Noname manuscript No.
(will be inserted by the editor)

Expected Perception based control for reaching a moving target

Nino Cauli · Egidio Falotico · Alexandre Bernardino · José Santos-Victor · Cecilia Laschi

Received: date / Accepted: date

Abstract Expected Perception based control systems use the robotic system's internal models and interaction with the environment in order to predict the future response of their sensory inputs. By comparing the sensory predictions with the actual sensory data, the expected perception (EP) control system monitors the error between the predicted and the actual sensor observations. If the error is small the system may decide to neglect the input and skip any corrective action, thus saving computational and energy resources. If the mismatch is large the system will further process the sensor signal to compute a corrective action via feedback. So far, EP systems have been implemented for predictions based on robot's own motion. In this work, an EP system is applied to predict the dynamics and anticipate the motion of an external object. The new control system is implemented in a humanoid robot, the iCub. The robot reaches in anticipation for an object's future position, by predicting its trajectory and correcting the arm's position only when necessary. The results of the EP based controller are analysed and compared against a standard controller. The new EP based controller is less computationally demanding and more energy efficient, for a marginal loss in the tracking error.

N. Cauli · E. Falotico · C. Laschi
The BioRobotics Institute - Scuola Superiore Sant'Anna - Viale Rinaldo Piaggio, 34 - 56025 - Pontedera (PI) - Italy
E-mail: nino.cauli@sssup.it
E-mail: egidio.falotico@sssup.it
E-mail: cecilia.laschi@sssup.it

A. Bernardino · J. Santos-Victor
Instituto de Sistemas e Robotica - Instituto Superior Tecnico - Torre Norte, 7 - Av. Rovisco Pais, 1, 1049-001 Lisboa. Portugal
E-mail: alex@isr.ist.utl.pt
E-mail: jasv@isr.ist.utl.pt

Keywords Sensory prediction · Kalman filtering · Internal models · Humanoid robotics

1 Introduction

Robots are gradually becoming a part of our daily life. Robotic toys for children and robotic home cleaners are now easy to find in commodity stores and robotic companions and caretakers for elderly people may be available in a near future. Robots' working environment is no longer the strictly structured one of a manufacturing industry, but a hyper-dynamic world with multiple uncertainties.

In order to deal with a constantly changing environment, a robot (just as a human) needs to anticipate future conditions in order to properly control its own movements. Humans appear to solve this problem by predicting changes in their sensory system as a consequence of their actions (Berthoz, 2002). Predictions are obtained using internal models which represent their own bodies as well as external objects dynamics (Johansson, 1998; Miall and Wolpert, 1996; Nguyen-Tuong and Peters, 2011).

There are three main types of internal models (Miall and Wolpert, 1996): the forward models, the environment models and the inverse models. Forward models make it possible to predict the future data from past perceptions and planned actions; environment models predict the dynamics of external objects or agents; inverse models find the actions needed to obtain a desired state starting from the actual one.

A number of implementations of anticipatory sensory-motor systems based on internal models have been created so far in robotics. Examples of control systems for mobile robots that predict visual sensory data through

forward models are shown in (Gross et al, 1999; Hoffmann, 2007), while in (Bauml et al, 2011; Kim and Billard, 2012; Kober et al, 2012; Vannucci et al, 2014) the authors proposed different systems anticipating the dynamics of moving objects in order to accomplish catching and pursuit tasks.

All the previous works use internal models to simulate the future and perform actions with anticipation. Although actions are anticipated, control loops maintain a strict sequentiality between perception and action: obtaining data from the sensors, predicting future response and planning the action to perform. A problem with this approach is that long term predictions and action planning are time and resource consuming processes. As an alternative, sensory feedback can be "switched-off" if predictions are compatible with the current observations. This means that the executed actions are working as planned and no modifications to the plan are necessary. This is the main principle of Expected Perception (EP) control architectures (Datteri et al, 2003; Laschi et al, 2006, 2008; Barrera and Laschi, 2010). The main idea is to execute sensory processing and behaviour planning only if the expected sensory feedback is different from the actual one. EP controllers use a forward model to predict future sensory responses that are compared to the actual one, when it arrives. If the error is lower than a threshold the system continues with the previously planned behaviour (like a feed-forward controller), otherwise it processes the sensory data again and updates the plan.

EP architectures have been successfully implemented in different applications. (Datteri et al, 2003) first used the EP concept to visually control an 8 DOF robotic arm. With a camera on its end effector, the robot was able to predict the next camera images based on the old ones and on the arm motor commands. In (Laschi et al, 2006, 2008) the EP architecture was implemented to accomplish a grasping task. The robot had the ability to grasp an object by predicting the tactile image that would be perceived after reaching for it. The EP's most recent implementation used the predicted images to locate unexpected objects in the scene (Moutinho et al, 2011).

The previous EP based works focused mainly on sensory-motor anticipation. They used a forward model to predict the future sensory data from the past perceptions and the planned actions. The static nature of the environment was a fundamental requirement to obtain a correct Expected Perception. The sensory data prediction was based only on the robot's self movements and motions of external moving objects were recognised as prediction errors. There is still a lack of applications of EP systems that exploit the dynamics of external objects using environment models. In this work we present an EP system that anticipates the motion of an external object and show its advantages with respect to a standard control scheme in terms on computational effort and energy efficiency. The new EP controller was implemented in a humanoid robot, the iCub. The robot anticipates the state of an object undergoing a regular motion and moves its hand towards a position where grasping can be achieved more reliably.

This paper is organised as follows: Section 2 describes the experimental paradigm and presents the two control modalities under analysis (EP and standard controllers). Section 3 describes the implementation of the control systems, Section 4 analyses and compares the results and Section 5 draws the conclusions.

2 EP based control on the iCub Robot

The main goal of this work is to analyse the advantages of EP control over a standard approach in a situation where the anticipation of external object dynamics is needed. The control systems were tested in a task of reaching for a moving target. The iCub humanoid robot was used to perform that task (see Sect. 3).

2.1 Task description

To illustrate the principles of Expected Perception in non static environments, we consider the problem of a robotic arm reaching for an object following a damped pendular motion. This is a simple non-periodic regular motion that can be analytically modelled, implemented by simple means in a laboratory setup, and does not demand for a too high bandwidth in the perceptual and computational system. We note that the focus of this paper is not on the estimation of dynamical motions but on novel control methodologies involving the Expected Perception concept. In principle any predictable dynamical system (periodic or aperiodic) could be used as external motion. However, using a simple, low-velocity, regular motion for the external object prevents issues arising from the complexity of the motion and the bandwidth of the involved computational architecture, allowing us to compare different algorithms under controlled situations.

In the chosen task, the iCub robot is placed in front of a pendulum (the target) suspended on the ceiling by a wire and oscillating on a vertical plane[1]. The target ref-

[1] This is an approximation. In general the ball may have out of plane motion. During the experiments we initialised the position of the pendulum such as to minimise out of plane motion.

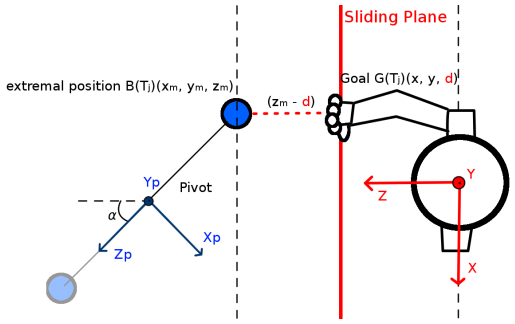

Fig. 1 Top view of the reaching task performed by the robot. At each oscillation the iCub moves its right hand to the position $G(T_j) = (X_m, Y_m, d)$, where $B(T_j) = (X_m, Y_m, Z_m)$ is the position of the target at the minimum distance $(Z_m - d)$ during the oscillation

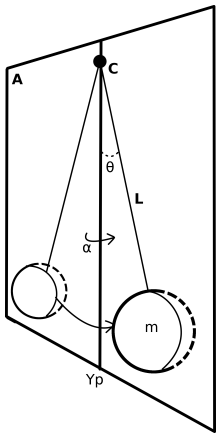

Fig. 2 Pendulum model. 2D pendulum oscillating on a plane A rotated of an angle α on the Y_p axis

erence frame $X_pY_pZ_p$ is centred on its own *pivot* point and is rotated around the Y_p axis of an angle α with respect to the robot reference frame XYZ (see Fig. 1). The robot moves its right hand along a *sliding plane* (a fixed imaginary vertical plane along which the hand is moving) perpendicular to its Z axis. The *sliding plane* is placed at distance d along the Z axis from the robot reference frame. During each oscillation j, the target reaches a minimum distance to the plane at the extremal positions of the oscillation, $B(T_j) = (X_m, Y_m, Z_m)$, where its velocity is zero. T_j is the time when the ball reaches the extremal position and $B(t)$ is the position of the ball at time t. The robot aims at grasping the ball at the goal position $G(T_j) = (X_m, Y_m, d)$, corresponding to the projection of $B(T_j)$ on the *sliding plane*. All the positions are expressed in the iCub reference frame (placed in the pelvis of the robot).
The humanoid robot tracks the target moving its head and eyes. After an observation period Δ_{obs}, necessary to observe the target movement and bootstrap its internal state, the robot starts placing the hand at the closest point on the plane to the predicted goal position. The hand position is corrected either periodically, using the classical controller, or aperiodically, using the EP based controller. The iCub repeats this procedure at each oscillation. The aim is to have the hand at the goal position when the ball reaches the extremal position and to achieve this through a computationally efficient and energy saving methodology.

2.2 System model

The target movement is approximated as a damped oscillation of a 2D pendulum on a plane A rotated of an angle α around the Y_p vertical axis (see Fig. 2). Defining Θ, $\dot{\Theta}$ and $\ddot{\Theta}$ as, respectively, angular position, velocity and acceleration, g as gravity, L as wire length, μ as damping factor and m as ball mass, the equation for describing the motion of the simplified model of the 2D pendulum is:

$$\ddot{\Theta} + \frac{\mu}{m}\dot{\Theta} + \frac{g}{L}\sin(\Theta) = 0 \tag{1}$$

Because there is uncertainty in the parameters μ, m, L, the pivot position (C_x, C_y, C_z) and α, we opt to estimate not only the motion variables but also the pendulum parameters. This results in a eight element state vector:

$$x = \left[\Theta, \dot{\Theta}, \tfrac{\mu}{m}, \tfrac{g}{L}, C_x, C_y, C_z, \alpha\right]^\top \tag{2}$$

The state vector includes all unknown variables, either time varying or constant. Despite only Θ and $\dot{\Theta}$ are time varying, we also have to estimate the others in order to make good predictions. Using Eq. (1) and (2) the transition and observation models become respectively:

$$f(x) = x + \Delta t\dot{x} = x + \Delta t \begin{bmatrix} x_2 \\ -x_3x_2 - x_4\sin(x_1) \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{3}$$

$$h(x) = \begin{bmatrix} x_5 + \frac{g}{x_4}\sin(x_1)\cos(x_8) \\ x_6 - \frac{g}{x_4}\cos(x_1) \\ x_7 - \frac{g}{x_4}\sin(x_1)\sin(x_8) \end{bmatrix} \tag{4}$$

where $x_1, \ldots, x_8$ are the entries of state vector x.

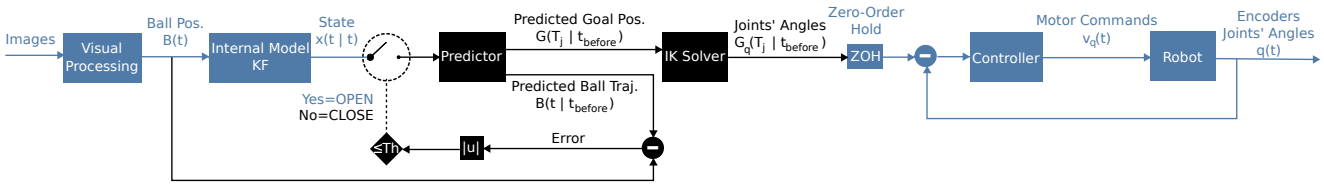

Fig. 3 Expected Perception (Th > 0) and standard controllers (Th $= 0$) (see Sect. 2.3 for more details)

2.3 The control systems

Two control modalities were tested and compared. Fig. 3 presents a block diagram showing the main components of both controllers. Their input is the 3D position of the ball $B(t)$ and the outputs are the arm motor commands expressed in joints velocities $v_q(t)$. The first control system is based on a standard architecture, while the second one is based on the Expected Perception concept. Both systems have an internal model of the pendulum implemented with an Extended Kalman Filter (EKF). The filter input is the target position computed by the visual processing block, $B(t)$. This position is used by the filter to update its internal state x. The EKF is iterated through time to obtain the predicted trajectory of the ball for future time steps $B(t+k \mid t), k = 1 \ldots (T_j - t)$. The predicted extremal position $B(T_j \mid t)$ is computed as the position for which velocity changes sign in x or y, and is used to obtain the predicted goal position $G(T_j \mid t)$. The Inverse Kinematics Solver uses $G(T_j \mid t)$ to compute the desired target robot joints' angles $G_q(T_j \mid t)$ and the Controller computes the motor joints' velocity trajectory $v_q(t)$ (Pattacini, 2011). The Controller frequency is 50 Hz, while the predictor is limited by the camera frame rate to a frequency of 30 Hz.

Under the assumption of Gaussian noises, EKF performs two main steps to estimate the state: the prediction and the correction steps. During the prediction step the filter computes its predicted internal state $x(t \mid t-1)$ and the predicted covariance matrix $P(t \mid t-1)$:

$$x(t \mid t-1) = f(x(t-1 \mid t-1)) \tag{5}$$

$$P(t \mid t-1) = A(t)P(t-1)A^T(t) + Q \tag{6}$$

where $x(t \mid t)$ is the estimate of the state at t, f is the transition function, A is the Jacobian of f, P is the state covariance matrix and Q is the process noise covariance matrix. After the prediction step the filter performs the correction step to update the previously predicted internal state and covariance matrix based on the observations:

$$K(t) = P(t \mid t-1)H^T(t)(H(t)P(t \mid t-1)H^T(t) + R)^{-1} \tag{7}$$

$$x(t \mid t) = x(t \mid t-1) + K(t)(B(t) - h(x(t \mid t-1))) \tag{8}$$

$$P(t) = (I - K(t)H(t))P(t \mid t-1) \tag{9}$$

where K is the Kalman gain, h is the observation function, H is the Jacobian of h on x, R is the covariance matrix of the observation noise, $(B - h(x(t \mid t-1)))$ is the innovation and I is the identity matrix. For more details about Kalman filtering see (Bishop and Welch, 2001).

Expected Perception based Controller: The EP based controller uses the sensory prediction to skip some of the computations done on sensory processing and kinematics solving. In particular, the EP control system re-calculates the predicted trajectory $B(t+k \mid t)$, $k = 1 \ldots (T_j - t)$, and the desired joints' angles $G_q(T_j \mid t)$ only if the error between the current ball position $B(t)$ and the old predicted trajectory $B(t \mid t_{before})$ is bigger than a threshold (Th). In $B(t \mid t_{before})$, t_{before} represents the time when the predicted trajectory was last computed. During each control cycle the robot performs the following actions (see Fig. 3 for Th > 0):

1. processes the camera images calculating the 3D ball position, $B(t)$
2. updates the filter state $x(t \mid t)$ and the covariance matrix $P(t)$
3. calculates the absolute value of the error between the current ball position and the old predicted trajectory, $E(t) = \mid B(t) - B(t \mid t_{before}) \mid$
4. **if the error is bigger than Th:** re-calculates the predicted trajectory of the ball $B(t+k \mid t)$ and the desired joints' angles $G_q(T_j \mid t)$; $t_{before} = t$
   if the error is less than Th: keeps the old prediction and inverse kinematics solutions

Standard Controller: The standard control system recomputes the desired joint positions $G_q(T_j \mid t)$ at every time step. The standard controller corresponds to the EP based controller with Th equal to 0 and $t_{before} = t$ (see Fig. 3 for Th $= 0$).

3 Implementation

The standard and EP based controllers were implemented on the Lisbon iCub robot and its simulator

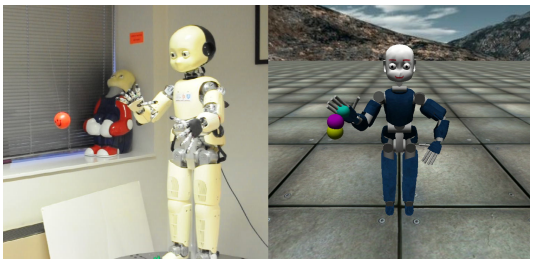

Fig. 4 Left: The iCub humanoid robot tracking the oscillating target (red ball). Right: the iCub Simulator during the test. The blue ball is the predicted goal position, the yellow ball is the output of the 3D tracker and the purple ball is the output of the filter

(Tikhanoff et al, 2008) using Yarp modules written in C++. The iCub (see Fig. 4 left) is a humanoid robot which imitates a three year old child. It has 53 DOF in total (7 for each arm, 8 for each hand, 6 for the head, 3 for the trunk and 7 for each leg). The robot has 2 DragonFly cameras in the eyes. The camera resolution is 320x240 and the images are acquired at 30 fps. The iCub simulator (see Fig. 4 right) mimics the kinematics and dynamics of the iCub robot. It is based on the Open Dynamics Engine (ODE) and uses the OpenGL as graphics engine. The middleware YARP (Yet Another Robot Platform) is used to control both the simulated and the real robot.

An EKF (Kalman et al, 1960; Bishop and Welch, 2001) was used to implement the Internal Model representing the dynamics of the pendulum. The Visual Processing, the arm Planner and the Controller were realised using the iCub modules library: the pf3DTracker (Taiana et al, 2010) was used to calculate the 3D position of the target in the left eye reference frame; the iKinGazeController (Pattacini, 2011) was employed to control the movement of the eyes and the head in order to maintain the target in the center of the camera image; the iKinCartesianController (Pattacini, 2011) was used to calculate the inverse kinematics and to control the arm. The hand $H(t)$ and ball $B(t)$ positions were both expressed in the robot reference frame (centered in the iCub pelvis).

The entire Algorithm 1 describes the EP based controller, while the same algorithm without the red coloured lines (3, 10, 20 and 21) describes the standard controller.

```
1  filter initialization;
2  predicted ball trajectory B(t+k | t) initialization;
3  t_before = t;
4  forall the control steps do
5      i = 0;
6      read new image;
7      calculate 3D position of the target, B(t);
8      execute the filter prediction step,
       x(t | t-1), P(t | t-1) ;
9      execute the filter correction step, x(t | t), P(t);
10     if (B(t) - B(t | t_before)) > Th or new oscillation
       started then
11         save filter state, x_old(t | t) = x(t | t);
12         repeat
13             i = i + 1;
14             execute the filter prediction step,
               x(t+i | t-1+i), P(t+i | t-1+i);
15             add the predicted ball position to the
               predicted ball trajectory,
               B(t+i | t) = h(x(t+i | t-1+i)) ;
16         until (t+i) < T_j;
17         calculate the goal position from the predicted
           ball trajectory, G(T_j | t) = B(t+D | t);
18         calculate the desired joints position G_q(T_j | t)
           to reach the predicted goal position G(T_j | t);
19         load filter state x(t | t) = x_old(t | t);
20         t_before = t;
21     end
22 end
```

Algorithm 1: EP based (with red lines) and standard (without red lines) control loops

4 Results

In order to test the performance of the two control algorithms, real data taken from the iCub robot was run on the iCub simulator. We decided to use the simulator in order to have a more extensive statistical analysis of the data and to avoid stressing the robot given the high number of runs to be performed. A ball with a radius of 3 cm was attached to the ceiling (2 meters high in the robot reference frame) using a wire 1.4 meters long. The pivot point was placed at 80 cm in front of the robot and slightly shifted to the left side (5 cm). An oscillation sequence was captured using the left camera of the iCub robot. The duration of the sequence was 40 seconds. During the oscillation, to prevent the target from going out from the camera's field of view, the robot tracked the ball moving both eyes and head. The data obtained from the 3D tracker and the robot encoders was saved and used as input for the simulator to test both control algorithms. The dataset was created recording 20 different sequences with the orientation angle α varying slightly around 70-80 degrees (see Fig. 5). By suitably defining the initial position of the ball, we could guarantee that α was set around these values, in order to have oscillation on all the axis, with significant ball movements especially on the x axis. This choice was taken mainly due to experimental conditions (available lab space) and making sure the target did not collide with the robot during its trajectory. On the real pendulum the α angle tended to slowly drift

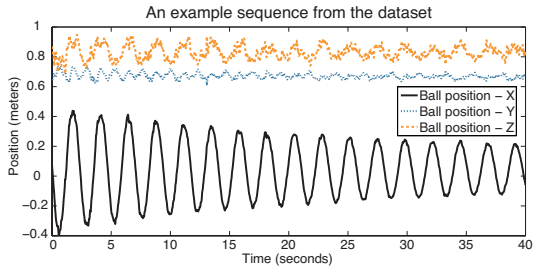

Fig. 5 One of the 20 sequences of the real pendulum trajectory as recorded by our system and used as dataset for the experiments. The first three oscillations were used to make the filter converge before driving actual arm movements

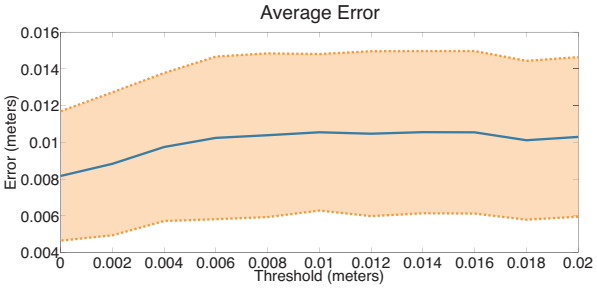

Fig. 6 Average Error changes on varying the threshold. The plot represents the EP based controller results. The standard controller corresponds to the point with Th = 0 m. Mean and standard deviation on 20 sequences are shown

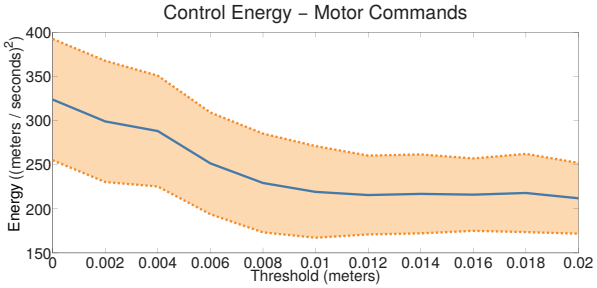

Fig. 7 Control Energy changes on varying the threshold. The plot represents the EP based controller results. The standard controller corresponds to the point with Th = 0 m. Mean and standard deviation on 20 sequences are shown

through time, but the filter was able to adapt to these changes. The presented results were calculated as the mean and variance of the system's performance across the 20 sequences. The first three oscillations of each sequence were used as observation period, Δ_{obs}, to make the EKF converge to reasonable initial values to trigger arm movements.

The EP based and standard controllers were tested on 3 different metrics: the computational time, the average error and the control energy.

Computational time (CT): Each time step the filter must update its state and, when needed, predict the target position. The computational time is the mean of all the update and prediction times calculated during one sequence.

Average error (AE): To evaluate the precision of the control algorithms, the absolute values of the errors between the extremal positions, $B(T_j)$, and the hand position at the time the ball reaches the extremal point $H(T_j)$ were calculated. Their mean over all oscillations of the sequence was used to compare the performances of the systems.

Control energy (CE): Energy consumption during the arm movement is also among the most important evaluation criteria. In order to calculate the control energy, the vector of joints velocities v_q was used. The control energy is the sum of squared joints velocities through time. The following equation is an approximation taking into account unit masses and moments of inertia in the joints:

$$CE = \sum_i \sum_q \|v_q(i) * \Delta t_i\|^2, \quad (m/s)^2 \tag{10}$$

In order to implement both control systems, the values of some parameters have to be chosen. The parameters which are shared between standard and EP based control systems are:

1. The initial state of the filter, x_0
2. The initial covariance matrix of the filter, P_0
3. The process noise of the filter, Q_0
4. The observation noise of the filter, R_0

In addition the EP based control system needs the EP threshold to be defined.

In the initial filter state the angular position and velocity were initialised to 0. Same initial value was assigned to the ratio between the damping factor μ and the ball mass m. For what concerns the length of the pendulum wire L, we used the approximate length of the real pendulum (1.4 meters). We assigned 0 to both the C_x and C_z position of the pivot, while we decided to approximate C_y by the position of the ceiling compared with the robot reference frame. We also initialised the rotation angle α of the pendulum plane to 0.

The initial covariance matrix P_0 was defined as an 8×8 Identity matrix.

The values of process and observation noise gains were decided after performing empirical tests in order to obtain a good compromise between fast convergence and precision of the filter. The value chosen for the process noise gain q was $1e^{-6}$ while the value chosen for the observation noise gain r was 0.05 (5 cm). The two error matrices Q_0 and R_0 were obtained by multiplying q and r respectively by 8×8 and 3×3 Identity matrices.

EP threshold values varied between 0cm and 2cm, in

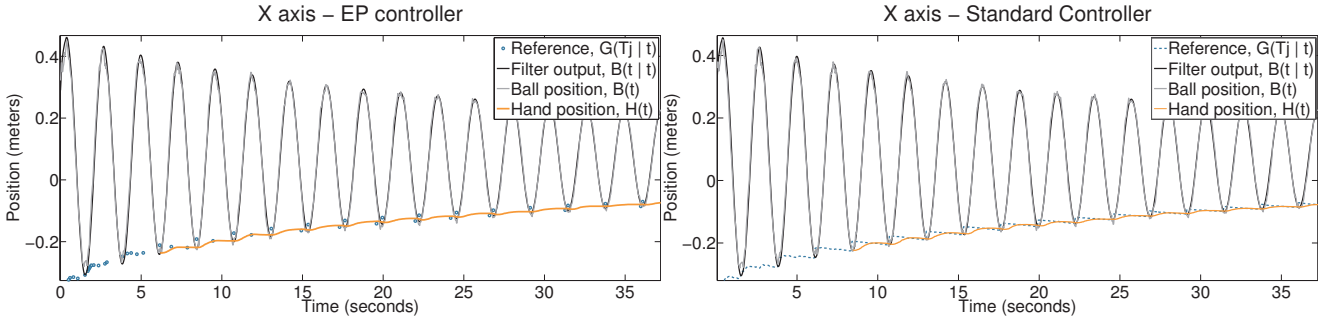

Fig. 8 Behaviour of the two algorithms during an oscillation sequence. The EP controller has Th = 4 mm. Results on the X axis are shown

steps of 2mm. A 0cm threshold corresponds to the standard control system. For each threshold value we computed the performance metrics defined at the beginning of this Section. The results of the controllers for different Th values are shown in Figs. 6 and 7. In the plots, the solid line represent the results on EP based and standard controllers.

The standard approach is more time consuming, but it is also more precise in placing the hand in the goal position (see Tab. 1). This happens because the standard controller updates the predicted target trajectory at each step while the EP based controller updates are more seldom.

In Fig. 6 the results on the average error are shown. It is possible to notice that the EP based controller error grows monotonically for small values of the threshold. Due to the initial observation period Δ_{obs} of three oscillations, the system's average prediction errors start with values close to 1 cm (1.06 cm to be precise).

The control energy is shown in Fig. 7. The figure shows decreasing control energy with increasing thresholds. This behaviour is due to the fact that the less updates of the target trajectory are executed, the less movement the iCub arm performs. The control energy stabilises for threshold values bigger than or equal to 1 cm. This happens because all the cases with thresholds bigger than 1 cm execute only few trajectory predictions per oscillation keeping the arm behaviour similar.

Therefore, the EP based controller resounds to the need of consuming less energy and sparing computational time. Based on our results, a general rule is to set the EP threshold as high as possible while keeping the system error lower than a predefined limit. This can be done empirically, determining the threshold value by experiments, or automatically, changing the threshold dynamically. In our case we can decide a maximal error and initialize the threshold accordingly. In case we are not interested in the maximal system error but, for example, in the maximal update frequency, we can decide to dynamically change the threshold based on the update frequency.

Let's assume that, for a successful grasp, a reasonable limit for the hand position error is 1 cm. Among the EP based systems with errors lower than 1 cm, the one with the threshold value of 4 mm is the one with lower Control Energy and Computational Time. The results on the X axis for the two controllers are shown in Fig. 8 and summarised in Tab. 1.

Table 1 Comparison between the computational time, average error and control energy of the EP based (Th = 4 mm) and the standard controllers

Algorithm	CT	AE	CE
EP	$0.97e^{-4}$ sec	$9.30e^{-3}$ m	$3.37e^{2}$ $(m/s)^2$
Standard	$3.08e^{-4}$ sec	$6.70e^{-3}$ m	$3.99e^{2}$ $(m/s)^2$

5 Conclusions

In this paper a bio-inspired predictive controller based on internal models has been implemented and compared with a standard predictive controller. The Expected Perception concept was used to create such predictive control system able to anticipate the dynamics of an external object. The experiments conducted on the iCub robot show that the system is able to predict the future target trajectory and anticipate the target dynamics moving the robot hand to the predicted goal position. The EP based controller is less time consuming and more energy efficient than a standard controller. Although the position error of the standard controller is lower, the position error of the EP based controller can be kept within limits by defining the EP threshold. In our case, we have set a tracking error of 1 cm according to the task and mechanical precision limits of the robot. The internal simulation of the target motion makes it possible for the robot to simulate future

events and only change the plan of action when significant deviations from the predictions occur. The results confirm the advantages of using an EP based control system that models the dynamics of external objects.

Acknowledgements This work has been partially supported by the European Commission with the RoboSoM Project, FP7-ICT-248366. The authors would like to thank Italian Ministry of Foreign Affairs, General Directorate for Cultural Promotion and Cooperation, for its support to the establishment of the RoboCasa joint laboratory, the Portuguese National Funding FCT project [UID/EEA/50009/2013] and the European Commission project POETICON++ (FP7-ICT-288382).

References

Barrera A, Laschi C (2010) Anticipatory visual perception as a bio-inspired mechanism underlying robot locomotion. In: Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE, pp 3206–3209

Bauml B, Birbach O, Wimbock T, Frese U, Dietrich A, Hirzinger G (2011) Catching flying balls with a mobile humanoid: System overview and design considerations. In: Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on, pp 513–520

Berthoz A (2002) The brain's sense of movement. Harvard University Press

Bishop G, Welch G (2001) An introduction to the kalman filter. Proc of SIGGRAPH, Course 8:27,599–3175

Datteri E, Teti G, Laschi C, Tamburrini G, Dario P, Guglielmelli E (2003) Expected perception: an anticipation-based perception-action scheme in robots. In: Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on, vol 1, pp 934–939 vol.1

Gross HM, Heinze A, Seiler T, Stephan V (1999) Generative character of perception: A neural architecture for sensorimotor anticipation. Neural Networks 12(7):1101–1129

Hoffmann H (2007) Perception through visuomotor anticipation in a mobile robot. Neural Networks 20(1):22–33

Johansson RS (1998) Sensory input and control of grip. Sensory guidance of movement pp 45–59

Kalman RE, et al (1960) A new approach to linear filtering and prediction problems. Journal of basic Engineering 82(1):35–45

Kim S, Billard A (2012) Estimating the non-linear dynamics of free-flying objects. Robotics and Autonomous Systems 60(9):1108 – 1122

Kober J, Glisson M, Mistry M (2012) Playing catch and juggling with a humanoid robot. In: Humanoid Robots (Humanoids), 2012 12th IEEE-RAS International Conference on, pp 875–881

Laschi C, Asuni G, Teti G, Carrozza MC, Dario P, Guglielmelli E, Johansson R (2006) A bio-inspired neural sensory-motor coordination scheme for robot reaching and preshaping. In: Biomedical Robotics and Biomechatronics, 2006. BioRob 2006. The First IEEE/RAS-EMBS International Conference on, IEEE, pp 531–536

Laschi C, Asuni G, Guglielmelli E, Teti G, Johansson R, Konosu H, Wasik Z, Carrozza MC, Dario P (2008) A bio-inspired predictive sensory-motor coordination scheme for robot reaching and preshaping. Autonomous Robots 25(1-2):85–101

Miall R, Wolpert DM (1996) Forward models for physiological motor control. Neural networks 9(8):1265–1279

Moutinho N, Cauli N, Falotico E, Ferreira R, Gaspar J, Bernardino A, Santos-Victor J, Dario P, Laschi C (2011) An expected perception architecture using visual 3d reconstruction for a humanoid robot. In: Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on, IEEE, pp 4826–4831

Nguyen-Tuong D, Peters J (2011) Model learning for robot control: a survey. Cognitive processing 12(4):319–340

Pattacini U (2011) Modular cartesian controllers for humanoid robots: Design and implementation on the icub. PhD thesis, Ph. D. dissertation, RBCS, Italian Institute of Technology, Genova

Taiana M, Santos J, Gaspar J, Nascimento J, Bernardino A, Lima P (2010) Tracking objects with generic calibrated sensors: An algorithm based on color and 3d shape features. Robotics and Autonomous Systems 58(6):784 – 795, omnidirectional Robot Vision

Tikhanoff V, Cangelosi A, Fitzpatrick P, Metta G, Natale L, Nori F (2008) An open-source simulator for cognitive robotics research: the prototype of the icub humanoid robot simulator. In: Proceedings of the 8th workshop on performance metrics for intelligent systems, ACM, pp 57–61

Vannucci L, Cauli N, Falotico E, Bernardino A, Laschi C (2014) Adaptive visual pursuit involving eye-head coordination and prediction of the target motion. In: Proceedings of the 14th IEEE-RAS International Conference on Humanoid Robots, pp 541–546