2007 IEEE International Conference on
Robotics and Automation
Roma, Italy, 10-14 April 2007

On with the Visuomotor Function: A 6DOF Adaptive Approach for Modeling Image-Based Variations and Visual Servoing

Simon Léonard
Department of Computing Science
University of Alberta
Edmonton, Alberta
Email: sleonard@cs.ualberta.ca

Martin Jägersand
Department of Computing Science
University of Alberta
Edmonton, Alberta
Email: jag@cs.ualberta.ca

***Abstract*— In this paper, we proposes a visual servoing method that approximates the relation between the variations of image points and the variations of a stereo rig in Euclidian space. As with most image-based visual servoing methods, commands are expressed in the space of image features. However, instead of relating instantaneous image-based variations to instantaneous variations in Euclidian space, the visuomotor function relates arbitrary image-based variations to Euclidian transformations. The visuomotor function is approximated in real-time by using on-line estimation techniques. The system improves its performance with experience and is able to adapt to different configurations of the cameras or environment. Given the disparities between two sets of corresponding image points, the visuomotor function provides the Euclidian transformation the robot must execute in order to align the image coordinates.**

I. Introduction

One of the challenges of autonomous robots is to interact and control their motion in non-engineered environments. In particular, the problem of using visual feedback to control motion, also known as visual servoing [1], [2], has consistently been at the forefront in recent years. This paper proposes a visual servoing approach based on approximating the relation between the variations of image coordinates and the variations of a stereo rig in Euclidian space. More specifically, we extend the method presented in [3] for translations to six degrees of freedom (DOF). As with image-based visual servoing methods (IBVS), commands are expressed by image coordinates. However, instead of generating a twist, the visuomotor function generates an Euclidian transformation that corresponds to the errors of image coordinates. The approximation of the visuomotor function is based on on-line techniques from the field of machine learning.

This paper is presented as follow. First, we derive a linear expression of the visuomotor function defined for the coordinates of a three dimensional point. Then, we perform rank-1 updates to estimates the parameters of the linear expression by using a recursive least squares algorithm. Finally, we exploit the dependence of the parameters on the coordinates of the three dimensional point to generalize these parameters over a neighbohood of points with a coarse coding method. Furthermore, in order to increase the performance we present a queuing technique that estimates several sets of parameters simultaneously. Finally, we use the resulting approximation to predict variations of image coordinates and to estimate the transformation corresponding to visual alignments.

II. Background

Visual servoing methods are classified according to whether a task is expressed by the parameters of image features or by an Euclidian transformation. In the latter case, an Euclidian transformation represents the desired relative pose between the robot and the scene. Given the error between the desired and current poses, the controller moves the robot toward the desired pose. One implication of this strategy is that the relative pose must be estimated at each control iteration. By itself, pose estimation is a broad area within computer vision and several algorithms have been proposed over the years. However, these methods often make important assumptions that have consequences on the applications for which this control architecture can be used. Chief among those is that knowledge about the environment is often required or must be acquired. Example of such knowledge are maps of the surrounding environment [4] or a CAD model of the object being manipulated [5]. Furthermore, the mere fact of estimating a position from images implies that the cameras are calibrated and that a world coordinate frame has been defined. Thus, the dependence on prior knowledge and calibrations often make such architecture not suitable for visual control with autonomous robots operating in an unknown environment.

A different strategy consists of using the parameters of image features as input. These parameters represent the configuration that each feature must reach under the desired view. The challenge consists of regulating the corresponding features in the current image toward their desired parametrization. Most visual servoing methods developed around this image-based paradigm use differential control laws to guide the robot. Most notably, the image Jacobian J is defined as the mapping between instantaneous variations of image feature parameters $\mathbf{p}$ and instantaneous variations of a coordinate frame in Euclidian space $\mathbf{q}$

$$\delta\mathbf{p} = J\delta\mathbf{q}. \tag{1}$$

This formulation has been the backbone of visual servoing for several years and the image Jacobians of several features have been documented [6]. Most of the research in this area focuses on developing new image features with desirable properties [7] or on the estimation of the image Jacobian [8]. One important consequence of using this strategy is that its global convergence and stability must be taken into account. Furthermore, the trajectory of the robot is implicitly dictated by the controller, which can lead to erratic motions [9]. Several methods have addressed this issue [10] but none truly replaces the versatility of Euclidian trajectory generators. Others have addressed these issues by developing different control strategies such as second order controllers [11] or by fragmenting the control task into independent sub-tasks [12], [13].

Our approach is similar to the image-based formulation of Equation 1. As with IBVS, our commands are defined within the same space of image coordinates. However, instead of computing the twist $\delta\mathbf{q}$ corresponding to the instantaneous variations $\delta\mathbf{p}$ by using $\delta\mathbf{q} = J^{-1}\delta\mathbf{p}$, we approximate the visuomotor function

$$\Delta\mathbf{p} = V(E), \tag{2}$$

which relates arbitrary variation $\Delta\mathbf{p}$ of image coordinates to a corresponding Euclidian transformations E. In the next sections, we derive a linear formulation of $V(E)$ and propose a method to estimate its parameters on-line.

III. Visuomotor Function

We begin by formulating the eye-in-hand configuration illustrated in Fig. 1. The illustration shows two cameras (left and right) with their respective coordinates frames L and R. Both cameras are mounted on a stereo rig S according to the homogeneous transformations ${}^{L}E_{S}$ and ${}^{R}E_{S}$. The stereo rig is attached to the end-effector of the robot and since ${}^{L}E_{S}$ and ${}^{R}E_{S}$ are arbitrary, we assume that the frame of the end-effector coincides with the frame of the stereo rig. The coordinate frame B represents the base coordinate frame of the robot and all three dimensional points ${}^{B}\mathbf{P}$ are expressed with respect to that frame. Therefore, the frames B and S are related by the forward kinematics given by ${}^{S}E_{B}$. Since ${}^{S}E_{B}$ is known from the forward kinematics we express the following derivation in the frame S.

For conciseness, we only show the derivation for the left camera. Results for the right camera are in every way identical. The first step is to derive a linear equation representing the image coordinates of a point ${}^{S}\mathbf{P}$ before any displacement of the frame S. The second step is to repeat the derivation while considering an arbitrary displacement of the frame S. Then, the visuomotor function is derived by subtracting the two equations.

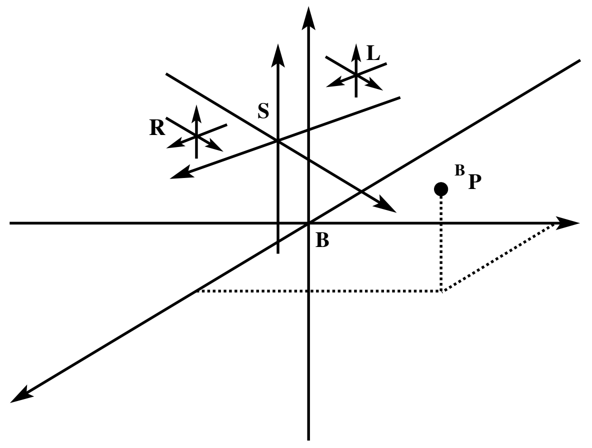


Fig. 1. Geometric model: Camera frames L and R are mounted on a stereo rig S.

Under a projective camera model, the projective coordinates of a point ${}^{S}\mathbf{P}$ with homogeneous coordinates ${}^{S}\mathbf{P} = \begin{bmatrix} X & Y & Z & 1 \end{bmatrix}^T$ are given by

$$\mathbf{p} = {}^{L}K_{L}\,{}^{L}E_{S}\,{}^{S}\mathbf{P} = \begin{bmatrix} x & y & z \end{bmatrix}^T \tag{3}$$

where ${}^{L}K_{L}$ is the projection matrices of the left camera.

As a first step, we define the following matrix

$${}^{L}K_{L}\,{}^{L}E_{S} = \begin{bmatrix} \mathbf{m}_1 \\ \mathbf{m}_2 \\ \mathbf{m}_3 \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 \\ b_1 & b_2 & b_3 & b_4 \\ c_1 & c_2 & c_3 & c_4 \end{bmatrix}. \tag{4}$$

Using Equations 4 in Equations 3 we express the image coordinates of ${}^{S}\mathbf{P}$ before a displacement of the stereo rig by

$$\mathbf{p}_i^1 = \begin{bmatrix} x_i^1 \\ y_i^1 \end{bmatrix} = \begin{bmatrix} \frac{x}{z} \\ \frac{y}{z} \end{bmatrix} = \begin{bmatrix} \frac{\mathbf{m}_1 {}^{S}\mathbf{P}}{\mathbf{m}_3 {}^{S}\mathbf{P}} \\ \frac{\mathbf{m}_2 {}^{S}\mathbf{P}}{\mathbf{m}_3 {}^{S}\mathbf{P}} \end{bmatrix}. \tag{5}$$

By expanding Equation 5 and rearranging the terms we obtain

$$\begin{aligned} x_i^1 = {} & \frac{a_1 X}{c_4} + \frac{a_2 Y}{c_4} + \frac{a_3 Z}{c_4} + \frac{a_4}{c_4} \\ & - \frac{c_1 X}{c_4} x_i^1 - \frac{c_2 Y}{c_4} x_i^1 - \frac{c_3 Z}{c_4} x_i^1 \end{aligned} \tag{6}$$

$$\begin{aligned} y_i^1 = {} & \frac{b_1 X}{c_4} + \frac{b_2 Y}{c_4} + \frac{b_3 Z}{c_4} + \frac{b_4}{c_4} \\ & - \frac{c_1 X}{c_4} y_i^1 - \frac{c_2 Y}{c_4} y_i^1 - \frac{c_3 Z}{c_4} y_i^1 \end{aligned} \tag{7}$$

Now, we derive similar equations for ${}^{S}\mathbf{P}$ after moving the coordinate frame S according to the homogeneous transformation

$$H = \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 & \mathbf{t} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

and

$$\mathbf{p}_i^2 = \begin{bmatrix} x_i^2 \\ y_i^2 \end{bmatrix} = \begin{bmatrix} \frac{\mathbf{m}_1 H {}^{S}\mathbf{P}}{\mathbf{m}_3 H {}^{S}\mathbf{P}} \\ \frac{\mathbf{m}_2 H {}^{S}\mathbf{P}}{\mathbf{m}_3 H {}^{S}\mathbf{P}} \end{bmatrix}. \tag{8}$$

In order to reduce the length of the equations we introduce the following vectors

$$\mathbf{a} = \begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix};$$
$$\mathbf{b} = \begin{bmatrix} b_1 & b_2 & b_3 \end{bmatrix}; \quad \mathbf{c} = \begin{bmatrix} c_1 & c_2 & c_3 \end{bmatrix}.$$

Using a manipulation similar to the one used for Equations 6 and 7 we find

$$\begin{aligned} x_i^2 = &\frac{X\mathbf{a}}{c_4}\mathbf{r}_1 + \frac{Y\mathbf{a}}{c_4}\mathbf{r}_2 + \frac{Z\mathbf{a}}{c_4}\mathbf{r}_3 + \frac{\mathbf{a}}{c_4}\mathbf{t} + \frac{a_4}{c_4} \\ &- \frac{X\mathbf{c}}{c_4}x_i^2\mathbf{r}_1 - \frac{Y\mathbf{c}}{c_4}x_i^2\mathbf{r}_2 - \frac{Z\mathbf{c}}{c_4}x_i^2\mathbf{r}_3 - \frac{\mathbf{c}}{c_4}x_i^2\mathbf{t} \end{aligned} \tag{9}$$

$$\begin{aligned} y_i^2 = &\frac{X\mathbf{b}}{c_4}\mathbf{r}_1 + \frac{Y\mathbf{b}}{c_4}\mathbf{r}_2 + \frac{Z\mathbf{b}}{c_4}\mathbf{r}_3 + \frac{\mathbf{b}}{c_4}\mathbf{t} + \frac{b_4}{c_4} \\ &- \frac{X\mathbf{c}}{c_4}y_i^2\mathbf{r}_1 - \frac{Y\mathbf{c}}{c_4}y_i^2\mathbf{r}_2 - \frac{Z\mathbf{c}}{c_4}y_i^2\mathbf{r}_3 - \frac{\mathbf{c}}{c_4}y_i^2\mathbf{t} \end{aligned} \tag{10}$$

Finally, subtracting Equation 6 from 9 and 7 from 10 we obtain the six degrees of freedom visuomotor function for the point ${}^S\mathbf{P}$

$$\begin{aligned} x_i^2 - x_i^1 = &\frac{X\mathbf{a}}{c_4}(\mathbf{r}_1 - \mathbf{e}_1) + \frac{Y\mathbf{a}}{c_4}(\mathbf{r}_2 - \mathbf{e}_2) \\ &+ \frac{Z\mathbf{a}}{c_4}(\mathbf{r}_3 - \mathbf{e}_3) + \frac{\mathbf{a}}{c_4}\mathbf{t} \\ &- \frac{X\mathbf{c}}{c_4}(x_i^2\mathbf{r}_1 - x_i^1\mathbf{e}_1) - \frac{Y\mathbf{c}}{c_4}(x_i^2\mathbf{r}_2 - x_i^1\mathbf{e}_2) \\ &- \frac{Z\mathbf{c}}{c_4}(x_i^2\mathbf{r}_3 - x_i^1\mathbf{e}_3) - \frac{\mathbf{c}}{c_4}x_i^2\mathbf{t} \end{aligned} \tag{11}$$

$$\begin{aligned} y_i^2 - y_i^1 = &\frac{X\mathbf{b}}{c_4}(\mathbf{r}_1 - \mathbf{e}_1) + \frac{Y\mathbf{b}}{c_4}(\mathbf{r}_2 - \mathbf{e}_2) \\ &+ \frac{Z\mathbf{b}}{c_4}(\mathbf{r}_3 - \mathbf{e}_3) + \frac{\mathbf{b}}{c_4}\mathbf{t} \\ &- \frac{X\mathbf{c}}{c_4}(y_i^2\mathbf{r}_1 - y_i^1\mathbf{e}_1) - \frac{Y\mathbf{c}}{c_4}(y_i^2\mathbf{r}_2 - y_i^1\mathbf{e}_2) \\ &- \frac{Z\mathbf{c}}{c_4}(y_i^2\mathbf{r}_3 - y_i^1\mathbf{e}_3) - \frac{\mathbf{c}}{c_4}y_i^2\mathbf{t} \end{aligned} \tag{12}$$

where the units vectors $\mathbf{e}_i$ are defined by

$$\mathbf{e}_1 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T \quad \mathbf{e}_2 = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T \quad \mathbf{e}_3 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T.$$

In order to use Equations 11 and 12 for any practical purpose, the following problems must be addressed. First, all 36 parameters $X\mathbf{a}/c_4$, $Y\mathbf{a}/c_4$, $Z\mathbf{a}/c_4$, $\mathbf{a}/c_4$, $X\mathbf{b}/c_4$, $Y\mathbf{b}/c_4$, $Z\mathbf{b}/c_4$, $\mathbf{b}/c_4$, $X\mathbf{c}/c_4$, $Y\mathbf{c}/c_4$, $Z\mathbf{c}/c_4$ and $\mathbf{c}/c_4$ must be estimated. Also, these parameters will only be valid for a specific three dimensional point, namely ${}^S\mathbf{P}$. We address the first problem by using a recursive least squares algorithm with rank-1 updates and the later by using a function approximation known a coarse coding. The combination of these two methods allows the system to estimate the parameters on-line and to improve its performances in real-time. Also, this on-line scheme allows the system to adapt to occasional variations of imaging parameters as well as some actuating parameters. To show this, we write Equation 3 as

$$\mathbf{p} = {}^LK_L{}^LE_S(DH){}^S\mathbf{P},$$

where D is some distortion matrix acting on the end-effector. Instead, by postmultiplying ${}^LK_L{}^LE_S$ with D, the distortion can be included in the parameters.

IV. Recursive Least Squares with Rank-1 Updates

To estimate the 36 parameters of Equations 11 and 12, we use a recursive least squares algorithm based on rank-1 updates of the normal equation [14]. This provides the capability to improve the estimation of the parameters as new data becomes available. Even though this approach can adapt to changing camera parameters, it is assumed that they remain constant for a minimal period of time in order for the estimation to converge.

Writing the Equations 11 and 12 as $A\boldsymbol{\theta} = \mathbf{b}$, the recursive least-squares algorithm uses M blocks of equations $A_m\boldsymbol{\theta} = \mathbf{b}_m$ to minimize

$$\underset{\boldsymbol{\theta}}{\text{argmin}} \sum_{m=1}^{M} ||\mathbf{b}_m - A_m\boldsymbol{\theta}||^2. \tag{13}$$

Given the kth block of equations, the algorithm recursively computes the kth update of the parameters $\boldsymbol{\theta}_k$ with

$$\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} + \Sigma_k A_k^T(\mathbf{b}_k - A_k\boldsymbol{\theta}_{k-1}). \tag{14}$$

where Σ_k is the inverse of the kth cross-product matrix computed from

$$\Sigma_k^{-1} = \Sigma_{k-1}^{-1} + A_k^T A_k. \tag{15}$$

with the initial condition $\Sigma_0^{-1} = 0$.

However, in our problem, this involves the inversion of a 36×36 matrix at each update. Instead, we use the equivalent formulation presented in [15] obtained by applying the matrix inversion lemma [16]

$$(\Sigma^{-1} + AR^{-1}A^T)^{-1} = \Sigma - \Sigma A(A^T\Sigma A + R)^{-1}A^T\Sigma.$$

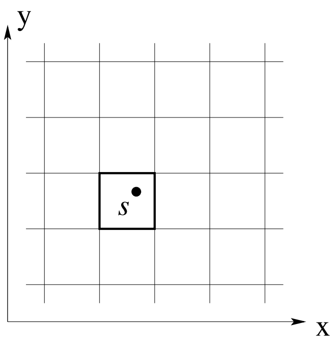
to Equation 15 and we obtain

$$\Sigma_k = \Sigma_{k-1} - \Sigma_{k-1}A_k^T(A_k\Sigma_{k-1}A_k^T + I)^{-1}A_k\Sigma_{k-1}. \tag{16}$$

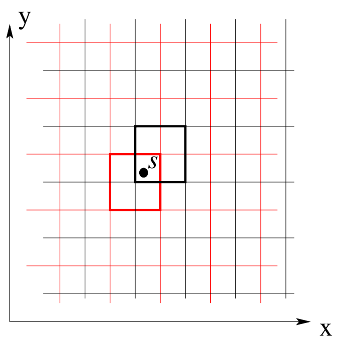
This formulation avoids the inversion of Σ_k^{-1} and, when using rank-1 updates, the matrix inversion of Equation 16 collapses to a scalar inversion. However, the initial condition $\Sigma_0^{-1} = 0$ does not hold anymore. We solve this problem by performing a bootstrapping iteration by accumulating a sufficient number of equations and using Equation 15 with $\Sigma_0^{-1} = 0$. After this initial step, the matrix Σ_1 is computed from Σ_1^{-1} and Equation 16 is used afterward.

V. Function Approximation

Equations 11 and 12 indicate the dependence of the parameters on the coordinates of ${}^S\mathbf{P}$. It follows that each three dimensional point must hold a distinct set of parameters. Nevertheless, this dependence is linear. As such, small variations in X, Y and Z will result in small variations of the parameters. We exploit this property to avoid the estimation of the parameters for each ${}^S\mathbf{P}$ by generalizing the parameters over a



(a) Single tiling of a 2D state space.



(b) Superposition of two tilings.

Fig. 2. Tile Coding.

neighborhood. In particular, we use a function approximation method known as *tile coding* [17].

Tile coding rasterize a space $\mathcal{S}$ by a set of tiles t laid out as a tiling as illustrated in the two dimensional example of Fig 2(a). Each state $s \in \mathcal{S}$ is encoded by the identification number of the tile to which it belongs. In Fig. 2(a), the state s is displayed and its associated tile is highlighted. Though effectively reducing the cardinality, this encoding also reduces the resolution. To overcome this side effect, several tilings are superposed with different offsets as shown in Fig. 2(b). A state s is then encoded by all the tiles, one per tiling, containing s. In general, for N tilings, a state is encoded by the union of all such tiles

$$s = \bigcup_{n=1}^{N} t_{m,n} \tag{17}$$

where $t_{m,n}$ is the m^{th} tile of the n^{th} tiling. In Fig. 2(b), an additional tiling is used and the state s is encoded by the two tiles highlighted. An important observation is that the volume of a tile represents the space in which its parameters are generalized, whereas the number of tilings can be adjusted to achieve a desired resolution.

In our problem, each point ${}^{S}\mathbf{P} \in R^3$ is mapped to a state $s \in R^4$ consisting of a stereo point $({}^{L}x_i, {}^{L}y_i, {}^{R}x_i, {}^{R}y_i)$, which is then encoded by a set of tiles $t_{m,n}$ according to Equation 17. Each tile contains a set of parameters and the parameters θ_s for s are computed by taking the average of all the associated parameters

$$\theta_s = \frac{1}{N}\sum_{n=1}^{N} \theta_{m,n} \tag{18}$$

where $\theta_{m,n}$ are the parameters of the m^{th} tiles of the n^{th} tiling.

We conclude this section by connecting the tile coding approximation with the least squares estimation of section IV. Let a three dimensional point ${}^{S}\mathbf{P}$ be encoded by a state s_1. After a rigid transformations H of the stereo rig, ${}^{S}\mathbf{P}$ is encoded by the state s_2. Using the image-based variations $\Delta x_i = x_i^2 - x_i^1$ and $\Delta y_i = y_i^2 - y_i^1$ and H, the parameters of all the tiles associated with s_1 are updated with rank-1 updates using the recursive least squares algorithm.

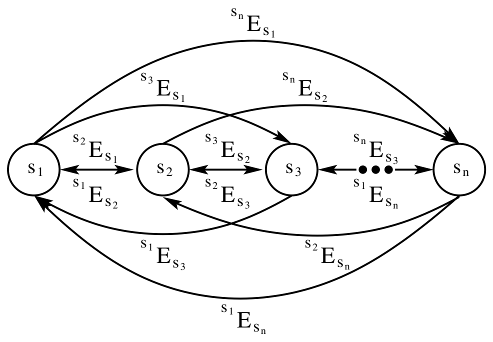


Fig. 3. List of visited states and the possible updates combinations.

VI. Prediction and Control

The visuomotor function can be used to predict image variations and for motion control. For the prediction aspect, Equations 11 and 12 can be use directly to predict the image variations resulting from a transformation H. Using the visuomotor formulation, a visual servoing task is specified in the image space by the current coordinates ${}^{L}\mathbf{p}_i^1$, ${}^{R}\mathbf{p}_i^1$, and the command coordinates ${}^{L}\mathbf{p}_i^2$ and ${}^{R}\mathbf{p}_i^2$, where the left superscripts indicate the cameras. Using $\Delta^{L}x_i^i$, $\Delta^{L}y_i^i$, $\Delta^{R}x_i^i$ and $\Delta^{R}y_i^i$, Equations 11 and 12 are formulated as a least squares problem and solved for $\mathbf{r}_1$, $\mathbf{r}_2$, $\mathbf{r}_3$ and $\mathbf{t}$. Given that each stereo point provides four equations, two for the left camera and two for the right camera, three corresponding stereo points are required.

VII. Implementation

The on-line least squares procedure described in section IV is computationally efficient when each update consists of a rank-1 update. Taking advantage of the symmetry of Σ_k, each update takes about 10μs on our system. Given that our cameras are operating at 30 frames per second, this provides a budget of roughly 3000 updates between frames after deducting time for image processing and tracking. Since we use 32 tilings to encode the parameters of each cameras, we are left with the possibility to update about 45 different states between frames[1].

Since tracking that many targets is a practical challenge and a computational burden, we only track a single target. However, we update the states visited by the target along the way by storing them in a list as shown in Fig 3. Given an initial state s_1, this state will be transformed in a state s_2 after moving the end-effector according to ${}^{S_2}E_{S_1}$. At this point, s_2 is used to update s_1 and s_1 is used to update s_2 by computing ${}^{S_1}E_{S_2}$. As the end-effector moves further, the new state s_3 is used to update s_1 and s_2 and vice versa. This process is repeated at each iteration. For a list containing n states, this effectively allows n^2 updates combinations at each iteration.

VIII. Experiments and Results

We implemented our method on our platform consisting of a seven degrees of freedom Whole Arm Manipulator (WAM).

[1]In practice, this number is even higher since we only update when the image coordinates vary by more than two pixels, which, depending on the velocity, is typically every two or three frames

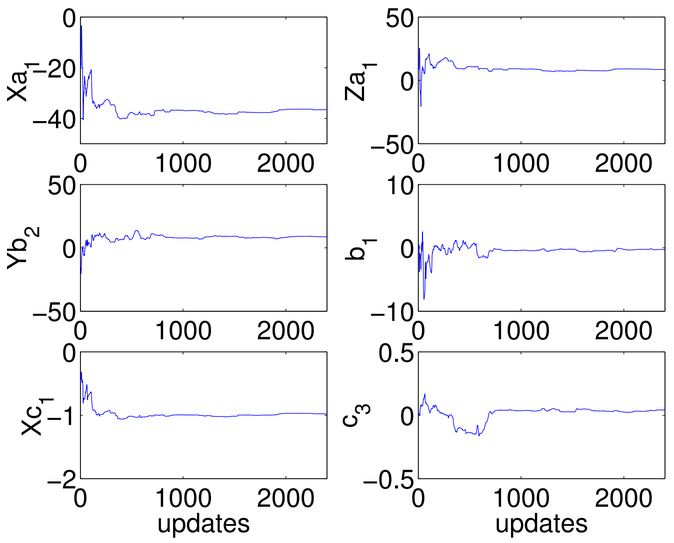


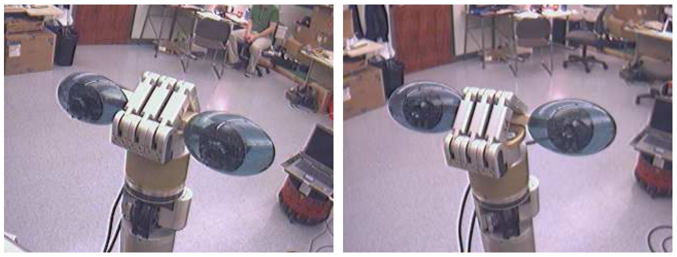
Fig. 4. Convergence of various parameters.

We used the robot hand to hold a stereo rig consisting of two IEEE1394 webcams operating at 30 frames per second as shown in Fig 5(a). We emphasize the fact that no *a priori* model of the hand, stereo rig or cameras was necessary and that the coordinates of the end-effector, as provided by the forward kinematics, was the only coordinate frame used. The computer used for the experiments was used a 3.2GHz CPU with 1GB or RAM. We used tiles of $32 \times 32 \times 32 \times 32$ in size and 32 tilings for each cameras.

In the first set of experiments, we observed the convergence of the parameters after each rank-1 update. Trackers were initialized on a target, which resulted in an initial state s_1. Then, the arm was moved and the parameters of s_1 were estimated on-line. After each update, all 36 parameters of a tile corresponding to s_1 were recorded. Samples of these values are plotted in Fig 4. The figure shows that most parameters converge within 500 updates. In comparison, results for 3DOF reaching movements (9 parameters) presented in [3] converged within 60 updates.

In the second experiment, we tested the adaptiveness of our algorithm. For this, we repeated the previous experiment until 2500 updates were performed. Then, we changed the grasp of the hand on the stereo rig. This modification, illustrated in Fig 5(a) and 5(b), resulted in a panning and tilting of the stereo rig as well as a small translation. Then, the end-effector was positioned in order to move the target at the same initial state s_1. Finally, the arm was then moved for another 1000 updates and the parameters of the same tile were recorded. Accordingly, the result sample illustrated in Fig 6 shows the reaction of a parameter around the 2500th update. From there, 500 additional updates are required before the parameters converges to a new value.

In our final experiment, we used the visuomotor function for a visual servoing task by estimating the Euclidian transformation relating two different views of the same scene. Four states



(a) Before. (b) After.

Fig. 5. Modification of the hand-eye configuration.

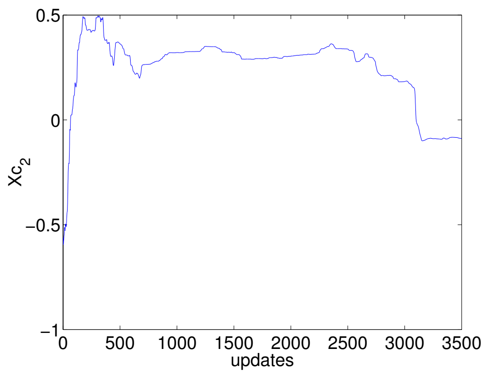


Fig. 6. Convergence of the estimation after modifying camera paramers at update 2500.

originating from four different targets were initialized and tracked. At each frame, we used the visuomotor function to estimate the Euclidian transformation between the initial states and the current states as provided by the trackers. The result was then compared with the actual transformation provided by the forward kinematics. The translational error was obtained by subtracting the respective components, $\begin{bmatrix} t_x & t_y & t_z \end{bmatrix}^T = |\hat{\mathbf{t}} - \mathbf{t}|$ while the error of orientation was obtained according to [18]

$$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \frac{1}{2}(\hat{\mathbf{r}}_1 \times \mathbf{r}_1 + \hat{\mathbf{r}}_2 \times \mathbf{r}_2 + \hat{\mathbf{r}}_3 \times \mathbf{r}_3)$$

where the $\hat{}$ denotes vectors obtained from the visuomotor function.

However, these results do not properly reflect the performance of the system due to the conditioning of the imaging process. That is, since our controller measures errors in the image space, as opposed to Euclidian, a more meaningful experiment is to reproject the estimated transformations in the image space in order to predict image based variations of the transformations. These predictions were compared to measured variations and the results for one of the targets are illustrated in Fig 8. Again, the plots indicate that the error of image-based predictions drops after 500 frames. The mean absolute errors of the plots are reported in Table I.

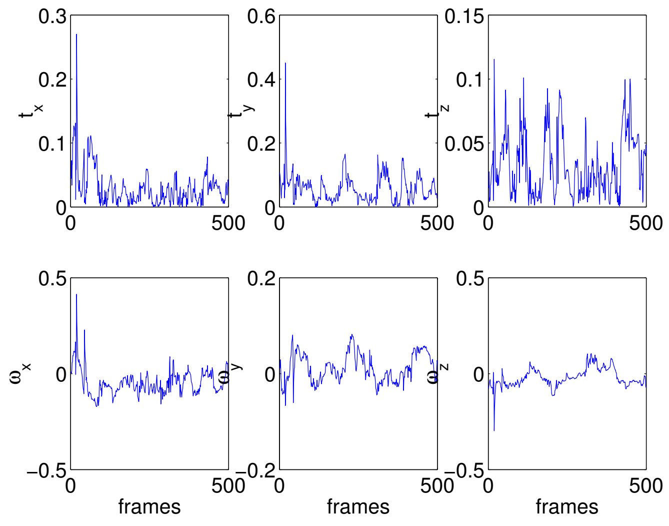


Fig. 7. Error between the Euclidian transformation obtained from the visuomotor function and the one measured from the kinematics.

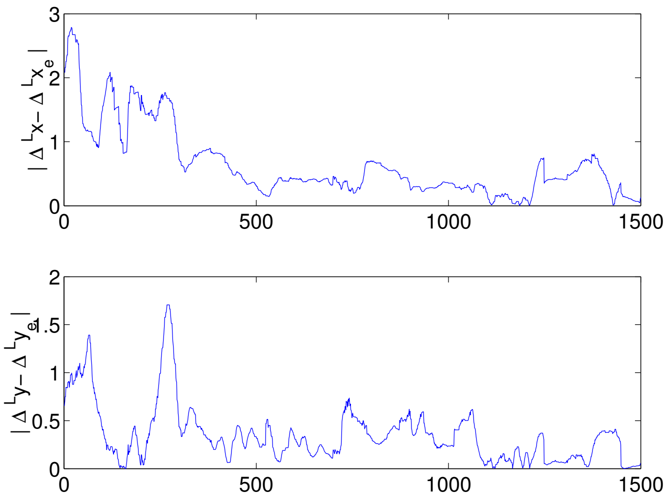


Fig. 8. Error between reprojected transformations and measured variations of a target.

TABLE I

MEAN ABSOLUTE ERROR BETWEEN VARIATIONS OR REPROJECTED TRANSFORMATIONS AND MEASURED VARIATIONS

$\lvert\Delta^L x - \Delta^L x_e\rvert$	$\lvert\Delta^L y - \Delta^L y_e\rvert$	$\lvert\Delta^R x - \Delta^R x_e\rvert$	$\lvert\Delta^R y - \Delta^R y_e\rvert$
0.3579	0.3745	0.7356	0.5094

IX. Conclusion

We presented an on-line approach for modeling image-based variations and visual servoing. The method is based on the linear relationship between image coordinates and Euclidian transformations. Recursive least squares is used to estimate on-line the parameters of the relationship as the robot moves. Since the parameters are only valid for the coordinates of a 3D point, we use coarse coding to generalize the parameters over a neighborhood around that point. Also, we presented an algorithm for increasing the speed of the approximation. Results showed that our method performs well at minimizing image-based error despite begin limited to a single control iteration.

References

[1] S. Hutchinson, G. D. Hager, and P. I. Corke, "A tutorial on visual servo control," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 5, pp. 651–670, October 1996.

[2] K. Ashimoto, "A review on vision-based control of robot manipulators," *Advanced Robotics*, vol. 17, no. 10, pp. 969–991, 2003.

[3] S. Léonard and M. Jägersand, "Adaptive control for estimating translations from image-based variations," in *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, May 2006, pp. 4106–4111.

[4] F. Dellaert, W. Burgard, D. Fox, and S. Thrun, "Using the condensation algorithm for robust, vision-based mobile robot localization," in *Proceedings of the 1999 Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, June 1999, pp. 588–594.

[5] D. G. Lowe, "Fitting parameterized three-dimensional models to images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 5, pp. 441–450, May 1991.

[6] B. Espiau, F. Chaumette, and P. Rives, "A new approach to visual servoing in robotics," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 3, pp. 313–326, June 1992.

[7] F. Chaumette, "Image moments: a general and useful set of features for visual servoing," *IEEE Transactions on Robotics*, vol. 20, no. 4, pp. 713–723, August 2004.

[8] M. Jagersand, "Visual servoing using trust region methods and estimation of the full coupled visual-motor jacobian," in *IASTED Applications of Robotics and Control '96*, 1996.

[9] F. Chaumette, "Potential problems of stability and convergence in image-based and position-based visual servoing," in *The Confluence of Vision and Control*, ser. Lecture Notes in Control and Information Systems, D. Kriegman, G. Hager, and A.Morse, Eds. Springer-Verlag, 1998, vol. 237, pp. 66–78.

[10] Y. Mezouar and F. Chaumette, "Path planning for robust image-based control," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 4, pp. 534–549, August 2002.

[11] S. Benhimane and E. Malis, "Real-time image-based tracking of planes using efficient second-order minimization," in *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2004.

[12] P. I. Corke and S. A. Hutchinson, "A new partitioned approach to image-based visual servo control," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 4, pp. 507–515, August 2001.

[13] E. Malis, F. Chaumette, and S. Boudet, "Positioning a coarse-alibrated camera with respect to an unknown object by 2d 1/2 visual servoing," in *Proceedings of the 1998 IEEE International Conference on Robotics and Automation*, May 1998, pp. 1352–1359.

[14] D. P. Bertsekas, "Incremental least-squares methods and the extended kalman filter," *SIAM J. on Optimization*, 1995.

[15] J. B. Moore, "On strong consistency of least squares identification algorithms," *Automatica*, 1978.

[16] B. D. O. Anderson and J. B. Moore, *Optimal Filtering*. Prentice Hall, 1979.

[17] R. Sutton and A. Barto, *Reinforcement Learning*. The MIT Press, 1999.

[18] J. Y. S. Luh, M. W. Walker, and R. P. C. Paul, "Resolved acceleration control of mechanical manipulators," *IEEE Transactions on Automatic Control*, vol. 25, no. 3, pp. 468–474, 1980.