A Testbed for Robotic Visual Servoing and Catching of Moving Objects

Dinis G. Fernandes* dinis@isr.ist.utl.pt
Pedro U. Lima pal@isr.ist.utl.pt
Instituto de Sistemas e Robótica
Instituto Superior Técnico
Av. Rovisco Pais,1 - 1096 Lisboa Codex - PORTUGAL

Abstract

This paper describes an experimental testbed, suitable for experiments on visual servoing and catching of moving objects by a robotic manipulator, which allows the implementation of different control architectures. This testbed is composed by: (1) a vision system responsible for object following and for the real-time computation of the parameters characterizing the object motion, (2) a system devoted to the prediction of the object motion, (3) a manipulator control system suitable for the object interception and catching tasks. An application is presented, consisting on the catching, by a Puma 560 manipulator, of ping-pong balls rolling on a table.

1 Introduction

Robotic vision and manipulator based servoing, catching and manipulation of moving objects are dynamic tasks of current interest for industrial applications.

Despite the considerable amount of work in the area of visual control of robotic manipulators, little prior work directly adresses catching of moving objects. The research published under this topic can be classified according to the type of visual servoing architecture used: position-based control architectures (e.g., [3][5]), where the control error is defined according to end-effector and object positions in world coordinates, and image-based control architectures (e.g., [4][10]), where the control error is directly computed in terms of image features.

Most of the work describes successful case studies, without much emphasis on the performance evaluation of the architectures used. At ISR/IST, we developed a testbed for the implementation of case studies, aimed at comparatively studying the performance of different robotic visual servoing and catching architectures. In this paper, we present the results of implementing on this testbed a modified version of a case study described by Andersson [1]. The experiment consists on the catching, by a Puma 560 manipulator, of ping-pong balls rolling on a table.

*Supported by the PRAXIS XXI grant BM/6838/95.

2 Object Trajectory Prediction

Thresholding and centroid computation [6] are used by the vision system to obtain successive locations of the ball in the image, given by the following image feature parameter vector:

$$\mathbf{f} = [u_{ball}\ v_{ball}]^T, \tag{1}$$

where u_{ball} and v_{ball} are respectively the image column and row coordinates of the center of the ping-pong ball.

The ball trajectory in image plane coordinates can be described by the following generic equation:

$$v_{ball}(i) = r(u_{ball}(i)) + e(i), \tag{2}$$

where r is a function that models the ball motion, and e is a residue which represents the observation noise and non-modeled dynamics. The function r can be expressed by the linear combination of m basis functions which model the ball trajectory:

$$r(u_{ball}(i)) = \mathbf{K}^T \varphi(u_{ball}(i)), \tag{3}$$

where $\mathbf{K} = [k_1\ \cdots\ k_m]^T$ is the parameter vector, $\varphi(u_{ball}(i)) = [r_1(u_{ball}(i))\ \cdots\ r_m(u_{ball}(i))]^T$ is the data ball vector, and i is the sampling index.

The basis functions choosen to model the ball motion are $\varphi(u_{ball}(i)) = [1\ u_{ball}(i)\ u^2_{ball}(i)]^T$. $u_{ball}(i)$ provides a linear contribution, which is the main representative of the ball trajectory. Most of the times the ball describes a slight curve, normally due to an initial spin and to some imperfections on the table, namely the table being wrapped and not being leveled. This is modeled by the quadratic function.

The problem consists on the determination of estimates $\hat{\mathbf{K}} = [\hat{k}_1, \hat{k}_2, \hat{k}_3]^T$ of the parameters k_1, k_2, k_3, each time a new image is acquired and a new ball

position is extracted. These estimates are computed using a least squares approach [2], from the set $\{[u_{ball}\ v_{ball}]_1^T, \ldots, [u_{ball}\ v_{ball}]_n^T\}$, discrete in time, of image feature parameter vectors associated to each of the n images.

This way, after n images, and consequently n successive ball locations on the image plane, a least squares estimate $\hat{\mathbf{K}}$ that minimizes the function

$$J(\hat{\mathbf{K}}) = \frac{1}{2}\sum_{i=1}^{n}[v_{ball}(i) - \hat{v}_{ball}(i)]^2, \qquad (4)$$

is computed, where

$$\hat{v}_{ball}(i) = \hat{\mathbf{K}}^T\varphi(u_{ball}(i)). \qquad (5)$$

Defining $\mathbf{\Phi} = [\varphi^T(u_{ball}(1)) \ \cdots \ \varphi^T(u_{ball}(n))]^T$ and $\mathbf{V} = [v_{ball}(1) \ \cdots \ v_{ball}(n)]^T$, if the matrix $\mathbf{\Phi}^T\mathbf{\Phi}$ is nonsingular, the least squares estimate is unique and given by:

$$\hat{\mathbf{K}} = (\mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{\Phi}^T\mathbf{V}. \qquad (6)$$

The catch is always performed along a straight line near the rear end of the table, called intercept line. The intersection of the predicted ball trajectory with the intercept line determines the catch point, given in image plane coordinates as:

$$\mathbf{f}_{catch} = [u_{catch}\ v_{catch}]^T. \qquad (7)$$

3 Visual Servoing Architectures

Two distinct visual servoing architectures were implemented:

- position-based, with endpoint open-loop (EOL) — the system only observes the object;
- image-based, with endpoint closed-loop (ECL) — the system observes both the object and the manipulator end-effector.

3.1 Position-Based

Fig. 1 shows the EOL architecture implemented. It uses a joint space PD controller with gravity compensation [9], to control the manipulator.

In the sequel, we assume the robot base frame as the reference frame. In this particular task, the desired end-effector orientation $[\alpha_d\ \beta_d\ \gamma_d]^T$ (where α_d, β_d, γ_d are ZYX Euler angles [9]) and position Z_d coordinate are constant, because the catch is always performed along an intercept line on a plane parallel to the XY plane of the robot base frame.

The vector $\mathbf{x}_{id} = [X_d\ Y_d]^T$, with the desired coordinates in X and Y, is obtained from the image predicted catch point $\mathbf{f}_{catch} = [u_{catch}\ v_{catch}]^T$ by the camera inverse calibration relationship [6]. We also

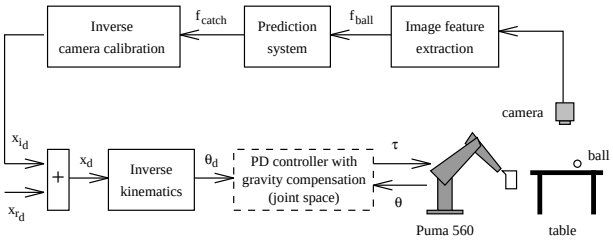

Figure 1: Position-based architecture (EOL).

define the constant vector $\mathbf{x}_{rd} = [Z_d\ \alpha_d\ \beta_d\ \gamma_d]^T$ with the desired orientation and Z coordinate. The desired pose for the manipulator end-effector, $\mathbf{x}_d$, is given by $\mathbf{x}_d = [\mathbf{x}_{id}^T\ \mathbf{x}_{rd}^T]^T$.

Finally, the inverse kinematics of the manipulator is used to compute the vector θ_d of desired joint values from the desired pose $\mathbf{x}_d$.

3.2 Image-Based

The image-based architecture block diagram is depicted in Fig. 2.

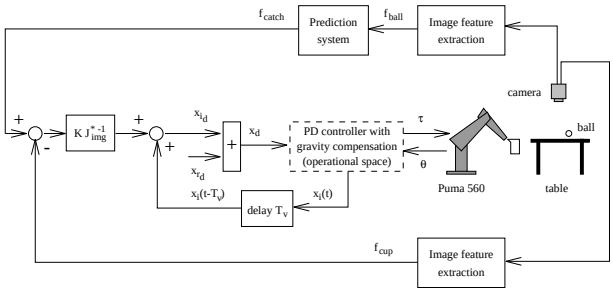

Figure 2: Image-based architecture (ECL).

The error of the external control loop is defined on the image feature space and is given by:

$$\mathbf{\Delta f} = \mathbf{f}_{catch} - \mathbf{f}_{cup}, \qquad (8)$$

where $\mathbf{f}_{catch}$ and $\mathbf{f}_{cup}$ are, respectively, the image plane coordinates of the predicted catch point and of the cup gripped by the Puma 560.

It is necessary to relate changes in the image feature space, where the error is defined, to changes in the manipulator pose. This relationship is given by the image Jacobian [7], which describes how image feature parameters change with respect to a change in the manipulator pose:

$$\mathbf{df} = \mathbf{J}_{img}\mathbf{dx}, \qquad (9)$$

where:

$$\mathbf{J}_{img} = \left[\ \frac{\partial \mathbf{f}}{\partial \mathbf{x}}\ \right]. \qquad (10)$$

Since the catch is always performed along the intercept line, where the end-effector orientation and Z coordinate are intended to be constant, it can be

assumed that $\mathrm{d}Z = \mathrm{d}\alpha = \mathrm{d}\beta = \mathrm{d}\gamma \approx 0$. This way, from the change in image feature parameters we only need to determine the corresponding changes in X and Y coordinates of the manipulator end-effector. Hence Eq. (9) simplifies to:

$$\mathbf{df} = \mathbf{J}^{*}_{img}\mathbf{dx}_i, \tag{11}$$

with $\mathbf{dx}_i = [\mathrm{d}X\ \mathrm{d}Y]^T$ and

$$\mathbf{J}^{*}_{img} = \begin{bmatrix} \frac{\partial u(X,Y)}{\partial X} & \frac{\partial u(X,Y)}{\partial Y} \\ \frac{\partial v(X,Y)}{\partial X} & \frac{\partial v(X,Y)}{\partial Y} \end{bmatrix}, \tag{12}$$

where $u(X,Y)$ and $v(X,Y)$ are obtained from the camera model.

The following control law was used in the external control loop:

$$\mathbf{\Delta x}_i = \mathbf{K}_{img}\mathbf{J}^{*-1}_{img}\mathbf{\Delta f}, \tag{13}$$

where $\mathbf{\Delta x}_i = [\Delta X\ \Delta Y]^T$ is the desired change in X and Y manipulator end-effector coordinates, and $\mathbf{K}_{img}$ is a constant gain matrix which allows to compensate for errors in camera parameters.

The vector $\mathbf{x}_{id}$ will now be given by:

$$\mathbf{x}_{id}(t) = \mathbf{x}_i(t - T_v) + \mathbf{\Delta x}_i(t - T_v), \tag{14}$$

where $\mathbf{x}_i(t - T_v)$ is the end-effector position at the time when the image was acquired. T_v is the average image sampling rate [1].

Manipulator control is performed by an operational space PD controller with gravity compensation.

4 Experimental Results

4.1 Experimental Setup

The vision system consists of an optical RAM camera Electrim EDC-1000L, the corresponding ISA bus interface card, and a PC Pentium 133 MHz where all the image processing and feature extraction algorithms run. The images have a resolution of 242x753 with a 256 levels grayscale. The sampling time is not constant, but under regular light conditions, and acquiring just half of the lines, is about 104.5 miliseconds, corresponding to 9.56 images per second.

The ball trajectory and catch point prediction algorithms, run in the same PC.

The manipulation system includes a Puma 560 manipulator, a PC Pentium 133 MHz and two Trident Robotics cards — TRC-004 and TRC-006 — which allow, from the PC, reading the robot encoders and potentiometers, and apply joint torques. The robot controller has a sampling time of 2.5 miliseconds, corresponding to a 400 Hz frequency.

These three systems are based on an open control architecture for a Puma 560 robot, recently developed at ISR/IST [8]. The two PC communicate via TCP/IP on a local ethernet, and both use MS-DOS.

In the configuration used, the table surface, where the balls roll, 1.20 meter long and 0.80 meter wide. The camera is located approximately 1.435 meter above the table surface, and points down, vertically.

4.2 Experimental methodology

The catching task consists of an individual throwing ping-pong balls, with 4 cm diameter, in the direction of the manipulator. The balls are rolled from the side of the table opposite to the robot towards the side of the table within the manipulator workspace.

Each 104.5 miliseconds a new image of the table with the ball rolling is acquired by the vision subsystem. This leads to an iterative update of the ball catch point prediction. In parallel, according to the hardware architecture used, the robot control system makes the manipulator end-effector move to the latest predicted catching point. A cup with 8 cm diameter is gripped by the manipulator. The catching is successful when the ball falls into the cup.

4.3 Prediction results

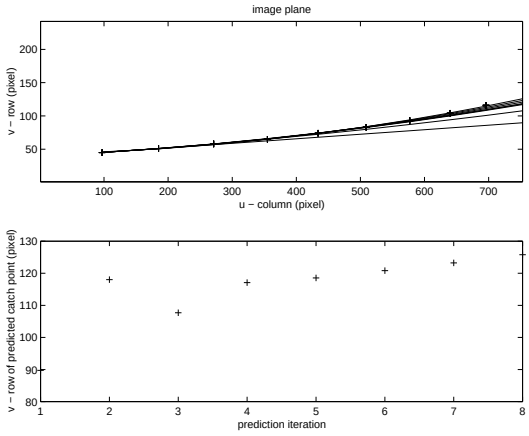

Figure 3: Prediction system results: (a) sequence (bottom-up direction) of predicted ball trajectories; (b) sucessive predicted catch point v coordinate.

Prediction results for a ball toss are presented in Fig. 3. The first predicted trajectory is a straight line, because it is determined from the two first ball locations only. The estimated parameters were $\hat{\mathbf{K}} = [38.386\ 0.068\ 0.0]^T$, with an associated quadratic error (measured for all the nine known ball locations at the end of the trajectory) given by $J(\hat{\mathbf{K}}) = 883.693$. The parameters $\hat{\mathbf{K}} = [42.997\ 0.018\ 0.000122]^T$ estimated for the last trajectory prediction clearly show

[1] Our vision system does not allow a fixed image sampling time, but rather an assyncronous image acquisition, at a rate depending on the image acquisition and processing times.

the need for a quadratic basis function to model the curvilinear path. This last predicted trajectory has a smaller associated quadratic error, $J(\hat{\mathbf{K}}) = 3.103$.

4.4 Catching results

Figs. 4 and 5 show plots of the end-effector X and Y coordinates versus time for each of the architectures implemented. The dashed line is the position set-point for the manipulator end-effector and the solid line represents the actual trajectory.

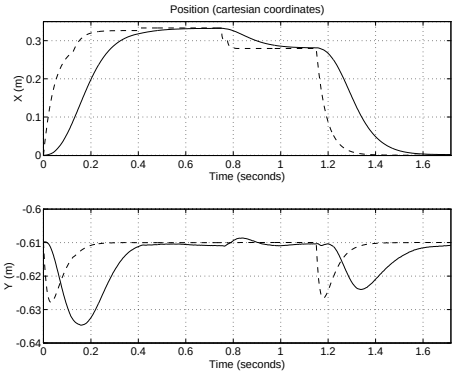

Figure 4: Results for the position-based architecture (EOL).

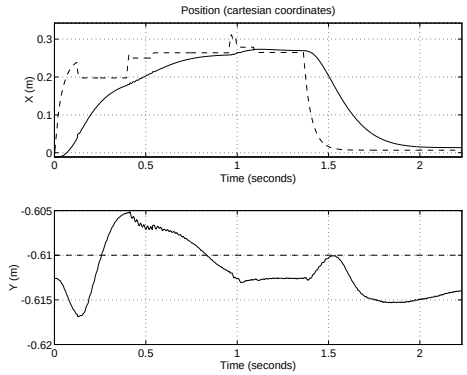

Figure 5: Results for the image-based architecture (ECL).

The shortest settling time is obtained for the position-based architecture. The steady-state error is also minimized under this architecture, making it well suited for catching of faster balls.

The high sensitivity of the image-based controller to noisy data from the vision system, causes "nervous" driving torques, leading to a shaky end-effector and lower catch rate. Another reason for this is the fact that the average image sampling rate, T_v, is not constant, leading to inaccuracies in the computation of $\mathbf{x}_{id}$ using (14).

5 Conclusions and Future Work

The image-based architecture eliminates camera calibration errors while the ECL scheme reduces the end-effector positioning errors due to errors on the kinematic relationship with the camera. Nevertheless, the ECL approach is very sensitive to noisy data from the vision system and creates field-of-view constraints, since the end-effector must always be in the image. Furthermore, due to the increase in image processing time, significant delays are introduced in the control loop, leading to oscillations in control signals and vibrations in the manipulator end-effector. Therefore, the image-based architecture is the one with the worse successful catches rate, since the cup is shaky while catch is being attempted (e.g., check Fig. 5 around t=0.5 s). The best success rate was obtained for the position-based architecture.

In the short term, we plan to switch to a faster vision system, including the possibility of image acquisition at a fixed rate. This will also let us study the dynamics of the complete control loop, improving the design of control parameters. Long-term work will include the catch of moving objects in 3-D under a less structured environment.

References

[1] R. Andersson. Real-time gray-scale video processing using a moment-generating chip. *IEEE Journal of Robotics and Automation*, RA-1(2), June 1985.

[2] K. Åström and B. Wittenmark. *Computer Controlled Systems*. Prentice-Hall, 1984.

[3] G. Buttazzo, B. Allotta, and F. Fanizza. Mousebuster: a robot for real-time catching. *IEEE Control Systems Magazine*, 14(1), February 1994.

[4] B. Ghosh, T. Tarn, N. Xi, Z. Yu, and D. Xiau. Calibration free visually controlled manipulation of parts in a robotic manufacturing workcell. In *Proc. of the IEEE International Conference on Robotics and Automation*, pages 3197–3202, April 1996.

[5] W. Hong and J. Slotine. Experiments in hand-eye coordination using active vision. In *Experimental Robotics IV, Springer-Verlag, Proc. of ISER 95*, Stanford, CA, July 1995.

[6] B. Horn. *Robot Vision*. MIT Press, McGraw-Hill, 1986.

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

[8] N. Moreira, P. Alvito, and P. Lima. First steps towards an open control architecture for a puma 560. In *Proc. of the Portuguese Conference on Automatic Control, CONTROLO 96*, Porto - PORTUGAL, September 1996.

[9] L. Sciavicco and B. Siciliano. *Modeling and Control of Robot Manipulators*. McGraw-Hill, 1996.

[10] C. Smith and N. Papanikolopoulos. Vision-guided robotic grasping: Issues and experiments. In *Proc. of the IEEE International Conference on Robotics and Automation*, pages 3203–3208, April 1996.