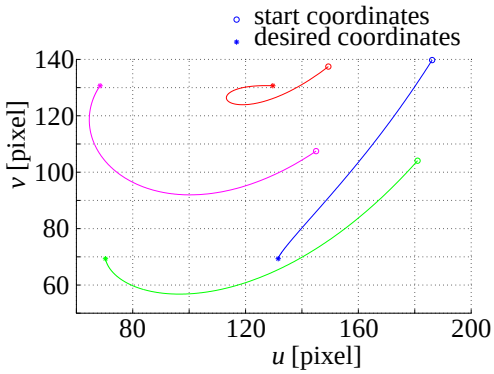

Figure 10: Coordinates of the target points in image space for the coupled motion.

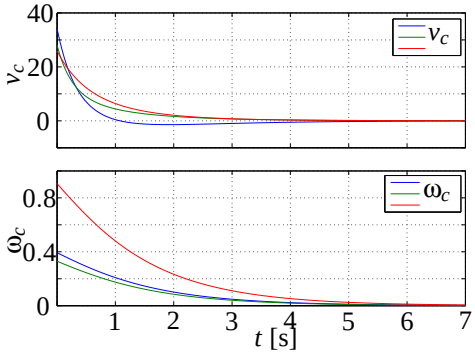

Figure 11: Control input for the coupled motion.

4.4 Influence of Noise

The setup for this simulation is the same as in Section 4.3, but with noise added to the pixels. That is, at each discrete time an image is grabbed a random Gaussian noise N(0.5,2) is added to the pixel coordinates.

As in Section 4.1 the figures 12, 13 and 14 show the movement of the camera in euclidean space, the image coordinates of the four feature points and the control variables.

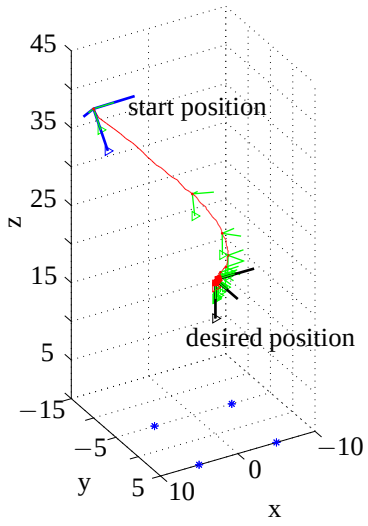

Figure 12: Coupled motion with noise simulation in euclidean space.

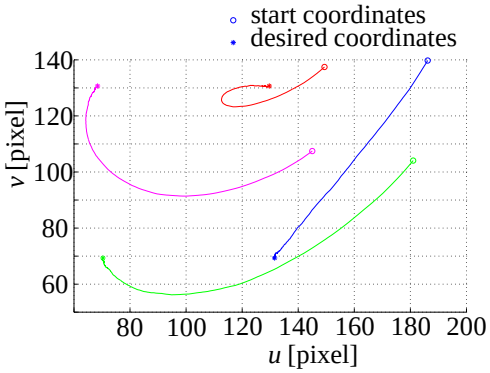

Figure 13: Coordinates of the target point in image space for the coupled motion with noise.

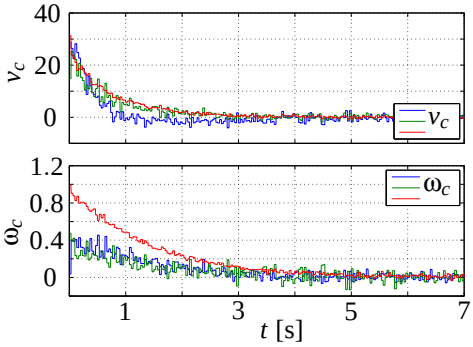

Figure 14: Control input for the coupled motion with noise.

5 CONCLUSIONS

An implementation of an image-based visual servo controller using Matlab and C++ was presented. Various simulations with and without noise were conducted and the controller achieved asymptotic regulation in all cases. This implementation experimentally validates the controller developed in (Hu et al., 2006) and now that the controller is safe to use, new experimenations using the real robot can be carried out.

At this point, the control gains were kept small and the discretized intervals were based on a normal camera (30fps). Those choices let us achieve a convergence in less than 7 seconds. However, for real-world applications, those same choices must be revised so that the convergence can be made a lot faster.

The control model and all software modules used in this paper will be made available on line at http://vigir.missouri.edu

REFERENCES

Chou, J. C. K. and Kamel, M. (1991). Finding the position and orientation of a sensor on a robot manipulator using quaternions. *International Journal of Robotics Research*, 10(3):240–254.

DeSouza, G. N. and Kak, A. C. (2004). A subsumptive, hierarchical, and distributed vision-based architecture for smart robotics. *IEEE Transactions on Systems, Man and Cybernetics - Part B*, 34(5).

Hirsh, R., DeSouza, G. N., and Kak, A. C. (2001). An iterative approach to the hand-eye and base-world calibration problem. In *Proceedings of 2001 IEEE International Conference on Robotics and Automation*, volume 1, pages 2171–2176. Seoul, Korea.

Hu, G., Dixon, W., Gupta, S., and Fitz-Coy, N. (2006). A quaternion formulation for homography-based visual servo control. In *IEEE International Conference on Robotics and Automation*, pages 2391–2396.

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

Saeed, B. (2001). *Introduction to Robotics, Analysis, Systems, Applications*. Prentice Hall Inc.

Spong, M. W. and Vidyasagar, M. (1989). *Robot Dynamics and Control*. John Wiley & Sons.