Design of an object sorting system using a vision-guided robotic arm

Zaid A. El-Shair
Department of Electrical and Computer Engineering
University of Michigan - Dearborn
Dearborn, Michigan 48128
Email: zelshair@umich.edu

Samir A. Rawashdeh
Department of Electrical and Computer Engineering
University of Michigan - Dearborn
Dearborn, Michigan 48128
Email: srawa@umich.edu

Abstract

High precision arms for manufacturing use are typically large, massive, and expensive. Small low-cost arms made from budget servo motors suffer from precision and accuracy problems. This paper presents a visual feedback system along with a simple kinematic approach to control and allow a small robotic arm to detect and identify different objects when present in the field of view, as well as being grasp-guided by visual feedback. The real-time vision system provides visual feedback for the arm to compensate for the low precision and accuracy of budget servo motors of the robotic arm. The feedback ensures that the gripper of the arm is aligned with the object involved. This project was undertaken as a senior capstone design project.

I. Introduction

Robotic arms play different roles in several domains such as manufacture, humanoid robots, mobile robots and others requiring some form of persistent feedback to ensure correct functionality. In terms of object grasping and manipulation, providing the robot with vision is quite imperative for proper operation, as it offers improved knowledge of its surroundings[1]. Similarly, in this project, vision plays an important role in providing visual feedback for the prototype designed in this senior capstone design project.

The main objective of this project was to design a stationary, object sorting, "pick-and-place" robotic arm that relies on a vision system for control and guidance. The vision system would be able to identify a simple object used and communicate with the robotic arm informing it of the exact type of the object, then providing it with visual guidance and feedback to ensure proper alignment with the object. Afterwards, the arm would be able to pick and place the object in the predetermined position for its specific shape, then wait for the detection of another object and reperform the same operation in an automated way. The focus of this project was to design a prototype system that efficiently controls the robotic arm in a well-timed manner and provide correct object detection and identification using image processing provided by the vision system. Section II goes more into detail on the overall design of this system, while section III further emphasizes the design of the vision system.

Servo motors play a major role in robotic arms where they are usually used to control their movement, which was the case for the arm used in this design. Although servo motors are

considered a closed-loop system[2], however, the feedback it provides is only internal to the servo motor itself. In other words, no information regarding the current exact position is provided to the microcontroller when moving the arm. The microcontroller would consider every servo motor to be at the exact position as the latest position command it issued. However, servo motors usually suffer from some movement inaccuracies, whether it was due to backlash or for using poorly tuned low-cost motors[3]. This would cause the intended use case to fail, especially in the application of being able to grasp an object, such as in this project. Using a simple kinematic approach, a design is introduced in this paper that provides visual feedback to the arm's movement to ensure successful object grasping and manipulation, by the robotic arm, when required.

II. System Description

This project was a real-time system with soft timing requirements, incorporating 2 main modules:

A. Vision System

The vision system consists of 2 components. The first one being an inexpensive webcam, to provide continuous feed of images of a top-down view of the area of operation, as illustrated in figure 1. The other component is a computer running MATLAB image processing toolbox[4] software, which provides all the required visual feedback. This toolbox was chosen due to the sheer volume of interactive tutorials, therefore, making it ideal for students with limited to no background in image processing, serving as an appropriate introduction to this field.

The top-down view allows for an easier development of image processing algorithms. Therefore, simple 3D-printed objects were used to allow a more convenient recognition, provided with a cross-sectional image of each.

Finally, the vision system is also in charge of providing visual feedback to the arm when aligning with the object. Section III goes more in depth with the various image processing techniques implemented.

The vision system is able to synchronize its operation with the robotic arm module using a universal asynchronous receiver-transmitter (UART) serial connection[5], providing a point-to-point connection while sending 8 bits of data at a time in half-duplex mode (taking turns in transmitting and receiving). The images were taken at a resolution of 640x480 to better support real-time processing.

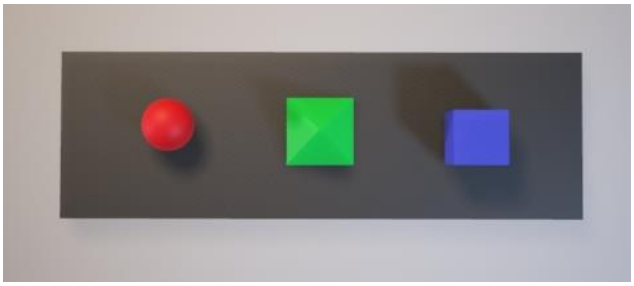

Figure 1. Webcam's top-down view illustration

B. Robotic Arm Module:

The other main subsystem of this project is the robotic arm module. This module consists of several components to achieve the functionality desired. The main component being BotBoarduino microcontroller developed by Lynxmotion, which is an Arduino based microcontroller that simplifies the control of the robotic arm using its open-source Integrated Development Environment (IDE) software. This microcontroller was specifically chosen due to it having several servo motors output pins that uses a separate power input.

Furthermore, an inexpensive pre-built 6 degrees of freedom (DOF) robotic arm was used. This robotic arm includes 7 low-cost servo motors which are used to control the arm's movements and actuate the gripper, controlling each in a sequential manner. Servo motors are controlled using pulse width modulation (PWM) signals generated by the microcontroller. Moreover, simple kinematics were implemented to initialize the arm to a fixed starting location then from there continue to the intended pick up position right under the camera, as shown in figure 2.

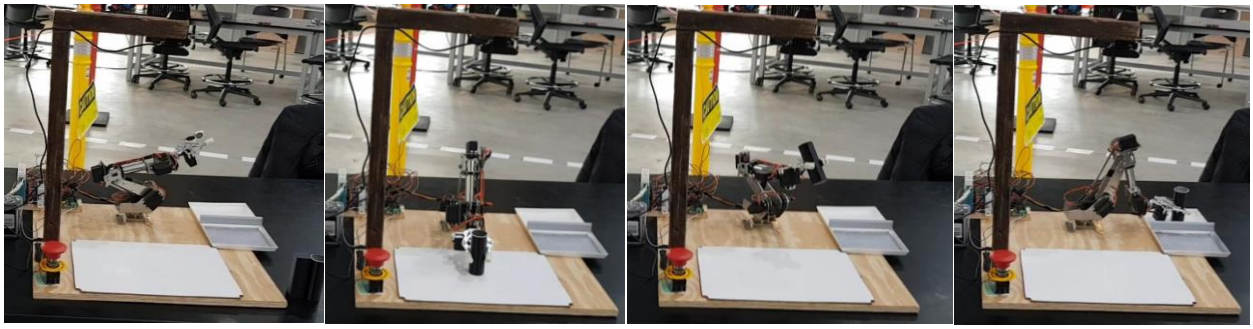

Figure 2. Full system operation from starting position (left) to placing an object (right)

Other components include a force-sensitive resistor placed on one of the ends of the gripper, where it applies resistance depending on how much pressure is being applied at the sensing area. It is used to ensure that the object is held firm by the gripper and that no excessive force is applied. Thus, once the analog voltage reading (after getting converted to a digital value) reaches a set threshold, then the servo motor controlling the gripper would suspend its actuation and hold the gripper's position firmly. This is achieved by having the servo motor controlling the gripper apply a constant stall current. The same stall current is also applied at the other servo motors to hold the robotic arm in a fixed position.

Safety issues were realized as in having a robotic arm misbehaving or functioning abnormally. However, this was handled by implementing an emergency stop button in case the arm behaves in an unforeseen way. This emergency stop button was placed at an easy to reach location in case something goes wrong, where it disconnects the power supplied to the microcontroller when pressed, therefore, suspending the system to ensure a safe operation.

The full prototype developed, including all its components, is demonstrated in figure 3.

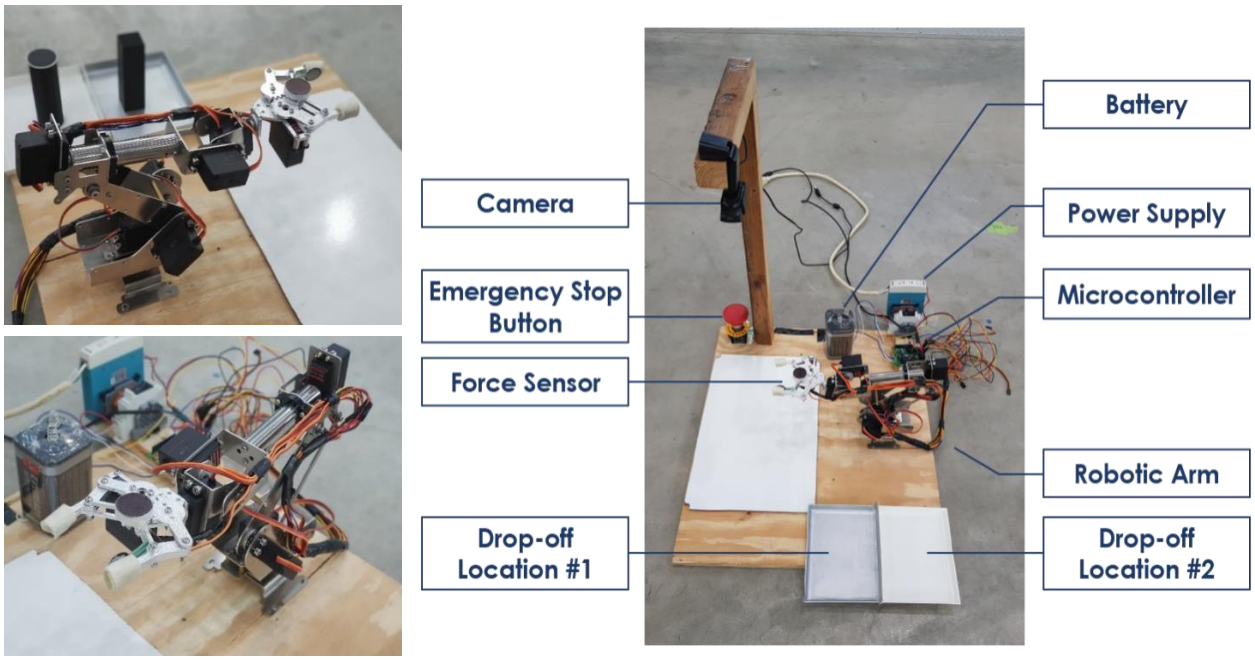

Figure 3. Prototype designed showing all the components included

Moreover, a system block diagram is shown in figure 4 which demonstrates the basic interactions and interfaces between the system's modules, while figure 5 provides a flow chart representation of the full system's operation.

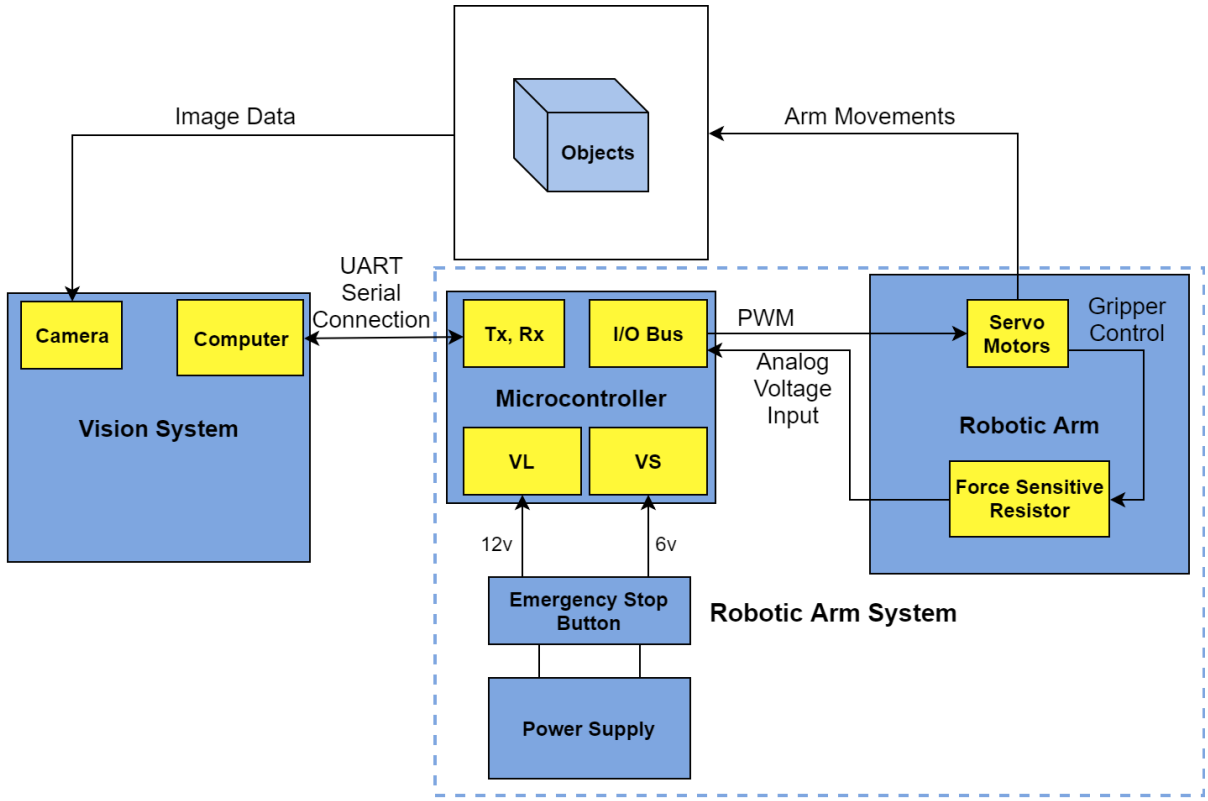

Figure 4. System Block Diagram

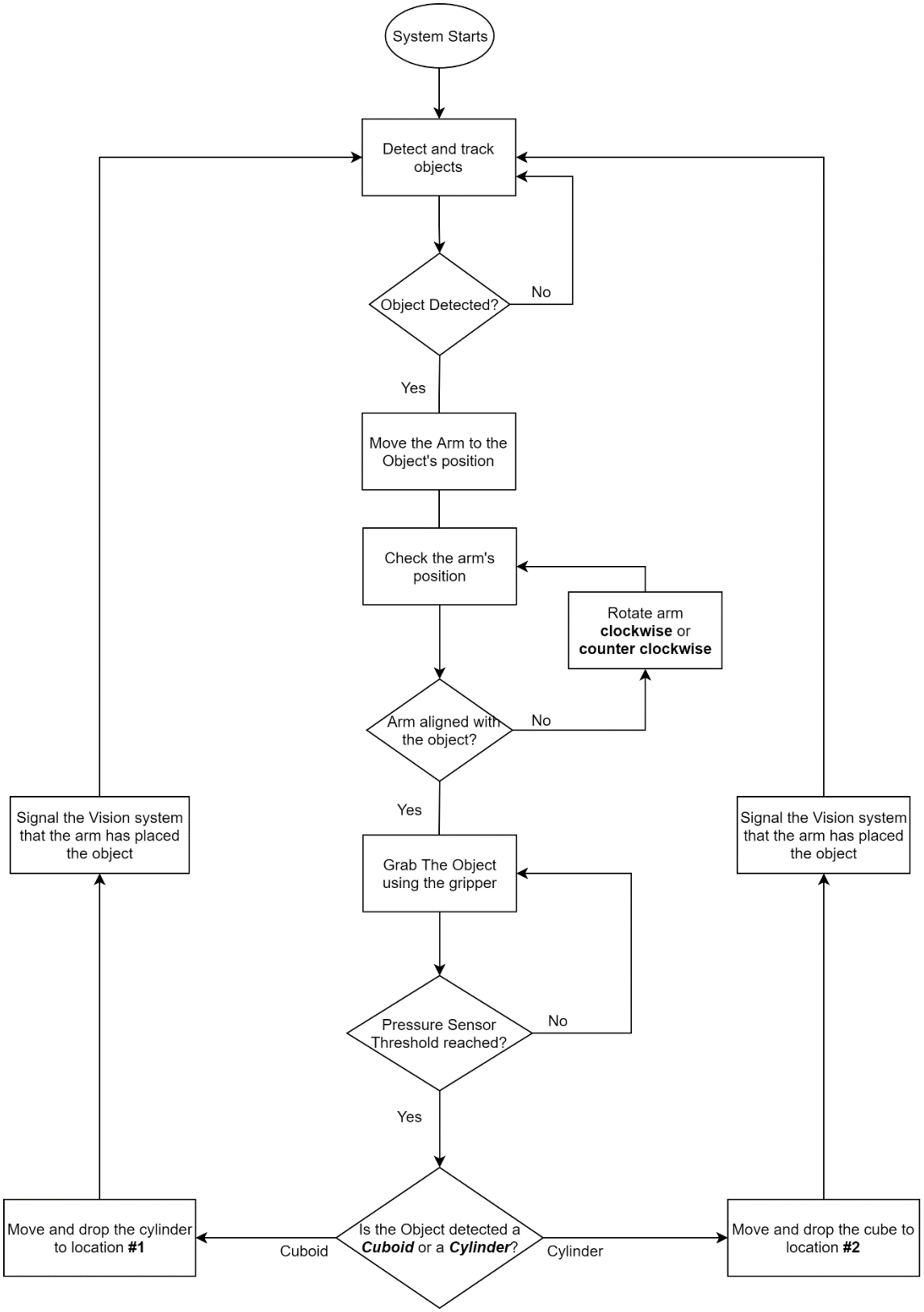

Figure 5. Flow chart demonstrating the full system's operation

III. Vision System Implementation

The system was designed to operate with simple objects in mind, mainly a cuboid and a cylinder. These 2 models were selected due to their basic 2d-shaped top-view of a square and a circle for the cuboid and the cylinder, respectively.

The vision system relies heavily on the software designed, where the images processed are constantly supplied by the camera. As shown earlier, the camera is mounted at a height of approximately 55 cm providing a top-down view to provide 2-dimensional images of the objects as shown in figure 6. Moreover, a white board was used to provide a clear background to provide high contrast that enables proper image filtering, as well as a more consistent correct object identification process.

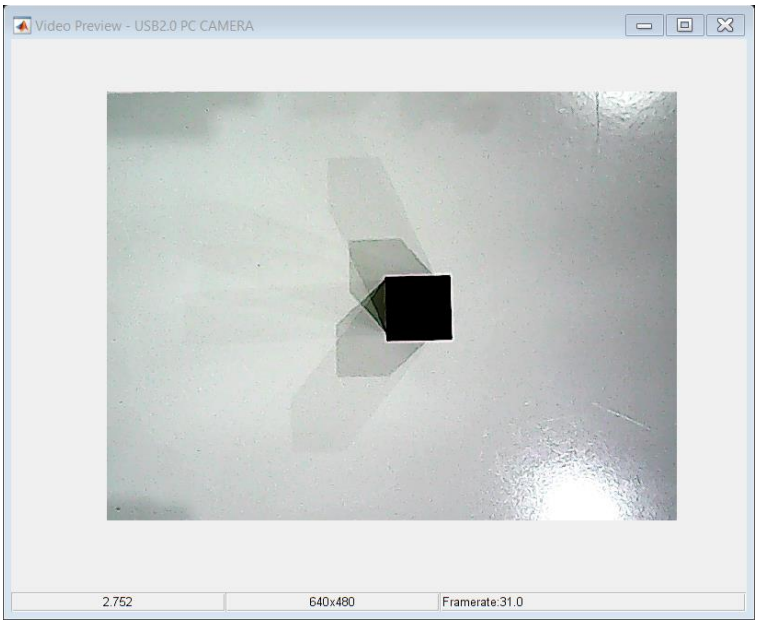

Figure 6. Sample top-view image of a cuboid captured by the camera

The operation of the vision system is divided into 3 sequential stages:

A. Object detection:

The first step of the vision system is to constantly scan the field of view to detect if an object is present. This was implemented by using corner detection technique. Corners are important features detected in an image that are the points that have high curvature and are existing in intersecting regions of different brightnesses[6].

Initially, 5 images are consecutively taken by the camera, each of which gets converted to a binary image by the image processing software, using a set threshold of a value between 0 and 1. Therefore, filtering out the images taken from any shadows or unwanted details to get a clear and a better representation of the field of view. The system then detects all the corners of the resulting binary images. Using a set threshold number, if the total number of

corners exceeds that, then the system has detected an object. Otherwise, there was none. Therefore, determining whether an object was placed or not. Within these 5 images processed at a time, 4 of which (80%) must have indicated the detection of an object to confirm the existence of one, improving its accuracy.

In order to set the appropriate binary conversion threshold, the total number of corners detected was tested for both good and bad lighting conditions while having no objects placed, using different binary threshold levels between 0 and 1. The point behind this was to get the lowest number of corners detected when no object is present in the field of view, while not losing important information in the image captured when an object is actually available. The results were demonstrated in figure 7.

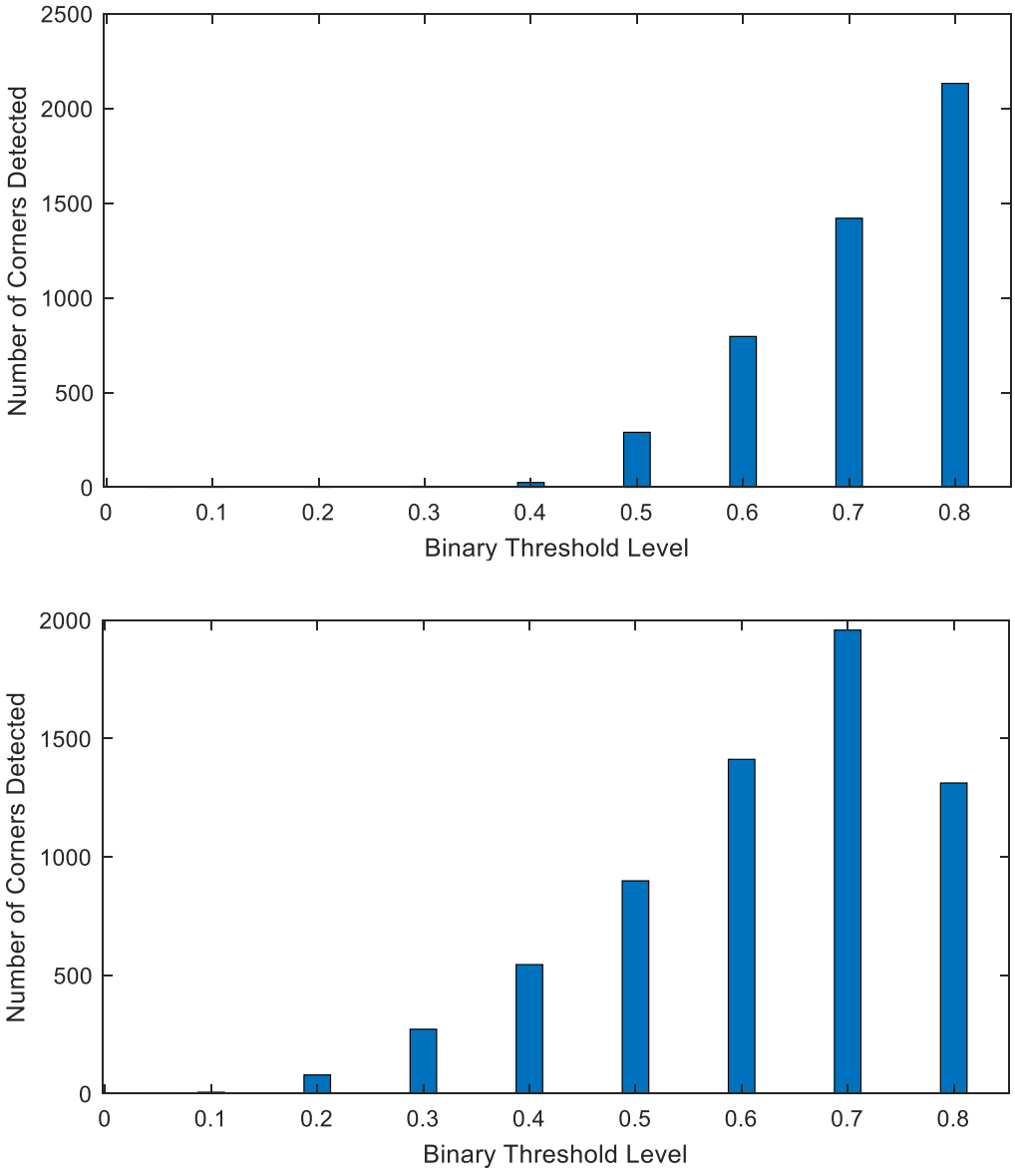

Figure 7. Good (top) vs bad (bottom) lighting conditions test results

In good lighting conditions, the vision system detected less than 10 corners when the binary threshold levels used were less than 0.4. However, in the second test, under bad lighting conditions, the system detected 3 corners with a threshold level set as 0.05, while 6 corners were detected with a threshold level of 0.1, then increased dramatically to 79 corners when raising the threshold level to 0.2. Therefore, the binary conversion threshold was set as 0.05 to ensure that less than 10 corners are detected even in bad lighting conditions when having no objects placed. Accordingly, the corner detection count threshold was set to 10, to signify that no object is available if a smaller number of corners were detected. RGB to Binary image conversion was done using a MATLAB function called *imbinarize( ),* while corner detection was done using the function *detectHarrisFeatures( ),* which basically detects corners using Harris–Stephens algorithm[7].

Using these threshold levels, the output of the object detection is demonstrated in figure 8 below.

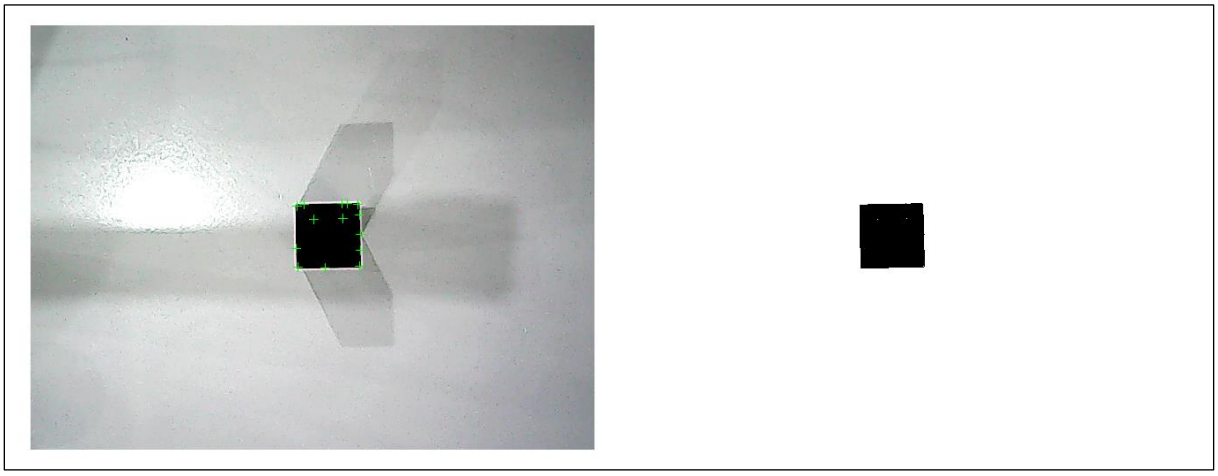

Figure 8. Object detection showing the corners detected (left) using the binary image generated (right)

B. Object Identification:

After the vision system has confirmed that an object was indeed detected, it needs to properly identify its shape. As mentioned earlier, this system was designed to recognize 2 different simple-shaped objects; a cuboid and a cylinder each with a top view of a square and a circle, respectively.

Initially, the software designed tries to check if a cylinder exists. Therefore, the code will first try to detect any circles found in the images taken. Similar to the object detection stage, it would take 5 new images as well and check each for circles of radius between 20 and 60 pixels, using the MATLAB function *imfindcircles( )* that is based on Circular Hough Transform[8,9]. If a circle was detected in at least 4 (80%) of the images taken, then

the object detected is considered to be a cylinder. A sample output of a successful circle detection is demonstrated in figure 9.

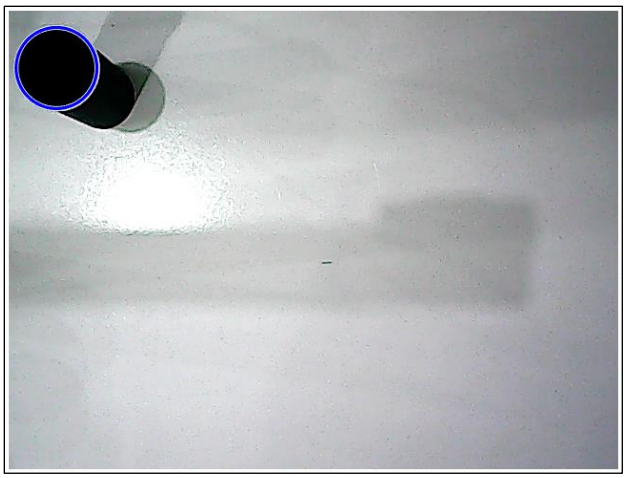

Figure 9. Circle detection output for a captured image of a cylinder

If the vision system didn't detect a cylinder, it will then try to check if the object is a cuboid. Similarly, the camera would capture 5 top-view images while checking each for a square. To determine if a square exists in an image, a simple algorithm was designed. This algorithm first converts the images taken to binary using a different binary conversion threshold, then finds the edges in each using a combination of several MATLAB functions including *edge( )*, *hough( )* and *houghpeaks( )*. Afterwards, it uses Hough Transform to detect lines longer than or equal to the minimum length set of 50 pixels (set after tuning the system with the 3d printed cuboid), using the function *houghlines( )*. Hence, a square would require 4 lines to be detected representing the 4 sides of a typical square. Therefore, if a square was detected in at least 4 (80%) of the images taken, then the object detected is a cuboid. Else, the system has detected an unknown object and would take no action. A sample output of a successful square detection process is shown in figure 10.

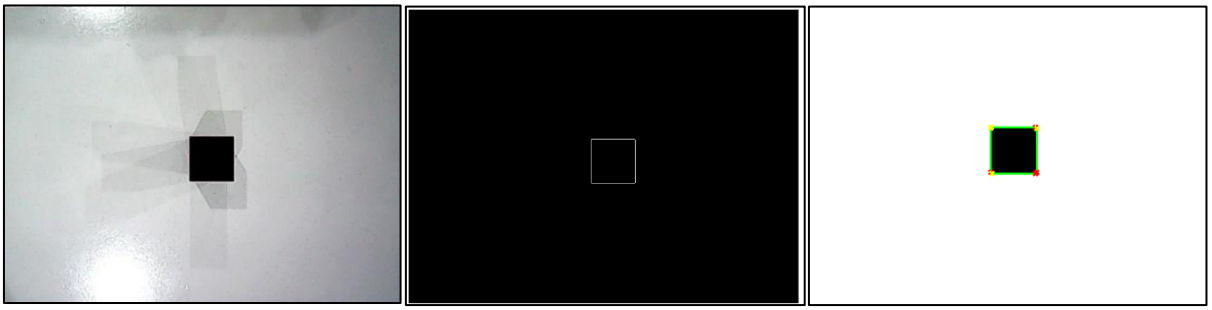

Figure 10. Square detection process for an image of a cuboid with the raw image (left), edges (center) and lines (right) detected

C. Visual Feedback:

Just before the robotic arm is about to pick up an object successfully detected and identified, the vision system is required to provide visual feedback to be able to ensure its alignment with the object. This is due to the fact that servo motors are not persistently accurate and provide no feedback, as emphasized earlier in this paper. The visual feedback process would initiate when the arm is roughly in position to pick the object detected within the camera's field of view. It is guaranteed that arm would at least reach that position before grabbing the object from the fixed pick-up spot, due to servo motors being accurate to ±10 degrees, however, a lesser range of error is required for successful object grasping.

To incorporate a simple feedback mechanism, a bottle cap painted black, was used to act as a marker attached at top of the gripper at its center. Thus, the position of the arm would be accurately detected using the same algorithm used to detect circles that was described earlier.

In order to ensure proper alignment of the arm with the object, the vision system compares the marker's center x-coordinate with a predefined acceptable range set for the pickup point of the objects, which are assumed to be placed within this range. Therefore, if the marker was not within that range, the arm would, under 1 DOF utilizing the robotic arm's base servo motor, either rotate clockwise or counter-clockwise until the center of the marker is within the acceptable range to be able to align with the object properly. This is demonstrated in figure 11, where the predefined range set is indicated by the 2 dotted blue lines, while the arm's range of motion is represented by the dashed green arc.

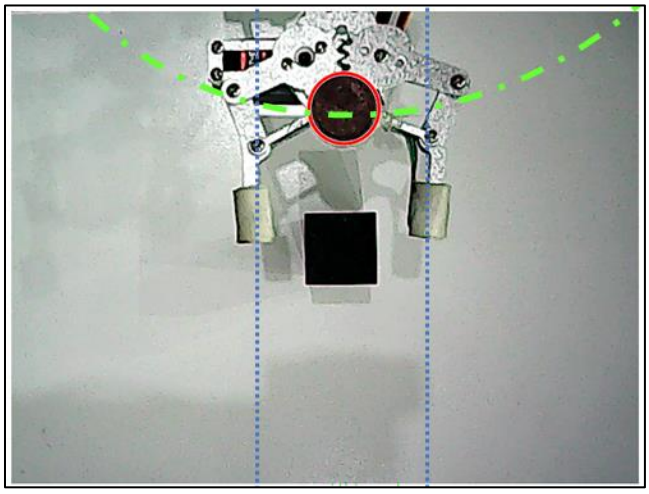

Figure 11. Visual feedback output of the arm's alignment with a cuboid, demonstrating the arm's range of motion (green) and the predefined object pick-up range (blue)

IV. Vision system testing and results

The prototype designed was tested to evaluate all aspects of the functional and performance requirements for the major subsystems, as well as the full system. To test the vision system,

multiple test runs were held under consistent good lighting conditions where the results of the object detection and identification were recorded. The performance of the system designed was evaluated using a confusion matrix utilizing these results.

A confusion matrix is a tool that can be used to evaluate the performance of a classification model of any type. It would consist of:

1. True positives: an object placed and correctly identified
2. True negatives: no object was placed, and no object was detected.
3. False positives: detecting an object when there was none, or an incorrect type was detected.
4. False negatives: not detecting an object when it an object was present.

The sample confusion matrix used is demonstrated below in table 1, with the light green cells denoting correct outputs, while the light blue cells denoting incorrect outputs.

Table 1. Confusion Matrix used in performance evaluation

Detection Output				
Actual	**Object**	Cuboid	Cylinder	No Object
	Cuboid	**X**	E(3)	E(5)
	Cylinder	E(1)	**Y**	E(6)
	No Object	E(2)	E(4)	**Z**

The diagonal represents the true values (correct value), while all the other values count the false results. Therefore, the accuracy of the system can be calculated by:

$$\frac{X+Y+Z}{(X+Y+Z)+\sum_{n=1}^{6} E(n)} \qquad (1)$$

Hence, the overall error rate can be calculated by:

$$\frac{\sum_{n=1}^{6} E(n)}{(X+Y+Z)+\sum_{n=1}^{6} E(n)} \qquad (2)$$

Consequently, the results of object detection and identification of 10 tests conducted were recorded and demonstrated in table 2, giving an overall accuracy rate of 100% with 0% error. However, these values resulted after tuning the design to accurately fit the controlled lab environment the system was tested in.

Table 2. Test Results Confusion Matrix for a total of 10 test runs

Detection Output				
Actual	**Object**	Cuboid	Cylinder	No Object
	Cuboid	**4**	0	0
	Cylinder	0	**4**	0
	No Object	0	0	**2**

V. Conclusion

This paper discusses the design of an object sorting system using a vision-guided robotic arm. Low-cost components, such as inexpensive servo motors and a basic webcam were used in the design of this system. Moreover, a simple kinematic approach was used to control the arm to provide pick-and-place functions. Several different image processing techniques were included in the design of the vision system to provide proper object detection and identification as well visual feedback required to ensure proper arm control under 1 DOF, to accommodate for the low precision and accuracy of the servo motors implemented.

Finally, a confusion matrix was used to demonstrate the effectiveness and accuracy of the system, by testing it several times with different inputs, resulting in a very accurate operation under controlled environmental conditions.

Bibliography

1. Rouhollahi, Ali, Mehdi Azmoun, and Mehdi Tale Masouleh. "Experimental study on the visual servoing of a 4-DOF parallel robot for pick-and-place purpose." In Fuzzy and Intelligent Systems (CFIS), 2018 6th Iranian Joint Congress on, pp. 27-30. IEEE, 2018.
2. McComb, Gordon. The robot builder's bonanza. McGraw-Hill, Inc., 2002.
3. Greenway, Bryan. "Robot accuracy." Industrial Robot: An International Journal 27, no. 4 (2000): 257-265.
4. MATLAB and Image Processing Toolbox Release 2017b, The MathWorks, Inc., Natick, Massachusetts, United States.
5. Adam Osborne, An Introduction to Microcomputers Volume 1: Basic Concepts, Osborne-McGraw Hill Berkeley California USA, 1980 ISBN 0-931988-34-9 pp. 116–126
6. Chen, Jie, Li-hui Zou, Juan Zhang, and Li-hua Dou. "The Comparison and Application of Corner Detection Algorithms." Journal of multimedia 4, no. 6 (2009).
7. Harris, Chris, and Mike Stephens. "A combined corner and edge detector." In Alvey vision conference, vol. 15, no. 50, pp. 10-5244. 1988.
8. Rizon, Mohamed, Haniza Yazid, Puteh Saad, Ali Yeon Md Shakaff, Abdul Rahman Saad, Masanori Sugisaka, Sazali Yaacob, M. Rozailan Mamat, and M. Karthigayan. "Object detection using circular Hough transform." (2005).
9. Pedersen, Simon Just Kjeldgaard. "Circular hough transform." Aalborg University, Vision, Graphics, and Interactive Systems 123 (2007): 123.