JOURNAL OF SCIENCE
SAKARYA UNIVERSITY

Sakarya University Journal of Science

ISSN 1301-4048 | e-ISSN 2147-835X | Period Bimonthly | Founded: 1997 | Publisher Sakarya University | http://www.saujs.sakarya.edu.tr/en/

Title: Real Time Application for Automatic Object and 3D Position Detection and Sorting with Robotic Manipulator

Authors: Tichaona Jonathan MAKOMO, Kenan ERİN, Barış BORU
Recieved: 2019-12-05 15:29:52

Accepted: 2020-05-24 22:29:03

Article Type: Research Article
Volume: 24
Issue: 4
Month: August
Year: 2020
Pages: 703-711

How to cite
Tichaona Jonathan MAKOMO, Kenan ERİN, Barış BORU; (2020), Real Time Application for Automatic Object and 3D Position Detection and Sorting with Robotic Manipulator. Sakarya University Journal of Science, 24(4), 703-711, DOI: https://doi.org/10.16984/saufenbilder.655716
Access link
http://www.saujs.sakarya.edu.tr/en/pub/issue/55932/655716

Sakarya University Journal of Science 24(4), 703-711, 2020

Real Time Application for Automatic Object and 3D Position Detection and Sorting with Robotic Manipulator

Tichaona Jonathan MAKOMO*[1], Kenan ERİN[2], Barış BORU[3]

Abstract

This work deals with the likelihood of merging a 3D sensor into a robotic manipulator, with an objective to automatically detect, track and grasp an object, placing it in another location. To enhance the flexibility and easy functionality of the robot, MATLAB, a versatile and powerful programming language is used to control the robot. For this work, a common industrial task in many factories of pick and place is implemented. A robotic system consisting of an ABB IRB120 robot equipped with a gripper and a 3D Kinect for Windows camera sensor is used. The three-dimensional data acquisition, image processing and some different parameters of the camera are investigated. The information in the image acquired from the camera is used to determine the robot's working space and to recognize workpieces. This information is then used to calculate the position of the objects. Using this information, an automatic path to grasp an object was designed and developed to compute the possible trajectory to an object in real time. To be able to detect the workpieces, object recognition techniques are applied using available algorithms in MATLAB's Computer Vision Toolbox and Image Acquisition Toolbox. These give information about the position of the object of interest and its orientation. The information is therefore sent to the robot to create a path through a server-to-client connection over a computer network in real time.

Keywords: Kinect, object recognition, 3D vision system, MATLAB, RobotStudio

1 INTRODUCTION

This work demonstrates a simple and common industrial task of manipulating objects by implementing a pick and place method using information from a vision system that is integrated with a high precision robotic manipulator. The main thrust of this

* Corresponding Author: tichaona.makomo@ogr.sakarya.edu.tr
[1]Sakarya University, Faculty of Technology, Department of Mechatronics, Sakarya, Turkey.
ORCID: https://orcid.org/0000-0002-9860-6179
[2]Sakarya University of Applied Sciences, Faculty of Technology, Department of Mechatronics, Sakarya, Turkey.
ORCID: https://orcid.org/0000-0003-4714-1161. E-mail: kenanerin@sakarya.edu.tr
[3]Sakarya University of Applied Sciences, Faculty of Technology, Department of Mechatronics, Sakarya, Turkey.
ORCID: https://orcid.org/0000-0002-0993-3187. E-mail: barisb@sakarya.edu.tr

work is to create a successful high precision grasp planning methodology using the image information from one Kinect for Windows sensor. Due to the increase in the advanced control applications, powerful hardware platforms and enhanced sensing capabilities, robotic systems have started to find place into different functionalities like mapping, exploration, entertainment industry, and wellbeing of people among others. Not a long time ago, robotic systems have been finding much use in many processing and manufacturing industries. They have been used semi-automatic, that is working side by side with humans whereas in some cases they are fully automatic, that is robots working with each other to complete a task. The objects being worked on would come with invariable physical attributes like color, size and shape. A real-time sensing system is therefore needed to be able to facilitate real time information about every object in the robot's control system. [1]. A number of technological advancements have been made in the development of flexible manufacturing systems (FMS). An FMS is described as a system consisting of one or more handling devices like robotic manipulators along with the robot controllers and machine tools, arranged so that it can handle different family of parts for which it has been designed and developed. [2]. With the lack of effective sensing abilities, many manufacturing units and assembly points cannot act intelligently in recognizing the workpieces and seeing workspace. Many of these systems have to work with predefined positions and orientations that are sent to the robot for manipulation. They however do not give room for change in those positions and orientation with which it becomes difficult for the robot to navigate to the new position. To cater for this anomaly, the robot system has to be equipped with an intelligent and flexible vision system that is able to communicate in real time with the robot to deal with imprecisely positioned objects and handle uncertainties and variations in the work environment.

This work's center of interest is to come up with a real-time sensor-based control system to applications where vital changes to varying objects is a necessity. [1] and [3], say that visual data received from the camera sensor is used for closed-loop robot control, normally known as visual servoing system. A review of the operation, characteristics and difficulties of visual servoing systems are found in [4] and [5]. In this work, object information is extracted from the acquired frame in a 2D format and then transformed to 3D pose with position of the object and its orientation for control of the robot. The control task is performed in 3D cartesian space thus the model of the camera required is that for mapping the data from 2D to 3D space. For one to be able to set up a visual servoing system, understanding of various aspects like robot modelling (dynamics and kinematics), control theory, computer vision functions including camera calibration, image processing, sensor system integration and so on is required. [6], [7], [8].

The other important capability of the vision information is enhancing the robot's ability to continuously update its field of view. This configuration is normally called eye-to-hand or eye-in-hand configuration. The architecture of the whole system is shown in figure 1 below:

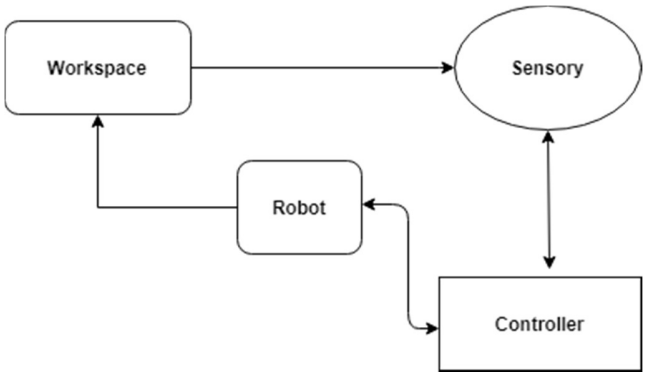

Figure 1 System of Robotic vision-based control

The diagram can be explained as follows

Workspace: includes fixtures, workpieces and tools.

Sensory system: is the vision or eyes of the robot to allow it to perceive its environment and to recognize the objects.

Control system: houses the system computer software and robot controller to organize tasks and the robot respectively.

Robot manipulator: does the action under the control of the robot controller.

With the automotive industry being traditionally the driving force and the largest consumer of automatic robot manipulators, it can be projected that as the robots are useful in the automotive industry, so shall they be for businesses with lower production capacity [9]. However, for these companies, it would cost them a lot of money in trying to automate their systems and having skilled labor for programming. Increase in production patterns will result in having cost of programming being higher than that of investment [10].

The pick and place operation in this study demonstrates how a robot is able to track, detect. and grasp an object in its field of view. The task here is to grasp a colored object placed on a table and for the task to be completed, a vision system detects the colored object by using color segmentation algorithms in real time. The object is then moved to a designated place by the robot manipulator. All this is done using communication protocols provided by TCP/IP. The image processing algorithm, the coordinate transformations, navigation algorithms and robot arm control are discussed. The validation of the system is discussed and results presented.

2. METHODOLOGY

2.1 Camera Calibration

The Kinect sensor used in this project was calibrated to bring about accuracy and effectiveness of it in the purpose for which it is being used. As is clearly highlighted in the introduction, many methods can be used to calculate the calibration parameters of the Kinect. The manufacturer's distortion parameters are not accurate so the sensor must be calibrated to correct that distortion. J.R Terven's method is used in this project to calibrate the Kinect camera. Kinect camera has an RGB camera and a depth camera which are both simultaneously calibrated using Terven's method. Terven designed a toolbox for Kinect V2 calibration in MATLAB called Kin2 Toolbox for MATLAB. The toolbox comprises of classes and functions that embeds the Microsoft Kinect 2 SDK. MATLAB is a high performance, versatile and powerful programming language developed by Mathworks which amalgamates computation and visualization to solve problems in a mathematical manner [11]. Largely based on C++ through MATLAB executable files, the version used for this project contains two classes and 30 functions put in different features such as coordinate mapping,

skeleton tracking, 3D reconstruction and face gestures recognition. [11].

2.2 Experimental setup

Figure 2 below shows the experimental setup that was used in the project.

Figure 2 Schematic arrangement of work cell

The above figure is hereby explained. The Kinect camera sensor is placed above the workspace of the robot or the work cell. From there it is able to take images of objects which are placed on the work table or a conveyer. After an image is taken, object detection algorithm starts by taking a snapshot and analyzing it using MATLAB's toolboxes for computer vision and image processing. For every frame, color segmentation and region properties techniques are employed. This results in giving the coordinates of the centroid of the detected object (x, y). However, Z axis is considered to be known from the camera frame since the camera is fixed on one place vertically. That distance is measured and remains the same throughout the course of the experiment.

It should be known that in this project, the transformation matrix was not used to transform camera frame coordinates to robot frame coordinates. However, in light of this, the coordinates that we find from the camera frame are referenced to work object that is created in the robot workspace. Work object reference point is therefore used to map the found coordinates so that it makes the robot's base settle on the work object reference point. The coordinates detected in the workspace of the robot will not be referenced from the base of the robot but from the work object reference coordinate system that has been created.

The coordinates of the colored object detected are then sent to the real robot from MATLAB via TCP/IP communication. Once the robot receives the coordinates in its sockets, the robot moves to the detected object, pick it up and place it to another place in real time. A while loop was created both in the rapid program of the robot so that the robot continually listens if there is any input in its sockets. The socket continually listens and at the same time coordinates are being sent from MATLAB. The cycle continues like that as the camera keeps on taking images of the workspace and processing them to get coordinates, sends them to the robot via TCP/IP in real time and the robot continually keeps looking to receive coordinates of where it must go.

2.3 Object Detection

After the Kinect camera is calibrated and distortion corrected, it means that the camera can now acquire accurate images which can properly be processed in MATLAB without distortion. Object detection follows calibration of the camera sensor. MATLAB has toolboxes that are used in computer vision. Most popular toolboxes used in this work are Image processing toolbox and Image acquisition toolbox.

Object detection starts with image acquisition and processing before we can conclude about position and orientation of the object. Image acquisition toolbox has functions that allow cameras of different types and properties to be used in MATLAB for acquiring images. Kinect that is used in this work is compatible to be used in MATLAB using 'videoinput' function. After an image is taken and a snapshot is taken then image processing starts.

2.4 Image Processing

Image processing toolbox brings in quite a number of algorithms which can be used to find properties of the detected object. When a snapshot of the object is taken, that image undergoes a series of algorithms as shown in Figure 3, to process it so that we can get what we want. Properties like area, centroid, minor axis, major axis, bounding box and so forth can be determined using 'regionprops' property in MATLAB. In this study, however, color segmentation was used to detect objects.

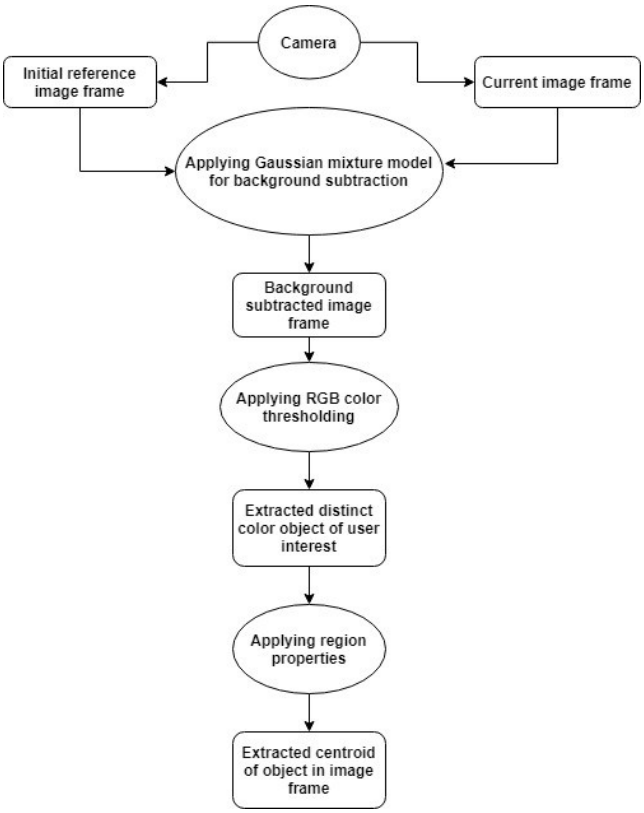

Figure 3 Flow chart involved in feature extraction

2.4.1 RegionProps and Centroid of the Object

Regionprops is a function that is very useful in image processing. It gives quite a number of properties depending on what you want to achieve. It is able to calculate 22 different properties but for this project it only focuses on three, including centroid. Centroid (x, y) is horizontal coordinate of the center of mass of the object and is represented by x-coordinate. The vertical component of the center of mass is the y-coordinate [12]. For this study, the centroid is a very important property since it is key in determining the exact location of the object. The centroid is calculated by the following equation;

$$\bar{x} = \frac{1}{n}\sum_{i-1}^{n} x_i, \;\; \bar{y} = \frac{1}{n}\sum_{i-1}^{n} y_i \qquad (1)$$

where $\bar{x}, \bar{y}$ denote location of the centroid of the target. (x_i, y_i), i= (1,2…n) denote boundary points of the object in the x and y directions.

2.5 Getting Depth from Centroid

The Kinect toolbox of J. R. Terven was used for calibration of the Kinect camera. External and internal calibration parameters for depth and color camera were found. Compared to the manufacturers' calibration parameters, this method yielded almost the same results as the manufacturers. Calibration parameters play a major role in determining the conversion to practical distance. After capturing the depth, the object must have a distance in millimeters from the sensor. This distance is found in the OpenNI, depth (X_d, Y_d) function format. In the Kin2 toolbox for Kinect, JR Terven contains this OpenNI function, which contains the depth coordinates corresponding to the centroid of the object in the RGB frame. The coordinates used at the depth of function depth (X_d, Y_d) to find the depth or distance of this object in the sensor in millimeters. This is the Z coordinate from the camera.

2.6 TCP/IP Communication

Connection is established between a computer and robot. The robot is controlled by a programming language called RAPID in a graphical user interface called RobotStudio for ABB robots. The connection is between a computer in which MATLAB is running and the robot controller which drives the robot as shown in Figure 4 below. TCP/IP is the best choice used for this project because the robotic program RAPID used for ABB robots facilitates better support for socket communication than for any other communication protocol. The other advantage that it brings is that the controller of the ABB robot has an option for Local Area Network (LAN).

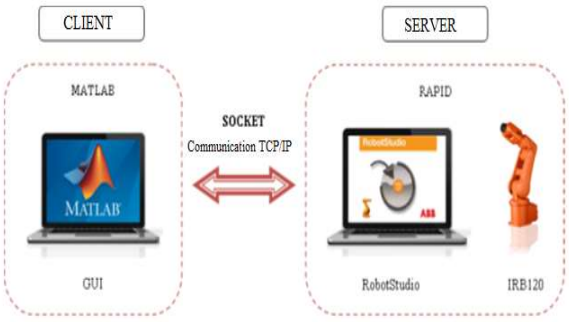

Figure 4 Relationship between client and server [13]

3. RESULTS

3.1 Experimental Setup

As shown in Figure 5 below, a cylindrical object of diameter 50mm and height of 45mm was used. It was placed on a plane surface perpendicular to the camera with zero elevation. Depth readings were taken at different distances from the camera. Depth measurement is the main parameter that we need to make sure that the robot moves to the correct height of a detected object. The actual distances were measured using a measuring tape.

The Kinect depth distance measured in this project was compared against the manufacturer's Microsoft Kinect for Windows SDK which is believed to give accurate measurements of depth and every other parameter. The SDK measurement was compared with two other measurements, the actual distance and the MATLAB program designed for the project, which in this case shall be called Kinect (project).

Figure 5 Experimental setup

3.2 Results analysis

Measurements were made with the Kinect sensor and compared with the SDK which is known to give accurate measurements. Of all the measurements taken, measurements from the Kinect were close to the measurement of the SDK. The graph of the recorded measurement was compared in Figure 6 below:

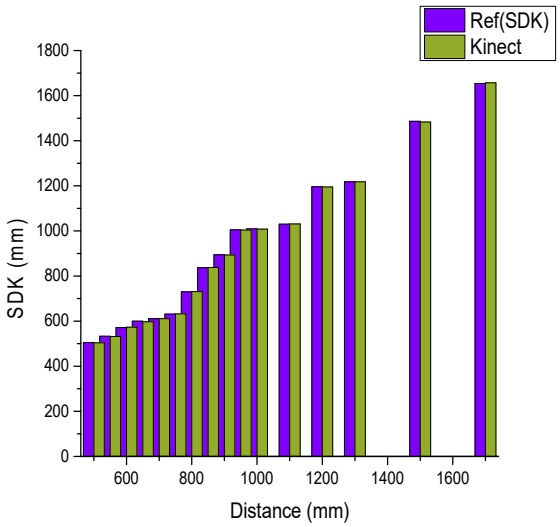

Figure 6 SDK and Kinect (Project) depth measurements

There was an average of 1.3mm of all the 16 measurements between the Kinect and the SDK. Figure 7 below shows the average differences between the Kinect (Project) and SDK.

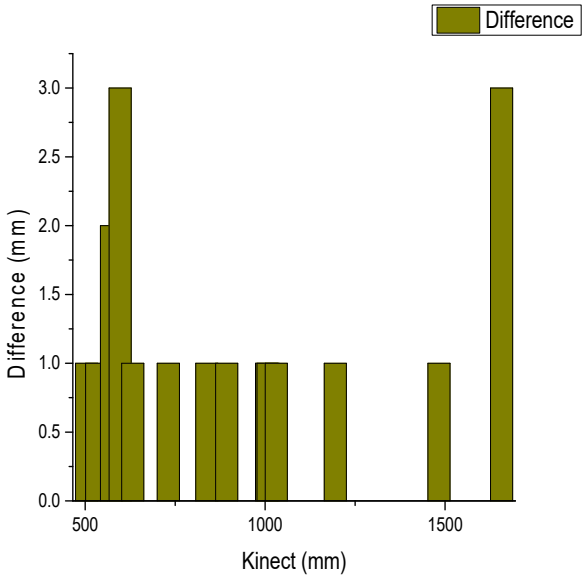

Figure 7 Average difference between Kinect and SDK

Absolute mean percentage error (AMPE), was calculated using the following formula;

$$Difference(mm) = \left(\begin{matrix} SDK \\ Depth \\ Measurement \end{matrix}\right) - \left(\begin{matrix} Kinect \\ Dept \\ Measurement \end{matrix}\right) \quad (2)$$

$$AMPE(\%) = \left(\frac{|difference|}{\begin{matrix} SDK \\ measurement \end{matrix}}\right) \text{x} 100\% \quad (3)$$

The graph of the distance measured by Kinect against AMPE is shown in Figure 8 below:

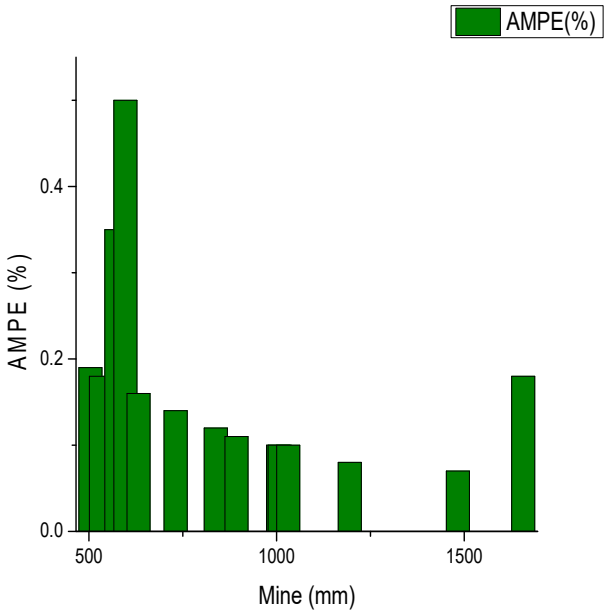

Figure 8 Distance measured by Kinect Vs AMPE

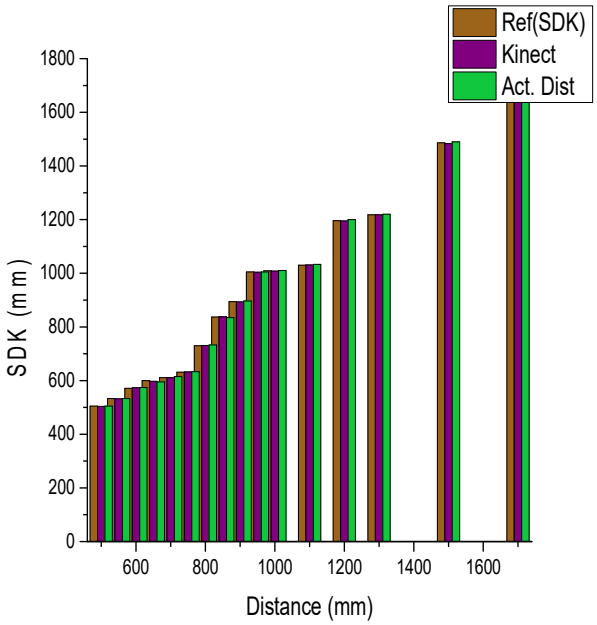

Figure 9 Comparison between SDK, Kinect and actual measured distance

The absolute mean percentage error of the measurements between the SDK and the Kinect measured depth in our project was 0.15%. This is almost a very insignificant error signifying that the code developed for this project was very accurate to SDK standard.

The difference between the SDK, Kinect (Project) and the actual distance was also analyzed. The graph in Figure 9 shows the measurement taken between the three. It can be seen from the graph that the measurement of these three gives almost the same measurement. We can therefore conclude that the actual measurement can also be used as equal to depth measurement because they are almost the same.

The average difference between the SDK and the actual measurement is just 2.8 mm. This turns the AMPE for the actual measurement with SDK to be 0.3%.

3.3 Conclusion

The algorithm for finding 3D object position according to color segmentation in real time was developed. The main focus was on how to find the depth of an object from the Kinect sensor. The height of the object was found by subtracting the distance from the depth of the plane where the objects were located to the sensor. This height was given to the robot and the robot was successfully moved to the location of the object, picked up and then placed in a specified location in real time. The average time taken to detect, select, and place an object was 8 seconds. Kinect was able to correctly distinguish color, and the robot could accurately navigate to the detected object. The algorithm developed was successful.

Research and Publication Ethics

This paper has been prepared within the scope of international research and publication ethics.

Ethics Committee Approval

This paper does not require any ethics committee permission or special permission.

Conflict of Interests

The authors declared no conflict of interest.

Author Contributions

Conception:TJM-BB, Design:TJM-KE, Supervision:BB-KE-TJM, Materials:BB, Data Collection and/or Processing:TJM, Analysis and/or Interpretation:TJM, Literature Review:TJM, Writer:TJM, Critical Review:BB-KE-TJM

REFERENCES

[1] J. Hill and W. Park, "Real time control of a robot with a mobile camera," 9th International Symposium on Industrial Robots, pp. 233–246, 1979.

[2] K. Rezaie, S. Nazari Shirkouhi, and S.M. Alem, "Evaluating and selecting flexible manufacturing systems by integrating data envelopment analysis and analytical hierarchy process model," Asia International Conference on Modelling and Simulation, pp. 460–464, 2009.

[3] K. Hashimoto, "Visual Serving: Real Time Control of Robot Manipulators Based on Visual Sensory Feedback," 1993.

[4] Hutchinson, F. and Chaumette, S., "Visual servo control basic approaches," Robotics & Automation Magazine, IEEE, vol. 13, pp. 82-90, 2006.

[5] F. C. S. Hutchinson, "Visual servo control. ii. advanced approaches [tutorial]," Robotics & Automation Magazine, IEEE, vol. 14, pp. 109–118, 2007.

[6] D. Kragic, and H. I. Christensen, "Survey on visual servoing for manipulation," Computational Vision and Active Perception Laboratory Fiskartorpsv, vol. 15, 2002.

[7] H. Wu, W. Tizzano, T. Andersen, N. Andersen, and O. Ravn, "Hand-Eye Calibration and Inverse Kinematics of Robot Arm using Neural Network," Springer, pp. 581–591, 2013.

[8] H. Wu, L. Lu, C.-C. Chen, S. Hirche, and K. Khnlenz, "Cloud-based networked visual servo control," I E E E Transactions on Industrial Electronics, vol. 60, no 2, pp. 554 – 566,, 2013.

[9] Meyer, R. D. and Schraft, C., "The need for an intuitive teaching method for small and medium enterprises," ISR Robotik, Germany, 2012.

[10] B. Akan, "Human Robot Interaction Solutions for Intuitive Industrial Robot Programming," Västerås: Mälardalen University, 2012.

[11] Juan R. Terven and Diana M. Cordova, Kin2 User Guide, 2016.

[12] MathWorks, "Image Processig Toolbox User's Guide," 2014.

[13] N. B. Fernández, "“Generación de trayectorias y evitación de obstáculos para el robot IRB120 en entorno Matlab”," UNIVERSIDAD DE ALCALÁ, pp. 47, 2015.