3D Monocular Robotic Ball Catching with an Iterative Trajectory Estimation Refinement

Vincenzo Lippiello and Fabio Ruggiero

***Abstract*— In this paper, a 3D robotic ball catching algorithm which employs only an eye-in-hand monocular visual-system is presented. A partitioned visual servoing control is used in order to generate the robot motion, keeping always the ball in the field of view of the camera. When the ball is detected, the camera mounted on the robot end-effector is commanded to follow a suitable baseline in order to acquire measurements and provide a first possible interception point through a linear estimation process. Thereafter, further visual measures are acquired in order to continuously refine the previous prediction through a non-linear estimation process. Experimental results show the effectiveness of the proposed solution.**

I. INTRODUCTION

Smart sensing, object tracking, motion prediction, on-line trajectory planning and motion coordination are capabilities required in a robotic system to catch a thrown ball.

One of the first approaches where robot manipulators are used in catching moving objects can be found in [1], while in [2] a stereo vision system with a large baseline, an extended Kalman filter (EKF) and a ball trajectory predictor are exploited so as to build a robotic ball catcher. The same system equipped with a dexterous multi-fingered hand is tested in [3], while recently a new version of this work is proposed in [4] involving a mobile humanoid and a circular gradient method to detect the ball in the images.

In [5], [6] a high-speed multi-fingered hand and a high-speed stereo vision system are employed to catch a falling ball and a falling cylinder. A robotic arm whose aim is to catch a ball before it fell from a table is considered in [7], where uncalibrated cameras are employed to track the moving ball. In [8] the control is applied to achieve simultaneously all the 2D tasks defined for all the images: if all the respective goals defined in the images are accomplished at the same time, then the 3D task can be interpreted as successful. A DSP is employed as a computational platform in the catcher robot system developed in [9], while in [10] an iterative prediction algorithm determines the final time and position of a humanoid motion, whose primitives derive from studies on human movements.

Other examples of robotic ball catching can be found in [11], where the ball path is predicted with the help of a proper neural network. The ball catching task is also considered as a case study in several virtual-reality applications [12], [13], [14].

The authors are with PRISMA Lab, Dipartimento di Informatica e Sistemistica, Università degli Studi di Napoli Federico II, via Claudio 21, 80125, Naples, Italy {lippiello, fabio.ruggiero}@unina.it

Several papers make use of Chapman's strategy – the fielder should run at a proper speed to maintain a constant increasing rate of the tangent to the ball's elevation angle [15] – to catch a ball. In [16] reinforcement learning models are used, while an autonomous mobile robot is considered in [17] for a ball catching task using a visual feedback control method based on a *Linear Optimal Trajectory* strategy. In [18] it is introduced an alternative strategy still based on Chapman's hypothesis, called *Gaining Angle of Gaze*, which requires only the information about the elevation angle of gaze captured as a 2D information. Finally, in [19] a motion-analysing technique over a finite time is used inside a closed-loop system in order to catch a thrown ball.

Most of the presented approaches use either a stereo visual system to solve the 3D catching problem or a single camera for the 2D case. This scenario is reasonable because 3D tracking of the ball takes benefits from triangulation methods while, in the case of a single camera, only 2D information is directly available. However, a high frame rate and optics with a good accuracy are required to achieve an accurate and fast trajectory prediction, i.e. a successful catch. By using only one camera, the cost of the equipment can be reduced. Moreover, the calibration procedure for one camera is easier than in the stereo case. In [20] the estimation of the 3D state of a thrown object by using a least-squares solution starting from a sequence of images given by a single camera is presented. Further, in [21] a combination of image-based and position-based visual servoing with an eye-to-hand camera configuration is employed in order to catch a ball whose trajectory is estimated through a RLS algorithm.

In this paper, a monocular robotic 3D ball catching is proposed. A robot manipulator with a standard CCD camera mounted in an eye-in-hand configuration is driven by visual information in order to track a thrown ball. When the ball is detected for the first time, the camera is commanded to follow a suitable baseline in the 3D space in order to increase the estimation robustness. The ball is always kept in the camera field of view through a partitioned visual servoing control. During this starting motion, 2D information is collected and elaborated in order to get a first prediction of the ball trajectory through a rough linear estimation: such prediction is employed as a starting point for a more precise trajectory refinement through a nonlinear estimator. Hence, the visual measurements are continuously elaborated in order to update the estimation of the ball trajectory on-line, and thus the prediction of the interception pose. Finally, whenever the continuous refinement does not improve significantly the prediction of the trajectory, the final catching pose can be

computed by taking into account the ball and the robot dynamics, in such a way as to accommodate the ball into the robotic hand. Experimental results using a common industrial robot demonstrate the effectiveness of the proposed solution.

II. Image processing

The images provided by a calibrated camera mounted in an eye-in-hand configuration is continuously elaborated in order to identify and extract the position of the ball, i.e. the centroid, in the normalized image plane. The whole image is elaborated until the ball is detected, then a dynamic windowing technique, which is based on a first order prediction algorithm of the ball motion in the image plane, is employed after the first detection, so as to reduce the computational requirement of the image elaboration process. Moreover, by adopting the Region of Interest (RoI) camera acquisition modality, which is available on most of the current USB cameras, the camera frame rate can be significantly speeded up (e.g. easily more than 100 Hz).

An equalized color-based clustering is adopted in the image processing, and it makes use of the *Hue, Lightness, and Saturation Color Space* so as to limit as much as possible the problems related to the variations of the environmental lightness along the ball path. In details, a binarization process is performed through an equalized test, which is based on a histogram of the H-channel and centred around the known ball color, together with a min/max S-channel threshold. After some post-elaboration process employed to reduce the image noise, all the blobs present in the binarized image are collected and filtered so as to eliminate the background and all the blobs with a very small area. If more than one blob overcomes this filtering process, a neighbourhood selection criteria with respect to the predicted ball position is adopted. Finally, the centroid of the selected blob is evaluated as a good approximation of the ball center.

This algorithm can also be executed to find other relevant objects in the environment. For example, a color tuned on the hand of the pitcher is also employed, which allows recognizing up to a fully hand occlusion whether the ball is held in the hand or it has been thrown, avoiding in such a way unnecessary initial movements of the robot.

III. On-line motion planning and Control law

Without loss of generality, the camera frame is here considered coincident with the hand (end-effector) frame, with the camera optical axis aligned with the hand approaching axis. Obviously, being the camera fixed with respect to the hand in the considered eye-in-hand camera configuration, a fixed transformation should be considered in the cases where these reference frames are not coincident. Hence, in the remainder of the paper, only the camera frame will be considered. Moreover, the proposed visual control law belongs to the category named *Resolved-Velocity Image-Based Visual Servoing* [22], for which it is assumed that the manipulator dynamics is taken into account directly by the low-level robot controller.

A necessary but non sufficient condition to accomplish a visual ball catching task is to keep the ball in the field of view of the camera; since with small movements of the camera orientation (i.e. with small movements of the robot joints) large parts of the scene can be observed, in the proposed method the partitioned approach presented in [23] has been employed. The rotational components of the robot motion will be reserved to the ball tracking task, while the positional components of the camera motion have to be generated in a way as to intercept the ball trajectory.

By adopting a calibrated camera, the normalized image coordinates of the ball centroid $\boldsymbol{s} = \begin{bmatrix} X & Y \end{bmatrix}^{\mathrm{T}}$, corresponding to a ball position $\boldsymbol{p}_o^c$ with respect to the camera frame $\Sigma_c = O_c - x_c y_c z_c$, are considered

$$\boldsymbol{p}_o^c = \begin{bmatrix} x^c & y^c & z^c \end{bmatrix}^{\mathrm{T}} = z^c \begin{bmatrix} X & Y & 1 \end{bmatrix}^{\mathrm{T}} = z^c \tilde{\boldsymbol{s}},$$

where $\tilde{\boldsymbol{s}} = \begin{bmatrix} \boldsymbol{s}^{\mathrm{T}} & 1 \end{bmatrix}^{\mathrm{T}}$. The absolute velocity of the camera $\boldsymbol{v}_c^c = \begin{bmatrix} \dot{\boldsymbol{p}}_c^{c\mathrm{T}} & \boldsymbol{\omega}_c^{c\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$, the absolute velocity of the thrown object $\boldsymbol{v}_o^c = \begin{bmatrix} \dot{\boldsymbol{p}}_o^{c\mathrm{T}} & \boldsymbol{\omega}_o^{c\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$, both expressed with respect to Σ_c, and the velocity of the image feature $\dot{\boldsymbol{s}}$ in the image plane are related by the following linear equation [22]

$$\dot{\boldsymbol{s}} = \boldsymbol{L}_s \boldsymbol{v}_c^c + \boldsymbol{L}_s \boldsymbol{\Gamma}(-\boldsymbol{p}_o^c) \boldsymbol{v}_o^c,$$

where $\boldsymbol{L}_s$ is (2×6) interaction matrix for a point image feature defined as follows [22]

$$\boldsymbol{L}_s = \begin{bmatrix} -1/z^c & 0 & X/z^c & XY & -1-X^2 & Y \\ 0 & -1/z^c & Y/z^c & 1+Y^2 & -XY & -X \end{bmatrix}, \tag{1}$$

while $\boldsymbol{\Gamma}(\cdot)$ is the following (6×6) matrix

$$\boldsymbol{\Gamma}(\cdot) = \begin{bmatrix} -\boldsymbol{I}_3 & \boldsymbol{S}(\cdot) \\ \boldsymbol{0} & -\boldsymbol{I}_3 \end{bmatrix},$$

in which $\boldsymbol{I}_n$ denotes the $(n \times n)$ identity matrix and $\boldsymbol{S}(\cdot)$ the skew-symmetric matrix.

In the proposed approach, the translational component $\dot{\boldsymbol{p}}_c^c$ is reserved to move the hand in order to intercept the ball. A fifth-order polynomial vector is thus considered so as to compute the desired trajectory for the camera on-line, i.e. for the hand, in the 3D Cartesian space

$$\boldsymbol{p}_{c,d}(t) = \boldsymbol{a}_5 t^5 + \boldsymbol{a}_4 t^4 + \boldsymbol{a}_3 t^3 + \boldsymbol{a}_2 t^2 + \boldsymbol{a}_1 t + \boldsymbol{a}_0, \tag{2}$$

where $\boldsymbol{p}_{c,d}$ is the (3×1) desired absolute position of the camera, and $\boldsymbol{a}_h$ with $h = 0, \ldots, 5$ are (3×1) coefficient vectors. Hence, the translational components of the desired camera velocity and acceleration are namely equal to

$$\dot{\boldsymbol{p}}_{c,d} = 5\boldsymbol{a}_5 t^4 + 4\boldsymbol{a}_4 t^3 + 3\boldsymbol{a}_3 t^2 + 2\boldsymbol{a}_2 t + \boldsymbol{a}_1, \tag{3}$$

$$\ddot{\boldsymbol{p}}_{c,d} = 20\boldsymbol{a}_5 t^3 + 12\boldsymbol{a}_4 t^2 + 6\boldsymbol{a}_3 t + 2\boldsymbol{a}_2. \tag{4}$$

In order to update the parameters $\boldsymbol{a}_h$ on-line, ensuring a smooth re-planned trajectory when a new interception point is available, the continuity between the current motion state and the new initial one have to be imposed. Hence, by denoting with t_i, t_f, $\boldsymbol{p}_{c,d,i}$, $\boldsymbol{p}_{c,d,f}$, $\dot{\boldsymbol{p}}_{c,d,i}$, $\dot{\boldsymbol{p}}_{c,d,f}$, $\ddot{\boldsymbol{p}}_{c,d,i}$ and $\ddot{\boldsymbol{p}}_{c,d,f}$ the initial and final planned time, position, linear velocity and acceleration, respectively, with

$\bar{\boldsymbol{a}}_h = \begin{bmatrix}\boldsymbol{a}_5 & \boldsymbol{a}_4 & \boldsymbol{a}_3 & \boldsymbol{a}_2 & \boldsymbol{a}_1 & \boldsymbol{a}_0\end{bmatrix}^T$ and by taking into account (2), (3) and (4), a linear quadratic system of 18 equations and unknowns is obtained. These initial conditions correspond to the current robot motion state, while the final ones depend on the current estimation.

The translation velocity input for the camera frame can thus be generated as

$$\dot{\boldsymbol{p}}_c^c = \boldsymbol{R}_c^{\mathrm{T}} \left(\dot{\boldsymbol{p}}_{c,d} + \boldsymbol{K}_p \boldsymbol{e}_p \right), \tag{5}$$

where $\boldsymbol{R}_c$ is the rotational matrix of the camera frame with respect to the absolute world frame, $\boldsymbol{K}_p$ is a diagonal constant (3×3) gain matrix, and $\boldsymbol{e}_p$ is the (3×1) error vector at time t between the desired planned trajectory (2) and the current one provided by the robot direct kinematics.

By denoting with $\boldsymbol{L}_{sp}$ and $\boldsymbol{L}_{so}$ the (2×3) sub-matrices corresponding to the first and last three columns of (1), the rotational components of the velocity input for the camera frame are generated through the following control law:

$$\begin{aligned}\boldsymbol{\omega}_c^c = \boldsymbol{L}_{so}^{\dagger} [& \boldsymbol{K}_{so,e_{b2}}(\boldsymbol{e}_s)\boldsymbol{\tau}_{e_{b1}}(\boldsymbol{e}_s) \\ & - \hat{\boldsymbol{L}}_{sp} \left(\dot{\boldsymbol{p}}_c^c - \hat{\dot{\boldsymbol{p}}}_o^c + \boldsymbol{S}(-\hat{\boldsymbol{p}}_o^c)\hat{\boldsymbol{\omega}}_o^c \right)] + \hat{\boldsymbol{\omega}}_o^c,\end{aligned} \tag{6}$$

where the symbol † denotes the pseudo-inverse of a matrix, $\hat{\boldsymbol{p}}_o^c$ is the estimation of the unknown distance between the ball and the object expressed in Σ_c, $\dot{\boldsymbol{p}}_c^c$ is evaluated in (5), while $\hat{\dot{\boldsymbol{p}}}_o^c$ and $\hat{\boldsymbol{\omega}}_o^c$ are the estimations of both the unknown absolute linear and angular velocities of the ball with respect to Σ_c. Moreover, $\boldsymbol{e}_s = -\boldsymbol{s}$ is the image error vector that becomes zero when the camera is pointed towards the centroid of the ball, $\boldsymbol{\tau}_{e_b}(\boldsymbol{e}_s)$ is a threshold function defined as follows

$$\boldsymbol{\tau}_{eb1}(\boldsymbol{e}_s) = \begin{cases} \boldsymbol{0} & \text{if } \|\boldsymbol{e}_s\| \le e_{b1} \\ \left(1 - \frac{e_{b1}}{\|\boldsymbol{e}_s\|}\right) \boldsymbol{e}_s & \text{if } \|\boldsymbol{e}_s\| > e_{b1}, \end{cases}$$

and $\boldsymbol{K}_{so}(\boldsymbol{e}_s)$ is a (2×2) gain matrix defined as follows

$$\boldsymbol{K}_{so}(\boldsymbol{e}_s) = \begin{cases} k_o \boldsymbol{I}_2 & \text{if } \|\boldsymbol{e}_s\| \le e_{b2} \\ k_o e^{\beta_o \left(\frac{\|\boldsymbol{e}_s\|}{e_{b2}} - 1 \right)} \boldsymbol{I}_2 & \text{if } e_{b2} < \|\boldsymbol{e}_s\| \le e_{b3} \\ k_o e^{\beta_o \left(\frac{e_{b3}}{e_{b2}} - 1 \right)} \boldsymbol{I}_2 & \text{if } \|\boldsymbol{e}_s\| > e_{b3} \end{cases}$$

where $e_{b3} > e_{b2} > e_{b1} > 0$, $k_o > 0$ is a gain factor and $\beta_o > 0$ is a barrier factor that tunes the increasing rate of $\boldsymbol{K}_{so}$ in accordance to the image plane limits. In this way, when the ball centroid approaches these limits, the gain $\boldsymbol{K}_{so}$ rapidly increases in order to avoid the lost of the ball view.

It is worth noticing that in (6) only $\boldsymbol{L}_{sp}$ needs to be estimated, since it depends on the third component of the unknown $\boldsymbol{p}_o^c$, while $\boldsymbol{L}_{so}$ depends on the available visual measurements. The stability analysis of the system with control laws (5) and (6) is here omitted for brevity, but it can be asked to the authors, while the details about the unknown parameters estimation are presented in Section IV-D.

Finally, the joints velocity input to the robot control unit can be evaluated as follows [22]

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^{\dagger}(\boldsymbol{q})\boldsymbol{T}_c \boldsymbol{v}_c^c + \boldsymbol{N}_J \boldsymbol{K}_r \dot{\boldsymbol{q}}_r, \tag{7}$$

where $\boldsymbol{q}$ is the joint position vector, $\boldsymbol{J}(\boldsymbol{q})$ is the robot Jacobian matrix, which provides the relationship between the joints and the camera velocity, $\boldsymbol{T}_c$ is the (6×6) matrix which relates the velocity of the camera with respect to the camera frame to the velocity of the robot end-effector with respect to the base frame, $\boldsymbol{N}_J$ denotes the projector matrix onto the null space of $\boldsymbol{J}$, $\boldsymbol{K}_r$ is a gain diagonal matrix, and $\dot{\boldsymbol{q}}_r$ is a set of joint velocities exploited in the redundancy management in order to optimize some sub-tasks. In particular, $\dot{\boldsymbol{q}}_r$ is employed to avoid joint limits and kinematic singularities.

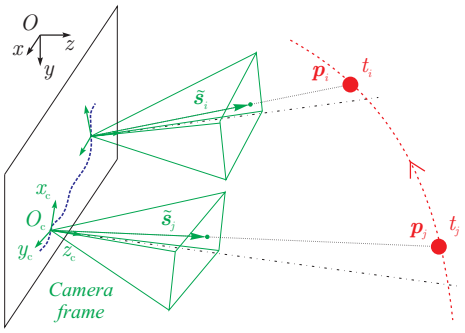

Fig. 1. Absolute world frame (black), and the camera frame $O_c - x_c y_c z_c$.

IV. Trajectory Estimation

A. Initial baseline and linear initialization

By using the measures provided by a single camera system, a classic static triangulation method cannot be employed like for the stereo vision. Hence an estimation process that interpolates the 2D measurements along time is proposed.

In order to improve the robustness of the estimation process, the visual data collection has to be acquired moving the camera along a significant baseline. Therefore, when the ball is detected for the first time, the camera is moved with high velocity along a straight line in the 3D Cartesian space, while its orientation is controlled to keep the ball in its field of view, as explained in the previous section.

By exploiting the motion performed by the camera during the initial baseline, a sequence of image measurements of the ball trajectory can be acquired and collected.

By denoting with t_k the k-th visual sample time, $\tilde{\boldsymbol{s}}_k$ the corresponding acquired image feature vector, the points $\boldsymbol{p} = \begin{bmatrix} x & y & z \end{bmatrix}^{\mathrm{T}}$ which belong to the *optical ray* passing through the current absolute origin of the camera $\boldsymbol{c}_k = \begin{bmatrix} c_{x,k} & c_{y,k} & c_{z,k} \end{bmatrix}^{\mathrm{T}}$ and the k-th feature vector $\boldsymbol{r}_k = \begin{bmatrix} r_{x,k} & r_{y,k} & r_{z,k} \end{bmatrix}^{\mathrm{T}} = \boldsymbol{c}_k + \boldsymbol{R}_{c,k}\tilde{\boldsymbol{s}}$ (see Fig. 1), can be defined with the following straight line equations

$$\begin{cases} (r_{y,k} - c_{y,k})x + (c_{x,k} - r_{x,k})y + r_{x,k}c_{y,k} - r_{y,k}c_{x,k} = 0 \\ (r_{z,k} - c_{z,k})x + (c_{x,k} - r_{x,k})z + r_{x,k}c_{z,k} - r_{z,k}c_{x,k} = 0, \end{cases} \tag{8}$$

where $\boldsymbol{R}_{c,k}$ is the rotation matrix of the camera frame with respect to the absolute world frame at time t_k. Notice that both $\boldsymbol{c}_k$ and $\boldsymbol{R}_{c,k}$ are provided by the robot direct kinematics.

By supposing negligible the effects of the air drag factor, only for this linear initialization part, the points of the absolute ball trajectory can be described as a parabolic function of the time t with the equation $\boldsymbol{p} = \boldsymbol{p}_0 + \dot{\boldsymbol{p}}_0 t + 0.5\boldsymbol{g}t^2$, where $\boldsymbol{g}$ is the gravity acceleration, and $\boldsymbol{p}_0$ and $\dot{\boldsymbol{p}}_0$ are the initial absolute ball position and velocity (for $t = 0$, corresponding to the

time of the first ball detection), respectively. Without loss of generality, the gravity acceleration is considered aligned to the axis y of the world frame, with $g = 9.81\mathrm{m/s}^2$.

The measured optical rays intersect the ball trajectory at each sample time t_k(see Fig. 1). Hence, by replacing $\boldsymbol{p}$ of the ball trajectory model into (8), the system $\boldsymbol{A}_k \begin{bmatrix}\boldsymbol{p}_0^{\mathrm{T}} & \dot{\boldsymbol{p}}_0^{\mathrm{T}}\end{bmatrix}^{\mathrm{T}} = \boldsymbol{b}_k$, of 2 equations in the 6 unknowns $\boldsymbol{p}_0$ and $\dot{\boldsymbol{p}}_0$, which fully describe the ball trajectory, can be achieved, where

$$\boldsymbol{A}_k = \begin{bmatrix} r_{y,k} - c_{y,k} & c_{x,k} - r_{x,k} & 0 & \cdots \\ r_{z,k} - c_{y,k} & 0 & c_{x,k} - r_{x,k} & \cdots \\ (r_{y,k} - c_{y,k})t_k & (c_{x,k} - r_{x,k})t_k & 0 & \\ (r_{z,k} - c_{y,k})t_k & 0 & (c_{x,k} - r_{x,k})t_k & \end{bmatrix}$$

$$\boldsymbol{b}_k = \begin{bmatrix} r_{y,k}c_{x,k} - r_{x,k}c_{y,k} - \frac{1}{2}(c_{x,k} - r_{x,k})gt_k^2 \\ r_{z,k}c_{x,k} - r_{x,k}c_{z,k} \end{bmatrix}.$$

By stacking into rows the n_l measurements $\boldsymbol{A}_k$ and $\boldsymbol{b}_k$, a least-squares solution is considered for the system $\boldsymbol{A}\begin{bmatrix}\boldsymbol{p}_0^{\mathrm{T}} & \dot{\boldsymbol{p}}_0^{\mathrm{T}}\end{bmatrix}^{\mathrm{T}} = \boldsymbol{b}$ of n_l equations and 6 unknowns.

A weighted pseudo-inverse of $\boldsymbol{A}$ is employed in order to increase the relevance of the last measurements with respect to the first ones, since they are less affected by the negligence of the air drag (the ball velocity is smaller) and are characterized by a higher image resolution (the ball is closer to the camera).

The assumptions about the knowledge of the ball trajectory model and of the gravity vector implicitly solve the scaling factor problem, due to the use of 2D visual data, especially for the y component. Moreover, the acquisition of measurements from several camera positions also contribute to the good conditioning of the least-squares problem. However, the simplified ball trajectory model, in which the air drag factor is missing, lacks in accuracy [2].

Once the estimation process produces its result, the first candidate for the interception point is evaluated. Namely, starting from the actual state (time, position, velocity and acceleration) of the camera motion along the baseline, the parameters $\boldsymbol{a}_h$ are evaluated in such a way as to reach the new predicted position, whose computational details are provided in the Section IV-C. The rotational part of the camera, instead, is kept free to track the ball in order to acquire more measurements.

The trajectory estimation provided by this linear algorithm is employed as a starting point for a non-linear estimation algorithm, that continuously refines the current estimation using new available ball observations and a more accurate ball trajectory model, as described in the following subsection.

B. Nonlinear estimation refinement

During the time required by the previous linear estimation process, a new set of visual measurements are acquired. Then, both the new measures and the old ones are employed in a nonlinear estimation process whose starting condition is the result obtained with the previous linear method.

By denoting with $\boldsymbol{s}_k$ the ball centroid acquired at a certain time t_k, the considered cost function to minimize is

$$\min_{\boldsymbol{p}_0, \boldsymbol{v}_0} \sum_{k=1}^{n} \left\| \frac{1}{\hat{z}_k^c} \begin{bmatrix} \hat{x}_k^c \\ \hat{y}_k^c \end{bmatrix} - \boldsymbol{s}_k \right\|, \tag{9}$$

where n is the current number of available ball observations and $\hat{\boldsymbol{p}}_k^c$ is the estimated ball position with respect to the camera frame at time t_k

$$\hat{\boldsymbol{p}}_k^c = \begin{bmatrix}\hat{x}_k^c & \hat{y}_k^c & \hat{z}_k^c\end{bmatrix}^{\mathrm{T}} = \boldsymbol{R}_{c,k}^{\mathrm{T}} \left(\hat{\boldsymbol{p}}_k - \boldsymbol{c}_k\right).$$

The absolute estimated ball position $\hat{\boldsymbol{p}}_k(\boldsymbol{p}_0, \dot{\boldsymbol{p}}_0, t_k)$ is obtained by numerically integrating, in the time interval $[0, t_k]$, with initial conditions $\boldsymbol{p}_0$ and $\dot{\boldsymbol{p}}_0$, the following ballistic ball motion with the air drag [2]:

$$\ddot{\boldsymbol{p}}(t) = \boldsymbol{g} - \frac{c_w \pi d_b^2 \rho_a}{2m_b} \left\|\dot{\boldsymbol{p}}(t)\right\| \dot{\boldsymbol{p}}(t), \tag{10}$$

where c_w is a coefficient depending on the thrown object, d_b is the ball diameter, ρ_a is the density of the air and m_b is the mass of the ball.

The minimization of the cost function (9), performed by using a well known *Levembert-Marquart* algorithm, means that the initial conditions about the ball trajectory are found in such a way as to generate an estimated ball trajectory which minimizes the distance between the predicted projection of the ball onto the image plane and the corresponding measured ball observations.

During the time in which the nonlinear estimation process computes the update values of $\boldsymbol{p}_0$ and $\dot{\boldsymbol{p}}_0$, and thus of the new interception point, new measures are acquired and collected with the old ones. This new data set is employed during the next non-linear refinement that adopts the previous optimal solution as initial condition, resulting in an efficient iterative optimization algorithm: with this arrangement, the computational time of the non-linear optimization algorithm is significantly reduced and fully compatible with a real-time application. Then, once the update interception point is available, the parameters $\boldsymbol{a}_h$ are again tuned in order to have a trajectory (2) and (3) which starts from the current camera motion state and ends with the estimated position at the update final interception time.

When either the estimation refinement converges to a constant value or the current estimated catching time is approaching, the refinement process is stopped and the final catching trajectory is planned to accommodate the ball motion into the robotic hand, as explained in the next subsection.

C. Interception pose selection

The current catching point $\boldsymbol{p}_\times$ along the estimated ball trajectory, which is reachable with the minimum effort of the robot joint torques, is evaluated. In details, the catching time $t_\times$ and the ball velocity $\dot{\boldsymbol{p}}_\times$ at the position $\boldsymbol{p}_\times$ can be computed from the current predicted trajectory and the kinematic and dynamic robot models. Hence, starting from the actual motion state (position, velocity, acceleration) of the robotic hand, the parameters $\boldsymbol{a}_h$ can be tuned so as to lead the hand to the point $\boldsymbol{p}_\times$ at time $t_\times$ with the same velocity of the ball (or scaled if that is too high).

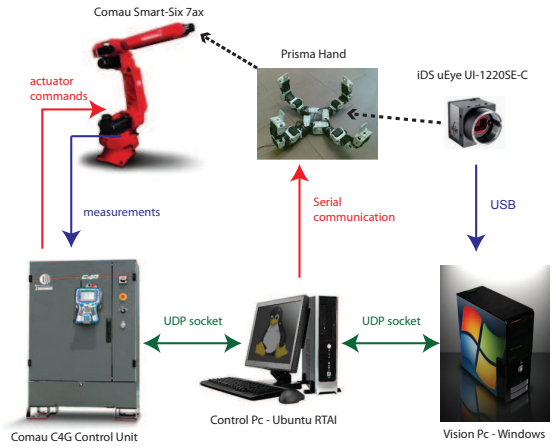

Fig. 2. Architecture of the ball catching system.

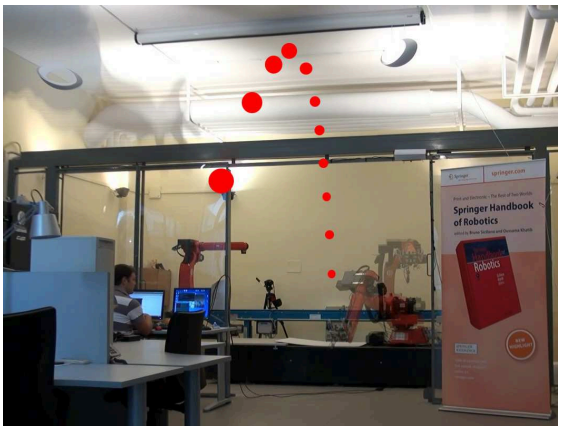

Fig. 3. Overlay of the ball trajectory and robot motion.

When the refinement process stops, the *catching path* can be generated: the hand orientation is controlled in order to have a direction of the camera equal to the tangent to the estimated ball trajectory at the predicted catching point $\boldsymbol{p}_{\times}$. Once this point has been reached at the estimated catching time, the hand starts to close its fingers and is moved following the same predicted path of the ball, while its velocity will be decreased in a fixed time (or displacement) until zero, in order to allow the dissipation of the impact energy in a sufficient time interval.

D. Estimated parameters

In order to compute the estimated quantities in (6), starting form the current estimated initial ball position $\boldsymbol{p}_0$ and velocity $\dot{\boldsymbol{p}}_0$, the ballistic motion model of the ball (10) is numerically integrated in the time interval $[0, t_i]$ so as to obtain $\hat{\boldsymbol{p}}_o^c$ at a certain time t_i, while the term $\hat{z}^c$ required for the evaluation of $\hat{\boldsymbol{L}}_{sp}$ is the third component of such vector.

In order to complete the terms required in (6), the two (3×1) vectors $\hat{\dot{\boldsymbol{p}}}_o^c$ and $\hat{\boldsymbol{\omega}}_o^c$ should be computed. The former can be again retrieved by the previous numerical integration, while the latter can be obtained as $\hat{\boldsymbol{\omega}}_o^c = (\hat{\boldsymbol{p}}_o^c \times \hat{\dot{\boldsymbol{p}}}_o^c)/\|\hat{\boldsymbol{p}}_o^c\|^2$.

Since before of n_l measurements it is not possible to have any estimation of $\boldsymbol{p}_0$ and $\dot{\boldsymbol{p}}_0$, an initial rough estimation should be provided in order to compute the above quantities. For such a reason, a statistical calibration has been preliminary realized, and the results have in turn been employed in the experiments presented in the next section.

V. Experiments

Figure 2 shows the experimental set-up implementing the proposed control algorithm. A Comau Smart-Six robot manipulator mounted on a sliding track and equipped with a 4-fingered hand composed of 16 Bioloid Dynamixel AX-12 servomotors has been employed. The Comau C4G control unit is in charge of the compensation of the robot dynamic model, while an external PC with Ubuntu OS patched with the RTAI-real time kernel generates the position/orientation references at 2 ms. The control PC communicates with a second Windows OS PC that is responsible of the visual elaboration. An industrial USB iDS UEYE UI-1220SE-C camera has been mounted, behind a transparent plexiglass, directly in the center of the palm of the hand. A high-priority multi-thread programming together with the synchronization signal provided by the camera have been employed to improve the stability of the elaboration time.

In order to increase the acquisition frame-rate, which affects the overall performances of the algorithm, an image size of (375×500) pixels and a dynamic RoI windowing with a dimension of (150×150) pixels have been employed, yielding a visual sampling frequency of about 130 fps. A ball with a radius of 3.5 cm and a weight of about 26 g has been considered. The coefficients of the air drag factor have been chosen as follows: $c_w = 0.45$ and $\rho_a = 1.293$ kg/m^3.

The control gains of the vision-based ball tracking control have been tuned to $k_o = 7$, with with $e_{b1} = 10$ pixels, $e_{b2} = 100$ pixels and $e_{b3} = 300$ pixels. The control gains, instead, for the positional part have been tuned to $\boldsymbol{K}_p = 50\boldsymbol{I}_3$. The redundancy management in (7) has been employed in order to avoid joint limits, kinematic singularities and to reduce the sliding track motions, being this last the slowest one.

A baseline of 30 cm is performed by the camera in 350 ms. Hence, for a fixed camera frame rate, the first trajectory estimation starts when a number of about $n_l = 40$ samples have been collected. In the employed set-up latency periods and delays between the robot control PC, the C4G control unit and the visual elaboration PC are present. An estimation of these parameters has been performed so as to synchronize the direct kinematic measurements with the visual data.

Several experiments have been carried out with different light conditions and pitchers. The percentage of ball interception evaluated over a set of 50 shots is about 90%, while the percentage of catching is about 70%. This difference is due to the poor performance of the available hand, which has very slow dynamics. This last is partially compensated by starting in advance the hand closing on the basis of the estimated catching time, but obviously some limitations remain. Other sources of inaccuracy are related to the noisy visual measurements due to the quick change of the illumination conditions along the shot path, especially when the ball is close to a light source on the ceiling. In Fig. 3 it is possible to observe the complete ball trajectory for a given throw,

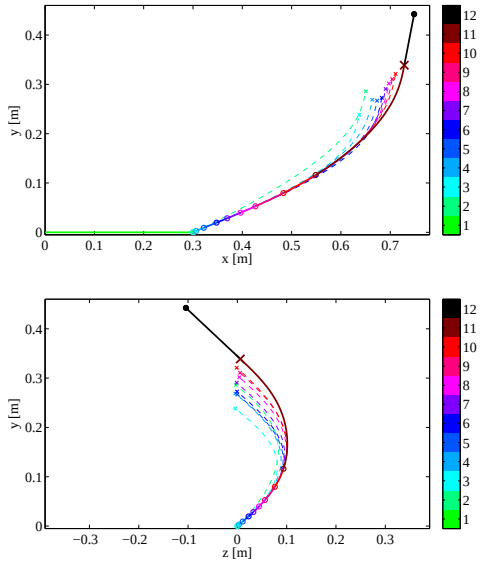

Fig. 4. Sequence of the interception points (cross) projected into the $(x-y)$ and $(z-y)$ planes. The dashed lines are the planned path, starting from the current hand position (circle). The continuous lines are the real path followed by the hand, starting with the initial baseline (green) and leading to the final catching path (black).

with the overlay of the robot motion.

In Fig. 4, with respect to the throw represented in Fig. 3, all the estimated positions, projected in the $(x-y)$- plane of the world frame, are represented with a cross point. The color bar identifies the ordered sequence of all the predicted interception points, while the biggest brown cross represents the final position in which the estimation has been considered stable. The dashed lines represent the planned path for the hand, which is achieved using (2) starting from the current motion state and leading to the current estimated interception position, while the continuous line is the real path followed by the hand, which starts with the baseline (green piece of the path) and ends with the final catching trajectory (black piece of the path): the big black dot represents the final configuration of the hand in which this last is still and the ball has been caught (see Section IV-C).

It is worth noticing that the first estimated point, the green one, is given by the linear estimation process. Further, since the palm of the adopted hand is a square with a side-length of about 10 cm, the estimation would have been considered as stable when the refinements become less than the half of such dimension. However, in the proposed experimental results a conservative threshold of 1 cm has been considered.

VI. Conclusion

A solution for the eye-in-hand robotic ball catching problem has been presented. A partitioned visual approach has been employed in order to move the robot and keep the ball in the camera view. The ball trajectory has been estimated through an iterative nonlinear optimization algorithm, which has been initialized by a fast linear estimation method. The approach has been demonstrated with experimental results.

References

[1] B. Hove and J. Slotine, "Experiments in robotic catching," in *American Control Conference*, (Boston), 1991.

[2] U. Frese, B. Bauml, S. Haidacher, G. Schreiber, I. Schaefer, M. Hahnle, and G. Hirzinger, "Off-the-shelf vision for a robotic ball catcher," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, (Maui), 2001.

[3] C. Borst, M. Fischer, S. Haidacher, H. Liu, and G. Hirzinger, "DLR hand II: Experiments and experiences with an anthropomorphic hand," in *IEEE International Conference on Robotics and Automation*, (Taipei), 2003.

[4] O. Birbach, U.Frese, and B. Bauml, "Realtime perception for catching a flying ball with a mobile humanoid," in *IEEE International Conference on Robotics and Automation*, (Shanghai), 2011.

[5] Y. Imai, A. Namiki, K. Hashimoto, and M. Ishikawa, "Dynamic active catching using a high-speed multifingered hand and a high-speed vision system," in *IEEE International Conference on Robotics and Automation*, (New Orleans), 2004.

[6] A. Namiki and M. Ishikawa, "The analysis of high-speed catching with a multifingered robot hand," in *IEEE International Conference on Robotics and Automation*, (Barcelona), 2005.

[7] C. Santos and M. Ferreira, "Ball catching by a Puma arm: A nonlinear dynamical systems approach," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, (Beijing), 2006.

[8] K. Deguchi, H. Sakurai, and S. Ushida, "A goal oriented just-in-time visual servoing for ball catching robot arm," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, (Nice), 2008.

[9] C. Lin and Y. Chiu, "The DSP based catcher robot system with stereo vision," in *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, (Xi'an), 2008.

[10] M. Riley and C. Atkeson, "Robot catching: towards engaging human-humanoid interaction," *Autonomous Robots*, vol. 12, no. 1, pp. 119–128, 2002.

[11] K. Nishiwaki, A. Konno, K. Nagashima, M. Inaba, and H. Inoue, "The humanoid Saika that catches a thrown ball," in *IEEE International Workshop on Robot and Human Communication*, (Sendai), 1997.

[12] M. Kitazawa, J. Wu, and Y. Sakai, "A new method of 3-d movie based on 2-d photo images for the virtual playing catch system," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, (Takamatsu), 2000.

[13] M. Bratt, C. Smith, and H. Christensen, "Minimum jerk based prediction of user actions for a ball catching task," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, (San Diego), 2007.

[14] C. Smith, M. Bratt, and H. Christensen, "Teleoperation for a ball-catching task with significant dynamics," *Neural Networks*, vol. 21, no. 1, pp. 604–620, 2008.

[15] S. Chapman, "Catching a baseball," *American Journal of Physics*, vol. 36, no. 10, pp. 868–870, 1968.

[16] S. Das and R. Das, "Using reinforcement learning to catch a baseball," in *IEEE International Conference on Neural Networks*, (Orlando), 1994.

[17] R. Mori and F. Miyazaki, "Examination of human ball catching strategy though autonomous mobile robot," in *IEEE International Conference on Robotics and Automation*, (Washington), 2002.

[18] R. Mori and F. Miyazaki, "GAG (gaining angle of gaze) strategy for ball tracking and catching task," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, (Lausanne), 2002.

[19] F. Takagi, H. Sakahara, T. Tabata, H. Yamagishi, and T. Suzuki, "Navigation control for tracking and catching a moving target," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, (St. Louis), 2009.

[20] E. Ribnick, S. Atev, and N. Papanikolopoulos, "Estimating 3D positions and velocities of projectiles from monocular views," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 5, pp. 938–944, 2009.

[21] R. Herrejon, S. Kagami, and K. Hashimoto, "Composite visual servoing for catching a 3-D flying object using RLS trajectory estimation from a monocular image sequence," in *IEEE International Symposium on Computational Intelligence in Robotics and Automation*, (Daejeon), 2009.

[22] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics. Modelling, Planning and Control*. London: Springer, 2008.

[23] K. Deguchi, "Optimal motion control for image-based visual servoing by decoupling translation and rotation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, (Victoria), 1998.