Memorias del XVI Congreso Latinoamericano
de Control Automático, CLCA 2014
Octubre 14-17, 2014. Cancún, Quintana Roo, México

Eye-to-Hand Position Based Visual Servoing and Human Control Using Kinect Camera in ViSeLab Testbed

Roger Esteller-Curto*, Alberto José Álvares, Raul Marin *****

**Department of Computer Science and Engineering. Jaume-I University. Spain. (e-mail: esteller@uji.es)*
***Departamento de Engenharia Mecânica e Mecatrônica. Universidade de Brasilia. Brasília, Brasil (e-mail: alvares@alvarestech.com)*
****Department of Computer Science and Engineering. Jaume-I University. Castelló, Spain (e-mail: raul.marin@icc.uji.es)*

Abstract: This paper shows the potentialities and lacks of the use of a Kinect camera for Position Based eye-to-hand Visual Servoing (PBVS). This technique has been tested using a 6 DOF robot (ABB IRB 140) and using human control. The PBVS requires the 3D position calculation of the object to extract the feature points using it as input for the control loop of the VS. The use of Kinect camera reduced the complexity of the PBVS because that camera is capable to gather depth information, so estimation was unnecessary. On the other hand, new problems have arisen. Compared with most common 3D systems that use two cameras, the Kinect avoids complex mathematics calculation to extract the 3D feature points, but add the need for calibration (of the work space), and use of other more confident references (not a single point, but a skeleton). Experimental results are presented to show the most convenient scenarios where the use of Kinect could be feasible and others where VS should be improved using other configurations (cameras or other sensors). This paper shows also the ViSeLab (Visual Servoing Laboratory) architecture, how some of the challenges have been solved. We offer ViSeLab to the researchers community.

Keywords: Kinect, Robot, Kinematics, Visual Servoing, ViSeLab.

1. INTRODUCTION

The Visual Servoing (VS) techniques are commonly used to control a robot end-effector; a camera gathers information about the laboratory scenario extracting feature points of the targets Suradjuddin et al. (2012) and Chaumette (2006). The robot then should be capable to interact with those targets, the robot can pick up, track or perform any other action on the scenario. This all depends on the specific objectives of the VS algorithm. In case of Position Based Visual Servoing (PBVS) with an eye-to-hand configuration, the image processing algorithm should be capable to extract the feature points from the image and later to obtain 3D points that will be used by the PBVS algorithm to move the robot end-effector.

As a single camera can only obtain 2D points, to get these 3D points have been always a challenging task. Solutions for this problem have been usually to use 2-cameras or to know the real object features, using invariant moments or other techniques to calculate the 3D position.

In this paper, we have tested the use of a Kinect sensor, which incorporates two cameras and an infrared light emitter. The first is an ordinary RGB camera; the second is an Infrared camera that ”sees” the points that an infrared light emits. 2-cameras configuration, need to consider exact points in both cameras to get a single 3D point. In this case, the depth information is added to every single pixel of the RGB image.

There exists previous research that use the Kinect camera for tracking Borenstein (2012), Nakamura (2011) and Siradjuddin et al. (2012), but these papers focus on the possibilities of using the Kinect camera to command instructions to the robot, to move to a certain position, grasp an object or interact with the environment. To get this, it is necessary to know first, to which extend the Kinect is precise and reliable.

There exist other frameworks and toolboxes to manage robots. Sometimes they are focused on education Perez-Vidal et al. (2011), this is the case of RoKiSim (2011), that simulates the Kinematics of several models of robots, others focused on research on Visual Servoing, as the ViSP (Marchand et al., 2005) coded in C++ or the Robotics Toolbox for MATLAB (Corke, 1996 and 2011). In this case, ViSeLab aims to be a completely functional software package, capable to work in simulation and real robots but also very modular; where pieces could be removed and changed without having to worry about other parts of the software.

In the scientific literature we can find two main Visual

Servoing techniques, the Image Based Visual Servoing and the Position Based Visual Servoing (Chaumette, 2006) and (Chaumette and Hutchinson, 2007) that is IBVS and PBVS respectively. Improvements to these techniques have been proven to be more effective, as the 2D1/2 (Chaumette et al., 1997), but there are always new applications and challenges.

When researching on new VS techniques, new robots or sensors, or in new applications, it becomes necessary to have a framework or test-bed, to implement them, and at the same time providing statistics and marks, that could be used later as benchmarks.

2. ARCHITECTURE ViSeLab

ViSeLab (Visual Servoing Laboratory) has been programed using Java Swing components for the interface, but the main functionalities are Java Classes independent from the user interface. On figure 1 can be seen the first window that appears once the program is started. As it can be seen, the robot, the VS method and the camera are configurable. This interface acts using the Factory Pattern, which means that instantiates the convenient classes depending of the option closed.

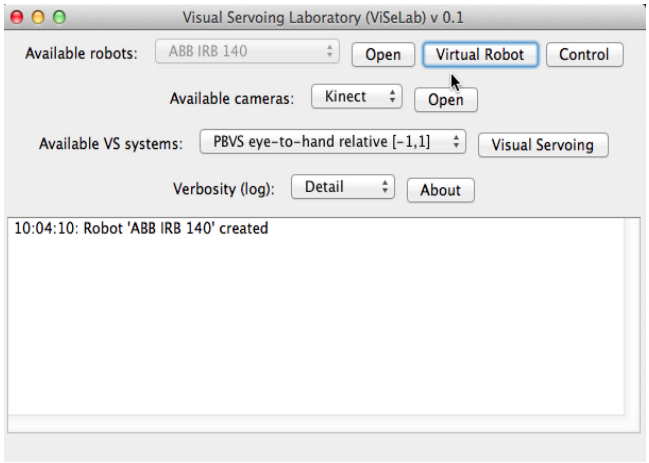

Fig. 1. ViSeLab first window.

2.1 Robot

Two robots have been tested but more can bee added (figure 2). As far as they fit the Robot interface, they can be used in ViSeLab. Robot interface says that any implementation of that class should have methods for moving the robot to a position and get the position and orientation of the end-effector.

It is not possible to create an instance of the robot without knowing the model or if it will be real or virtual. Only with that information, the Robot Manager is then capable to create a robot using the Factory Pattern. That information is provided from the starting window (figure 1). The Robot Manager follows the Singleton Pattern, it can be called from anywhere from the code without the risk of duplicating the instance.
A ”Simulated Robot” class has been implemented to allow the simulation any robot arm, therefore it is capable to calculate direct and inverse kinematics. That is done knowing the Denavit-Hartemberg parameters. The inverse kinematics is calculated using an analytical Jacobian Matrix and interactively trying to reduce the error. This general-purpose solution makes easy to simulate any kind of robot (real or imaginary). To be able to process this direct and inverse kinematics, and also because it is necessary to know the parameters of the robot during the simulation (e.g. Denavit-Hartemberg, the degrees of freedom or velocity.) static classes provide that information. This static classes should be seen as data stores of information (figures 2 and 3).

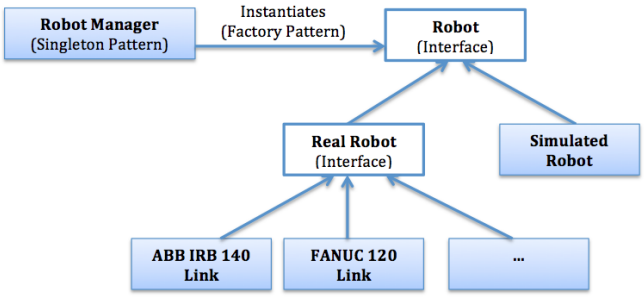

Fig. 2. Robot class interface.

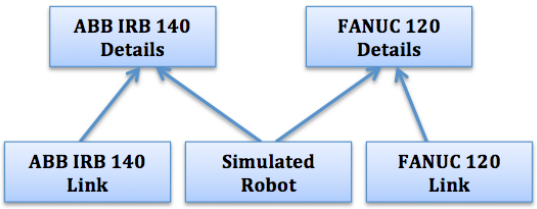

Fig. 3. Detailed classes, providing static information.

One important functionality of the Robot Manager is that it is capable to register listeners. That means, that any other class that is interested to know something about the robot, can request its registration. So when the robot connects, moves or disconnects, all the registered classes will be notified.

The robot can be controlled through the panel in figure 4, what is useful to test the link with the real robot or to test the direct and inverse kinematics on the virtual robot.

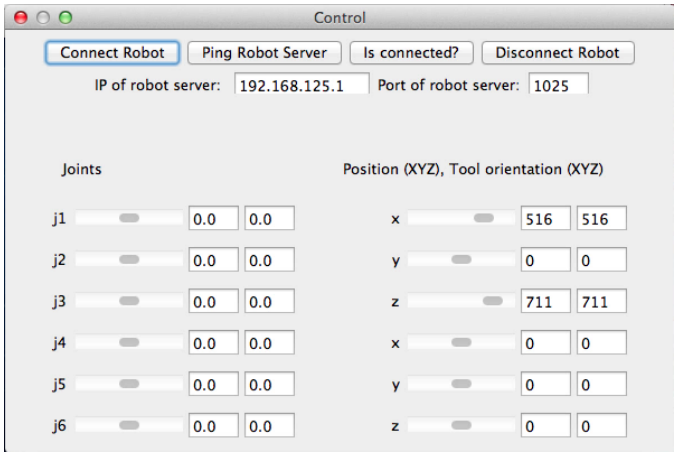

Fig. 4. Robot control windows.

2.2 Visual Servoing

The Visual Servoing class (VS) is an interface. Any method of VS should implement the description defined by the interface. Also, they should implement a Singleton pattern, which means, that any other object can ask and request an instance of VS and also prevents to have more than one instance running. Figure 5 shows this configuration; it has been implemented a PBVS, an IBVS which are the most traditional, but also, in this experiment, a PBVS 3D relative pose has been tested (figs. 5 and 6).

VS registers itself as a listener in the Robot Manager, resulting that any operation to the robot (as connection or disconnection) will be notified. This can be used for starting the VS control loop or to stop it, also as a way of monitoring the pose. When there is a robot connected, VS class requests an instance of the robot to the Robot Manager (it does not matter the model of the robot, if it is real or virtual). From then, the VS class is capable to send movement commands to the robot.

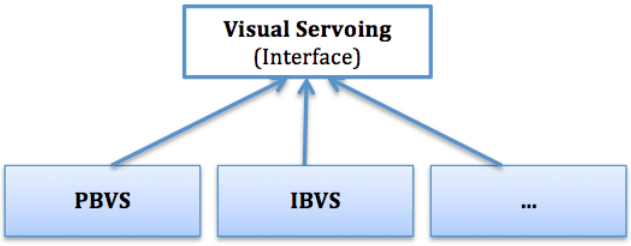

Fig. 5. Visual Servoing interface – Classes.

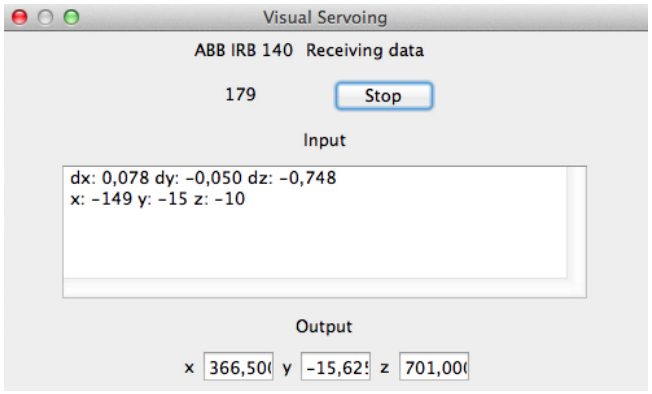

Fig. 6. Visual Servoing interface – Java GUI.

2.3 Camera

The Camera class is again an interface. Its main mission is to send the feature points to the Visual Servoing class. Thanks that the VS class is Singleton, it can be called from any other object without risk of creating another instance. The Camera class objective is to capture the image, gather the feature points, and depending of the VS method, to send the feature points to VS class that should process conveniently.

2.4 Visual Robot

A 3D virtual robot (figure 7) can be shown and will move accordingly to the Robot (no matter if the robot is real or simulated). Once created, will call RobotManager and register it as a listener. Any new event to the robot as a change in the pose, will receive a notification with the new pose.

3. TESTING ViSeLab

Two models of 6 DOF robots have been tested with ViSeLab, the ABB IRB 140 and the Fanuc 120. As it can be seen in Figure 3, the class ”Robot Link” is the one that connects directly to the robot. The IRB uses sockets and the Fanuc uses REST to connect using Internet to the robot. Any new robot added to ViSeLab should implement the interface. No matter if it is a 6 DOF or any other kind of robotic arm.

The Robot interface only manages coordinates and the orientation of the end-effector Any class interested to know something about the robot (if it is connected, if it has moved, etc.), should register a listener in the Robot Manager class (figure 2). This is the case of the Virtual Robot, which does not have to check the position of the robot. It only waits for notification of the position change.

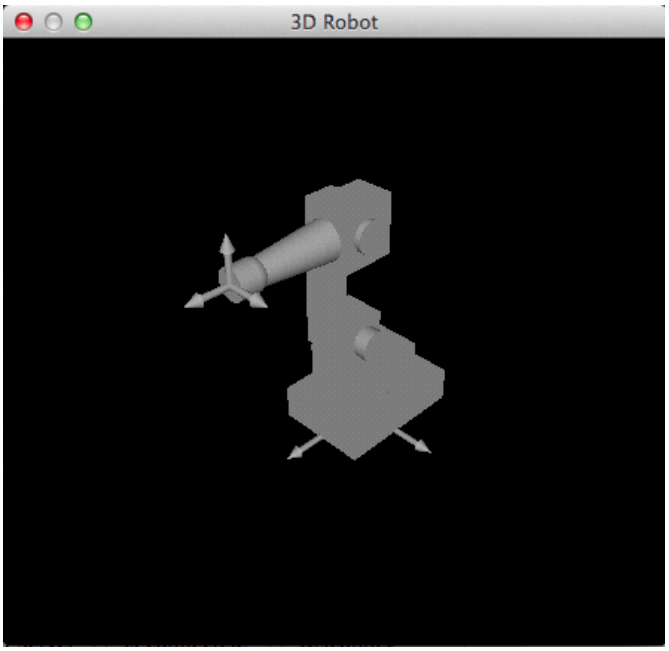

Fig. 7. 3D virtual robot.

The Visual Servoing used to test the platform is a PBVS eye to-hand using a 3D camera. To design this VS algorithm, it has been enough to implement the Class Visual Servoing based on the standard interface. By this way, behind of the traditional PBVS and IBVS it has been possible to add a new method for VS without having to worry about the other parts of the framework (the robot, connection, simulation or control). That was the first aim to the ViSeLab; to make easy experimentation.

Secondly, the ViSeLab offers a class for logging, which is used for the typical feedback to the user (information, warning, errors, and debug) depending on the level required. Furthermore, registers any input and output to the Visual

Servoing Class (fig. 8). This provides automatically statistics of how the process went.

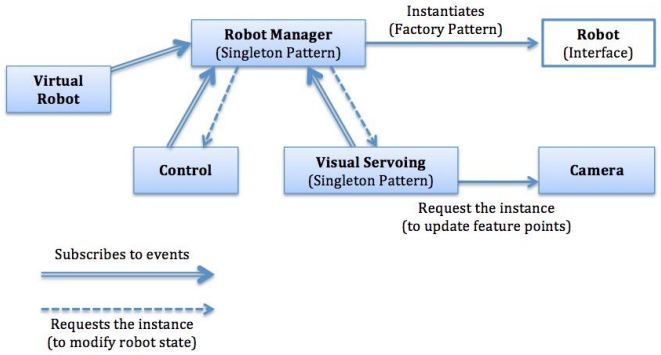

Fig. 8. Data flow between classes.

The ViSeLab platform aims to be a highly modular model, where functionalities could be implemented and tested ensuring thanks to the interface that the pieces of the control loop could be interchanged. In the next experiment, we wanted to show one specific case applied to ViSeLab, and how it can be implemented and tested. Thanks to this, the procedure of a student or a researcher can focus what exactly is important for their experiments, while all other components can be maintained (camera, image processor, robot, statistics, etc).

4. CONFIGURATION ABB IRB 140 MODEL

In this configuration we aimed to set the experiment as simple as possible, focusing only on Kinect capabilities, its potentialities and disadvantages. The camera used has been the Microsoft Kinect (Nakamura, 2011), which is capable to obtain a RGB image as any ordinary camera, but also the depth information about almost any pixel. Information about depth was lost in pixels that were at borders of the objects.

The experiment has been implemented using Java language. The 6 DOF used has been an ABB IRB 140 model (figure 9). The communication between Java program and the robot has been done using Sockets; the robot had a RAPID server program listening and waiting for commands. The robot provided pose functionalities, so it was not need for calculus of direct and inverse kinematics.

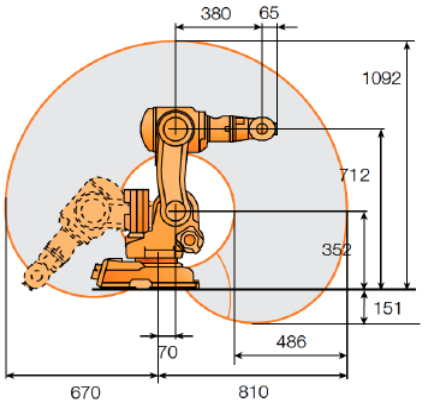

Fig. 9. 6 DOF robot, model ABB IRB 140.

To gather information of the Kinect camera, it has been used the SimpleOpenNi (Borenstein, 2012) library which provides an interface to be used in Java.

5. VISUAL SERVOING CLOSED-LOOP

The most common architectures for VS are the Position Based Visual Servoing (PBVS) and the Image Based Visual Servoing (IBVS), in this experiment, a configuration eye to-hand in PBVS was tested. The reason to choose PBVS is that the Kinect can provide depth information that would be very useful in the 3D pose object estimation. The eye to-hand configuration would also take advantage of other potentialities of the Kinect, that is the skeleton estimation. We will see that this has been very useful for pose estimation.

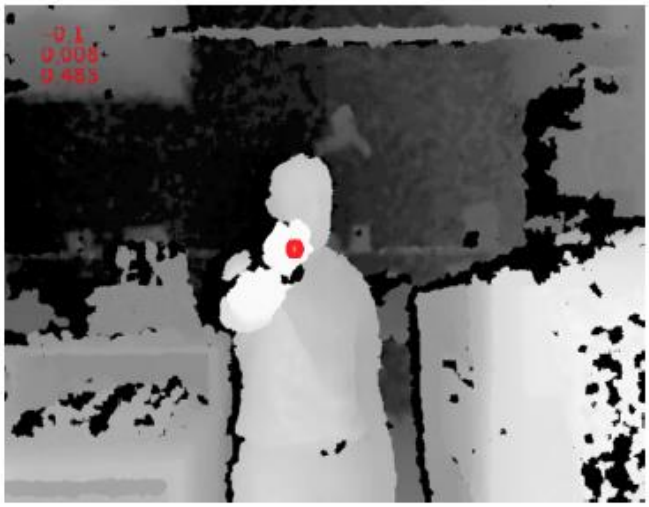

Fig. 10. Depth features points capture.

In traditional PBVS (Siradjuddin et al., 2012 and Chaumette, 2006), eye-on-hand there exists 3 coordinates frames, the camera frame: **Fc**, the desired camera frame **Fc*** and the reference frame of the object **Fo**. As **Fo** is unknown, it should be estimated. The **Fc** of the camera can be easily transformed to the robot frame **Fr** as the camera is at the end effector of the robot, the same way, it is easy to obtain **Fc** from **Fr** and viceversa. Having estimated the **Fo** then it is possible to move the robot to reach **Fc***. But in the eye-to-hand configuration, the camera is not at the end-effector of the robot (now **Fc** and **Fr** could not be easily obtained from each other), so now the information gathered about the **Fo** relative to **Fc** is used as an input to move the end-effector, that is **Fr***, now the aim to move the camera no longer exists, but to move the robot (Eq. 1).

$$F_{r^*} = f(F_c, F_o) \qquad (1)$$

The desired position of the robot (**Fc_**) is based in the camera frame (**Fc**) and the object (**Fo**). This configuration would be useful when a person is going to give instructions to the robot to move, to pick up an object or to do any other action in the workspace. As the Kinect camera was designed to be used to recognize human shape, a first assumption would be that this is a good approach.

6. EXPERIMENTS

The experiments were designed to make the most profit of two of the characteristics of the Kinect, the first is the depth estimation (figure 10), the second the use of the Skeleton detection (figure 11).

The first experiment consisted in tracking using the depth. The SimpleOpenNi would provide the (u,v,z) point of a specific object in the scene. To avoid processing time and complexity in the image algorithm, the feature point would be the nearest to the Kinect camera. In this case it will be a person with the hand raised. Figure 12 shows the points gathered during 10 seconds at a distance of 60 cm while the person on figure 13 was at 230 cm.

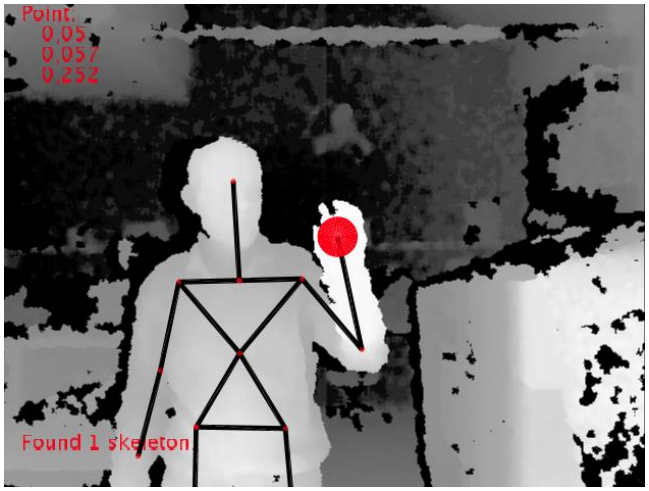

Fig. 11. Skeleton features point's capture.

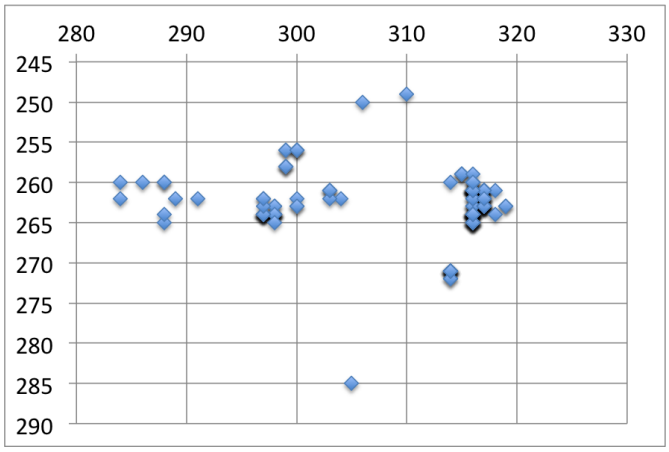

Fig. 12. Gathering data during 10 s in depth configuration at 60 cm.

Being a person that remains as still as possible, the data gathered in both experiments (figure 12 and figure 13 show very disperse values, with a typical deviation in **(**x; y; z**)** of (9:4; 3:8; 1) at 60 cm and (4:3; 1:6; 16) at 230 cm.

Using the information of the depth estimation and the skeleton, and applying a PBVS eye-to-hand configuration as in equation (1), where f is a linear function than displaces **Fr** as many millimeters as **Fo** pixels, resulted that the robot were very unstable, moving around 1 cm even when the hand remained still, this could be observed during the experiment with a skeleton (Figure 10) or during the movement tests (Figure 15)

In case that it was need to use gestures to order the robot to pick up an object, or repeat a movement, it would have been necessary to add a tolerance of distance of 10 pixels, considering a new robot position valid only if it was 10 pixels (distance in x, y, z). This made the robot more stable, but it loosed the precision.

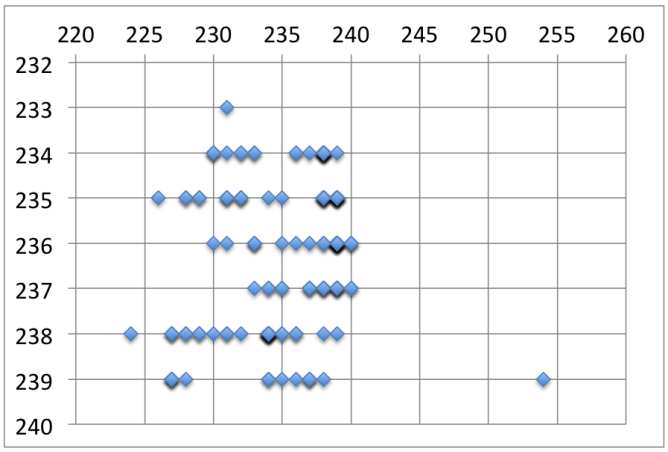

Fig. 13. Gathering data during 10 s in depth configuration at 230 cm.

When using the skeleton for tracking a point (figure 14) the typical deviation of the coordinates obtained are (3:3; 2:2; 6:4) at 120 cm.

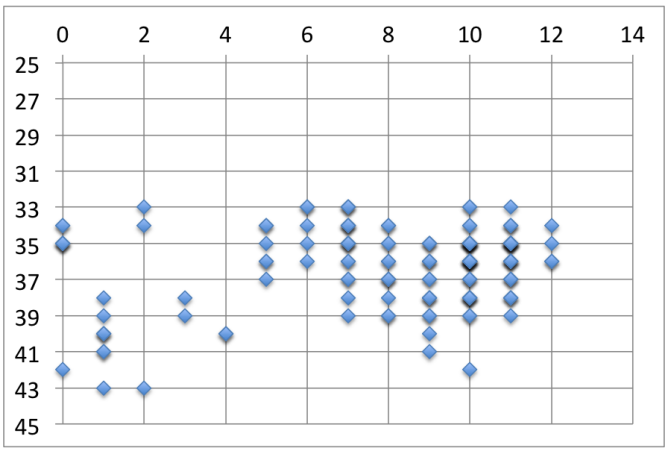

Fig. 14. Gathering data during 10 s in skeleton configuration at 120 cm.

To test if the error caused when gathering the position of a person was because the Kinect or that the difficulties in being still, we performed another experiment. It consisted in gathering the depth data of a ball (5 cm diameter) that was at the end effector of a robot. The movement was repeated 3 times, making an arc. The graphical representation can be

seen in figure 15 and in detail in figure 16. It can be seen that the data gathered by the Kinect camera was not precise. All the points were ranging between +- 9 pixels, although it can be observed in figure 15 that some points are completely outside of the path followed by the robot.

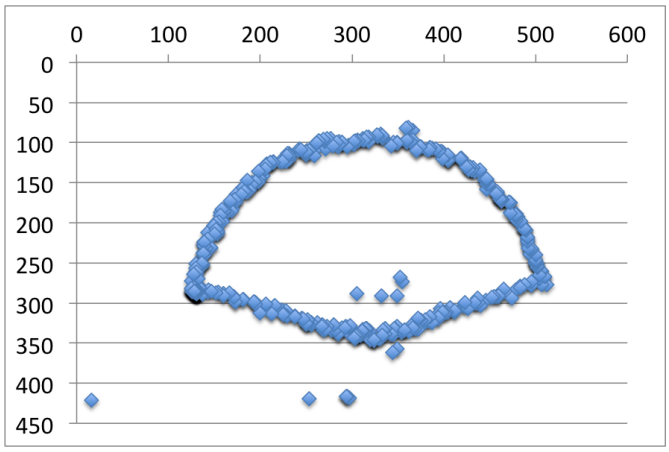

Fig. 15. Gathering data during movement of the robot end-effector (3 times).

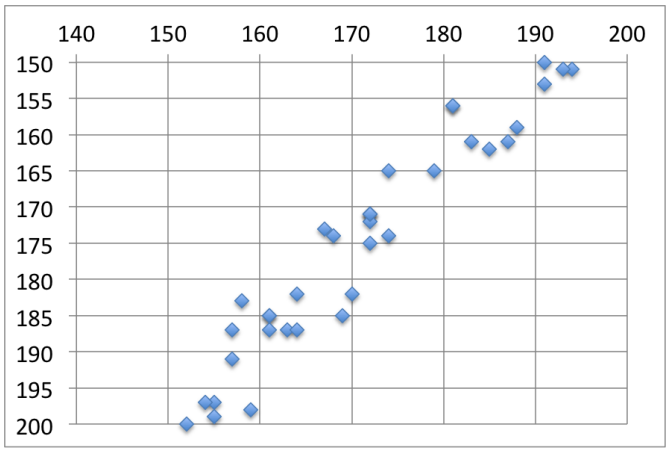

Fig. 16. Detail of the data gathered during movement of the robot and-effector (3 times).

7. CONCLUSIONS

The use of the Kinect sensor provides great advantages because we have the RGB image together with depth information that can be used to help to extract feature points. The Kinect is useful when the user does not require precision. Unfortunately, when it is necessary to perform any action with objects (as welding, grasping, etc.) it becomes necessary to reduce the error as much as possible.

The ViSeLab software is available at SourceForge and can be used freely by any developer under the GNU GPL license. The framework has been proven to be useful in a first stage making possible to add, remove and modify components (e.g. the robot, camera, VS system). On the other hand, it has been also difficult to make the Visual Servoing system more independent from the Camera and the Robot.

In case the robot is not a robot-arm but a mobile robot, or the VS system is more complex and need a specific camera, then the common established interface becomes too simple. In this case, the camera and the VS method are too dependent; it is necessary to get a wider abstraction and generalization. It is planned to go further in the project by increasing the potentiality of the ViSeLab platform with more algorithms and cameras and keep updating the project in SourceForge (http://sourceforge.net/projects/viselab).

REFERENCES

Borenstein, G. (2012), Making Things See. *Sebastopol: O'Reilly*, 2012.

Chaumette, F. (2006). "Visual servo control. part i: Basic approaches," *IEEE Robotics and Automation Magazine*, vol. 13, no. 4, pp. 82–90, 2006.

Chaumette, F., Malis, E. and Boudet, S. (1997), "2d 1/2 visual servoing with respect to a planar object," 1997.

Chaumette, F., Hutchinson, S. (2007), "Visual servo control. part ii: *Advanced approaches,*", *Robotics & Automation Magazine*, IEEE, Vol. 14, vol. 14, no. 1, pp. 109–118, 2007.

Corke, P. (1996), "A robotics toolbox for MATLAB*,"* *Robotics Automation Magazine*, IEEE, vol. 3, no. 1, pp. 24 –32, Mar. 1996.

Corke, P. (2011), "Robotics, Vision and Control Fundamental Algorithms in Matlab", *Spring-Verlag*, Berlin, Heidelberg, 2011.

Marchand, E., Spindler, F. and Chaumette, F. (2005), "Visp for visual servoing: a generic software platform with a wide class of robot control skills," *Robotics & Automation Magazine*, IEEE, vol. 12, no. 4, p. 4052, 2005.

Nakamura, T. (2011), "Real-time 3-d object tracking using kinect sensor," *in 2011 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Dec. 2011, pp. 784 –788.

Perez-Vidal, C., Garcia, N., Cervera, E., Gracia, L. and Sabater, J. M. (2011), "The on-line teaching of robot visual servoing," *International Journal of Electrical Engineering Education*, vol. 48, no. 2, p. 202216, 2011.

Rokisim (2013), Control and Robotic Lab, "RoKiSim - robot kinematics simulator", http://www.parallemic.org/RoKiSim.html. [Online].

Siradjuddin, I., Behera, L., McGinnity, T. and Coleman, S. (2012), "A position based visual tracking system for a 7 DOF robot manipulator using a kinect camera," in The 2012 *International Joint Conference on Neural Networks (IJCNN)*, June 2012, pp. 1 –7.