Guidance of Robot Arms using Depth Data from RGB-D Camera

G. J. García[1], P. Gil[1] , D. Llácer[2] and F. Torres[1]
[1]*Physics, Systems Engineering and Signal Theory Department, University of Alicante, San Vicente del Raspeig, Spain*
[2]*Computer Science Research, University of Alicante, San Vicente del Raspeig, Spain*

Keywords: Robotic Manipulators, Visual Servoing, RGB-D Camera, Robot Operating System, Kinect.

Abstract: Image Based Visual Servoing (IBVS) is a robotic control scheme based on vision. This scheme uses only the visual information obtained from a camera to guide a robot from any robot pose to a desired one. However, IBVS requires the estimation of different parameters that cannot be obtained directly from the image. These parameters range from the intrinsic camera parameters (which can be obtained from a previous camera calibration), to the measured distance on the optical axis between the camera and visual features, it is the depth. This paper presents a comparative study of the performance of D-IBVS estimating the depth from three different ways using a low cost RGB-D sensor like Kinect. The visual servoing system has been developed over ROS (Robot Operating System), which is a meta-operating system for robots. The experiments prove that the computation of the depth value for each visual feature improves the system performance.

1 INTRODUCTION

Previously, IBVS systems used optical-visual sensors to guide robots and control their position. Usually, these sensors have been standard cameras. These cameras can be used isolated or arranged in a stereo pair to provide two images from different viewpoints and then create a perception of depth (Maru et al., 1993). Generally, these systems use radial models. Nevertheless, lately, visual servoing applications have used catadioptric cameras that provide panoramic images which give 360º of scene views (Hadj-Abdelkader et al., 2008).

In the last years, new sensors have changed the way in which the environment is sensorised and it is perceived by intelligent automatic systems. The Time of Flight (ToF) (Pomares et al., 2010) and RGB-D (Red, Green, Blue and Depth) (Teuliere and Marchand, 2012) cameras are an example. These visual sensors play an important role in the new perception context because they usually use a light source to compute the depth information. The result is a dense depth map. Therefore, ToF and RGB-D sensors do not require previous calibration step to estimate the depth as in IBVS composed of one camera. The visual servoing systems which use these sensors are known as D-IBVS. Moreover, the time computation of depth decreases in relation to the time taken by a stereo pair. Kinect is a low cost RGB-D camera which uses a depth sensor combined with another RGB. It can acquire colour images with a resolution of 640x480 pixels and a capture frame rate of 30fps at that resolution. Kinect uses an infrared projector which emits a fixed pattern of light and dark speckles. Depth is calculated by triangulation against a known pattern from the projector. The pattern is memorized at a known depth and then for each pixel, a correlation between known pattern and current pattern is done.

Traditionally, the development of new robotic applications has required a lot of researcher's time. The configuration of the different involved sensors or their integration with the robot is not a simple task and consumes a great percentage of the research time. Robot Operating System (ROS) is a middleware that provides to the user a set of libraries to easily develop new robotic applications (Quigley et al., 2009). The platform proposed in this paper is developed over ROS, taking advantage of the different stacks and packages provided by the ROS community to capture from Kinect (with OpenNI) or develop visual servoing applications with ViSP (Marchand et al., 2005) for ROS.

The paper is organized as follows: in Section 2 the classical IBVS system is described, Section 3 presents the platform proposed to perform IBVS

J. Garcia G., Gil P., Llácer D. and Torres F..
Guidance of Robot Arms using Depth Data from RGB-D Camera.
DOI: 10.5220/0004481903150321
In *Proceedings of the 10th International Conference on Informatics in Control, Automation and Robotics* (ICINCO-2013), pages 315-321
ISBN: 978-989-8565-71-6

tests using the depth information from a Kinect, Section 4 and 5 describe the experiments performed to validate the platform, and finally, Section 6 shows the conclusions.

2 IMAGE BASED VISUAL SERVOING

A visual servoing task can be described by an image function, $\mathbf{e_t} = \mathbf{s} - \mathbf{s}'$, which must be regulated to 0, where $\mathbf{s}$ is a M x 1 vector containing M visual features corresponding to the current state, while $\mathbf{s}'$ denotes the visual features values in the desired state.

The interaction matrix, $\mathbf{L_s}$, relates variations in the image with variation in the velocity of the camera, $\mathbf{\dot{s}} = \mathbf{L_s} \cdot \mathbf{\dot{r}}$, where $\mathbf{\dot{r}}$ indicates the camera velocity (Chaumette and Hutchinson, 2006). The control law of classical image-based visual servoing is obtained by imposing an exponential decrease of $\mathbf{e_t}$, as follows: $\mathbf{\dot{e}_t} = -\lambda \mathbf{e_t}$ where:

$$\mathbf{v}_c = -\lambda \hat{\mathbf{L}}_\mathbf{s}^{+} (\mathbf{s} - \mathbf{s}') \quad (1)$$

where $\hat{\mathbf{L}}_\mathbf{s}^{+}$ the pseudoinverse of an approximation of the interaction matrix and λ is the proportional control gain. When the visual feature is a point in the image, the interaction matrix can be obtained from:

$$\mathrm{L_s} = [\mathrm{L_{s1}}, \mathrm{L_{s2}}]$$

$$\text{with } \mathrm{L_{s1}} = \begin{bmatrix} \frac{f_u}{{}^{c}z_0} & 0 & \frac{-(f_x - u_0)}{{}^{c}z_0} \\ 0 & \frac{f_v}{{}^{c}z_0} & \frac{-(f_y - v_0)}{{}^{c}z_0} \end{bmatrix} \text{ and} \quad (2)$$

$$\mathrm{L_{s2}} = \begin{bmatrix} \frac{-(f_x - u_0)(f_y - v_0)}{f_v} & \frac{(f_x - u_0)^2 + f_u^2}{f_u} & \frac{-f_u (f_y - v_0)}{f_v} \\ \frac{-(f_y - v_0)^2 + f_v^2}{f_v} & \frac{(f_x - u_0)(f_y - v_0)}{f_u} & \frac{f_v (f_x - u_0)}{f_u} \end{bmatrix}$$

where (f_x, f_y) is the camera focal length in the axis X and Y respectively, and (u_0, v_0) is the camera principal point. These are the intrinsic parameters of the camera, and can be obtained with high precision through a calibration process performed in an off-line step of the visual servoing task (Zhang, 1996). The image point, (f_x, f_y) is directly measured in pixels from the image acquired by the camera. And the ${}^{c}\mathbf{z}_0$ value represents the depth between the camera reference system and the 3D point represented in the image by (f_x, f_y). In the next section, a test platform to study the significance of this depth in the interaction matrix calculation is proposed. The visual features, **s,** for our IBVS experiments are the pixel values of the centroid detected from circle marks. The depth involved in the interaction matrix computation are the distances between each feature and the RGB-D camera (Figure 1).

3 TEST PLATFORM PROPOSED

In order to remark the advantages of a RGB-D camera as Kinect for the guidance of a robot using a classical IBVS system, a new testing platform has been developed (see Figure. 1). The Kinect is placed in the end-effector of a Mitsubishi PA-10. The PA-10 is a robot manipulator of 7 degrees of freedom.

The visual servoing scheme described in (1) is used to guide the camera from any initial position to the desired position with reference to an object in the robot's workspace. This control law provides a velocity measured in the camera frame. Eye-in-hand configuration permits to compute the end-effector velocity from this camera velocity through the homogeneous matrix transformation between both reference systems. This matrix transformation is constant, as the camera is rigidly fixed to the robot's end-effector. Thus, the end-effector velocity is computed as $\mathbf{v_E} = \mathbf{T_{EC}} \cdot \mathbf{v_c}$, using the twist matrix obtained for the configuration proposed:

$$\mathbf{T_{EC}} = \begin{bmatrix} 0 & 0 & 1 & 0 & -155 & 0 \\ -1 & 0 & 0 & 0 & -15 & 0 \\ 0 & -1 & 0 & 15 & 0 & -155 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 \end{bmatrix} \quad (3)$$

The PA-10 robot controller provides a function to move the robot with a velocity command for the end-effector. However, the manufacturer firmware limits do that the robot can only be programmed over Microsoft Windows®. Using the proposed client-server scheme, the user can enter commands or operating parameters such as start-up, movement or speed. Thus, the client can be programmed for any operating system, such as Microsoft Windows® or Linux. The client-server connection is established via UDP protocol. In this way, the connection technology does not depend on the operating system. In our system, the computer network is safe. Thus, the sent commands always arrive in order and the errors control is managed from application-level of PA-10 controller. The flow and congestion control of transmission-level is not required so TCP is not necessary. Moreover, the controller of robot and the low volume data of the robot commands determine

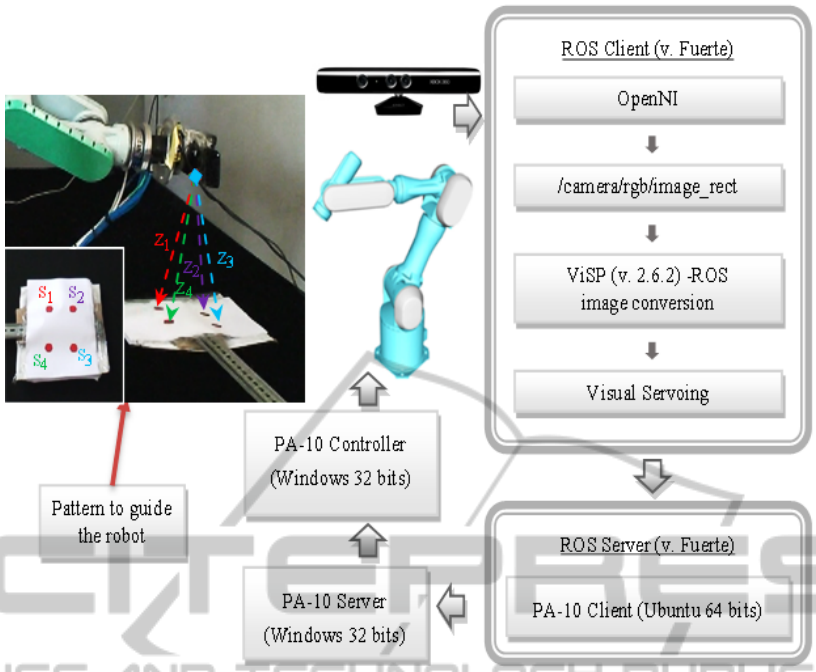

Figure 1: Robot end-effector in the Cartesian space and evolution of features in the image space.

that UDP is better than TCP because it does not introduce any delay to establish a connection. Further, this robot client-server scheme based on UDP allows that the test platform can be mounted over any operating system. Therefore, it is now possible to work directly over ROS on an Ubuntu system. ROS provides a set of stacks and packages that facilitate the robotic software development. ROS offers hardware abstraction, device drivers, libraries, visualizers, message-passing, package management, and more. Thus, between the two computers (PA-10 controller and perception system), it is required to send commands to the robot every time the controller computes a camera velocity from image features. To address this need, a service provider and a client are implemented in ROS. The service provider node contains robot's client. It expects orders or commands sent by the client of ROS, and then it replies them to the server of the robot. The client of ROS performs the functions of: image capturing by OpenNI, converting ROS to ViSP and computing the visual servoing control law using the ViSP libraries, sending the data to the ROS server and this in turn to the robot server which demands the correspondent speed to the robot controller.

4 TRAJECTORIES ANALYSIS

In order to test the visual servoing system described in Sections 2 and 3, and to successfully guide a robot according to an on-line depth estimation of each considered feature, three different experiments have been performed. The image processing issues are not the objective of this work, so, in order to guide the robot in these experiments, four visual features, $\mathbf{s}$ (representing the centroid points from circle marks) have been used (see Figure. 1).

The three experiments have the same initial conditions, because we want to compare the performance of the system with different depth information. Kinect acquires a first image and the set of visual features, $\mathbf{s}' = [s_1', s_2', s_3', s_4']$, is computed from it. This image is acquired with the robot located in its desired position. Then, the robot is positioned at its initial pose. The same set of visual features, $\mathbf{s}$, is acquired from a new image. IBVS aims to guide the robot comparing the two set of features. The property and type of these features determine the most appropriated IBVS to guide a robot. The first and last image positions (measured in pixels) are the same for the three experiments as they are shown in:

$$\mathbf{s}^{\mathrm{T}} = [316{,}42; 408{,}41; 325{,}158; 412{,}153]$$
$$\mathbf{s'}^{\mathrm{T}} = [349{,}177; 464{,}183; 457{,}314; 342{,}307] \quad (4)$$

where $\mathbf{s}_i = (f_{x_i}, f_{y_i}), \mathbf{s}_i' = (f_{x_i}', f_{y_i}')$ $\forall i = 1..4$.

4.1 Visual Servoing Task with the off-Line Depth of each Feature at the desired Position

The first experiment assumes that the only information provided by Kinect is the image provided by the RGB sensor. Thus, in this first experiment, the depth between the sensor and the object is not used. The experiment shows the system behavior when using a standard camera to guide the robot. The main problem detected with this scheme is that depth information of each visual feature is required for the computation of the interaction matrix, $\mathbf{L_s}$. A solution to achieve the positioning of the robot when the current depth information is not known at each iteration consists on computing a fixed interaction matrix, $\mathbf{L_s}$. In order to compute this fixed interaction matrix, the value of the depth component of each visual feature, at the desired position, z_i' is estimated and fixed for each iteration from a previous off-line camera calibration (Zhang, 1996). Then, it is not computed from Kinect. Thus, the same depth value is used for the four features and it is given as $\mathbf{z}' = {}^{C}\mathbf{z_0}' = (z_1', z_2', z_3', z_4')$.

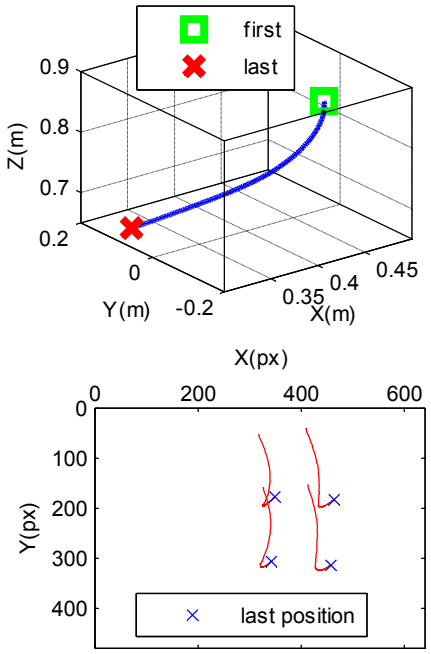

Figure 2: Robot end-effector in the 3D-Cartesian space and evolution of features in the 2D-image space without depth information.

Afterwards, it is used at each iteration in the computation of the interaction matrix as described in (2). It must be remarked that using this interaction matrix the visual servoing system is still able to achieve the goal position (see Figure. 2). Nevertheless, a non-desired behavior can be observed in the image trajectory of the visual features. The evolution of the features in the image is not a straight line between the initial and the final position. This must be the optimal behavior of an IBVS. Therefore, any of the visual features can be lost during the visual servoing task, not allowing the correct positioning of the robot. Taking this issue into account, the system is able to converge towards the desired position; it is last position. Nevertheless, this experiment has evidenced the need of measuring the depth between the camera and each of the visual features during the visual servoing task so a straight path image can be obtained as much as possible.

4.2 Advantage of using the on-Line Depth in the Computation of the Interaction Matrix

In order to improve the visual servoing behavior, in this experiment, the depth information provided by Kinect, from the fusion of RGB and IR sensors, is used to compute the interaction matrix. Kinect permits to introduce depth information in the calculation of the interaction matrix in the same online process. In this case, $\mathbf{z} = {}^{c}\mathbf{z}_o = (z_1, z_2, z_3, z_4)$ and the interaction matrix changes the values of z_i at each iteration of the visual servoing task. This update solves the problem detected in the previous experiment. Figure. 3 shows how the evolution of the features in the image follows a straight line. Nevertheless, it seems that a correct behavior in the image does not imply a correct behavior of the robot in the 3D space. In this case, the end-effector trajectory is not so straight than the trajectory performed in the previous experiment. Notwithstanding, this is the correct behavior of an IBVS system. The control law of an IBVS scheme (see equation (1)), is a proportional controller that minimizes the error in the image features but not in Cartesian space of the robot. So, the controller reduces this error exponentially and the system performs a straight line in the image space.

In the next section, a study of the exponential reduction of the error and an analysis of velocity can be seen for this case (Figure 5).

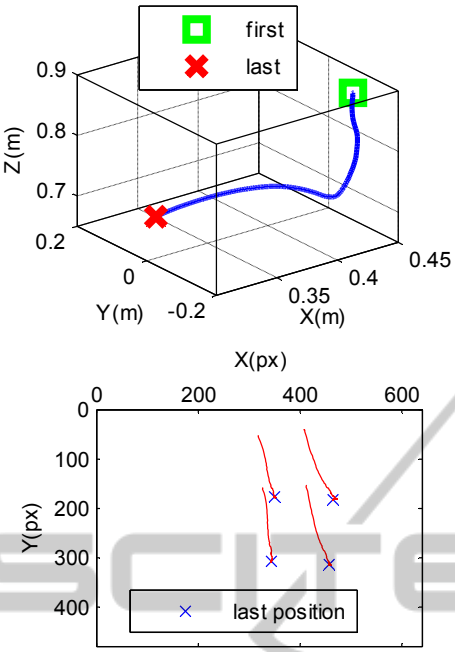

Figure 3: Robot end-effector in the 3D-Cartesian space and evolution of features in the 2D-image space using the z_i of each feature (depth data).

4.3 Visual Servoing Task with a Mean Interaction Matrix Computation

After testing the performance of the visual servoing system developed for the Mitsubishi PA10 over ROS with the use of Kinect in the two previous experiments, the system is now tested by fusing the two ways of computing the interaction matrix. In this last experiment the interaction matrix is computed as $\mathbf{L}_m$, a mean of the fixed interaction matrix computed as shown in (5), and the variable interaction matrix computed as shown in (2):

$$\mathbf{L}_m = 0.5(\mathbf{L}_s + \mathbf{L}_s') \quad (5)$$

As before, in the previous study case, to perform this computation, the depth information obtained by Kinect is required. This visual servoing scheme cannot be tested by using a standard camera. Indeed, a stereo system, a ToF camera, or a RGB-D camera is needed to perform the on-line calculation of the z_i for each visual feature.

Figure. 4 shows the evolution of the robot end-effector and the trajectory tracked by each feature in the image. The main conclusion is that the evolution of the features in the image is more straight than the obtained in the first experiment and it is similar to second experiment. In addition, the robot behavior has been improved with reference to the 3D-cartesian space obtained by the variable interaction matrix used in the second experiment (computed using the current z_i of each visual feature at each control loop iteration). This is an interesting method to compute the interaction matrix as it improves the behavior of the two previously tested methods when Kinect is used as capture sensor.

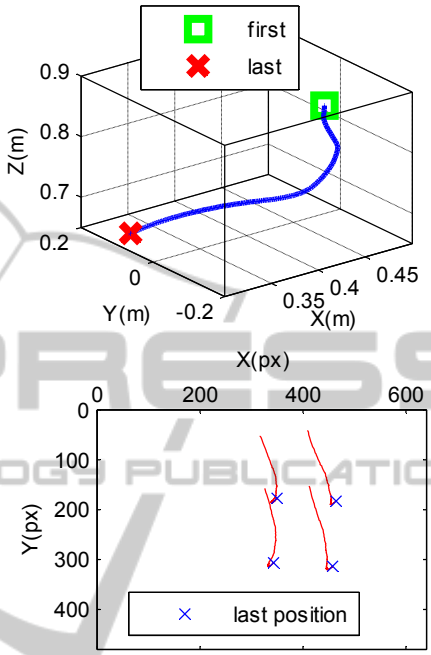

Figure 4: Robot end-effector in the 3D-Cartesian space and evolution of features in the 2D-image space using a mean value in the interaction matrix.

5 ERROR ANALYSIS

In this section, the other differences in performance of the studied cases are presented. From the results shown in Section 4, the position error measured from visual features (Figure 5a) and the linear velocity of PA-10 end-effector (Figure 5b) determine the quality of performance for PA-10 position control in the three experiments. The gain, involved in the control law is chosen constant and a small value in order that the camera velocity is not too big at the beginning of the servoing (λ=0.5).

The output of the visual controller is the velocity that the camera must perform at each iteration in order to decrease the error computed among the current and the desired visual features. Initially, in the first position, the error of each feature is maximum because the starting position of the robot is the farthest. While the robot goes near to the desired position, the error minimizes to almost zero

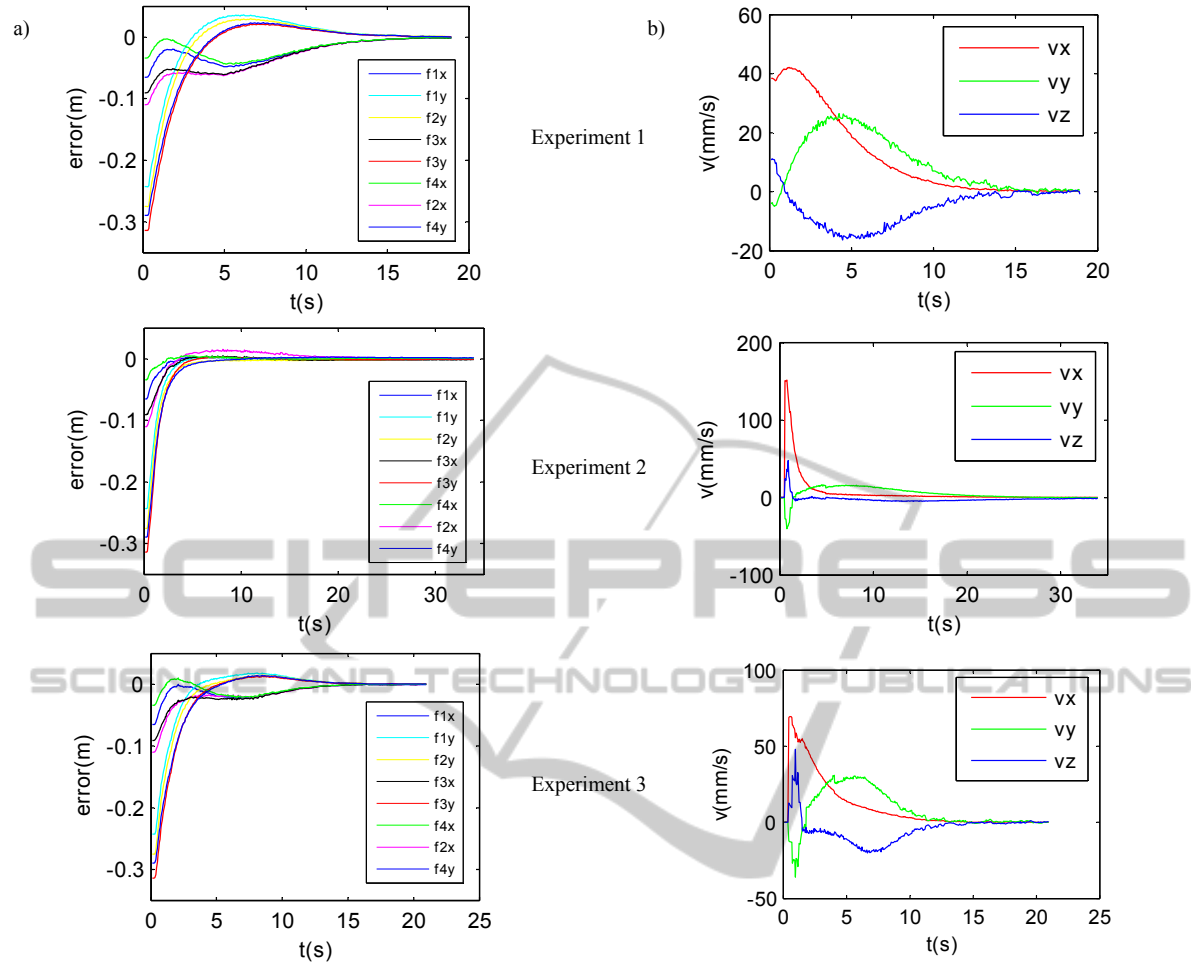

Figure 5: Visual features error measured in meters and end-effector linear velocity during the visual servoing task. a) Without depth information. b) Using the z_i of each feature.

(Figure 5a). Velocity can be obtained with reference to the end-effector by computing the fixed transformation between both reference systems: the end-effector and the Kinect one (as explained in Section 3)

On the one hand, the comparison of the evolution of the computed error determines that the computation of the dynamic depth of each feature, z_i, from Kinect (experiment 2) presents an exponential decrement of the error with respect to the other experiments, with a static depth fixed previously or with mean interaction matrix. Even though, in the second experiment, more time is required to complete the task than the first and third experiments if the same gain, λ=0.5, is used. In addition, the oscillation peaks are due to the change of depth between iterations.

The second and third experiments take longer to converge because they use more input data: both (f_{x_i}, f_{y_i}) in pixels and z_i in mm are used. But the results are more accurate and more realistic because the robot is moved in 3D, it is changed the depth between effector and pattern-object.

In all experiments, the reader can note that the convergence of the features coordinates to their desired value, permit to demonstrate the correct realization of the guided task. The robustness of the guided task depends on the method used to estimate the features depth: off-line camera calibration, dynamic computation of depths or mean interaction matrix.

Furthermore, in order to evaluate the capabilities of the proposed test platform for running in real-time, the runtimes of the different steps are calculated from experimental results (Figure 6). Each step is executed each time a new image is captured until the desired position is reached, it is at each iteration. Figure 6 shows the runtime of the D-

IBVS task identifying the three steps of algorithm: time of image processing, visual servoing and transmission and robot movement.

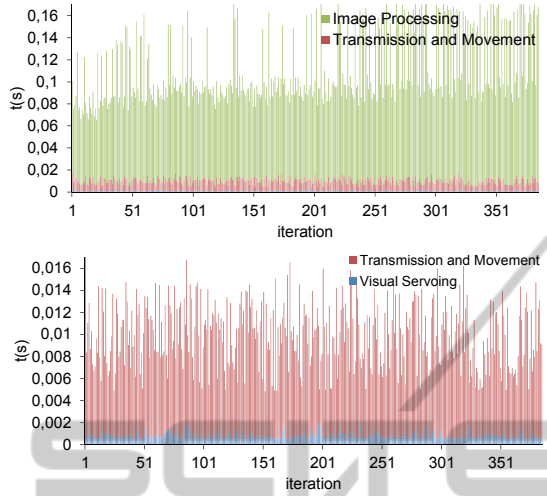

Figure 6: Runtime comparison of different steps for each iteration.

The runtimes have been calculated over an Intel Core i7-263QM 2Ghz of CPU with 4GiB of memory RAM and GeForce GT 540 with 2GiB of GPU. Thus, in this hardware, the mean runtime for the different computing steps, according the iterations shown in Figure 6, are shown in Table 1.

Table 1: Mean runtimes.

	Image Processing	Visual Servoing	Network transmission
Mean runtime	100.58ms	0.80ms	9.14ms
Standard deviation	0.0324ms	0.0003ms	0.0022ms

6 CONCLUSIONS

A new platform to perform visual servoing tests using RGB-D sensors has been proposed. Thus, this platform implements an IBVS based on low cost RGB-D sensors (D-IBVS) that permits to simultaneously control the distance and Cartesian position of a robot arm with reference to objects that will be manipulated. D-IBVS does not require the estimation of the 3D pose as position based visual servoing techniques. It provides extra-information such as depth to improve positioning accuracy if it is compared with classic IBVS. In addition, the response time was measured through a series of experiments. Note that a D-IBVS is advantageous in a variety of applications requiring robot guidance such as control to grasp and manipulating tasks by robot arms where the robot tracks trajectories for moving away and/or moving near of the objects. As future works, some issues related to extract on-line 3D features for visual servoing without fiducial markers are being investigated over the same testing platform proposed.

ACKNOWLEDGEMENTS

The research leading to these results has received funding from the Spanish Ministry of Education and Science and European FEDER funds, the Valencia Regional Government and the Research and Innovation Vice-president Office of the University of Alicante, through the research projects DPI2012-32390, GV2012/102 and PROMETEO/2013/085, GRE10-16, respectively.

REFERENCES

Chaumette, F., Hutchinson, S., 2006. Visual Servo Control, Part I: Basic Approaches. *IEEE Robotics and Automation Magazine* 13(4), 82-90.

Hadj-Abdelkader, H., Mezouar, Y., Martinet, P., Chaumette, F., 2008. Catadioptric visual ser-voing from 3-D straight lines. *IEEE Transactions on Robotics*, 24, 652-665.

Malis, E., Chaumette, F., Boudet, S., 1999, 2-1/2-D Visual Servoing. *IEEE Transactions on Robotics and Automation*, 15(2), 238-250.

Marchand, E., Spindler, F., Chaumette. F., 2005. ViSP for visual servoing: a generic software platform with a wide class of robot control skills. *IEEE Robotics and Automation Magazine*, 12(4), 40-52.

Maru, N., Kase, H., Yamada, S., Nishikawa, A., Miyazaki, F., 1993. Manipulator control by visual servoing with stereo vision. In *Proceedings of IROS'93, IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE-Press.

Pomares, J., Gil, P., Torres, F., 2010. Visual control of robots using range images. Sensors. 10(8), 7303-7322.

Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T. B., Leibs, J., Wheeler, R., Ng, A.Y., 2009. ROS: an open-source robot operating system. In *Proceedings of ICRA'09, IEEE International Conference on Robotics and Automation-Workshop on Open Source Software.*

Teuliere, C., Marchand, E., 2012. Direct 3d servoing using dense depth maps. In *Proceedings of IROS'12, IEEE/RSJ International Conference on Intelligent Robots and Systems, 1741-1746.*

Zhang, Z., 1996. Three-dimensional reconstruction under varying constraints on camera geometry for robotic navigation scenarios. *PhD. Thesis*, University of Massachussets.