Image-based Visual Servoing of a Real Robot Using a Quaternion Formulation

T. Koenig, Y. Dong, and G. N. DeSouza
ECE Department
University of Missouri
Columbia, MO, USA
post@tikey.de, ydfff@mizzou.edu, DeSouzaG@missouri.edu

***Abstract*—In this paper we present the results from our visual servoing system for a real industrial robot. In contrast to other visual servoing system in the literature, the one presented here, which was derived from [4], uses a quaternion representation of the rotation instead of the more common matrix representation. By doing so, the proposed system avoids potential singularities introduced by the rotational matrix representation. After performing exhaustive tests in a simulated environment, our controller was applied to a Kawasaki UX150. In the case of simulation, the movement of the camera and the image processing were performed using Matlab-Simulink, which 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. In the case of the real robot, the controller was tested initially using another simulation program provided by Kawasaki Japan and later with the real Kawasaki robot. The setup for testing and the results for all three cases above are presented here, but for more details on the simulations, the reader is encouraged to check [8].**

***Index Terms*—Image-based servo, visual servoing, quaternion.**

I. 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* [5].

For various reasons including simplicity of design, most systems developed to date fall into the position-based, dynamic look-and-move category [2]. 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 [4].

The proposed controller was tested using two simulations and later it was tested on a real robot setting (Figure 1). In that case, a camera was attached on the end-effector

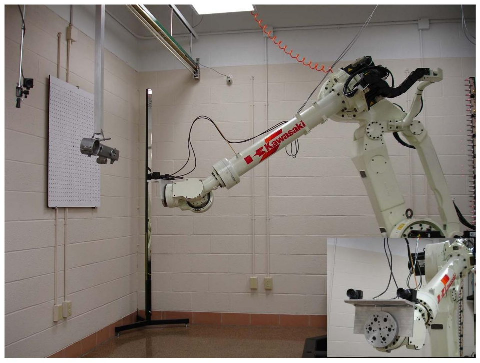
Fig. 1: Robot/Camera setting for testing with detailed view of the cameras on the end-effector (only one is actually used)

and the full closed loop control was applied using the signal from the cameras (image-based control).

We based the development of our controller on the ideas introduced in [4], 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: ${}^{\circ}\mathcal{F}(t)$ and ${}^{*}\mathcal{F}$, where ${}^{\circ}\mathcal{F}(t)$ is affixed to the moving camera and ${}^{*}\mathcal{F}$ represents the desired position of the camera. Figure 2 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:

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

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

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 ${}^{\circ}\mathcal{F}$.

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 Euclidean coordinates of these feature points. Instead, it can only extract the image

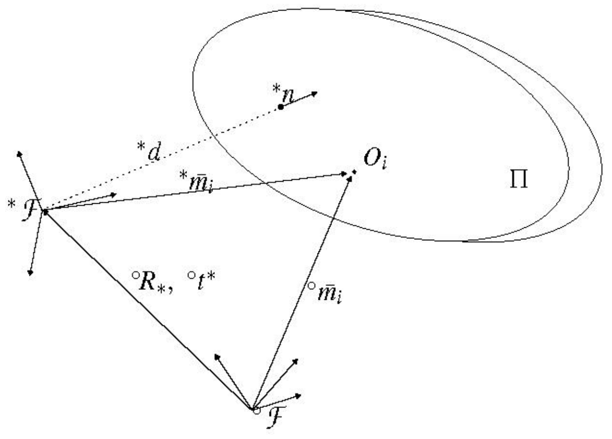

Fig. 2: Relationships between the frames and the plane

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.

II. 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.

A. Homography

In order to estimate the Euclidean coordinates of the feature points from their observed pixel coordinates we use a homography relating the desired and the current pixel coordinates. That is:

$$^{\circ}p_i = \alpha_i A ^{\circ}H_* A^{-1}\, ^*p_i \ . \tag{4}$$

where $^{\circ}\hat{H}_*$ is the 3x3 Homography matrix, A contains the intrinsic parameters of the camera and $\alpha_i(t)$ represents a depth ratio, since the above relation can only be known within a scale factor. The equation above is then used to estimate $^{\circ}\hat{H}_*$ given the observed pixel points and A.

As we will explain next, the actual Euclidean coordinates of the feature points are not explicitly calculated. Instead, once $^{\circ}\hat{H}_*$ is obtained, the rotation and translation components embedded in it are employed in the direct derivation of the control inputs using a quaternion formulation. The conversion from rotation and translation into a quaternion form as well as other derivations omitted here can be found in [9].

B. Controller

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

$$^{\circ}R_*(t) \to I_3 \qquad \text{as} \qquad t \to \inf \tag{5}$$

$$||^{\circ}t^*(t)|| \to 0 \qquad \text{as} \qquad t \to \inf \tag{6}$$

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

$$\begin{aligned} e &= {^{\circ}m_e} - {^*m_e} \\ &= \left[\frac{^{\circ}x_i}{^{\circ}z_i} - \frac{^*x_i}{^*z_i} \quad \frac{^{\circ}y_i}{^{\circ}z_i} - \frac{^*y_i}{^*z_i} \quad \ln\left(\frac{^{\circ}z_i}{^*z_i}\right) \right]^T \end{aligned} \tag{7}$$

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

$$||e(t)|| \to 0 \qquad \text{as} \qquad t \to \inf \ . \tag{8}$$

It can be easily verified that if (8) is satisfied, the extended normalized coordinates will approach the desired extended normalized coordinates, i. e.

$$^{\circ}m_i(t) \to {^*m_i(t)} \ \text{ and } \ ^{\circ}z_i(t) \to {^*z_i(t)} \tag{9}$$

as $t \to \inf$. Moreover, if (8) and (9) are satisfied, (6) is also satisfied.

Similarly, the rotation regulation objective in (5) can be expressed in terms of its quaternion vector $q = [q_0 \ \tilde{q}\ ]^T$, $\tilde{q} = [q_1 \ q_2 \ q_3]^T$[1] by:

$$||\tilde{q}(t)|| \to 0 \qquad \text{as} \qquad t \to \inf \ . \tag{10}$$

In that case, if (8) and (10) are satisfied, the control objective stated in (5) is also satisfied.

For such translational and rotational control objectives, it was shown in [4] that the closed-loop error system is given by:

$$\dot{q}_0 = \frac{1}{2}\tilde{q}^T K_\omega \left(I_3 - \tilde{q}^\times\right)^{-1} \tilde{q} \tag{11}$$

$$\dot{\tilde{q}} = -\frac{1}{2} K_\omega \left(q_0 I_3 - \tilde{q}^\times\right)\left(I_3 - \tilde{q}^\times\right)^{-1} \tilde{q} \tag{12}$$

$$^*z_i \dot{e} = -K_v e + \tilde{z}_i L_\omega \omega_c \tag{13}$$

and the control inputs by:

$$\omega_c = -K_\omega \left(I_3 - \tilde{q}^\times\right)^{-1} \tilde{q} \tag{14}$$

$$v_c = \frac{1}{\alpha_i} L_v^{-1} \left(K_v e + {^*\hat{z}_i} L_\omega \omega_c\right) \tag{15}$$

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; $K_\omega, K_v \in \mathbb{R}^{3\times3}$ are diagonal matrices of positive constant control gains; and the estimation error $\tilde{z}(t) \in \mathbb{R}$ is defined as $\tilde{z}_i = {^*z_i} - {^*\hat{z}_i}$.

A proof of stability for the controller above can be found in [4].

III. IMPLEMENTATION

In summary, the task of controlling a robot with a visual-servoing algorithm was divided into four major steps:

1) Capturing images and obtaining the coordinates of the feature points.
2) Estimating the Homography from the desired and the current pixel coordinates of the feature points.
3) Calculating the quaternion from the Homography.
4) Calculating the input variables, i. e. the velocities of the robot endeffector.
5) Moving the robot according to the given input variables.

The hardware available for this task consisted of a Kawasaki industrial robot with a camera mounted on its endeffector (Figure 1) and a vision-sensor network using GPUs 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). Basically, it follows the steps above until 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 intially replaced by two different simulators. The first simulator, which was implemented in MATLAB-SIMULINK, simulates an arbitrary motion of the camera in space. The camera is represented using a simple pin-hole model and a coordinate frame. With this simulator, it was possible to move the camera according to exact and arbitrary velocities.

In all real testing scenarios performed, the system needed to extract reliable image coordinates of the feature points. To achieve that, the system relied on a very accurate calibration procedure [3]. Given that, the system could then return the image coordinates of the feature points at each time instant t. In the case of the simulated scenarios, a camera simulator was also implemented using C++ classes.

For the second set of simulation scenarios, a program provided by Kawasaki Japan was employed. This program simulates the exact same environment of a real robot, including the possibility of executing other programs we developed and that would eventually be required to move the real robot. This fact, allowed us to test a complete version of the code developed before the real deployment of the system. Some of this code is responsible for performing the forward and inverse kinematics, as well as the dynamics of the robot.

In order to demonstrate the system in a more realistic setting even during simulation, noise was added to the image processing algorithm and a time discretization of the image acquisition was introduced to simulate the camera.

A. 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. [6] and [7]. 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 ${}^{\circ}\mathcal{F}$ 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: $\mathcal{X} = \begin{bmatrix} x & y & z & \phi & \theta & \psi \end{bmatrix}^T$

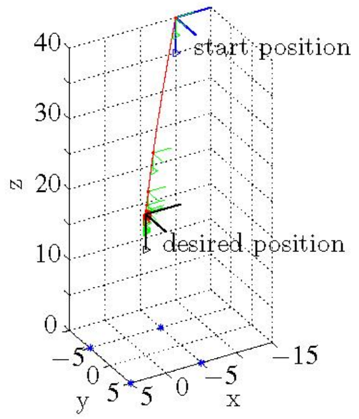

Fig. 3: Linear-motion simulation in euclidean space.

IV. RESULTS

The controller was tested in three different environments: simulated, simulated with noise, and real robot. For each of these environments, the controller was then tested for three different scenarios: pure linear motion; pure angular motion; and combined motion. The four feature points were arranged in a square around the origin of the reference frame: 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:

$$ {}^{*}\mathcal{X}_w = \begin{bmatrix} 0 & 0 & 20 & 0^\circ & 180^\circ & 0^\circ \end{bmatrix}^T \tag{16}$$

The control gains used for the controller were kept small so that we could analyse the trajectory at small intervals.

As mentioned in Section III, we simulated both the noise 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.

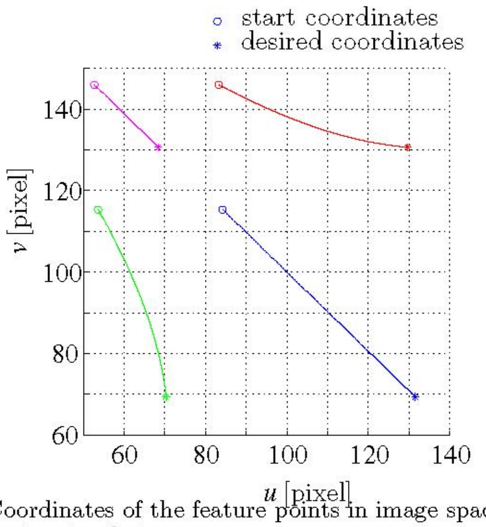

Fig. 4: Coordinates of the feature points in image space for the linear-motion simulation.

A. Linear Motion

In this experiment 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 a few units along the z-axis of the reference frame as well as the x- and the y-axis of the reference frame. A typical start pose used in the tests is (in XYZOAT coordinates):

$$^{o}\mathcal{X}_w = \begin{bmatrix} -10 & -10 & 40 & 0^\circ & 180^\circ & 0^\circ \end{bmatrix}^T \qquad (17)$$

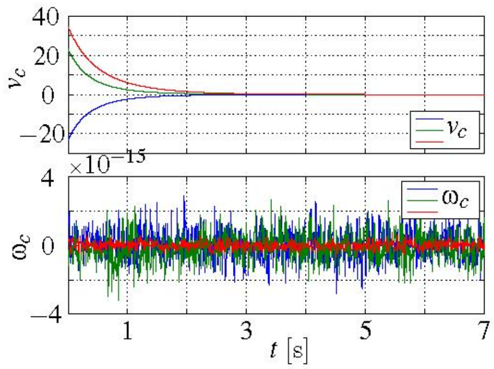

Fig. 5: Control inputs for the linear-motion simulation.

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.

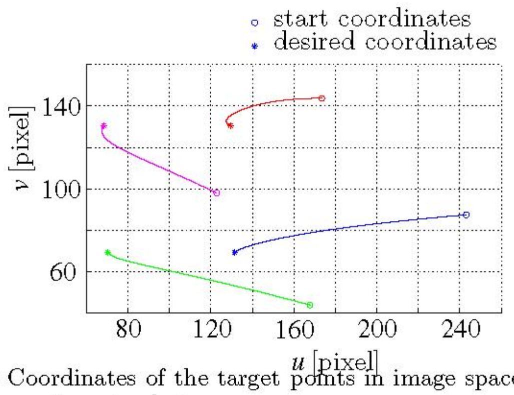

Fig. 6: Coordinates of the target points in image space for the angular-motion simulation.

B. Angular Motion

In this part of the experiments, the start position of the camera was kept identical to the desired position, only the orientation was changed. A typical start pose used was given by:

$$^{o}\mathcal{X}_w = \begin{bmatrix} 0 & 0 & 20 & 45^\circ & 150^\circ & 5^\circ \end{bmatrix}^T \qquad (18)$$

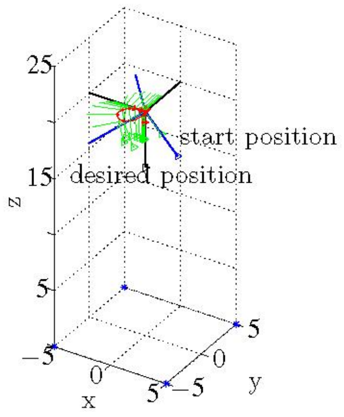

Fig. 7: Angular-motion simulation in euclidean space.

As in Section IV-A, 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.

C. Coupled Motion

Here, the camera could perform any combined motion, i. e. both the position and the orientation at the beginning differ from the desired pose of the camera. Once again, a typical start pose was:

$$^{o}\mathcal{X}_w = \begin{bmatrix} 10 & -10 & 40 & 90^\circ & 140^\circ & 10^\circ \end{bmatrix}^T \qquad (19)$$

As in Section IV-A, Figures 9, 10 and 11 show the movement of the camera in euclidean space, the coordinates of

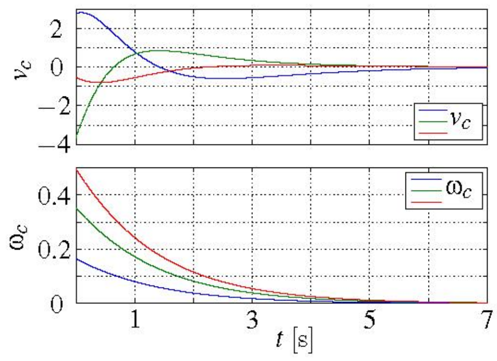

Fig. 8: Control inputs for the angular-motion simulation.

the four feature points in the image plane, and the control variables.

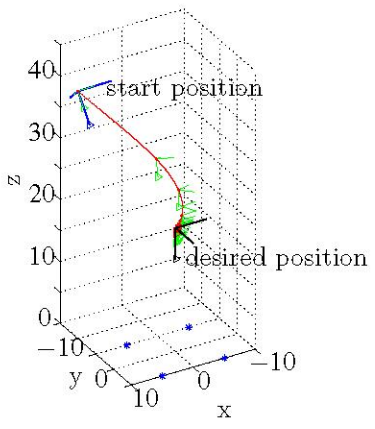

Fig. 9: Coupled-motion simulation in euclidean space.

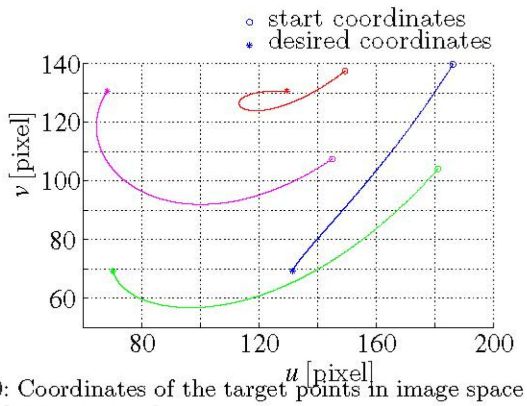

Fig. 10: Coordinates of the target points in image space for the coupled-motion simulation.

D. Influence of Noise

The setup for this simulation is the same as in Section IV-C, but with noise added to the pixel coordinates. That is, at each discrete time a set of image coordinates is simulated and random Gaussian noise N(0.5,2) is added to these same pixel coordinates.

As before, 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, respectively. However, due to the limitation in space, here we depict only the scenario for the coupled motion.

E. Tests with Real Robot

After the controller was validated through the simulations above, we run the same scenarios as in Section IV-C for the real robot.

As in Section IV-D, Figures 15, 16 and 17 show, respectively, the movement of the camera in euclidean space, the image coordinates of the four feature points and the control

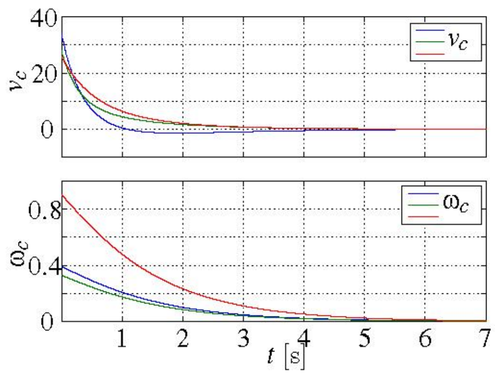

Fig. 11: Control inputs for the coupled-motion simulation.

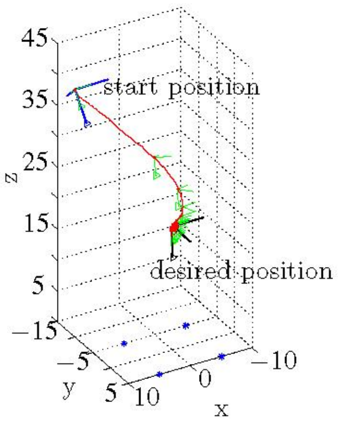

Fig. 12: Display in Euclidean space for the simulation of coupled motion with added noise.

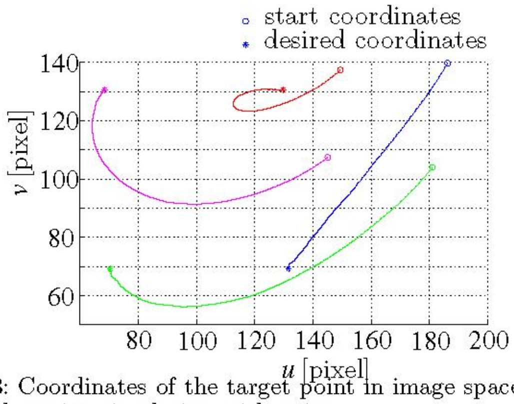

Fig. 13: Coordinates of the target point in image space for the coupled-motion simulation with noise.

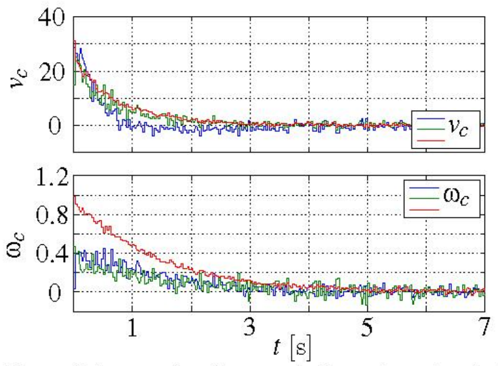

Fig. 14: Control inputs for the coupled-motion simulation with noise.

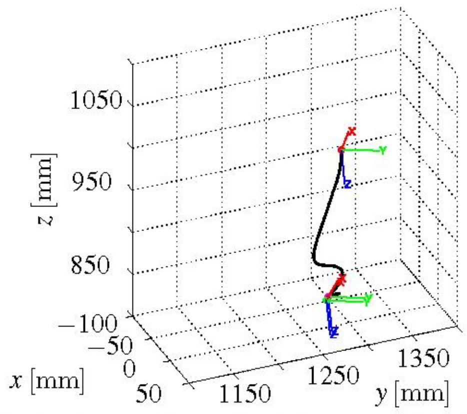

Fig. 15: Real robot moving in Eucledean space for the coupled-motion scenario.

variables. Also as in Section IV-D, here we only depict the case of coupled motion (rotation and translation).

V. CONCLUSIONS

An implementation of an image-based visual servo controller using C++ was presented. Various experiments were conducted using a simulation of the robot, as well as the real robot. For the simulation performed in Matlab-Simulink and a Kawasaki robot simulator, two scenarios were utilized: with and without noise. In all cases, the controller achieved asymptotic regulation. This implementation experimentally validates the controller developed in [4].

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

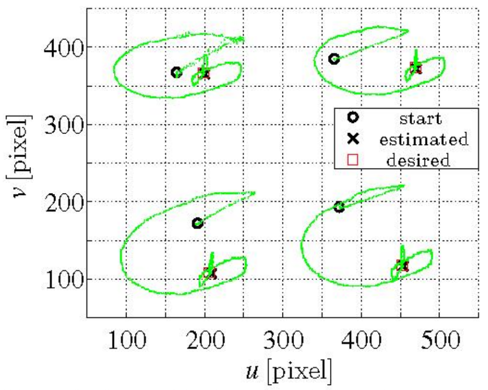

Fig. 16: Coordinates of the target point in image space for the real robot in coupled motion.

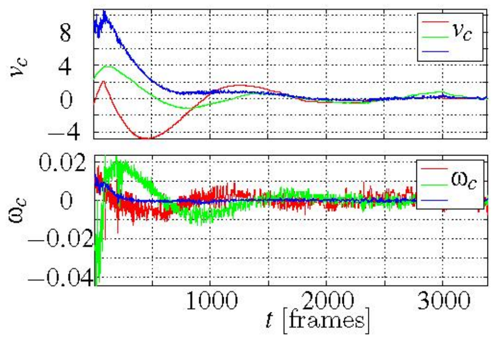

Fig. 17: Control inputs for the coupled motion of the real robot.

ACKNOWLEDGMENT

The authors would like to thank Kawasaki, in special Mitsuichi Hiratsuka, for the use of their simulator.

REFERENCES

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

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

[3] R. Hirsh, G. N. DeSouza, and A. C. Kak. 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, May 2001. Seoul, Korea.

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

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

[6] B.N. Saeed. *Introduction to Robotics, Analysis, Systems, Applications.* Prentice Hall Inc., 2001.

[7] Mark W. Spong and M. Vidyasagar. *Robot Dynamics and Control.* John Wiley & Sons, 1989.

[8] T. Koenig and G. N. DeSouza. Implementation of a Homography-based Visual Servo Control using a Quaternion Formulation. *Proceedings of Fifth International Conference on Informatics in Control, Automation and Robotics*, , May 2008.

[9] T. Koenig and G. N. DeSouza. Implementation of a Homography-based Visual Servo Control using a Quaternion Formulation. *MU-ViGIR Technical Report*, http://vigir.missouri.edu/publications.htm