Real-Time Least-Square Fitting of Ellipses Applied to the RobotCub Platform

Nicola Greggio[1], Luigi Manfredi[1], Cecilia Laschi[2], Paolo Dario[1], and Maria Chiara Carrozza[1]

[1] ARTS Lab - Scuola Superiore S.Anna, Polo S.Anna Valdera
Viale R. Piaggio, 34 - 56025 Pontedera, Italy
[2] IMT Institute of Advanced Study Via San Micheletto, 3, 55100 Lucca, Italy

nicola.greggio@ieee.org

Abstract. This paper presents the first implementation of a new algorithm for pattern recognition in machine vision developed in our laboratory. This algorithm has been previously presented only theoretically, without practical use. In this work we applied it to the RobotCub humanoid robotics platform simulator. We used it as a base for a circular object localization within the 3D surrounding space. The algorithm is a robust and direct method for the least-square fitting of ellipses to scattered data. RobotCub is an open source platform, born to study the development of neuro-scientific and cognitive skills in human beings, especially in children. 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 manipulation. In this work we present an improvement of the RobotCub project in terms of machine vision software, by implementing the method of the least-square fitting of ellipses of Maini (EDFE), previous developed in our laboratory, in a robotics context. Moreover, we compared its performance with the Hough Tranform, and others least-square ellipse fittings techniques. We used our system to detect spherical objects by applying it to the simulated RobotCub platform. We performed several tests to prove the robustness of the algorithm within the overall system. Finally we present our results.

1 Introduction

The impressive advance of research and development in robotics and autonomous systems over the past few years has led to the development of robotic platforms of increasing motor, perceptual, and cognitive capabilities. These achievements are opening the way for new application opportunities that will require these systems to interact with other robots or nontechnical users during extended periods of time. The final goal is creating autonomous machines that learn how to execute complex tasks and improve their performance throughout their lifetime.

Motivated by this objective the RobotCub (ROBotic Open-Architecture Technology for Cognition, Understanding and Behavior) project has been developed.

This is a research initiative dedicated to the realization of embodied cognitive systems [1], [2].

One of the basic assumptions of this project is that manipulation plays a key role in the development of cognitive capability. A ball is a perfect example of a very common and simple to manipulate toy every children uses. Fitting a ball may be very problematic in image processing because of the light, likely curvatures of the camera's lens, etc. These phenomena cause a deformation on the ball, making it look more like an ellipse. In addition, a ball is a particular case of an ellipse, i.e. when the ellipse has its axes of the same length.

Two main approaches can be considered for circle detection.

The first one is to use the Hough Transform [3], [4]. This approach can be divided into several steps. Since spatial perspective alters the perceived objects, the first step is calibrating the camera (in terms of reconstructing the original image proportions by computing the inverse perspective and the camera's lens distortion). By doing this, a ball previously mapped to an ellipse returns to be drawn as a circle. Subsequently, a pattern recognition algorithm, such as a simple color detection, can be applied and then the Hough circle transform can be applied in order to estimate all the ball's characteristics (e.g. center of gravity position - COG - within the 2D space and dimension). However, this approach can be complex to be implemented, and even elevate resource consumption. First, it requires the camera calibration. Moreover, the Hugh transform needs to be set well, in terms of the accumulator threshold at the center detection stage parameter. We will give a full explanation of our experiments later, in section 5. Finally, all these mathematical procedures require the implementation of complex and therefore error-prone functions, likely also resulting in an excessive computational burden.

The second one is to use ellipse pattern recognition algorithms. We prefer processing a ball thinking of it as it were an ellipse, in order to overcome these distortion problems. Circles in man-made scenes are almost always distorted when projected onto the camera image plane, therefore generating ellipses. The latter provide a useful representation of parts of the image since 1) they are more convenient to manipulate than the corresponding sequences of straight lines needed to represent the curve, and 2) their detection is reasonably simple and reliable. Thus they are often used by computer vision systems for model matching [5], [6]. There are many techniques for ellipse detection. Most of thew work in real-time (even if depending on the image size) [7], [8].

In this paper we implemented for the first time in a real context the *Enhance Direct Fitting of Ellipses* (EDFE) technique [8] for the least-square fitting of ellipses, previously developed by our group. We implemented this as a continuation of a work started as a pure mathematical context in our team by Maini *et Al* [8]. In their first version, the authors implemented and tested their work only with theoretical simulations, in *Matlab*. We implemented and tested these techniques under a robotics context for the first time. We tested our new algorithm and the previous related existing techniques ([4], [7]) under the same experimental conditions. First, we would check the performance differences among these methods,

intended as produced error. It is worth note we are not interested in the absolute error of each procedure (yet evaluated for each method in [4], [7], and [8]); nonetheless we are interested 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 [4], [7], and [8], but their usage in practical applications and scenarios instead, such as finding the position of a target object within the 3D surrounding space. We used the simulation of a state of art robotics platform, the RobotCub, in order to test it at best before doing this with the real platform. So far, we not only improved a growing open project by adding new capabilities to the robot, but also made our program open-sorce, available to whose need it as tool for their personal research or for improving our work as well. In fact, RobotCub is a completely open project, and by adhering to it we made our work fully available to everybody [9].

This paper is organized as follows. In section 2 we will describe the RobotCub robotics platform, in terms of its mechanics and the simulator we used. Then, in section 3 we will discuss the state of the art problem of the least-square fitting of ellipses. Furthermore, in section 4 we will briefly explore our vision algorithms. In sec. 5 we will describe our experimental set-up. In section 6 we will discuss our results. Finally, in section 7 we will conclude our work and explain our projects as future research.

2 The iCub Robotics Platform

In this section the iCub robotics platforms is described. It is one of the most advanced state of the art robots. It is dimensionally inspired to be a two-year-old human being.

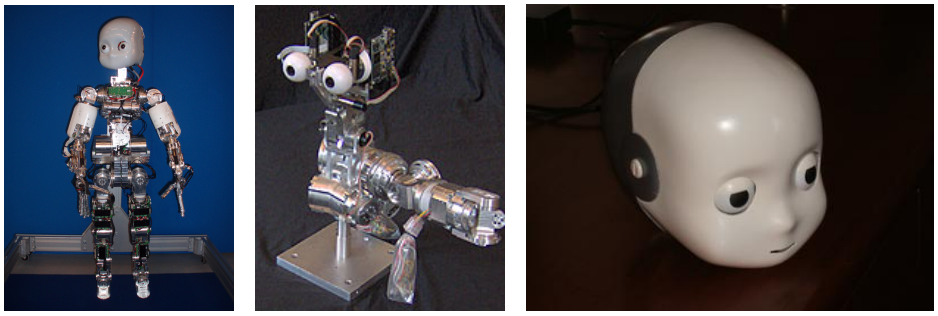
2.1 The iCub Mechanics

The robot is composed of 53 degrees of freedom (DOFs). Most of them are directly actuated, such as the shoulders [10], others are under-actuated, such as the hands [2]. 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.

In Fig. 1 is shown the iCub in its final configuration [9].

In vision the head is of particular interests. The iCub's head is completely based on the Faulhaber motors. 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 rigth), 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 [10].



(a) The whole robot (b) The iCub's head without cover (c) The complete iCub's head

Fig. 1. The RobotCub. On the left image the whole robot is depicted (a); on the central image the head without the cover is shown (b) while in the right image the cover is shown.

An exaustive explanation about a kinematic and a dynamic analysis for the upper body structure can be found in [11].

2.2 The ODE iCub Simulator

There are many reasons for which it is important to test new algorithms within a simulator in order to debug them safely [12]. 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 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.

Tikhanoff *et al.* developed a completely open source simulator for the iCub [13], [14], based entirely on the *ODE* (Open Dynamic Engine).

We use this simulator in order to test our algorithms.

3 Least Square of Ellipses: The State of the Art

A new interesting LS technique, the Enhanced Least-Square Fitting of Ellipses (EDFE), has been developed recently by our work team by Maini *et Al*, and it was proposed in [15], [8]. This is a LS procedure that improves the work described

in [7]. In this work, Fitzgibbon *et al.* developed a direct computational method (i.e. B2AC) based on the algebraic distance with a quadratic constrain. Our new approach overcomes the state of the art by solving the problems of numerical instability that can produce completely wrong results, such as infinite or complex solutions, not reported in the original work [7].

Essentially, it is a upgrade of the Fitzgibbon's original work, aimed by the idea of making it *1.* faster (in order to use it in real-timeapplications) and *2.* more precise (it works better on noisy data, loosing precision with better data). The first result has been obtained by using an *affine transformation*, that recenters all the points belonging to the ellipse to be fitted within a square of side length equal to 2 and centered at the origin of the referring cartesian plane. This represents an ellipse similar to the original but normalized within this square. Clearly, the ellipses parameters have to be denormalized after having solved the fitting problem. This overcomes the problem of having huge numbers representing the ellipse's points coordinates, due to the fact the frame grabbers and cameras have ever bigger resolution, therefore making the fitting faster. The second result has been solved by resampling the ellipse data with perturbations. Specifically, if the data points lie exactly on the ellipse the eigenvalue corresponding to the optimal solution is zero thus the original Fitzgibbon's algorithm [7] does not lead to any solution. Moreover, this happens even if the data points are close to the ideal ellipse therefore B2AC performs poorly both when noise is absent and low [15]. The problem has been solved by slightly perturbing the original data by adding a known Gaussian noise. Therefore, the fitting is performed.

For a more precise analysis of the method one can refer to [15], and [8].

However, the author describes his technique only as a mathematical procedure, without inserting it within an actual robotics context. In fact, the authors tested its characteristics only in *Matlab* simulation [15], [8].

We implemented this *state-of-the-art* pattern recognition algorithm and we tested it in a real robotics project for the firs time.

4 ExPerCub: The Robot Controlling Tool

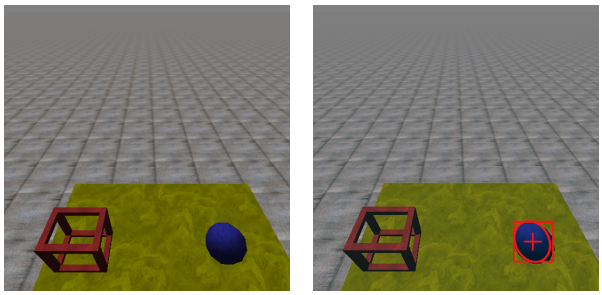
We implemented this algorithm as a tool for our research objective. It is comprehensive of a more complete project. We have to fulfill the deliverable 3.5 of the RobotCub project [9], relative to the implementation of the sensorimotor coordination for reaching and grasping.

4.1 Our Vision Algorithm

The vision module receives the images from the two cameras mounted on the iCub head. It is responsible for processing these images 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 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 a simple but efficient image processing algorithm. We identify the ball by means of a color filter. The object detection is performed by using of a sample color recognition procedure.



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

Fig. 2. The input image, as seen by the robot 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 EDFE (b).

For the identification of the blob corresponding to the ball, we use a *connected components* labeling algorithm. We assume the largest blob is the ball, so we look for the blob with the largest area. Subsequently, we proceeded by applying our LS technique [8] to the found blob, in order to detect all the parameters of the curve that describes the boundary of the blob. We used the iCub ODE simulator present in the iCub repository. Moreover, 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. 5 an example of the ball detection algorithm output is shown. In Fig. 2(a) the input to the left camera is presented, i.e. the experimental scenario, while in Fig. 2(b) output of the algorithm is presented. These images are the input image as seen by the robot with the egocentric view (Fig. 2(a)) and the same image with the superimposition of an ellipse, drawn by using the characteristic parameters obtained by computing the EDFE (Fig. 2(b)).

In addition, we implemented a tracking algorithm that directly commands the head of the robot, in order to be able to reconstruct the target object position (in terms of its centroid) by triangulating the information of the neck and head encoders.

In Fig. 3 a screenshot is depicted, that shows an operative situation in which the simulator tracked the ball. The iCub program we implemented is able to localize 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 [3].

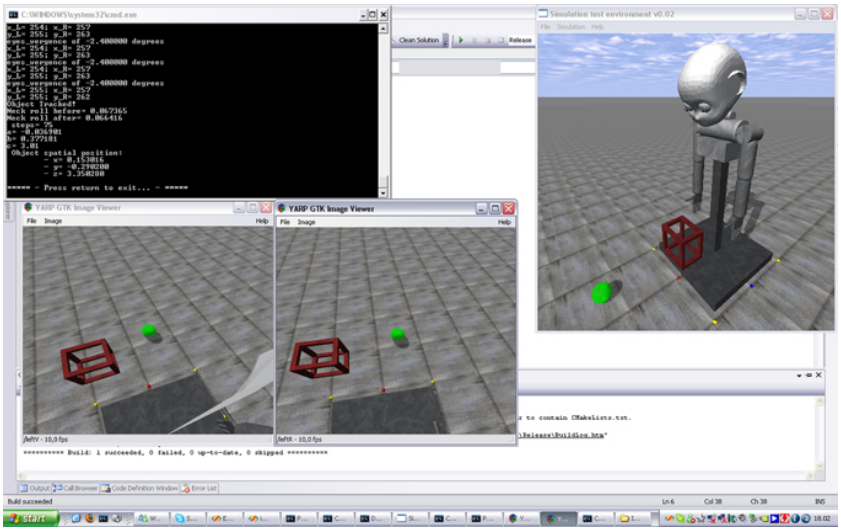


Fig. 3. A screenshot depicting the moment in which the simulated robot tracked the position of the ball in the 3D surrounding environment. Therefore, our program uses the encoders information to triangulate the position of the centroid of the object within the simulated space.

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

5 Experimental Set-Up

We performed three types of experiments, in order to validate the EDFE pattern recognition algorithm [8] compared with the Hough transform and the least square ellipse fitting algorithm, B2AC [7]. 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 uncalibrated 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.

[3] 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.

It is worth noticing we are not interested in the absolute error of each procedure (yet evaluated for each method in [4], [7], and [8]); nonetheless we are interested 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 already done by the authors, but their usage in practical applications and scenarios instead. The final error is a combination of all the previous imprecisions. In the next section we will analyze the error propagation process, and we will quantize it in our specific case.

5.1 Scenarios

At each trial the Hough transform, the B2AC, and the EDFE algorithms are used in order to evaluate the ball's center of gravity (COG) 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.

We made the experiment in the scenario no. 1 by using a cylinder considered having a a null depth. (hence reducing the prospective effect). In this way we can test the algorithms by isolating the perspective error, while exploiting them in a real situation at the same time. The experiment in the scenario no. 2 is quite similar, but made using a ball instead of the cylinder, and letting it varying its position not only in the x-axis direction.

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 and the cylinder center. In the next section we will reconsider it for a more complete explanation.
2 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 plotted as function of the distance between the middle point of the eyes-axes and the ball's center. In the next section we will reconsider it for a more complete explanation.
3 The robot has to evaluate the ball's radius while an 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.

6 Results and Discussion

In the scenario *1* and *2* 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 follows:

$$rms_{err} = \sum_{i=1}^{3} \sqrt{(p_{real_i} - p_{eval_i})^2} = \sqrt{(x_{real} - x_{eval})^2 + (y_{real} - y_{eval})^2 + (z_{real} - z_{eval})^2} \tag{1}$$

where the $(x_{real}, y_{real}, z_{real})$ and the $(x_{eval}, y_{eval}, z_{eval})$ 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's is orthonormal, and its orientation is as follows:

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

6.1 Error Propagation Evaluation

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

The absolute errors have been evaluated as:

$$err_{i-th-axis} = \sqrt{err_{pixel}^2 + err_{encoders}^2 + err_{misure-iCub}^2}$$
$$err = \sqrt{err_{x-axis}^2 + err_{y-axis}^2 + err_{z-axis}^2} \tag{2}$$

each of them measured in *simulator measure unit (we use the abbreviation 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 EDFE):

$$err_{pixel-x} = err_{pixel-y} = 0.17/radius_{eval} \tag{3}$$

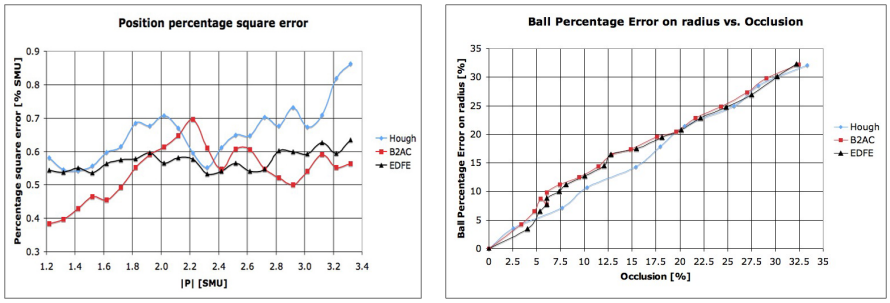
Therefore, according with the error propagation theory, the error of a square pixel is:

$$err_{pixel} = \sqrt{err_{pixel-x}^2 + err_{pixel-y}^2} = \sqrt{2} \cdot err_{pixel-x} \tag{4}$$

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, wihch 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.

6.2 Scenarios' Evaluation

As a first results the object's position error as function of the distance while considering the perspective effect null is presented in Fig. 4(a). Here, it is possible seeing that with exception for 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 EDFE 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.



(a) Cylinder's position error (b) Ball Percentage Error on radius

Fig. 4. The Cylinder's position error as function of the distance while considering the perspective effect null (a) and the Ball Percentage Error on radius, in % of the radius value (b).

The experiment of the scenario no. 3 shows a great linearity between the occlusion of the ball and the error on its radius evaluation. Fig. 4(b) illustrates the results of this experiment.

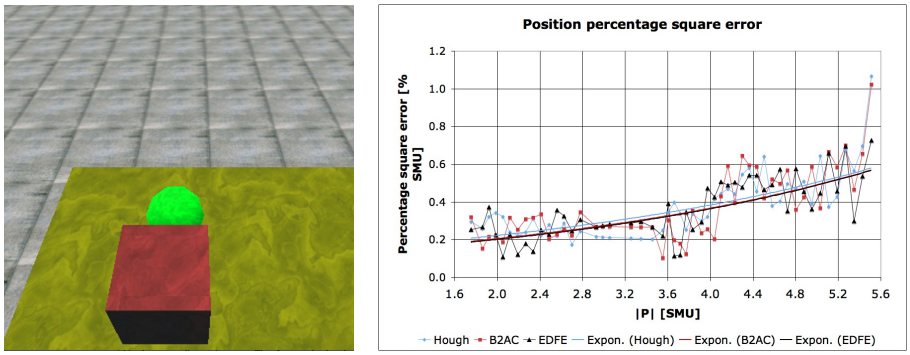
Here, the Hough Transform gets better results within the range [5 % - 20 %] of occlusion (defined as in equation 5, where P_r is the residual number of pixels, and P_t is the total number of target object pixels, determined with no occlusion), then almost superimposing with the other two approaches after he 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. Fig. 5(a) shows the target object partially hided by the occluding object.

$$occlusion[\%] = (P_t - P_r) \cdot 100/P_t \tag{5}$$

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. Therefore 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.

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 EDFE algorithms do not.

Finally, the scenario no. 2 is discussed. We keep this as the last discussion in order to show that, 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. 5(b) 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. Therefore, we inserted three trend lines (one for each technique, each of them with exponential characteristic) in order to evidence the most fruiting approach. Here, the B2AC's and the EDFE'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.



(a) The target object partially hidden by the occluding object.

(b) Percentage square error.

Fig. 5. The target object partially hidden by the occluding object (a) and the Percentage square error, measured in % of the simulator measure unit (b).

7 Conclusions

In this work we presented the first implementation of the EDFE ellipse square fitting algorithm, a technique developed by our team by Maini *et Al*, and we applied it to a humanoid robotics platform. The task we planned is the spatial localization of a circular object (i.e. a ball) placed within the surrounding environment. Therefore, we developed a computer vision algorithms in order to implement the EDFE technique for the firs time. Moreover, we implemented a tracking algorithm to localize the object with the Robot's binocular vision, and subsequently we triangulated these information in conjunction with those of the robot's head encoders to determine the position of the object's centroid in the environment, in terms of 3D coordinates with the reference system located at the base of the robot. Therefore, we performed some experiments in order to validate the precision of the overall system in presence of induced artifacts (such as the ball occlusion by another object) and as function of the distance of the target. We made the same experiments by using the Hough Transform, the B2AC, and the EDFE under the same assumption, in order to have the fairest examination as possible. We found that the B2AC and EDFE give rise to a more precise results in terms of the overall system than the Hough Transform.

7.1 Future Work

In the near future we plan to apply our techniques to the real RobotCub robotics platform, in order to compare and validate our results with the real robot, and not only with the ODE simulator.

Acknowledgements

This work has been partially supported by the EU Project RobotCub (European Commission FP6 Project IST-004370).

References

1. Sandini, G., Metta, G., Vernon, D.: Robotcub: An open research initiative in embodied cognition. Proceedings of the Third International Conference on Development and Learning (ICDL '04) (2004)
2. 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. IEEE BioRob, Pisa February (2006) 20–22
3. Yuen, H.K., Illingworth, J., Kittler, J.: Detecting partially occluded ellipses using the hough transform. Image Vision and Computing **7**(1) (1989) 31–37
4. Leavers, V.F.: Shape detection in computer vision using the hough transform. Springer-Verlag (1992)
5. Dhome, M., Lapreste, J.T., Rives, G., Richetin, M.: Spatial localisation of modelled objects of revolution in monocular perspective vision. (1990) 475–485
6. Forsyth, D., Mundy, J., Zisserman, A., Coelho, C., Heller, A., Rothwell, C.: Invariant descriptors for 3-d object recognition and pose. IEEE Trans. PAMI **13**(10) (1991) 971–991
7. Fitzgibbon, A., Pilu, M., Fisher, R.: Direct least square fitting of ellipses. IEEE Trans. PAMI **21** (1999) 476–480
8. Maini, E.S.: Enhanced direct least square fitting of ellipses. IJPRAI **20**(6) (2006) 939–954
9. RobotCub: Url: http://www.robotcub.org/
10. 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. Humanoids Workshop, IEEE –RAS International Conference on Humanoid Robots (December 2005)
11. Nava, N., Tikhanoff, V., Metta, G., Sandini, G.: Kinematic and dynamic simulations for the design of robocub upper-body structure. ESDA (2008)
12. Greggio, N., Silvestri, G., Antonello, S., Menegatti, E., Pagello, E.: A 3d model of a humanoid robot for the usarsim simulator. In: First Workshop on Humanoid Soccer Robots. Number ISBN 88-900426-2-1 (December 2006) 17–24
13. Tikhanoff, V., Fitzpatrick, P., Metta, G., Nori, F., Natale, L., Cangelosi, A.: An open-source simulator for cognitive robotics research: The prototype of the icub humanoid robot simulator. Performance Metrics for Intelligent Systems Workshop, PerMIS '08, National Institute of Standards and Technology (NIST) (Gaithersburg, MD, 20899 August 19-21, 2008 2008)
14. Tikhanoff, V., Fitzpatrick, P., Nori, F., Natale, L., Metta, G., Cangelosi, A.: The icub humanoid robot simulator. International Conference on Intelligent RObots and Systems IROS **Nice, France** (2008)
15. Maini, E.S.: Robust ellipse-specific fitting for real-time machine vision. BVAI (2005) 318–327