J Intell Robot Syst
DOI 10.1007/s10846-010-9527-3

Real-Time 3D Stereo Tracking and Localizing of Spherical Objects with the iCub Robotic Platform

Nicola Greggio · Alexandre Bernardino · Cecilia Laschi · José Santos-Victor · Paolo Dario

Received: 28 September 2010 / Accepted: 17 December 2010

Abstract Visual pattern recognition is a basic capability of many species in nature. The skill of visually recognizing and distinguishing different objects in the surrounding environment gives rise to the development of sensory-motor maps in the brain, with the consequent capability of object reaching and manipulation. This paper presents the implementation of a real-time tracking algorithm for following and evaluating the 3D position of a generic spatial object. The key issue of our approach is the development of a new algorithm for pattern recognition in machine vision, the Least Constrained Square-Fitting of Ellipses (LCSE), which improves the *state of the art* ellipse fitting procedures. It is a robust and direct method for the least-square fitting of ellipses to scattered data. In this work we applied it to the iCub humanoid robotics platform simulator and real robot. We used it as a base for a circular object localization within the 3D surrounding space. We compared its performance with the Hough Transform and the *state of the art* ellipse fitting algorithms, in terms of robustness (succes/failure in the object detection) and fitting precision. Our experiments involve robustness against noise, occlusion, and computational complexities analyses.

Keywords Machine vision · Pattern recognition · Least-square fitting · 3D tracking · Humanoid robotics

N. Greggio (✉) · C. Laschi
ARTS Lab—Scuola Superiore S.Anna, Polo S.Anna Valdera, Viale R. Piaggio, 34-56025, Pontedera, Italy
e-mail: nicola.greggio@sssup.it

N. Greggio · A. Bernardino · J. Santos-Victor
Instituto de Sistemas e Robótica, Instituto Superior Técnico, 1049-001, Lisboa, Portugal

P. Dario
ARTS & CRIM Lab—Scuola Superiore S.Anna, Polo S.Anna Valdera, Viale R. Piaggio, 34-56025, Pontedera, Italy

1 Introduction

The identification and the following of different objects is one the of the most valuable tasks for computer vision and image processing. Vehicle guidance, rescue robots, or surveillance systems are only few examples of applications that exploit this concept. Being the unpredictability of the real environment, these techniques are required to be robust and computational efficient. Robustness is closely linked to the environment the application has to perform into. For instance, in industrial applications (i.e. in structured scenarios) a non-robust algorithm can perform well because of the absence of noising disturbances. However, in non-constrained scenarios this is no longer true. The color, the shape, the moving speed of the object can affect the identification greatly. In non structured scenarios, like a rescue context, or simpler a video surveillance, there is the need of a substantial adaptability [33]. Therefore, the robustness for the adopted algorithms is a must. The other fundamental issue required by a vision identification algorithm is its good performance, in terms of computational efficiency. For instance, in hand-object manipulation or visual serving the camera and the algorithm must follow the dynamic responses of the robot manipulator for having a real-time control [20, 25]. It can be argued that, for example in video surveillance, a precise but very slow algorithm leads to high (and sometimes unacceptable) latency time between the image acquisition and its feature extraction. This is also more relevant in applications like vehicle guidance [23, 31].

The ellipse is the one of the nearly all common pattern to be recognized, due to their adaptability in modeling different primitives. They have been studied since many decades, due to their occurrence within several different contexts. For instance, ellipses are used in astronomy [11], medicine [24, 30], and robotics [36, 38]. Nevertheless, a great drawback is that the ellipse detection is generally highly computationally demanding [1, 41]. However, in robotics ellipses occur in many actual applications. Therefore, it is necessary to develop an algorithm that robustly detects and tracks an ellipse while operating at field rate. Many solutions have been proposed and used during the last decades. In 1994 Nelson and Khosla used ellipses for visual servoing applications [22]. Then, in 1999, Vincze studied the problem of fitting ellipses with a real camera by focussing most on the constraint of the real-timing [36]. Moreover, in 2002, Deniz et al. used an ellipse detection algorithm for face detection [3]. In their work the authors focussed more on Human-computer interaction. In 2004, Kwolek developed a method for tracking human heads with a mobile sterovision camera [17]. He characterized faces by first performing a color filtering, and then by modeling the head in the 2D image domain as an ellipse. Therefore, they formulated the tracking problem as a probabilistic one in which a particle filter is used to approximate the probability distribution by a weighted color cue, shape information, and stereovision sample collection. Then, in 2006 Teutsch et al. applied the ellipse recognition in industrial processes, focussing on the real-time characteristics of their approach [34].

1.1 Our Contribute

In this work we propose a 3D stereo tracker featuring the first implementation of a new pattern recognition algorithm for the least square of ellipses that improves the original one [6]. Herein we implemented for the first time in a real context our least-square fitting of ellipses technique [12].

Our algorithm takes advantages of the first proposal of [19], but implements a new solution for solving the numerical instability of the original technique. Moreover, a great advantageousness of our procedure is its very low computational complexity compared against [19]. We will demonstrate it in Section 3.2. This is twofold, because on one hand it allows us to apply it anytime, and not only when strictly necessary, while on the other hand it renders it perfect for those real-time applications in which the frame rate should be relatively high, like video surveillance, video guidance or in our case in robotics for tracking.

We tested our 3D stereo tracking algorithm for localizing of some objects in spatial coordinates. We used the a state of art robotics platform, the iCub, together with its simulator, in order to test our tracker at best [26]. So far, we not only improved a growing open project by adding new capabilities to the robot, but also made our program open-source, available to whose need it as tool for their personal research or for improving our work as well. We use a velocity control for moving the iCub head joints. Then, we will report the direct kinematics equations of the robot's head for the object space triangulation. Our aim is to compare the performance of the whole tracking procedure while using the previous ellipse recognition techniques [6, 19] and our new algorithm for tracking elliptical objects, and these three algorithms more the Hough Transform in case of circular objects (e.g. a ball) [15].

1.2 Paper Organization

This paper is organized as follows. In Section 2 we will discuss the state of the art problem of the least-square fitting of ellipses. Subsequently, in Section 3 we will propose our approach. Particularly, in Section 3.1 we illustrate our solution for the numerical instability of the original problem, while in Section 3.2 we provide a computational complexity evaluation of the whole procedure, comparing our approach versus the the state of the art algorithm. Then, in Section 4 we will describe the iCub robotics platform, in terms of its mechanics and the simulator we used. Furthermore, in Section 5 we will briefly explore our tracking algorithms, with its vision module (Section 5.1), its motor commands module (Section 5.2), and its kinematics module (Section 5.3). In Section 6 we will describe our experimental set-up. In Section 7 we will discuss our results. Finally, in Section 8 we will conclude our work.

2 Least Square of Ellipses

2.1 Least Square of Ellipses and Hough Transform: The State of the Art

Two main approaches can be considered for circle detection.

The first one is to use the Hough Transform [18, 42]. Since spatial perspective alters the perceived objects, there is the need of calibrating the cameras. Then, a pattern recognition algorithm, such as a simple color detection, can be applied and subsequently the Hough circle transform can be applied in order to estimate all the ball's features.

However, this approach can be complex to be implemented, and even elevate resource consumption. First, it requires the camera calibration. Moreover, it can be argued that using a Hough Transform, for instance, by augmenting the image's resolution the computational burden increases as well. In fact, since each pixel of the image can generate a five-dimensional space surface and the solution parameters are recovered by intersecting all these surfaces, it is clear that the higher is the resolution of the image and the bigger is the memory-consumption and the processor-consumption. Besides, much of the efficiency of the Hough Transform is dependent on the quality of the input data: the edges must be detected well for the Hough Transform to be efficient. Finally, the Hugh transform needs to be set well, in terms of the accumulator threshold at the center detection stage parameter.

The second one is to use ellipse specific pattern recognition algorithms, such as [6, 19]. By processing a ball thinking of it as it were an ellipse, we overcome the distortion problems. Circles in man-made scenes are almost always distorted when projected onto the camera image plane, therefore generating ellipses. Ellipses provide a useful representation of parts of the image since their detection is reasonably simple and reliable. Thus they are often used by computer vision systems for model matching [5, 7].

During the past few years there have been studied and developed many algorithms to fit an ellipse at best. Two different major approaches have been studied: The clustering/voting (CV) techniques, and the least-square techniques. The first ones adopt algorithms as RANSAC [27, 37], the previously cited Hugh Transform [18, 39, 40] and fuzzy clustering [2, 10]. The second ones adopt an optimization criteria that looks for the parameters of the objective equations that best fit the given set of data points [9]. Each approach presents advantages and disadvantages. In order to choose the best solution one has to consider the compromise between computational burden and algorithm's robustness. CVs are extremely robust techniques, very suitable for this applications. However, they are much too time-demanding and memory-intensive in order to be adopted in real-time applications, such as in mobile robotics localization or in object tracking, successfully [4, 29].

Therefore, some approaches based on Least Square (LS) techniques come out in recent years [6, 8, 19]. The principal reason is because of its computational costs. There are two main kinds of LS techniques: those based on the minimization of the algebraic distance between (Algebraic Distance Least Square, ADLS) the data points and the ideal curve (intended as the deviations of the implicit equation from the expected value (i.e. zero) at each given point) and those based on the minimization of the geometric distance (intended as the orthogonal, or shortest, distances from the given points to the geometric feature to be fitted) of the same data points (Geometric Distance Least Square, GDLS). ADLSs suffer of high curvature bias [16] with the the non-invariance to Euclidean transformation [43]. However, GDLSs suffer of being dependent of iterative algorithms [28] as do cluster/voting (CV) techniques, therefore making them not suitable for real-time applications [6]. This is a notable drawback, because iterative algorithms do not have a fixed computational time. Nevertheless, algebraic fitting algorithms may guarantee a direct one-step convergence. We will focus on this way, starting from the work of Fitgibbon et al., called B2AC, which will be described in the next section [6].

2.2 Analytical Background

A central conic can be expressed by a second order equation in its implicit form, as follows in the Eq. 1:

$$F(x, y) = ax^2 + bxy + cy^2 + dx + ey + f = 0 \tag{1}$$

This can also be expressed in the vectorial form:

$$F_a(\mathbf{x}) = \mathbf{x} \cdot \mathbf{a} = 0 \tag{2}$$

where $\mathbf{a} = [a, b, c, d, e, f]^T$ is the vector of the equation coefficients, and $\mathbf{x} = [x^2, xy, y^2, x, y, 1]$ is the vector of the points' coordinates, both relative to the conic section.

Considering that we have this set of data points:

$$T = \{(x_i, y_i) : i = 1...N\} \tag{3}$$

our aim is to minimize the sum of the squared distances of the curve (Eq. 1) to the given points (Eq. 3). In other words, by assuming $F(\mathbf{a}, \mathbf{p_i})$ as the *algebraic distance* from the point $\mathbf{p_i} = (x_i, y_i)$ to the conic expressed by Eq. 2 the following non-linear minimization problem has to be solved [4]:

$$\min_a \left(\sum_{i=1}^{N} F(\mathbf{a}, \mathbf{p_i}) \right) = \min_a \left(\sum_{i=1}^{N} F(\mathbf{a} \cdot \mathbf{p_i})^{\mathbf{2}} \right) \tag{4}$$

In [6] Fitzgibbon et al. demonstrated that solving the problem with the following constraints gives rise to a unique exact solution:

$$\begin{cases} \min \|\mathbf{D} \cdot \mathbf{a}\|^2 \\ \mathbf{a^T} \cdot \mathbf{C} \cdot \mathbf{a} = \mathbf{1} \end{cases} \tag{5}$$

where

$$\mathbf{D} = \begin{pmatrix} x_1^2 & x_1 y_1 & y_1^2 & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_N^2 & x_N y_N & y_N^2 & x_N & y_N & 1 \end{pmatrix} \tag{6}$$

and

$$\mathbf{C} = \begin{pmatrix} 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{7}$$

Now, by using the Lagrange multiplier λ and differentiating we obtain:

$$\begin{cases} 2\mathbf{D^T D a} - \mathbf{2}\lambda\mathbf{C a} = \mathbf{0} \\ \mathbf{a^T} \cdot \mathbf{C} \cdot \mathbf{a} = \mathbf{1} \end{cases} \tag{8}$$

or in the form:

$$\begin{cases} \mathbf{Sa} = \lambda \mathbf{Ca} \\ \mathbf{a^T} \cdot \mathbf{C} \cdot \mathbf{a} = \mathbf{1} \\ \mathbf{S} = \mathbf{D^T D} \end{cases} \tag{9}$$

Finally, Fitzgibbon and colleagues demonstrated that $\tilde{a}_i = \mu_i \mathbf{u_i}$ is a unique solution of the system equations in Eq. 5 [6], where:

$$\mu_i = \sqrt{\frac{1}{\mathbf{u_i^T C u_i}}} = \sqrt{\frac{\lambda_i}{\mathbf{u_i^T S u_i}}} \tag{10}$$

Therefore, the correspondent affine anti-transformation [19] needs to be performed after having found the optimal solution $\tilde{a}_6$.

Some improvements to the original method [6] have been made within the last years. One deserves particular noticing. In [19] it has been proposed to compute the following affine transformation to the input points before applying the Fitzgibbon's et al. algorihm [6]:

$$\tilde{x} = \frac{x - x_m}{s_x} - 1 \qquad \tilde{y} = \frac{y - y_m}{s_y} - 1 \tag{11}$$

where

$$x_m = \min_{i=1}^{N} x_i \qquad y_m = \min_{i=1}^{N} y_i \tag{12}$$

and

$$s_x = \frac{\max_{i=1}^{N} x_i - \min_{i=1}^{N} x_i}{2} \quad s_y = \frac{\max_{i=1}^{N} y_i - \min_{i=1}^{N} y_i}{2} \tag{13}$$

2.3 Drawbacks and Our Improvements

In 2006 Maini criticized the ill-conditioning of the scatter matrix $\mathbf{S} = \mathbf{D^T D}$ (Eq. 9) and proposed an affine transformation for solving it by recentering the ellipse points within a square with side length equals to 2 [19]. Moreover, in [19] it has been reported that the algorithm in [6] has a specific source of errors not mentioned in the paper, and that this causes numerical instabilities, giving rise to the fact that the closer the data points are to the ellipse (i.e. the less noise is present), the more difficult is to locate a unique solution. This results in the impossibility of having a solution (i.e. a precise and unique ellipse curve equation) when the data points lie exactly on, or too close to, the ideal ellipse curve. In [19], a resampling procedure has been proposed, that perturbs the data points with gaussian noise in the case of they are too close to the ellipse. However, this requires an excessive computational burden. In fact, he claims that the procedure must be applied *an adequate number of times M* in order to make the algorithm effectively robust. This makes this approach it not suitable for real-time applications. Nevertheless, in [19] it has been reported this great limitation in his work, advising to use it only when it is strictly required.

In this work we propose a new pattern recognition algorithm for the least square of ellipses that improves the original one [6]. Our solution takes advantage of the improvements given by [19] in terms of the ill-conditioning of the scattered matrix **S** (Eq. 9), and implements an alternative solution to the problem of the impossibility of having a solution when the data points lie too close to the ideal ellipse curve. Moreover, our approach is twofold, because on one hand it allows to overcome this instability problem, while on the other hand it can be always applied because of its extremely low computational complexity.

3 LCSE: Least Constrained Square-Fitting of Ellipses

3.1 Instability of the Exact Ellipse Solution

Despite the positive results published by the authors, the algorithm proposed in [6] has been reported to have an intrinsic source of error. This causes to find wrong results for the vector $\mathbf{a}$ in the system (Eq. 8), resulting in having $\mathbf{a} = \mathbf{0}_6$. This numerical instability comes when the ellipse points lie exactly on the ideal curve [19]. Being more specific, the closer the data points are to the ellipse (i.e. the less noise is present), the smaller is the solution.

In order to overcome this limitation, in [19] it has been proposed a resampling procedure that perturbs the data points with gaussian noise in the case of they are too close to the ellipse. However, the whole procedure is very time consuming, and therefore not suitable for real-time applications, contrariwise to what claimed in [19]. In fact, he reported that his resampling procedure must be applied at least from 50 to 100 times in order to make the algorithm effectively robust. The resultant ellipse is obtained by averaging the set of M ellipses obtained by repeating the resampling procedure M times [19]. Nevertheless, due to the time consuming, he suggested to apply it in the case of the solution $\mathbf{a}_6 = \mathbf{0}$ is found.

In this section we propose a technique that overcomes the previous problems. Instead of perturbing the original points with gaussian noise for many times, we decided to perturb the ellipse's polar transformation by adding a periodic symmetric function. We apply the data perturbation only if the case of not stable numerical solution, as in [19]. However, due to our low computational burden, it does not affect the total computation sensibly, and can therefore be used any time is required. The scheme is illustrated in Fig. 1.

The procedure is then described as follows:

(a) Application of the affine transformation [19].

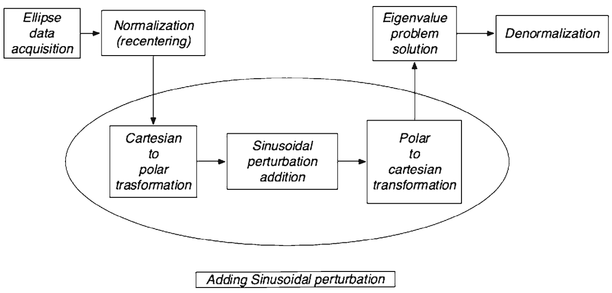

Ellipse data acquisition
Normalization (recentering)
Eigenvalue problem solution
Denormalization
Cartesian to polar trasformation
Sinusoidal perturbation addition
Polar to cartesian transformation
Adding Sinusoidal perturbation

Fig. 1 The iCub's Head. On the *left image* the head without the cover is shown, while in the *right image* the cover is shown

(b) Transformation from the cartesian coordinates (x, y) to the polar ones (ρ, θ). The point i results:

$$\begin{aligned}
\rho_i &= \sqrt{x_i^2 + y_i^2} \\
\theta_i &= \arctan\left(\frac{y_i}{x_i}\right); \qquad with: \ x_i \geq 0, y_i \geq 0 \\
\theta_i &= \frac{\pi}{2} - \arctan\left(\frac{x_i}{y_i}\right); \quad with: \ x_i \leq 0, y_i \geq 0 \\
\theta_i &= \pi + \arctan\left(\frac{y_i}{x_i}\right); \quad with: \ x_i \leq 0, y_i \leq 0 \\
\theta_i &= \frac{3\pi}{2} - \arctan\left(\frac{x_i}{y_i}\right); \quad with: \ x_i \geq 0, y_i \leq 0
\end{aligned} \tag{14}$$

Our aim is to move the points around their initial position, but maintaining the ellipse average over the whole polar representation period (2π) within its polar representation. Any symmetric periodic function with period taken as integer multiplier of 1 added to the original scattered data leaves the ellipse polar average unaltered. Therefore, we choose the sinusoidal function, being continuos, easy to be implemented, with zero average and infinitely derivable.

(c) We choose the amplitude equals to $A = 0.001$ and the frequency equals to $f =$ 1,000 Hz. The point i obeys to:

$$\hat{\rho}_i = \rho_i + A \cdot sin(2\pi f \theta_i); \tag{15}$$

(d) When the ellipse is remapped in cartesian coordinates, its results equally slightly perturbed inside and outside its ideal curve, which is the curve that best interpolates these data. It results, for the point i:

$$\begin{aligned}
\hat{x}_i &= \hat{\rho}_i \cdot \cos(\theta_i); \\
\hat{y}_i &= \hat{\rho}_i \cdot \sin(\theta_i);
\end{aligned} \tag{16}$$

(e) Now the ellipse is ready to be fitted by building the design matrix (Eq. 7), and by solving the eigenvalues problem (Eq. 9).

(f) Finally, the affine denormalization transformation of the point **(a)** has to be applied.

Figure 2 shows the original ellipse after the recentering procedure (top-left), that represented in polar coordinates (middle), the polar transformed ellipse with the sinusoidal perturbation (bottom), and the resultant perturbed ellipse (top-right).

3.2 Computational Burden Analysis

Now we analyze the computational complexity of the algorithm proposed in [19], and our technique. Low computational complexity means higher frame rates in real-time applications, and therefore faster control loops. This results essential in many actual applications [17, 36]. Our new approach is able to eliminate the numerical instability

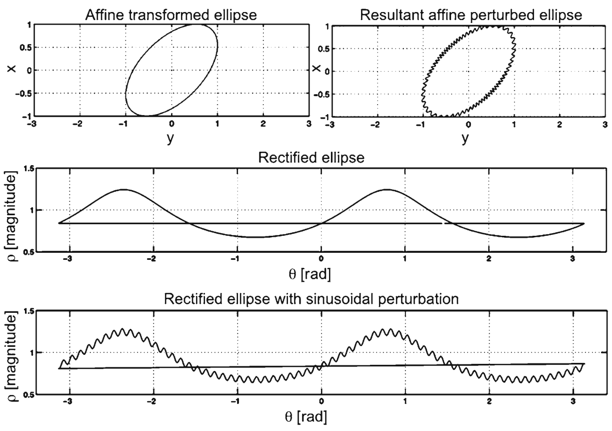


Fig. 2 The original ellipse after the recentering procedure (*top-left*), that represented in polar coordinates (*middle*), the polar transformed ellipse with the sinusoidal perturbation (*bottom*), and the resultant perturbed ellipse (*top-right*)

that affects the original algorithm [6] as [19] does, but greatly faster. We consider N being the number of points composing the ellipse scattered data.

Now we will describe:

a. The resampling procedure proposed in [19];
b. Our new approach;
c. The final comparison between these two algorithms.

→ a. *Resampling procedure*—[19]: The complete procedure has been explained in [19]. For each point a gaussian noise component is added. Therefore, this operation goes with $O(N)$. Therefore the sequence of operations 2, 3, 4, 5 has to be performed. This process goes with $O(6N) + O(42N)$, repeated for M times. Thus, the resultant complexity is $O(49MN)$. Finally, an averaging procedure through all the ellipse data has to be performed, which makes the overall process going with $O(MN) + O(49MN) = O(50MN)$.

→ b. *Add sinusoidal perturbation—our new method*: This adds the sinusoidal perturbation to the ellipse data after having been transformed into polar coordinates (originally, they are expressed in cartesian representation). Thus there are three operations to be performed: the first one is the transformation of all the data points from cartesian to polar representation. This takes $O(2N)$. After that, the addition of the sinusoidal perturbation takes $O(N)$ operations. Then, the polar coordinates are remapped into cartesian ones, taking $O(2N)$. Therefore, the whole operation goes with $O(5N)$.

→ c. *Computational comparison and improvement*. In [19] it has been suggested $50 \leq M \leq 200$. Moreover, in [19] it has been reported that EDFE performed better performances than B2AC for $M \geq 200$. However, it is clear that repeating the resampling procedure more than 200 times costs a very high computational burden. Even if M were equal to 50, our procedure is 500 times faster than [19]. In fact, by comparing [19] and our procedures for eliminating the numerical instability, i.e. the passages 8 and 9, respectively, it is possible to see that our procedure is faster of $O(50MN)/O(5N) \Rightarrow 10M = 10 \cdot 50 = 500$ times.

4 The iCub Robotics Platform

The iCub robotics platform is one of the most advanced state of the art robots. This robot has been anthropomorphically developed to fit the mechanical abilities of a two-year-old child at best. This particular morphology gives rise the scientists to test developmental cognitive models based on related children studies on it with a reasonable approximation of the results. It can be argued that a neuro-scientific model of a particular child behavior needs to be tested on a platform capable of replicating analogous movements in order to be validated. In fact, if the robotics platform performs the developmental way suggested by the neuroscientists obtaining analogous results as the tested subjects, it signifies that the model can explain these particular human internal models as well. However, if not, it can lead to new suppositions and new ideas that can advance the comprehension of the human behaviors, in any case. Therefore, new models will be developed and new theories will be validated.

4.1 The iCub Overall Mechanics Specifications

The robot is composed of 53 degrees of freedom (DOFs). Most of them are directly actuated, such as the shoulders, others are under-actuated [32]. This has been decided according to the placement of the actuators which is heavily constrained by the shape of the body. Of course, the shape is not the only important factor in the robot's realization.

The shoulders contain the three motors required for each shoulder. A single aluminum block encapsulates these three motors. The joint is tendon driven, while the motors do not move with respect to each other. A particular characteristic suitable for reaching and manipulation is the orientation of the shoulder's joints and their motor group. In fact, they have been designed at an angle with respect to the front-back midline to position the range of motion as frontal as possible which clearly enhances the manipulation workspace of the arms. In Fig. 3 is shown the iCub in its final configuration [26].

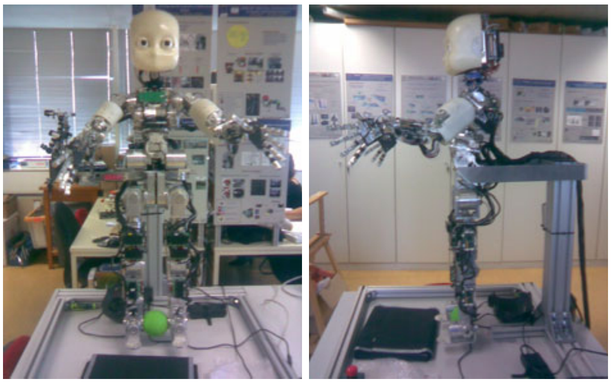


Fig. 3 The iCub whole robot

(a) The iCub: Full front view (b) The iCub: Full left side view

4.2 The iCub Head Specifications

The iCub's head is completely based on the Faulhaber motors (Fig. 4). These are driven by DC micromotors (Faulhaber) with planetary gearheads. The neck consists of a serial chain of rotations and it has three DOF, which have been placed in a configuration that best resembles human movements. The mechanism of the eyes has been developed in order to achieve three degrees of freedom too. Both eyes can tilt (i.e. to move simultaneously up and down), pan (i.e. to move simultaneously left and right), and verge (i.e. to converge or diverge, with respect to the vision axes). The pan movement is driven by a belt system, with the motor behind the eye ball. The eyes' tilt movement is actuated by a belt system placed in the middle of the two eyes [21].

4.3 The ODE iCub Simulation

This can be addressed by considering a general and specific issue.

4.3.1 Why Using a Simulator? General Issue

There are many reasons for which it is important to test new algorithms within a simulator in order to debug them safely [14]. One is the low number of available platform. Since there have been build only a few prototypes (less than ten), it is not easy to access the robot. One of these prototype is in the Italian Institute of Technology, in Genoa, Italy (this is the center the robot has been developed in). Clearly, this can be very expensive, especially when more people have to stay abroad for many days in order to perform their experiments. A simulator solves these

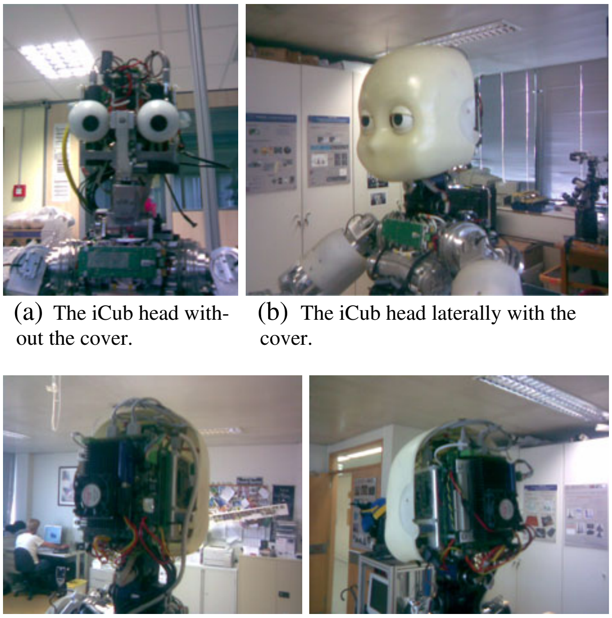


Fig. 4 The iCub's Head. The first two pictures represent the robot in a frontal view, while the other two from behind (where it is possible to see the pc104 board that commands its movements)

(a) The iCub head without the cover.

(b) The iCub head laterally with the cover.

(c) The head from behind, right view.

(d) The head from behind, left view.

problems. Scientists can perform their experiments without being close and compare the results finally. Moreover, the iCub platform can be extremely dangerous if not used properly. The motors torque and power can injury a human being seriously.

4.3.2 Why Using a Simulator? Specific Issue

On one side, it can be argued that the simulator's information is not exhaustive, but it is a good approximation for the software debugging before using it on the real robot. On the other side, our algorithm claims to overcome the original Fitzgibbon's approach drawback of failing in detecting the ellipse when the curve lies on the ideal curve (i.e. is case of noise absence) [19]. It is clear that image segmentation in the real robot will never, or very seldom, produce perfect ellipses after image segmentation, due to all the imperfection within the real word (light gradients, light contrasts, color gradients, not regular object shapes, etc.), therefore testing this ellipse pattern recognition algorithm to the real robot will not produce comprehensive results. Contrariwise, the simulator does not present these artifacts, or at least it limits them. For instance, Figs. 7 and 9 show the ball after our image processing with the simulator and the real robot, respectively. In Fig. 7 the object is perfectly recognized as an ellipse, while the real one with the real robot, in Fig. 9, presents an irregular boundary (as clearly expected). However, this make its fake world perfunctory in some ways, like in fact the image acquisition and processing. Therefore, at a first glance, it may seem worthless presenting an algorithm for the simulator specifically. Nevertheless, the simulator provides many remarkable advantages, as previously mentioned before in Section 4.3.

4.3.3 The Choice of the Simulator

There exist different kinds of simulators. Tikhanoff et al. developed a completely open source simulator for the iCub [35], based entirely on the *ODE* (Open Dynamic

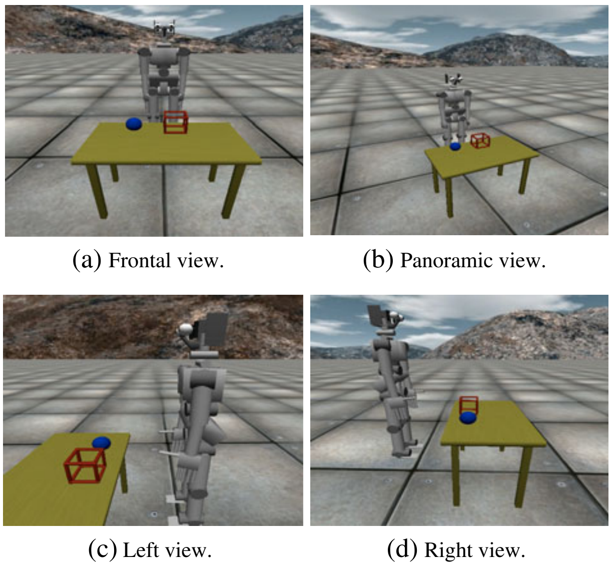


Fig. 5 The iCub's simulator. This shows the environment, together with some objects (a table where lie a ball an an empty cube), from different views

(a) Frontal view. (b) Panoramic view. (c) Left view. (d) Right view.

Engine). A pair of important points are that, first it is freely available, and second it resemble the real robot in part. The simulated robot has the same number of degrees of freedom of the real one. Then it allows to get information about encoders for each joint and tactile sensors on the hands like the original iCub. Moreover, it can be driven by using the same commands as for the real one. Therefore, it is possible to interface to both robots without changing the code.

We use this simulator in order to test our algorithms (Fig. 5).

5 AI034-Cub: The Robot Controlling Tool

The robot controlling tool is a common tracker program. It identify a single unknown non-convex object by means of the robot's cameras, and then it follows it with the eyes and the neck for finally detecting its position within the 3D surrounding space. The latter is determined by the robot's (and in particular the head's) direct kinematics, subject to a reference spatial coordinate system which origin is located on the ground, where the robot lies. Its main feature is the usage of our pattern recognition algorithm. We implemented this program as a tool for our research objective. It is comprehensive of a more complete project. We have to fulfill the deliverable 3.5 of the RobotCub project [26], relative to the implementation of the sensorimotor coordination for reaching and grasping.

The program is composed by three modules:

- The Vision Module;
- The Motor Control Module;
- The Kinematics Module.

The whole program architecture is shown in Fig. 6. Here it is illustrated how AI034-Cub proceeds for tracking the target. There is a main process. At each main process iteration the images from both the robot's cameras are sent to the Vision Module. This is a double thread module (one for each camera), which operates the image processing needed for extracting the object features (position, dimension, etc.). The target's position within 2D reference system given by each camera pixel division is then sent to the Motor Command Module, together with the cameras dimension (actually it is sufficient to try to position the target within only one camera's center—we considered the left one—the other eye will receive the opposite correspondent moving commands to pair with the so-called reference one). The Motor Command Module will send velocity commands to the eyes in order to position the target's COG (Center Of Gravity) within both cameras 2D coordinates. This is repeated until the latter achievement is reached. Then the Kinematics Module evaluates the object's COG position within the 3D surrounding environment by means of the known robot's direct kinematics together with the joint's angles information provided by the encoders.

5.1 The Vision Module

The Vision Module is responsible for processing the input images from the iCub head cameras in order to obtain the relevant information about the object to be grasped. These are: shape, dimension, orientation, and position within the 3D surrounding environment (this is accomplished by triangulating the information received from

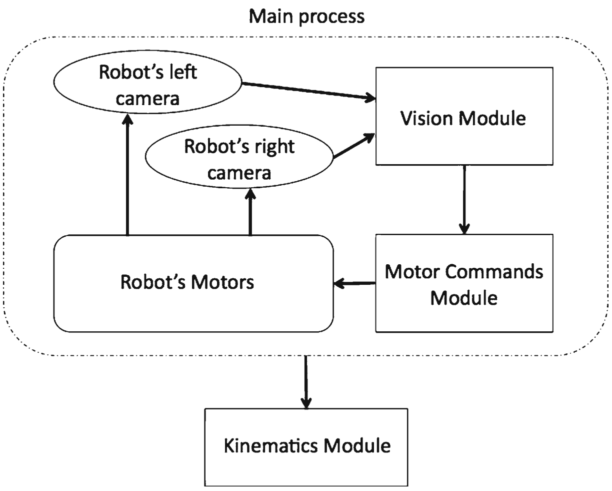


Fig. 6 The whole AI034-Cub architecture. At each main process iteration the images from both the robot's cameras are sent to the Vision Module. Then the target's position within 2D reference system for each camera pixel is then sent to the Motor Command Module which provides velocity control commands to the head motors in order to center the target within each camera. Then (actually, after each main process iteration) the Kinematics Module evaluates the object's COG position within the 3D surrounding environment by means of the known robot's direct kinematics together with the joint's angles information provided by the encoders

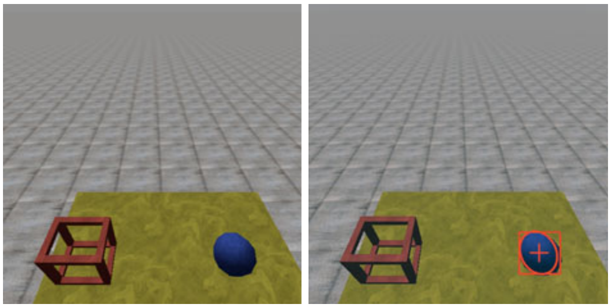
the binocular vision, the head and the neck encoders). In our particular case we made our experiments by using a ball of different colors as test object.

In order to detect the ball, and all its features, we implemented an image processing algorithm. It identify the ball region within the image by means of a color histogram filter.

We identify the ball by means of a color filter, The object detection is performed by using of a sample color recognition procedure. The reference color threshold has been selected from images captured by simulated robot's cameras. Each color (detected with an image of interest) is characterized by its HSV (Hue, Saturation, and Value) color histogram representation. Once the ball pixels are identified by the color filtering, the image is converted into a binary one with ball pixels set to '1'. Since the binary image contains not only the blob relative to the ball, but also other smaller blobs caused by color variation in the image, we applied *connected components* labeling algorithm for distinguishing each blob. We simply assume the largest blob is the ball, so we look for the blob with the largest area. Subsequently, we proceeded by applying our LCSE algorithm, previously described in Section 3, to the found blob, in order to detect all the parameters of the curve that describes the boundary of the blob.

We slightly modified the simulator in order to create different scenarios for our experiments (such as by changing the color of the ball, by removing the table, etc.). In Fig. 7a the input to the left camera is presented, i.e. the experimental scenario, while in Fig. 7b output of the algorithm is presented. These images are the input image as seen by the robot with the egocentric view (Fig. 7a) and the same image with the superimposition of an ellipse, drawn by using the characteristic parameters obtained

Fig. 7 The input image, as seen by the robot within the simulator with the egocentric view (**a**) and the same image with the superimposition of an ellipse, drawn by using the characteristic parameters obtained by computing the LCSE (**b**)



(a) The left camera output. (b) The object recognized within the left camera.

by computing the EDFE (Fig. 7b). In Fig. 8a the input to the real robot left camera is presented, while in Fig. 8b the backprojection of the same scene is presented.

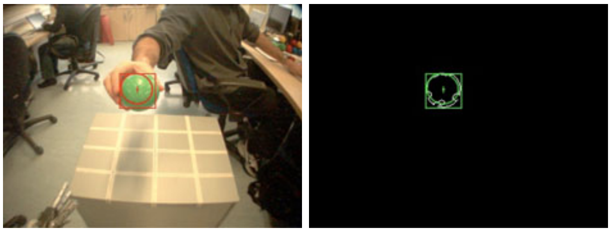
5.2 The Motor Control Module

This is the part responsible for moving the robot's joints. Due to the robot's firmware, we opted for a velocity control rather than a position control, which renders the movements more fluids. In this case, the velocity is sent to each joint instead of an angular position.

The target COG's coordinates are sent by the Vision Module, with respect to the (x, y) camera pixel coordinates, with 0 located at top-left. This is compared with the left camera's center (using the left or right camera is not important), and a velocity is send to the eyes' tilt, version, and vergence with opposite sign of the measured pixel difference. Then, opposite values are sent also to the neck pitch and yaw motors in order to compensate the eyes' movements (Fig. 9). The pseudocode is shown in Algorithm 1 (with reference to Table 1 for the velocity control gains).

In order to have the best compromise between fast tracking and convergence (too high values for the gains generates over-oscillations) we set the joint velocity gains and velocity to acceleration ration as in Table 1.

Fig. 8 The processed input image, as seen by the real robot (**a**) and the backprojection of the same image with object EDFE highlighting (**b**)



(a) The real robot processed left camera output. (b) The back projection of the object recognized within the real robot left camera.

Algorithm 1 Motor Commands Module: Object Tracking Pseudocode

```
1: Camera resolution = $width \cdot height$;
2: Object coordinates within the left camera = $(x_L, y_L)$;
3: Object coordinates within the right camera = $(x_R, y_R)$;
4: - tracked = false;
5: **while** (tracked) **do**
6:     - Read target position from the Vision Module;
7:     - $diffTilt = \frac{Height}{2} - y_L$
8:     - $diffVersion = x_L - (Width - x_R)$
9:     - $diffVergence = x_L - \frac{Width}{2}$
10:    **if** (diffTilt > $err_Y$ || diffTilt < $-err_Y$) **then**
11:        - tracked = tracked && false;
12:    **end if**
13:    **if** (diffVersion > $err_X$ || diffVersion < $err_X$) **then**
14:        - tracked = tracked && false;
15:    **end if**
16:    **if** (diffVergence > $err_X$ || diffVergence < $err_X$) **then**
17:        - tracked = tracked && false;
18:    **end if**
19:    Eyes movement
20:    - Set velocity eyes (tilt, version, vergence) = (eyesTiltGain, eyesVersionGain,
       eyesVergenceGain) · | (diffTilt, diffVersion, diffVergence) |;
21:    - Set acceleration eyes (tilt, version, vergence) = acceleration2VelocityRatio ·
       velocity eyes (tilt, version, vergence);
22:    - Send velocity commands to eyes joints;
23:    Neck movement
24:    - Read the encoders values.
25:    - Set velocity neck (pitch, yaw) = (neckPitchGain, neckYawGain) · |eyes (tilt,
       version) value |;
26:    - Set acceleration neck (pitch, yaw) = acceleration2VelocityRatio · velocity
       neck (pitch, yaw);
27:    - Send velocity commands to neck joints;
28: **end while**
```

5.3 The Kinematics Module

The iCub program we implemented localizes the position of the ball (which is the target to be grasped in this case), in terms of 3D cartesian position. We adopted the same system reference as the simulator, in order to be fully compatible with the measures and the signs adopted in the virtual environment (as shown by the blue axes in Fig. 10).[1]

[1]The reference system is centered on the floor plane, at the center of the pole that sustains the robot. The x axis evolves along the front of the robot, the y axis runs along the left of the robot, and the z axis evolves along its height.

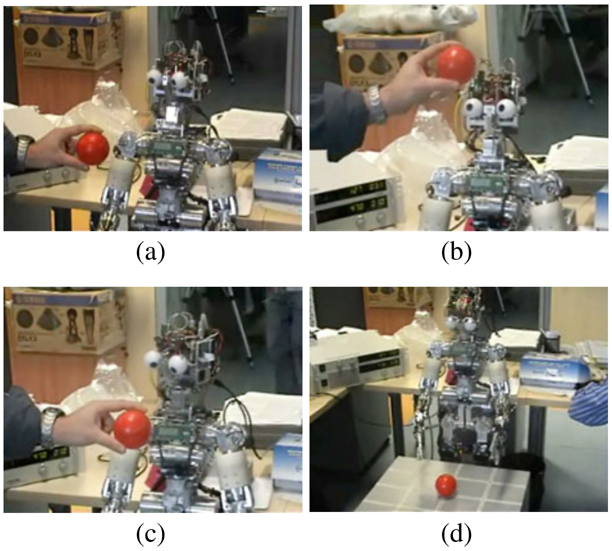


Fig. 9 The real robot tracking the ball. It follows the ball and then it evaluates its 3D position

The object's COG coordinates $(x_{\mathrm{COG}}, y_{\mathrm{COG}}, z_{\mathrm{COG}})$ are evaluated, with reference to the convention adopted in Fig. 10 both in cartesian notation and with the Denavit–Hartemberg convention. Using the cartesian notation these are as follows:

$$\begin{aligned}
x_{\mathrm{COG}} &= \frac{1}{2} E_{YD} \frac{\sin\left(90 - (E_{VG} + E_{VS}) \cdot \frac{\pi}{180}\right)}{\sin\left(E_{VG} \cdot \frac{\pi}{180}\right)} \cos\left(E_{VS} \cdot \frac{\pi}{180}\right) \\
&\quad + E_{XO} + (E_{ZO} - H_{ZO}) \left(-\sin\left(N_{PI} \cdot \frac{\pi}{180}\right)\right) \\
y_{\mathrm{COG}} &= -\frac{1}{2} E_{YD} \frac{\sin\left(90 - (E_{VG} + E_{VS}) \cdot \frac{\pi}{180}\right)}{\sin\left(E_{VG} \cdot \frac{\pi}{180}\right)} \sin\left(E_{VS} \cdot \frac{\pi}{180}\right) \\
z_{\mathrm{COG}} &= \left\{ \frac{1}{2} E_{YD} \frac{\sin\left(90 - (E_{VG} + E_{VS}) \cdot \frac{\pi}{180}\right)}{\sin\left(E_{VG} \cdot \frac{\pi}{180}\right)} \cos\left(E_{VS} \cdot \frac{\pi}{180}\right) + E_{XO} + (E_{ZO} - H_{ZO}) \right. \\
&\quad \left. \times \left(-\sin\left(N_{PI} \cdot \frac{\pi}{180}\right)\right) \right\} \cdot \tan\left((E_{TI} + N_{PI}) \cdot \frac{\pi}{180}\right) \\
&= x_{\mathrm{COG}} \cdot \tan\left((E_{TI} + N_{PI}) \cdot \frac{\pi}{180}\right) \qquad (17)
\end{aligned}$$

Table 1 Velocity control gains

		Simulated robot	Real robot
Eyes tilt gain	G_{E-TI}	3	0.24
Eyes version gain	G_{E-VS}	3	0.2
Eyes vergence gain	G_{E-VG}	2.6	0.28
Neck pitch gain	G_{N-PI}	3	0.28
Neck yaw gain	G_{N-YW}	3	0.16
Acceleration to velocity ratio	$A2V$	40	40

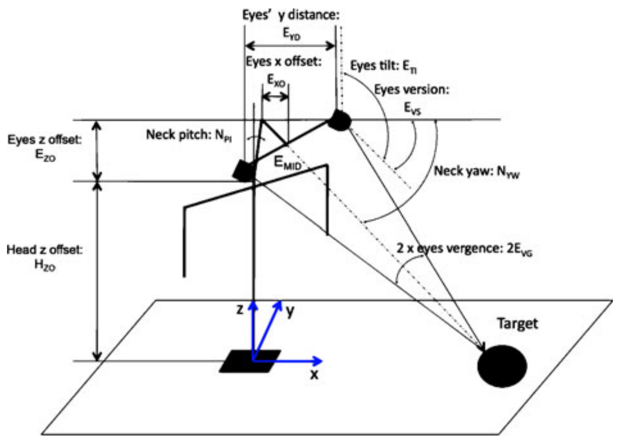


Fig. 10 Schematization of the iCub's kinematics. This is not all the kinematics, of course. We focussed on the head and neck's joints

The eyes' middle axis point $E_{\text{MID}} = (x_{\text{MID}}, y_{\text{MID}}, z_{\text{MID}})$ is given by:

$$\begin{aligned}
x_{\text{MID}} &= E_{XO}\cos\left(N_{PI}\cdot\frac{\pi}{180}\right) + (E_{ZO}-H_{ZO})\sin\left(N_{PI}\cdot\frac{\pi}{180}\right)\\
y_{\text{MID}} &= 0\\
z_{\text{MID}} &= E_{XO}\cos\left(N_{PI}\cdot\frac{\pi}{180}\right) + (E_{ZO}-H_{ZO})\cos\left(N_{PI}\cdot\frac{\pi}{180}\right) + H_{ZO} \qquad (18)
\end{aligned}$$

Since there's no documentation about the simulator internal kinematics, i.e. the robot's body part length, we derived them by reading the source code. In Table 2 we report them.

Then, the Denavit–Hartemberg convention for the object's COG coordinates is analyzed. In Fig. 11 we shared the system into different reference systems. No. 0 is where the iCub lies on the ground. This is the final reference system the target's coordinates are referred to (as in the previous notation, the cartesian one). Nos. 1 and 2 are the neck pitch and yaw, respectively. Then, nos. 3 and 4 are the eyes

Table 2 Body kinematics lengths

Simulated robot (SMU)	Real robot (mm)	Symbol	Meaning
0.35	68	E_{YD}	Eyes y distance: Related to the y axis, it represents the distance between the eyes along their axis
0.36	54	E_{XO}	Eyes x offset: Related to the x axis, it represents the distance between the neck pitch axis and the eyes axis
3.50	115.5	E_{ZO}	Eyes z offset: Related to the z axis, it represents the distance between the neck yaw axis and the eyes axes
3.01	1,023	H_{ZO}	Head z offset: Related to the z axis, it represents the distance between the origin of the reference system and the neck pitch joint

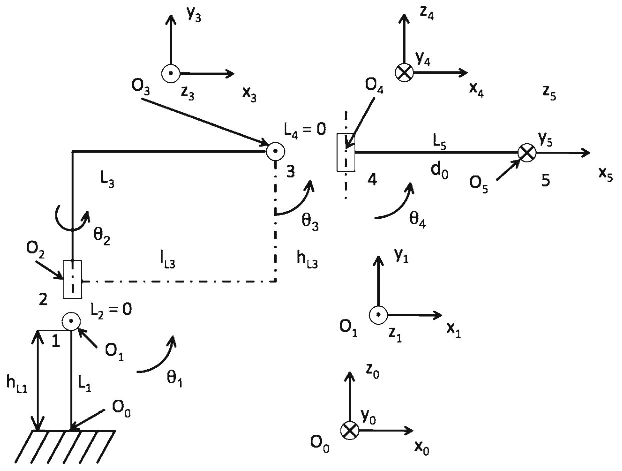


Fig. 11 Kinematic chain model for the Denavit–Hartenberg convention

tilt and version, respectively. Finally, no. 5 represents the object. Table 3 shows the parameters of the D-H symbols notation of Fig. 11.

Here, the angles represent:

θ_1^*:	Neck pitch—positive up
θ_2^*:	Neck yaw—positive left
θ_3^*:	Eyes tilt—positive up
θ_4^*:	Eyes version—positive left

Then, d_0 represents the target's COG distance from the eyes' middle axis point E_{MID} (see Fig. 10). This is evaluated with a simple geometrical relationship, as follows:

$$d_0 = \frac{E_{YD}}{2}\tan\left(\frac{\pi}{2} - E_{VG}\right) = \frac{E_{YD}}{2}\tan^{-1}\left(E_{VG}\right) \tag{19}$$

The target's COG coordinates $(x_{\text{COG}}, y_{\text{COG}}, z_{\text{COG}})$ are evaluated, with reference to the convention adopted in Fig. 11 as follows:

$$\begin{bmatrix} x_{\text{COG}} \\ y_{\text{COG}} \\ z_{\text{COG}} \\ 1 \end{bmatrix} = \left[T_0^5\right] \begin{bmatrix} x_5 \\ y_5 \\ z_5 \\ 1 \end{bmatrix} \tag{20}$$

Table 3 Body kinematics Denavit–Hartenberg parameters

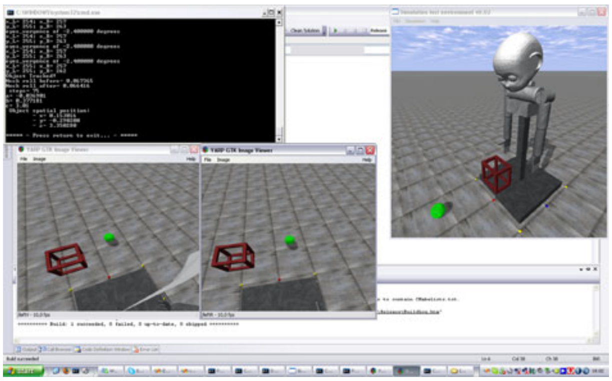
Link	a_i	α_i	d_i	θ_i
L_1	0	$\pi/2$	h_{L1}	0
L_2	0	$-\pi/2$	0	θ_1^*
L_3	l_{L3}	$\pi/2$	h_{L3}	θ_2^*
L_4	0	$-\pi/2$	0	θ_3^*
L_5	d_0	0	0	θ_4^*

with:

$$T_0^5 = \begin{bmatrix} \begin{matrix}\cos\theta_1^*\cos\theta_2^*\cos\theta_3^*\cos\theta_4^* + \\ -\cos\theta_1^*\sin\theta_2^*\sin\theta_4^*\end{matrix} & \begin{matrix}-\sin\theta_4^*\cos\theta_1^*\cos\theta_2^*\cos\theta_3^* + \\ -\cos\theta_1^*\sin\theta_2^*\cos\theta_4^*\end{matrix} & -\sin\theta_3^*\cos\theta_1^*\cos\theta_2^* & \begin{matrix}d_0\cos\theta_1^*\cos\theta_2^*\cos\theta_3^*\cos\theta_4^* + \\ -d_0\sin\theta_4^*\cos\theta_1^*\sin\theta_2^* + \\ l_{L3}\cos\theta_1^*\cos\theta_2^*\end{matrix} \\ \begin{matrix}\sin\theta_2^*\cos\theta_3^*\cos\theta_4^* + \\ \cos\theta_2^*\sin\theta_4^*\end{matrix} & \begin{matrix}\sin\theta_2^*\sin\theta_4^*\cos\theta_3^* + \\ \cos\theta_2^*\cos\theta_4^*\end{matrix} & -\sin\theta_2^*\sin\theta_3^* & \begin{matrix}-d_0\cos\theta_3^*\cos\theta_4^*\sin\theta_2^* + \\ d_0\cos\theta_2^*\sin\theta_4^* \\ l_{L3}\sin\theta_2^*\end{matrix} \\ \begin{matrix}\sin\theta_1^*\cos\theta_2^*\cos\theta_3^*\cos\theta_4^* + \\ -\sin\theta_1^*\sin\theta_2^*\sin\theta_4^*\end{matrix} & \begin{matrix}-\sin\theta_1^*\sin\theta_4^*\cos\theta_2^*\cos\theta_3^* + \\ -\sin\theta_1^*\sin\theta_2^*\cos\theta_4^*\end{matrix} & \begin{matrix}-\sin\theta_1^*\sin\theta_3^*\cos\theta_2^* + \\ \cos\theta_1^*\cos\theta_3^*\end{matrix} & \begin{matrix}d_0\cos\theta_2^*\cos\theta_3^*\cos\theta_4^*\sin\theta_1^* + \\ -d_0\sin\theta_1^*\sin\theta_2^*\sin\theta_4^* \\ l_{L3}\sin\theta_1^*\cos\theta_2^* + \\ h_{L1}\cos\theta_1^* + h_{L1}\end{matrix} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

In Fig. 12 a screenshot is depicted (an old version, the second one, under Windows), that shows an operative situation in which the simulator tracked the ball. Then, Fig. 13 shows a more recent release (under OS X). Clearly the behavior of the simulators is the same, regarding the functionalities we use.

Clearly, the simulator information is not exhaustive, but it is a good approximation for the software debug before using it on the real robot.



```
x_L= 354; x_R= 357
y_L= 355; y_R= 363
eyes_vergence of -2.400000 degrees
x_L= 354; x_R= 357
y_L= 355; y_R= 363
eyes_vergence of -2.400000 degrees
x_L= 354; x_R= 357
y_L= 355; y_R= 363
eyes_vergence of -2.400000 degrees
x_L= 355; x_R= 357
y_L= 355; y_R= 363
Object Tracked!
Neck roll before= 0.067765
Neck roll after= 0.064416
 steps= 75
a= -0.036981
b= 0.177181
c= 1.00
 Object spatial position:
        - x= 0.153016
        - y= -0.2540300
        - z= 3.3540300

===== - Press return to exit... - =====
```

Fig. 12 A screenshot depicting the moment in which the simulated robot tracked the position of the ball in the 3D surrounding environment. Therefore, our program uses the encoders information to triangulate the position of the centroid of the object within the simulated space. This is the second version of the simulator

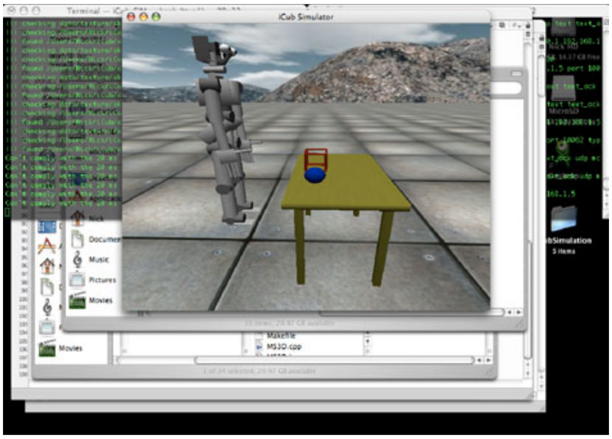


Fig. 13 A more recent screenshot of the iCub's simulator during an operative situation

6 Experimental Set-Up

We performed three types of experiments, in order to validate the LCSE pattern recognition algorithm [19] compared with the Hough Transform and the least square ellipse fitting algorithm, B2AC [6]. Each of these tests has a well specified scenario, described in the next section. For each scenario we performed the same experiments with the Hough Transform, the B2AC, and the EDFE algorithms. We used calibrated cameras. We tested these techniques under the same experimental conditions aimed by several reasons. First, we would check the performance differences among these methods, intended as produced error. It is worth noticing we are not interested in the absolute error of each procedure (yet evaluated for each method in [6, 18] and [19]); nonetheless we are concerned in verifying the systems' execution dissimilarities under the same situation. Moreover, we are not interested in analyzing these dissimilarities in terms of mathematical performance, as done by the authors in [6, 18] and [19], but their usage in practical applications and scenarios instead. The importance of choosing the best method is obvious, because reducing the error since the beginning brings about a more precise results at the end. In the next section we will analyze the error propagation process, and we will quantize it in our specific case (see Section 7.1).

Of course, we made the same algorithms working both on the simulator and the real robot, although here we present the results obtained with the simulator due to their major validity with regard to this specific work, as explained in the previous section.

6.1 Scenarios

At each trial the Hough Transform, the B2AC, and the LCSE algorithms are used in order to evaluate the ball's center of mass (COM) within the 2D camera images. Therefore this information is triangulated with the encoders' values in order to determine the ball spatial position. For each scenario we performed at least 30 trials.

Since there is a prospective error, introduced by the spatial perspective, the ball is not seen as a 2D circle by the two camera. Instead, it is perceived distorted. This causes an artifact during these scenarios experiments that is not due to the goodness of the three tested algorithms. Therefore, in order to evaluate their performance without this systematic error, we made another test. We made the experiment in the scenario no. 1 by using a cylinder with a negligible height (so that it can be assumed to have a null depth, hence reducing the prospective effect), and putting the cylinder exactly in front of the eyes axis midpoint. The cylinder has been obtained as a section of the ball. In this way we can test the algorithms by isolating the perspective error, while exploiting them in a real situation at the same time.

1. The robot has to localize a green cylinder in front of it, in terms of 3D cartesian coordinates. The robot stands up and remains in the same position, while the cylinder goes away along the x-axis direction at each trial. The error between the cylinder real coordinates and the evaluated ones is plotted as function of the distance between the middle point of the eyes-axes, evaluated with Eq. 18, and the cylinder center (see next section for a more complete explanation).
2. The robot has to evaluate the ball's radius while an varying occlusion hides the object. The robot stands up in front of the ball, which remains in the same position during all the trials. The ball is occluded by a cube placed in front of it more and more at each trial. Both the ball and the cube have been placed over a table, in front of the robot.
3. The robot has to localize a green ball in front of it, in terms of 3D cartesian coordinates. The robot stands up and remains in the same position, while the ball changes its coordinates at each trial. The error between the ball's real coordinates and the evaluated ones is evaluated as in the previous scenario. In the next section e will reconsider it for a more complete explanation.

The object position in the scenario nos. *1* and *3* changes at each trial.

7 Results and Discussion

In the scenario nos. *1* and *3* the error between the real and the evaluated cylinder's and ball's position is determined, while in the scenario *3* the error between the real and evaluated ball's radius is calculated. The position error is evaluated as in Eq. 24. Since the algorithm is innovative in the ill-conditioning of the original approach (B2AC) in the particular case in which the ellipse lies very close to the optimal equation curve, and since we already performed a preliminary study about it in [13], we take advantage of our previous results in this work, adapting them for our new algorithm (LCSE).

7.1 Error Propagation Evaluation

We evaluated the error propagation for the position detection as follows.

The absolute errors have been evaluated as in Eq. 21. All of the terms are measured in *simulator measure unit (SMU)*. The err_{pixel} is the absolute error relative to the value of one square pixel. In order to evaluate it we referred to the known ball's radius. By knowing it (as a fixed value, i.e. 0.17 SMU) and by evaluating it at each measure we can estimate the value of a square pixel in SMU (this is the image resolution at the object distance) as the ratio between the known radius and the one estimated with each of the three algorithms considered (i.e. Hough Transform, B2AC, and LCSE) (see Eq. 22).

Therefore, according with the error propagation theory, the error of a square pixel is obtained as in Eq. 23.

The errors of the encoders can be considered negligible within the simulator. Since there is no documentation on the encoders' resolution within the simulator, we considered the accuracy of their information approximated to their last digit, which is the forth one (therefore negligible). Finally the errors due robot's lengths need to be considered. Again, there is no information about the error the lengths of the robot's parts have been expressed with. Therefore, in order to fix their accuracy we analyzed the simulator's source code. So far, we found that the lengths of the robot's parts were expressed with the second digit of approximation. Hence, we approximated them as 0.01 SMU.

7.2 Scenarios' Evaluation

7.2.1 Scenario No. 1

As a first results the scenario no. *1* is analyzed. The object's position error as function of the distance while considering the perspective effect null is presented in Fig. 14. Here, it is possible seeing that, with exception for the quadratic error within the range [2.15–2.35], the Hough Transform gives rise to the highest error.

The B2AC algorithm is the most precise in terms of quadratic error, within the ranges [1.2–1.9] and [2.7–3.4]. However, it presents several discontinuities, and a total non-linear characteristic emerges, even following the Hough Transform approach's error (but keeping almost lowest). The LCSE seems to be not the lowest error prone, but it has a very regular characteristic of the function of the distance. By increasing the distance it fits the B2AC error curve well, while keeping little bit higher.

7.2.2 Scenario No. 2

The experiment of the scenario *2* shows a great linearity between the occlusion of the ball and the error on its radius evaluation (see. Fig. 15). Here, the Hough Transform gets better results within the range [5–20%] of occlusion (defined as in

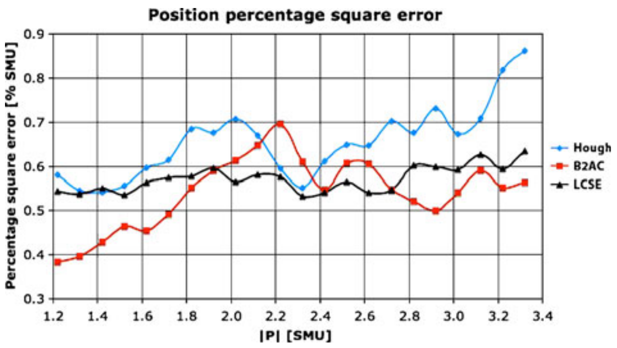


Fig. 14 Cylinder's position error as function of the distance while considering the perspective effect negligible

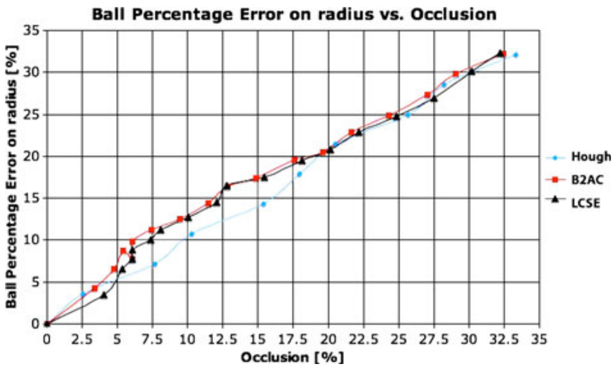


Fig. 15 Ball percentage error on radius, in % of the radius value

Eq. 25, where P_r is the residual number of pixels, and P_t is the total number of target object pixels, determined with no occlusion), then almost superimposing with the other two approaches after the 20% of occlusion. The characteristic is quite linear for all the techniques adopted, with the exception of the cited range, in terms of a slight decrease from the linear ideal line for the Hough Transform and a slight increment for both ellipse detection approaches.

Subsequently, the error introduced by spatial perspective is mapped as a function of the object's distance from the eyes axis midpoint. We isolate the perspective error by comparing the absolute error obtained within the tests in the scenario no. *1* and in the scenario no. *2*, as absolute errors. It is worth noting that in order to compare these errors, the cylinder and and the ball we used have the same radius (0.17 SMU) within the trials. The percentage perspective error has been evaluated as the ratio between the absolute perspective error and the module of the distance between the eyes axis midpoint and the object. This is plotted in Fig. 15. Here, it is possible to see that the two ellipse recognition techniques are more sensitive than the Hough Transform to the spatial perspective. This seems quite obvious, due to the fact that the latter looks for circles, and the first two for yet deformed circles, i.e. ellipses. Nevertheless, the Hough Transform smoothes this artifact by bringing it back as a circle, before evaluating the centroid and radius parameters. The B2AC and the LCSE algorithms do not.

7.2.3 Scenario No. 3

Finally, the scenario no. *3* is discussed. In spite of the fact that the ellipse detection approaches give rise to a bigger spatial perspective error than the Hough Transform, the precision given within the overall system is superior than the one obtained with the Hough Transform. In fact, despite the amount of the perspective error value, the major precision guaranteed by an ellipse detection rather than a circle one brings about to a more exact final result in determining the spatial position of the ball. In Fig. 16 this is showed. We did not filter the results, in order to keep them as natural as possible. By acting in this way, the noise affects the trend of the curves most. Here, the B2AC's and the LCSE's trend lines appear superimpose, so that it is not possible distinguishing them from each other. However, the Hough Transform's trend line shows of this technique is the most error prone for balls' spatial position detection in image processing. In fact, it is always higher than the other two.

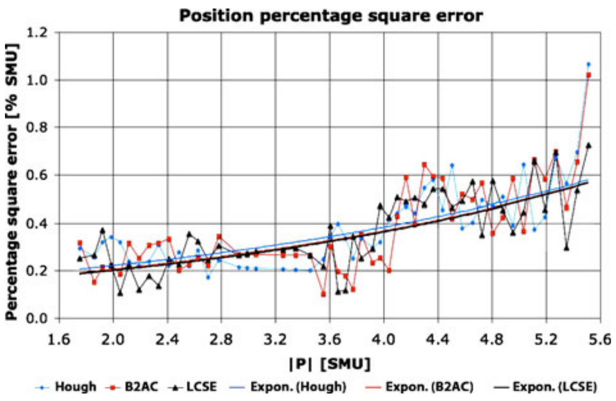


Fig. 16 Percentage square error, measured in % of the simulator measure unit

7.3 Some Aspects of the Hough Transform

Finally, we would like to pay attention to an important aspect of the Hough Transform.

It is worth mentioning, that the Hough transform depends on some parameters in order to be well set. Of most importance is the *accumulator threshold at the center detection stage* (*ATCDS*). The smaller its value is, the more false circles may be detected, but the higher it is, the less circles may be detected. We tested that the smaller the *ATCDS* is, the more instability is produced on the Hough computation. In the first case, an *ATCDS* lower setting causes that other curvatures, e.g. artifacts on the ball's border caused for instance by inaccuracies of the color filtering, may be detected as additional objects. Thee inaccuracies, in fact, can be interpreted as small circles by the Hough transform, giving rise to wrong results. However, setting *ATCDS* too high can cause the opposite problem. In our case there was only one circular object within the image (i.e. the ball), but wrong setting values (e.g. *ATCDS* too high) were sufficient to not detect it. This was true even if the detected ball was the only color blob after the erosion of the image, and even if it was substantially big too not be negligible (i.e. not to be misinterpreted as a color artifact). This means that one has to find the right value in every condition, in terms of the best compromise between sensibility (intended as the capability of detecting all the possible circles in the image) on one hand, and the stability (intended as the accuracy of the algorithm of not detecting false circles) on the other hand. Therefore, we looked for the biggest value that is able to perform all the experiments without avoiding the detection of the circles and maintaining the best possible stability. In our experiments we set $ATCDS = 2$. In Table 4 there are some *ATCDS* values: each one represents

Table 4 Maximum value for the accumulator threshold for getting stability in our experiments

ATCDS	3	2.6	2.3	2.1	2	2
Occlusion (%)	5	10	15	20	25	30
ATCDS	3.4	3	2.6	2.4	2.1	2
Distance (SMU)	1.2	1.6	2	2.4	2.8	3.2

the maximum worth able to perform the experiments at some fixed occlusion and distance values.

However, both B2AC and LCSE algorithms do not present a similar drawback, permitting them to be used in any situation without any previous setting. This can be considered a great advantage, since they do not require any *a priori* information of the scene to be analyzed. This is twofold, because allows not only to build a robust and *scene–independent* technique, but also it fits with the concept of cognitive robotics perfectly.

8 Conclusions

In this work we propose a 3D stereo tracker featuring the first implementation of a new pattern recognition algorithm for the least square of ellipses that improves the original one [6]. With our tracker we are able to localize an object with the Robot's binocular vision, and subsequently to triangulate these information in conjunction with those of the robot's head encoders in order to determine the position of the object's centroid in the environment, in terms of 3D coordinates. We tested our 3D stereo tracking algorithm for localizing of some objects in spatial coordinates. Moreover, we compared our pattern recognition technique against the Hough Transform, and the original algorithm, the B2AC, in terms of localization precision and failure performances as function of the distance of the target, and in presence of induced artifacts (such as the ball occlusion by another object). We used the a state of art robotics platform, the iCub, together with its simulator in order to test our tracker at best. Then, we will make our code freely available within the iCub repository.

Acknowledgements We thank Dr. Andrea Cini for his contribution in writing the DH matrix for the robot forward kinematics.

This work was supported by the European Commission, Project IST-004370 RobotCub and FP7-231640 Handle, and by the Portuguese Government—Fundação para a Ciência e Tecnologia (ISR/IST pluriannual funding) through the PIDDAC program funds and through project BIO-LOOK, PTDC / EEA-ACR / 71032 / 2006.

Appendix

Error Propagation

We evaluated the error propagation for the position detection as follows. The absolute errors have been evaluated as:

$$
\begin{aligned}
err_{x-\mathrm{axis}} &= \sqrt{err^2_{\mathrm{pixel}} + err^2_{\mathrm{encoders}} + err^2_{\mathrm{misure-iCub}}} \\
err_{y-\mathrm{axis}} &= \sqrt{err^2_{\mathrm{pixel}} + err^2_{\mathrm{encoders}} + err^2_{\mathrm{misure-iCub}}} \\
err_{z-\mathrm{axis}} &= \sqrt{err^2_{\mathrm{pixel}} + err^2_{\mathrm{encoders}} + err^2_{\mathrm{misure-iCub}}} \\
err &= \sqrt{err^2_{x-\mathrm{axis}} + err^2_{y-\mathrm{axis}} + err^2_{z-\mathrm{axis}}} \qquad (21)
\end{aligned}
$$

where each of term is measured in *simulator measure unit (SMU)*. The err_{pixel} is the absolute error relative to the value of one square pixel. In order to evaluate it we referred to the known ball's radius. By knowing it (as a fixed value, i.e. 0.17 SMU) and by evaluating it at each measure we can estimate the value of a square pixel in SMU (this is the image resolution at the object distance) as the ratio between the known radius and the one estimated with each of the three algorithms considered (i.e. Hough transform, B2AC, and LCSE):

$$err_{\text{pixel}-x} = err_{\text{pixel}-y} = \frac{0.17}{radius_{\text{eval}}} \tag{22}$$

Therefore, according with the error propagation theory, the error of a square pixel is:

$$\begin{aligned} err_{\text{pixel}} &= \sqrt{err^2_{\text{pixel}-x} + err^2_{\text{pixel}-y}} \\ &= \sqrt{2} \cdot err_{\text{pixel}-x} \end{aligned} \tag{23}$$

The position error during the three-scenario experiments is evaluated as in Eq. 24:

$$\begin{aligned} rms_{\text{err}} &= \sum_{i=1}^{3} \sqrt{(p_{\text{real}_i} - p_{\text{eval}_i})^2} \\ &= \sum_{i=1}^{3} \left\{ \sqrt{(x_{\text{real}} - x_{\text{eval}})^2 + (y_{\text{real}} - y_{\text{eval}})^2} + (z_{\text{real}} - z_{\text{eval}})^2 \right\} \end{aligned} \tag{24}$$

where the $(x_{\text{real}}, y_{\text{real}}, z_{\text{real}})$ and the $(x_{\text{eval}}, y_{\text{eval}}, z_{\text{eval}})$ are the real and evaluated 3D coordinates of the ball's center, respectively. Indeed, this can be considered as the *root-mean square error*. These values are relative to the simulator's reference system, which has the origin in the center of the robot's floor base is located where. The reference system is orthonormal, and its orientation is as follows:

- x axis: parallel to the floor plane; it increases with direction orthogonal to the eyes' axis and going away in front of the robot;
- y axis: parallel to the floor plane; it increases with direction parallel to the eyes' axis and going away to the left of the robot;
- z axis orthogonal to the floor plane; it increases going away along the height.

The object occlusion is evaluated as the difference between the total number of the ball's pixel and the number of pixel detected. This is showed in Eq. 25:

$$occlusion\ [\%] = P_c \cdot 100 \tag{25}$$

where $P_c = (P_t - P_r)/P_t$, with P_t number of ball's pixel without occlusion, and P_r the number of pixels receipted being the ball partially occluded.

Ellipse Equations

Once obtained the parameters of the ellipse that fits the ball at best, we evaluated the center of the ball automatically. In fact, once we solve the eigenvector problem [19], we obtain the vector in Eq. 26.

$$\mathbf{a} = [a, b, c, d, e, f]^T \tag{26}$$

Therefore, it is possible to estimate the following characteristic parameters of the ellipse as follows, with a mathematical transformation:

$$
\begin{aligned}
x_c &= \frac{c \cdot \frac{d}{2}}{\left(\frac{b}{2}\right)^2 - a \cdot c} \\
y_c &= \frac{a \cdot \frac{a}{2}}{\left(\frac{b}{2}\right)^2 - a \cdot c} \\
\psi &= \frac{1}{2} \cdot \tan^{-1}\left(\frac{b}{c-a}\right) \\
a_{\frac{1}{2}} &= \left(\frac{2\left(a \cdot \left(\frac{e}{2}\right)^2 + c \cdot \left(\frac{d}{2}\right)^2 + f \cdot \left(\frac{b}{2}\right)^2 - \frac{b \cdot d \cdot e}{4} - acf\right)}{\left(\left(\frac{b}{2}\right)^2 - a \cdot c\right)\left((c-a)\left(1 + \frac{4 \cdot \left(\frac{b}{2}\right)^2}{(a \cdot c)^2}\right)^{\frac{1}{2}} - (c+a)\right)}\right)^{\frac{1}{2}} \\
b_{\frac{1}{2}} &= \left(\frac{2\left(a \cdot \left(\frac{e}{2}\right)^2 + c \cdot \left(\frac{d}{2}\right)^2 + f \cdot \left(\frac{b}{2}\right)^2 - \frac{b \cdot d \cdot e}{4} - acf\right)}{\left(\left(\frac{b}{2}\right)^2 - a \cdot c\right)\left((a-c)\left(1 + \frac{4 \cdot \left(\frac{b}{2}\right)^2}{(a \cdot c)^2}\right)^{\frac{1}{2}} - (c+a)\right)}\right)^{\frac{1}{2}} \\
e &= \frac{a \cdot (1 - e^2)}{e} \\
p &= \frac{\frac{b}{2}^2}{a^2 - \frac{b}{2}^{2\frac{1}{2}}} \\
A &= \pi \cdot a_{\frac{1}{2}} \cdot b_{\frac{1}{2}} \\
C &= \pi \cdot \left(3\left(a + \frac{b}{2}^2\right) - \left(3a + \frac{b}{2}^2\right)\left(a + 3\frac{b}{2}^2\right)^{\frac{1}{2}}\right)
\end{aligned}
\tag{27}
$$

where x_c and y_c are the coordinates of the center, ψ is the rotation angle, $a_{\frac{1}{2}}$ and $b_{\frac{1}{2}}$ are the major and minor semi-axis length, respectively, and e is the *eccentricity*. p is the *focal parameter*. ψ is measured in radians, $a_{\frac{1}{2}}$, $b_{\frac{1}{2}}$ in pixels. A and C are the area and the circumference, respectively. The unit of measure for each parameter is: (x_c,y_c) is with respect to an orthogonal reference system with the origin in the image's top left corner, ψ is measured in radians, $a_{\frac{1}{2}}$, $b_{\frac{1}{2}}$, and C are measured in pixels, when A is measured in square pixels.

References

1. Cheng, Y., Lee, S.: A new method for quadrative curve detection using k-ransac with accelereration techniques. Pattern Recogn. **28**(5), 663–682 (1995)
2. Dave', R.N., Bhaswan, K.: Adaptive fuzzy c-shells clustering and detection of ellipses. IEEE Trans. Neural Netw. **3**, 643–662 (1992)
3. Deniz, O., Castrillon, M., Lorenzo, J., Guerra, C., Hernandez, D., Hernandez, M.: Casimiro: A robot head for human-computer interaction. In: Proceedings of the 11th IEEE International

Workshop on Robot and Human Interactive Communication, pp. 319–324. ISBN: 0-7803-7545-9 (2002)
4. DeSouza, G.N., Kak, A.C.: Vision for mobile robot navigation: A survey. IEEE Trans. Pattern Anal. Mach. Intell. **24**, 237–267 (2002)
5. Dhome, M., Lapreste, J.T., Rives, G., Richetin, M.: Spatial Localisation of Modelled Objects of Revolution in Monocular Perspective Vision, pp. 475–485 (1990)
6. Fitzgibbon, A., Pilu, M., Fisher, R.: Direct least square fitting of ellipses. IEEE Trans. Pattern Anal. Mach. Intell. **21**, 476–480 (1999)
7. Forsyth, D., Mundy, J., Zisserman, A., Coelho, C., Heller, A., Rothwell, C.: Invariant descriptors for 3-d object recognition and pose. IEEE Trans. Pattern Anal. Mach. Intell. **13**(10), 971–991 (1991)
8. Gander, W., Golub, G., Strebel, R.: Fitting of Circles and Ellipses Least Squares Solution. Technical Report tr-217, Institut für Wissenschaftliches Rechen, ETH, Zurich, Switzerland (1994)
9. Gander, W., Strebel, G.G.R.: Least-square fitting of circles and ellipses. BIT **34**, 558–578 (1994)
10. Gath, I., Hoory, D.: Fuzzy clustering of elliptic ring-shaped clusters. Pattern Recogn. Lett. **16**, 727–741 (1995)
11. Gauss, C.F.: Theory of the Motion of the Heavenly Bodies Moving About the Sun in Conic Sections (Theoria Motus Corporum Coelestium in Sectionibus Conicis Solem Ambientum). Translation by C.H. Davis. Dover, New York (1963). First published in 1809
12. Greggio, N., Bernardino, A., Laschi, C., Santos-Victor, J., Dario, P.: An algorithm for the least square-fitting of ellipses. In: 22th International Conference on Tools with Artificial Intelligence (ICTAI 2010). Arras, France (2010)
13. Greggio, N., Manfredi, L., Laschi, C., Dario, P., Carrozza, M.: Robotcub implementation of real-time least-square fitting of ellipses. In: IEEE-RAS International Conference on Humanoid Robots (HUMANOIDS-08)—Daejeon (Korea), December 1–3 (2008)
14. Greggio, N., Silvestri, G., Menegatti, E., Pagello, E.: Simulation of small humanoid robots for soccer domain. J. Franklin Inst. Eng. Appl. Math. **346**(5), 500–519 (2009)
15. Hough, P.: Method and means for recognizing complex patterns. US Patent No. 3 069 654 (Dec. 18, 1962)
16. Kanatani, K.: Statistical bias of conic fitting and renormalization. IEEE Trans. Patt. Anal. Mach. Intell. **16**, 320–326 (1994)
17. Kwolek, B.: Real-time head tracker using color, stereovision and ellipse fitting in a particle filter. Informatica **15**(2), 219–230 (2004)
18. Leavers, V.F.: Shape Detection in Computer Vision Using the Hough Transform. Springer, Berlin (1992)
19. Maini, E.S.: Enhanced direct least square fitting of ellipses. Int. J. Pattern Recogn. Artif. Intell. **20**(6), 939–954 (2006)
20. Mariottini, G.L., Oriolo, G., Prattichizzo, D.: Image-based visual servoing for nonholonomic mobile robots using epipolar geometry. IEEE Trans. Robot. **23**(1), 87–100 (2007)
21. Metta, G., Sandini, G., Vernon, D., Caldwell, D., Tsagarakis, N., Beira, R., Santos-Victor, J., Ijspeert, A., Righetti, L., Cappiello, G., Stellin, G., Becchi, F.: The robotcub project—an open framework for research in embodied cognition. In: Humanoids Workshop, IEEE–RAS International Conference on Humanoid Robots (2005)
22. Nelson, B.J., Khosla, P.K.: The resolvability ellipsoid for visual servoing. In: Proc. of the 1994 Conf. on Computer Vision and Pattern Recognition (CVPR94) (1994)
23. Paromtchik, I.E.: Optical guidance method for robots capable of vision and communication. Robot. Auton. Syst. **54**(6), 461–471 (2006)
24. Paton, K.: Conic sections in chromosome analysis. Pattern Recogn. **2**, 39–51 (1970)
25. Perdereau, V., Passi, C., Drouin, M.: Real-time control of redundant robotic manipulators for mobile obstacle avoidance. Robot. Auton. Syst. **41**(1), 41–59 (2002)
26. RobotCub: The Robotcub Project—European Commission fp6 Project ist-004370. URL: http://www.robotcub.org/ (2004–2009)
27. Rosin, P.L.: Ellipse fitting by accumulating five-point fits. Pattern Recogn. Lett. **14**, 661–699 (1993)
28. Rosin, P.L., West, G.A.: Non parametric segmentation of curves into various representations. IEEE Trans. Pattern Anal. Mach. Intell. **17**, 140–153 (1995)
29. Sabatini, A., Genovese, V., Maini, E.: Low-cost vision-based 2d localization systems for application in rehabilitation robotics. In: IEEE Proc. IROS, pp. 1355–1360 (2002)

30. Shim, H., Kwon, D., Yun, I., Lee, S.: Robust segmentation of cerebral arterial segments by a sequential monte carlo method: particle filtering. Comput. Methods Programs Biomed. **84**(2–3), 135–145 (2006)
31. Song, X., Seneviratne, L.D., Althoefer, K.: A kalman filter-integrated optical flow method for velocity sensing of mobile robots. IEEE/ASME Trans. Mechatron. **PP**(99), 1–13 (2010)
32. Stellin, G., Cappiello, G., Roccella, S., Carrozza, M.C., Dario, P., Metta, G., Sandini, G., Becchi, F.: Preliminary design of an anthropomorphic dexterous hand for a 2-years-old humanoid: towards cognition. In: IEEE BioRob, Pisa, February, pp. 20–22 (2006)
33. Stentz, A.: Robotic technologies for outdoor industrial vehicles. Unmanned ground vehicle technology. In: Conference No. 3, Orlando FL, ETATS-UNIS (16/04/2001), vol. 4364, pp. 192–199 (2001)
34. Teutsch, C., Berndt, D., Trostmann, E., Weber, M.: Real-time detection of elliptic shapes for automated object. Machine vision applications in industrial inspection XIV. Edited by Fabrice Meriaudeau and Kurt S. Niel. Proc. SPIE **6070**, 171–179 (2006)
35. Tikhanoff, V., Fitzpatrick, P., Nori, F., Natale, L., Metta, G., Cangelosi, A.: The iCub humanoid robot simulator. In: International Conference on Intelligent RObots and Systems IROS. Nice, France (2008)
36. Vincze, M.: Robust tracking of ellipses at frame rate. Pattern Recogn. **34**, 487–498 (2001)
37. Werman, M., Geyzel, G.: Fitting a second degree curve in the presence of error. IEEE Trans. Pattern Anal. Mach. Intell. **17**(2), 207–211 (1995)
38. Wu, C.J., Tsai, W.H.: Location estimation for indoor autonomous vehicle navigation by omni-directional vision using circular landmarks on ceilings. Robot. Auton. Syst. **57**(5), 546–555 (2009)
39. Wu, W.Y., Wang, M.J.J.: Elliptical object detection by using its geometric properties. Pattern Recogn. **26**, 1499–1509 (1993)
40. Yin, R.K.K., Tam, P.K.S., Leung, N.K.: Modification of Hough transform for circles and ellipses detection using 2-d array. Pattern Recogn. **25**(9), 1007–1022 (1992)
41. Yoo, J., Sethi, I.: An ellipse detection method from the polar and pole definition of conics. Pattern Recogn. **26**(2), 307–315 (1993)
42. Yuen, H.K., Illingworth, J., Kittler, J.: Detecting partially occluded ellipses using the Hough transform. Image Vis. Comput. **7**(1), 31–37 (1989)
43. Zhang, Z.: Parameter estimation techniques: a tutorial with application to conic fitting. Image Vis. Comput. **15**, 59–76 (1997)