**CONTROLO 2008**
**8th Portuguese Conference on Automatic Control**

# PREDICTIVE REACHING CONTROL WITH MULTIPLE MOTION MODELS

**Paulo Carreiras, Alexandre Bernardino, José Santos-Victor**

*Institute for Systems and Robotics, Instituto Superior Técnico, Lisboa, Portugal.*

Abstract: In this work we address the problem of controlling the arm of a humanoid robot to reach for moving objects. 3D target's trajectory is measured by the robot's active stereo head with color based segmentation and tracking methods. Future positions of the target are predicted, at an appropriate time horizon, by fusing the information from multiple motion model estimators, including constant velocity, acceleration, circular and periodic motions. The arm positioning system is controlled by setting its reference to the target's position at the prediction horizon, to cope with the arm slow dynamics. Experimental results show that, compared to a non-predictive approach, the proposed method reduces the average tracking error in about 50%.

Keywords: 3D prediction, Kalman Filters, Multiple Models Adaptive Estimation.

## 1. INTRODUCTION

A common task in robotics manipulation is to position an end-effector in the vicinity of a target. Whereas most authors address this issue considering static objects only, in this work we aim at controlling the robot arm with respect to moving objects. Due to the slow dynamics of robot manipulators, as well as latencies in the control and perception systems, a predictive control strategy must be employed to minimize the positioning error. A suitable choice of the prediction horizon is done in run-time to ensure that the robot is able to reach the target independently of its initial conditions.

Trajectory prediction had a huge development due to military proposes. Some applications in the robotic field regarding the tracking delay can be found in the literature. In (Piepmeier, *et al*., 1998) a prediction method was proposed to control a robotic arm. A Kalman filter was used to estimate the position of the target assuming a constant velocity and zero mean Gaussian acceleration errors. Our method, instead of a single motion model, uses several models running in parallel, providing better results in complex trajectories and dealing explicitly with the slow dynamics of robotics manipulators.

The 3D position of the target is measured by a pair of cameras. A color based segmentation algorithm tracks in 2D the image region corresponding to a pre-selected object through the frames. Then, a stereo triangulation method and the robot head kinematics are employed to convert the 2D image based measurements in the 3D position of the target, in a world fixed reference frame. The position of the manipulator is controlled to intersect the target, but delay compensation has to be done in order cope with the manipulator slow dynamics. If we command the robot to the current target position, it won't reach it because the robot can't move instantaneously. We need to predict the position of the target in a future time and move the robot arm to that position instead.

Section 2 of this article describes our robot, Baltazar, some aspects of its kinematics and the 3D position measurement system. The purpose is to track a selected area of an image through time in both cameras, and process this information to get the 3D target location. The following three sections deal with the target prediction problem, starting with the mathematical models, their fusion, and finally the computation of the necessary time span to compensate the inertia effects. The last chapter presents some experimental results obtained with our robot. Two platforms were used: a virtual robot, simulated with Webots™, and the real robot available in the laboratory. Finally we present the conclusions of this paper and some directions for future work.

## 2. ROBOT SETUP

In this section we present the humanoid robot platform used in this work, describing its kinematics and visual measurement system. A picture of the robot (Baltazar) is shown in Fig. 1. It is composed by and active stereo head with 4 degrees of freedom (DOF), an anthropomorphic manipulator with 6 DOFs and a human-like hand with 4 actuated DOFs.

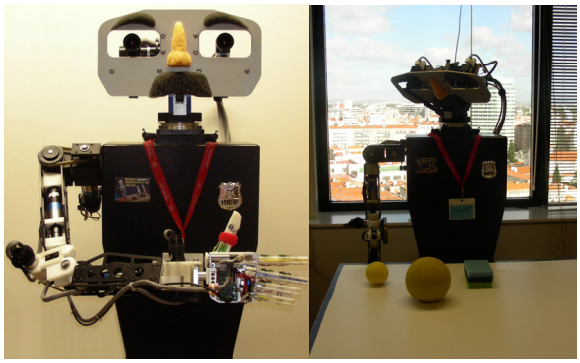

Fig. 1. The ISR/IST Humanoid Head-Torso, Baltazar, on its working scenario.

### *2.1. Kinematics*

The kinematics of Baltazar's manipulator and head are both motivated by the anatomy of humans. The arm consists in six joints, two of them associated with the shoulder, two with the elbow and the remaining joints are positioned in the wrist. The head has four joints: two for the neck pan/tilt and two for the eyes' pan. The robot's arm and head joints are represented in Fig. 2.

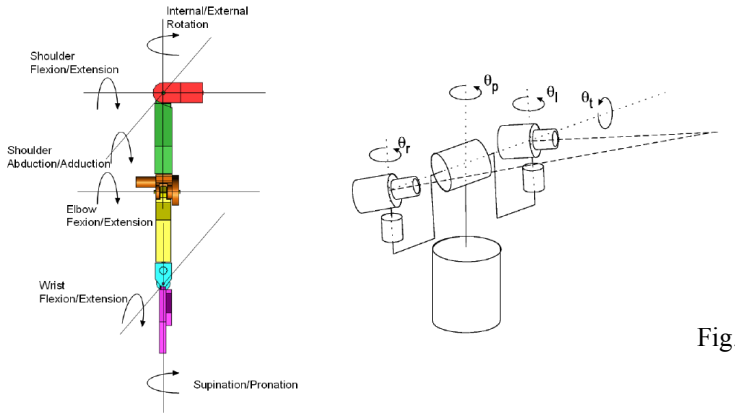


Fig. 2. Robotic Arm and Head schematics

To control the robot arm, we convert 3D coordinates corresponding to the target's future position into the joint angles required to achieve that position (inverse kinematics). A solution to compute the inverse kinematics on this robot arm is presented in (Lopes, *et al.*,2004) and will be employed in this work. To control the head we use a tracking system to keep the target in the center of the retinas (Bernardino and Santos-Victor, 1999).

### *2.2. The 3D Position Measurement System*

An active stereo head computes the 3D position of targets in the environment with respect to a fixed reference frame. In a first stage, the 2D position of the target in the two images is extracted, using a color based segmentation and tracking method. Then, using the head kinematics and a triangulation process we estimate the 3D instantaneous position of the target in a fixed reference frame.

#### *2.2.1. Image Segmentation and Tracking*

This section describes the algorithm used to follow the selected object trough the image frames. There are a plenty of approaches available in the literature but much of them assume static cameras and can't comply with the fact that cameras are moving during the robot's operation. Additionally the algorithm should be computationally efficient, because it is supposed to work in real time, feeding the prediction module with the updated 3D localization.

The *camshift* algorithm (Bradski, 1998) is able to deal with the above issues and is already implemented in open source code libraries. It is based on the color histogram of an image region. A negative aspect of this algorithm is that the region to track must be initialized manually and the objects to deal with should have sufficiently distinct colors. Anyway, due to its good real-time performance, we have adopted this algorithm to perform 2D target tracking in the stereo images. An illustrative example of the algorithm output is shown in Fig. 3.

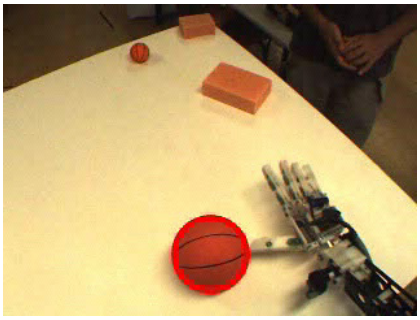

Fig. 3. Color based object segmentation with the *camshift* algorithm.

#### *2.2.2. 3D Position Measurement*

Measurement of the 3D target position is done with a stereo camera system. Since the cameras aren't static two classes of problems emerge: one that deals with the representation of coordinates in a moving axis and conversion to a fixed system, and the other that merges the information from both cameras to get 3D coordinates.

Firstly we compute the 3D coordinates of the target in the neck (moving) reference frame. A sketch of

Baltazar's head and its cameras is presented in Fig. 4.

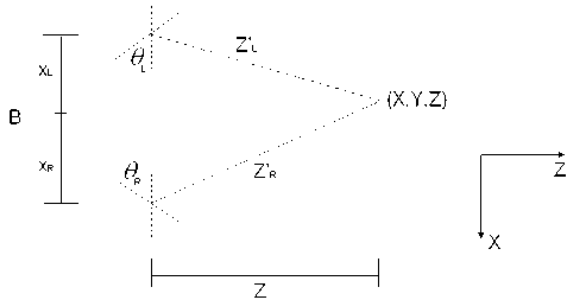


Fig. 4 – Geometry of the verging system.

It can be shown (Olson, 1993) that, using this configuration, the 3D coordinates are given by:

$$\begin{aligned} X &= -\tfrac{Z}{2}\left(tan\left(x_L/F+\theta_L\right)+tan\left(x_R/F+\theta_R\right)\right) \\ Y &= \tfrac{Zy_R}{2Fcos\theta_R}+\tfrac{Zy_L}{2Fcos\theta_L} \\ Z &= \tfrac{B}{tan(x_L/F+\theta_L)-tan(x_R/F+\theta_R)} \end{aligned} \quad (1)$$

where ($x_R$, $y_R$) and ($x_L$, $y_L$) are the 2D coordinates obtained with the image segmentation algorithm in the right and left images, respectively.

To express the coordinates of the target referenced to a static frame we just have to use the head kinematics (Bernardino and Santos-Victor, 1999).

## 3. MOTION MODELS

Targets in the real world may undergo very diverse types of trajectories. If trajectories are smooth, it is often the case that a simple constant velocity model is suffucient to locally approximate the target's trajectory. However, this may only provide good predictions at very short time horizons and for objects with large inertial mass. In our case, however, prediction time horizons may be relatively large, due to the slow response of the robotic arm. Thus, a simple constant velocity model may not be sufficient to model target's motion, unless objects slide or roll in a table with slow friction. Since we aim at being as general as possible, we must consider a much more enlarged set of possible motion models. Therefore, the development of suitable target's trajectory prediction methods, capable of dealing with a diversity of motion types, is the major task of this work. This section presents a very short description for each one of the implemented motion models.

We consider two types of motion models: linear (Gaussian) and non-linear/(non-Gaussian). The former can be optimally estimated with Kalman filters and fused with Multiple Model Adaptive Estimation (MMAE) theory. The latter cannot be easily fused with the former, but may be selected by analysing the residues of the estimations and applying a chi-square test.

### *3.1. Linear Models*

We assume the target is a particle with all mass at its center, and that measurements are taken at discrete instants of time. A linear discrete time model in a noisy environment can be written as:

$$\begin{aligned} x(k) &= Ax(k-1)+Bu(k-1)+Gw(k-1) \\ y(k) &= Cx(k-1)+Du(k-1)+Hv(k) \\ f(w) &\sim N(0,Q) \\ f(\mu) &\sim N(0,R) \end{aligned} \quad (2)$$

where the noise introduced in the model and sensor equations is assumed to be Gaussian, with zero mean and known variance. A standard tool in this setting is the Kalman filter. It consists in two steps: time update (prediction) and measure update (correction):

$$\begin{aligned} &\hat{x}^-(k) = A\hat{x}(k-1)+Bu(k-1) \\ &P^-(k) = AP(k-1)+Q \end{aligned} \quad \text{Prediction}$$

$$\begin{aligned} &K(k) = P^-(k)C^T\left(CP^-(k)C^T+R\right)^{-1} \\ &\hat{x}(k) = \hat{x}^-(k)+K(k)\left(z(k)-C\hat{x}^-(k)\right) \\ &P(k) = \left(I-K(k)C\right)P^-(k) \end{aligned} \quad \text{Update} \quad (3)$$

To implement a model using this technique six matrices should be specified: A, B, C, D, Q and R. These matrices specify the type of trajectory considered to model target's motion, the measurement noise and the model uncertainty. A detailed account of the considered linear models can be found in (Li, *et al*., 2003).

For example, the constant velocity model assumes a low power acceleration noise. The position (*p*) and speed (*v*) at the k+1 sample given the k[th] state is given by

$$\begin{aligned} p^{(i)}(k+1) &= p^{(i)}(k)+Tv^{(i)}(k)+\tfrac{T^2}{2}w^{(i)}(k) \\ v^{(i)}(k+1) &= v^{(i)}(k)+Tw^{(i)}(k) \end{aligned} \quad (4)$$

The supra-index *i* refer to i[th] coordinate of the 3D Cartesian system. Rewriting those equations using matrix notation results in:

$$\begin{aligned} x^{(i)}(k+1) &= A^{(i)}x^{(i)}(k)+G^{(i)}w^{(i)}(k) \\ z^{(i)}(k) &= C^{(i)}x^{(i)}(k)+H^{(i)}\mu^{(i)}(k) \end{aligned} \quad (5)$$

with

$$A^{(i)} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \quad G^{(i)} = \begin{bmatrix} \frac{T^2}{2} \\ T \end{bmatrix}$$

$$C^{(i)} = \begin{bmatrix} 1 & 0 \end{bmatrix} \quad H^{(i)} = [1] \quad (6)$$

The covariance noises Q and R for each one of the coordinates are

$$Q^{(i)} = cov\left(G^{(i)}w^{(i)}(k)\right) = var\ w^{(i)}(k) \times \begin{bmatrix} \frac{T^4}{4} & \frac{T^3}{2} \\ \frac{T^3}{2} & T^2 \end{bmatrix}$$

$$R^{(i)} = \sigma_i^{\,2}H^{(i)} \quad (7)$$

Since no correlation is assumed between the

coordinates, the global matrix is diagonal where each element corresponds to the sub-matrices presented above. This very simple model can deal with almost any motion given that, with an adequate time scale, all motions can be approximated by constant velocity models. However, for long predictions horizons like the ones required in our case, the inclusion of models specific for likely target motions will let us improve the performance of the tracking system. We have implemented the acceleration models (Singer Model and Wiener Process) described in (Singer, 1970), the Curvilinear and Circular motion models presented in (Li, *et al*., 2003), and a Ballistic motion model with collisions. The latter will be detailed in next section.

#### *3.1.1. Ballistic Model and Collisions*

Humanoid robots have to deal several times with Ballistic like trajectories. These are the trajectories of a non-maneuvering target subject to the acceleration of gravity. The present model is just an adaptation of the constant speed/acceleration models for the particular workspace of the Baltazar robot.

For simplification proposes, the implemented algorithm deals only with collisions in a plane defined *a priori*. The differences between this and the constant speed model are: (i) instead of white noise acceleration, the mean of the acceleration in the y axis is set to the gravity acceleration; (ii) adjusts the speed in the position where the collision is expected. The easier solution was adopted which means that the speed is set to its symmetric value. The A and Q matrices must be changed whenever a collision is imminent. The mechanism used to detect collisions is very simple and is based just in the *y* coordinate. If the *y* position gets close to the plane height, the changes are performed and immediately restored.
The modified A and Q matrix are:

$$A^{(2)} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad Q^{(2)} = \begin{bmatrix} q'_{11} & q'_{12} \\ q'_{21} & q'_{22} \end{bmatrix} \tag{8}$$

$$q'_{11} = \frac{T^4}{4} \quad q'_{11} = -\frac{T^3}{2} \quad q'_{11} = -\frac{-T^3}{2} \quad q'_{11} = T^2$$

To properly modulate the acceleration, the B matrix should also be changed, leading to:

$$B = \begin{bmatrix} 0 & 0 & -gT^2 & T & 0 & 0 \end{bmatrix}^T \tag{9}$$

### *3.2. Non-Linear Model - Periodic Motion*

The proposed approach uses the autocorrelation of each one of the coordinates to detect the presence of periodic motion. If the maximum of the autocorrelation is higher than a certain threshold a periodic trajectory is reported and the output becomes the sequence of recorded samples that exactly match a period of the trajectory. This operation is computationally heavy so some changes must be done. Instead of continuously computing the autocorrelation, a burst strategy was adopted. The measured data is stored in an appropriate buffer, and the whole buffer is processed when full. The counter part is that the update rate of this model is a function of the buffer size, which can't be arbitrary set. To correctly deal with a period of N samples a 3N window is required. If the maximum detected period is 50 samples, it requires a 150 positions buffer. The way to merge this model with the remaining ones is given in the next section.

## 4. MODEL ESTIMATION

The proposed estimator handles two different types of models. It merges all the Kalman (linear) models using the Multiple Model Adaptive Estimator (MMAE) proposed in (Maybeck, 1994) and then blends them to the periodic model with a chi-squared based hypothesis test. A block diagram of the full estimator is shown in Fig. 5.

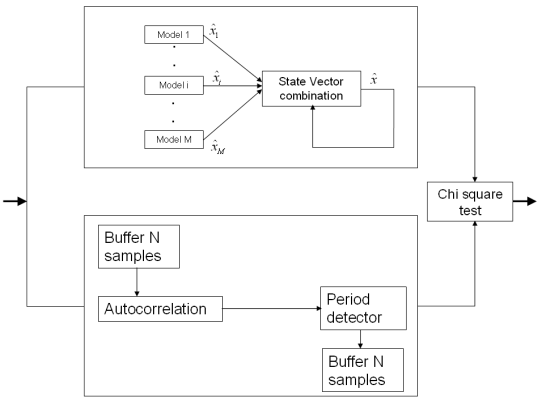


Fig. 5 – Multiple Model Estimator block diagram.

#### *4.1.1. Linear Model Fusion*

The MMAE algorithm weights the outputs produced from each of the models with the conditional model probabilities given the data. The output is thus, a weighted average of all models.

$$x_{MMAE}(k) = \sum_{j=1}^{M} p_j(k) x_m(k) \tag{10}$$

It can be shown that the residues covariance matrix of a model that fits the target's trajectory is given by

$$A_m = C_m P_m^- C_m^T + R_m \tag{11}$$

The conditional density function for model m at sample *k*, knowing all the past measures $z_{k-1}$ is

$$f_{z(k)|h,z_{k-1}}(z \mid h_m, z_{k-1}) = \beta_m e^{-2q_m(k)}$$

$$\beta_m = \frac{1}{(2\pi)^{n/2}|A_m|^{1/2}} \tag{12}$$

$$q_m(k) = r_m^T(k) A_m^{-1} r_m(k)$$

where $q_m$ is the likelihood ratio at sample $k$ and $h_m$ is the hypothesis of the $m^{th}$ model.

The conditional probability that weights each vector to produce the estimation is finally given by

$$p_m(k) = \frac{f_{z(k)|h,z_{k-1}}(z|h_m,z_{k-1})p_m(k-1)}{\sum_{j=1}^{M} f_{z(k)|h,z_{k-1}}(z|h_j,z_{k-1})p_j(k-1)} \quad (13)$$

### *4.1.2. Non-linear Model Fusion*

To mix the linear estimate with the trajectory computed by the periodic model, a simple hypothesis test assuming chi-squared distribution is applied. The hypothesis test is a statistical tool that produces a decision based on the analysis of the probability density function using only sampled data. Given this function and a required significance level, two decision areas are set, the critical region and the acceptance region. Given a sample we just have to see if it's outside or inside the critical region to reject or accept the hypothesis respectively.

The chi-square distribution is usually employed since it can fit a large variety of situations, although it is not the optimal solution in many cases. The most commonly used test statistic in the literature is the one used in this work, and is given by:

$$\epsilon(\mathrm{k}) = (z(\mathrm{k}) - \hat{z}(\mathrm{k} \mid \mathrm{k}-1))^T S(\mathrm{k})(z(\mathrm{k}) - \hat{z}(\mathrm{k} \mid \mathrm{k}-1))$$
$$S(\mathrm{k}) = cov(z(\mathrm{k}) - \hat{z}(\mathrm{k} \mid \mathrm{k}-1)) \quad (14)$$

## 5. COMPUTING THE PREDICTION HORIZON

Once a model has been identified, it is still necessary to predict the position of the target at the time the arm will intercept it. The prediction can be easily obtained by iterating the state space equations. If the model selection stage chooses the periodic model, no iterations are required since the prediction N-samples-ahead is found in the $N^{th}$ position of the buffer.

We have observed that the prediction error becomes too large if a horizon bigger than 50 samples is used. In our setup, however, the robot arm can reach any position of the workspace in no more than 20 steps.
To estimate the requested anticipation we check, for each possible time horizon between 1 and 20, if the arm is able or not to reach the predicted position from its initial position. This is done by computing the joint velocities required to go from the initial position to the predicted final position for a given time horizon. If the computed speed exceeds the joint maximum values, then a bigger time horizon will be tested. The selected prediction horizon thus minimizes the time to intercept the target, given the speed limitations in the arm joints.

This approach is sufficient to perform basic reaching, but other approaches could be devised to allow for more demanding actions like grasping or hitting. These modes would require not only to control the position of the arm to a certain set point, but also to control its velocity and acceleration at the intercept point. These modes will be subject of future work.

## 6. EXPERIMENTAL RESULTS

The proposed method was applied in the Baltazar robot. To improve the development stage, and allow testing on ideal conditions with ground truth, a simulator was also designed based in the *Webots*™ platform (Michael, 2004), see Fig. 6. All the available data from CAD models used to assemble the robot were imported to the simulator and an interface was developed to easily commute between the simulator and the real robot.

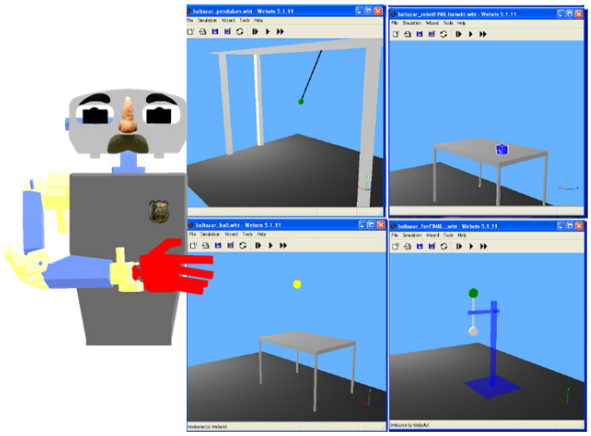

Fig. 6. – The Webots simulator permits testing our methods with several types target motions: pendulum (periodic), constant velocity, ballistic with collisions, and circular motion.

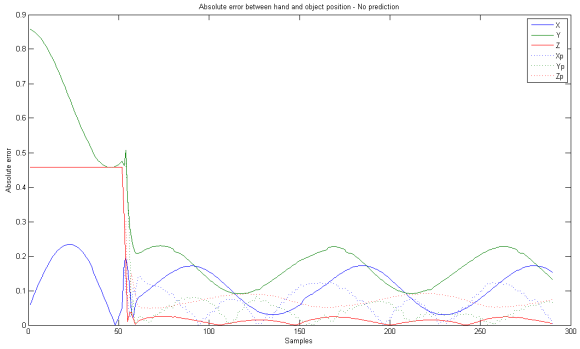


Fig. 7.– Absolute error between hand and object position. Solid lines: without prediction. Dotted lines: with prediction. The arm starts moving at time 50.

In Fig. 9 we present some simulation results on tracking a circular motion. The plots show the absolute errors between the hand and the object positions, comparing the cases with and without prediction. The prediction approach is able to achieve a 50% reduction in the absolute error, going from 0.2m to 0.1m, in average.

The experiment performed in the real robot consists

in the manual generation of an approximately circular motion (see Fig. 8). In the plots on Fig. 9 we can also observe that, despite some overshoot, the use of prediction allows to reduce significantly the lag on the hand position with respect to the object.

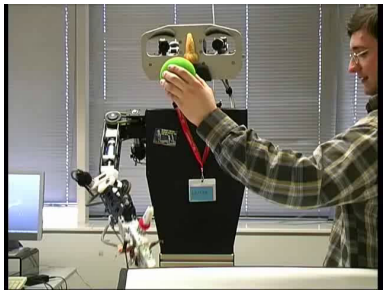

Fig. 8. – Experiments with the real robot.

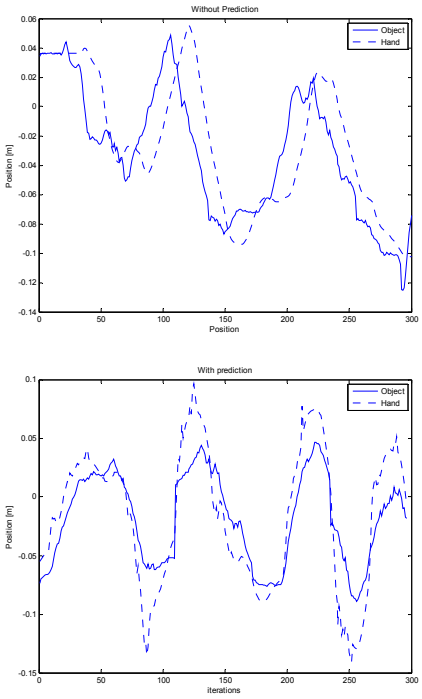


Fig. 9 – Experiments with the real robot. Comparison between the position of the object (solid lines) and the hand (dotted lines). Top: without prediction. Bottom: with prediction.

The algorithm proved it can correctly handle noisy environments allowing a standard deviation up to 12cm in each of the 3D coordinates. However, we found that the major error source wasn't due to the prediction itself but was a consequence of modeling errors involved in the head and arm kinematics.

## 7. CONCLUSION

The suggested approach effectively reduces the delay to reach a moving target. The method revealed a good performance in handling the very noise measurements acquired in the real system. The main performance limitations right now are due to modeling errors in the kinematics models. In fact there is a systematic bias on the arm positioning. This will be addressed in future work by using visual feedback on the end-effector.

The adaptive estimation using multiple filters was impressive in the sense that it has reduced the tracking error in about 50%.

The motion segmentation algorithm provided an excellent performance providing good results even with background changes. There is however space for future work, especially to deal with occlusions. The predicted position could also be useful in the presence of occlusions providing for example the search window localization and its size.

## ACKNOWLEDGMENTS

We would like to thank Luis Montesano, Manuel Lopes and Samuel Ferreira for their valuable help on this work.

This work was supported by the European commission, Project IST-004370 RobotCub, and by the Portuguese Government – Fundação para a Ciência e Tecnologia (ISR/IST plurianual funding) through the POS_Conhecimento Program that includes FEDER funds, and through project BIO-LOOK, PTDC/EEA-ACR/71032/2006.

## REFERENCES

Bernardino, A. and J. Santos-Victor, (1999)
"Binocular Visual Tracking: Integration of Perception and Control", IEEE Trans. Robotics and Automation, 15(6).

Bradski, G. R. (1998).
"Computer Vision Face Tracking For Use in a Perceptual User Interface", Intel Tech Journal, Q2

Li, X. R. and V. P. Jilkov, (2003)
"Survey of Maneuvering Target Tracking. Part I: Dynamic Models", IEEE Trans. Aerospace and Electronic Systems, 39(4): 1333-1364.

Lopes, M., R.Beira, M. Praça and J.Victor (2004)
"An Antropomorphic robot torso for imitation: design and experiments", IEEE IROS

Maybeck, P. (1994)
"Stochastic Models, Estimation and Control" vol.2, New York Academic Press

Michael, O. (2004)
"Webots: Professional Mobile Robot Simulation", International Journal of Advanced Robotic Systems, 1(1): 39-42

Olson, T.J., (1993)
"Stereopsis for Verging Systems", CVPR.

Piepmeier, J., G. McMurray and H. Lipkin, (1998)
"Tracking a Moving Target with Model Independent Visual Servoing: a predictive Estimation Approach", IEEE ICRA.

Singer, R. (1970)
"Estimating Optimal Tracking Filter Performance for Manned Maneuvering Targets", IEEE Trans. Aerospace and Electronic Systems, 6(4): 473-483