Actor-Critic Architecture to Increase the Performance of a 6-DOF Visual Servoing Task

Marco A. Perez, Peter A. Cook
Control Systems Centre, UMIST
Manchester UK
PO Box 88, M60 1QD
Email: marcopc@ieee.org p.cook@umist.ac.uk

Abstract— **In this paper an image-based visual servoing algorithm (IBVS) is used to achieve tracking of a moving object. Our objective is to investigate the use of reinforcement learning (RL) algorithms such as an adaptive critic to increase the overall performance. The IBVS is designed to drive the TQ MA2000 6-DOF robotic manipulator with a camera attached to its end-effector. Like other tracking problems, the aim is to keep the camera in a plane parallel to the tracked object. The classical RL actor-critic architecture performs on-line adjustment of the linear trajectory regulator in order to achieve real-time improvement and adaptation without losing an acceptable regulation. The RL system learns directly from data in the image space and the current state of the robot. The system presented in this paper uses an actor-critic scheme built on two feedforward neural networks. The actor network performs the adjustment whereas the adaptive critic stores and assigns action values. Several neuro-dynamic problems derived from the image-robot interaction had to be solved on the way. The scheme is simulated using a detailed model of the IBVS tracking task whereas real-time results are obtained from the experimental setup. The results are discussed in the last section of the paper.**

I. Introduction

Visual Servoing (VS) is a mature research area which shares common issues with robotic control, real-time systems and in special with computer vision, involving subjects often employed in VS schemes such as active vision, visual pose estimation and dynamic vision. Different implementations of VS have been traditionally identified within two main classifications: position-based and image-based visual servoing. An introductory overview of both classes is presented in the now classic VS tutorial in [1]. Basically, in the image-based visual servoing (IBVS) the error signal is computed in the image plane and the regulation commands are generated with respect to such error by means of a visual Jacobian. On the other hand, in the position-based schemes, the image features are used to estimate an object-workspace characterization in such a way that the error can be computed in the Cartesian space and used in the control loop.

Traditionally image-based systems have been regarded to posses a good robustness to calibration errors [2], [3], even in the absence of the object and workspace model. However image-based VS schemes also exhibit some weaknesses such as singularities in the visual Jacobian which may lead to conflicts in the control loop. Other major drawback resides in the fact that an image-based system does not control the robot's end-effector in the Cartesian space, sometimes resulting in complicated or even unrealistic joint configurations being demanded to the robot. New IBVS schemes have been proposed to avoid these problems. Some of the new schemes combine 2-D and 3-D information and pose estimation to create a more complete visual servoing algorithm, for instance the 2-1/2 Visual servoing [4]. However these servoing schemes often require the description of object or its visual features. Usually 2-D visual servoing schemes can not cope with objects which are not acceptably described which limits the applicability of servoing systems to those cases where the object of interest either is difficult to characterize or an exact description is not available due to alterations of the original model as a result of damages or accidents.

Notice as well that many VS schemes assume no significant mechanical problems, low latency and high resolution in the driven robot, which is not always the case. This occurs specially in old, low-cost 6-DOF manipulators which normally exhibit limited repeatability, low accuracy and sometimes high mechanical backlash. In this paper, a low-profile anthropomorphic planar manipulator is equipped with a commercial CCD camera attached to its end-effector. An IBVS algorithm is used to achieve low-speed tracking of a moving object, which is model-free in the sense that no object model is provided [5]. Like other tracking problems, the aim is to keep the camera in a plane parallel to the tracked object. The objective is to investigate the use of RL algorithms to increase the performance of the visual tracking task. The heuristics is incorporated into the system by designing an actor-critic system [6] which evaluates the performance in the image space, i.e. directly considering the location of the visual features in the image with respect to their target locations.

The classical RL actor-critic scheme performs on-line adjustment of the parameters driving the linear trajectory regulator. By using the adaptive heuristic critic (AHC) approach [7], the training aims to achieve real-time improvement and adaptation without losing an acceptable regulation of the visual servoing task. The RL system learns directly from data in the image space and the robot current state. Two feedforward networks are used, the actor directly modifies the regulator gains and the adaptive critic stores and assigns action values.

The following sections contains a description of the IBVS tracking system. Also the paper presents a brief overview of

$$J_v(s_i, Z_i, \mathbf{A}) = \begin{bmatrix} \alpha_u & \alpha_{uv} \\ 0 & \alpha_v \end{bmatrix} \cdot \begin{bmatrix} -\frac{1}{Z_i} & 0 & \frac{x_i}{Z_i} & x_i y_i & -(1+x_i^2) & y_i \\ 0 & -\frac{1}{Z_i} & -\frac{y_i}{Z_i} & (1+y_i^2) & -x_i y_i & -x_i \end{bmatrix} \quad (1)$$

the RL schemes to introduce a detailed description of the learning algorithm used in the experiments. Special emphasis is given to the way the learning environment is set up and the parameters participating in the on-line adaptive calibration. The results of the simulation are compared to performance graphs of the system working with no heuristic adjustment. The exposition finalizes by discussing some guidelines for future development.

II. Modelling the IBVS Experiment

One of the objectives is to use a commercial off-the-shelf low-profile CCD camera as visual sensor. In our setup, one Logitech Quick Cam Pro 4000 CCD camera is attached to the last actuator of the TQ MA2000 robot. This commercial camera can provide high-quality and low-noise images at low range of 30fps. The intrinsic and extrinsic camera parameters were experimentally determined by using the camera model and calibration method created by Heikkilä and Silven in [8]. The camera model is compacted into the $\mathbf{C}$ matrix as follows

$$\mathbf{C} = \begin{bmatrix} f_u & \alpha_c f_u & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 408.99 & 0 & 187.86 \\ 0 & 409.68 & 129.17 \\ 0 & 0 & 0 \end{bmatrix} \quad (2)$$

with f_u and f_v being the focal distance in pixel units whereas $(\alpha_c \cdot f_c)$, u_0 and v_0 are the camera intrinsic parameters. The matrix $\mathbf{C}$ is used to calculate the location of an image point (u_i, v_i) following the so-called camera pin-hole model, i.e. defining its metric value (x_i, y_i) in the camera coordinate frame. A smaller matrix $\mathbf{A}$ is a sub-matrix of $\mathbf{C}$ with the image centers (u_0, v_0) directly discounted from the pixel values as follows

$$\mathbf{A} = \begin{bmatrix} f_u & (\alpha_c \cdot f_u) \\ 0 & f_v \end{bmatrix} = \begin{bmatrix} \alpha_u & \alpha_{uv} \\ 0 & \alpha_v \end{bmatrix} \quad (3)$$

$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} = \begin{bmatrix} \frac{1}{\alpha_u} & -\frac{\alpha_{uv}}{\alpha_u \alpha_v} \\ 0 & \frac{1}{\alpha_u} \end{bmatrix} \cdot \begin{bmatrix} u_i - u_0 \\ v_i - v_0 \end{bmatrix} \quad (4)$$

As shown in [9], the visual Jacobian is now a function of the so-called metric pixel coordinates of each feature $s_i = [x_i\ y_i]^T$, the depth estimation Z_i and the camera matrix $\mathbf{A}$, taking the form of expression 1.
which allows an improved image-based visual servoing method as discussed in [10]. Notice that two rows like those in Equation 1 are defined for each feature in the image. The final visual Jacobian is thus conformed by stacking these rows. In a typical IBVS implementation with four tracked features, the visual Jacobian dimension is (8×6) with six corresponding to the number of the robot's DOF.

The computation of the inverse visual Jacobian is required to calculate the manipulator motion in response to changes in the image features. The velocity screw is calculated under the task function approach [11] as $\mathbf{v} = -\lambda \cdot \mathbf{J}_v^+ \cdot \mathbf{e}$ with $\mathbf{e}$ being the error vector between the current feature vector $\mathbf{f}$ and the target $\mathbf{f}^*$. The inverse Jacobian matrix $\mathbf{J}_v^+$ is calculated by the pseudo-inverse matrix. In order to facilitate the design and the integration of components of different nature, the well-known Robotic toolbox for Matlab [12] is used together with our CSC Visual Servoing toolkit [13] to create a detailed model of the visual tracking task which includes an object in motion over a circular trajectory in front of the robot's base.

III. Actor-Critic Learning Architecture

Reinforcement Learning (RL) resides in the middle of supervised and unsupervised algorithms. After receiving the system state, the learner receives a reinforcement from the the environment notifying about the usefulness of its output. The main objective is therefore to maximize this reward signal over the time. This can be achieved by trial and error training until the learner is able to discover those outputs with a maximum reward. RL matches well with a control environment [14], because the learner is able to optimize over the time and hence to minimize a given error tracking policy, for instance the mean of the sum of square errors (MSSE) in the tracking. Remarkably a RL scheme is naturally capable of dealing with system latency or delays and time constraints. Moreover a RL algorithm posses the ability to explore new states and to exploit past knowledge. So the learning algorithm should also be in charge of guiding the learning to equilibrate the operational modes of exploration and exploitation.

The main inconvenience of using RL on real-time control schemes is the necessity of giving time to the learner as to receive feedback from the environment and as to learn. This does not always is compatible to the time constraints imposed by a real-time system. The solution proposed in this paper, goes back to early efforts in RL architectures to use an actor-critic design. Basically, this scheme has two networks, one to perform the parameter adjustment, named the Actor and one to learn from rewards, known as the critic. This classic architecture, widely discussed in [6], is able to meet real-time demands because both the learner and the controller are implemented in one entity. However its main drawback is the complexity of the training algorithm which is alleviated in this paper by using a backpropagation-like procedure. The learning in the evaluation network is based on temporal differences [6], specifically in the adaptive heuristic critic (AHC) approach. The original algorithm explained in [7] had to be modify before its inclusion in the IBVS control loop. The variations are discussed below in section III-D

A. Learning System Description

Figure 1 shows a block representation of the overall system. The experimental plant is represented by the MA2000 Robot

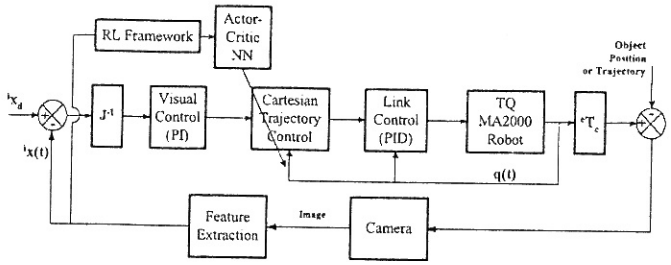


Fig. 1. Block diagram of the IBVS for tracking with RL actor-critic support.

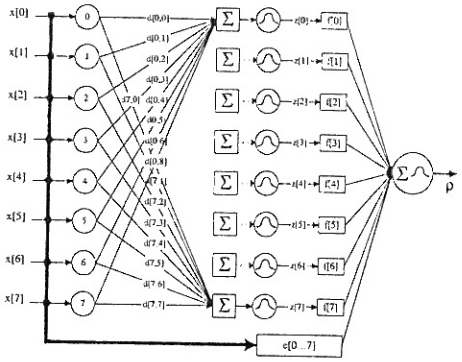


Fig. 2. The actor feed-forward neural network. Notice that the critic network is only a weighted average whereas this network includes a logarithmic sigmoid transfer function as probability output function.

model and two regulators, the trajectory generator and the robot's position controller. The internal feedback of the state is used by the position controller. The arrow crossing the trajectory generator block represents the adjustment process performed by the action neural network. The model of the camera attached to the robot and the visual processing are considered in the two blocks above the robot's block. Also the IBVS components are represented in two blocks, the visual Jacobian and the visual linear regulator which uses a classical PI algorithm. The blocks in the upper left corner represent the actor-critic agents. The critic is influenced by a composite reinforcement signal conformed by several operations as explained later in this section. The inputs to both neural networks include two normalized vectors: the robot's state vector and the image performance vector. The normalization suppresses the scaling problem derived from the differences in magnitude of the input signals. The training of the actor network is governed by the internal evaluation signal (v) provided by the critic.

The input to both networks is therefore a vector of 8 components conformed by the current normalized value of the pixel error in each direction u_e and v_e and the normalized robot state vector. The normalization scales the input values to fall between 0 and 1.

The pixel error vector is the average of the difference of each feature with respect to its correspondent desired location in the image. Given that the camera is to move in a plane parallel to the object, and that the features ideally do not move in the object, it can be assumed that this error as constant for all features. Of course, a constant value is lost mainly because of the image deformations resulting from the lenses architecture or from the motion itself. The normalized error in the image space is computed by applying the mean of the sum of square errors (MSSE) to the difference in pixels for the four features being tracked. This yields two values representing the error in each direction u, v, as follows:

$$[u_e, v_e] = \sqrt{\frac{\sum_{i=0}^{q-1}({}^i u_d(t) - {}^i u(t))^2}{q}} \cdot \eta_{\mathbf{uv}} \qquad (5)$$

with q being the total number of features. Notice the use of the normalization factor in each direction $\eta_{\mathbf{uv}} = [\eta_u, \eta_v]^T$ in order to scale the pixel error to a value between 0 and 1.

B. The actor and critic agents

Both networks, the Evaluation and the Critic network, are based on a three layer feedforward distribution which includes a direct bypass of the input vector in order to augment the mapping ability of the network (see Figure 2). This architecture, first used in experiments aiming to find optimal strategies for problem solving [7], uses the same number of hidden and input neurons. As argued in the discussion, this setup seemed to produce good results in finding successful strategies. However in the case of its application to visually-guided tasks, new dimensions of the hidden layer can also be considered in future experiments. The main feature in the nets in Figure 1 is that the critic network's output is only a weighted average whereas the actor network includes a logarithmic sigmoid transfer function at the output given the probabilistic search performed by the agent.

The output of the evaluation network -also named the critic, is the estimated action value assigned to that state. Recall again that an action-value in RL is the the probability of maximizing the total reward starting from that state. In the case of the actor agent, the network's output is put through a logarithmic sigmoidal function which results in a probability function with values between 0 and 1. This value represents the stochastic estimation of the required adjustment value. If the actor-critic scheme in this paper was to be used in solving deterministic systems with multiple actions at a time, then a simple competition between the probabilities for different actions will point out the best to follow[7]. This is not the case for the IBVS, because only one output is required, i.e. an increase or decrease in the gains of the IBVS trajectory regulator. Recall the exploitation and exploration properties of a RL scheme. A method to balance this dilemma is to apply a fair competition between different actions corresponding to increase or decrease the gains, winning the option with the higher probability of success. Any selection thus affects the gain vector in the IBVS and is sequentially evaluated by the critic which determines proper adjustments in case of undesired behavior or a positive "boost" for satisfactory responses.

C. The composite reinforcement signal

The only practical information about the performance of the learning system is the reinforcement signal. At its sim-

plest implementation, it may be only a success/failure signal. This paper uses a more informative approach which includes performance data from the image space. The idea is to match the neuro-controller response according to the evolution of the error in the image which, in the case of IBVS, drives the whole control algorithm. The average of the feature error in the image space in pixel in both directions u and v can be included in the reinforcement signal as long the current state is not of total failure. Recall that there are two total failure states. The first one occurs whenever any of the image features goes out of the visual field of view of the camera. The second failure signal is produced whenever any of the robot links goes out of valid ranges which evidently seeks to keep the robot operating in a safe region. The composite reinforcement signal should therefore guide the learning towards minimizing the error with respect to the location of the features in the image but without commanding the robot to undesired or unsafe states. So it is required a modification in the original algorithm to assign the value of the composite reinforcement signal, represented by $\hat{r}$. Its conformation depends on the particular state of the plant in a given time instant. For instance, $\hat{\mathrm{r}} = 0$ if the robot is at a start state. Also $\hat{\mathrm{r}} = r[t+1] + v[t]$, if the robot is at a failure state and finally $r[t+1] + \varphi \cdot v[t, t+1] - v[t]$, in case of a non-failure state.

At the initial point, the overall reinforcement is set to 0. If a major unsuccessful state is reached then the composite reinforcement signal is defined as the difference of -1 and the total action value provided by the critic (signal v). (see *Algorithm* 1). On the other hand, if no banned state is reached during the visual tracking then the composite reinforcement signal includes a measurement of how well the system is doing in a way to improve the performance with respect to the image plane. Thus the failure signal (represented in the table as plain r) contains the average of the overall image error $\varepsilon = -\frac{1}{2}(u_e + v_e)$ calculated from the expressions in Equation 5. This is the average of two normalized values which results in a bounded reinforcement signal. The negative sign in the expression comes from the fact that a higher error in the feature position accounts for a stronger negative reinforcement signal, whose worst case is the total failure state of -1.

D. Learning Algorithm

As a result of the new composite reinforcement signal, the original AHC learning algorithm from [7] has to be modified. The pseudo-code for the training procedure including the weight modification rules is presented in the *Algorithm* 1.

The control parameters for the learning algorithm are α and α_h for the action network and β and β_h for the critic net. The subindex h refers to hidden units within the feedforward arrangement. With $\hat{\mathrm{r}}$ assumed to be a good evaluation of the previous action, the training for the connections b and c in the critic network is updated by a simple product between β, the new reinforcement signal and the input to that level which is x_i in the case of b_i and the intermediate output y_i in case of c_i. The sign of c_i is an indication of whether the action probability should be increased or decreased.

E. Probability Competition

The training for the actor network is performed in similar fashion, but including the probability competition term $\mathrm{u(t)}$. Basically the competition encodes one of the two possible actions: either to increase or decrease the gain of the trajectory generator. So only two probability values can be assigned to the output, i.e. 1 is assigned to "increasing" action and 0 to the "decreasing" action.

Algorithm 1 Training algorithm for the actor-critic RL applied to IBVS

1: Initialize the critic and actor network with random weights.
2: **repeat**
3: Calculate critic network output $v[t]$
4: Compute action network output: control value p and the winner of the probabilistic competition p_{mod}
5: Preserve the robot state and the critic's intermediate and final output for future learning x_{old}, y_{old}, v_{old}
6: Modify the gains of the trajectory regulator.
7: Go one step into the IBVS model
8: **if** failure 1: features went out of field of view **then**
9: Update weights, reset states and re-start
10: **else if** failure 2: a link of the robot went out of the safety region **then**
11: Update weights, reset simulation states and start again
12: **end if**
13: Calculate the new critic's output $v[t+1]$
14: Define the composite reinforcement signal $\hat{\mathrm{r}}$
15: Learning by updating the weights in both networks.

Critic Network:

$$b_i[t+1] = b_i[t] + \beta\hat{r}[t+1]x_i[t]$$
$$c_i[t+1] = c_i[t] + \beta\hat{r}[t+1]y_i[t+1]$$
$$a_{ij}[t+1] = \ldots$$
$$a_{ij}[t] + \beta_h\hat{r}[t+1]y_i[t+1](1-y_i[t+1])\ sgn(c_i[t])x_j[t]$$

Actor Network:

$$e_i[t+1] = e_i[t] + \alpha\hat{r}[t+1]\mathbf{u_i}[t+1]x_i[t]$$
$$f_i[t+1] = f_i[t] + \alpha\hat{r}[t+1]\mathbf{u_i}[t+1]z_i[t+1]$$
$$d_{ij}[t+1] = \ldots$$
$$d_{ij}[t] + \alpha_h\hat{r}[t+1]z_i[t](1-z_i[t])\ sgn(f_i[t])x_j[t]$$

16: **until** epoch number is maximum.

So the difference between this value (identified as $q(t)$) and the output of the action network ($p(t)$) is a measure of the difference between the action (increase or decrease) and the action indicated by the current values in the network. It is easy to see that the current output of the actor network p is the expected probability of the output being 1, i.e. $\mathbf{P[q(t) = 1]} = p(t)$. Recall that an RL algorithm should provide exploration properties but under regulation with respect to the exploitation [6]. So a random probability source is used to compete with the output of the action network. The winner value is thus accompanied by a modification factor $u(t)$ equivalent to

the difference $u(t) = q(t) - p(t)$ which results in assigning the increasing action to $u(t) = 1 - p(t)$ and the decreasing action to $u(t) = 0 - p(t) = -p(t)$.

IV. Simulated and Real-time Experiment

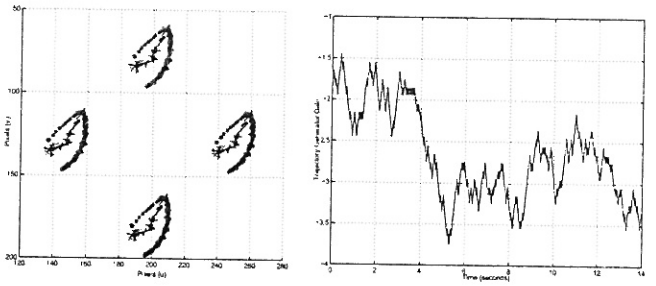
In the simulation the object is moving on a circular trajectory with $r = 0.44\ mts$ and a constant angular velocity of $\omega = 0.112$. So the target, a small replica train, is able to ride the arc of 45 degrees in approximately 13.99 seconds. Although different sample times are involved in a IBVS scheme [2], the simulation employs the image processing sample time of 0.033 msec which practically limits the overall real-time bandwidth to fall between 1.5 and 7.5 Hz.

In the experiment, one epoch lasts until a failure signal appears or the train has reached the end of the 45° arc. Given the exploratory behavior of the RL scheme, failure signals are normally expected at early training stages, forcing the system to abort the run and to end the epoch. In that case, the state variables of the robot and target object must be reset and the simulation re-starts with the robot manipulator at the initial position and the object within the camera's field of view. Notice however that the network weights are updated and stored according to the last evaluation signal which includes the failure state. The training is conducted for 10 epochs at most. If the system is able to complete 3 epochs with no failure states, the learning phase is considered complete and the network weights are saved to be applied to the system during the operational phase.

The parameter values for the learning algorithm used in the simulation are $\alpha_h = 0.2$ and $\alpha = 1.0$ for the action network. $\beta_h = 0.05$ and $\beta = 0.2$ for the critic net. By using these values, successful learning was accomplished in most cases. On the other hand, the temporal difference gain in the composite reinforcement constructor is $\varphi = 0.9$. This greedy value speeds up the convergence process and matches with other RL experiments reported in the literature. The initial gain vector for the trajectory generator has -1.75 for each of its six components, which is a low value with respect to the range of valid gains, experimentally determined to fall between -1.0 and -5.0. More details about the design and simulation of the IBVS system can be found in [15].

A. Real-time Experimental Setup

After the initial design and training stage, the IBVS system with the support from the RL neurocontroller was tested on the MA2000 robotic experimental setup presented in [13]. The performance is analyzed through the step-response of the VS tracking system. Initially the robot and the camera are located over the moving object following the teaching-by-showing technique. The object is a small replica train. The step response experiment begins with the train moving from its stationary position at a constant velocity of 3.65 cm/sec, following a straight trajectory. As the robot starts tracking, the object stops after 2 seconds allowing the robot to completely catch the object as its features return to the target location in the image space.



(a) (b)

Fig. 3. Simulation of the IBVS with RL support to adjust the trajectory generator gain. Graph (a) presents the trajectory of the features in the image plane with respect to their target location. It includes the trajectory obtained from the nominal system (darker asterisk points). Graph (b) shows the evolution of the trajectory generator's gain as modified by the action neural network, starting from -1.75

The trajectory regulator was initialized with $K = -2.0$, instead of the $K = -1.75$ used in the simulation, only to allow a better comparison with previous performance results from the nominal system. The gains in the PI visual regulator are $K_p = 2.0$ and $K_i = 0.1$, solving the joint space by means of a rate controlled integrator.

V. Results and Discussion

The complete system was simulated at least ten times with random initial values in the weights of both networks. The simulation results in this paper correspond to the third run of the system. After four epochs, the neuro-controller (the actor feedforward network) is able to increase the quality of the tracking task. Figure 3(a) shows the trajectory of the features in the image plane for the nominal linear system represented by the asterisks and for the RL-supported system represented with smaller dots. It is evident that under the neural influence, the error between the target position and the visual features has been reduced as the object moves. Moreover the usual circle-like trajectory of the features in the image [10] is changed as a result of the heuristic contribution of the RL agent. Figure 3(b) shows the evolution of the trajectory generator gain which is directly modified by the action network.

In practical experiments, the overall performance can be simply evaluated by comparing the trajectory of the features in the image space using the nominal controller and the RL-supported control structure. The response from the nominal controller is shown in Figure 4 whereas Figure 5 shows the response of the RL-supported control structure.

Although both controllers are able to perform the task, the overall error in the image space using the linear controller is higher. The improvement price however is that the smoothness in the trajectory of the features in the image space is lost while the neuro-control is used. This rough behavior results from the heuristic contribution of the RL neuro-controller which modifies the trend in IBVS schemes to follow straight trajectories in the image space. However it was observed during real-time tests that this roughness had no major affect in the tracking task nor in the image processing which is

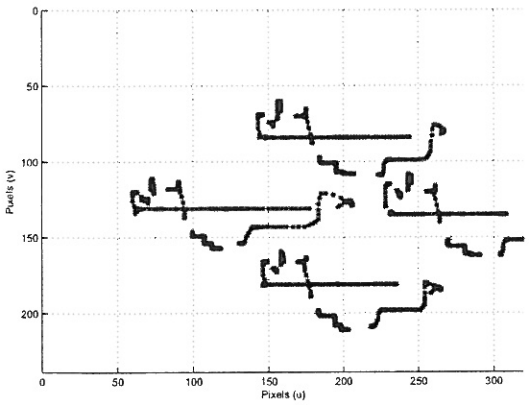


Fig. 4. Real-time trajectory of the features in the image plane with respect to their target location in the step response experiment using linear controllers within the IBVS control structure.

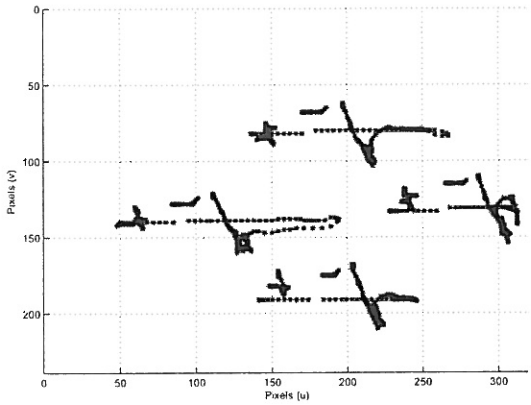


Fig. 5. Real-time trajectory of the features in the image plane with respect to their target location in the step response experiment using the RL-supported control structure.

usually disturbed by non-smooth movements on the robot's last actuator. Fortunately the robustness and high noise tolerance of the visual processing algorithm can effectively handle more radical movements of the robot's end-effector.

Figure 5 exhibits a more effective tracking with the camera attached to the end-effector following the straight trajectory in its way back to the target position of the features. However, as a consequence of the high initial visual error and the gain increment which follows, a considerable jump can be observed in the approaching zone of the step-response. This is later compensated in the final approach stage.

Notice that in simulation it is possible to determine the lowest and maximum value of the trajectory generator gain with an acceptable regulation. The neuro-controller contribution can therefore be fixed to fall at this range in order to avoid stability problems derived from the on-line gain adjustment. This is only a practical solution but further analysis must be conducted to define a set of valid stability criteria to assure robustness of the overall control structure.

New research ideas have suggested the construction of a new RL framework to include the analysis of the rate of change of the visual features on the image plane in order to generate new gain values to accordingly compensate for high oscillatory or striking trajectories in the visual features. A second idea is to decouple the gains in the trajectory generator for stronger robotic links like those on the robot's base and for other lighter actuators like those usually conforming the robot's end-effector. So a different gain set can be applied to less robust links and to those links demanding higher energy consumption. This may improve the adjustment policy applied by the actor network.

ACKNOWLEDGEMENTS

The authors would like to thank Dr. Martin Brown for his insightful advice and suggestions to this paper. Also many thanks to TQ Equipment LTD for providing us with the robot used in the real-time experiments. This research is supported by the Mexican National Council of Science and Technology, CONACYT, under the grant 72164.

REFERENCES

[1] S. Hutchinson, G. Hager, and P. Corke, "A tutorial on visual servor control," *IEEE Transactions on Robotics and Automation*, vol. 12, pp. 651–670, October 1996.

[2] P. Corke and M. Good, "Dynamic effects in visual closed-loop systems," *IEEE Transactions on Robotics and Automation*, vol. 12, pp. 671–683, October 1996.

[3] F. Chaumette, "Potential problems of stability and convergence in image-based and position-based visual servoing," *In the Confluence of Vision and Control, D. Kriegman, G. Hager, A. Morse*, vol. 237, pp. 66–78, 1998. Springer-Verlag.

[4] E. Malis, F. Chaumette, and S. Boudet, "2D-1/2 visual servoing," *IEEE Transactions on Robotics and Automation*, vol. 15, pp. 238–250, April 1999.

[5] C. Collewet and F. Chaumette, "Positioning a camera with respect to planar objects of unknown shape by coupling 2-d visual servoing and 3-d estimations," *IEEE Transactions on Robotics and Automation*, vol. 18, pp. 322–333, June 2002.

[6] R. S. Sutton and A. G. Barto, *Reinforcement Learning, An Introduction*. MIT Press, 1998. ISBN 0262193981.

[7] C. Anderson, "Strategy learning with multilayer connectionist representations," *Proceedings of the Fourth International Workshop on Machine Learning*, pp. 103–114, 1987. Also in GTE Laboratories Technical Report TR87-509.3.

[8] J. Heikkilä and O. Silvén, "A four-step camera calibration procedure with implicit image correction," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1106–1112, 1997.

[9] F. Malis, *Contributions à la Modélisation et à la Commande en Asservissement Visuel*. PhD thesis, Université de Rennes, France, November 1998.

[10] E. Cervera and P. Martinet, "Combining pixel and depth information in image-based visual servoing," *ICAR 1999, Tokio Japan*, pp. 445– 450, 1999.

[11] C. Samson, M. Le Borgne, and E. B., *Robot Control, a task function approach*. Oxford Engineering Series, 22, Oxford Science Publications, 1990. ISBN 0198538057.

[12] P. Corke, "A robotics toolbox for MATLAB," *IEEE Robotics and Automation Magazine*, vol. 3, pp. 24–32, March 1996.

[13] M. Perez-Cisneros and P. Cook, "Open platform for real-time robotic visual servoing," The 10th IASTED International Conference on Robotics and Applications (RA 2004), Honolulu, Hawaii, USA (Accepted for publication), August 2004.

[14] R. Sutton, A. Barto, and R. Williams, "Reinforcement learning is direct adaptive optimal control," *Proceedings of the American Control Conference*, June 1991.

[15] M. Perez-Cisneros and P. Cook, "Using an adaptive critic element to improve a 6-DOF visual servoing scheme," *Proceedings of the International Conference on Computational Intelligence, CIMCA'2004 Gold Coast, Australia*, July 2004.