Robotic Ball Catching with an Eye-in-Hand Single-Camera System

Pierluigi Cigliano, Vincenzo Lippiello, *Member, IEEE,* Fabio Ruggiero, *Member, IEEE,* Bruno Siciliano, *Fellow, IEEE*

***Abstract*—In this paper, a unified control framework is proposed to realize a robotic ball catching task with only a moving single-camera (eye-in-hand) system able to catch flying, rolling and bouncing balls in the same formalism. The thrown ball is visually tracked through a circle detection algorithm. Once the ball is recognized, the camera is forced to follow a baseline in the space so as to acquire an initial data-set of visual measurements. A first estimate of the catching point is initially provided through a linear algorithm. Then, additional visual measurements are acquired to constantly refine the current estimate by exploiting a nonlinear optimization algorithm and a more accurate ballistic model. A classic partitioned visual servoing approach is employed to differently control the translational and rotational components of the camera. Experimental results performed on an industrial robotic system prove the effectiveness of the presented solution. A motion-capture system is employed to validate the proposed estimation process via ground truth.**

***Index Terms*—Robotic Ball catching, Ball detection, Ball tracking tracking, Real-time trajectory estimation, Partitioned visual servoing, Bouncing and rolling balls**

I. Introduction

CATCHING a thrown ball is often a good exercise to check human reflexes. The same task could be applied to an advanced robotic platform to test new control methodologies. Furthermore, such a challenging scenario covers several areas like fast visual detection and tracking, motion estimation, prediction and coordination, on-line trajectory planning, and so on. Hence, the ball catching task might contribute in paving the way towards the next generation of robots which should be provided with the above mentioned skills.

The task of catching a thrown ball can be generally split into three sub-problems: ball detection, trajectory estimation, and robot motion control. Several works are present in the literature covering each of the aforementioned sub-problems. Usually, high-speed stereo vision system is required to solve the first sub-problem. Through the approach proposed in this paper, partially introduced in [1]–[3], it is shown both from a theoretical and practical point of view that the problem of catching a thrown ball can be solved by using only one camera. Moreover, in this way, the cost of the overall equipment can be reduced. Even if recently many low cost stereo systems are available off-the-shelf, the computational cost to elaborate at high frame rate (e.g., about 140 Hz) is still demanding in the ball catching task and this can be lowered by using a monocular system, saving computational resources. Moreover, with respect to previous works, the estimate of the trajectory is improved in this paper considering also bouncing and rolling balls within the same framework.

Each year an intensive progress in ball tracking and catching tasks is achieved by the RoboCup competition; many table tennis robots are challenging humans with great outcomes; plenty of videos can be found on the net showing different robotic ball catching techniques. Even mobile humanoid robots, with a stereo-camera system, have been used in such a task [4], [5]. Nevertheless, authors do believe that the first case of a robotic ball catcher employing only a single moving camera and coping with rolling, bouncing and flying balls in the same framework, i.e., without changing either the estimator or the control law, is presented in this paper. In more detail, a standard industrial robot manipulator is equipped with a CCD camera mounted directly on the manipulator end-effector (*eye-in-hand* configuration). Differently from [1]–[3], the ball is recognized through a circle detection algorithm based on the method developed in [6]. One of the novelties of this paper is the improvement of the approach proposed in [6] along the lines of the work introduced in [7], so as to provide a sub-pixel accuracy to have a measure of the ball centroid that ameliorates the trajectory estimation process. Therefore, the proposed estimator is composed of a continuous refinement of the ball interception point through a nonlinear algorithm, whose initial starting condition is provided by a fast linear estimation process. The initial camera motion is thus commanded along a suitable baseline so as to collect a sufficient initial number of visual data from different points of view and provide such initial estimate. As another novelty, the proposed ball trajectory estimator extends what already presented by the authors [1], [3] since now the process is able to cope with rolling and bouncing balls too. A statistical reliability test of the measures has been also introduced to discard possible outliers in the measurements set. Last, but not least, the employed control law is more detailed than in [1], [3], as well as the policy employed for the intersection point selection. Related stability proofs are now provided. Experimental results demonstrate the effectiveness of the presented solution. The performance of the proposed trajectory estimator is shown

The research leading to these results has been supported by the RoDyMan project, which has received funding from the European Research Council (FP7 IDEAS) under Advanced Grant agreement number 320992. The authors are solely responsible for its content. It does not represent the opinion of the European Community and the Community is not responsible for any use that might be made of the information contained therein.

Authors are listed in alphabetical order.

Vincenzo Lippiello, Fabio Ruggiero and Bruno Siciliano are with the PRISMA Lab, Dipartimento di Ingegneria Elettrica e Tecnologie dell'Informazione, Università degli Studi di Napoli Federico II, via Claudio 21, 80125, Naples, Italy.

Corresponding author's email fabio.ruggiero@unina.it

through a comparison with the ground truth provided by a motion-capture system.

The outline of the paper is as follows. The related work is presented in the next section. A general overview of the proposed algorithm is given in Section III. The algorithms for the ball detection and tracking are analysed in Section IV. The ball trajectory estimator and interception policies are described in Section V. Section VI is devoted to describe the replanning of the robot path. The partitioned visual servoing is revised in Section VII. The employed set-up is introduced in Section VIII and the experiments are critically discussed. Section IX gives concluding remarks.

II. Related work

A brief literature review is now presented. It is clear from the following that very few papers address the monocular case in ball detection and trajectory estimation operations. Moreover, in this paper, the camera is also moving during the tracking/estimation since it is mounted in an eye-in-hand configuration. Modifying and/or employing some methods already presented in the literature concerning ball detection, trajectory estimation and visual servoing, authors have achieved the goal of catch a flying/bouncing/rolling ball through a single moving camera system within the same framework.

A. Ball detection

Working in unstructured (or at least partially structured) environments requires recognizing some objects and key features in the scene [8]. Detecting a ball, or in general a thrown object, is thus crucial in robotic catching applications.

A first distinction can be made on the basis of the number of the employed cameras in the visual system. On the one hand, a stereo visual system benefits by using the triangulation method to reconstruct the 3D position of the ball in the scene [6], [9]–[11], but requires a more accurate calibration procedure and sophisticate elaboration hardware. On the other hand, a monocular visual system has an easier calibration procedure, but more effort has to be put in the 3D reconstruction of the scene [12]–[16]. Besides this, in a monocular visual system, a classification about the position of the camera with respect to the robot can be taken into account. An eye-to-hand configuration is considered in [13], while the camera is mounted in an eye-in-hand configuration in [1], [2].

Another distinction can be made about the employed techniques to detect the thrown object. By using a threshold method, the difference between the actual image and some reference images is employed in [17]. An equalized color-based clustering in the HSL color space is considered in [1], [2], while a circular gradient method to detect the ball in the images is utilized in [6].

Other objects rather than balls can be thrown and detected in the image. The work in [18] can be taken as a starting point to investigate further aspects and details.

B. Trajectory estimation

The problem about motion estimation has been addressed in several ways. This subsection takes into account the estimation performed by using a visual system in the particular context of the ball catching. Hence, in order to predict the correct catching point where the robot should intercept the thrown object, the motion trajectory of this last has to be estimated.

A 2D task is defined in [19] on the image plane of each available camera. The robotic arm may catch the ball if such 2D tasks are achieved simultaneously. The 3D position of the ball can be instead estimated by resorting to an extended Kalman filter (EKF) based on a Newtonian system which also considers the effect of the air drag [17]. Six nonlinear regression methods are compared in [20] to estimate translational and rotational velocities of free-flying objects.

By starting from a set of images taken from a monocular system, the estimate of the motion trajectory of a thrown object can be performed by using a least squares method [16] (without modeling the air resistance). A least squares solution is considered in [2], [13], [21], too.

Many estimators are based on the so-called Chapman's strategy –the fielder should run at a proper speed to maintain a constant increasing rate of the tangent of the ball's elevation angle [22]– for the ball catching [14], [15], [23].

When the ball hits the ground, or other objects, the estimate becomes more complicated due to the rebound effects. As an example, this condition always appears during table tennis games. In order to take into account not only the energy loss of the ball after the collision with the ground, but also the air resistance, the visual measurements errors and the friction between the ball and the ground, a first-order polynomial describing the bouncing model is employed in [24]. Another model relating flying and self-rotational velocities just before and after the rebound is considered in [25]. The lift and drag aerodynamic effects of the rebound are examined in [26], while the bouncing phenomenon between a ping-pong ball and both the rigid table and the racket rubber are exploited in [27], [28]. A 3D model for rigid body impact with tangential compliance, represented through springs, is presented in [29].

Finally, other estimators either make use of proper neural networks [30], or follow a programming-by-demonstration approach to find a feasible catching configuration in a probabilistic manner [31].

C. Robot motion control

On the basis of the configuration of the visual system, the robot controller has to take into account either the ball tracking, or the ball interception or both the tasks.

The most common visual servoing approaches are the position-based and image-based methods [32]. A combination of these lasts is used in [12] to catch a ball moving on a table, and in [13] to solve the complete 3D task. A dynamic version of the aforementioned approaches is proposed in [33]. Decoupling translational and angular components in a visual servoing task is employed in partitioned visual approaches [34]–[36].

The importance in regulating the impedance of the hand, or the arm, during a ball catching action is highlighted in [37]: a human could miss the ball when the arm is both stiff beyond necessity and too compliant.

A non-prehensile way to manipulate the thrown ball is considered in [21]. After the dynamic catch, a balancing

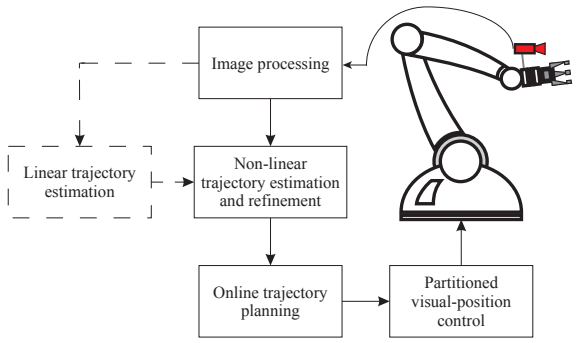


Fig. 1. Block diagram of the proposed monocular robotic ball catching system. The Linear Trajectory Estimation block (dashed lines) is executed just once during the robot starting motion.

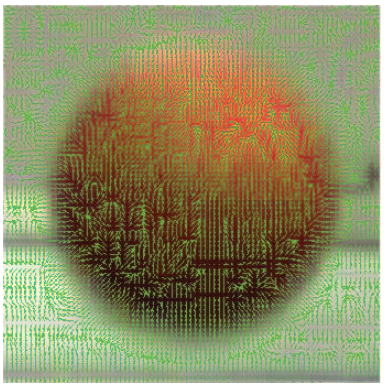


Fig. 2. Example of the contrast-normalized Sobel-filter matrix.

controller is considered to keep the ball on a plate mounted on the robot end-effector.

III. Algorithm Overview

The overview of the proposed method for catching a thrown ball in the 3D space with a monocular visual system is shown in the block diagram of Fig. 1. Once the ball is detected by the visual system, the camera mounted in an eye-in-hand configuration is forced to follow a suitable baseline in the 3D space. A partitioned visual-servoing control is employed to both keep the ball in the field of view of the camera (through the camera orientation) and follow the planned path (camera position). The starting baseline is performed to ensure a well-conditioned estimation problem and collect/process visual information so as to get a first prediction of the ball trajectory with a rough linear estimate. This prediction is employed as a starting point for a more precise nonlinear refinement process of the trajectory, that also considers the case of rolling and bouncing balls. When a new estimate of the robot interception pose is available, the on-line motion planner smoothly switches its target to the new one, always keeping the ball in the camera field of view. Hence, the visual measurements are continuously acquired and processed by the nonlinear optimization algorithm. Finally, when the continuous refinement does no longer improve the prediction of the trajectory significantly, the final catching pose of the robot can be computed. In order to accommodate the ball into the robotic gripper, the robot kinematics is taken into account.

IV. Visual Ball Detection and Tracking

The presence of a ball in the camera's field of view is evaluated with an algorithm derived from the circular-shape detection proposed in [6], [38]. The main advantages with respect to the classical Hough transform are the absence of decision thresholds or parameters calibration, and the high robustness with respect to changes in lighting conditions.

Let $\boldsymbol{I}(X_I, Y_I)$ denote the light intensity $(m \times n)$ matrix provided by the camera, where X_I and Y_I represent the pixel coordinates of the image sensor. The circular response function $\mathcal{C}_r(X_I, Y_I) \in [0, 1]$ considered in this paper represents the degree of affinity between the circular region with radius r of the image at the point (X_I, Y_I) with respect to a radial intensity gradient (i.e. it represents the average fraction of the radial intensity gradient on the circle). In details, this affinity is established with respect to the gradient of the image intensity at the border of the considered circular region with the following expression

$$\mathcal{C}_r(X_I, Y_I) = \frac{1}{2\pi} \int_0^{2\pi} \left(\begin{bmatrix} c_\alpha \\ s_\alpha \end{bmatrix} \boldsymbol{C} \left(\begin{bmatrix} X_I + r c_\alpha \\ Y_I + r s_\alpha \end{bmatrix} \right) \right)^2 \mathrm{d}\alpha, \tag{1}$$

where $c_\alpha = \cos(\alpha)$ and $s_\alpha = \sin(\alpha)$. The matrix $\boldsymbol{C}$ represents the contrast-normalized Sobel filter[1], which is defined as follows [6]

$$\boldsymbol{C} = \frac{\sqrt{2} \begin{bmatrix} \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * \boldsymbol{I} & \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * \boldsymbol{I} \end{bmatrix}^{\mathrm{T}}}{\sqrt{16 \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} * \boldsymbol{I}^2 - \left(\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} * \boldsymbol{I} \right)^2 + \epsilon^2}}, \tag{2}$$

where $\epsilon = 1$ is the discretization unit of pixel intensity preventing singularities in constant image areas, and the operator $*$ represents the linear convolution. Equation (1) is thus the integral of the squared scalar product of the contrast-normalized Sobel filter with the radial direction. For more details see [6], [38]. Moreover, notice how the matrix $\boldsymbol{C}$ has two different channels, namely $\boldsymbol{C} = \begin{bmatrix} \boldsymbol{C}_X & \boldsymbol{C}_Y \end{bmatrix}^{\mathrm{T}}$, with the same size of $\boldsymbol{I}$. Figure 2 shows the contrast-normalized Sobel-filter matrix of a ball in an image. Notice the radial distribution of $\boldsymbol{C}$ close to the border of the ball.

Let be $\mathcal{A}_r$ the affinity matrix for a given radius r, which is defined as follows

$$\mathcal{A}_r = \begin{cases} \mathcal{C}_r(X_I, Y_I) & (X_I, Y_I) \in \Xi_r \\ 0 & \text{otherwise} \end{cases}, \tag{3}$$

where $\Xi_r = [r, \ldots, m-r-1] \times [r, \ldots, n-r-1]$. The pixels of the image that are possible candidates as the center of a circle are evaluated by applying a threshold $\alpha_{\min}$ to the set of affinity matrices in the range of radius of interest $r^- \leq r \leq r^+$, with $0 < r^- < r^+ < (\min(m, n) - 3)/2$. In Fig. 3 the flowchart

[1]Differently from the classical Sobel operator, the vector length in the contrast-normalized Sobel filter indicates the gradient purity rather than the gradient intensity.

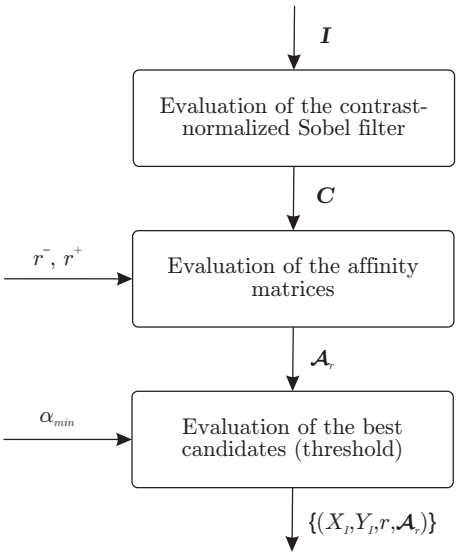


Fig. 3. Flowchart of the ball detection visual process.

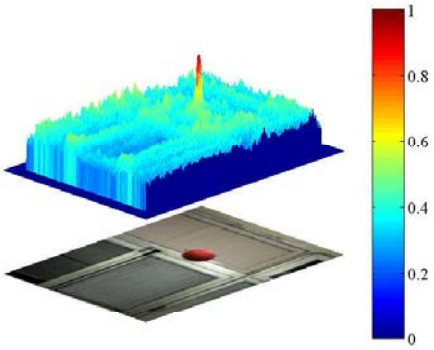


Fig. 4. Representation of the affinity matrix of an image corresponding to the maximum circular correspondence.

of the ball detection process is represented, while a graphical representation of the affinity matrix of an image corresponding to the maximum circular correspondence is shown in Fig.4.

A. Affinity matrix evaluation

The evaluation of the affinity matrix $\mathcal{A}_r$ strongly affects the overall performance of the ball detection algorithm. Hence, more details on the adopted approach are provided in this subsection.

Notice that discretizing $\mathcal{C}_r(X_I, Y_I)$, with $(X_I, Y_I) \in \Xi_r$, is equivalent to the arithmetic mean of the values of $\boldsymbol{C}$ in correspondence of the points of any circular region $\Psi_r(X_I, Y_I)$ of radius r and centered in (X_I, Y_I). Let $\widehat{\boldsymbol{C}}_r(X_I, Y_I)$ denote the square region of $\boldsymbol{C}$ with $2r+1$ elements for each side and centered at (X_I, Y_I). Then, the contribution of the element at point $P \in \Psi_r(X_I, Y_I)$ is $(\widehat{\boldsymbol{C}}_r(X_I, Y_I)\boldsymbol{r}_P)^2$, where $\boldsymbol{r}_P$ is the radial unit vector pointing from (X_I, Y_I) to P. Notice that the number of unit vectors employed does not depend on the points where $\mathcal{A}_r$ is evaluated. By considering the square dual-

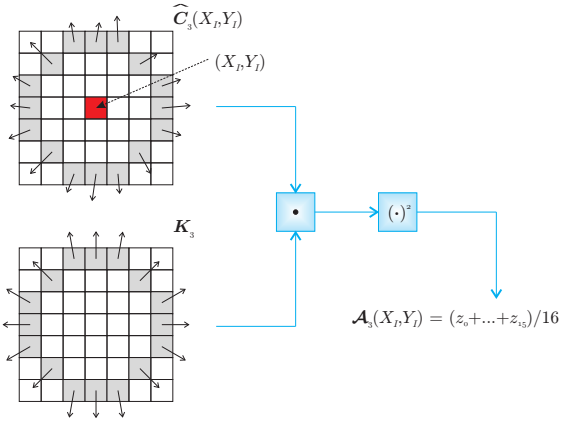


Fig. 5. Representation of the evaluation of the affinity matrix in the case $r = 3$.

channel matrix $\boldsymbol{K}$ of dimension $2r+1$ defined as follows

$$\boldsymbol{K}_r(X'_I, Y'_I) = \begin{cases} \boldsymbol{r}_{(X'_I, Y'_I)} & (X'_I, Y'_I) \in \Psi_r(X_I, Y_I) \\ 0 & \text{otherwise} \end{cases}, \quad (4)$$

the affinity matrix $\mathcal{A}_r$ can be evaluated with a linear convolution of the matrix $\boldsymbol{C}$ with $\boldsymbol{K}_r$. Figure 5 shows the evaluation of $\mathcal{A}_3$ by using the kernel $\boldsymbol{K}_3$. On the other hand, the evaluation of the points on a circumference of radius r is required to evaluate the kernel $\boldsymbol{K}_r$. To this purpose, the algorithm proposed in [39] is employed with the advantage of considering only integer values and reflection properties.

B. Parallelization

The number of operations required for the evaluation of the affinity matrices rapidly grows with the dimension of the image (e.g., the evaluation of the affinity matrix for an image of (512×368) pixels requires about 10^9 operations). In order to reduce the computational time, several arrangements should be introduced. The kernels required for the evaluation of the affinity matrices are pre-computed during the initialization phase of the algorithm. Moreover, the sparse nature of these matrices requires a specific sparse matrix representation.

The evaluation of $\boldsymbol{C}$ is performed by parallelizing the computation of its two channels. Moreover, each channel can be processed by parallelizing the horizontal and the vertical components of the gradient of $\boldsymbol{I}$ via the Sobel operator. Finally, also the denominator of (2) is evaluated in parallel with respect to the previous contributes, where the non-constant terms are evaluated by a linear convolution.

By employing specific SIMD (Single Instruction Multiple Data) operations, the evaluation of the affinity matrix is parallelized by rows. Thanks to the symmetry of the employed kernels it is possible to assume, without loss of generality, that the non-zero elements are $4p$, with p any integer number. Hence, for $r = r^-, \ldots, r^+$, and $(X_I, Y_I) \in \Xi_r$, it is possible to obtain

$$\mathcal{A}_r(X_I, Y_I) = \frac{1}{4p} \sum_{k=0}^{4p-1} \left(u_{k_x} w_{k_x} + u_{k_y} w_{k_y}\right)^2, \quad (5)$$

where $\boldsymbol{u}_k = \begin{bmatrix} u_{k_x} & u_{k_y} \end{bmatrix}^{\mathrm{T}}$ is an element of $\boldsymbol{C}$, and $\boldsymbol{w}_k = \begin{bmatrix} w_{k_x} & w_{k_y} \end{bmatrix}^{\mathrm{T}}$ is an element of $\boldsymbol{K}_r$, respectively. By denoting with $q_k = \boldsymbol{u}_k^{\mathrm{T}} \boldsymbol{w}_k$, with $k = 0, \ldots, 4p-1$, Equation (5) can be rewritten as follows

$$\mathcal{A}_r(X_I, Y_I) = \frac{1}{4p} \sum_{i=0}^{p-1} z_i, \tag{6}$$

where $z_i = q_{4i}^2 + q_{4i+1}^2 + q_{4i+2}^2 + q_{4i+3}^2$. This latter term can be evaluated as a scalar product that can be computed by using an intrinsic function available on modern processors.

C. Pyramidal approach and windowing

The localization of a shape into an image with algorithms of template-matching requires a high number of calculations because, in general, it is necessary to explore the entire image and repeat the search several times for different size of the shape. Moreover, with the extension of the shape, the number of calculations rapidly increases. In order to reduce the processing time, a pyramidal approach can be employed. The technique consists in generating a copy in half resolution of the image by selecting the rows and columns of odd place. Iterating this process creates a number of images reproducing the same scene but with a lower level of detail. The loss of resolution can be recovered by using the results achieved on a low-resolution image to recognize the region of interest where perform the search within the original image.

Once that a thrown ball has been detected for the first time, a windowing technique is employed. In details, a suitable region of interest for the searching algorithm can be set up on the base of the position, radius, and velocity of the ball measured in the previous sequence of images. With this approach, a square searching region is positioned on the current image on the basis of the previous ball position and velocity, while its size depends on the previous radius and a suitable safety factor. Also the radius range of search is windowed starting from the previous estimate and ball direction.

D. Sub-pixel refinement

The results achieved with the previous ball detection algorithm can be improved thanks to the use of a post-elaboration process resulting in a sub-pixel accuracy. For each image, it is possible to observe that the intensity of the pixels, lying on an outgoing radial direction from the estimated ball center (see previous subsections), presents a distribution of values with an inflection point (Fig. 6). A linear searching algorithm can be employed to find the exact position $\boldsymbol{p}_{\theta_i} = \begin{bmatrix} x_{\theta_i} & y_{\theta_i} \end{bmatrix}^{\mathrm{T}}$, $i = 0, \ldots, n_d$, of the ball contour along a finite number n_d of radial directions spanning the whole circumference.

Once that a number of ball contour measurements are available, a new circle with center (x_c, y_c) and radius r_c, all with sub-pixel accuracies, is evaluated. In detail, the equation of a circle in a plane is represented by $x_{\theta_i}^2 + y_{\theta_i}^2 + \xi_1(x_c) x_{\theta_i} + \xi_2(y_c) y_{\theta_i} + \xi_3(x_c, y_c, r_c) = 0$. Stacking all the data, with $i = 0, \ldots, n_d$, a least squares approach can be then employed to estimate ξ_1, ξ_2 and ξ_3 from which in turn the unknowns can be estimated as $x_c = -0.5\xi_1$, $y_c = -0.5\xi_2$, $r_c = \sqrt{0.25 * (\xi_1^2 + \xi_2^2) - \xi_3}$. Measurements that are statistically incoherent are discarded, and the estimation process is repeated until all measurements are reliable. In particular, the distance d_{θ_i} from each point $\boldsymbol{p}_{\theta_i}$ and the above estimated circle with center (x_c, y_c) and radius r_c is evaluated and saved in an array. The mean and standard deviation of such an array are then calculated. The points whose distance d_{θ_i} outperforms the standard deviation by a certain factor are discarded and the above described least squares approach is performed again with the surviving points until no elements are rejected.

Several examples of the achieved results are shown in Fig. 7. Moreover, it has been verified during the experimental tests that the proposed algorithm reduces the effect of the blur around the contour of the ball, caused by the high-velocity motion of the ball in the image, and improves the accuracy of the estimation process (see Sections V and VIII-B). Notice that other methods might be employed for sub-pixel accuracy for moving cameras [7].

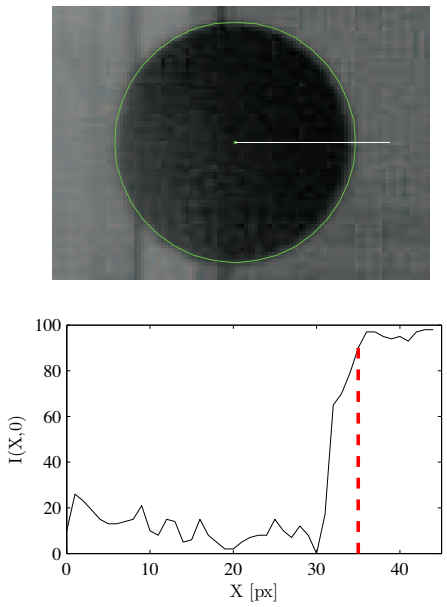


Fig. 6. Light intensity (on the bottom) along a radial direction from the estimated center of a ball (on the top).

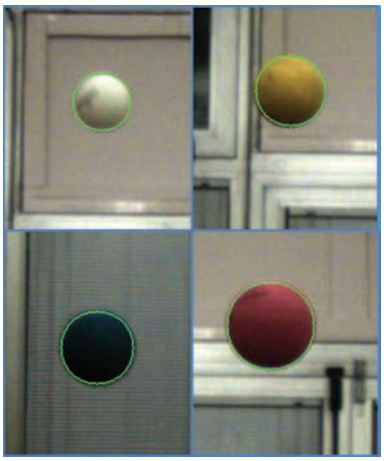


Fig. 7. Sub-pixel refinement examples.

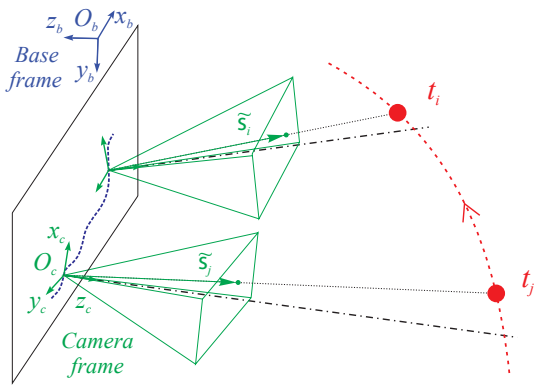


Fig. 8. The camera reference frame Σ_c is shown in two different sample times, t_j and t_i. The ball trajectory is shown with a red dotted line, while in blue is represented the corresponding camera trajectory.

V. Ball Trajectory Estimation

Let Σ_b, Σ_e and $\Sigma_c = O_c$–$x_c y_c z_c$ be the robot base frame, fixed with respect to the ground, the end-effector frame and the camera frame, respectively (see Fig. 8). Since Σ_e and Σ_c are fixed with respect to each other, only Σ_c is considered into the remainder of the paper. The camera optical axis is then aligned with the approaching axis of Σ_e.

Let $\boldsymbol{s} = \begin{bmatrix} X & Y \end{bmatrix}^{\mathrm{T}}$ be the normalized image coordinates vector of the centroid of the ball, i.e., the center of the circle detected in Section IV. By denoting with $\tilde{\boldsymbol{s}} = \begin{bmatrix} \boldsymbol{s}^{\mathrm{T}} & 1 \end{bmatrix}^{\mathrm{T}}$, the position of the ball center with respect to Σ_c is

$$\boldsymbol{p}_o^c = \begin{bmatrix} x^c & y^c & z^c \end{bmatrix}^{\mathrm{T}} = z^c \begin{bmatrix} X & Y & 1 \end{bmatrix}^{\mathrm{T}} = z^c \tilde{\boldsymbol{s}}. \tag{7}$$

A. Initial baseline and linear initialization

A classic static triangulation method cannot be adopted in the proposed scenario because of the single employed camera. Hence, the proposed process estimates the 3D trajectory of the ball interpolating 2D visual measurements along the time.

As soon as the ball is detected for the first time, the camera is commanded to move along a straight line, i.e., the initial baseline, in the 3D Cartesian space with high velocity. To gain better results, such initial movement should be orthogonal to the direction of the throw. The orientation of the camera is controlled to keep the ball in its field of view (see Section VII). The visual data collection along this path leads to a well-conditioned problem. Once this first data-set has been acquired, it is possible to start the procedure for the linear initialization. This last is explained in [2], hence only a brief description is here addressed.

Let t_k be the k-th visual sample time, $\tilde{\boldsymbol{s}}_k$ the corresponding measured image feature vector, $\boldsymbol{p} = \begin{bmatrix} x & y & z \end{bmatrix}^{\mathrm{T}}$ the points belonging to the camera *optical ray* and passing through the current origin of the camera $\boldsymbol{c}_k = \begin{bmatrix} c_{x,k} & c_{y,k} & c_{z,k} \end{bmatrix}^{\mathrm{T}}$. The feature vector $\boldsymbol{r}_k = \begin{bmatrix} r_{x,k} & r_{y,k} & r_{z,k} \end{bmatrix}^{\mathrm{T}} = \boldsymbol{c}_k + \boldsymbol{R}_{c,k}\tilde{\boldsymbol{s}}$ can be defined by the following equations representing a straight line in the 3D space (see Fig. 8)

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

where $\boldsymbol{R}_{c,k}$ is the rotation matrix of Σ_c with respect to Σ_b at time t_k. Both $\boldsymbol{c}_k$ and $\boldsymbol{R}_{c,k}$ are provided by the robot direct kinematics.

A simplified model of the ball trajectory is considered during this initialization phase, i.e. the effect of the air drag to the motion of the ball is neglected:

$$\boldsymbol{p}_o(t) = \boldsymbol{p}_0 + \boldsymbol{v}_0 t + 0.5\boldsymbol{g}t^2, \tag{9}$$

with $\boldsymbol{p}_o$ the (3×1) position vector of the ball with respect to Σ_b, $\boldsymbol{g}$ the gravity acceleration, $\boldsymbol{p}_0$ and $\boldsymbol{v}_0$ the (3×1) vectors of the initial position and velocity of the ball ($t = 0$), respectively, and corresponding to the time of the first ball detection. Notice that, without loss of generality, the gravity acceleration is aligned to the axis y of the chosen Σ_b, i.e., $\boldsymbol{g} = \begin{bmatrix} 0 & g & 0 \end{bmatrix}^{\mathrm{T}}$ with $g = 9.81\mathrm{m/s}^2$.

At each t_k, the optical ray intersects the ball trajectory. Folding (9) into (8) yields a system of 2 equations into 6 unknowns, $\boldsymbol{p}_0$ and $\boldsymbol{v}_0$, that fully describes the trajectory of the ball. Stacking the n_l measurements into rows yields a system of n_l equations into 6 unknowns that can be solved through a least squares solution. Additional considerations about this stage are given in [1], [2].

The first interception point candidate is then computed. This allows the robot to reach the predicted interception position, whose computational details are provided in Section V-E, at the estimated catching time. The robot path is replanned as described in Section VI. The rotational part of the camera, instead, is kept free to track the ball in order to acquire more visual measurements during the movement. The estimate provided by this linear algorithm is employed as a starting point for the next stage.

B. Nonlinear estimation and continuous refinement

Meanwhile the previous linear estimation process gives the result, new visual measurements are collected. Afterwards, both these new visual measurements and the old ones are employed in a nonlinear estimation process that starts initially from the result obtained by the linear method.

In details, let $\boldsymbol{s}_k$ be the centroid of the ball acquired at a time t_k, the cost function to minimize is

$$\min_{\boldsymbol{p}_0, \boldsymbol{v}_0} \sum_{k=1}^{n_i} \left\| \frac{1}{\tilde{z}_k^c} \begin{bmatrix} \tilde{x}_k^c \\ \tilde{y}_k^c \end{bmatrix} - \boldsymbol{s}_k \right\|, \tag{10}$$

with n_i the current number of available visual measurements at the current estimation time t_i, and $\tilde{\boldsymbol{p}}_k^c$ the (3×1) estimated position vector of the ball with respect to Σ_c, which is defined as follows

$$\tilde{\boldsymbol{p}}_k^c = \begin{bmatrix} \tilde{x}_k^c & \tilde{y}_k^c & \tilde{z}_k^c \end{bmatrix}^{\mathrm{T}} = \boldsymbol{R}_{c,k}^{\mathrm{T}} \left(\tilde{\boldsymbol{p}}_k - \boldsymbol{c}_k \right). \tag{11}$$

The estimated position of the ball $\tilde{\boldsymbol{p}}_k(\boldsymbol{p}_{0,i-1}, \boldsymbol{v}_{0,i-1}, t_k)$ is numerically obtained by integrating the following ballistic model [17]

$$\ddot{\boldsymbol{p}}_o(t) = \boldsymbol{g} - \frac{c_w \pi d_b^2 \rho_a}{2 m_b} \parallel \dot{\boldsymbol{p}}_o(t) \parallel \dot{\boldsymbol{p}}_o(t), \tag{12}$$

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

of the ball. Hence, the model in (12) includes the air drag, and its numeric integration is sequentially performed in the time intervals $[t_{k-1}, t_k]$, where $k = 1, \ldots, n_i$, $t_0 = 0$, with initial conditions $\boldsymbol{p}_{0,i-1}$ and $\boldsymbol{v}_{0,i-1}$.

By knowing the altitude of the floor with respect to Σ_b, it is possible to detect whether the ball hits the ground during the above mentioned numerical integration. Without loss of generality, let $\boldsymbol{p}_f = \begin{bmatrix} 0 & \gamma & 0 \end{bmatrix}^{\mathrm{T}}$ be a point of the floor surface, which is horizontally placed with respect to Σ_b, where γ represents the height of the ground. The adopted bouncing model is the following [24]

$$\boldsymbol{v}_{out} = \boldsymbol{K}_b \boldsymbol{v}_{in} + \boldsymbol{b}_b, \tag{13}$$

in which $\boldsymbol{K}_b$ is a (3×3) diagonal matrix of coefficients representing the loss of energy with respect to the (3×1) vector of the ball velocity $\boldsymbol{v}_{in}$ before the rebound, $\boldsymbol{v}_{out}$ is a (3×1) vector of the ball velocity after the rebound, and $\boldsymbol{b}_b$ is a (3×1) vector of coefficients added to refine the model taking into account, for instance, the effects of the air resistance, vision measure error, and the spin caused by friction on the ground. Hence, by defining $\boldsymbol{e}_2 = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^{\mathrm{T}}$, at each integration time the condition $\boldsymbol{e}_2^{\mathrm{T}} \tilde{\boldsymbol{p}}_k > \boldsymbol{e}_2^{\mathrm{T}} \boldsymbol{p}_f$ is verified during the forward integration of the model (12). When a rebound is detected, the current velocity is changed accordingly to (13).

The purpose of minimizing (10) means that the initial conditions of the ballistic model are tuned to generate an estimated trajectory of the ball that minimizes the distance between the predicted projection of the ball onto the image plane and the corresponding measured observations of the ball along the time.

In practice, the minimization of the cost function (10) is performed using the *Levembert-Marquardt* algorithm. The result at the estimation time t_i is the updated values of $\boldsymbol{p}_{0,i}$ and $\boldsymbol{v}_{0,i}$. As long as the current estimate of the catching time is sufficiently far from the actual time instant, if new measurements have been acquired then the estimation process restarts using the current estimation as starting solution, otherwise $\boldsymbol{p}_{0,i}$ and $\boldsymbol{v}_{0,i}$ become the final estimated results $\boldsymbol{p}_0$ and $\boldsymbol{v}_0$. Hence, such refinement process stops when the current estimate catching time is approaching with respect to the grasping time required by the available gripper. As soon as the new interception point is available, the robot path is replanned as described in Section VI.

C. Statistical reliability test

Differently from [1], a statistical procedure to deal with the presence of image noise is also proposed. During its motion, the ball can be subject to different illumination/shadow conditions that could generate different levels of noise in the measurements dataset. In details, once the minimization process ends at t_i, the mean error, the standard deviation and the contribution of each visual measurement $\boldsymbol{s}_k$ to the error residual are evaluated. The visual measurements that contribute to the error residual outperforming the standard deviation by a certain factor are temporarily excluded for the next estimation process at time t_{i+1} (*update outliers list* in Fig. 9). In particular, new measurements should be available at

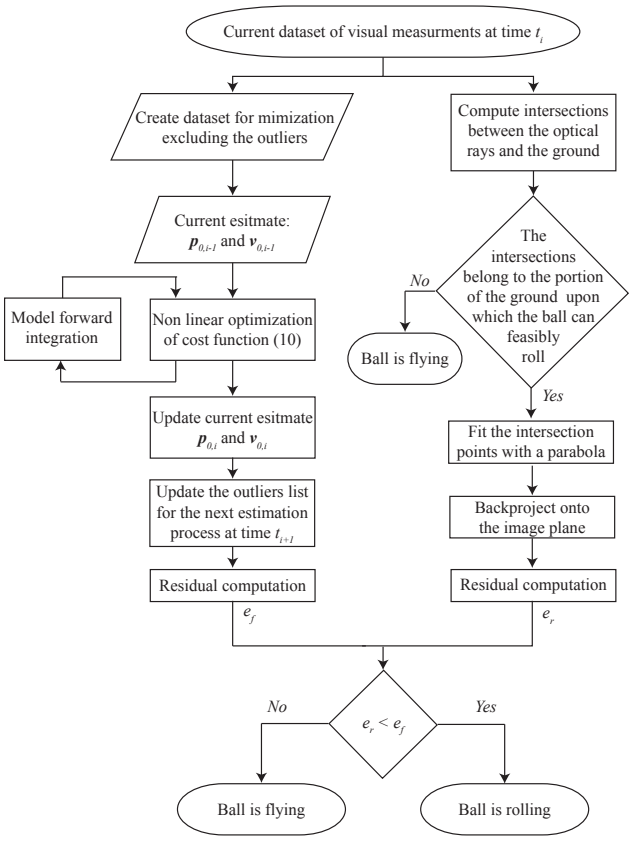


Fig. 9. Schematic flowchart of the implemented estimator, frozen at time t_i, to decide whether the ball is rolling or not.

the time in which such nonlinear estimation process computes the updated values of $\boldsymbol{p}_{0,i}$ and $\boldsymbol{v}_{0,i}$, i.e. the interception time/point. This new data set (without the measurements which have been classified as outliers at the previous estimation time) is employed during the current nonlinear refinement. Again, once the minimization process ends, all the visual measurements (even the outliers) that contribute to the error residual in a way that is not statistically coherent are excluded for the next optimization. Notice that in this way it should be possible to recover also measures that have previously considered as outliers due to a rough initial estimation.

This arrangement is hence able to improve significantly the accuracy of the estimation process when noisy visual measurements are available without losing significant measurements. Some figures regarding such improvement are given in Section VIII.

D. Rolling balls

In parallel to the nonlinear estimation refinement process, a condition is continuously tested to detect whether the balling is rolling or not on the ground. A schematic representation is provided in Fig. 9.

As previously introduced, for each acquired visual measurement $\tilde{\boldsymbol{s}}_k$ it is possible to associate the equations of the related optical ray (8). By knowing the position and orientation of the floor with respect to Σ_b, it is possible to compute the (3×1) intersection point $\boldsymbol{p}_{int,k}$ between the straight line (8) and the

ground. Afterwards, it is possible to fit the available points $\boldsymbol{p}_{int,k}$, with $k = 1, \ldots, n_i$, through a polynomial curve on the ground plane. Notice that this operation is performed only if each point $\boldsymbol{p}_{int,k}$ belongs to the portion of the ground upon which the ball can feasibly roll. In fact, if the ball is flying, the observation angle of the optical ray will produce interception points with the floor that are not located in such portion of the ground.

Without loss of generality, in order to take into account both the ball spin and the effects of the friction on the floor, a parabolic path has been chosen as fitting curve. Hence, the three coefficients of the parabola that best fits all the ground points can be retrieved through a least squares solution. Once the parabola has been defined, the points $\boldsymbol{p}_{cur,k}$ closest to the curve at point $\boldsymbol{p}_{int,k}$ are computed. Notice that in case of perfect fitting $\boldsymbol{p}_{cur,k} = \boldsymbol{p}_{int,k}$, with $k = 1, \ldots, n_i$. By knowing $\boldsymbol{c}_k$ and $\boldsymbol{R}_{c,k}$, each point $\boldsymbol{p}_{cur,k}$ is then back projected onto the image plane obtaining the feature vector $\tilde{\boldsymbol{s}}_{cur,k}$. The norm of the residual error between the visual measurements and these ones derived by the estimated model is employed as a quality measure. If the residual error achieved with this last method is less than the residual of (10), the ball is considered as rolling on the floor.

In case of rolling, the catching candidate point $\boldsymbol{p}_\times$ is computed as the intersection between the ground parabolic path and a predetermined goal line in the robot workspace. The camera's orientation is kept parallel to the ground, the gripper is positioned at a safety distance from the floor and its fingers are held open so as to lightly touch the floor. The final catching time $t_\times$ is determined by the timing of the measurement on the parabolic path, since at each point $\boldsymbol{p}_{cur,k}$ is associated a time t_k. On the basis of the above considerations, the robot path is replanned as described in Section VI.

E. Catching point selection

In case the estimation process does not detect a rolling ball, for which it has been already described how the interception pose is selected, the current catching point candidate $\boldsymbol{p}_\times$ is evaluated with respect to the actual configuration along the current estimated trajectory of the ball as deeply explained in the following. The catching time $t_\times$ and the ball velocity $\boldsymbol{v}_\times$ at $\boldsymbol{p}_\times$ can be evaluated from both the actual predicted trajectory and the robot kinematic model.

In detail, the current estimated path $\mathcal{T}$ belonging to the working space of the robot arm is partitioned in several candidate catching points $P_i \in \mathcal{T}$ by a fixed step. For each P_i, the inverse kinematics of the robot is calculated to compute the joint position of the robot arm. The inverse kinematics is computed through a closed-loop algorithm (CLIK) [32][2]. A set of quality indices can be considered for the selection of the best catching point. In this paper, joint limits and a manipulability measure are suitably coupled through a convex linear combination (see [32] for more details about the adopted quality indices). The candidate catching point maximizing the above defined convex linear combination of quality indices is chosen as the current catching point $\boldsymbol{p}_\times$.

Therefore, the robot path is replanned to lead the gripper from the current state of the robot to the point $\boldsymbol{p}_\times$ at the time $t_\times$ with the same velocity of the ball $\boldsymbol{v}_\times$ (details in the next Section). Notice that the planned trajectory could be not achievable with respect to the robot capabilities. If the velocity $\boldsymbol{v}_\times$ and the maximum required acceleration for the end-effector, i.e. the camera, are greater than a fixed limit chosen in a conservative way accordingly with robot capabilities, then the catching time is suitably scaled. Concerning the acceleration, denoting with a_{max} the norm of the maximum acceleration that the end-effector can reach and with $\ddot{\overline{\boldsymbol{p}}}_{c,d}$ the maximum planned acceleration, if $\|\ddot{\overline{\boldsymbol{p}}}_{c,d}\| > a_{max}$, then the catching time is scaled as $\bar{t}_\times = t_\times \sqrt{\|\ddot{\overline{\boldsymbol{p}}}_{c,d}\|/a_{max}}$. On the other hand, in case of $\|\boldsymbol{v}_\times\| > v_{max}$, where v_{max} is the chosen limit for the linear Cartesian velocity of the robot, a reduced velocity is considered for the interception time $\bar{\boldsymbol{v}}_\times = v_{max} \frac{\boldsymbol{v}_\times}{\|\boldsymbol{v}_\times\|}$, while the catching time is scaled similarly to the acceleration case.

The *catching path* is then generated when the estimation process stops. The camera's orientation is controlled to have a direction of the optical axis, i.e., of the gripper, equal to the tangent to the estimated trajectory of the ball at $\boldsymbol{p}_\times$. Once that the catching point is reached at $t_\times$ with the same (or reduced) velocity of the ball, the gripper is closed and moved along the predicted path of the ball, while its velocity is decreased to zero in a fixed time/space. In this way it is possible to dissipate the impact energy in a finite time interval.

VI. On-line path replanning

When a new h-th estimation is available at time t_h, the current robot path has to be modified in a smooth way to reach the new estimated catching point. This means that such new path must guarantee the continuity with the current motion state (position, velocity and acceleration). A fifth-order polynomial has been employed in [1], [2]. In order to lower the polynomial order, reducing as much as possible the oscillations of the planned path, a classic cubic spline [32] employing only third-order polynomials is instead used in this paper.

Hence, the following definition can be given for the desired trajectory of the robot

$$\boldsymbol{p}_{c,d}(t) = \Pi_h(t) = \begin{cases} \Pi_{h_1}(t) & t_h \le t \le t_{v_1} \\ \Pi_{h_2}(t) & t_{v_1} \le t \le t_{v_2} \\ \Pi_{h_3}(t) & t_{v_2} \le t \le t_f \end{cases} \tag{14}$$

where $\boldsymbol{p}_{c,d}(t)$ is a (3×1) vector denoting the desired trajectory for Σ_c with respect to Σ_b, $t_f = t_\times$ (or $t_f = \bar{t}_\times$, in case of time scaling), t_{v_1} and t_{v_2} are two time instants referred to two generic virtual points in $\Pi_h(t)$.

The following 12 equations in 12 unknowns–the 4 coefficients for each of the 3 cubic polynomials in (14)–can be

[2]Notice that the inverse kinematics can be evaluated iteratively between two consecutive candidate catching points, by using the joint configuration of the previous catching point to initialize the algorithm for the new Cartesian configuration. In this way, being consecutive candidates close to each other, few iterations are required to converge to the solution of the joint configuration corresponding to the current candidate.

imposed

$$\Pi_{h_1}(t_h) = \Pi_{h-1}(t_h), \quad \dot{\Pi}_{h_1}(t_h) = \dot{\Pi}_{h-1}(t_h), \tag{15a}$$
$$\ddot{\Pi}_{h_1}(t_h) = \ddot{\Pi}_{h-1}(t_h), \tag{15b}$$
$$\Pi_{h_3}(t_f) = \boldsymbol{p}_{\times}, \quad \dot{\Pi}_{h_3}(t_f) = \boldsymbol{v}_f, \tag{15c}$$
$$\ddot{\Pi}_{h_3}(t_f) = \boldsymbol{0}, \tag{15d}$$
$$\Pi_{h_1}(t_{v_1}) = \Pi_{h_2}(t_{v_1}), \quad \dot{\Pi}_{h_1}(t_{v_1}) = \dot{\Pi}_{h_2}(t_{v_1}), \tag{15e}$$
$$\ddot{\Pi}_{h_1}(t_{v_1}) = \ddot{\Pi}_{h_2}(t_{v_1}), \quad \Pi_{h_2}(t_{v_2}) = \Pi_{h_3}(t_{v_2}), \tag{15f}$$
$$\dot{\Pi}_{h_2}(t_{v_2}) = \dot{\Pi}_{h_3}(t_{v_2}), \quad \ddot{\Pi}_{h_2}(t_{v_2}) = \ddot{\Pi}_{h_3}(t_{v_2}), \tag{15g}$$

where $\boldsymbol{0}$ is the (3×1) null vector, while $\boldsymbol{v}_f = \boldsymbol{v}_{\times}$ (or $\bar{\boldsymbol{v}}_{\times}$ if $\|\boldsymbol{v}_{\times}\| > v_{max}$). Notice that the effective location of the virtual points is irrelevant, since their position constraints are exploited for continuity only, and depends on the choice of t_{v1} and t_{v2} (a possible solution is $t_{v1} = (t_f - t_h)/3$ and $t_{v2} = 2(t_f - t_h)/3$). Finally, notice that (15a)-(15b) are the equations relative to the initial conditions, (15c)-(15d) are the equations relative to the final conditions, and (15e)-(15g) are the equations relative to the two virtual points. Moreover, notice that for $h = 1$ the continuity is intended to be with respect to the planned baseline (see Section V-A).

VII. Partitioned Visual Servoing Control Law

The partitioned visual servoing control law described in this paper belongs to the category named *Resolved-Velocity Image-Based Visual Servoing* [32], for which it is assumed that the manipulator dynamics is taken into account directly by the low-level robot controller. Such a control law has been introduced in [34]–[36] and already employed in [1], [3]. Only the key features will be thus here reported: the reader may refer to the original papers for more details.

The (6×1) absolute velocity vector 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 (6×1) absolute velocity vector of the thrown ball $\boldsymbol{v}_o^c = \begin{bmatrix} \dot{\boldsymbol{p}}_o^{c\mathrm{T}} & \boldsymbol{\omega}_o^{c\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$, both expressed with respect to Σ_c, and the (2×1) velocity vector of the image feature $\dot{\boldsymbol{s}}$ in the image plane, are related by the following expression [32]

$$\dot{\boldsymbol{s}} = \boldsymbol{L}_s \boldsymbol{v}_c^c + \boldsymbol{L}_s \boldsymbol{\Gamma}(-\boldsymbol{p}_o^c) \boldsymbol{v}_o^c, \tag{16}$$

in which $\boldsymbol{L}_s = \begin{bmatrix} \boldsymbol{L}_{sp}(\boldsymbol{s}, \boldsymbol{p}_o^c) & \boldsymbol{L}_{so}(\boldsymbol{s}) \end{bmatrix}$ is the (2×6) interaction matrix of a point image feature defined as follows

$$\boldsymbol{L}_s = \begin{bmatrix} -\frac{1}{z^c} & 0 & \frac{X}{z^c} & XY & -1 - X^2 & Y \\ 0 & -\frac{1}{z^c} & \frac{Y}{z^c} & 1 + Y^2 & -XY & -X \end{bmatrix}, \tag{17}$$

where $\boldsymbol{L}_{sp}$ and $\boldsymbol{L}_{so}$ are the (2×3) sub-matrices corresponding to the first and last three columns of (17), respectively, and $\boldsymbol{\Gamma}(\cdot)$ is the (6×6) matrix

$$\boldsymbol{\Gamma}(\cdot) = \begin{bmatrix} -\boldsymbol{I}_3 & \boldsymbol{S}(\cdot) \\ \boldsymbol{0} & -\boldsymbol{I}_3 \end{bmatrix}, \tag{18}$$

where $\boldsymbol{I}_n$ denotes the $(n \times n)$ identity matrix and $\boldsymbol{S}(\cdot)$ the skew-symmetric matrix. Equation (16) can be rewritten as

$$\dot{\boldsymbol{s}} = \boldsymbol{L}_{sp} \left(\dot{\boldsymbol{p}}_c^c - \dot{\boldsymbol{p}}_o^c + \boldsymbol{S}(-\boldsymbol{p}_o^c) \boldsymbol{\omega}_o^c \right) + \boldsymbol{L}_{so} \left(\boldsymbol{\omega}_c^c - \boldsymbol{\omega}_o^c \right). \tag{19}$$

By exploiting the employed partitioned visual servoing controller, the translational components $\dot{\boldsymbol{p}}_c^c$ of the robot motion are devoted to move the robot to intercept the path of the ball with the gripper mounted on the robot end-effector [34].

Given the desired trajectory $\boldsymbol{p}_{c,d}$, $\dot{\boldsymbol{p}}_{c,d}$ and $\ddot{\boldsymbol{p}}_{c,d}$ for the camera frame, on the one hand, the translational components of the velocity input for the camera frame Σ_c can be generated as follows

$$\dot{\boldsymbol{p}}_c^c = \boldsymbol{R}_c^{\mathrm{T}} \left(\dot{\boldsymbol{p}}_{c,d} + \boldsymbol{K}_p \boldsymbol{e}_p \right), \tag{20}$$

with $\boldsymbol{R}_c$ the rotational matrix of the camera frame Σ_c with respect to the base frame Σ_b, $\boldsymbol{K}_p > 0$ a diagonal constant (3×3) gain matrix, and $\boldsymbol{e}_p$ the (3×1) error vector between the desired trajectory $\boldsymbol{p}_{c,d}$ and the one provided by the robot direct kinematics at time t. On the other hand, the rotational components of the velocity input for the camera frame Σ_c can be generated in the image space as follows:

$$\boldsymbol{\omega}_c^c = \boldsymbol{L}_{so}^{\dagger} \left[\boldsymbol{K}_{so,e_{b2}}(\boldsymbol{e}_s) \boldsymbol{\tau}_{e_{b1}}(\boldsymbol{e}_s) - \hat{\boldsymbol{L}}_{sp} \left(\dot{\boldsymbol{p}}_c^c - \hat{\dot{\boldsymbol{p}}}_o^c + \boldsymbol{S}(-\hat{\boldsymbol{p}}_o^c) \hat{\boldsymbol{\omega}}_o^c \right) \right] + \hat{\boldsymbol{\omega}}_o^c, \tag{21}$$

with $\dagger$ denoting the pseudo-inverse of a matrix, $\hat{\boldsymbol{p}}_o^c$ the estimate of the unknown position of the ball in Σ_c, and $\dot{\boldsymbol{p}}_c^c$ evaluated in (20). Notice that the matrix $\boldsymbol{L}_{so} \boldsymbol{L}_{so}^{\mathrm{T}}$ is never singular since its determinant is equal to $1/(X^2 + Y^2 + 1)^3$. The terms $\hat{\dot{\boldsymbol{p}}}_o^c$ and $\hat{\boldsymbol{\omega}}_o^c$ are the estimates of the unknown absolute translational and angular velocities of the ball with respect to Σ_c. Notice that $\hat{\boldsymbol{L}}_{sp}$ in (21) is an estimated term since it depends on $\hat{\boldsymbol{p}}_o^c$. Moreover, the error term $\boldsymbol{e}_s = -\boldsymbol{s}$ is the image error vector becoming null when the camera is pointed towards the centroid of the ball, while the term $\boldsymbol{\tau}_{e_b}(\boldsymbol{e}_s)$ is a threshold function defined as [3]

$$\boldsymbol{\tau}_{e_{b1}}(\boldsymbol{e}_s) = \begin{cases} \boldsymbol{0} & \text{if } \|\boldsymbol{e}_s\| \leq e_{b1} \\ \left(1 - \dfrac{e_{b1}}{\|\boldsymbol{e}_s\|} \right) \boldsymbol{e}_s & \text{if } \|\boldsymbol{e}_s\| > e_{b1}, \end{cases} \tag{22}$$

with $\boldsymbol{K}_{so,e_{b2}}(\boldsymbol{e}_s)$ a (2×2) gain matrix defined as

$$\boldsymbol{K}_{so,e_{b2}}(\boldsymbol{e}_s) = \begin{cases} k_o \boldsymbol{I}_2 & \text{if } \|\boldsymbol{e}_s\| \leq e_{b2} \\ k_o e^{\beta_o \left(\frac{\|\boldsymbol{e}_s\|}{e_{b2}} - 1 \right)} \boldsymbol{I}_2 & \text{if } \|\boldsymbol{e}_s\| > e_{b2}, \end{cases} \tag{23}$$

where $k_o > 0$ is a gain factor, $e_{b2} > e_{b1} > 0$ are proper thresholds, and $\beta_o > 0$ is a restraint factor tuning the increasing rate of $\boldsymbol{K}_{so}$. The details about how to estimate $\hat{\boldsymbol{p}}_o^c$ (and then also $\hat{\boldsymbol{L}}_{sp}$), $\hat{\dot{\boldsymbol{p}}}_o^c$ and $\hat{\boldsymbol{\omega}}_o^c$ are given in the next subsection.

Finally, the input to the robot controller, i.e., the joint velocity vector, can be computed as [32]

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^{\dagger}(\boldsymbol{q}) \boldsymbol{T}_c \boldsymbol{v}_c^c + \boldsymbol{N}_J \boldsymbol{K}_r \dot{\boldsymbol{q}}_r, \tag{24}$$

with $\boldsymbol{q}$ the vector of joint positions, $\boldsymbol{J}(\boldsymbol{q})$ the Jacobian matrix of the robot that is pseudo-inverted as denoted in [32], $\boldsymbol{T}_c$ the (6×6) matrix relating $\boldsymbol{v}_c^c$ to the velocity of the robot end-effector with respect to the base frame, $\boldsymbol{N}_J$ the projector matrix into the null space of the robot Jacobian, $\boldsymbol{K}_r$ a diagonal positive definite gain matrix, and $\dot{\boldsymbol{q}}_r$ a set of joint velocities employed in a possible redundancy management to optimize

some other sub-tasks [40], e.g. joint limits and singularities avoidance, increasing the manipulability.

A. Estimation of the ball-camera relative motion

The necessary quantities to be estimated in (21) are the linear position $\hat{\boldsymbol{p}}_o^c$, linear velocity $\dot{\hat{\boldsymbol{p}}}_o^c$, and angular velocity $\hat{\boldsymbol{\omega}}_o^c$ of the center of the ball with respect to Σ_c. Starting from the current estimate of the position $\boldsymbol{p}_0$ and velocity $\boldsymbol{v}_0$ of the ball, the ballistic model (12) is numerically integrated in the time interval $[0, t_i]$. In this way, $\hat{\boldsymbol{p}}_o^c$ can be obtained at a certain time t. With the same numerical integration, it is also possible to obtain the estimate of the linear velocity $\dot{\hat{\boldsymbol{p}}}_o^c$. It is worth noting that the first estimate of $\boldsymbol{p}_0$ and $\boldsymbol{v}_0$ is obtained after n_l measurements (see Section V-A). Before n_l measurements are collected, in order to compute the above mentioned quantities, an initial value of $\boldsymbol{p}_0$ and $\boldsymbol{v}_0$ should be anyhow provided. Hence, a statistical calibration has been preliminary realized to retrieve a rough initial estimation of $\boldsymbol{p}_0$ and $\boldsymbol{v}_0$.

Finally, the angular velocity can be retrieved as

$$\hat{\boldsymbol{\omega}}_o^c = (\hat{\boldsymbol{p}}_o^c \times \dot{\hat{\boldsymbol{p}}}_o^c)\|\hat{\boldsymbol{p}}_o^c\|^{-2} = \boldsymbol{S}(\hat{\boldsymbol{p}}_o^c)\dot{\hat{\boldsymbol{p}}}_o^c\|\hat{\boldsymbol{p}}_o^c\|^{-2}. \tag{25}$$

B. Stability proofs

Before starting with the proofs, it is necessary to give some bounds about the quantities given in Section VII-A. In particular, taking into account what written in Section VII-A and (12), the following expressions hold

$$\|\dot{\hat{\boldsymbol{p}}}_o^c\| \leq B_1 < \infty, \tag{26}$$
$$\|\hat{\boldsymbol{p}}_o^c\| \leq B_2 < \infty, \tag{27}$$

where B_1 and B_2 are two positive limited bounds since $\dot{\hat{\boldsymbol{p}}}_o^c$ and $\hat{\boldsymbol{p}}_o^c$ are numerically integrated as described in Section VII-A, and this means that they are a finite sum of finite elements. Moreover, they can be also saturated by programming code. For the angular part, taking into account (25) and the property that the Eucledian norm of a skew-symmetric matrix of a vector is equal to the norm of the vector itself, the following bound can be considered

$$\|\hat{\boldsymbol{\omega}}_o^c\| \leq B_3 < \infty, \tag{28}$$

with $B_3 = B_1/B_2$ a positive limited bound since B_1 and B_2 are positive and limited. Finally, the following physical bounds can be considered

$$\|\boldsymbol{p}_o^c\| \leq B_4 < \infty \tag{29a}$$
$$\|\dot{\boldsymbol{p}}_o^c\| \leq B_5 < \infty \tag{29b}$$
$$\|\boldsymbol{\omega}_o^c\| \leq B_6 < \infty \tag{29c}$$

For what concerns the stability proofs of the system, the following theorems hold.

Theorem 1. *Provided that $\boldsymbol{K}_p$ is a positive definite matrix, the control law* (20) *ensures an asymptotic convergence to zero of the position error $\boldsymbol{e}_p$.*

Proof. The time derivative of the position error can be computed as

$$\dot{\boldsymbol{e}_p} = \frac{\mathrm{d}}{\mathrm{dt}}(\boldsymbol{p}_{c,d} - \boldsymbol{p}_c(t)) = \dot{\boldsymbol{p}}_{c,d} - \dot{\boldsymbol{p}}_c, \tag{30}$$

where $\boldsymbol{p}_c(t)$ is the (3×1) vector denoting the current position of the camera with respect to Σ_b at time t, computed by solving the direct kinematics of the robot, while $\dot{\boldsymbol{p}}_c$ is the related translational velocity of the camera with respect to Σ_b. Pre-multiplying by $\boldsymbol{R}_c$ both sides of (20) and folding the result into the previous equation yields

$$\dot{\boldsymbol{e}}_p + \boldsymbol{K}_p \boldsymbol{e}_p = \boldsymbol{0}. \tag{31}$$

Since $\boldsymbol{K}_P$ is a positive definite matrix, usually a diagonal matrix, the previous system is asymptotically stable and the error $\boldsymbol{e}_p$ tends to zero along the trajectory with a convergence rate depending on the eigenvalues of $\boldsymbol{K}_p$. □

Theorem 2. *The system* (19)*, equivalent to* (16)*, is asymptotically stable under the control laws* (20)-(21)*, in the presence of a perfect estimate of the unknown terms. Otherwise, only stability can be ensured, where the bounds can be determined by tuning the gain k_o.*

Proof. The following analysis is performed by using the direct Lyapunov theorem. Consider the following candidate Lyapunov function $V(\boldsymbol{e}_s) = \boldsymbol{e}_s^{\mathrm{T}} \boldsymbol{K}_s \boldsymbol{e}_s$, where $\boldsymbol{K}_s$ is a (2×2) positive definite diagonal matrix. By noticing that $\dot{\boldsymbol{e}}_s = -\dot{\boldsymbol{s}}$, computing the time derivative of $V(\boldsymbol{e}_s)$ and taking into account (19), (20) and (21) yield $\dot{V} = -\alpha_1 - \alpha_2 - \alpha_3$, where

$$\alpha_1 = \boldsymbol{e}_s^{\mathrm{T}} \boldsymbol{K}_s \left(\boldsymbol{L}_{sp} - \hat{\boldsymbol{L}}_{sp} \right) \dot{\boldsymbol{p}}_c^c \tag{32a}$$
$$\alpha_2 = \boldsymbol{e}_s^{\mathrm{T}} \boldsymbol{K}_s \boldsymbol{K}_{so,e_{b2}}(\boldsymbol{e}_s) \boldsymbol{\tau}_{eb_1}(\boldsymbol{e}_s) \tag{32b}$$
$$\alpha_3 = \boldsymbol{e}_s^{\mathrm{T}} \boldsymbol{K}_s \left(\boldsymbol{L}_s \boldsymbol{\Gamma}(-\boldsymbol{p}_o^c) \boldsymbol{v}_o^c - \hat{\boldsymbol{L}}_s \boldsymbol{\Gamma}(-\hat{\boldsymbol{p}}_o^c) \hat{\boldsymbol{v}}_o^c \right). \tag{32c}$$

If each term in (32) is strictly positive, then $\dot{V} < 0$. However, no term in (32) is a quadratic form, hence only qualitative considerations can be achieved.

If $\hat{\boldsymbol{L}}_{sp} = \boldsymbol{L}_{sp}$, the term α_1 in (32a) vanishes, but there is no guarantee that such condition can happen. Nevertheless, the α_1 term is bounded since the condition for updating $\hat{\boldsymbol{L}}_{sp}$ through the estimate of $\hat{\boldsymbol{p}}_o^c$ seems to be the optimal one during the experiments [41].

By considering (22)-(23), the term α_2 in (32b) can be bounded as follows

$$0 \leq \alpha_2 \leq \boldsymbol{e}_s^{\mathrm{T}} \boldsymbol{K}_s \begin{pmatrix} \beta_o \left(\frac{\|\boldsymbol{e}_s\|}{e_{b2}} - 1 \right) \\ k_o e \end{pmatrix} \boldsymbol{I}_2 \boldsymbol{e}_s. \tag{33}$$

By choosing $\boldsymbol{K}_s = k_o e^{\beta_o \left(\frac{\|\boldsymbol{e}_s\|}{e_{b2}} - 1 \right)} \boldsymbol{I}_2$, the last term in (33) is positive definite. Hence, α_2 is always positive and limited.

The α_3 term in (32c) vanishes in case of perfect estimate. Otherwise, nothing can be said about the sign of α_3. Denoting with λ the maximum eigenvalue of $\boldsymbol{K}_s$ and with $\sigma_M(\cdot)$ the maximum singular value of a matrix, recalling that $\boldsymbol{L}_s = \begin{bmatrix} \boldsymbol{L}_{sp} & \boldsymbol{L}_{so} \end{bmatrix}$, $\hat{\boldsymbol{L}}_s = \begin{bmatrix} \hat{\boldsymbol{L}}_{sp} & \boldsymbol{L}_{so} \end{bmatrix}$, $\boldsymbol{v}_o^c = \begin{bmatrix} \dot{\boldsymbol{p}}_o^c & \boldsymbol{\omega}_o^c \end{bmatrix}$ and $\hat{\boldsymbol{v}}_o^c = \begin{bmatrix} \dot{\hat{\boldsymbol{p}}}_o^c & \hat{\boldsymbol{\omega}}_o^c \end{bmatrix}$, taking into account (29), the following

expression for α_3 holds $\alpha_3 = \alpha_{3,1} + \dots \alpha_{3,6}$, where each term can be bounded as follows

$$\alpha_{3,1} = \boldsymbol{e}_s^{\mathrm{T}} \boldsymbol{K}_s \boldsymbol{L}_{sp} \dot{\boldsymbol{p}}_o^c \leq \lambda \sigma_M(\boldsymbol{L}_{sp}) B_5 \|\boldsymbol{e}_s\| \quad (34a)$$

$$\alpha_{3,2} = \boldsymbol{e}_s^{\mathrm{T}} \boldsymbol{K}_s \boldsymbol{L}_{sp} \boldsymbol{S}(\boldsymbol{\omega}_o^c) \boldsymbol{p}_o^c \leq \lambda \sigma_M(\boldsymbol{L}_{sp}) B_6 B_4 \|\boldsymbol{e}_s\| \quad (34b)$$

$$\alpha_{3,3} = -\boldsymbol{e}_s^{\mathrm{T}} \boldsymbol{K}_s \hat{\boldsymbol{L}_{sp}} \dot{\hat{\boldsymbol{p}}}_o^c \leq \lambda \sigma_M(\hat{\boldsymbol{L}}_{sp}) B_1 \|\boldsymbol{e}_s\| \quad (34c)$$

$$\alpha_{3,4} = \boldsymbol{e}_s \boldsymbol{K}_s \hat{\boldsymbol{L}}_{sp} \boldsymbol{S}(-\hat{\boldsymbol{p}}_o^c) \boldsymbol{\omega}_o^c \leq \lambda \sigma_M(\hat{\boldsymbol{L}}_{sp}) B_2 B_6 \|\boldsymbol{e}_s\| \quad (34d)$$

$$\alpha_{3,5} = -\boldsymbol{e}_s^{\mathrm{T}} \boldsymbol{K}_s \boldsymbol{L}_{so} \hat{\boldsymbol{\omega}}_o^c \leq \lambda \sigma_M(\boldsymbol{L}_{so}) B_3 \|\boldsymbol{e}_s\| \quad (34e)$$

$$\alpha_{3,6} = \boldsymbol{e}_s^{\mathrm{T}} \boldsymbol{K}_s \boldsymbol{L}_{so} \boldsymbol{\omega}_o^c \leq \lambda \sigma_M(\boldsymbol{L}_{so}) B_6 \|\boldsymbol{e}_s\|. \quad (34f)$$

Hence, supposing α_1 is positive definite, taking into account (33), the particular choice for $\boldsymbol{K}_s$ and the expressions in (34), the following bounds for $\dot{V}$ hold

$$\dot{V} \leq -\boldsymbol{e}_s^{\mathrm{T}} \boldsymbol{K}_s \boldsymbol{K}_s \boldsymbol{e}_s + \lambda B \|\boldsymbol{e}_s\| \leq -\lambda \left(\lambda \|\boldsymbol{e}_s\| - B\right) \|\boldsymbol{e}_s\|, \quad (35)$$

where $B = \sigma_M(\boldsymbol{L}_{sp}) \left(B_5 + B_6 B_4\right) + \sigma_M(\boldsymbol{L}_{so}) \left(B_3 + B_6\right) + \sigma_M(\hat{\boldsymbol{L}}_{sp}) \left(B_1 + B_2 B_6\right)$. Hence, from (35), it is possible to conclude that when $\|\boldsymbol{e}_s\| > B/\lambda$ then $\dot{V} < 0$ and then the system is asymptotically stable. When $\|\boldsymbol{e}_s\| \leq B/\lambda$ nothing can be said about the sign. Hence, the term B/λ is the bound for the error $\boldsymbol{e}_s$. To reduce the bound, $\lambda \to \infty$, the gain k_o should be increased as much as possible with respect to the employed controller sample time.

In conclusion, in case of a perfect estimate, the terms α_1 and α_3 vanish, while α_2 is positive and limited. Then, the chosen control laws lead to an asymptotically stable system. In case of an imperfect compensation, instead, the error in the image plane is anyway bounded. □

For a ball catching task, this stability condition can be considered sufficient, because the visual control goal is mainly in keeping the ball in the field of view of the camera.

VIII. Experiments

The employed experimental set-up is depicted in Fig. 10. An USB iDS UEYE UI-1220SE-C camera is mounted in eye-in-hand configuration directly in the center of the base of the gripper. This last is made up of two brushed DC motors, with a metal gearbox and an integrated quadrature encoder. Through a rack-and-pinion mechanism the motion of these motors allows the closure of the gripper fingers. The gripper is in turn mounted on a COMAU Smart-Six robot manipulator standing on a sliding track. The COMAU C4G control unit is in charge of the compensation of the robot dynamics.

An external PC with UBUNTU OS and a patched RTAI real-time kernel generates the references for the robot each 2ms. A second PC with Windows OS is in charge of the visual elaboration process and communicates with the first one through an UDP protocol over a GigaBit dedicated LAN. A high-priority multi-thread programming has been employed to improve the stability of the elaboration time and synchronize the visual measurements with the robot motion.

In order to provide a ground truth for the proposed estimation algorithm, an OptiTrack motion-capture system composed of ten S250e cameras has been employed to track the ball during its motion.

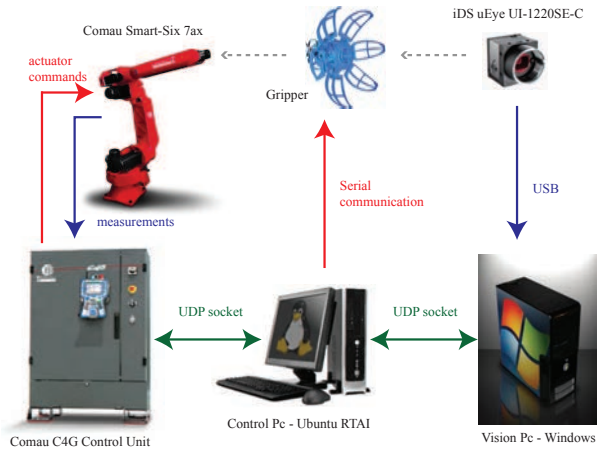


Fig. 10. Architecture of the ball catching system.

A. Technical details

The portion of the camera image employed for the ball detection is limited to (368×512) pixels. On the basis of the available ball, the range of radius of interest for the first detection is $[10, 14]$ pixels, which corresponds to a distance of the ball from the camera in the range of $[5, 6]$ m. For other balls, these parameters have to be retuned. The RoI window is dynamically placed and sized with respect to the current radius of the detected ball. At the first detection, its dimension is set to (38×38) pixels. With this configuration parameter the acquisition frame rate is speeded up to 140 fps.

The available ball has a diameter of 8.5 cm and a weight of about 32 g. Six soft reflecting markers are attached to the ball in order to make the OptiTrack system able to observe it. These markers do not affect the visual detection of the ball in the scene since they do not alter the ball shape. The coefficients of the air drag factor have been tuned to $c_w = 0.45$ and $\rho_a = 1.293$ kg/m^3.

On the basis of the available robot, the gain matrix in (20) has been set by experimental tuning to $\boldsymbol{K}_p = 500\boldsymbol{I}_3$, while the gains in (21) have been tuned to $k_o = 200$, $e_{b1} = 10$ and $e_{b2} = 100$.

By using the OptiTrack system, it has been possible to measure the rebound of the chosen ball on the ground. Hence, a calibration procedure has been adopted to tune the parameters $\boldsymbol{K}_b$ and $\boldsymbol{b}_b$ in (13). In details, the motion-capture system has provided the 3D position of the six markers at a frequency of 250 Hz. By knowing both the geometrical features of the ball and the position of these markers it has been possible to reconstruct the real ballistic trajectory of the ball. A set of 20 trajectories containing rebounded points have been acquired. The ball velocities before and after the rebound have been computed with a filtered derivative of such trajectories. Thereafter, the coefficients $\boldsymbol{K}_b$ and $\boldsymbol{b}_b$ have been retrieved through a least squares solution. Namely, the obtained values are $\boldsymbol{K}_b = \mathrm{diag}\begin{bmatrix}0.2 & -0.785 & 0.55\end{bmatrix}$, while $\boldsymbol{b}_b$ has been approximated to a null vector since its values are very small with respect to $\boldsymbol{K}_b$ for the current set-up.

The intrinsic redundancy of the chosen robotic platform has been exploited in (24) to avoid joint limits, kinematic

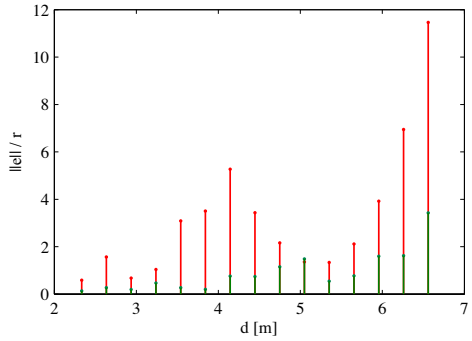


Fig. 11. Ball distance estimation error normalized to the radius in green (red) with (without) the sub-pixel refinement.

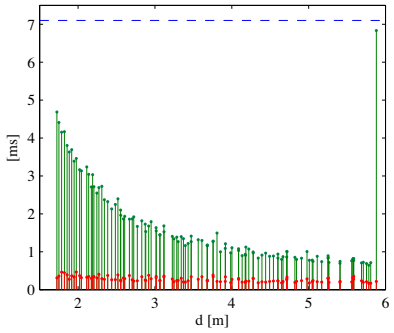


Fig. 12. Elaboration time of the ball detection and tracking process with respect to the ball distance from the camera (hence, read the picture from right to left). In green the whole elaboration time, in red the time required for the sub-pixel refinement process. The dashed line indicates the time limit corresponding to the camera rate of 140fps.

singularities and to reduce the movements of the sliding track since its dynamics is considerably slower than those of the other joints.

On the basis of the available set-up, the initial baseline has a planned length of 50 cm that should be performed by the camera in 500 ms. The first estimate of the trajectory starts when about $n_l = 45$ samples have been collected, i.e., after about 320 ms. Therefore, typically, the first catching trajectory starts before the end of the baseline path.

Latency periods and delays between the robot control PC, the C4G control unit and the visual elaboration PC have been estimated so as to synchronize at the best the direct kinematic measurements of the robot with the visual data.

B. Ball detection and tracking results

In order to evaluate the performance of the proposed sub-pixel refinement algorithm (see Section IV-D), a (fixed) ball with a diameter of 9 cm has been observed at 15 different distances from the camera, equally distributed by a step of 30 cm. Since the camera calibration parameters are known, from the observed radius of the ball in the image it is possible to estimate the distance of the ball from the camera. The error between the real distance and the estimated one, normalized with respect to the ball radius, is depicted in Fig. 11. The errors computed with the measurements taken through the sub-pixel refinement are represented in green, while the not refined measurements are represented in red. In all cases, the refined measure outperform significantly the integer estimations of the ball position and radius.

With respect to a generic throw, the elaboration time of the proposed detection and tracking process with respect to the distance of the ball from the camera (which is related to the ball radius) is shown in Fig. 12. The whole elaboration time is represented in green, while the time required for the sub-pixel refinement process is represented in red. The dashed line indicates the time limit corresponding to the camera rate of 140fps. Since Fig. 12 depicts the elaboration time with respect to the distance of the ball from the camera, the pictures goes from right to left. Hence, it is worth noticing how the first elaboration time is bigger than others because the detection process is applied to the whole image, while from the second step the elaboration time is significantly reduced thanks to the windowing and tracking strategy. At the end (extreme left of Fig. 12), once the ball is near to the camera, the elaboration time increases since the ball radius increases in the image and more pixels have to be elaborated.

C. Ball Catching Results

Several experiments have been performed with a number of pitchers and varying light conditions. The percentage of caught ball evaluated over a set of 300 shots (100 shots for each of the three trials: rolling, bouncing, flying ball) is about 95%. Namely, the caught balls in the rolling case have been 98, as well as in the flying ball case, while 89 balls have been grabbed in the bouncing case. Hence, the main problems originate from the employed rebound model and the related parameters. Concerning the statistical reliability test, the mean percentage of discarded outliers in each throw is about 11.5%, while the accuracy of the final catching point, on the basis of the OptiTrack measurements, has been improved about 5%. This corresponds to an improvement of the interception point of about 2–3cm in the available set-up.

An example of a possible ball trajectory for a given throw with the overlay of the motion of the robot is depicted in Fig. 13. Moreover, a multimedia attachment (available also online[3]) shows the performance of the proposed algorithm.

The OptiTrack system has been employed to give a ground truth about the estimate of the ballistic trajectory. Some examples are shown in Fig. 14: the case of a bouncing ball is depicted in Fig. 14(a), the case of a normal throw in Fig. 14(b), while Fig. 14(c) presents the case of a rolling ball. In all these pictures, the green tube is the space occupied by the ball during its motion towards the robot as measured by the OptiTrack system. The blue line is the final estimated trajectory of the centroid of the ball. It is possible to observe that the blue line is always inside the green tube: this gives a geometric quality measure about the performance of the proposed estimation process. The red line shows the path followed by the camera/gripper mounted on the robot. It is then possible to recognize both the initial baseline and the catching path in which the gripper follows the predicted ball

[3]http://www.youtube.com/watch?v=gT8Zn6L5PEk

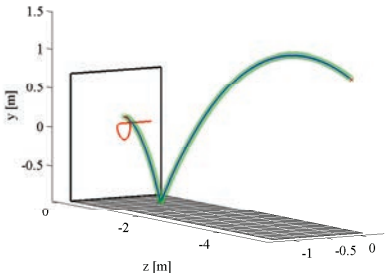
(a) Flying case (b) Bouncing case

Fig. 13. Overlay of the ball trajectory and robot motion in flying and bouncing cases.

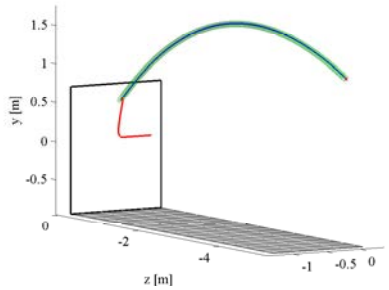
trajectory during the catch. In Fig. 14(c) the blue circle denotes the space occupied by the opened gripper that is placed at a safety distance from the ground.

Further to the geometrical path, another quality index to measure the performance of the estimation process is a comparison along the time between the ground truth and the final predicted trajectory. With reference to the throw depicted in Fig. 14(a), the time histories of both the ground truth (i.e., the space traced by the ball) and the predicted trajectories are shown in Fig. 15. Again, the time histories of the predicted trajectory fit inside the ground truth.

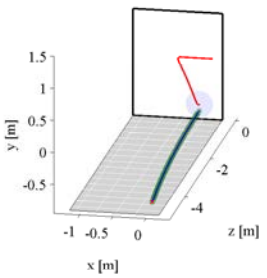
The predicted interception points $\boldsymbol{p}_{\times}$ for the throw of Fig. 14(a) at each estimation time t_i, projected in both the $(x-y)$ and $(z-y)$ planes of Σ_b, are represented with a cross point in Fig. 16. The color bar identifies the ordered sequence of such predicted interception points with reference to the number of employed measures, while the biggest brown cross represents the final position $\boldsymbol{p}_{\times}$ in which the estimate has been considered as stable. The dashed lines represent the planned path for the hand (see Section VI), 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 gripper, which starts with the baseline (green piece of the path). It is worth noticing that the first estimated point, the green one, is given by the linear estimation process. The big orange circle is the full representation of the ball in the final position measured by the OptiTrack system and projected in the above mentioned planes of Σ_b. The big blue circle in the background is instead the space occupied by the gripper base in the final estimated position and projected in the the above mentioned planes of Σ_b. The information provided by Fig. 16 is twofold. First, it is possible possible to recognize the tolerance between the real position of the ball and the space occupied by the gripper at the interception point; then, another way to measure the quality of the estimate is the evaluation of how far the final estimated $\boldsymbol{p}_{\times}$ is from the center of the measured position of the ball.



(a) Bouncing throw



(b) Flying throw



(c) Rolling throw

Fig. 14. 3D plots of three different throws. The blue line is the final estimated trajectory of the ball centroid. The green cylinder is the space occupied by the ball during the flight that has been measured by the OptiTrack system. The red path is the motion of the camera/gripper. The light blue circle is the space occupied by the gripper with all the fingers open.

IX. Conclusion

A new technique to catch a thrown ball with a robotic system endowed of only a single camera mounted in eye-in-hand configuration has been provided. Namely, the proposed novelties are: a sub-pixel refinement for the circle detector visual algorithm; a statistical reliability test to reduce image noise and possible outliers; deal with flying, bouncing and rolling balls in the same framework without changing neither the estimator nor the control law. The effectiveness of the proposed approach has been demonstrated in both theory and experimental results on a common industrial robotic set-up. An experimental comparison of the achieved results with the measurements given by an OptiTrack motion-capture system has been provided.

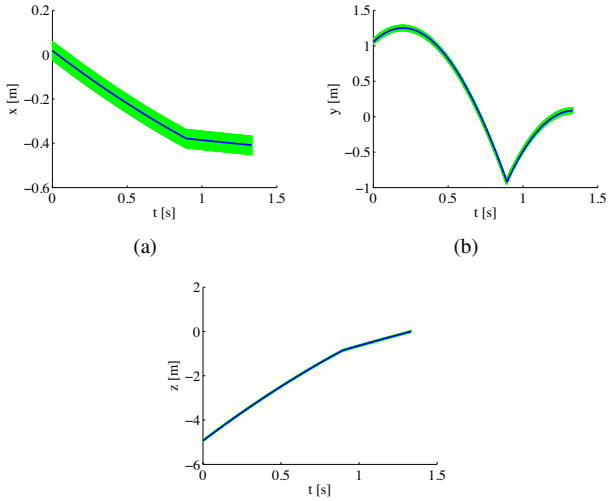
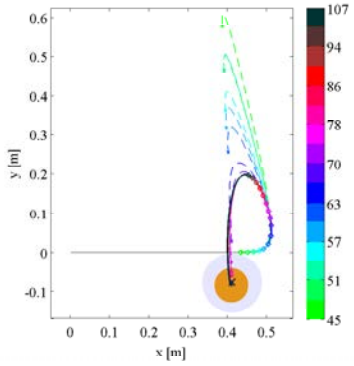
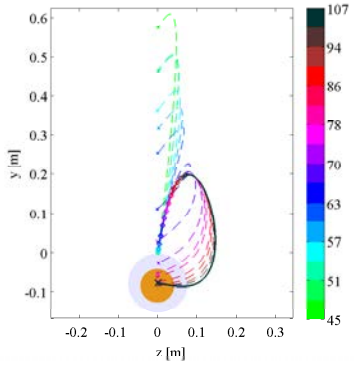


Fig. 15. Time histories of the ground truth (in green) and the final predicted trajectory (in blue) of the throw depicted in Figure 14(a).



(a) $x - y$ plane.



(b) $z - y$ plane.

Fig. 16. Sequence of the interception points (cross points) projected onto the $(x - y)$-plane and $(z - y)$-plane. The dashed lines represent the planned path, starting from the current hand position (circle points) and ending at the current estimated interception points. The continuous lines represent the real path followed by the gripper, starting with the initial baseline (green) and leading to the final catching point (biggest cross). The big orange circle represents the final position of the ball measured by the OptiTrack system. The big blue circle in the background is the estimated final position of the gripper catching base. The color bar on the right identifies the refinements of the interception points, while the related labels represent the number of visual measurements employed by the estimation process of each refinement.

REFERENCES

[1] V. Lippiello and F. Ruggiero, "3D monocular robotic ball catching with an iterative trajectory estimation refinement," in *IEEE International Conference on Robotics and Automation*, (St. Paul, MN), pp. 3950–3955, 2012.

[2] V. Lippiello and F. Ruggiero, "Monocular eye-in-hand robotic ball catching with parabolic motion estimation," in *10th International IFAC Symposium on Robot Control*, (Dubrovnik, HR), pp. 229–234, 2012.

[3] V. Lippiello, F. Ruggiero, and B. Siciliano, "3D monocular robotic ball catching," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1615–1625, 2013.

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

[5] J. Kober, M. Glisson, and M. Mistry, "Playing catch and juggling with a humanoid robot," in *2012 12th IEEE-RAS International Conference on Humanoid Robots*, (Osaka, J), pp. 875–881, 2012.

[6] 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, PRC), pp. 5955–5962, 2011.

[7] D. Keren, S. Peleg, and R. Brada, "Image sequence enhancement using sub-pixel displacements," in *1988 IEEE Conference on Computer Vision and Pattern Recognition*, (Ann Arbor, MI, USA), pp. 742–746, 1988.

[8] L. Torabi and K. Gupta, "An autonomous six-DOF eye-in-hand system for in situ 3D object modeling," *The International Journal of Robotics Research*, vol. 31, no. 1, pp. 82–100, 2011.

[9] 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, ROC), pp. 702–707, 2003.

[10] V. Lippiello, F. Ruggiero, B. Siciliano, and L. Villani, "Visual grasp planning for unknown objects using a multifingered robotic hand," *IEEE/ASME Transactions on Mechatronics*, vol. 18, no. 3, pp. 1050–1059, 2013.

[11] Q. He, C. Hu, W. Liu, N. Wei, M. Meng, L. Liu, and C. Wang, "Simple 3-D point reconstruction methods with accuracy prediction for multiocular system," *IEEE/ASME Transactions on Mechatronics*, vol. 18, no. 1, pp. 366–375, 2013.

[12] D. Fernandes and P. Lima, "A testbed for robotic visual servoing and catching of moving objects," in *IEEE International Conference on Electronics, Circuits and Systems*, (Lisboa, P), pp. 475–478, 1998.

[13] 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, KOR), pp. 526–531, 2009.

[14] R. Mori and F. Miyazaki, "Examination of human ball catching strategy though autonomous mobile robot," in *IEEE International Conference on Robotics and Automation*, (Washington, DC), pp. 4236–4241, 2002.

[15] R. Mori, K. Hashimoto, and F. Miyazaki, "Tracking and catching of 3d flying target based on GAG strategy," in *IEEE International Conference on Robotics and Automation*, (New Orleans, LA), pp. 5189–5193, 2004.

[16] 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.

[17] 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, HI), pp. 1623–1629, 2001.

[18] E. Ribnick, S. Atev, N. Papanikolopoulos, O. Masoud, and R. Voyles, "Detection of thrown objects in indoor and outdoor scenes," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, (San Diego, CA), pp. 979–984, 2007.

[19] 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, F), pp. 3034–3039, 2008.

[20] S. Kim. and A. Billard, "Estimating the non-linear dynamics of free-flying objects," *Robotics and Autonomous Systems*, vol. 60, no. 9, pp. 1108–1122, 2012.

[21] G. Batz, A. Yaqub, H. Wu, K. Kuhnlenz, D. Wollherr, and M. Buss, "Dynamic manipulation: Nonprehensile ball catching," in *18th Mediterranean Conference on Control and Automation*, (Marrakech, MA), pp. 365–370, 2010.

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

[23] 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, CH), pp. 281–286, 2002.

[24] Z. Zhang, D. Xu, and M. Tan, "Visual measurement and prediction of ball trajectory for table tennis robot," *IEEE Transactions on Instrumentation and Measurement*, vol. 59, no. 12, pp. 3195–3205, 2010.

[25] Y. Huang, D. Xu, M. Tan, and H. Su, "Trajectory prediction of spinning ball for ping-pong player robot," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, (San Francisco, CA), pp. 3434–3439, 2011.

[26] J. Nonomura, A. Nakashima, and Y. Hayakawa, "Analysis of effects of rebounds and aerodynamics for trajectory of table tennis ball," in *SICE Annual Conference*, (Taipei, ROC), pp. 1567–1572, 2010.

[27] A. Nakashima, Y. Ogawa, Y. Kobayashi, and Y. Hayakawa, "Modeling of rebound phenomenon of a rigid ball with friction and elastic effects," in *American Control Conference*, (Baltimore, MD), pp. 1410–1415, 2010.

[28] A. Nakashima, Y. Ogawa, C. Liu, and Y. Hayakawa, "Robotic table tennis based on physical models of aerodynamics and rebounds," in *IEEE International Conference on Robotics and Biomimetics*, (Phuket, T), pp. 2348–2354, 2011.

[29] Y.-B. Jia, "Three-dimensional impact: energy-based modeling of tangential compliance," *The International Journal of Robotics Research*, vol. 32, no. 1, pp. 56–83, 2013.

[30] 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, J), pp. 94–99, 1997.

[31] S. Kim, A. Shukla, and A. Billard, "Catching objects in flight," *IEEE Transactions on Robotics*, 2014 (in press).

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

[33] R. Dahmouche, N. Andreff, Y. Mezouar, O. Ait-Aider, and P. Martinet, "Dynamic visual servoing from sequential regions of interest acquisition," *The International Journal of Robotics Research*, vol. 31, no. 4, pp. 520–537, 2012.

[34] 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, CDN), pp. 705–711, 1998.

[35] E. Malis, F. Chaumette, and S. Boudet, "$2\frac{1}{2}$D visual servoing," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 2, pp. 238–250, 1999.

[36] P. Corke and S. Hutchinson, "A new partitioned approach to image-based visual servo control," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 4, pp. 507–515, 2001.

[37] Y. Tanaka, T. Tsuji, and M. Kaneko, "Task readiness impedance in human arm movements for virtual ball-catching task," in *Conference of the IEEE Industrial Electronic Society*, (Roanoke, VA), pp. 478–483, 2003.

[38] O. Birbach and U. Frese, "A multiple hypothesis approach for a ball tracking system," in *7th International Conference on Computer Vision Systems: Computer Vision Systems*, (Liege, B), pp. 435–444, 2009.

[39] J. Bresenham, "A linear algorithm for incremental digital display of circular arcs," *Communications of the ACM*, vol. 20, no. 2, pp. 100–106, 1977.

[40] G. Antonelli, "Stability analysis for prioritized closed-loop inverse kinematic algorithms for redundant robotic systems," *IEEE Transactions on Robotics*, vol. 25, no. 5, pp. 985–994, 2009.

[41] F. Chaumette, *Potential Problems of Stability and Convergence in Image-based and Position-based Visual Servoing*. London, UK: Springer, 1998.

Pierluigi Cigliano was born in Naples, Italy, on October 19, 1970. He received the Laurea degree in Electronic Engineering from the University of Naples in 2012. He works as consultant engineer in the aerospace and defence field for the Altran group. He worked as freelancer at Department of Electrical Engineering and Information Technology at University of Naples. He also worked as assistant project supervisor for the technological revamp of Naples Metro Line 1.

Vincenzo Lippiello (M'12) was born in Naples, Italy, on June 19, 1975. He received his Laurea degree in electronic engineering and the Research Doctorate degree in information engineering from the University of Naples, in 2000 and 2004, respectively. He is an Assistant Professor of Automatic Control in the Department of Electrical Engineering and Information Technology, University of Naples. His research interests include visual servoing of robot manipulators, hybrid visual/force control, adaptive control, grasping and manipulation, aerial robotics, robotic ball catching, and visual object tracking and reconstruction. He has published more than 90 journal and conference papers and book chapters. He is member of the IFAC Technical Committee on Robotics.

Fabio Ruggiero (S'07 - M'10) was born in Naples, Italy, on December 16, 1983. He received the Laurea Specialistica degree (M.Sc.) in Automation Engineering from the University of Naples in 2007. He got the Ph.D. degree from the same institution in 2010. Fabio Ruggiero has been holding a PostDoctoral position at Department of Electrical Engineering and Information Technology at University of Naples since 2011. His research interests are focused on dexterous and dual-hand robotic manipulation, even by using UAVs with small robotic arms, dynamic non-prehensile manipulation, 3D object preshaping and reconstruction. He has co-authored about 25 journal papers, book chapters and conference papers.

Bruno Siciliano (M'91 - SM'94 - F'00) was born in Naples, Italy, on October 27, 1959. He received the Laurea degree and the Research Doctorate degree in Electronic Engineering from the University of Naples in 1982 and 1987, respectively. He is Professor of Control and Robotics, and Director of the PRISMA Lab in the Department of Electrical Engineering and Information Technology at University of Naples Federico II. His research interests include force and visual control, human-robot interaction, aerial and service robotics. He has co-authored 13 books, 80 journal papers, 240 conference papers and book chapters. He has delivered 110 invited lectures and seminars at institutions worldwide, and he has been the recipient of several awards. He is a Fellow of IEEE, ASME and IFAC. He has served on the editorial boards of several peer-reviewed journals and has been chair of program and organizing committees of several international conferences. He is Co-Editor of the Springer Tracts in Advanced Robotics, and of the Springer Handbook of Robotics, which received the PROSE Award for Excellence in Physical Sciences & Mathematics and was also the winner in the category Engineering & Technology. His group has been granted 14 European projects, including an Advanced Grant from the European Research Council. Professor Siciliano is the Past-President of the IEEE Robotics and Automation Society.