Robotic Freehand Drawing

by

Muhammad Nazreen bin Zainal Abidin

14548

Dissertation submitted in partial fulfilment of

the requirements for the Degree of study (Hons)

(Electrical and Electronics Engineering)

MAY 2015

Universiti Teknologi PETRONAS

Bandar Seri Iskandar

31750 Tronoh

Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

Robotic Freehand Drawing

by

Muhammad Nazreen bin Zainal Abidin

14548

A project dissertation submitted to the

Electrical and Electronics Engineering Programme

Universiti Teknologi PETRONAS

In partial fulfilment of the requirement for the

BACHELOR OF ENGINEERING (Hons)

(ELECTRICAL AND ELECTRONICS)

Approved by,

____

(Dr. Syed Saad Azhar Ali)

UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

May 2015

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements and that the original work contained herein have not been undertaken or done by unspecified sources of persons.

____

(Muhammad Nazreen bin Zainal Abidin)

ABSTRACT

The objective of this project is to mimic the human motions and use back that information to convert it into robotic motion. The motion selected in this project is drawing activities on top a paper. A robotic arm system was build that can perform image processing from images obtained from a webcam, and use that information to convert it into robotic motion. The human will perform the simple drawings and the image will be captured by the webcam. Using MATLAB interface with its supported hardware, the image will be processed in real time to get the appropriate pulse width modulation (PWM) signals to be fed to the servo motors that acts as a joint for the robotic arm. Various image processing were tested and the object tracking based on colour was used in this project. The robotic arm system was managed to perform linear drawing, but are having difficulties to perform curved drawings perfectly. It was also programmed to detect the differences of when the user makes a drawing or not. Regardless, the important features of the drawing was successfully copied using images from the webcam.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLE

CHAPTER 1: INTRODUCTION

1.1: Background

Technology of Art

Entertainment and art are closely connected in terms of giving pleasure and delight to people. People tend to search and find entertainment every day in the form of music, films, dance and dramas. Art is a product of human activities that involves technical and imaginative skills. Visual art such as painting, drawing and photography can entertain people either during the making of the art, or after the art has finished. For example, people who are observing a street artist who are drawing a simple portrait can enjoy the art from the process of sketching until the final touch-up and still enjoy it after it was finished.

As technology develops, now we can enjoy art in a whole different level. The art making done by humans, now can be done by robots by mimicking the human drawing made by humans. With this technology, people can enjoy the live drawing and painting done by artists from far away. It is expected in this project that the robotic arm will have the ability to observe the human drawing and mimic the motions performed by humans in the midst of drawing.

In this project, the main aim to be accomplished is to mimic the human actions, where it can copy human motions, and can perform it whether from far away, or during the absence of human. Drawing is one of the actions performed by human that is quite complicated and requires a lot of finesse in order to be performed perfectly. Thus, this type of human action is selected in this project as an approach to track, capture and copy the human movements. This robotic concept of copying motion is very applicable in many industries, especially manufacturing industries, which is hoped to be developed in future times.

A robotic arm designed for this purpose differs significantly from any industrial robot in this world. The robot must interact physically with a human being, thus it needs to be designed to be safe, with minimum possibilities to bring injuries. The robotic arm also requires live command from a person, in order to perform complicate tasks such as performing the same drawing that was made by human hands. Industrial robots usually only have specific instruction that has been built in the robot to be done repetitively.

Figure 1: Example of Arduino Microcontroller

In this project, there are only one type of microcontroller that will be used, which is the Arduino Mega2560. The Arduino Mega2560 will be used to control the robotic arm to perform the human drawing (refer to the Figure 1 for example of Arduino). MATLAB software will also be used in this project to process the human drawing images due to its abilities to perform image processing techniques.

1.2: Problem Statement

Nowadays, people would prefer to do things with simplicity. If they are too lazy to walk, they will take an elevator or escalator to go from one place to another. Machines and robots are widely used throughout this world to make human's life easy. Most of them requires instructions that are embedded and programmed into them to do the same things every day, every time. For example, a robot used in a factory to select defective product will select the same defective product every time it runs. If the robot was expected to select something else, then it needs to be programmed again. To overcome this problem, a robot that can accept command from human using visual feedback is introduced, with the selected human action which is drawing actions. By doing this, the robot can be operated and interact with human being without having the need to know how to program the robot. This project aims to copy the drawings made by human hands and transfer it via a network to the robotic hand so that people can enjoy the drawings from far away. When a subject draws on the screen, the drawings will be captured by the camera and a signal processing will be done to the image. This image will be analysed to get the coordinates and lines so that the robotic arm can follow the same stroke that the subject did in the drawing.

1.3: Objectives and Scope of Study

The objective of this project is to develop a robotic arm that can mimic human drawings. Second is to transfer the information (human drawing) via a network to the robotic arm.

Throughout the duration, both minor and major project's requisite that lead to the final conclusion has been identified as the objectives as listed below;

- To study the kinematics of a robotic arm.
- To perform image processing of freehand drawing video frames.
- To control the movement of the robotic arm by visual feedback.
- To perform freehand drawing using the robotic arm.

CHAPTER 2: LITERATURE REVIEW AND THEORY

2.1: Theory

Visual Feedback

Visual feedback is a method that will be used in this project which uses feedback in the form of signal and information that can be extracted from a vision sensor, which in this case is a camera. The information will be analysed using some image processing and this will control the motion of the robot. The camera will be fixed to a screen where a subject will draw simple drawings, and the robotic arm will perform the same drawing as the subject will do.

Image Processing

The image processing work in this project will be done using MATLAB software and its support hardware packages. This image processing will treat the image as a two-dimensional signal and some signal processing techniques will be applied to the image. The output of the image processed will provide coordinates and lines for every strokes made during the image drawing done by human.

Manipulating Kinematics of Robotic Arm

This section will explain briefly about the kinematics of a robotic arm. The main point of the robotic arm consists of a set of links, grippers, joints and base. The links in the robotic arm is a fixed part where it must be connected to other links using joints. The motion of the robotic arm starts at the joints between the links, which are controlled by servo motors or hydraulic actuators. The figure below is some types of manipulator kinematics of robotic arms:

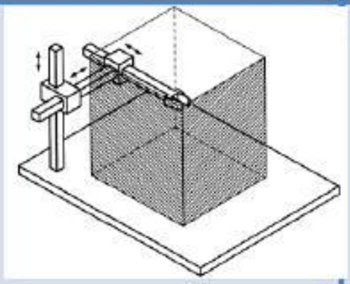
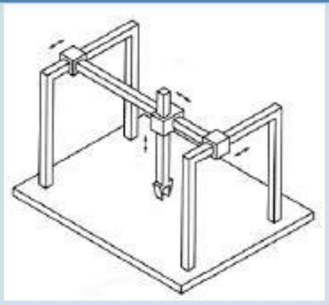
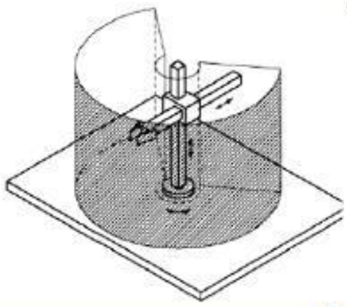
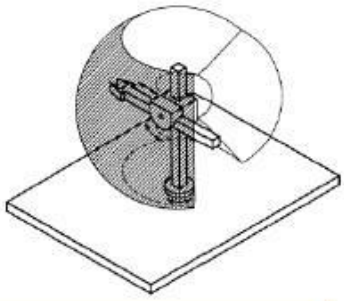
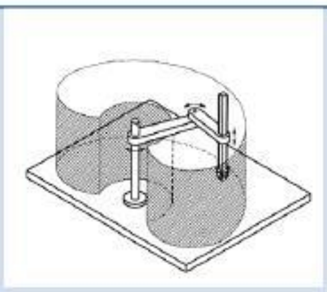
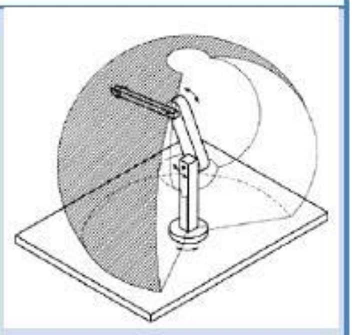
Name	Figure	Name	Figure
Cartesian		Gantry	
Cylindrical		Sphre	
Scara		Anthropomorhic	

Figure 2: Different Types of Manipulating Kinematics of Robotic Arm

2.2: Literature Review

In order to conduct this research, the author has reviewed a number of relevant literatures to get clear idea on this topic. The studies have shown that there are many types of robots used in entertainment field. For example, the movie industry used robot to illustrate dinosaurs in the movie Jurassic Park, and there are also some robots that was engineered to perform music of play musical instruments like humans do. Unlike these examples, drawing robots are hardly seen throughout this world [1]. Constructing a robotic arm that functions exactly like a human arm do is really difficult because the human limbs have a really high degree of freedom (DOF). Thus, to develop a robotic arm, it is expected that the designs of the links and joints are sufficient enough to perform drawing and writing tasks [2].

In order to capture the freehand drawings made by human, the usage of camera is needed. Any drawings or writings are consists of sequences of gestures and points. These gestures or strokes start when the pen is touching the screen or platform, and ends when it is lifted [3]. These strokes are also hard to be recorded because during writing and drawing, the human hands are caught inside the video and skin-tone segmentation technique is needed [1].

As for the robotic arm/hand, a few calculations are needed for the effectiveness of the drawings made by the robot which includes the following formulas:

$$T = F * L \tag{1}$$

Where T is the turning or twisting force, F is the force acted at a length and L is the length of the link from a pivot point.

$$F = m * g \tag{2}$$

Where F is the force acted, m is the mass and g is the gravitational force (9.81g/ms^2)

The next one is the image processing of the images captured by the camera. These images was processed under a few image processing techniques, such as the convolution, high pass filters, low pass filters, enhancements and noise filters in order to get the coordinates of the strokes of the hand drawing made by human. Basically, this image is a grayscale image, and is converted to binary image which denotes the grayscale intensities which denotes black=0 and white=1 [4].

Theory

A lot of mathematical models have been sought in order to find the best representative of the real system. In this project, there are three dynamic models that will be used. These mathematical models which are used widely throughout the world are, forward kinematics and inverse kinematics.

Forward Kinematics

In order to understand the three dynamic models, the important parameters of the robotic arm needs to be determined. The first joint which is fixed to the base of the robotic arm are denoted as J0, the second joint as J1, the third joint as J2, the length of the link from first joint to second joint as L0, the length of the link from the second joint to the third joint as L1, and the length of the link from the third joint to the end-effector as L2. The forward kinematics of the robotic arm is calculated as per the following Figure 3 [2]:

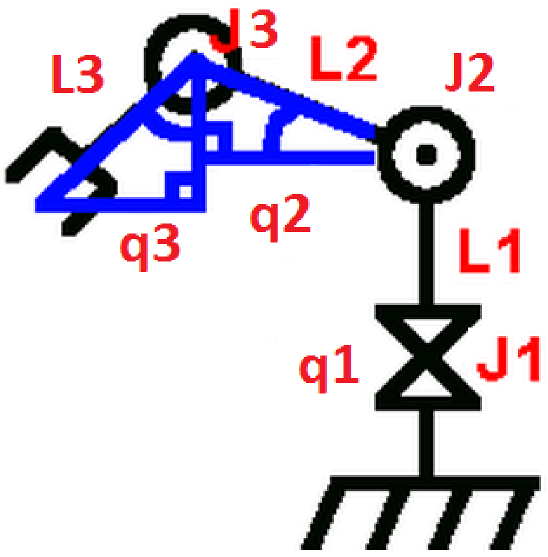


Figure 3: Defining parameters for robotic arm

Joint 0 (with x and y at base equalling 0):

$$x1 = 0 \tag{3}$$

$$y1 = L1 \tag{4}$$

Joint 1 (with x and y at J2 equaling 0):

$$\text{Cos } (q_2)\text{=x2/L2} \tag{5}$$

$$\text{Sin } (q_2)\text{=y2/L2} \tag{6}$$

Joint 2 (with x and y at J3 equaling 0):

$$\text{Sin } (q_3) = \text{x3/L3} \tag{7}$$

$$\text{Cos } (q_3) = \text{y3/L3} \tag{8}$$

End effector Location

$$\text{x1+x2+x3, or 0+L2*cos}(q_2)\text{+L3*sin}(q_3) \tag{9}$$

$$\text{y1+y2+y3, or 0+L2*sin }(q_2)\text{+L3*cos}(q_3) \tag{10}$$

Inverse Kinematics

Inverse kinematics is going to be used in this project in order to determine the joint angles of the robotic arm, given the position and orientation of the end effector. Inverse kinematics is a crucial factor in order to establish the PID-controller to be used to the robotic arm [2]. This is because the end-effector might encounter "wiggly" movements when starting and stopping the robotic arm that will cause errors during the drawing activities performed by the robotic arm [5]. The inverse kinematics helps to counter the errors by utilizing it as a feedback to the system.

In this project, the coordinates of the end-effector is the key to find the angle of the joints. By referring to the Figure 4 above, the calculations of the joint angles in reference to the end-effector and links between the joints are [5]:

Joint 2:

$$q_2 = \arccos\left(\frac{x^2+y^2-L_2{}^2-L_3{}^2}{2L_2L_3}\right) \tag{11}$$

Joint 3:

$$q_3 = \arcsin\left(\frac{y(L_2+L_3c_2)-xL_3s_2}{x^2+y^2}\right) \tag{12}$$

Where $c_2 = \left(\frac{x^2+y^2-L_2{}^2-L_3{}^2}{2L_2L_3}\right)$, and $s_2 = \sqrt{1-c_2{}^2}$

The calculation for joint 1 is simple, where the joint angle, q_1 is computed as per the following Figure 4:

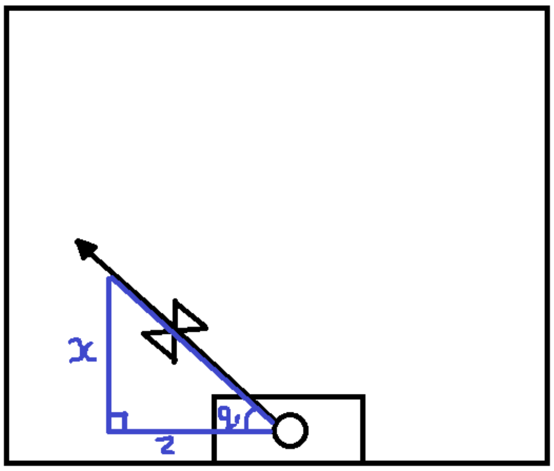


Figure 4: The robotic arm top view

$$q_1 = tan\left(\frac{x}{z}\right) \tag{13}$$

By using these equations, it is possible to determine how much every joint need to turn in degrees in order to go to the desired end effector position.

CHAPTER 3: METHODOLOGY/PROJECT WORK

3.1 Gantt Chart FYP

Below is the Gantt chart of this research for FYP 1:

	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13	Week 14
First meeting with coordinator and supervisor	■													
Problem Statement and analysis of image processing, video feedback and robot kinematics		■	■											
Preliminary research work to find the most optimal and easiest way to compute forward and inverse kinematics and literature review.	■	■	■	■	■	■								
Submission of extended proposal defense						■								
Finding the materials for fabrication of robotic arm				■	■	■	■	■	■	■				
Oral presentation (proposal defense)								■	■					
Fabrication of robotic arm together with the Raspberry Pi										■	■	■	■	■
Preparation of Interim Report												■		
Submission of Interim Draft Report													■	
Submission of Interim Final Report														■

Table 1: Gantt chart of Final Year Project 1

The following table shows the planned Gantt chart for FYP 2:

Activities	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13	Week 14
Designing and fabricating the robotic arm	■	■	■	■	■	■								
Coding and debugging of the Raspberry Pi							■	■	■	■	■	■	■	■
Submission of Progress Report								■						
Pre-Sedex										■				
Draft Report												■		
Final Report													■	
VIVA presentation														■

Table 2: Gantt chart of Final Year Project 2

3.2 Robotic Arm Flowchart

The activities that will be involved in this project is the interaction between the user and the robotic arm system where the main point of the activities is to copy the human motion which in this case, drawing. The robotic arm is expected to perform as per following:

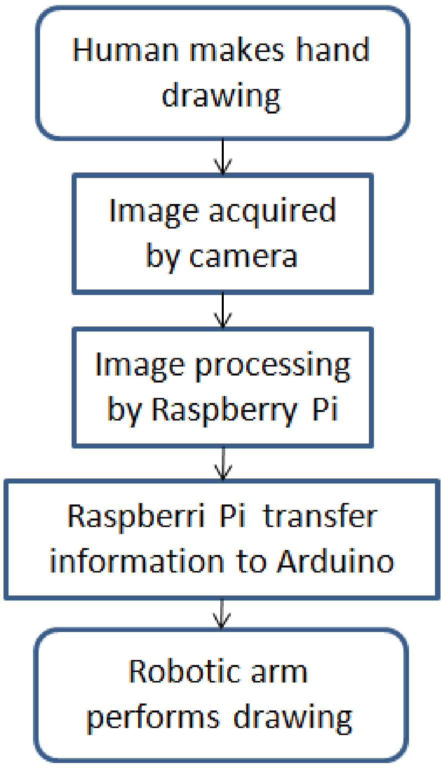

Human makes hand drawing
Image acquired by camera
Image processing by Raspberry Pi
Raspberri Pi transfer information to Arduino
Robotic arm performs drawing

Figure 5: Robotic Arm Expected Activities

From the expected outcome from the above diagram, the methodology for this project has been decided as follows:

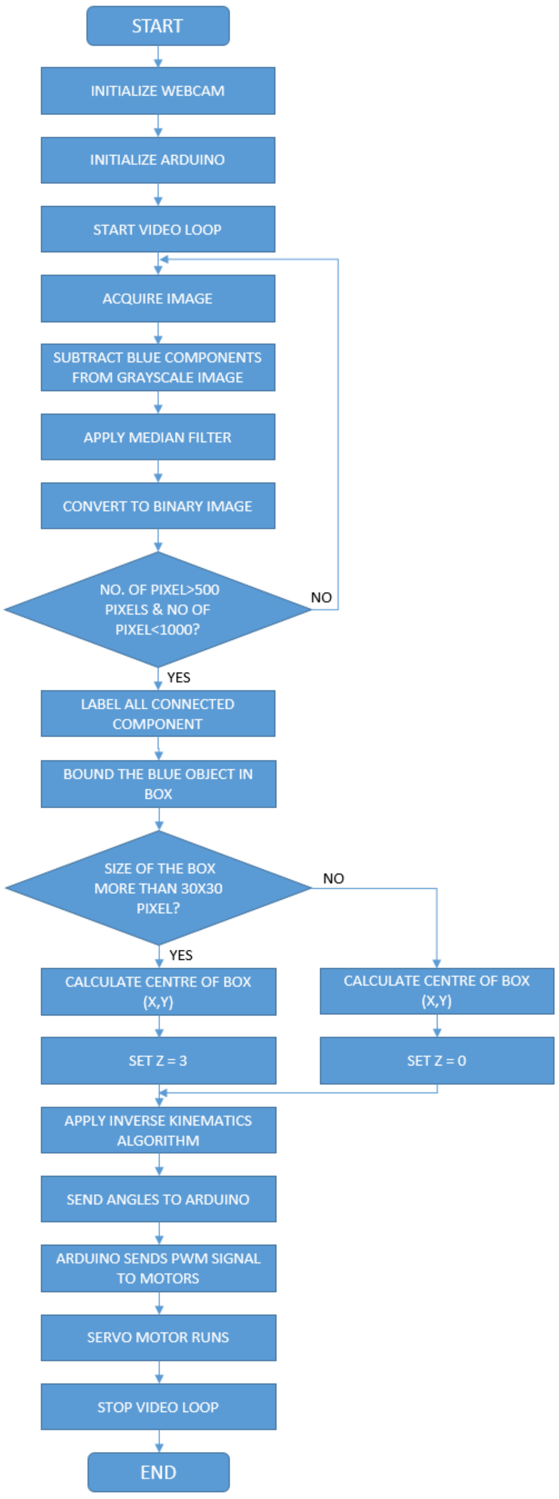

START
INITIALIZE WEBCAM
INITIALIZE ARDUINO
START VIDEO LOOP
ACQUIRE IMAGE
SUBTRACT BLUE COMPONENTS FROM GRAYSCALE IMAGE
APPLY MEDIAN FILTER
CONVERT TO BINARY IMAGE
NO. OF PIXEL>500 PIXELS & NO OF PIXEL<1000?
NO
YES
LABEL ALL CONNECTED COMPONENT
BOUND THE BLUE OBJECT IN BOX
SIZE OF THE BOX MORE THAN 30X30 PIXEL?
NO
YES
CALCULATE CENTRE OF BOX (X,Y)
CALCULATE CENTRE OF BOX (X,Y)
SET Z = 3
SET Z = 0
APPLY INVERSE KINEMATICS ALGORITHM
SEND ANGLES TO ARDUINO
ARDUINO SENDS PWM SIGNAL TO MOTORS
SERVO MOTOR RUNS
STOP VIDEO LOOP
END

Figure 6: Robotic Arm System Flowchart

The following figure shows the connections of every components needed in order for the robot to function. All components are divided into 3 subsystems, which is the image processing system, the inverse kinematics system and the power supply system.

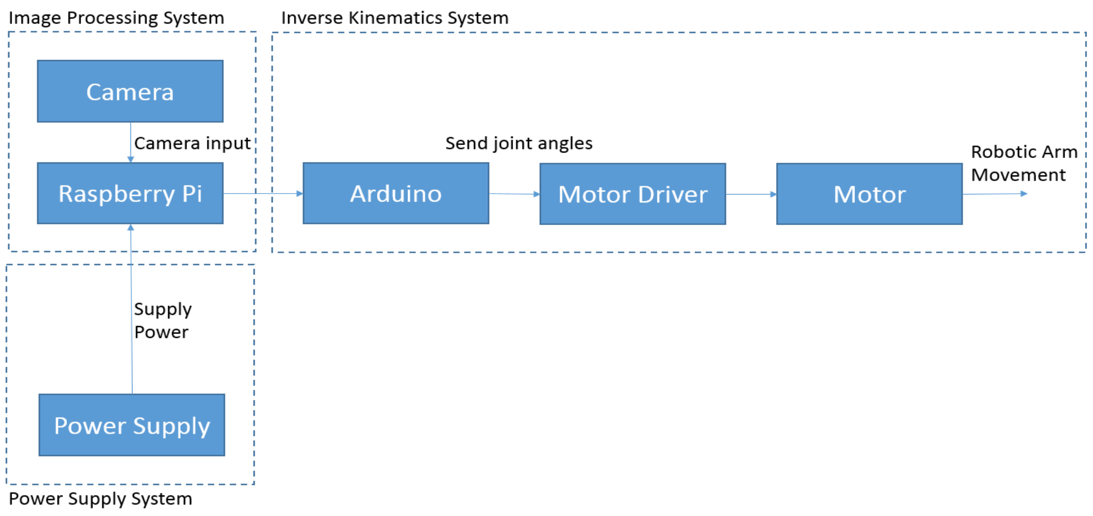


Figure 7: Robot’s component connections with subsystems

3.3 Image Processing

In the image processing activities of the robot, the camera will first capture the image and send them to be processed by the MATLAB software. For example, if the human makes drawing as per Figure 8, then the video of the drawing activity made by human will be processed frame by frame, until the drawing has finished.

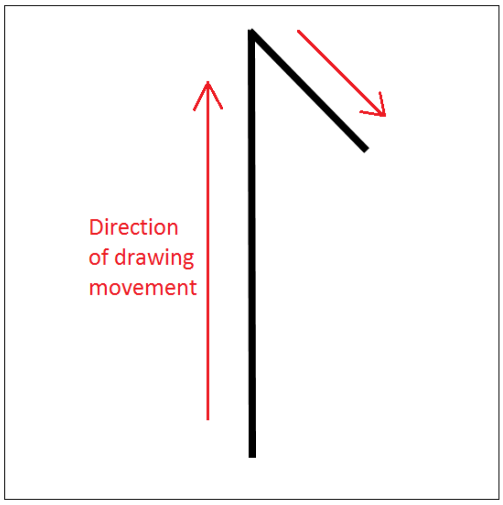


Figure 8: Drawing example made by human

When the video is being captured, the images from the video will be processed and the drawing image will be converted into coordinates in which will be used to instruct the robotic arm to perform the drawing, as showed in Figure 9 below:

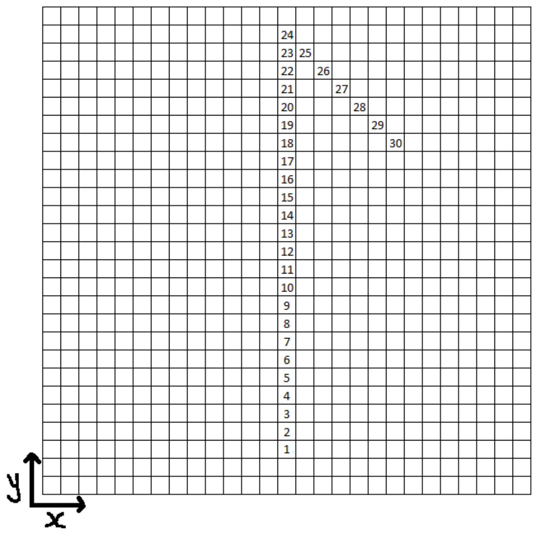


Figure 9: Drawings converted to coordinates in sequence

These (x, y) coordinates will then be sent to Arduino to convert the coordinates into angular rotation of every joint by using the algorithm that is based on the inverse kinematics of the robotic arm.

The real image was captured by a real webcam that is connected to a computer with MATLAB software. The image captured are mainly consists of three main thing, which is the blank white sheet, human hand, pen, and the drawing. A test run was done using a special equipment with the drawing platform as per the following Figure 10:

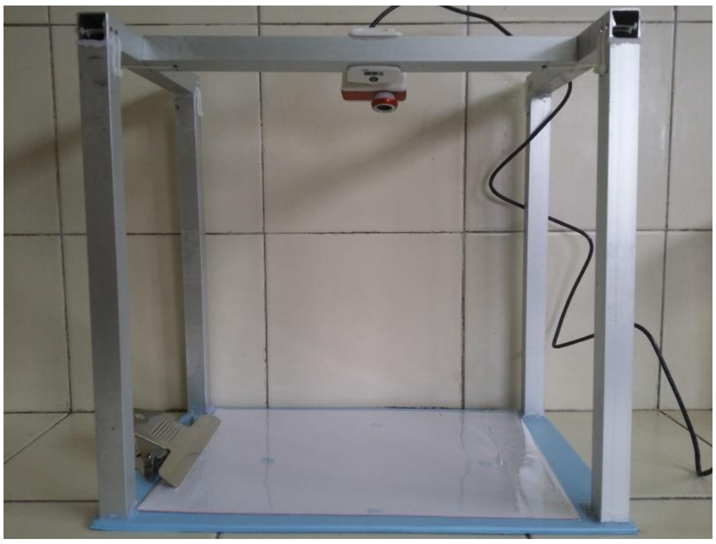


Figure 10: Drawing Platform for User

The test run was done with a person drawing on top of a blank paper using a black marker pen that is shown in Figure 11 below. The test run was done with the person drawing a dot as a simple simulation to get the coordinates from the image. The reason the dot was selected was to ensure that the image could be easily processed using the image processing algorithm to convert the drawing into coordinate values.

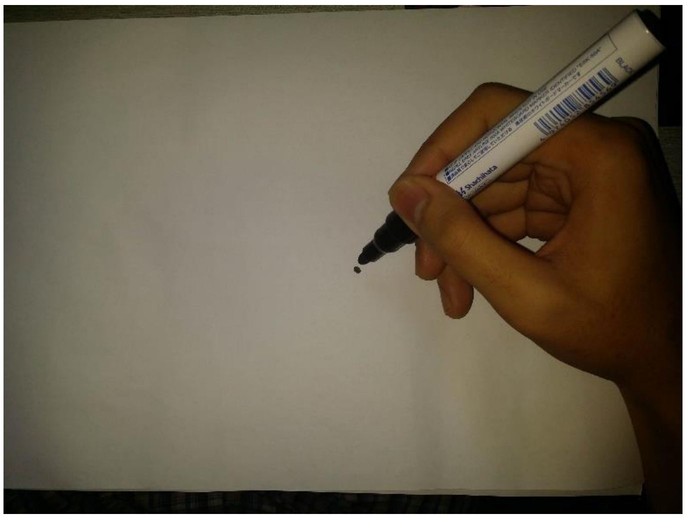


Figure 11: Image captured by the webcam to be processed

During this test run, the image was processed using colour segmentation techniques and subtracting the largest object in the image, which will be the person's hand and the pen that he was holding. Using a threshold of 250, the noise which is the largest object in the image was removed. The resulting drawing which is the dot was managed to be converted to coordinates by changing the indexed image of the drawing to subscripts. With this, image processing algorithm work on only a single image, but not with a live image from webcam.

Another test run was done using object tracking program, which uses simple image processing techniques such as subtracting the specific colour component in the image to detect that specific colour. The colour tracking worked successfully, and this approach looks promising in the later stages of this project.

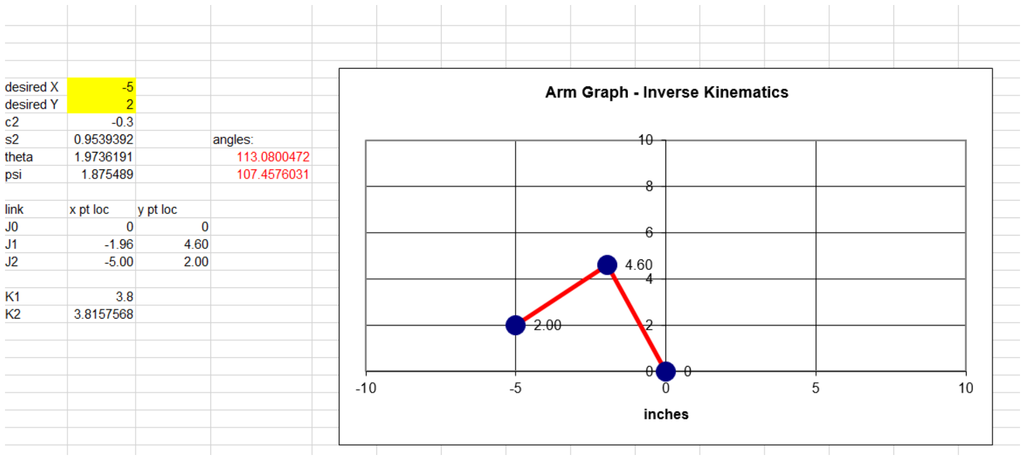
MATLAB Software

The platform that will be used throughout this project is the MATLAB R2015a software with Arduino Hardware Support Package. This requires a good basic knowledge of C language and communicating to Arduino. The advantages of the MATLAB software are:

- It is a very large and growing database of built-in algorithms especially for image processing and computer vision applications.
- It has the ability to read in a wide variety of both common and domain-specific image formats.
- It allows users to interact with data, keep track of files and variables and also simplifies programming and debugging tasks.
- The ability to process videos and use back the information using video feedback.

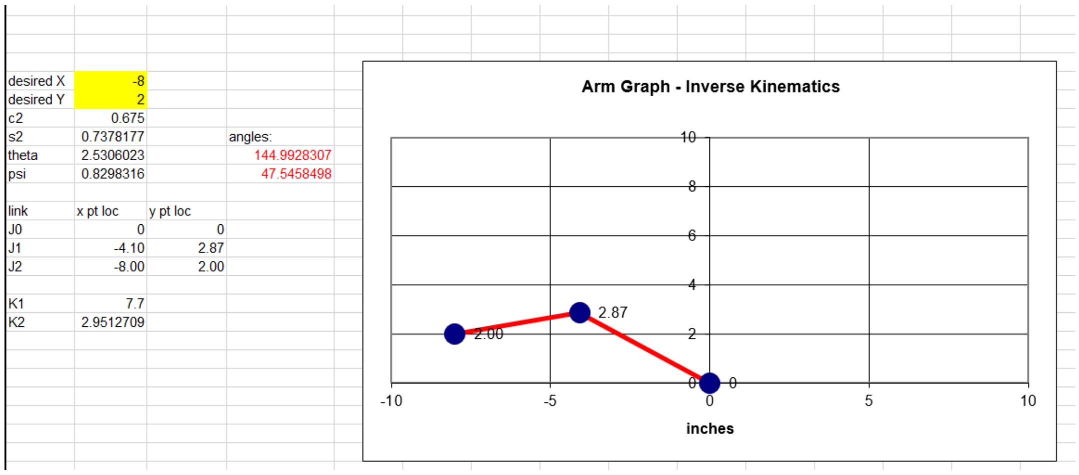
Based on the research that was conducted throughout FYP 1 and FYP 2 course, the robotic kinematics has been selected to be inverse kinematics. Forward kinematics are impossible to be used in this project because it finds the end-effector position with regards to the defined joint angles. On the other way, inverse kinematics finds the joint angles when the end-effector position is given. This suits the nature of this project where the end-effector position are taken from the coordinates that is going to be provided by the image processing subsystem that will be processed by the MATLAB software. From the video feedback taken from the camera, the image frames will be processed in every milliseconds, and the robotic arm will perform the drawing with the human simultaneously.

A simple robotic arm simulation was performed which focuses on the (x, y) plane. The simulation was performed in 2-dimension where the movement of the robotic arm can be seen in accordance to the inverse kinematics theory. Refer to Figure 12 and 13 below:



desired X	-5		
desired Y	2		
c2	-0.3		
s2	0.9539392		angles:
theta	1.9736191		113.0800472
psi	1.875489		107.4576031
link	x pt loc	y pt loc	
J0	0	0	
J1	-1.96	4.60	
J2	-5.00	2.00	
K1	3.8		
K2	3.8157568		

Figure 12: First simulation by feeding the first coordinates



desired X	-8		
desired Y	2		
c2	0.675		
s2	0.7378177		angles:
theta	2.5306023		144.9928307
psi	0.8298316		47.5458498
link	x pt loc	y pt loc	
J0	0	0	
J1	-4.10	2.87	
J2	-8.00	2.00	
K1	7.7		
K2	2.9512709		

Figure 13: Second simulation by feeding the second coordinates

The graph shown in both figures above are the expected movement of robotic arm in x and y plane. From both simulation, the joint angles (J2 and J3) are obtained, which is highlighted in red. These angles will be sent to the motors, and the motors will rotate in accordance to the angles given by the microcontroller.

3.4 Robotic Arm Parameter Selection

In order to select the best servo motor to be used in this project, the required torque that would ensure maximum efficiency needs to be calculated. A good servo with sufficient torque will ensure precision and accuracy of the end-effector which will make the robot to perfectly mimic the human drawing. Using the formula in (14) and (15) , and referring to the Figure 14 below, the required torque for every joint is calculated.

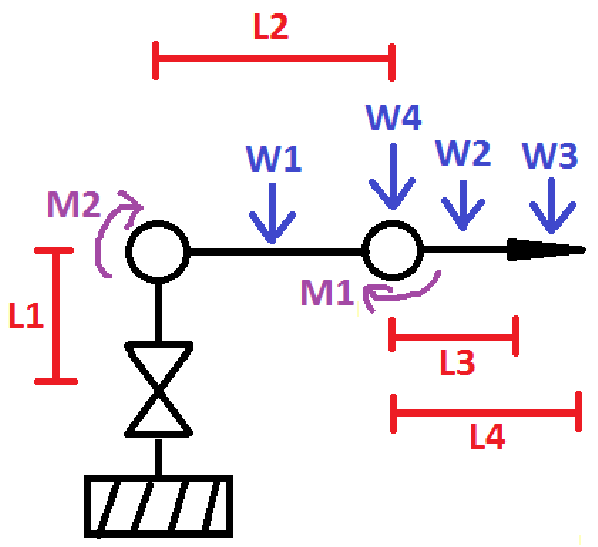


Figure 14: Robotic arm parameters

The following is the expected force calculation of joints where the following parameters are selected:

- Weight of each linkage
- Weight of each joint
- Weight of object to lift
- Length of each linkage

$$M1 = \left(\frac{L2}{2} * W1\right) + (L2 * W4) + (L2 + \frac{L3}{2}) * W2 + (L2 + L4) * W3 \quad (14)$$

$$M2 = \left(\frac{L3}{2} * W2\right) + L4 * W3 \quad (15)$$

As most of the robotic arm design is going to be made by using aluminium hollow square bar due to its strength and lightweight properties, the following parameter values are determined.

Parameter	Values
Length of the first link, L1	0.02m
Length of the second link, L2	0.15m
Length of the third link, L3	0.10m
Length from third joint to end-effector, L4	0.05m
Weight of the second link, W1	15g
Weight of the third link, W2	10g
Weight of end-effector, W3	10g
Weight of the third joint, W4	45.5

Table 3: Parameter values selected

When the parameter values are substituted in the equation, the result obtained was M1=21.949kg/cm and M2=1.379kg/cm. This shows that the required torque for the servo motor is M1=21.949kg/cm and M2=1.379kg/cm.

3.5 Robotic Arm

At the start of the FYP2 course, the fabrication of the robotic arm was initiated and the robotic arm are expected to be controlled by the end of week 6 as per the Gantt chart shown in Table 2. Thus, the robotic arm has been designed to be made using the following components:

Robot Component	**Specification**	**Quantity**
Linkage (Hollow Aluminium Tube)	0.150m x 0.018m x 0.018m	2 pieces
Joint (Servo Motor Tower Pro MG995)	0.041m x 0.020m x 0.043m	3 unit
Platform (Perspex)	0.300m x 0.210m x 0.003m	1 unit

Table 4: Robotic arm components

Combining all the components together, the robotic arm has been built in UTP's workshop as per the following figure:

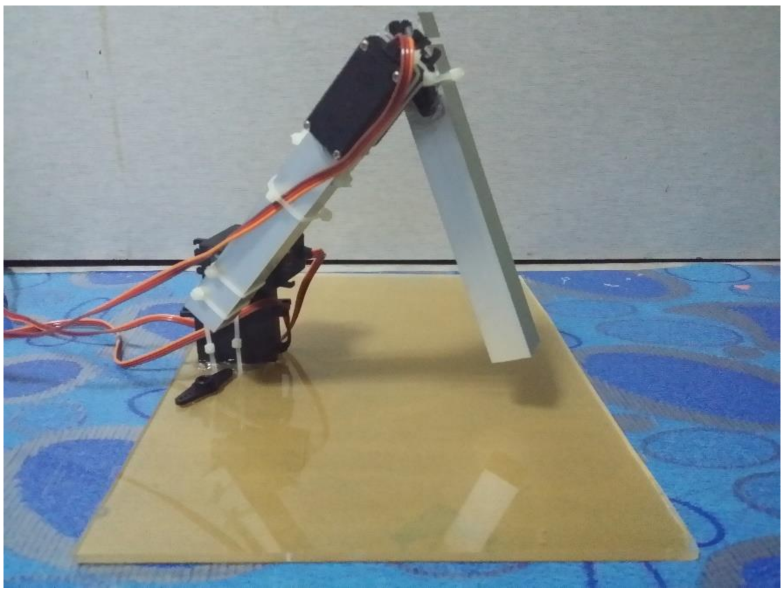


Figure 15: Fabricated Robotic Arm

In order to control the robotic arm, a microcontroller and supporting circuits must be used. The components used to control the robotic arm are as per the following table:

Robot Component	**Quantity**
Microcontroller (Arduino Mega2560)	1 unit
Breadboard	1 unit
Wire Strip	12 units
Power Supply (1.5V AA Batteries)	4 units
Arduino USB cable	1 unit

Table 5: Robotic Arm Controller Components

Using the Matlab software together with the Arduino Support Packages for Matlab, the robotic arm was programmed with the inverse kinematics algorithm (refer to appendix). The serial communication between the Arduino and every motor is shown in the circuit diagram in the figure below:

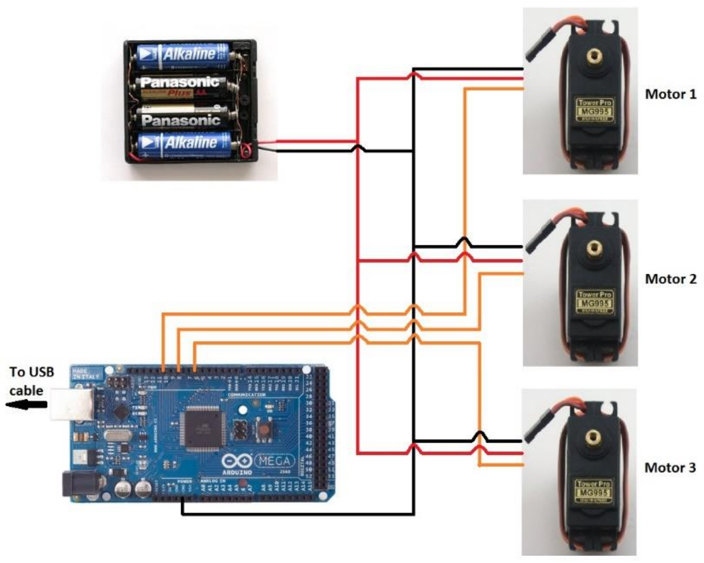


Figure 16: Serial configuration between Arduino Mega2560 and Motors (Joint)

In the figure above, the power for the motors are not supplied by the Arduino Mega2560 microcontroller due to insufficient current that is supplied by the Arduino pins. Thus, a battery supply is needed to supply power to the motor. The black lines indicates the ground (negative) connection, the red line indicates the positive supply of 6V, and the orange line indicates the PWM signal that will be produced by the Arduino to the motor. Figure below shows the real connection between the Arduino, motors and power supply.

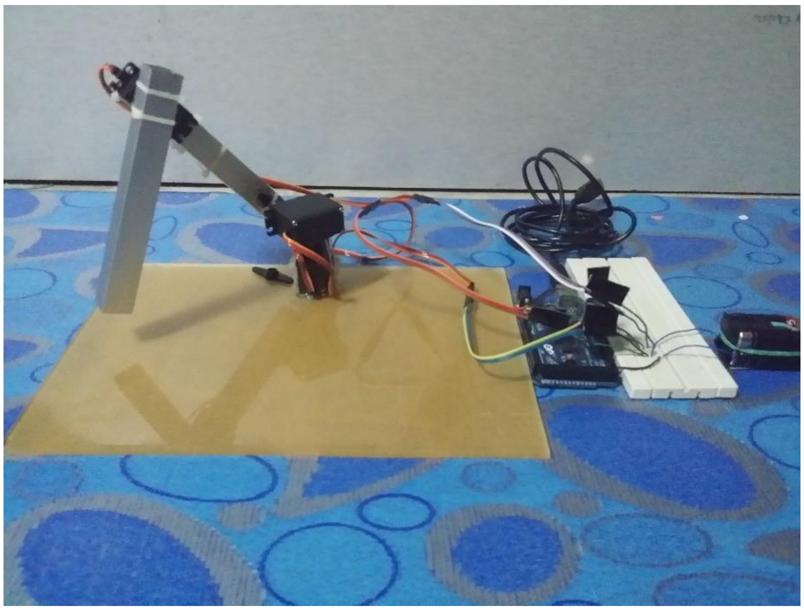


Figure 17: Connections between the robotic arm and Arduino

The Arduino was programmed to make the end effector of the robotic arm moves along a straight line in the x and y plane. The x and y plane is the same plane as the robotic arm's platform in the figure above. Using the lines of codes (refer to appendix) programmed via Matlab, the robotic arm moves to 7 points or coordinates which is (-3, 4), (-2, 4), (-1, 4), (0, 4), (1, 4), (2, 4), and (3, 4) respectively. Unfortunately, due to some disturbances or noise, the robotic arm did not move from point to point smoothly. Regardless of that, the robotic arm did succeed to move to every point given without fail.

CHAPTER 4: RESULTS AND DISCUSSION

4.1 Image Processing System

The image processing that was done in this project is to extract the coordinates of the blue dot from the image, where the blue dot represents the drawing motion that the user will make. It is to be expected that there will only be one blue object in the image as the detection of two or more blue objects in the image will result in error when the program feeds the separate coordinates to the inverse kinematics algorithm. The image captured by the webcam will result in an rgb image as in figure below. Take note that the user did not perform drawing in the image as this is a single image captured to check whether blue object in the image was successfully processed in order to obtain the coordinates. The blue object was determined to be 0.5cm radius so that the blue object can be detected clearly.

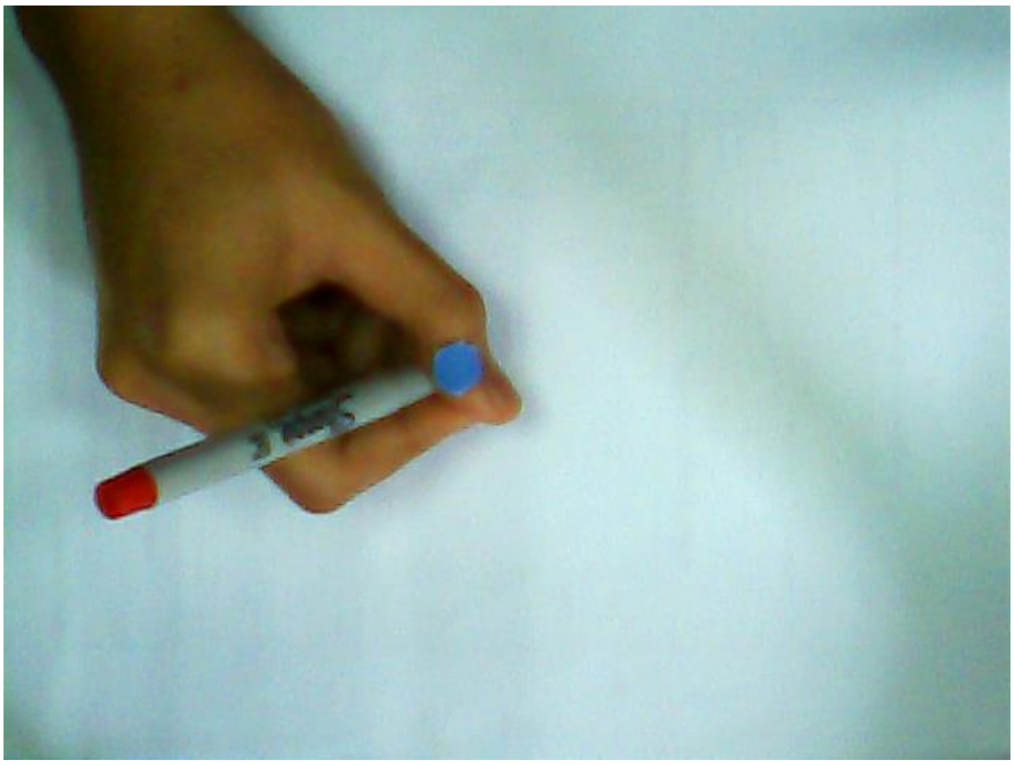


Figure 18: RGB Image of a Human Hand, Pen and Blue Dot

The first step of the image processing was done to subtract the blue components from the grayscale image, which will result in the detection of only the blue object in the image as in the figure below.

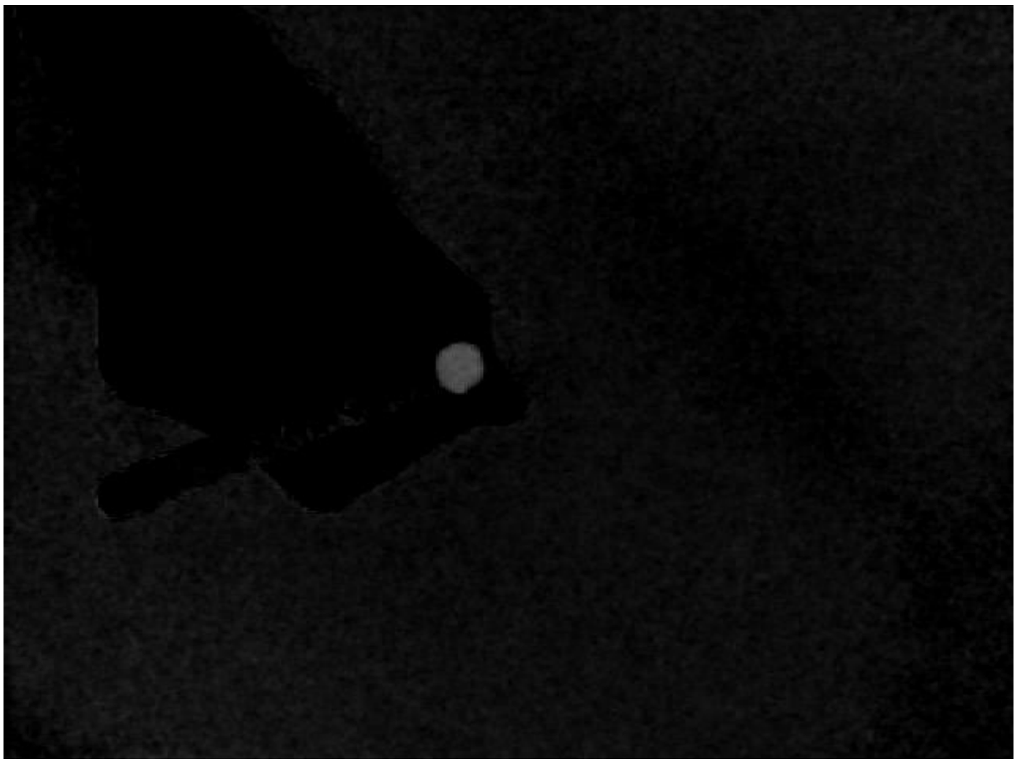


Figure 19: Blue Components Subtracted From Grayscale Image

Then the image will be applied with a median filter, where the median filter will filter out the noises, which will result in another grayscale image as in figure below.

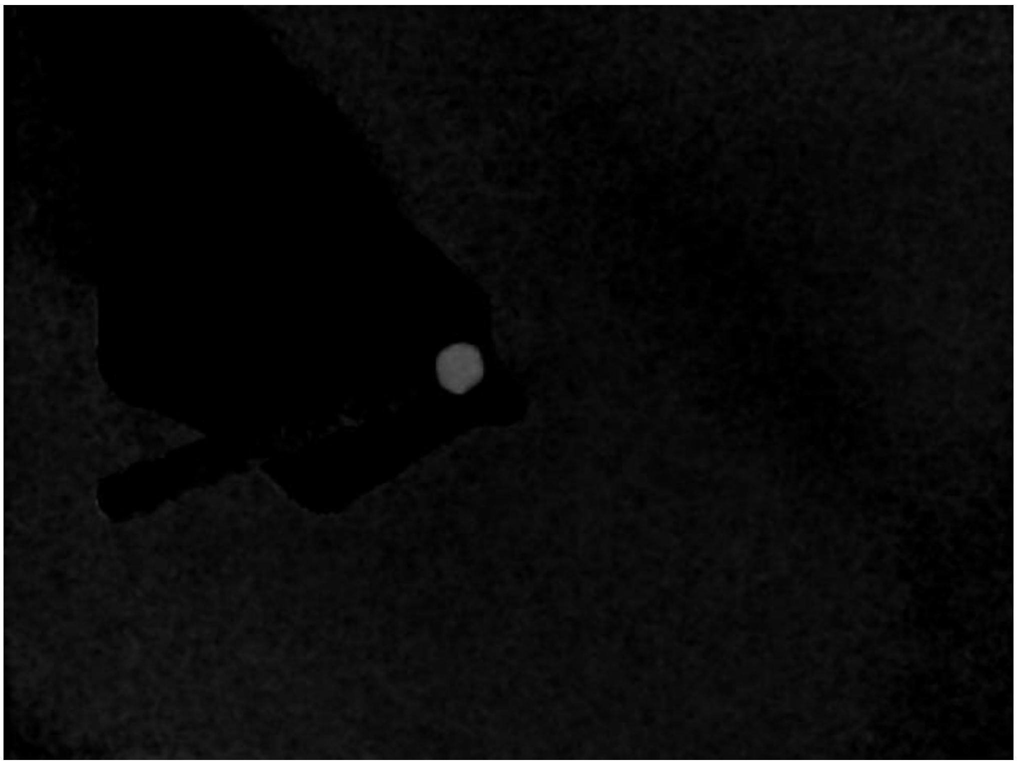


Figure 20: Image That Applied With Median Filter

In the grayscale image in Figure 20 above, the resulting image did not have much difference with Figure 19 because the image in did not have much noise in it. The image will then be converted from grayscale to binary to make the blue object more visible and connected as in figure 21 below.

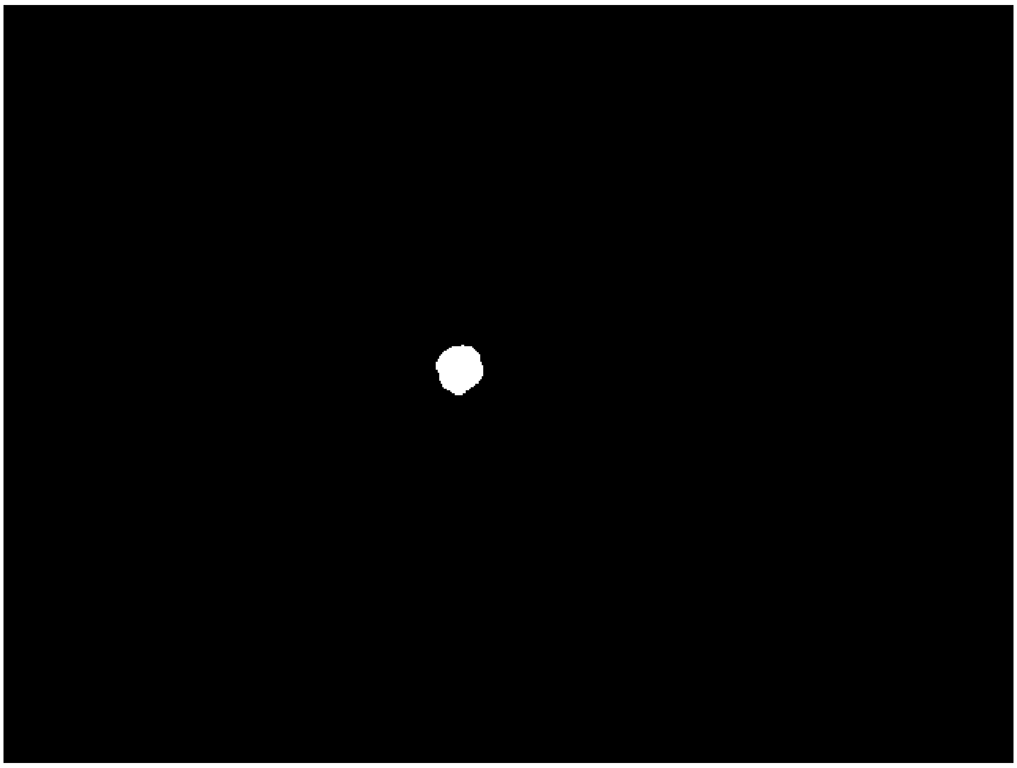


Figure 21: Binary Image Converted From the Grayscale Image

In Figure 21 above, the image now have a more visible dot which is the blue object. This image will then be applied with a filter that will remove all connected component that have less than 300 pixel and also more than 1000 pixel. As a result, only the blue object will remain in the object as in Figure 22 below.

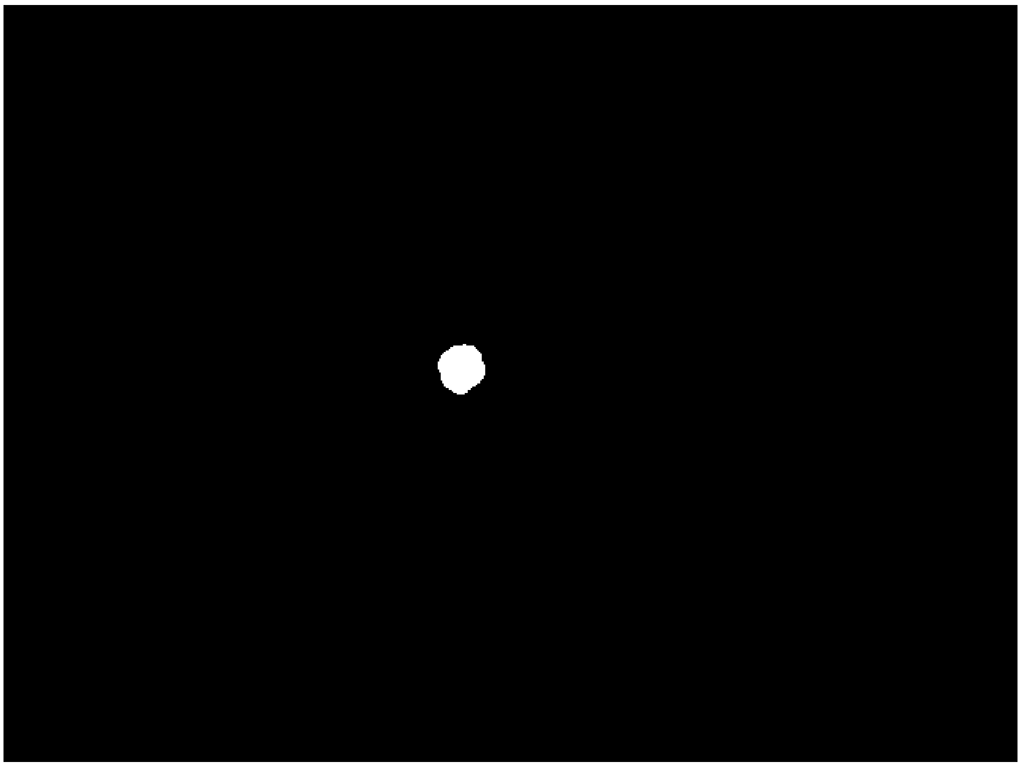


Figure 22: Subtraction of Other Blue Components in Image

Now, the resulting image shows that there is only the blue object in the image, where it represents the drawing motion that the user do when performing the drawing activities. Next, the connected component, which is the blue object will be labelled so that its information can be used such as size, coordinates and so on. As the main function of the image processing techniques are to extract the coordinates, the blue object will be bound in a red box (refer Figure 23 below), and the centre of the box will be the coordinates that is required to be used in the inverse kinematics algorithm.

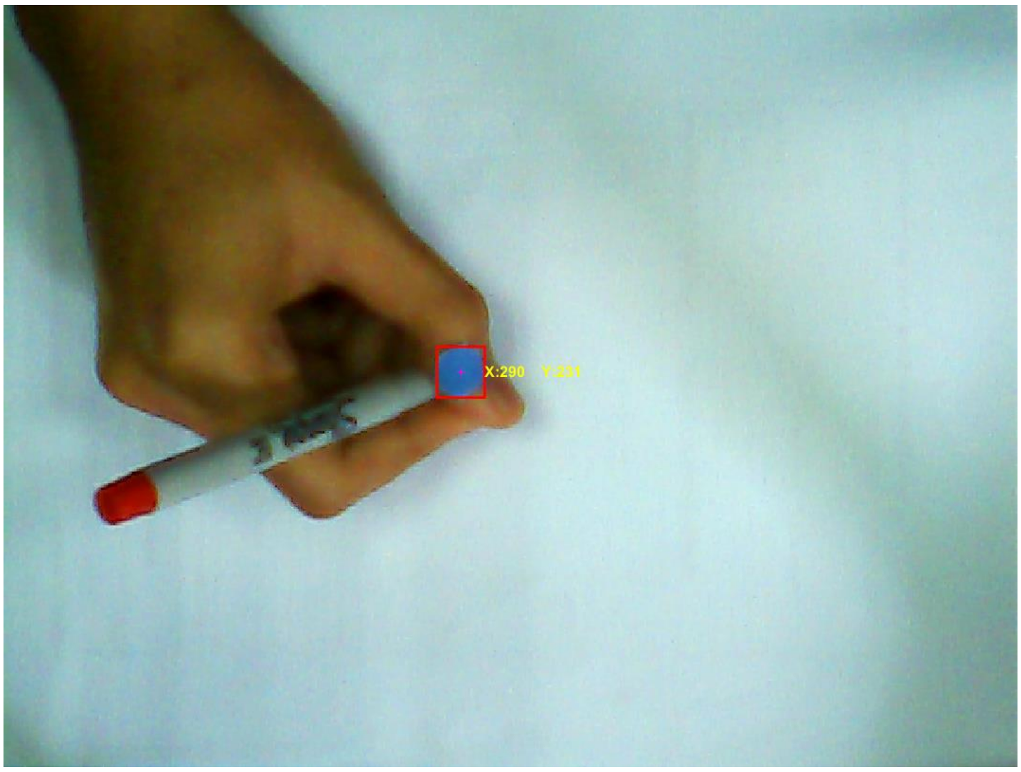


Figure 23: Blue Object Bound in Red Box with Coordinates

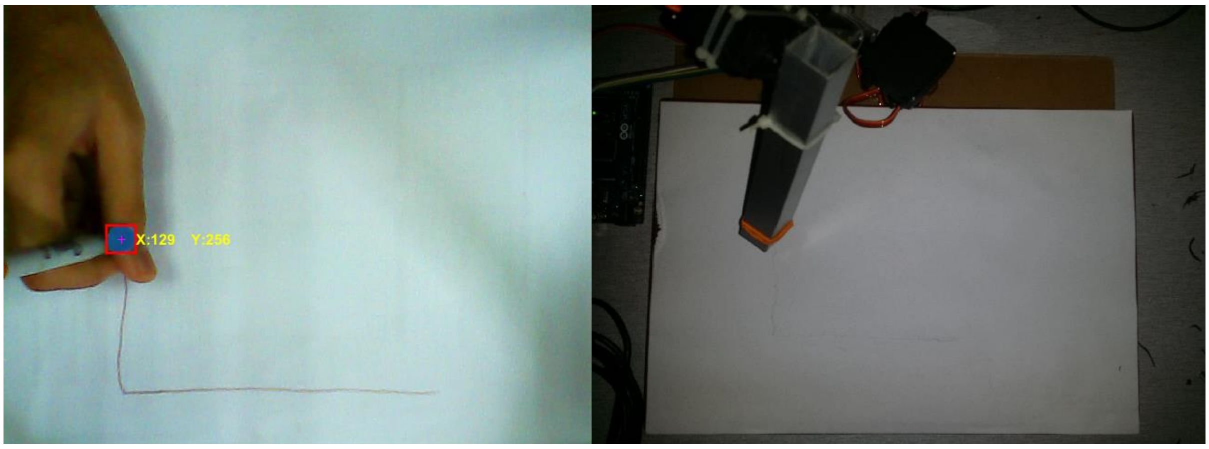
The figure above shows the resulting rgb image, with the red box to bound the blue object and displays the coordinates of the centre of the red box, which indicate the hand position of the pen in the image. This coordinate is going to be feed into the inverse kinematics algorithm to be used to convert the coordinates into motor angles.

4.2 Inverse Kinematics System

Towards the end of this project, the inverse kinematics system of the robotic arm was tested using the coordinates provided by the image processing system as in the previous section. Only this time, the coordinates were moving from one point to another, which will result in a series of coordinates which indicates the position of the human hand while drawing on top of the paper. All subsystems are integrated together in MATLAB software, where MATLAB can call the webcam function and receives input for the image processing subsystem and feeds the coordinates to Arduino hardware, which in the end gives us the proper drawings. From the start of this project in FYP1, it was expected that the robotic arm cannot give us a smooth drawing but more like a ‘robotic’ drawing, as discussed in the later section of this report.

<u>Square-Shaped Drawing</u>

The robotic arm system was tested by drawing a simple square shape on top of the paper and the robotic arm are expected to copy the drawings. The progress along the test run is shown in the figure below, side by side where the left one is the drawing made by human, and the right is the one that was made by the robotic arm. Note that during this test run, the human drawing was made very slowly in order for the program to work and capture every motion in the image acquired by webcam without missing the important coordinates.

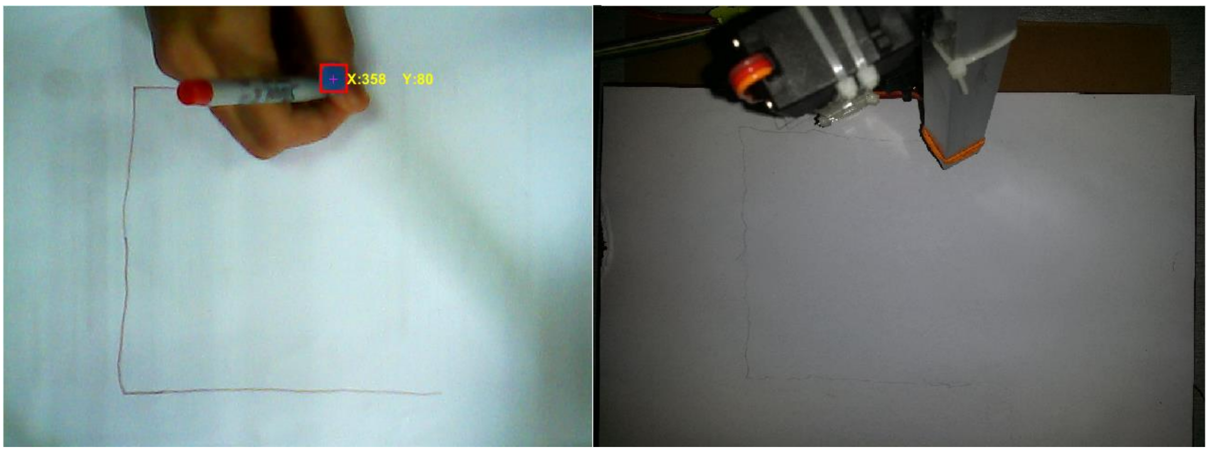


(a) Drawing made by human hand until it reaches (129,256) coordinates

(b) Drawing made by robotic arm until it reaches the same coordinates as Figure Y1(a)

Figure 24: Drawings made by both human and the robotic arm

During this test run, the robotic arm starts drawing only after the hand position which is the blue dot passes through the coordinate (x,y) = (129,256) as in Figure 25 above. If the human starts drawing at T = 0, the robot only starts reacting around 8 seconds after that (T ≈ 8). This shows that there are presence of unwanted delays in the program where some of the functions took too much time to compute, e.g. *getsnapshot* function that is used to get images from webcam.

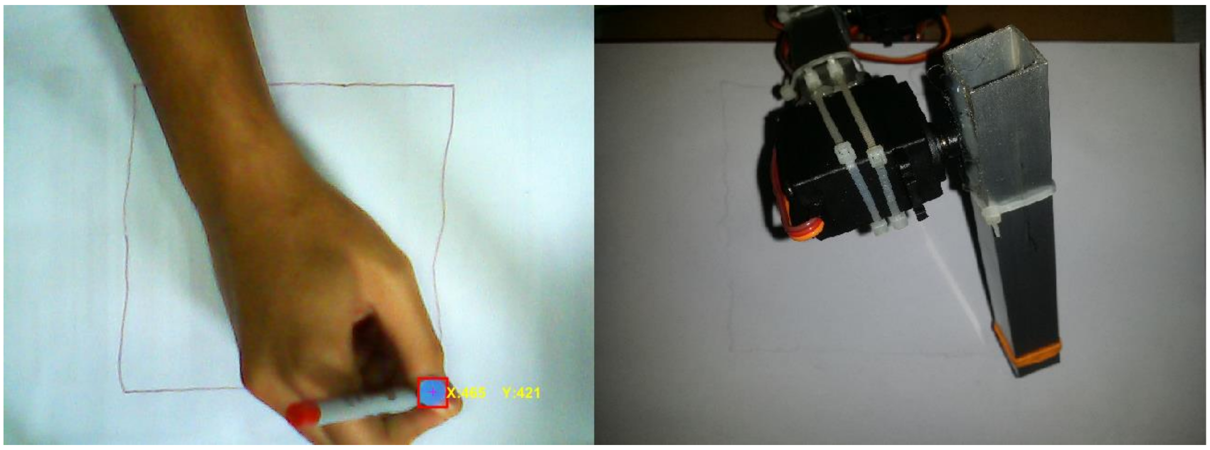


(a) Drawing made by human hand until it reaches (358,80) coordinates

(b) Drawing made by robotic arm until it reaches the same coordinates as Figure Y2(a)

Figure 25: Drawings made by both human and the robotic arm

At this point as in Figure 25, the robotic arm was managed to follow the square shape, but it is having small problem when the end effector was coming closer to the motors. The problem encountered at this point is that the motors and joints is becoming a huge load to the first motor and sometimes it cannot hold the position of the end effector which can result in bad drawing.



(a) Drawing made by human hand until it reaches (465,421) coordinates

(b) Drawing made by robotic arm until it reaches the same coordinates as Figure Y2(a)

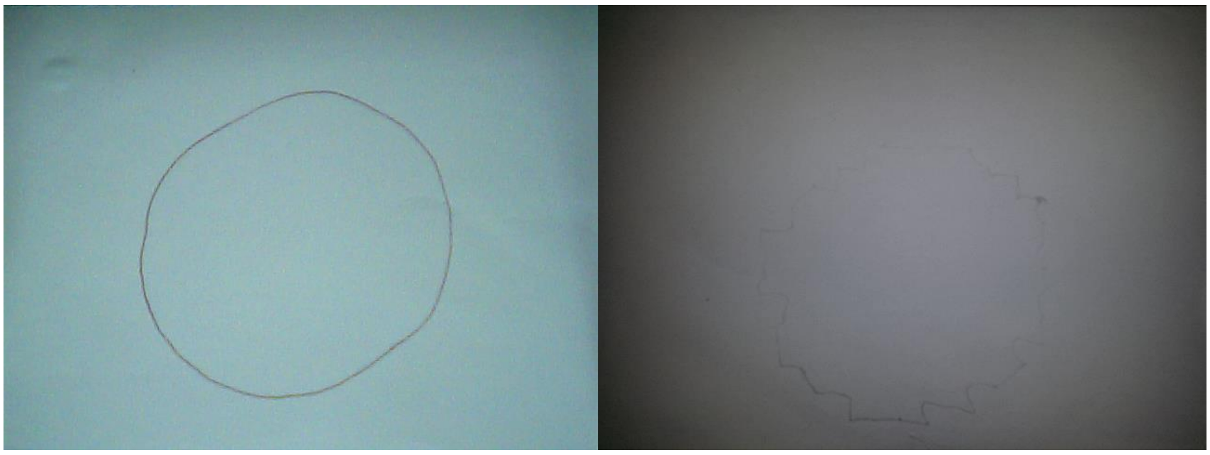
Figure 26: Drawings made by both human and the robotic arm

In the Figure 26 above, the human finishes his drawing and the robotic arm follows the drawing, but with an obvious delay. The time taken for the human to finish his drawing is 80.73 second, and the time taken for the robot to complete its drawing is 92.24 seconds.

From the test run that was conducted, it shows that the program have an unnecessary delay during the execution of the program. Regardless, the exact drawing was copied successfully for the square-shaped drawing, but smoothly because of the unsmooth motion of the robotic arm. This was expected because the robotic arm cannot produce a smooth drawing because there is no PID control are implemented in this project.

Circle-Shaped Drawing

Next is the test run for the curved-shape drawing which is going to be tested to check the capability of the robotic arm for curved-shaped drawings. The shape that is chosen here is a circle-shaped drawing, as shown in Figure 27 (a). The drawing that the robotic arm performs for the circle-shaped drawing is shown next to it (Figure 27 (b)).


(a) Circle-shaped drawing made by human hand

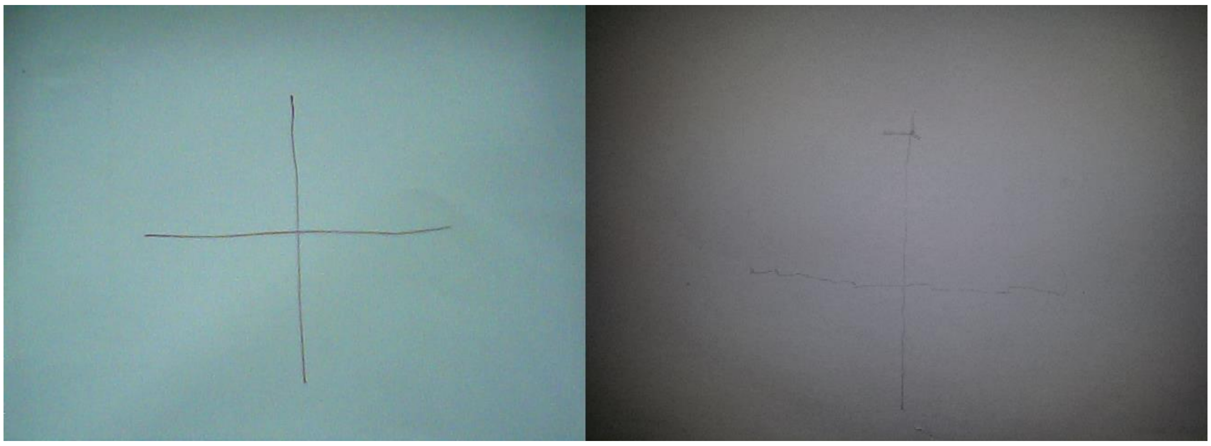
(b) Circle-shaped drawing made by robotic arm

Figure 27: Circle–shaped drawings made by both human and the robotic arm

Based on the above figure, the human made the circle-shaped drawing and the robotic arm made the circle-shaped drawing. The delay we discussed before is still present during this program execution. In Figure 27 (b), the drawing made by the robotic arm did not look exactly circle-shaped, but it still shows the important features of a circular drawing. This shows that this program is not suited for curved drawing, especially circle and oval shape. The time taken for the human to complete the drawing is 63.84 second, while the time taken for the robotic arm to complete its drawing is 78.21 second.

‘Plus’-Shaped Drawing

Next, we test the robotic arm system to draw shapes that are not in one stroke. This means that the pen that the human used to perform the drawing need to be disconnected from the drawing to make another stroke from other point, for example a ‘T’ shaped drawing or an ‘A’ shaped drawing. For this test run, plus (‘+’) shape were chosen as the drawing. The results are shown in the following figure below.


(a) ‘Plus’-shaped drawing made by human hand

(b) ‘Plus’-shaped drawing made by robotic arm

Figure 28: ‘Plus’–shaped drawings made by both human and the robotic arm

The Figure 28 (a) above shows the drawing made by human, and Figure 28 (b) above shows the drawing made by the robotic arm. The vertical stroke made by the robotic arm is almost perfect, while the horizontal stroke is not so smooth. Regardless, the important features of the human drawing is successfully mimicked by the robotic arm. Notice that the disconnected lines are successfully detected by the robotic arm system by detecting the increase in the number of pixel of the blue object when the user picks up his hand. The time taken for the human to complete his drawing is 59.56 second, while the time taken for the robotic arm to complete its drawing is 68.12 second.

CHAPTER 5: CONCLUSION AND RECOMMENDATION

With enough resources, this project can be done by using Arduino Mega2560 and MATLAB software and its support packages. The microcontroller and webcam need to be programmed and connected via MATLAB software so that this project will be successful.

From this project we can study the kinematic designs of robotic arm which requires a few calculations in order to imitate the drawings made by human hands. It was expected in this study that the drawings made by the robots are not exactly the same as the drawings by humans because of the noise factor.

The robotic arm succeeds in mimicking some linear drawings that was performed by human, but fails to mimic perfectly for the curved drawings. Regardless, the robotic arm can perform drawings that are disconnected from its line such as the T-shape and the A-shape. The delays that are present during the program execution are mainly because of the limitation of the MATLAB software where users cannot perform image processing as fast as the other platform such as OpenCV.

It is recommended in the future that the suitable platform to perform image processing is the OpenCV or Fortran/C due to its computation speed. Although MATLAB has a huge number of tools and libraries, but computing speed is important when it comes to real-time processing.

With this study we can help the artists to further develop their art skill and performance. With enough creativity and development, this project can contribute not only in the entertainment field, but also any other field such as manufacturing, automation and so on.

CHAPTER 6: References

1. Srikaew, A., Cambron, M. E., Northrup, S., Peters II, R. A., Wilkes, D. M., & Kawamura, K. (1998, July). Humanoid drawing robot. In *IASTED International Conference on Robotics and Manufacturing*.

2. Huang, J., Quan, B. T., Harada, M., & Yabuta, T. (2006, December). Emulating the Motion of a Human Upper Limb: Controlling a Finger-arm Robot by using the Manipulability of its Finger. In *Robotics and Biomimetics, 2006. ROBIO'06. IEEE International Conference on* (pp. 607-612). IEEE.

3. Ou, J., Fussell, S. R., Chen, X., Setlock, L. D., & Yang, J. (2003, November). Gestural communication over video stream: supporting multimodal interaction for remote collaborative physical tasks. In *Proceedings of the 5th international conference on Multimodal interfaces* (pp. 242-249). ACM.

4. Raihan, K. J., Rahaman, M. S., Sarkar, M. K., & Mahfuz, S. (2013). Raspberry Pi Image Processing based Economical Automated Toll System. *Global Journal of Researches In Engineering*, *13*(13).

5. http://www.societyofrobots.com/robot_arm_tutorial.shtml

6. Farhan, M. (2014). Parallel Kinematic Robot Controller. *Dissertation, University Technology Petronas.*

CHAPTER 7: Appendix

```
clear
imaqreset
% a = imaqhwinfo;
% [camera_name, camera_id, format] = getCameraInfo(a);


% Capture the video frames using the videoinput function
% You have to replace the resolution & your installed adaptor name.
vid = videoinput('winvideo', 1, 'YUY2_640x480');
% src = getselectedsource(vid)
% Set the properties of the video object
set(vid, 'FramesPerTrigger', Inf);
set(vid, 'ReturnedColorspace', 'rgb')
vid.FrameGrabInterval = 10;

%start the video aquisition here
start(vid)
Amotor=arduino('COM5');  % connect the board
% Set a loop that stop after 100 frames of aquisition

% while(vid.FramesAcquired<=1000)
%     preview(vid)
    % Get the snapshot of the current frame
    data = getdata(vid,1,'uint8');
    imtool(data)
    % Now to track red objects in real time
    % we have to subtract the red component
    % from the grayscale image to extract the red components in the image.
    diff_im1 = imsubtract(data(:,:,3), rgb2gray(data));
%     imtool(diff_im1)
    %Use a median filter to filter out noise
    diff_im1 = medfilt2(diff_im1, [3 3]);
%     imtool(diff_im1)
    % Convert the resulting grayscale image into a binary image.
    diff_im1 = im2bw(diff_im1,0.18);
%     imtool(diff_im1)
    % Remove all those pixels less than 300px
%     diff_im1 = xor(bwareaopen(diff_im1,500),bwareaopen(diff_im1,1000));
%blue
%     diff_im1 = xor(bwareaopen(diff_im1,30),bwareaopen(diff_im1,130));
%red
%     diff_im1 = xor(bwareaopen(diff_im1,30),bwareaopen(diff_im1,130));
%red
    diff_im1 = bwareaopen(diff_im1,100);
%     diff_im1 = bwareafilt(diff_im1,1);
%     imtool(diff_im1)
    % Label all the connected components in the image.
    bw = bwlabel(diff_im1, 8);

    % Here we do the image blob analysis.
    % We get a set of properties for each labeled region.
    stats = regionprops(bw, 'BoundingBox', 'Centroid');

    % Display the image
    imshow(data)

    hold on
    x = zeros(1, length(stats));
    y = zeros(1, length(stats));
    hyp = zeros(1, length(stats));
    c2 = zeros(1, length(stats));
    s2 = zeros(1, length(stats));
    k1 = zeros(1, length(stats));
    k2 = zeros(1, length(stats));
    T1rad = zeros(1, length(stats));
    T2rad = zeros(1, length(stats));
    T3rad = zeros(1, length(stats));
    T1deg = zeros(1, length(stats));
    T2deg = zeros(1, length(stats));
    T3deg = zeros(1, length(stats));
    Theta1 = zeros(1, length(stats));
    Theta2 = zeros(1, length(stats));
    Theta3 = zeros(1, length(stats));


    %This is a loop to bound the red objects in a rectangular box.
    for object = 1:length(stats)
        bb = stats(object).BoundingBox;
        bc = stats(object).Centroid;
        rectangle('Position',bb,'EdgeColor','r','LineWidth',2)
        plot(bc(1),bc(2), '-m+')
        a=text(bc(1)+15,bc(2), strcat('X: ', num2str(round(bc(1))), '    Y:
', num2str(round(bc(2)))));
        set(a, 'FontName', 'Arial', 'FontWeight', 'bold', 'FontSize', 12,
'Color', 'yellow');
        if bb(3)>29 && bb(4)>32

            clear M1 M2 M3
            x(object) = round((320-bc(1))*0.036458333);
            y(object) = round((bc(2))*0.036458333);
            z = 3;
            L1 = 12;
            L2 = 12;
            if x(object)<0
                T3rad(object) = atan(abs(x(object))/y(object));
                T3deg(object) = T3rad(object)*(180/pi);

                hyp(object) = abs(x(object))/sin(T3rad(object));

                c2(object)=(z^2+hyp(object).^2-L1^2-L2^2)/(2*L1*L2);
                s2(object)=sqrt(1-c2(object).^2);

                k1(object)=L1+L2*c2(object);
                k2(object)=L2*s2(object);

                T1rad(object) = atan2(z,-hyp(object))-
atan2(k2(object),k1(object));    % theta(1) in radian
                T2rad(object) = atan2(s2(object),c2(object));
% theta(2) in radian

                T1deg(object) = T1rad(object)*(180/pi);
% theta(1) in degree
                T2deg(object) = T2rad(object)*(180/pi);
% theta(2) in degree
```

```
                Theta1(object) = (180-T1deg(object))/180;
% angle (scale 0~1) for servo #1
                Theta2(object) = T2deg(object)/180;
% angle (scale 0~1) for servo #2
                Theta3(object) = (T3deg(object)+90)/180;
% angle (scale 0~1) for servo #3

                M1 = servo(Amotor, 'D8');              % attach servo #1
                M2 = servo(Amotor, 'D10');              % attach servo #2
                M3 = servo(Amotor, 'D5');             % attach servo #3

                writePosition(M1, Theta1(object)-0.03);
                writePosition(M2, Theta2(object)+0.04);
                writePosition(M3, Theta3(object)-0.05);

            elseif x(object)==0
                T3rad(object) = atan(abs(x(object))/y(object));
                T3deg(object) = T3rad(object)*(180/pi);

                hyp(object) = y(object);

                c2(object)=(z^2+hyp(object).^2-L1^2-L2^2)/(2*L1*L2);
                s2(object)=sqrt(1-c2(object).^2);

                k1(object)=L1+L2*c2(object);
                k2(object)=L2*s2(object);

                T1rad(object) = atan2(z,-hyp(object))-
atan2(k2(object),k1(object));   % theta(1) in radian
                T2rad(object) = atan2(s2(object),c2(object));
% theta(2) in radian

                T1deg(object) = T1rad(object)*(180/pi);
% theta(1) in degree
                T2deg(object) = T2rad(object)*(180/pi);
% theta(2) in degree

                Theta1(object) = (180-T1deg(object))/180;
% angle (scale 0~1) for servo #1
                Theta2(object) = T2deg(object)/180;
% angle (scale 0~1) for servo #2
                Theta3(object) = (T3deg(object)+90)/180;
% angle (scale 0~1) for servo #3

                M1 = servo(Amotor, 'D8');              % attach servo #1
                M2 = servo(Amotor, 'D10');              % attach servo #2
                M3 = servo(Amotor, 'D5');             % attach servo #3

                writePosition(M1, Theta1(object)-0.03);
                writePosition(M2, Theta2(object)+0.04);
                writePosition(M3, Theta3(object)-0.05);
            else
                T3rad(object) = atan(x(object)/y(object));
                T3deg(object) = T3rad(object)*(180/pi);

                hyp(object) = x(object)/sin(T3rad(object));

                c2(object)=(z^2+hyp(object).^2-L1^2-L2^2)/(2*L1*L2);
                s2(object)=sqrt(1-c2(object).^2);
                k1(object)=L1+L2*c2(object);
                k2(object)=L2*s2(object);

                T1rad(object)=atan2(z,-hyp(object))-
atan2(k2(object),k1(object));    % theta(1) in radian
                T2rad(object)=atan2(s2(object),c2(object));
% theta(2) in radian

                T1deg(object)=T1rad(object)*(180/pi);
% theta(1) in degree
                T2deg(object)=T2rad(object)*(180/pi);
% theta(2) in degree

                Theta1(object)=(180-T1deg(object))/180;
% angle (scale 0~1) for servo #1
                Theta2(object)=T2deg(object)/180;
% angle (scale 0~1) for servo #2
                Theta3(object)=(90-T3deg(object))/180;
% angle (scale 0~1) for servo #3

                M1 = servo(Amotor, 'D8');              % attach servo #1
                M2 = servo(Amotor, 'D10');              % attach servo #2
                M3 = servo(Amotor, 'D5');             % attach servo #3

                writePosition(M1, Theta1(object)-0.03);
                writePosition(M2, Theta2(object)+0.04);
                writePosition(M3, Theta3(object)-0.05);
            end
        elseif bb(3)<=29 && bb(4)<=32
            clear M1 M2 M3
            x(object) = round((320-bc(1))*0.036458333);
            y(object) = round((bc(2))*0.036458333);
            z = 0;
            L1 = 12;
            L2 = 12;
            if x(object)<0
                T3rad(object) = atan(abs(x(object))/y(object));
                T3deg(object) = T3rad(object)*(180/pi);

                hyp(object) = abs(x(object))/sin(T3rad(object));

                c2(object)=(z^2+hyp(object).^2-L1^2-L2^2)/(2*L1*L2);
                s2(object)=sqrt(1-c2(object).^2);

                k1(object)=L1+L2*c2(object);
                k2(object)=L2*s2(object);

                T1rad(object) = atan2(z,-hyp(object))-
atan2(k2(object),k1(object));    % theta(1) in radian
                T2rad(object) = atan2(s2(object),c2(object));
% theta(2) in radian
                T1deg(object) = T1rad(object)*(180/pi);
% theta(1) in degree
                T2deg(object) = T2rad(object)*(180/pi);
% theta(2) in degree

                Theta1(object) = (180-T1deg(object))/180;
% angle (scale 0~1) for servo #1
```

```
                Theta2(object) = T2deg(object)/180;
% angle (scale 0~1) for servo #2
                Theta3(object) = (T3deg(object)+90)/180;
% angle (scale 0~1) for servo #3

                M1 = servo(Amotor, 'D8');              % attach servo #1
                M2 = servo(Amotor, 'D10');              % attach servo #2
                M3 = servo(Amotor, 'D5');              % attach servo #3

                writePosition(M1, Theta1(object)-0.03);
                writePosition(M2, Theta2(object)+0.04);
                writePosition(M3, Theta3(object)-0.05);

            elseif x(object)==0
                T3rad(object) = atan(abs(x(object))/y(object));
                T3deg(object) = T3rad(object)*(180/pi);

                hyp(object) = y(object);

                c2(object)=(z^2+hyp(object).^2-L1^2-L2^2)/(2*L1*L2);
                s2(object)=sqrt(1-c2(object).^2);

                k1(object)=L1+L2*c2(object);
                k2(object)=L2*s2(object);

                T1rad(object) = atan2(z,-hyp(object))-
atan2(k2(object),k1(object));   % theta(1) in radian
                T2rad(object) = atan2(s2(object),c2(object));
% theta(2) in radian
                T1deg(object) = T1rad(object)*(180/pi);
% theta(1) in degree
                T2deg(object) = T2rad(object)*(180/pi);
% theta(2) in degree

                Theta1(object) = (180-T1deg(object))/180;
% angle (scale 0~1) for servo #1
                Theta2(object) = T2deg(object)/180;
% angle (scale 0~1) for servo #2
                Theta3(object) = (T3deg(object)+90)/180;
% angle (scale 0~1) for servo #3

                M1 = servo(Amotor, 'D8');              % attach servo #1
                M2 = servo(Amotor, 'D10');              % attach servo #2
                M3 = servo(Amotor, 'D5');              % attach servo #3

                writePosition(M1, Theta1(object)-0.03);
                writePosition(M2, Theta2(object)+0.04);
                writePosition(M3, Theta3(object)-0.05);
            else
                T3rad(object) = atan(x(object)/y(object));
                T3deg(object) = T3rad(object)*(180/pi);

                hyp(object) = x(object)/sin(T3rad(object));

                c2(object)=(z^2+hyp(object).^2-L1^2-L2^2)/(2*L1*L2);
                s2(object)=sqrt(1-c2(object).^2);

                k1(object)=L1+L2*c2(object);
                k2(object)=L2*s2(object);
                T1rad(object)=atan2(z,-hyp(object))-
atan2(k2(object),k1(object));   % theta(1) in radian
                T2rad(object)=atan2(s2(object),c2(object));
% theta(2) in radian

                T1deg(object)=T1rad(object)*(180/pi);
% theta(1) in degree
                T2deg(object)=T2rad(object)*(180/pi);
% theta(2) in degree

                Theta1(object)=(180-T1deg(object))/180;
% angle (scale 0~1) for servo #1
                Theta2(object)=T2deg(object)/180;
% angle (scale 0~1) for servo #2
                Theta3(object)=(90-T3deg(object))/180;
% angle (scale 0~1) for servo #3

                M1 = servo(Amotor, 'D8');              % attach servo #1
                M2 = servo(Amotor, 'D10');              % attach servo #2
                M3 = servo(Amotor, 'D5');             % attach servo #3

                writePosition(M1, Theta1(object)-0.03);
                writePosition(M2, Theta2(object)+0.04);
                writePosition(M3, Theta3(object)-0.05);
            end
        end
    end
    hold off
% end
% Both the loops end here.

% Stop the video aquisition.
stop(vid);

% Flush all the image data stored in the memory buffer.
flushdata(vid);

% Clear all variables
% clear all
```