University of Nottingham
UK | CHINA | MALAYSIA

EEE Department of Electrical and Electronic Engineering

EEEE3001

Final Year Individual Project Dissertation

Gesture-controlled Robot Arm

AUTHOR:	**Mr. Guan-Sheng Chen**
ID NUMBER:	**20314386**
SUPERVISOR:	**Dr. Richard Davies**
MODERATOR:	**Dr. Fernado Perez-Cota**
DATE:	**09 May 2024**

This third-year project Dissertation is submitted in part fulfilment of the requirements of the degree of Bachelor of Engineering.

Abstract

The robot arm is widely used in various applications from industry to medical applications. The thesis is to develop a combination of software design and hardware design for the robot arm controlled more natively by gesture to help rehabilitation therapists in rehabilitation treatment. To achieve effective treatment, patients are required to attend rehabilitation treatment regularly. However, the process of rehabilitation treatment is monotony and repetitive as repetitive and coordinated movements are crucial in rehabilitation treatment. It will greatly reduce patient engagement. The use of gesture-controlled robot arms in rehabilitation has the potential to enhance patient engagement and cost-effective option for rehabilitation centres. The project is expected to create various functionalities to realize a gesture-controlled robot arm at low-cost.

Typically, the control system of robot arm is challenging to program and control. In this project, a computer vision method for gesture recognition will be established to combine with robot arm of Dobot Magician to fulfil a real-time gesture-controlled robot arm with high precision and high accuracy control to demonstrate the potential of a comprehensive rehabilitation robot arm controlled by gestures. Four main functionalities are built including human hand identification, customized gesture recognition, combinations of Dobot Magician to achieve complicated pattern drawing tasks, and real-time control of robot arm by gestures.

The experimental results demonstrate the robustness of the built system, high accuracy control and high effective control of robot arm to demonstrate the potential of the accomplishment of gesture-controlled robot arm in rehabilitation applications.

Key Words: Xbox One Kinect Motion Sensor, Dobot Magician, OpenCV, MediaPipe, CVZone, Tensor Flow, Teachable Machine

Contents

1 Introduction and Background

1.1 Project Description

The major application of robotic arms is in manufacturing in industries, since robotic arms can provide assistance in assembly, welding, and material handling [1]. It can greatly enhance the efficiency, precision, and speed of production processes in order to improving the productivity and cost-effectiveness for industries [1]. Based on its multi-functionality, robotic arms can be used in automobile industry, electrical or electronic industry, medical industry and so on [1, 2]. According to the prediction of market, the value of robotic arm market will achieve USD 84.66 billion by 2031 [1]. The prediction indicates the potential of robotic arms in the future tendency. Moreover, the application of robotic arms can also be utilized in the fields such as physiotherapy, agriculture, education and so on [3-5].

The combination of using gesture to control robotic arms can significantly expands the application fields such as a rehabilitation robot, a wireless control industrial robot, using gesture to navigate robot and so on [6-8]. Moreover, combining the computer vision (using a camera for gesture recognition) can fulfil real-time control robot arm with gesture [9].

Based on the above research, the objective of this project is to build a gesture-controlled robot arm for rehabilitation applications. The robot arm can help rehabilitation therapists precisely control the patient's motion range to prevent overextension and ensure patients can perform exercises correctly [6].

1.2 Specification

Stage 1: Camera Setting and Image Processing

Aim:

To set up the camera with OpenCV and using OpenCV to remove the background noise for the process of hand detection.

Objectives:

- Literature review about the strategy of OpenCV to process image
- Camera setting with OpenCV to be a webcam to recognize
- Remove background noise for identifying human hands

Stage 2: Hand Detection

Aim: To identify the position of the hand and obtain the detailed information of the hand

Objectives:

- Literature review about the strategy of OpenCV to identify the hand
- Literature review about the strategy of Python packages to identify the hand
- Hand position recognition
- Obtain the information of fingers e.g. which fingers are up
- Identify the number of hands detected
- Left hand or right hand classification
- Multi-hand detection

Stage 3: Gesture Recognition (Mode)

Aim: To recognize the hand pose for the corresponding tasks in specified gesture recognition

Objectives:

- Literature review about the strategy of complex gesture recognition
- Complicated hand pose recognition
- Multi-hand pose recognition

Stage 4: Specified Gesture Recognition
Aim: To implement corresponding tasks from gesture recognition
Objectives:

- Finger Counting
- Multi-hand recognition
- Distance measurement of centre palm of two hands
- Using index finger for virtual painter on the screen
- Face depth distance measurement
- Shapes classification from the image
- Coordinate system identification of hands and fingers to implement real-time control Dobot Magician
- Angle algorithm of specified points of fingers to implement real-time control Dobot Magician

Stage 5: Communication with Robot Arm and Robot Arm Movement
Aim: Using physical transmission cable to connect Dobot Magician (robot arm) and transmit commends to implement corresponding tasks
Objectives:

- Circle movement
- Symbol X movement
- Straight line movement
- Using pen to implement circle drawing
- Using pen to implement symbol X drawing
- Using pen to implement straight line drawing
- Real-time control by hands
- Using grippers combined with real-time control by hands to grip the objects on the desk of random location

2 Design and Implementation

2.1 System Modelling (Design)

2.1.1 Dobot Magician

In this project, the tasks assigned for the robot arm are to realise circle drawing and movement, symbol X drawing and movement, and real-time control movement by hands combined with grippers and suction cup. Initially, the data transmission between the laptop and the robot arm is designed through microcontrollers. Consequently, the robot arm should be capable of high precision, high accuracy, repeatability, stable connection, compatibility for microcontrollers. Moreover, Dobot Magician is an industry-standard robot arm, and thus it can help the user to comprehend industrial robot arm in lower cost [10]. Compared to various robot arms, Dobot Magician is a multifunctional and educational robot arm and provides various superiorities for this project:

- Versatility: Dobot Magician is capable of performing high precise tasks such as circle drawing and real-time gesture control movement using depth cameras [10].
- High Precision and High Accuracy: Dobot Magician can achieve low error rate in position tasks [10]. Therefore, Dobot Magician can ensure the precision and accuracy of the position, and thus this robot arm is suitable for this project to execute the tasks such as circle drawing and real-time gesture control movement.

- Compatibility with microcontrollers: Dobot Magician is equipped with I/O pin to connect the microcontrollers such as ESP32, Arduino, Raspberry Pi and so on [11]
- Cost-effectiveness: As the budget for the individual project is limited and compared to other robot arms such as industrial robot arms, Dobot Magician is the most cost-effective robot arm. Moreover, the university has Dobot Magician already, so the budget of purchasing Dobot Magician can be saved.
- Friendly interface and diverse accessibility – Dobot Studio created by Dobot company has an easy-understanding interface and diverse functionalities as follows [12]:
    - Teaching and Playback – Record the position of Dobot Magician (Enhancement in ease of recording data)
    - Operation Panel – Display the pose of Dobot Magician directly
    - Write and Draw – Ease-understanding function to allow users to draw or write on paper by Dobot Magician
    - Blocky – Easy-comprehension for users to code in Python to control Dobot Magician
    - Script – Advanced coding platform to code in Python to control Dobot Magician

Based on the above superiorities, Dobot Magician is appropriate for this project, and thus it is used in implementing the tasks of circle drawing, symbol X drawing, real-time gesture control movement with depth camera and so on. Moreover, the workshop in University of Nottingham can provide Dobot Magician directly without purchasing online. It can save the budget for other use.

The following is the parameters of Dobot Magician, and the tools of Dobot Magician used in this project are also shown in following Figures:

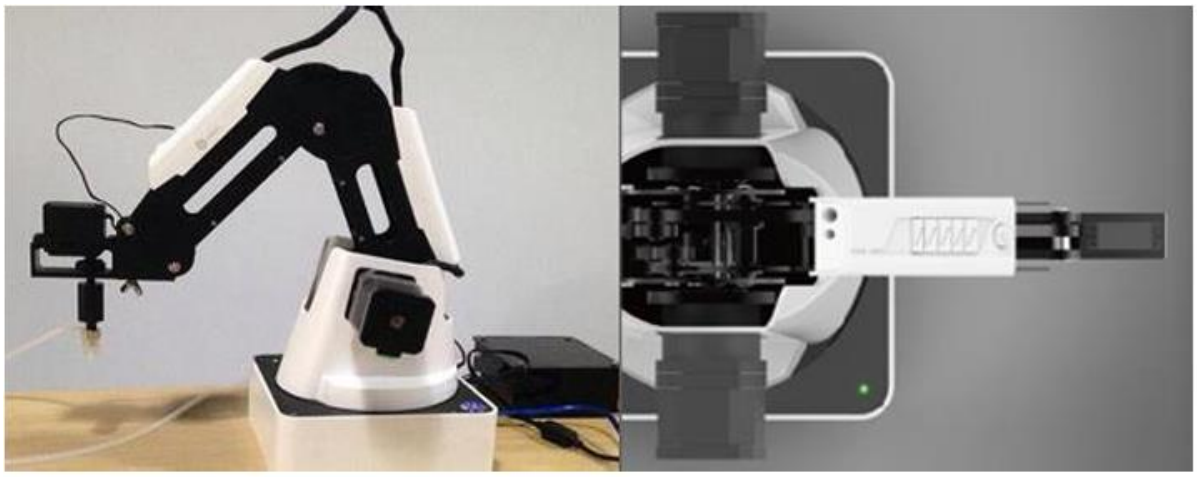

Figure 2.1.1.1. Dobot Magician [5]

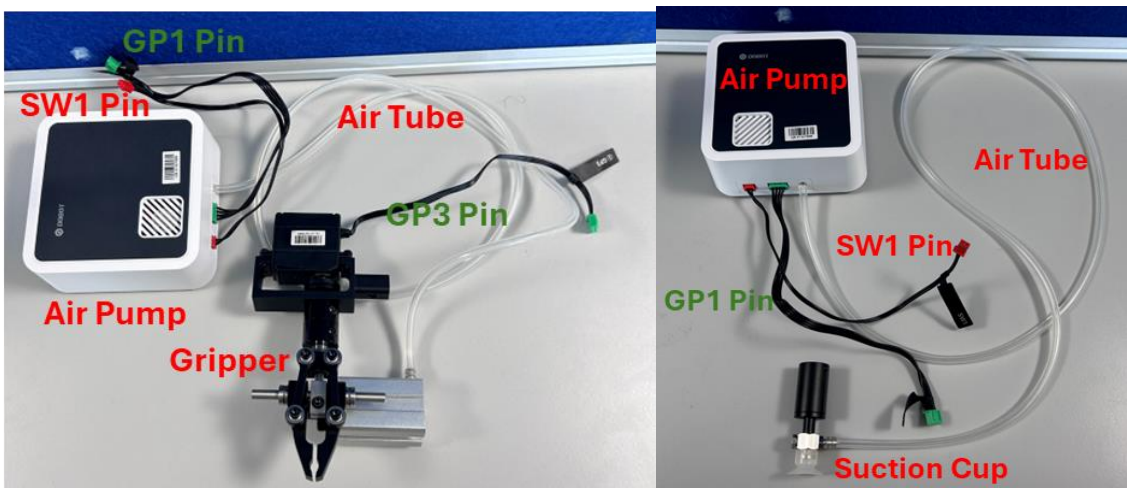

Figure 2.1.1.2. Gripper, Suction Cup and Air Pump

In this project, the gripper and the suction cup are used, and both of tools are needed to connect the air pump in Figure 2.1.1.2. The air pump is to rapidly evacuate the air from inside the pump to create negative pressure (a vacuum) so that the suction cup can absorb tightly to the surface of the object [12]. Conversely, the air pump stop evacuating the air to release the object [12]. For the gripper, the air pump through injecting compressed air into the cylinder through the air tube to push the piston inside the cylinder to open the gripper to release the object [12]. On the contrast, when the air is released from the cylinder, the gripper will close to grip the object tightly [12].

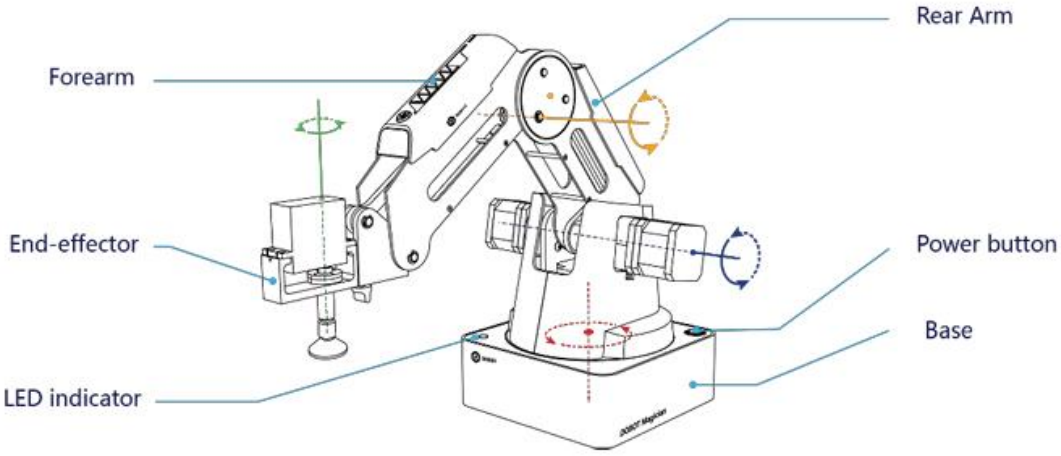

Figure 2.1.1.3. Schematic of Dobot Magician [12]

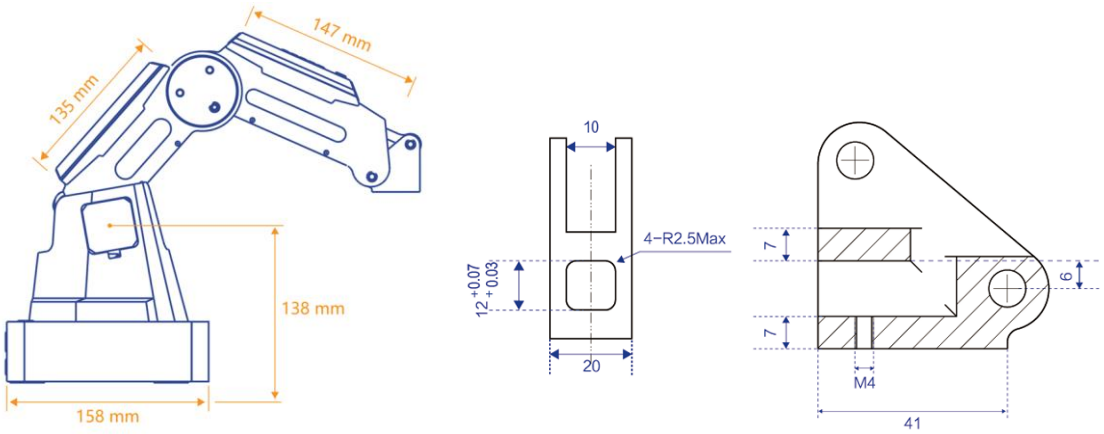

Figure 2.1.1.4. Length of Dobot Magician [12]

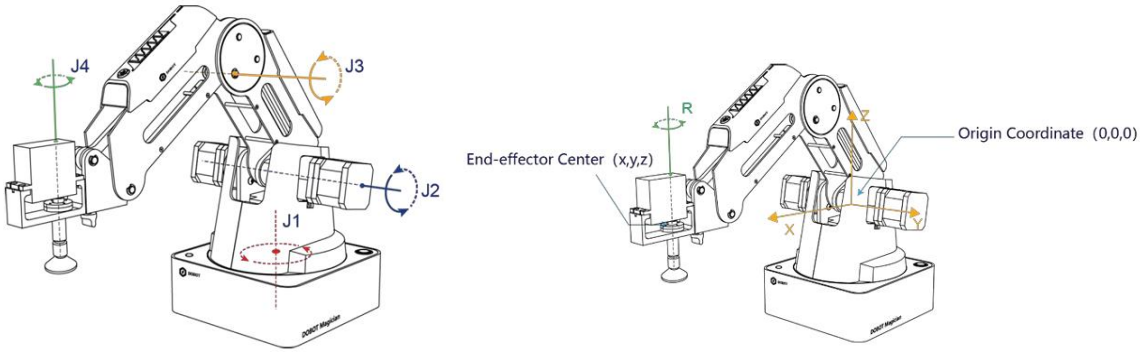

Figure 2.1.1.5. Joint Angles and Original Coordinate [12]

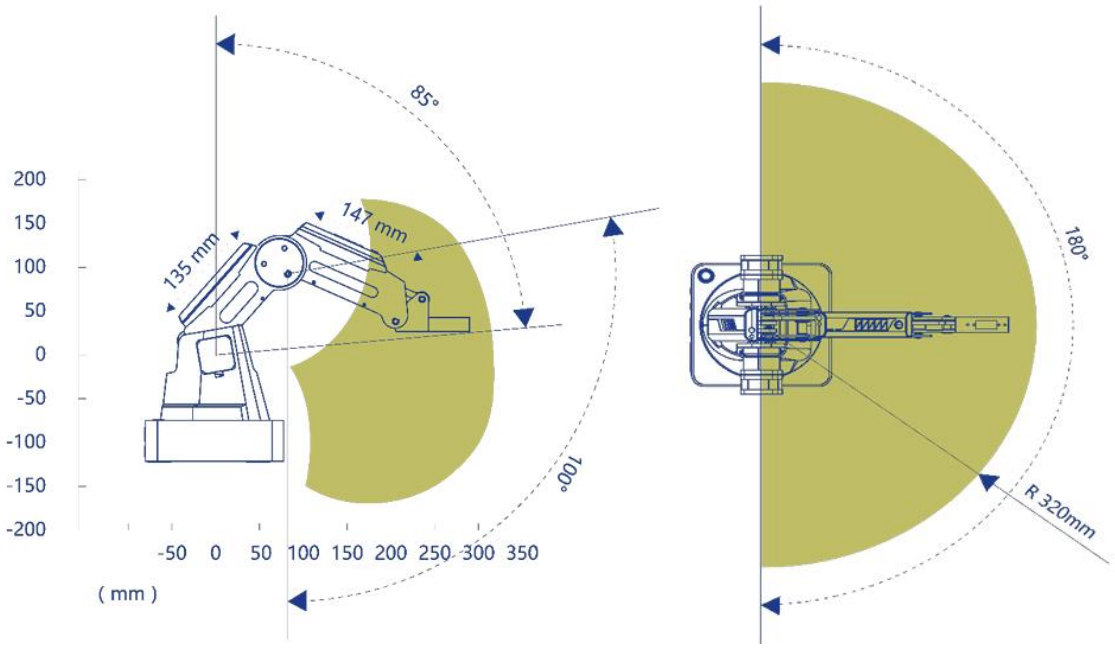

Figure 2.1.1.6. Joint Angles Range [12]

Based on the above figures, the parameters of Dobot Magician are shown in Table 2.1.1.1.

Table 2.1.1.1. Dobot Magician Parameters [12]

Maximum payload	500g	
Maximum reach distance	320mm	
Motion Range	Base (Joint Angle 1)	$-90^{\circ}\sim+90^{\circ}$
	Rear Arm (Joint Angle 2)	$0^{\circ}\sim85^{\circ}$
	Forearm (Joint Angle 3)	$-10^{\circ}\sim+90^{\circ}$
	End-effector rotation (Joint Angle 4)	$-90^{\circ}\sim+90^{\circ}$
Maximum speed (with 250g payload)	Rotational speed of Rear arm, Forearm and base	$320^{\circ}/s$
	Rotational speed of servo	$480^{\circ}/s$
Repeated positioning accuracy	0.2mm	
Power supply	100V-240V AC, 50/60Hz	
Power in	12V/7A DC	
Communication	USB, WIFI, Bluetooth	
I/O	20 extensible I/O interfaces	
Working temperature	$-10℃\sim60℃$	

2.1.2 Xbox One Kinect Motion Sensor

The use of the camera can help robot arms to sensing the environment and the gestures to realize the gesture-controlled robot arm, and it is a common combination with robot arms to achieve various gesture recognition and robot arm control [13, 14]. Based on the project specification, the camera used in this project should be equipped with the following characteristics:

- High resolution – More detailed recognition
- High accuracy and precision – Necessary for real-time recognition
- Depth recognition – Real-time gesture control
- Low cost – Limited budget
- Motion Capture – Gesture recognition
- Compatible with PC, laptop – As the connection between robot arm and the camera will through a laptop

Table 2.1.2.1. Xbox One Kinect Motion Sensor [15, 16]

Depth Sensor Type	Time of Flight (ToF)
Resolution of Red, Green and Blue (RGB) camera	1920x1080, 30 fps
RGB image field of view	$84.1^{\circ} \times 53.8^{\circ}$
Depth image field of view	$70^{\circ} \times 60^{\circ}$
USB	3.0

For the requirement of high resolution, the resolution of a webcam can also reach 1920x1080, and the cost of webcam is lower than the depth camera [17]. However, for the depth recognition requirement, the webcam is incapable of providing the depth information [17]. Moreover, compared to a webcam (2D) and a depth camera (3D), the depth camera provides better performance in recognition [18]. Therefore, depth cameras are more appropriate for this project. The common depth cameras are as follows:

- Xbox One Kinect Motion Sensor

  Xbox One Kinect motion sensor is able to perform the tasks such as gesture recognition [19]. This Kinect sensor can provide the superiority of high accuracy and precision compared to Xbox 360 Kinect Sensor [20].
- Xbox 360 Kinect Sensor

  Xbox 360 Kinect Sensor is able to implement gesture recognition [19]. Compared to eyebrow detection, Xbox 360 Kinect is better than Xbox One Kinect motion sensor [20].
- Intel RealSense

  Intel RealSense is also able to execute gesture recognition [21]. This depth camera can provide high accuracy, high efficiency, high resolution, depth recognition [21].
- Azure Kinect

  Azure Kinect DK is capable of gesture recognition [22], and it is also able to provide high accuracy and precision, high resolution and depth recognition [22].

Based on the above statement, all depth cameras are capable of implementing all tasks in this project. However, compared to the price of each depth camera, Xbox One Kinect Motion Sensor and Xbox 360 Kinect Sensor are more suitable for this

project on account of limited budgets [23-26]. Moreover, the workshop in University of Nottingham can provide Xbox One Kinect Motion Sensor and Xbox 360 Kinect Sensor directly without purchasing online. It can greatly save the budget for other use. Compared to Xbox One Kinect Motion Sensor and Xbox 360 Kinect Sensor, Xbox One Kinect Motion Sensor is more appropriate for this project than Xbox 360 Kinect Sensor since the ability of Xbox One Kinect Motion Sensor is superior to Xbox 360 Kinect Sensor [20]. Moreover, both cameras are borrowed from the workshop to determine which one is better. The self-testing result also proves that Xbox One Kinect Motion Sensor is better than Xbox 360 Kinect Sensor for this project as Xbox 360 Kinect Sensor is still unable to connect to the computer and the laptop through a USB adapter after a lot of attempts.

The number of the camera use can depend on the budget and the ability of the camera is sufficient or not. The capability of Xbox One Kinect Motion Sensor is sufficient to perform the tasks in this project, and the use of one camera can prevent extra expense and reduce the complexity of the system and power consumption to enhance the system's stability and reliability although the use of two cameras can improve the accuracy of gesture recognition [27].

2.1.3 Python vs C++ and others

According to [28], the top five popular languages are as follows:

1. Python
2. Java
3. JavaScript
4. C/C++
5. C#

Based on [29], the Dobot company provide the package to allow the user to control the Dobot in different interfaces and programming languages, and it supports the programming languages of Python, Java and C#. As a result, C/C++ and JavaScript are not suitable for this project. However, the project will focus on gesture recognition, and it can be achieved with the use of OpenCV [30]. According to [31], OpenCV supported languages include Python, C++, Java and MATLAB, and thus C# is not applicable in this project. Compared to Java and Python, Python is superior to Java in readability, simplicity, extensive libraries and integration capabilities [32], and therefore Python is the most appropriate language for this project.

2.1.4 Gloves vs Hands

Compared to the glove and the hand, the use of the glove can cause some challenges for gesture recognition [33]:

- Interface – some sensors might be placed on the glove
- Cleaning procedure – the glove is needed to clean before the next use
- Uncomfortable for users – wearing gloves can make users uncomfortable

Based on the above challenges, using a real hand without a glove is more appropriate for this project.

2.1.5 Teachable Machine

The use of machine learning in gesture recognition can extremely enhance the accuracy of gesture recognition to realize complex gesture recognition [34]. Compared to other machine learning, Teachable Machine, which is created by Google, is a free and user-friendly platform which allows users to train machine learning models online without further self-generation of the code [34]. As shown in Figure 2.1.5.1, the user only needs to upload the picture and can rename the class name to distinguish the type of recognition for the model [34]. Thus, Teachable Machine is used in this project to realize the complex gesture recognition and shape identification, and there are various cases proving the use of Teachable Machine can fulfil complex and customizes gesture recognition [34-36].

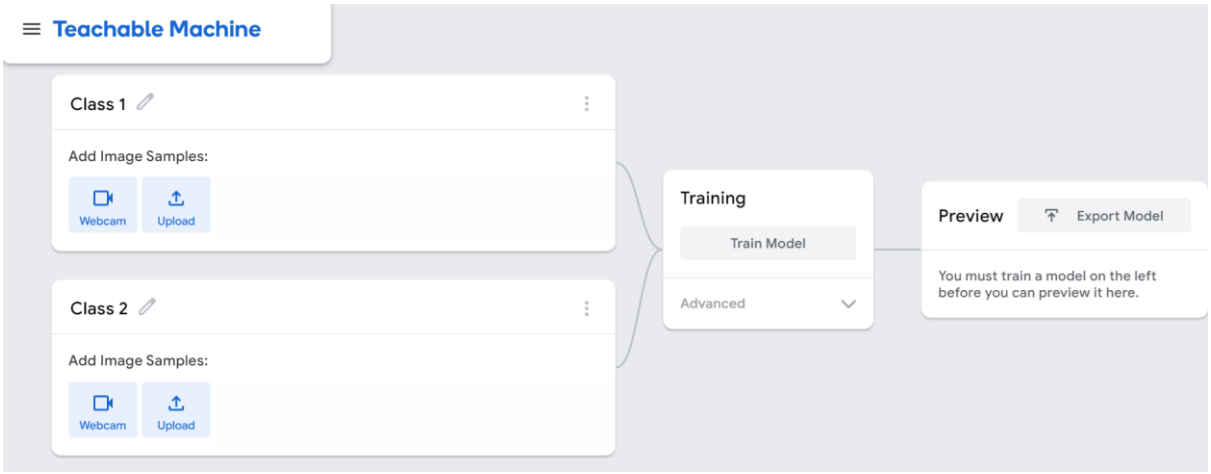

Figure 2.1.5.1 [37]

2.2 System Development (Implementation)

2.2.1 Libraries:

In this project, various Python libraries are used as follows:

2.2.1.1 OpenCV

OpenCV is created by OpenCV, and the package which is the crucial library in this project is used to combine with other libraries to realize the tasks such as image conversion, camera connection, putting texts on the frame, drawing the shapes on the frame and so on [38].

2.2.1.2 MediaPipe

The package “mediapipe” created by MediaPipe is used in this project to identify the human hand [39]. It will be used to combine the “cvzone” package to realize Hand Detection in section 2.2.2.1, and it is also utilized to provide the coordinates of the hands to implement Real-time Gesture Control in section 2.2.2.11.

2.2.1.3 CVZone

The package “HandTrackingModule” created by CVZone is used and modified to combine with the package “mediapipe” to obtain the information of the hands such as the hand position, right or left hand, hand landmarks, which fingers are up [39, 40]. The following function name and the variable name are the names in “HandTrackingModule” [40]. Moreover, the package “ClassificationModule” in CVZone [40] is also used to achieve Complex Gesture Recognition in section 2.2.2.7 and shape identification in Virtual Drawing in section 2.2.2.10.

2.2.1.4 DobotDllType

The package “DobotDllType” created by Dobot company is utilized to allow the user to control Dobot Magician in secondary [29], involving in the tasks of Virtual Drawing in section 2.2.2.10 and Real-time Gesture Control in section 2.2.2.11.

2.2.1.5 math and numbers

The libraries “math” and “numbers” are used to perform the math calculations and determine the number types of the variables in this project [41, 42].

2.2.1.6 numpy

The library “numpy” is used to execute the image processing such as creating image array, image cropping and resizing and so on [43].

2.2.1.7 TensorFlow

TensorFlow is a machine learning platform used in perception and language understanding tasks [44, 45]. The library is used to realize Complex Gesture Recognition in section 2.2.2.7 and shape identification in Virtual Drawing in section 2.2.2.10 to allow the system to read the machine learning training model [46].

2.2.1.8 os

The library is used to read the file from the device [47].

2.2.1.9 time

The library is used to obtain the time in the system to calculate the delay time [48].

2.2.2 Overall System Design (Methodology)

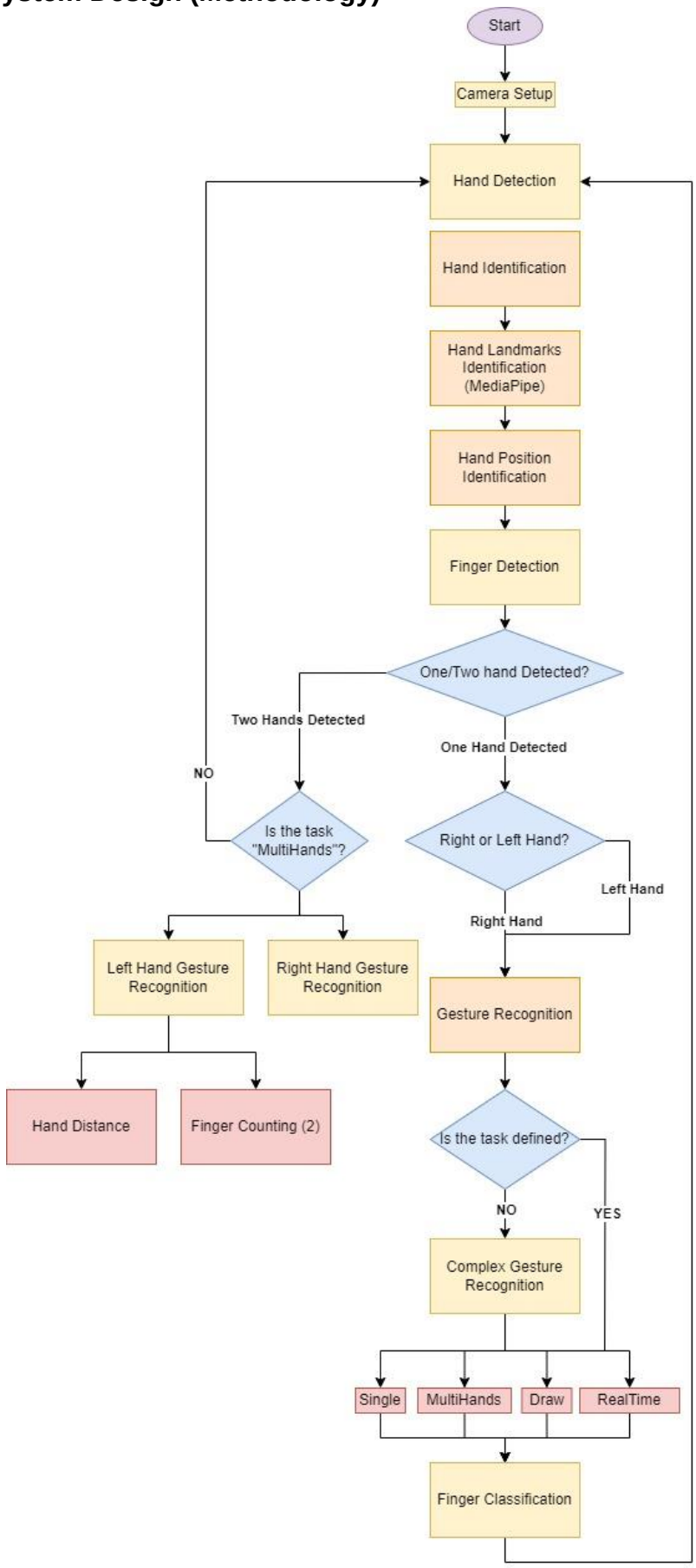

Figure 2.2.2.1. Overall System Flow Chart

The flow chat in Figure 2.2.2.1 illustrates how the system works, and the following explanation will explain the flow chart in detail.

2.2.2.1 Hand Detection

The process of Hand Detection is to detect the human hands on the frame and obtain the information of the hands such as left or right hand, hand landmarks, hand position, which fingers are up and so on.

2.2.2.2 Hand Identification

In this section, the process of how the human hand be identified by the camera, and how to recognise the characteristics of specified key points of hands. The key point to identify by MediaPipe is in Figure 2.2.2.2.1.

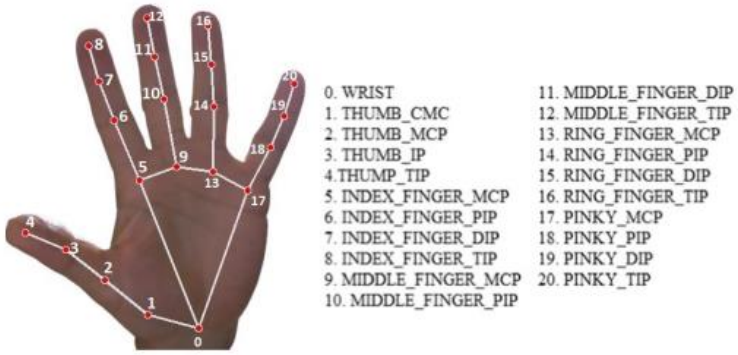

Figure 2.2.2.2.1. MediaPipe Hand Landmarks [49]

The coordinate of each landmark is also obtained by MediaPipe. The method to calculate the x and y coordinates of the landmark on captured frame is to normalize the image width (x) and height (y) to 0 to 1 [50]. The method is similar to [51]. To calculate the x coordinate, the model will measure the width of the camera and the width is normalized in the range of 0 and 1 (the nearest point is 0 and the farthest point is 1] and then the x coordinate is projected into the new range (0 to 1) [51, 52]. Similarly, to calculate the y coordinate, the model will measure the height of the camera, and the height is normalized in the range of 0 and 1 (the nearest point is 0 and the farthest point is 1) and then the x coordinate is projected into the new range (0 to 1) [50-52]. When the image is captured from the camera, the image is sent to “findHands” function to identify the human hand and draw the landmarks of virtual points on hands using line to connect each point as shown in Figure 2.2.2.2.2.

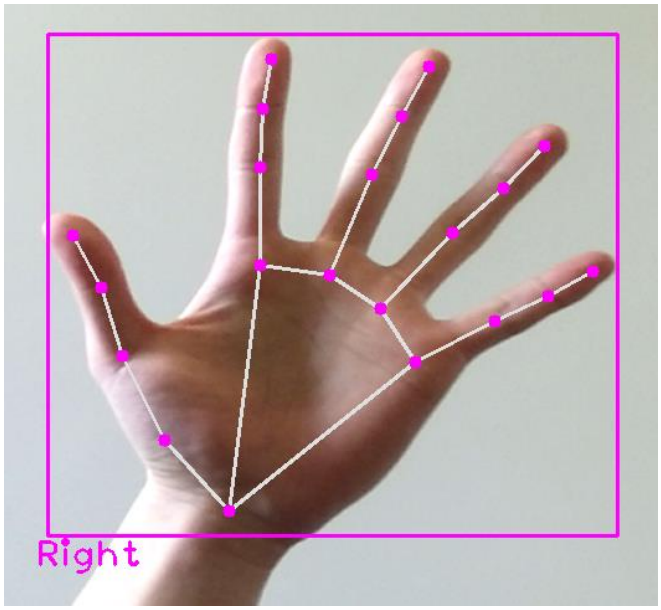

Figure 2.2.2.2.2. Hand Identification

Figure 2.2.2.2.3. BGR to RGB

In Figure 2.2.2.2.3, the image is converted from BGR (Blue, Green and Red) to RGB (Red Green Blue) which is the necessary process for hand detection using MediaPipe. After RGB formatted image is processed by MediaPipe hand detection model, the information of each hand including the coordinates of key points of the hand (landmarks), bounding box dimensions, and hand type (left or right) is extracted. The information is stored in a list for each hand which is in the variable “hands” and return to the caller.

As the flag “draw=True”, it allows the function visually annotates the identified hands on the original image by overlaying hand landmarks and connections, drawing rectangular bounding box around each hand, and the text " Left" or "Right" is displayed bear the bounding box to indicate the hand type. The visualisation provides an assistance for users to comprehend the detected hands and their spatial relationships within the image. Intuitively, the function “findHands” is used to identify the information of detected hands including their hand types, position on the image and landmarks of detected hands, and using text and bounding to highlight the detected hands. The function facilitates gesture recognition and hand tracking.

2.2.2.3 Hand Landmarks Identification (MediaPipe)

Figure 2.2.2.3.1. Flow Chart of Hand Lnadmarks Identification

The procedure of hand landmarks identification using MediaPipe is illustrated in Figure 2.2.2.3.1, and the processes include the use of Single Shot Detector (SSD) approach and deep learning architectures of convolutional neural networks (CNNs).

1. **Image Input**

   To facilitate the hand detection, the image should be converted from BGR to RGB as shown in Figure 2.2.2.2.3.

2. **Palm Detection**

   The process is to detect the palm in the frame by a pre-trained machine learning model "BlazePalm", which is trained by MediaPipe. Single Shot Detector (SSD) method, which is designed to detect the palm of the human hand, is used in BlazePalm model, which is trained by convolutional neural networks (CNN) [49]. As a result, if the palm is detected, the region of the hand will be identified by the bounding box from cropping the areas of the hand. The process of CNN is discussed in section 2.2.2.7.

3. **Hand Landmark Detection**

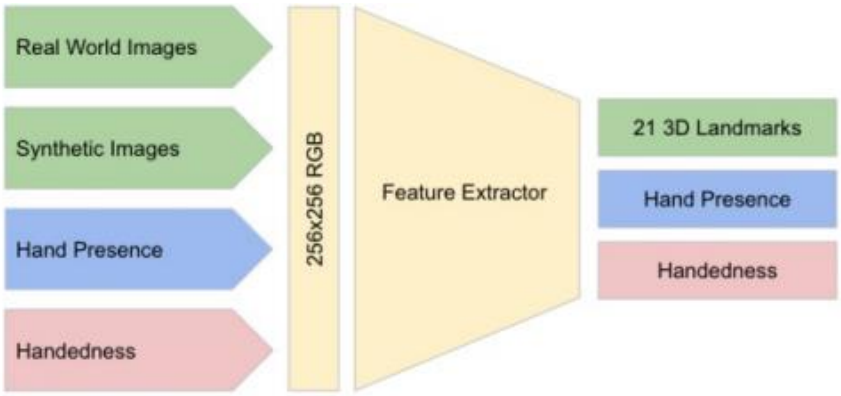

Figure 2.2.2.3.2. Hand Landmark Detection Model (MediaPipe) [52]

The bounding box identified in the previous stage is then used to identify the hand landmarks such as joints of fingers and the wrist [49]. The process of hand landmark detection is shown in Figure 2.2.2.3.2. After the process of feature extractor, 21 hand landmarks in 3D are identified [49]. In hand presence, the procedure is to determine whether the hand exists in the frame or not, and the handedness is to distinguish whether the hand is right hand or left hand [49]. Consequently, the hand landmarks will be identified as shown in Figure 2.2.2.2.1.

2.2.2.4 Hand Position Identification

The purpose of the process to recalculate x, y coordinates of hand landmarks although the coordinates are recorded through MediaPipe hand detection model and save the enhanced landmarks in variable “lmList”. The function “findPosition” is a combined method of the results from MediaPipe hand detection model in order that the process in “findPosition” function can only focus on the localized hand regions already detected by MediaPipe, and the image size is also considered. The process extracts the x and y coordinated saved in hand landmarks, and then x coordinates and y coordinates times the width and the height of the camera respectively to be more accurate coordinates and return to the caller. The calculation is as follows [40]:

$$new\ x = x * width\ of\ the\ camera \quad \text{eqn 2.2.2.4.1}$$

$$new\ y = y * height\ of\ the\ camera \quad \text{eqn 2.2.2.4.2}$$

The method will map the coordinates of hand key points to the actual length and width of the image [40].

2.2.2.5 Finger Detection

The aim of the process is to identify which fingers are up for each detected hand and store in an array “fingers” for the further use of gesture recognition. Based on Figure 2.2.2.2.1, each finger contains key points to be its tip and its base respectively, and the indices of the key points are stored in the following initialisation. In the initialisation, the ID of fingertips of thumb tip (4), index fingertip (8), middle fingertip (12), ring fingertip (16) and pinky tip (20) are initialised according to Figure 2.2.2.2.1 for the further process of gesture recognition.

For the fingers of index finger, middle finger, ring finger and pinky finger, the z-coordinate (depth coordinate) of its tip and base is the virtual value to distinguish whether the fingers are up or down. If z-coordinate of tip is less than the z-coordinate of base, the finger is considered straight; otherwise, the finger is considered bent.

To distinguish whether the thumb is up or down, the method makes use of the y-coordinate of its tip and base to determine the finger condition. If y-coordinate of tip is smaller than the y-coordinate of base, the thumb is considered to be straight; otherwise, the thumb is considered bent. However, this method is only suitable for the thumb of right hand since the method depends on the order of the specified points (landmarks) in the list. In right hand, the thumb order is from right wrist to the end of right thumb. In left hand, the thumb order is opposite is from the end of left thumb to the left wrist.

Consequently, the right hand is served as the major hand to implement the gesture recognition and all actions. Since the functionalities of the multi-hand detection and real-time control of Dobot Magician using both hands are also realised in this project, the function “fingerUpMulti” is created for left-hand detection. Moreover, to avoid unnecessary identification that the left-hand detection is gratuitous in the previous stage, two functions of “fingerUpMulti” and “fingerUp” are created, and it can also reduce the delay of the execution time of the action.

2.2.2.6 Gesture Recognition

In this section, the process of gesture recognition is explained in detail, and this is also the critical part of the whole system as the part is to recognise the action of user wanted to execute. Four options are provided for the users, and their corresponding conditions and gestures are shown in Figure 2.2.2.7.1. The fundamental technique

to identify complicated gesture is combined with the machine learning called “Teachable Machine “and the classifier in “cvzone” package of Python.

After the process of hand detection and obtained the necessary information of hands, the number of detected hands can be extracted from the necessary information of hands to recognise the number of hands are detected to be the condition for the following steps. If only one hand is detected, the distinction of left hand or right hand is conducted to implement the corresponding action.
The following processes can be divided into two parts. First, the frame captured from the camera is sent to the “gesture” function to check whether the particular operation is determined or not and the need to implement the corresponding tasks or not. The purpose of “gesture” function is to prevent incorrect detection in complex gesture recognition as the classifier in cvzone must distinguish the most similar gesture from the image as the prediction if the hand is detected. Therefore, the system is designed that the executed operation is not defined until the specific gesture is recognised in complex gesture recognition more than ten times. Moreover, after the action is defined, the corresponding function is called by the “gesture” function.

Second, if the certain operation is not defined, the operation of classification and complex gesture recognition are implemented. The classification function is to print out which fingers are up on the image through the information in “fingers” which is gained from the previous process of finger detection in section 2.2.2.5.

2.2.2.7 Complex Gesture Recognition

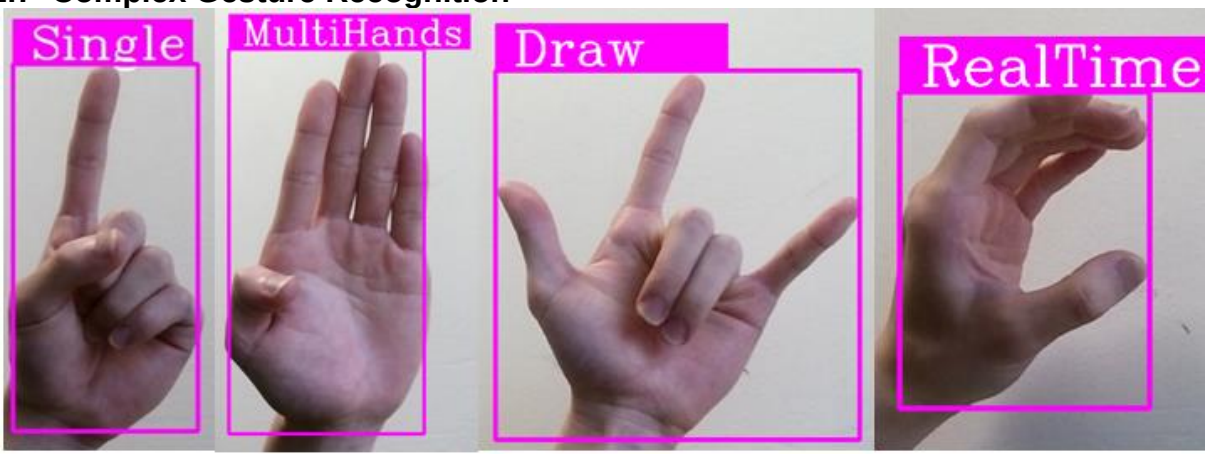

Figure 2.2.2.7.1. Gesture Options

The gesture options are shown in Figure 2.2.2.7.1. The ideas are learned from the courses on CVZone website, and the method is modified to realize complex gesture recognition [44, 53]. The process includes the training data collection, training model from Teachable Machine, image processing, Convolution neural network (CNN) architecture, image classification, and prediction and index. The Classifier class from the “cvzone” is used in the code. Using ClassificationModule module, the specified gestures can be recognised from the image [40].

Figure 2.2.2.7.2. Flow chart of Complex Gesture Recognition

Finally, complex gesture recognition is using the classifier in “cvzone” package to analysis the results in Teachable Machine to identify the gesture from the image. The process includes hand gestures training utilizing Teachable Machine and then through the image classification using the classifier in “cvzone” package to identify the most similar gesture from pre-trained gesture data to the gesture in the image.

- **Training Data Collection**

  The system is designed so that the webcam camera connection utilizes the function “VideoCapture()” in OpenCV, and the system will iteratively capture frames. Six types of gestures are trained, and each type collects approximately 20 images to create a dataset which is used for complex gesture recognition of providing options for users. The capture frames will be processed and saved in the particular folders when pressing the “s” button on the keyboard. Until the program is terminated manually, the loop is continuously collecting the data to be trained.

- **Image Processing**

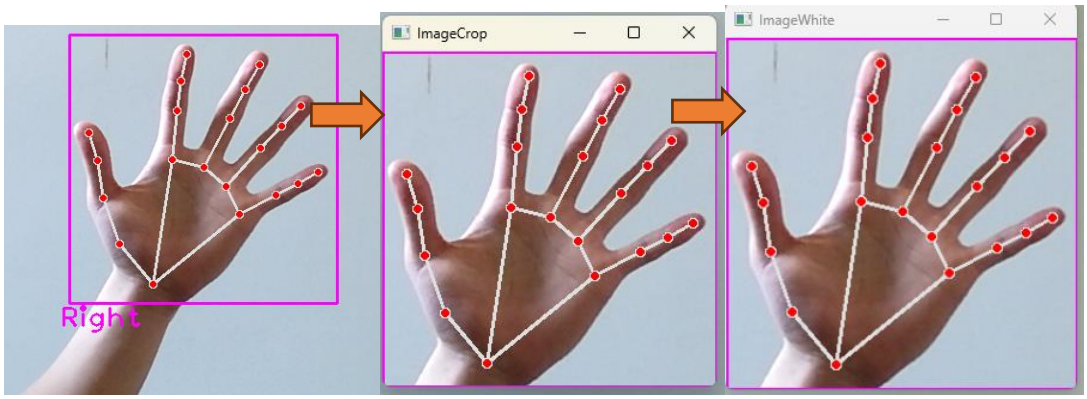

Figure 2.2.2.7.3. Flow Char of Image Processing

The flow chart of image processing is manifested in Figure 2.2.2.7.3. The image in the left is the captured frame from the camera, and each frame is analysised by the same method discussed in 2.2.2.2. After the process of hand idemtification, the image is then resized and cropped using OpenCV based on the bounding box edge gained from the return variable “hands” from hand detection module. Therefore, the region of detected hand is isolated and normalised [44]. The cropped image is displayed in the middle image of Figure 2.2.2.7.3, and the right image of Figure 2.2.2.7.3 is the combination of white background image and the processed image.

- **Data Model Training**

  Teachable Machine contains convolutional, pooling, and full connectivity [54], and can be used to train a machine-learning model to recognize the customized gestures imported by the user. In the training process, the libraries such as TensorFlow, Matplotlib, MediaPipe, OpenCV, and CVZone [44] is used to identify characteristics of edges, textures, and shapes from the image. After the training process is completed, the user can utilize the pre-trained model file "keras_model.h5", and index file "labels.txt" to identify the customized gestures imported by the users from the captured image and the classifier made by CVZone can return the prediction to recognise the specific gestures [54].

Convolutional Neural Network (CNN) architecture:

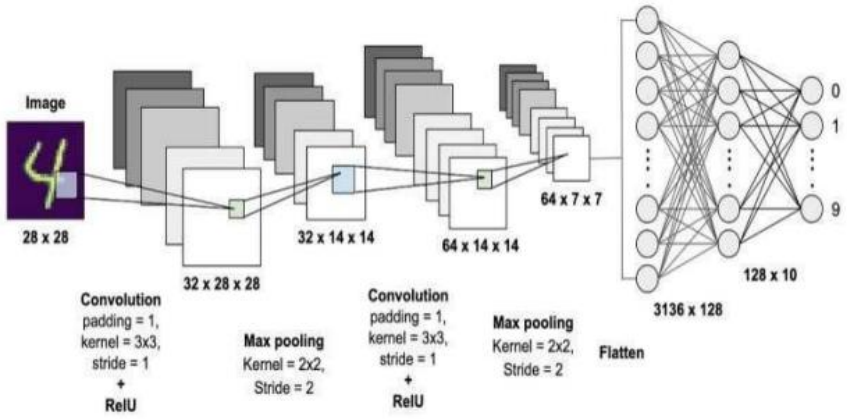

Figure 2.2.2.7.4. Convolutional Neural Networks (CNN) architecture [54]

Convolutional Neural Networks (CNN) are widely used for image identification, analyzing spatial connections and patterns [54]. The structure of CNN networks consists of layers for convolution, pooling and full connectivity. Teachable Machine trains the model input by the users using a comprehensive neural network configuration comprising 28 CNN layers as shown in Figure 2.2.2.7.4. Based on the input data from the user, the process automatically optimizes the network structure and setting so that the user can concentrate on analyzing data and results and the system manages the technical details simultaneously [54].

- **Convolutional Layer:**

  The filters (kernels) are used to detect low-level characteristics such as edges, textures, or shapes from the images, and then the output is served as feature maps [54]. The formula to compute a single element of a feature map is as follows [54]:

  $$Feature\ Map[i,j] = sum(Input[i+m, j+n] * Filter[m,n]) + Bias \quad \text{eqn.2.2.2.7.1}$$

  $Where$:

  $$Input: the\ input\ image\ ot\ previous\ layer's feature\ map$$
  $$Filter: Convolutional\ Filter$$
  $$Bias: Bias\ term$$

- **Activation Function:**

  In general, after each convolutional layer, an activation function is employed elementwise to the feature maps to incorporate non-linear characteristics [54]. The Rectified Linear Unit (ReLU) is frequently utilized as an activation function [54]:

  $$Output = \max(0, Feature\ Map) \quad \text{eqn. 2.2.2.7.2}$$

- **Pooling Layer:**

  Pooling layers are used to diminish the spital size of feature maps, which can minimize computation and the risk of overfitting [54]. Max-pooling is widely used [54]:

  $$Output[i,j] = \max(Feature\ Map[i*stride: i*stride + pool\ size, j*stride: j*stride + pool\ size]) \quad \text{eqn. 2.2.2.7.3}$$

  $Where$:

  $$Pool\ size: Size\ of\ pooling\ window$$

$$stride: Step\ size\ for\ moving\ the\ window$$

- **Fully Connected Layer:**

  The output through the process of the convolutional and pooling is transformed into a flat structure, before the output is input into one or more dense (fully connected) layers [54]. The process is to decode more complicated properties and arrive at the final decision for classification [54]. The equation used to calculate the output of a single neuron is [54]:

$$Z = (Input * Weights) + Bias \quad \text{eqn. 2.2.2.7.4}$$

$$Activation = activation\ function(Z) \quad \text{eqn. 2.2.2.7.5}$$

  The activation function might include a sigmoid, softmax (for multi-class classification), or another appropriate function [54].

- **Sofemax Layer (for multi-class classification):**

  Within the ultimate softmax layer, the softmax activation function recalibrates neuron scores (logits) into class probabilities [54]. The approach is widely used for classification into multiple categories [54].

$$ProbabilityClass\_i = \frac{\exp(LogitClass\_i)}{sum(\exp(Logits))Model\ Train} \quad \text{eqn. 2.2.2.7.6}$$

- **Image Classification**

The previous procedure of image processing is similar to the previous process of image processing discussed in the previous section. First, for the function of complex gesture recognition, the size of the analysed image is needed to be set to 300 x 300 pixels since the setting can streamline the process of gesture analysis in the classifier to ensure the input images to the classifier are consistent. Next, a white background image is created to be used as the background for the resized gesture image, and the size is the same as the input images. The size of the original image is also needed to be resized to crop the area of the detected hand to improve the accuracy of the recognition. The approach utilizes the “findHand” function discussed in section 2.2.2.2 and extracts the bounding information, and then based on its aspect ratio to ensure the priority of width or height to resize the image. The processed image is then placed on the white background image created in the previous stage to become a new cropped image and maintain the proportions of the gesture image for the further gesture recognition.

The further gesture recognition is to use the classifier from “cvzone” package to analysis and obtain the prediction of the gesture. The combined image of white background image and cropped image is served as an input to the classifier from “cvzone”. The function “getPrediction()” in Classifier class of “cvzone” is used to calculate the prediction and relevant indexes and then precisely recognise the specified gestures from the training model [40]. The process is to analysis the pre-trained model file keras_model.h5 and index file labels.txt obtained from Teachable Machine [54].

2.2.2.8 Finger Counting

The process of task of finger counting is shown in the following flow chart in Figure 2.2.2.8.1. The ideas of finger counting are learned from the courses on CVZone website, and the method is modified to realize complex gesture recognition and finger counting [53, 55].

Figure 2.2.2.8.1. Finger Counting Flow Chart

Based on Figure 2.2.2.8.1, the example process of finger counting is shown in Figure 2.2.2.8.2.

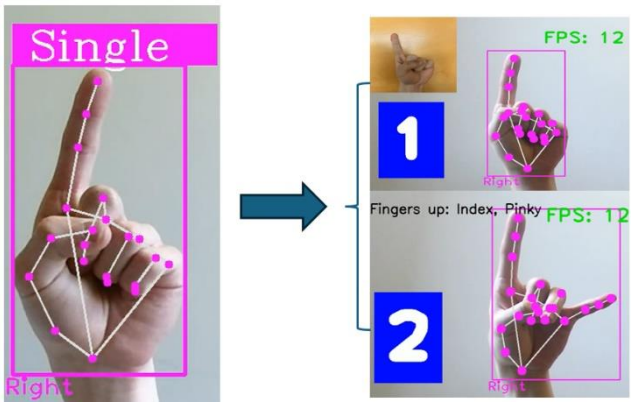

Figure 2.2.2.8.2. Example process

1. **Complex Gesture Recognition: Single**

Figure 2.2.2.8.3. “Single” Gesture

The procedure of complex gesture recognition is explained in section 2.2.2.7. After the “Single” gesture is recognised continuously ten times as shown in Figure 2.2.2.8.3, the system will stop recognising the gesture and the motion is defined to implement “Single” task.

2. **Finger Detection**

   This procedure is the same as the process explained in section 2.2.2.5 to obtain which fingers are up, and the results will be served as a standard in step 4.

3. **Total Finger Calculation**

   To count the total number of fingers are up, the function “count()” which is a built-in function in Python is utilized to calculate total number of elements in the array of the results gained in step 2.

4. **Does pointing-up finger match**

   The process is utilized the difference in the array element obtained in step 2 to distinguish the type as shown in Figure 2.2.2.8.1:

   - A - No finger (0)
   - B - Index finger up (1)
   - C - Index and middle fingers up (2)
   - D - Index, middle and ring fingers up (3)
   - E - Index, middle, ring and pinky fingers up (4)
   - F - Thumb, index, middle, ring and pinky fingers up(5)
   - G - Thumb and pink fingers up (6)
   - H - Thumb and index fingers up (7)
   - I - Thumb, index and middle fingers up (8)
   - J - Thumb, index, middle and ring fingers up (9)

   If the fingers do not match, it will implement step 5 directly.

5. **Display total fingers and Finger Classification**

   The process is to display the results of total number of pointing-up fingers calculated in step 3 and using “classification” function to extract the data in an array gained from step 2 to show which fingers are up or no fingers up on the screen.

2.2.2.9 Multi Hand Recognition

The following flow chart in Figure is shown the process of Multi Hand Recognition, and the ideas of finger counting (Multi-hand) and Hand Distance are learned from the courses on CVZone website, and the method is modified to realize complex gesture recognition, finger counting and Hand Distance measurement [53, 55, 56].

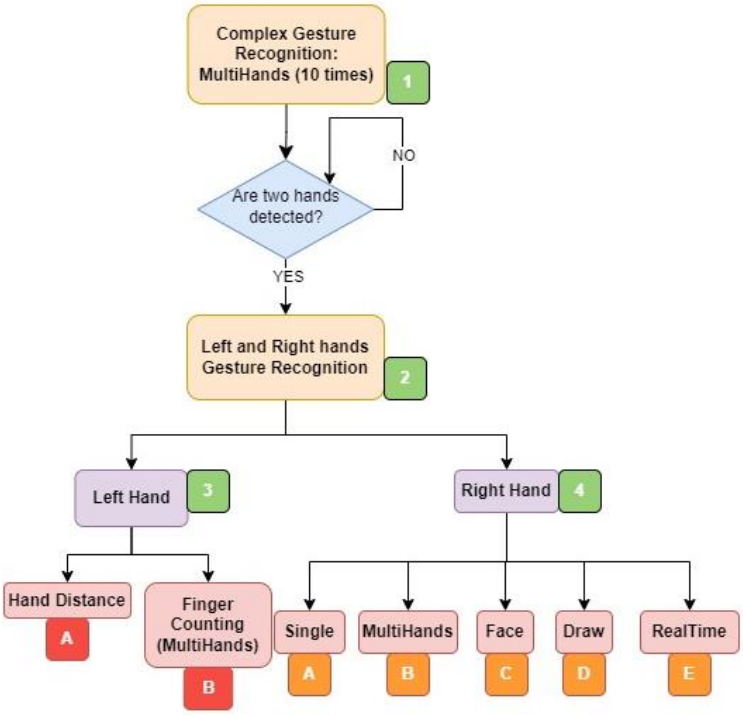

Figure 2.2.2.9.1. Flow Chart of MultiHands

1. **Gesture Recognition: MultiHands**

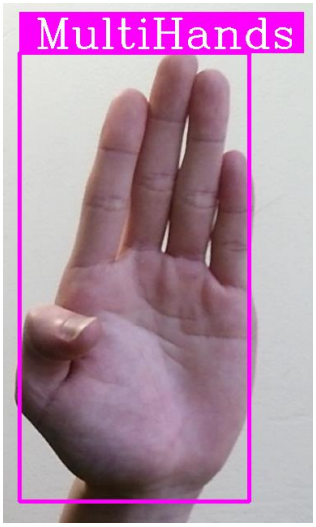

Figure 2.2.2.9.2. “MultiHands” Gesture

The procedure of complex gesture recognition is explained in section 2.2.2.7. After the “MultiHands” gesture is recognised continuously ten times as shown in Figure 2.2.2.9.2, the system will stop recognising the gesture and the motion is defined to implement “MultiHands” task.

2. **Left and Right hands Gesture Recognition**

The theory is the same as complex gesture recognition, but the models are different. The model for right hand gesture recognition is the same as complex

gesture recognition in Figure 2.2.2.7.1, and the model for left hand gesture recognition is distinctive as shown in Figure 2.2.2.9.3 and Figure 2.2.2.9.5. Both hand gestures are recognised simultaneously in Figure 2.2.2.9.3 and Figure 2.2.2.9.5. However, the design of right-hand gesture recognition will not stop until the action from left hand recognition is defined.

3. **Left Hand**

For the left-hand recognition, the design is similar with complex gesture recognition that the recognition process will stop until the same gesture is recognized ten times continuously. The options are shown in Figure 2.2.2.9.3 and Figure 2.2.2.9.5, after the action is defined the corresponding tasks will be executed.

A. Hand Distance

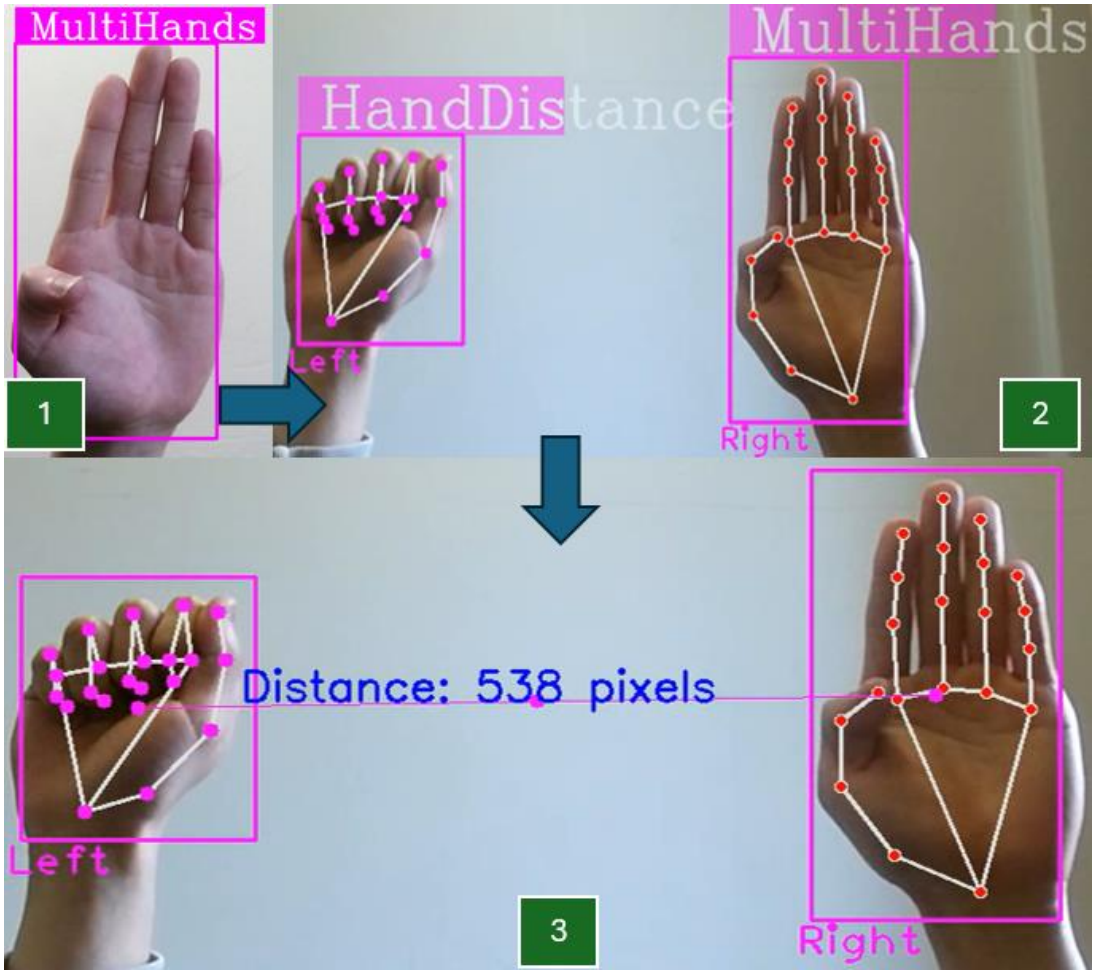

Figure 2.2.2.9.3. Example Process of Hand Distance

The example process is shown in Figure 2.2.2.9.3. The measurement of hand distance is to extract x and y coordinate of both hands from the hand information which includes the data from “findHands” function, and then add both x value and y value to average to calculate the centre point. Next, Pythagorean theorem is used to calculate the straight-line length of two centre points as shown in Figure 2.2.2.9.4. Then, the centre point of two hands is connected using a straight line and the centre point of the straight line is indicated by a circle as shown in Figure 2.2.2.9.3.

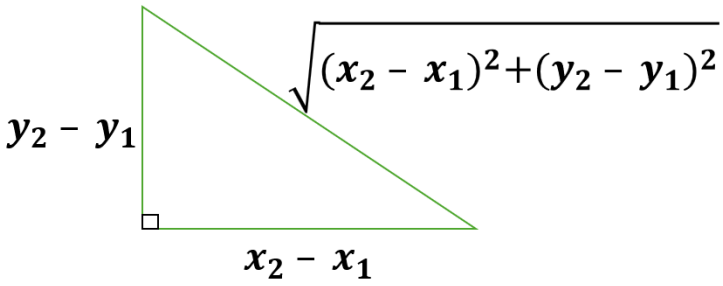

Figure 2.2.2.9.4. Pythagorean theorem

B. Finger Counting (MultiHands)

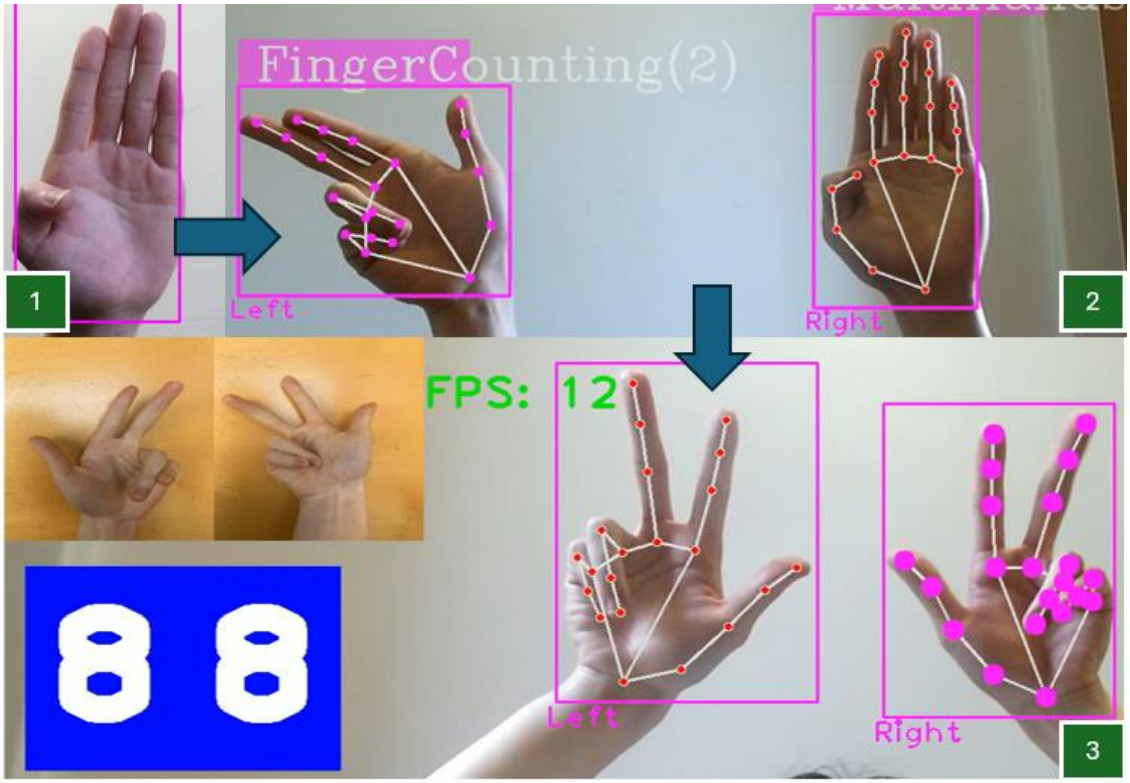

Figure 2.2.2.9.5. Example Process of Finger Counting (MultiHands)

The example process is shown in Figure 2.2.2.9.5. The principle of two-hands finger counting is the same as the previous finger counting which is explain in section 2.2.2.8. The only difference is that the number of hands is changed from one to two hands to realize 0 to 99 counting by hands. However, if one or two hand gestures do not match the standard gesture of finger counting, the number of fingers and which fingers are up are similarly displayed on the screen.

4. Right Hand

The gesture recognition of right hand is the same as complex gesture recognition which is explained in section 2.2.2.7. The purpose of right-hand gesture recognition is to achieve two-hands recognition simultaneously. However, the right-hand gesture recognition will not be stopped until the left-hand gesture recognition is finished.

2.2.2.10 Virtual Drawing

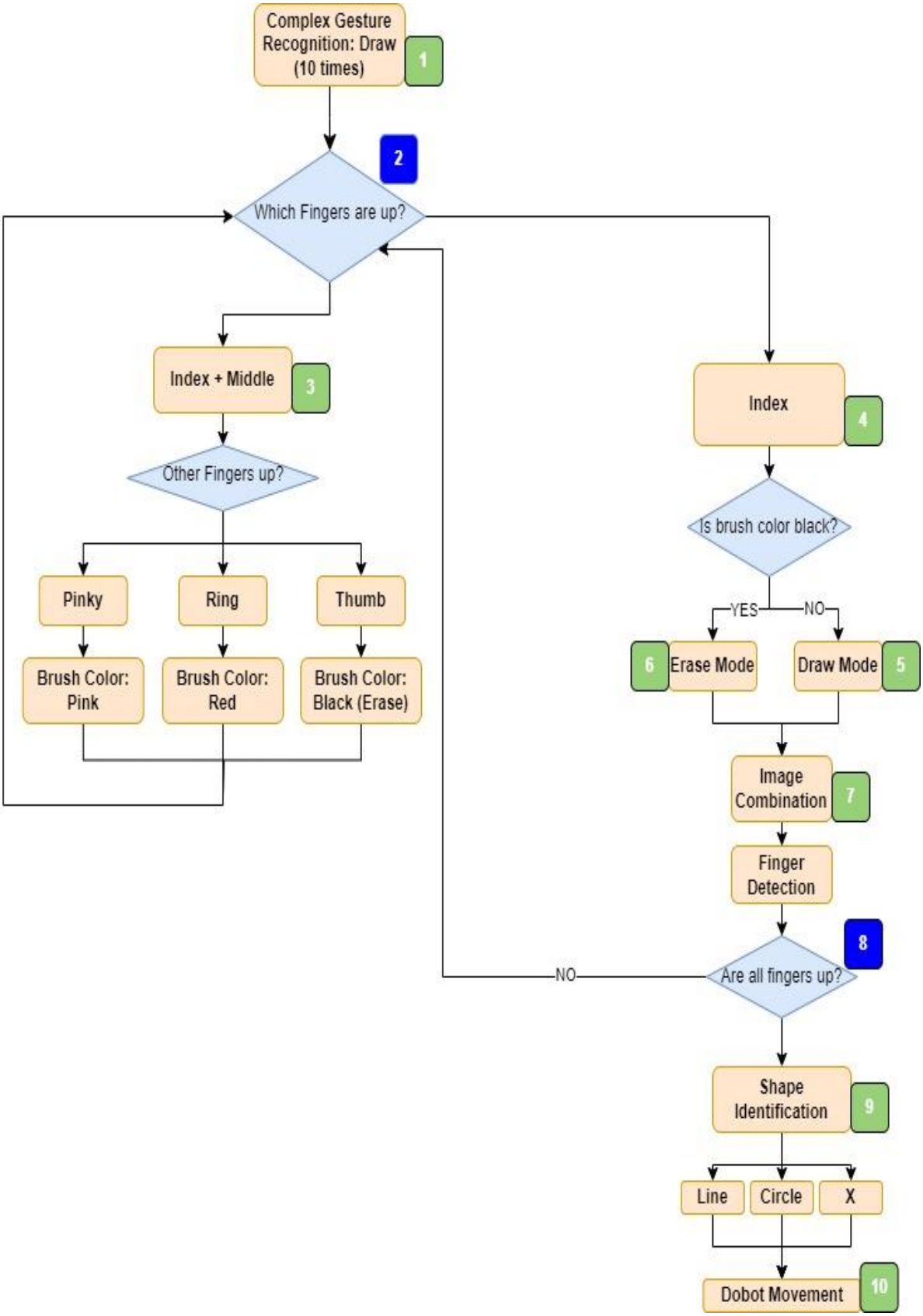

Figure 2.2.2.10.1. Flow Chart of Virtual Drawing

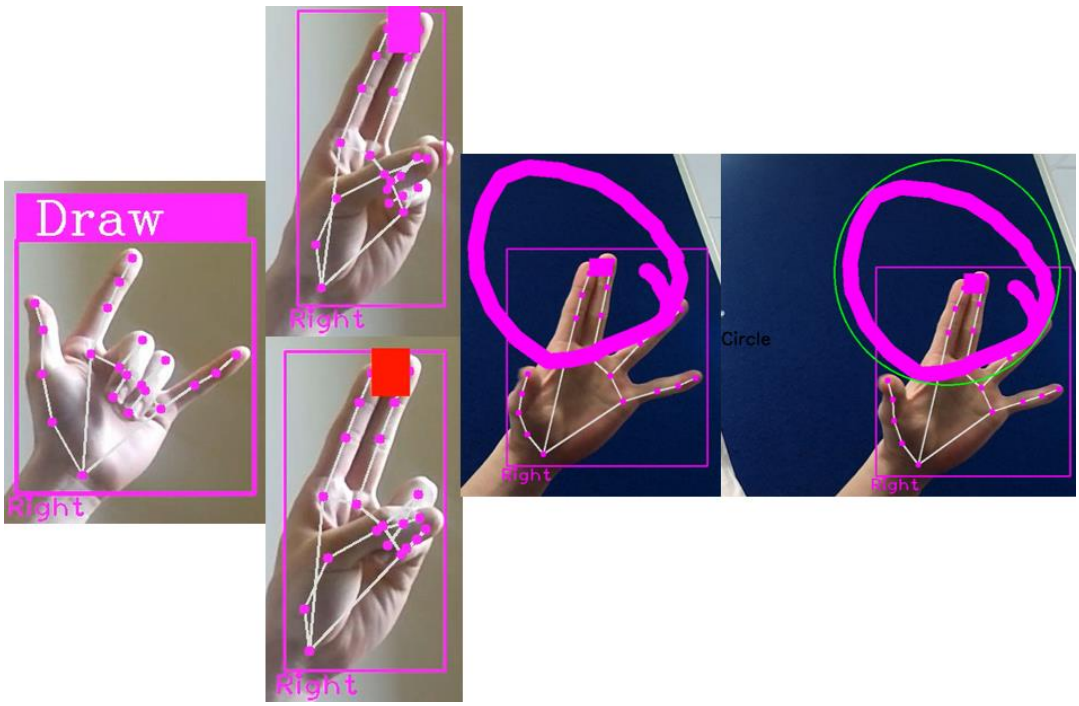

Figure 2.2.2.10.2. Example process of Virtual Drawing

The flow chart in Figure 2.2.2.10.1 indicates the process of Virtual Drawing, and the example process with draw mode and shape identification is shown in Figure 2.2.2.10.2. The ideas are learned from the courses on CVZone website and the academic research to modify to fulfil Virtual Drawing [57, 58]. After the shape is recognised, the data will be sent to Dobot as shown in Figure 2.2.2.10.3. The Figure 2.2.2.10.3 is an example of drawing, and three shapes can be drawn in this project.

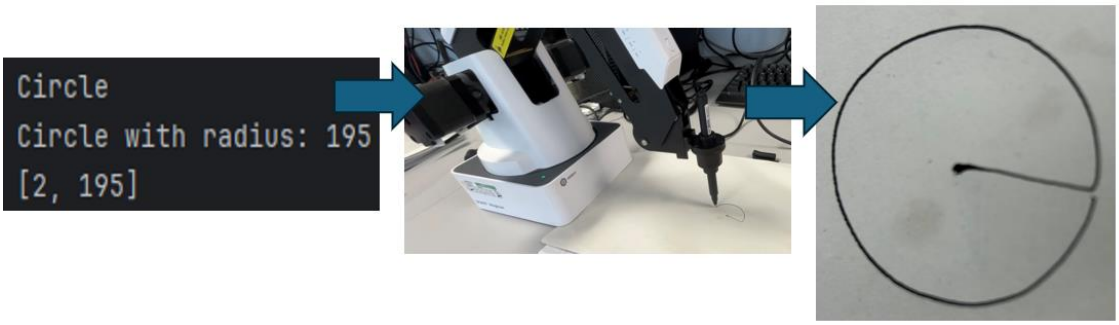

Figure 2.2.2.10.3. Example Process of Dobot Implementation

1. Complex Gesture Recognition: Draw (10 times)

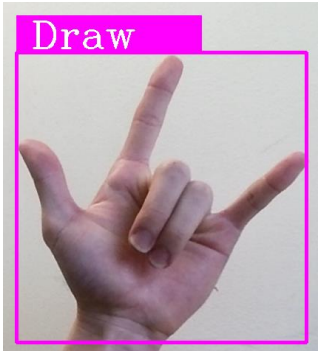

Figure 2.2.2.10.4. “Draw” Gesture

The procedure of complex gesture recognition is explained in section 2.2.2.7. After the “Draw” gesture is recognised continuously ten times as shown in Figure 2.2.2.10.1, the system will stop recognising the gesture and the motion is defined to implement “Draw” task.

2. **Which Fingers are up**

   The process is to use Finger Detection, which is explained in section 2.2.2.5, to change the colour of the brush, draw a shape on the frame, erase the shape on the image or identify the shape on the frame.

3. **Index Finger and Middle Finger are up**

   This process is select mode to allow the user to change the colour of the brush. The defaulted colour is pink, if the ring finger is also up and detected in “Other fingers up?”, the colour is changed to red. If the pink finger is up and detected in “Other fingers up?”, the colour of the brush is changed back to pink. However, if the thumb is up and detected in “Other fingers up?”, the colour of the brush is changed to black, and it is utilized to erase the drawn route, and the process will be explained in step 5.

4. **Only Index Finger is up**

   This process is to distinguish draw mode and erase mode by Finger Detection and the colour of the brush. If the colour of the brush is not black and only index finger is up, the system will enter step 6. Conversely, if the colour of the brush is black and only index finger is up, the system will implement step 5. Moreover, the extra canvas image is created to record the drawing route and then will be combined with the original image to realize the virtual painter.

5. **Draw Mode**

   If draw mode is active, the position of the route of index-finger movement is saved in an array continuously, and then utilize OpenCV function "cv2.line" to print the route on the image and the canvas. Next, renew the initial point to the actual point, since the OpenCV function is used point-to-point to print the line as shown in Figure 2.2.2.10.2, otherwise the line will always connect the original point.

6. **Erase Mode**

   The theory is quite similar to the draw mode. The position of the route of index-finger movement is also printed on the image and the canvas. However, the position will not be saved in the array like the draw mode. The erase mode is to overwrite the pixels in the canvas with background colour which is black to realize removing the drawn lines as shown in Figure 2.2.2.10.5.

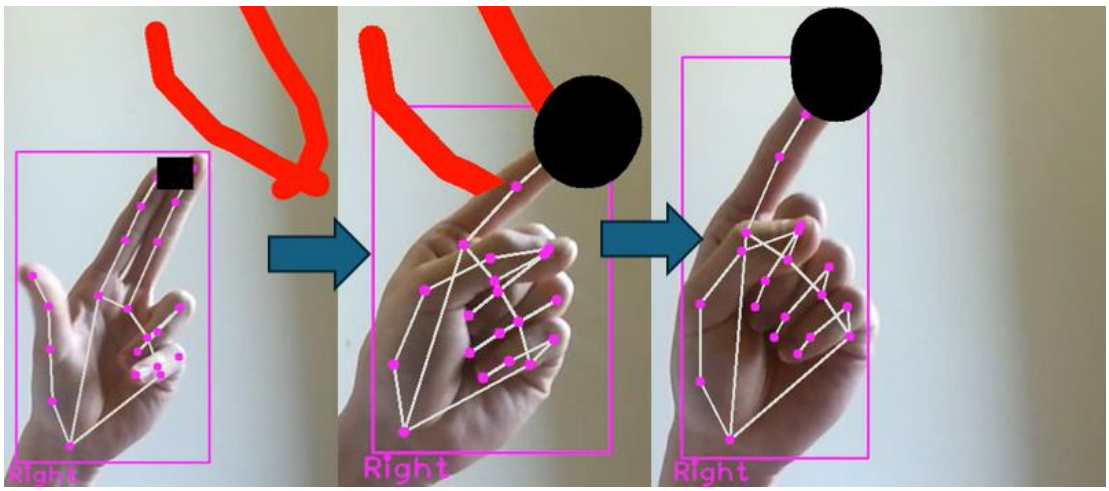

Figure 2.2.2.10.5. Erase Mode

7. Image Combination

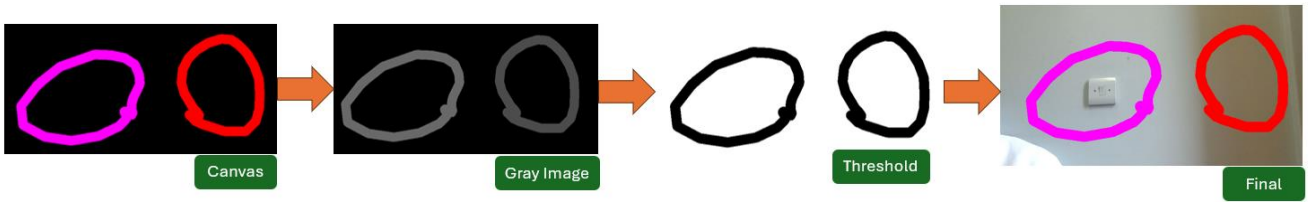

Figure 2.2.2.10.6. Image Combination Process

The process is shown in Figure 2.2.2.10.6. The canvas with black background is converted to the grey image. Then, the OpenCV function “cv2.threshold()” to convert the grey image into threshold image by setting to 0 if pixels are smaller than threshold which is black colour and setting to 255 if pixels are greater than threshold which is white colour. Finally, combining all images to be the final image as shown in Figure 2.2.2.10.6.

The original method is used OpenCV function “cv2.addWeighted()” to combine the canvas and the actual image (original camera frame). However, the result is not appropriate since the image is like a virtual image as shown in Figure 2.2.2.10.7.

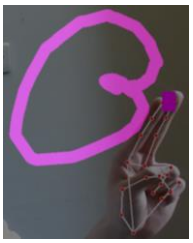
Figure 2.2.2.10.7. Original Image Combination

8. Are all fingers up

Finger Detection, which is explained in section 2.2.2.5, is used in this procedure to be a flag of shape identification.

9. Shape Identification

After the detection of five fingers up (Thumb, index, middle, ring, pinky), the process of shape identification is active. To identify the shape, the image is needed to convert to the grey image and then use OpenCV function to implement threshold operation. Next, using OpenCV function “cv2.findContours()” to find the contours in the binary image. Then, the pre-trained machine learning model from

Teachable Machine is used to recognize the shape on the frame, after the shape is defined, the system will analysis the shape.

If the shape is circle, the radius of the circle will be calculated by using OpenCV function “cv2.minEnclosingCircle()” to find the centre point of the circle and its radius. Since the circle is drawn by the user on the frame, the shape of the circle might not be perfect. As a result, the function “cv2.minEnclosingCircle()” is to use the circle with the minimum radius that can contain all the drawn shape. Finally, the smallest circle will be printed on the frame as shown in Figure 2.2.2.10.3, and the data will be sent to Dobot Magician to draw a circle on the paper in Figure 2.2.2.10.11.

If the shape is “X”, the length of “X” in one side will be calculated by using OpenCV function “cv2.boundingRect()” to use the smallest rectangle to cover all the drawn shape, and then divide the width of the rectangle by two to obtain the length of the symbol “X” since the symbol "X” is assumed to be symmetrical. To find the centre point of the symbol “X”, the width of the rectangle and the height of the rectangle are divided by two to be the centre x and centre y of the symbol “X” respectively. At the end, a perfect “X” symbol will be drawn on the frame as shown in Figure 2.2.2.10.14 and then send the data to Dobot Magician to draw a “X” on the paper in Figure 2.2.2.10.14.

The theory to calculate the length of the line on the frame is similar to the theory of the symbol “X”. After the identification of the shape of the line, OpenCV function is used again to use the smallest rectangle to cover all the drawn shape, and then obtain the width and the height of the rectangle. Next, using Pythagorean theorem:

$$length\ of\ the\ line = \sqrt{(Width)^2 + (Height)^2} \qquad \text{eqn. 2.2.2.10.1}$$

At the end, the centre point of the line is calculated by the width of the rectangle and the height of the rectangle are divided by two to be the centre x and centre y of the line. The rectangle and the centre point will be printed on the frame and send the data to Dobot Magician to draw a line on the paper.

The original method is used the number of vertexes to distinguish the shape on the frame is circle, line, or symbol “X”. To identify the shape, the image is also needed to convert to the grey image and then use OpenCV function to implement threshold operation. Next, using OpenCV function “cv2.findContours()” to find the contours in the binary image. To identify the number of vertexes, the perimeter is needed to calculate in advance by OpenCV function “cv2.arcLength”, and then using the function “cv2.approxPolyDP” to streamline the contours to calculate the vertexes. Finally, use the function “len()” to obtain the number of the vertexes to be the critical to determine its shape. If the number of the vertexes is less than three, it will be served as a line. If the number of the vertexes is larger than eight, it will be served as the circle. If the number of the vertexes is eight, it will be served as the symbol “X”.

10. Dobot Movement

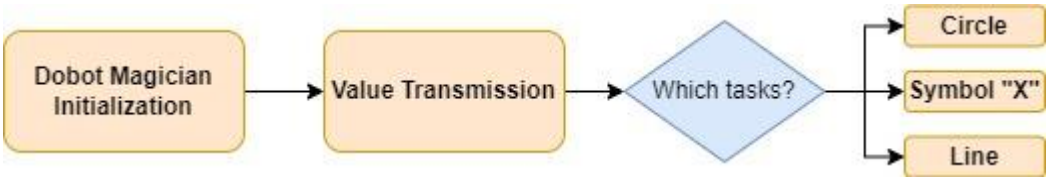

Figure 2.2.2.10.8. Flow Chart of Dobot Movement

The process of Dobot Movement is shown in Figure 2.2.2.10.8.

Dobot Magician Initialization:

The connection between Dobot Magician and the laptop is using a physical USB wire to connect. The status of connection with Dobot Magician will be checked by a connection string “CON_STR” [29]:

- DobotConnect_NoError - Successful connection
- DobotConnect_NotFound - Dobot is not detected
- DobotConnect_Occupied - Dobot is currently used by other processes

Next, the library “DobotDllType”, which is created by Dobot company to allow user to control and connect the robot arm produced by Dobot company in Python [29], is imported and load API (Application Programming Interface) using dType.load(). Then, the code can connect to the Dobot using "dType.ConnectDobot(api, "", 115200)" where "api" is API reference, "" is an empty string presumably for the IP address or identifier, and 115200 is the baud rate [59]. The result of this attempt is stored in "state", and it is sent to the console, indicating whether the connection was successful based on the earlier defined CON_STR [59]. After the initialization and the setting for the connection of Dobot, the Dobot Magician can be controlled using Python codes [29]. Moreover, the movement of Dobot in following processes is used the point-to-point connection to draw the shape, and there are four options provided by Dobot company in Table 2.2.2.10.1 [12]:

Table 2.2.2.10.1. Dobot Movement [12]

Mode	**Appropriate Application**
MOVL	Straight line movement
MOVJ	Trajectory is not required, but high speed is needed
JUMP	Movement of two points is needed to raise to specific height such as suction cup application, gripper application
ARC	Trajectory is required as an arc

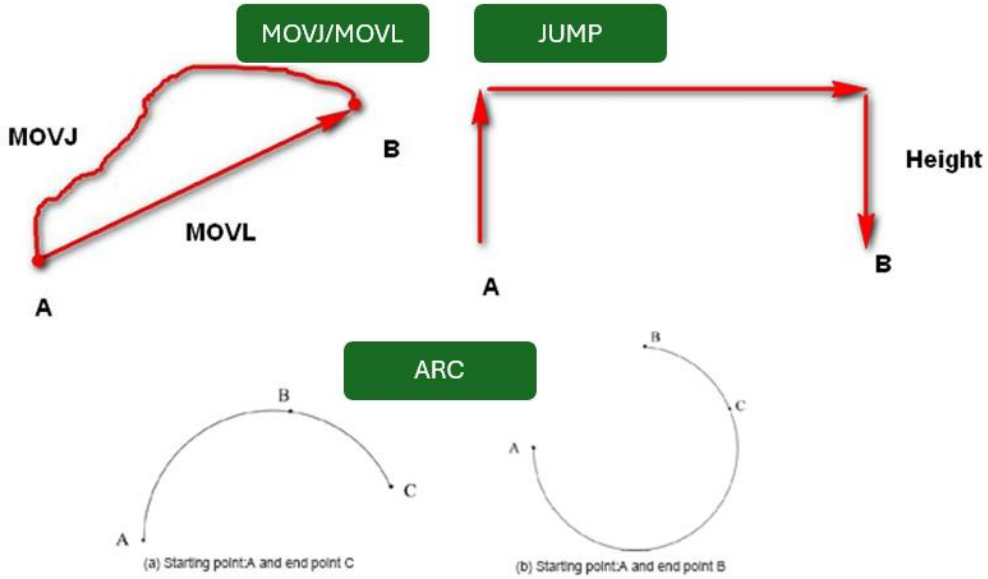

Figure 2.2.2.10.9. Movement Option [12]

The movement in Table 2.2.2.10.1 is shown in Figure 2.2.2.10.9 [12].

Value Transmission:

The package “sys” is used to receive the value of length of the circle, length of the symbol “X” and the length of the line from the previous step since the previous step is in a different file.

Circle:

The movement chosen is MOVJ since the calculation is for two-point connection. The calculation is shown in following Figure 2.2.2.10.10:

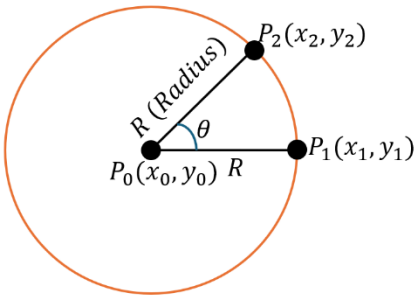

Figure 2.2.2.10.10. Schematic of Circle Calculation

P_0 is the centre point of the circle and it is assumed to be the actual point of Dobot Magician which is obtained by “dType.GetPose(api)” to get the location of the Dobot Magician. The drawing of the circle is the connection of P_1 and P_2 using MOVJ as shown in Figure, and then P_1 and P_2 will be renewed by the following equation:

$$x_2 = x_1 + R \times \cos\theta \qquad \text{eqn. 2.2.2.10.2}$$

$$y_2 = y_1 + R \times \sin\theta \qquad \text{eqn. 2.2.2.10.3}$$

The results are shown in Figure 2.2.2.10.11.

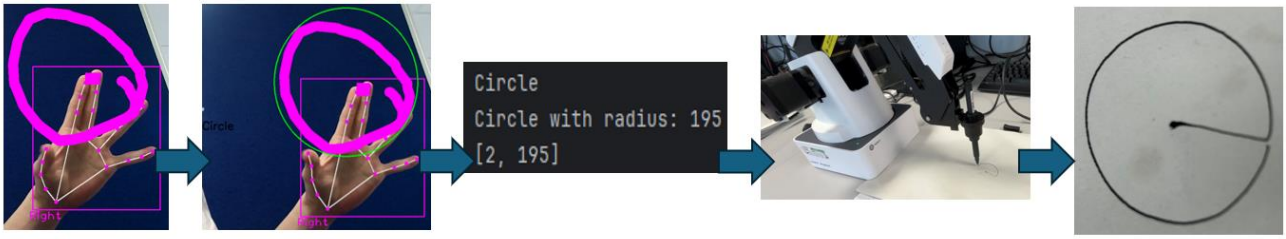

Figure 2.2.2.10.11. Circle Drawing

Symbol "X":

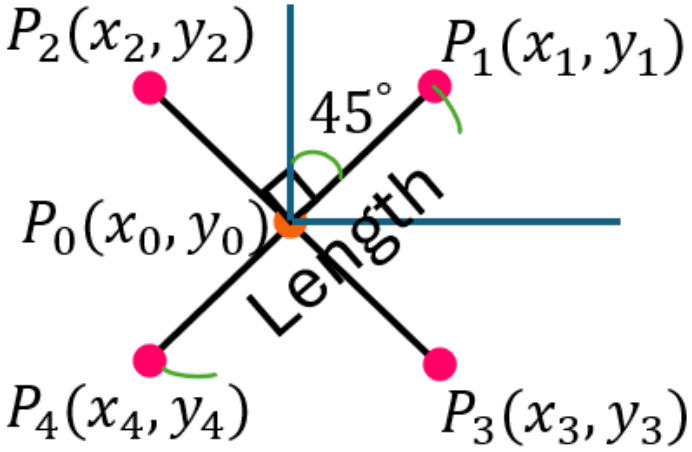

Figure 2.2.2.10.12. Schematic of X Calculation

The movement is also using point-to-point MOVJ and JUMP. The length from P_1 to P_4 and from P_2 to P_3 is obtained from step 9 using the package "sys" to receive. The points in Figure 2.2.2.10.12 can be calculated:

$$x_1 = \frac{Length}{2} \times cos45^\circ \quad \text{eqn. 2.2.2.10.4}$$

$$x_1 = x_3 = -x_2 = -x_4 \quad \text{eqn. 2.2.2.10.5}$$

$$y_1 = \frac{Length}{2} \times sin45^\circ \quad \text{eqn. 2.2.2.10.6}$$

$$y_1 = -y_3 = y_2 = -y_4 \quad \text{eqn. 2.2.2.10.7}$$

Thus, other points can be calculated based on Figure 2.2.2.10.13:

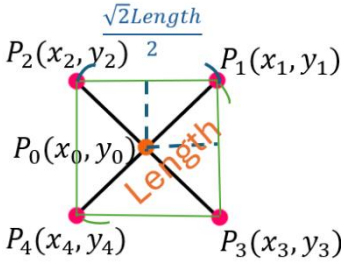

Figure 2.2.2.10.13. Schematic of X Calculation

$$P_4 = (x_1 - \frac{\sqrt{2}Length}{2},\ y_1 - \frac{\sqrt{2}Length}{2}) \quad \text{eqn. 2.2.2.10.8}$$

$$P_3 = (x_1,\ y_1 - \frac{\sqrt{2}Length}{2}) \quad \text{eqn. 2.2.2.10.9}$$

$$P_2 = (x_1 - \frac{\sqrt{2}Length}{2},\ y_1) \quad \text{eqn.2.2.2.10.10}$$

Initially, the movement of Dobot Magician is from $P_0 \rightarrow P_1 \rightarrow P_4$. Next, the location of Dobot Magician is in P_4, to draw the X, JUMP movement is needed to use to jump from P_4 to P_3 so that it will not have a line between these two points, and

then using MOVJ again to draw a line from P_3 to P_2. The result is shown in Figure 2.2.2.10.14.

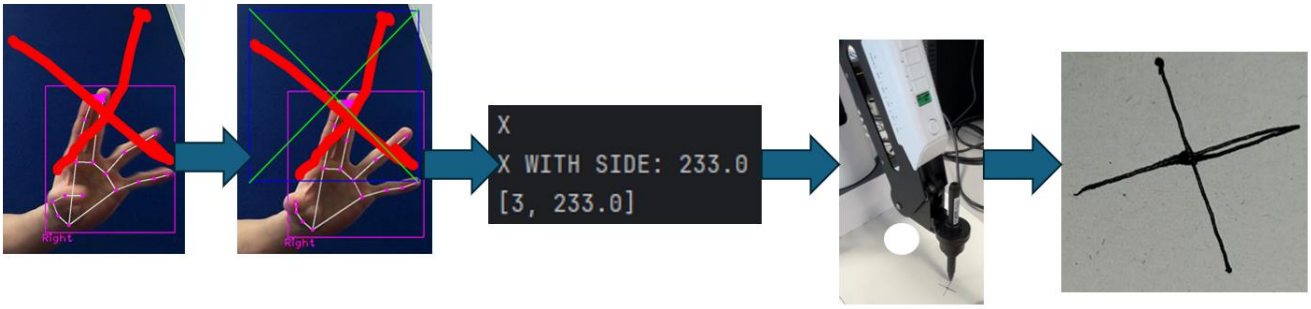

Figure 2.2.2.10.14. Symbol “X” Drawing

Line:
After obtaining the length of the line, the movement is to utilize the current position as an initial point, and then using MOVL to move to the new position. The x coordinate of both points is the same, and the difference of y coordinate between two points is the length of the line.

2.2.2.11 Real-Time Gesture Control

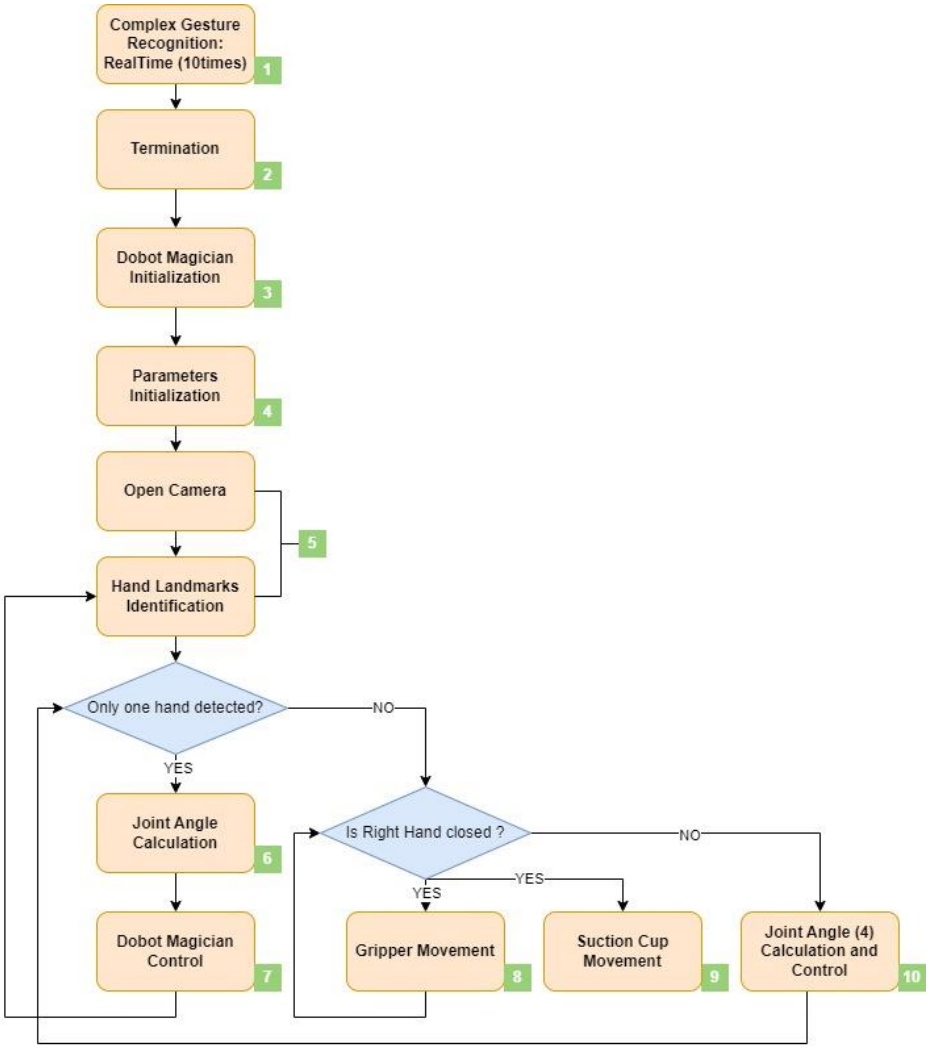

Figure 2.2.2.11.1. Flow Chart of Real-time Control

The flow chart in Figure 2.2.2.11.1 illustrates the process of real-time gesture control of Dobot Magician. The ideas of Real-Time Gesture Control are learned from the courses [60] to modify to realize real-time gesture control with Dobot Magician.

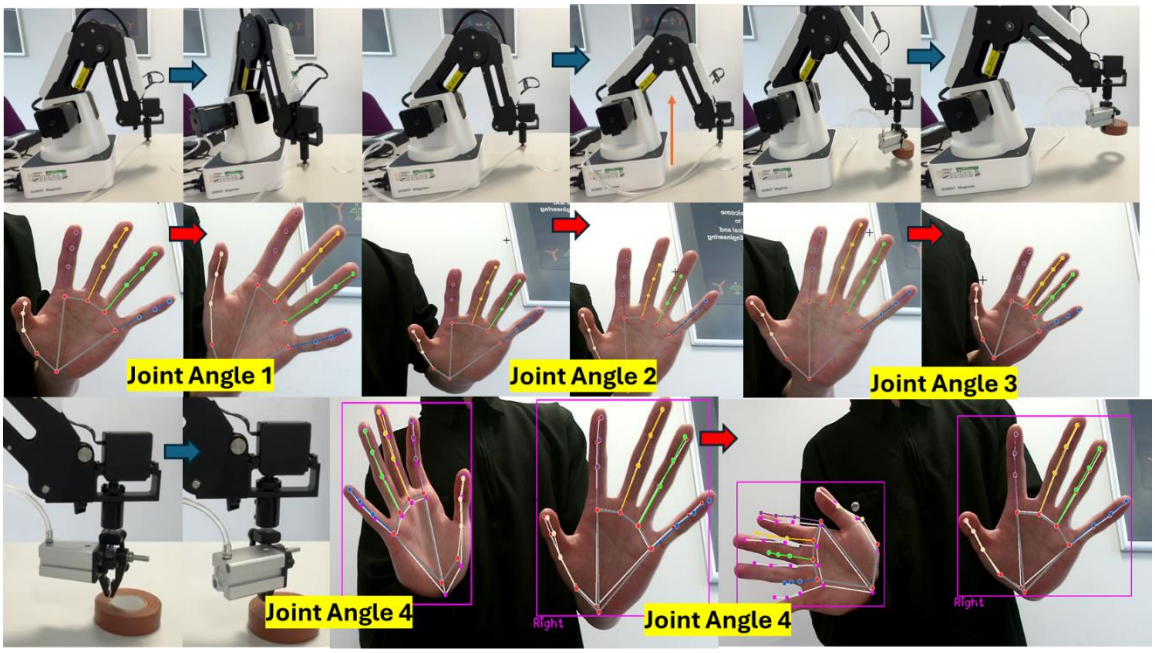

Figure. 2.2.2.11.2 Example Movement to Change Joint Angles

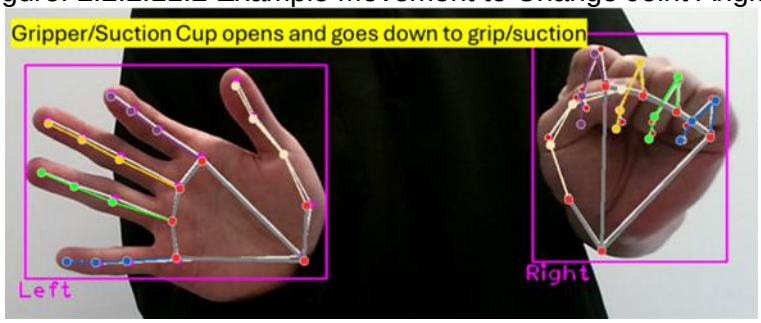

Figure 2.2.2.11.3. Gesture to control the Gripper or Suction Cup

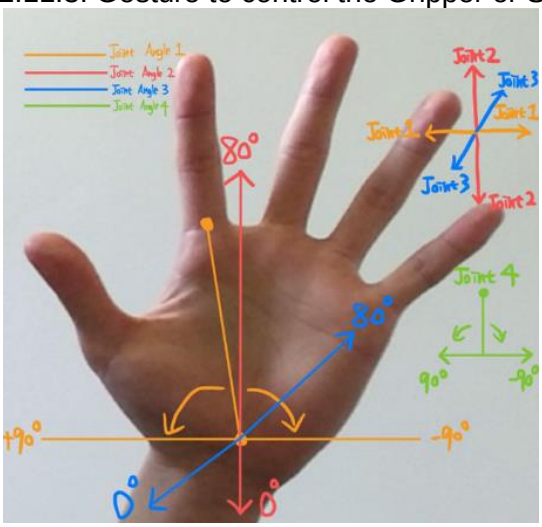

Figure 2.2.2.11.4. Schematic of Joint Angles Movement and Gesture to Control

In Finger 2.2.2.11.2, the process how to change the joint angles (1,2,3,4) is shown. In Figure 2.2.2.11.4, the schematic is to show the movement of the hands to change the joint angles. The picture in Figure 2.2.2.11.3 is the gesture to start the gripper or the suction to grip or suction the object.

1. **Complex Gesture Recognition: RealTime (10 times)**

Figure 2.2.2.11.4. “RealTime” Gesture

The procedure of complex gesture recognition is explained in section 2.2.2.7. After the “RealTime” gesture is recognised continuously ten times as shown in Figure 2.2.2.11.4, the system will stop recognising the gesture and the motion is defined to implement “RealTime” task.

2. **Termination**

To minimize the delay of real-time gesture control with Dobot Magician, after the determination of RealTime, the original program is terminated, and the camera is also released. Using the function “subprocess.run()” in subprocess package can open the other file without manually opening the file. As a result, the data in the file of real-time control task is not returned into the main program.

3. **Dobot Magician Initialization**

The process is the same as previous which is explained in section 2.2.2.10.

4. **Parameters Initialization**

The parameters of Dobot Magician are shown in Table 2.2.2.10.1. Therefore, the range of each joint angle is set as shown in Table 2.2.2.11.1.

Table 2.2.2.11.1. Joint Angle Parameters

Joint Angle	Angle Range
Joint Angle 1 (Base)	$-90^{\circ}\sim+90^{\circ}$
Joint Angle 2 (Rear Arm)	$0^{\circ}\sim80^{\circ}$
Joint Angle 3 (Forearm)	$0^{\circ}\sim80^{\circ}$
Joint Angle 4 (End-effector rotation)	$-90^{\circ}\sim+90^{\circ}$

5. **Open Camera and Hand Landmarks Identification**

To open camera, the OpenCV function “cv2.VideoCapture(0)” is used, and the process of Hand Landmarks Identification is the same as previous which is explained in section 2.2.2.3.

6. **Joint Angle Calculation**

The process is used the x, y and z coordinate of wrist (ID:0) and metacarpophalangeal joint of index finger (ID:5) as shown in Figure 2.2.2.2.1 to calculate the joint angles.

Joint Angle 1 (Base):

The theory is to calculate the angle through the difference of x coordinate values between wrist and metacarpophalangeal joint of index finger as shown in Figure 2.2.2.11.5.

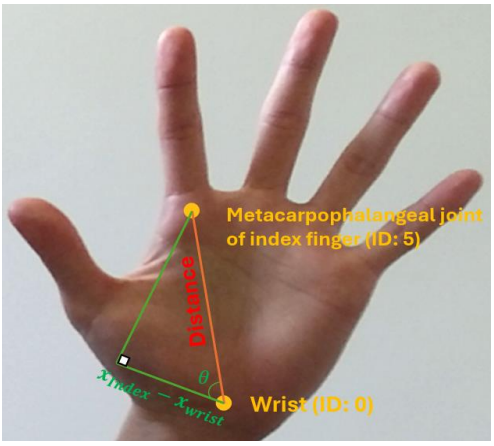

Figure 2.2.2.11.5. Schematic of Joint Angle Calculation

$$Distance = \sqrt{(x_{Index} - x_{Wrist})^2 + (y_{Index} - y_{Wrist})^2 + (z_{Index} - z_{Wrist})^2}$$ eqn. 2.2.2.11.1

$$\theta = \frac{x_{Index} - x_{Wrist}}{Distance} \times \frac{180^{\circ}}{\pi}\ [degrees]$$ eqn. 2.2.2.11.2

After the calculation of the angle, the math function "lambda" is used to limit the angle in the range of $-50^{\circ} \sim 50^{\circ}$. The range is adjusted by obtaining the initial angle θ when the hand is shifted to left or right as shown in Figure 2.2.2.11.2. The parameters can be different since it depends on the sensitivity of the system to the change of angle of the hand. Next, another "lambda" function is used to linearly maps a value from one range to another, and the principle is as follows:

For example, x = 50 and it is needed to be mapped from range [0, 100] to [0, 200]:

$$Ratio\ Calculation: \frac{50-0}{100-0} = 0.5$$ eqn. 2.2.2.11.3

$$Map\ to\ the\ new\ range: 0.5 \times (200 - 0) = 100$$ eqn.2.2.2.11.4

$$Plus\ the\ minimum\ of\ the\ new\ range: 100 + 0 = 100$$ eqn.2.2.2.11.5

As a result, using this theory can map the angle range to the new range which is $-90^{\circ} \sim +90^{\circ}$ which is referred to the limitation of joint angle 1 [12].

Joint Angle 2 (Rear Arm):

The theory is the same as the joint angle 1, but joint angle is mapped to the Y-axis operating range of the robotic arm based on the y-coordinate of the wrist. The old range is from 0.3 to 0.7 which is also adjusted by the moving the hand to the upper border of the frame and the lower border of the frame. The range can be different since it also depends on the sensitivity of the system to the change of angle of the hand. The new range is $0^{\circ} \sim 80^{\circ}$ which is referred to the limitation of joint angle 2 [12].

Joint Angle 3 (Forearm):

The theory is similar to the previous of joint angle 1 and joint angle 2, but joint angle is used the size of the hand "Distance" in Figure 2.2.2.11.5 to map to the operating range of the Z-axis. In other words, it depends on the distance between the hand and the camera. In fact, the size of the palm is approximately the same to the user. However, to the camera, the size of the palm is not the same. If the hand is far from the camera, the size of palm will be smaller and vice versa. The initial range is from 0.1 to 0.25 which is also adjusted by moving the hand far from the camera and close to the camera. The range can be different since it also depends on the sensitivity of the system to the change of angle of the hand. The new range is $0^{\circ} \sim 80^{\circ}$ which is referred to the limitation of joint angle 3 [12].

7. Dobot Magician Control

To control the joint angles of Dobot Magician, the function “dType.SetPTPCmdEx” is used to set the joint angles based on the previous calculation in step 6. Since the joint angles are calculated to many digits after the decimal point and it is hard for the user to keep the hand static without slight moves, the joint angles change continuously. Moreover, if the joint angles of Dobot Magician are needed to change continuously, the delay time will increase, and Dobot Magician will not be able to follow the hand movement in real time. As a result, the joint angles are only taken to inter digits (unconditionally round off decimal places), and if the angle is the same as the previous, the joint angles of Dobot Magician wouldn’t change. However, the joint angle 4 is still the same as the initial pose in this stage, and it can be changed in the following steps.

8. Gripper Movement

The process is designed that if the right hand is closed, the air pump will inject compressed air into the cylinder through the air tube to push the piston inside the cylinder to open the gripper and move down. Then, the air is released from the cylinder to grip the object (Close) and move up to the previous height. Whether the right hand is closed or not is determined by Finger Detection which is explained in section 2.2.2.5.

9. Suction Cup Movement

The working principle is also designed that if the right hand is closed, the suction cup will move down and the air pump evacuate the air from inside the pump, and thus the suction cup absorb to the surface of the object. Then, the suction cup will move up to the initial height.

10. Joint Angle (4) Calculation and Control

The mode is active when two hands are detected on the frame, and the pose of Dobor Magician is paused only the joint angle 4 can be changed by the gesture of the left hand. The theory is the same as the joint angle 1 to map the angle from the initial range to the new range. The initial range is also $-50^{\circ} \sim 50^{\circ}$, and the new range is from $0^{\circ} \sim 150^{\circ}$ which is referred to the limitation of joint angle 4 [12]. Consequently, using this function the angle of the gripper and the suction cup can be modified to adjust the appropriate angle to grip or absorb the objects.

3 Final System Testing and Validation

3.1 Camera

Figure 3.1.1. Testing environment setting

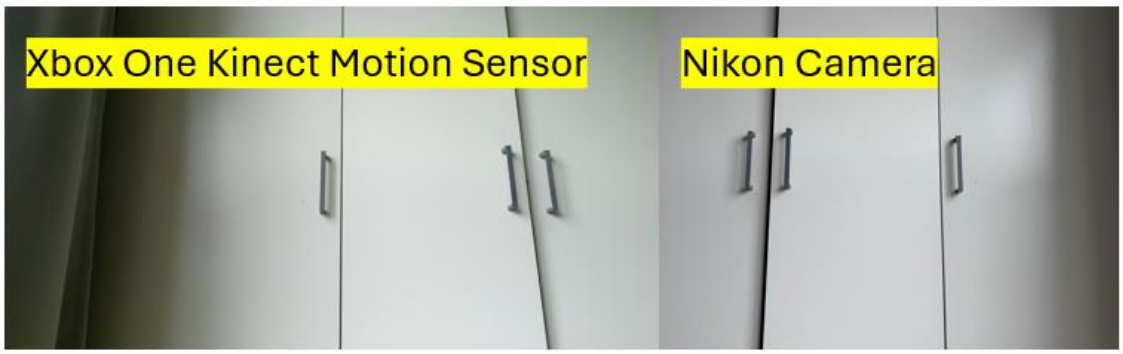

Figure 3.1.2. Comparison

The testing environment was set as shown in Figure 3.1.1 to compare the differences between Xbox One Kinect Motion Sensor and the actual vision in the real world. Based on Figure 3.1.2, the results proved that the frame captured by Xbox One Kinect Motion Sensor is approximately the same as the actual vision in the real world. After repeated testing, Xbox One Kinect Motion Sensor could be turned on successfully every time. Unless the camera continuously turned on and off more than 20 times, it might overheat and be unable to display images as shown in Figure 3.1.3.

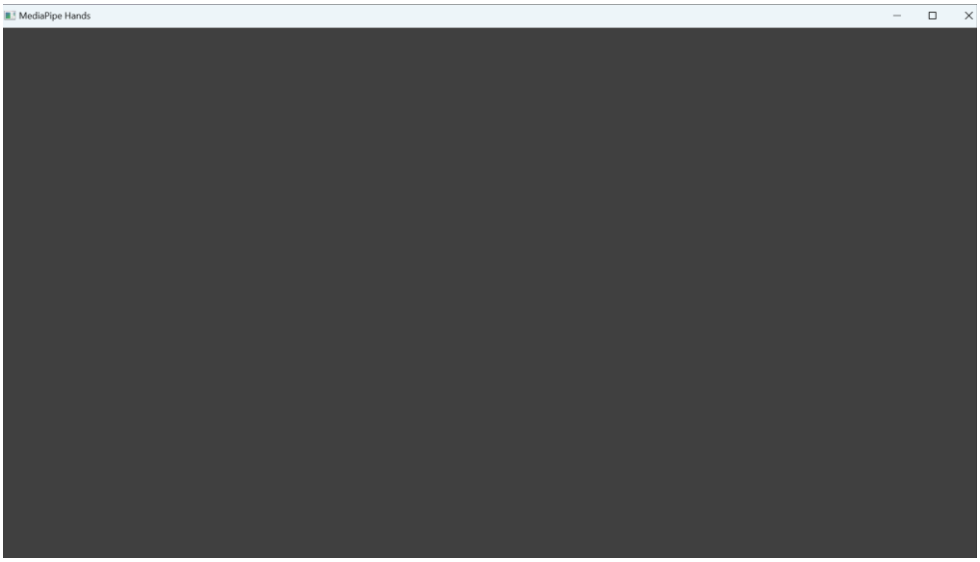

Figure 3.1.3. Error Display

3.2 Hand Identification

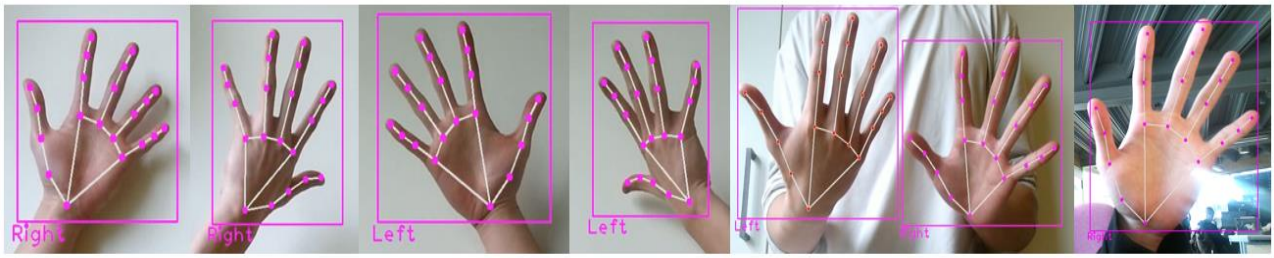

Figure 3.2.1. Hand Identification

As shown in Figure 3.2.1, the system was able to identify the palm, back, left and right hand clearly, and also was able to identify two hands simultaneously. Moreover, even in the messy background, the hands were also detected properly.

Figure 3.2.2. Hand Identification Testing

Based on Figure 3.2.2, the system of hand identification was able to recognise the hands clearly in the situations of low-light environments, hands with obstacles, 2D images. When the obstacles were in front of the hands, the system would take more time to recognise the hands. Hand Identification was only not working if the light was comprehensively dark. However, if the hands were in 2D, it was unable to perform the tasks.

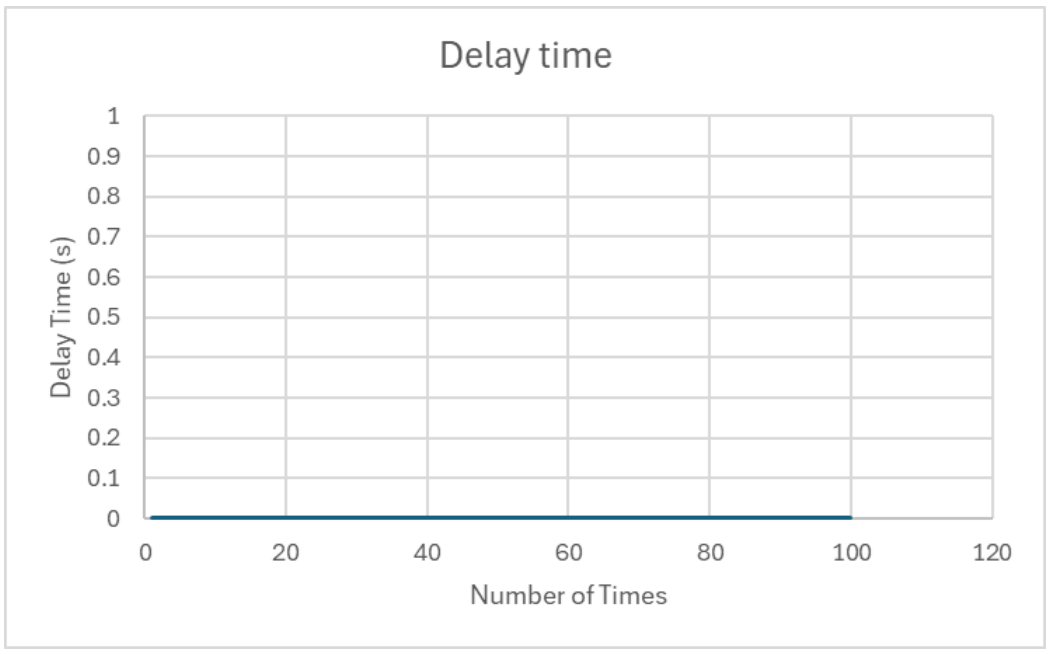

Figure 3.2.3. Delay time of Hand Identification

As shown in Figure 3.2.3, the system could recognise the hand in real-time, and the testing results that delay time is 0 second could indicate that once the hands were

shown in the frame, the hands would be captured by the camera and be recognised instantly.

Figure 3.2.4. Gloves Recognition

According to Figure 3.2.4 the system is unable to recognise the hand if the hand in gloves as the recognised model is trained by a real human hand [61].

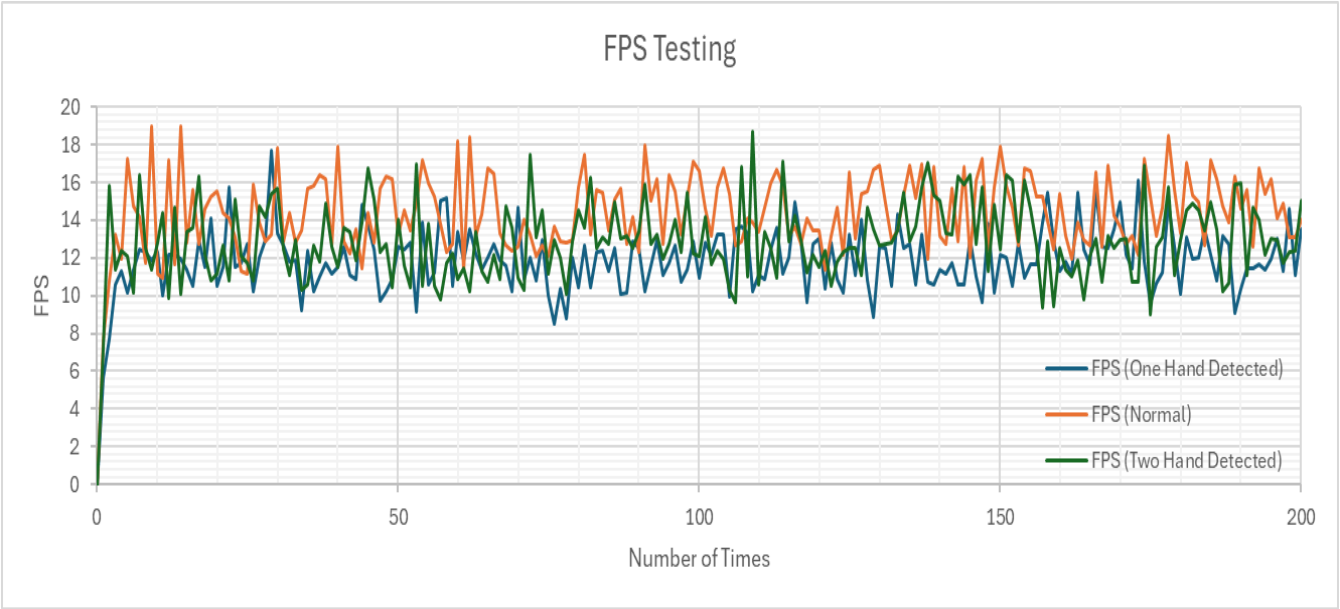

Figure 3.2.5. FPS Testing

The system was tested by testing Frames Per Second (FPS) in Figure 3.2.5 based on the following formula:

$$FPS = \frac{1}{Current\ Time - Previous\ Time\ (Recorded\ from\ previous\ iteration)}\ [FPS] \quad \text{eqn. 3.2.1}$$

As shown in Figure 3.2.5, the initial FPS was less than 10 fps since the system was needed to import the data from the libraries. Once the camera captured the hands, the initial was also low than 10 fps as the process of obtaining the hand information took some time to launch. After that, the FPS was approximately over than 10 fps, and the frame was stable and smooth. The FPS sometimes was still lower than 10 fps when the hands in the frame moved rapidly or changed the pose quickly.

Based on the above results, Hand Identification was worked in 2D and 3D recognition, low-light conditions, hands with obstacles and hands in messy backgrounds. The system only not worked in dark conditions and the hands was in 2D images. From the above testing, Hand Identification was able to work in real time. As the hand identification was unable to work if the hands were in gloves and the light was dark, the following testing would not test the system in these two conditions.

3.3 Hand Landmarks Identification (MediaPipe)

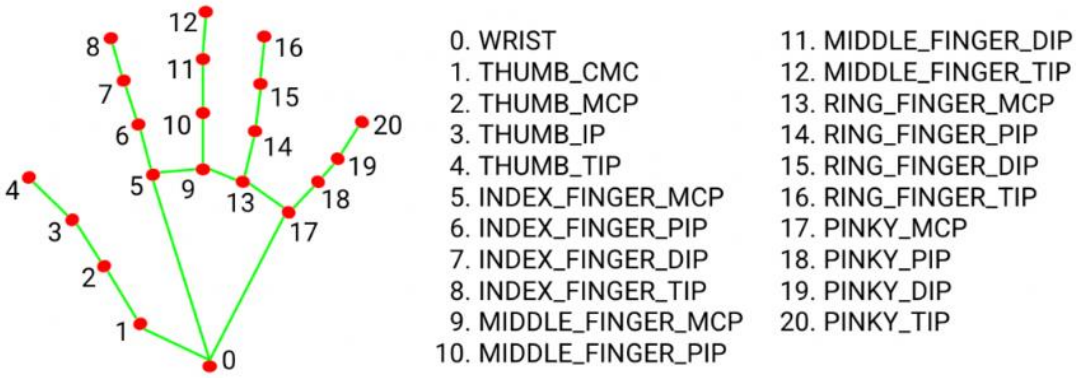

Figure 3.3.1. Hand Landmarks (MediaPipe) [39]

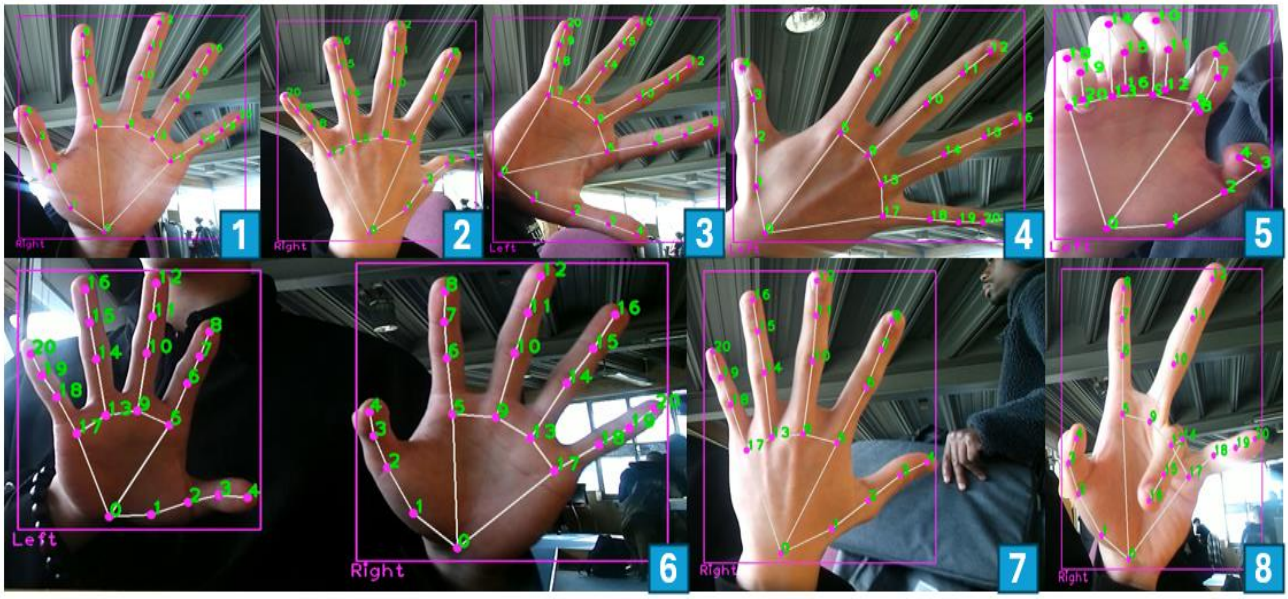

Figure 3.3.2. Testing Setting

In Figure 3.3.2, the numbers were as follows: 1. Palm (Right) 2. Right (Back) 3. Palm (Left) 4. Back (Left) 5. Bending (Left) 6. Two Hands 7. Other hands Affecting 8. Bending (Left)

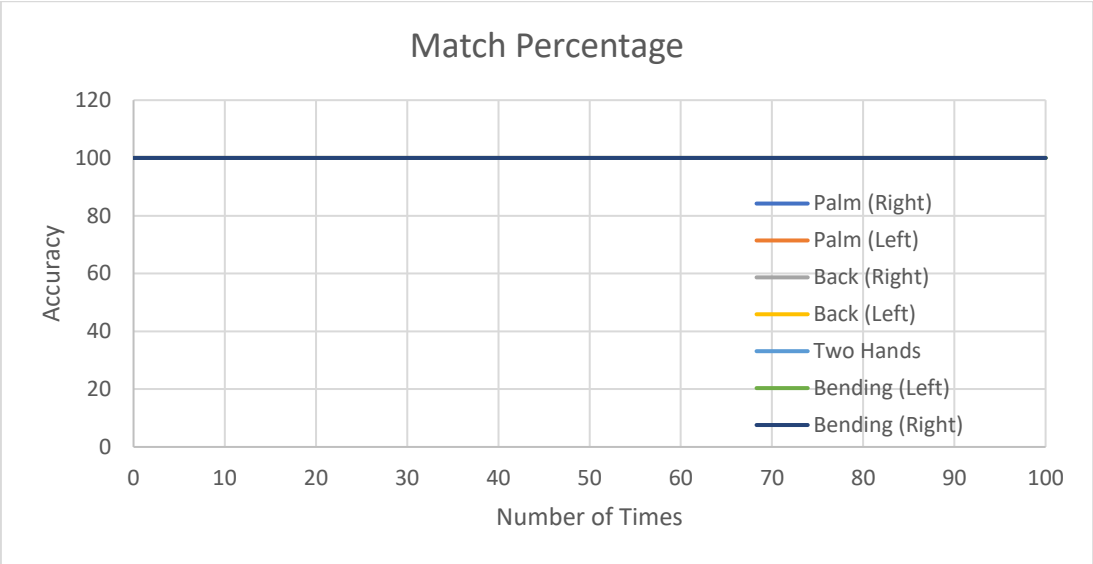

Figure 3.3.3. Match Percentage

The method was to print the hand landmarks on the corresponding position of the frame to compare the hand landmarks provided by MediaPipe in Figure 3.3.1 to the hand landmarks shown in Figure 3.3.2. In the seventh picture in Figure 3.3.2, it indicated that when an integral human hand was shown on the frame and had detected already, it would not detect other incomplete hands. This phenomenon was also in line with the previous testing results in Hand Identification. Based on Figure 3.3.3, the recognised model trained by MediaPipe could identify the human hands in any pose, even the hands were in the messy background.

3.4 Hand Position Identification

According to section 2.2.2.2, as the coordinate of each hand landmark was calculated through a normalization of camera's width and height, and the coordinate of each hand landmark would be projected to the new range based on the actual camera's width and height. Consequently, the testing method was to print the original point and the projected point simultaneously on the frame. The calculation of the accuracy was to calculate the length of wrist to middle fingertip in the range of 0 to 1 (MediaPipe) and the range of the camera respectively and subtracted these two values to obtain the difference (Distance 1). The next step was to calculate the length of wrist to index fingertip in the range of 0 to 1 (MediaPipe) and the range of the camera respectively and subtracted these two values to obtain the difference (Distance 2). Next, the ratio could be calculated as follows:

$$Ratio = \frac{Distance\ 1}{Distance\ 2} \qquad \text{eqn. 3.4.1}$$

If the ratio was closer to 1, it indicated that the two results were amplified by the same factor. Since the original coordinates would be mapped to the new coordinates (multiplied by the width and height of the camera), the closer the ratio was to 1, it manifested that the two values were the same for their corresponding range distances.

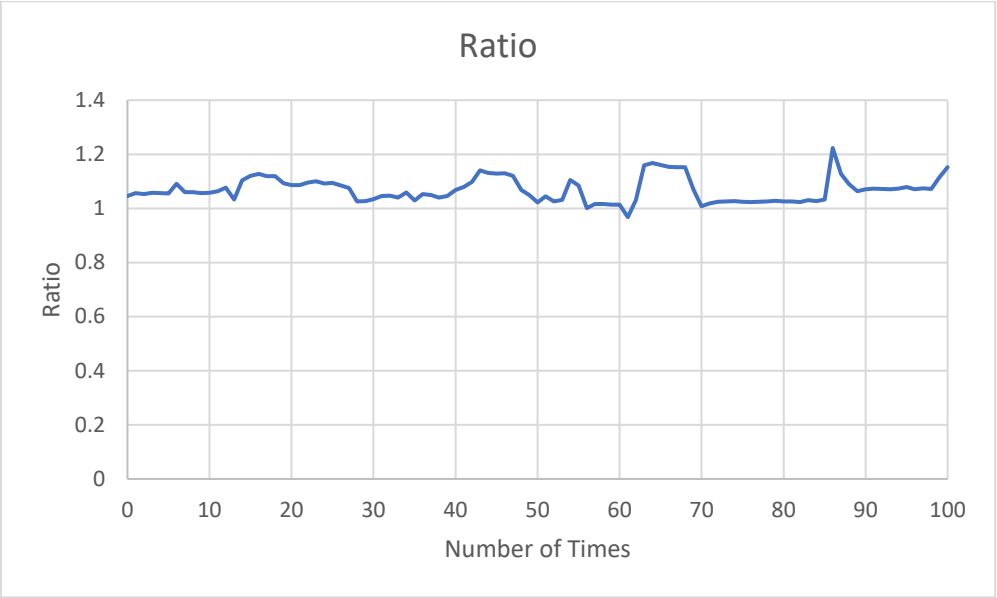

Figure 3.4.1. Ratio

Based on Figure 3.4.1, the ratio was roughly in the range of 0.9 to 1.2. Therefore, the recalculation of x and y coordinates were consistent with the original coordinates calculated by MediaPipe.

3.5 Finger Detection, Finger Classification and Finger Counting

As Finger Classification and Finger Counting were based on the data from Finger Detection in section 2.2.2.5, three functionalities were tested together. The approach was to test the success percentage of correct recognition in the conditions of well-lit environment, low light environment, right and left hands, obstacles in front of the hand. The FPS of the system was also tested.

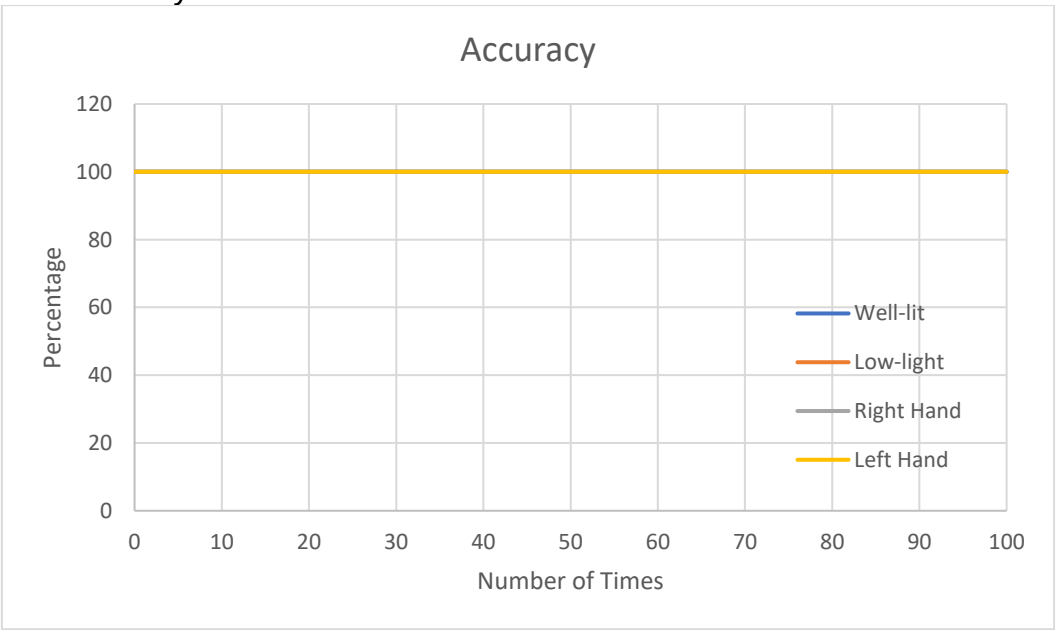

Figure 3.5.1. Recognition Accuracy

Table 3.5.1. Recognition Accuracy when the obstacle was in front of the hand

Number	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Accuracy (%)	70	76	91	80	84	84	47	51	38	21
Number of Samples	357	342	290	343	344	275	340	291	212	264

Based on Table 3.5.1, the obstacle in front of the hand would have a significant influence on gesture 6, 8 and 9, and the accuracy percentage would be less than 50%. In Figure 3.5.1, the results manifested that Finger Detection, Finger Classification and Finger Counting worked precisely with 100% accuracy under well-lit condition and low light condition. Overall, the obstacle would extremely affect the recognition of Finger Detection, Finger Classification and Finger Counting.

3.6 Complex Gesture Recognition

The method was to examine the correct recognition of the gestures under the well-lit environment, clear background, messy background and low-light environment, and the FPS during the recognised process was also tested. The number of trained samples were shown in Table 3.6.1.

Table 3.6.1. Number of Training Samples

Gesture	Single	MultiHands	Draw	RealTime
Trained Samples	77	91	86	107

Table 3.6.2. Testing (Right Hand, Low-light Environment, Clear Background)

Gesture	Single	MultiHands	Draw	RealTime
Accuracy	0.826087	0.805128	1	1
Samples	138	195	233	145

Table 3.6.3. Testing (Left Hand, Low-light Environment, Clear Background)

Gesture	Single	MultiHands	Draw	RealTime
Accuracy	0.62701	0.683077	0.957983	1
Samples	311	325	238	287

Table 3.6.4. Testing (Right Hand, Well-lit Environment, Clear Background)

Gesture	Single	MultiHands	Draw	RealTime
Accuracy	0.995327	0.990291	1	1
Samples	214	309	310	325

Table 3.6.5. Testing (Left Hand, Well-lit Environment, Clear Background)

Gesture	Single	MultiHands	Draw	RealTime
Accuracy	0.951111	0.967442	1	1
Samples	225	215	258	267

Table 3.6.6. Testing (Right Hand, Low-light Environment, Messy Background)

Gesture	Single	MultiHands	Draw	RealTime
Accuracy	0.826087	0.805128	1	1
Samples	138	195	233	145

Table 3.6.7. Testing (Right Hand, Well-lit Environment, Messy Background)

Gesture	Single	MultiHands	Draw	RealTime
Accuracy	0.626943	0.962567	0.942857	0.954082
Samples	193	187	210	196

Table 3.6.8. Testing (Right Hand, Various Distance, Well-lit environment, Clear Background)

Gesture	Single	MultiHands	Draw	RealTime
Accuracy (Close)	0.823529	1	1	1
Samples (Close)	153	155	180	136
Accuracy (Medium)	0.816456	0.513636	0.644809	1
Samples (Medium)	158	220	183	156
Accuracy (Far)	0.108247	0.197719	0.37551	1
Samples (Far)	194	263	245	187

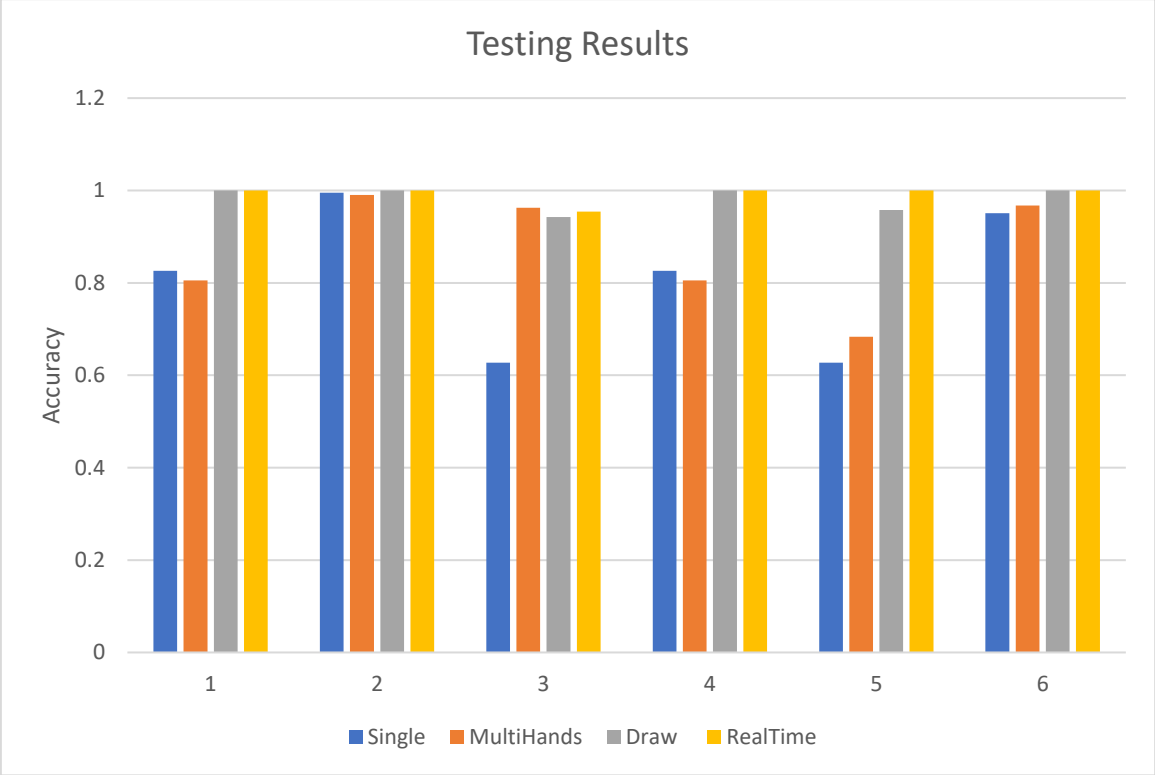

Figure 3.6.1. Testing Results under six conditions

In Figure 3.6.1, the six conditions were as follows: 1. Right hand, low light environment and clear background 2. Right hand, well-lit environment and clear background 3. Right hand, low light environment and messy background 4. Right hand, well-lit environment and messy background 5. Left hand, low light environment and clear background 6. Left hand, well-lit environment and clear background

The data from Table 3.6.2 to Table 3.6.8 were organised in Figure 3.6.1. Based on Figure 3.6.1, the performance of left-hand recognition was weak as the trained data was trained by the right-hand image. To improve the left-hand recognition, the brightness could enhance the accuracy of left-hand recognition. In the messy background, it would significantly affect “Single” gesture recognition, and it could be improved in low-light environment since the low light could block the messy background.

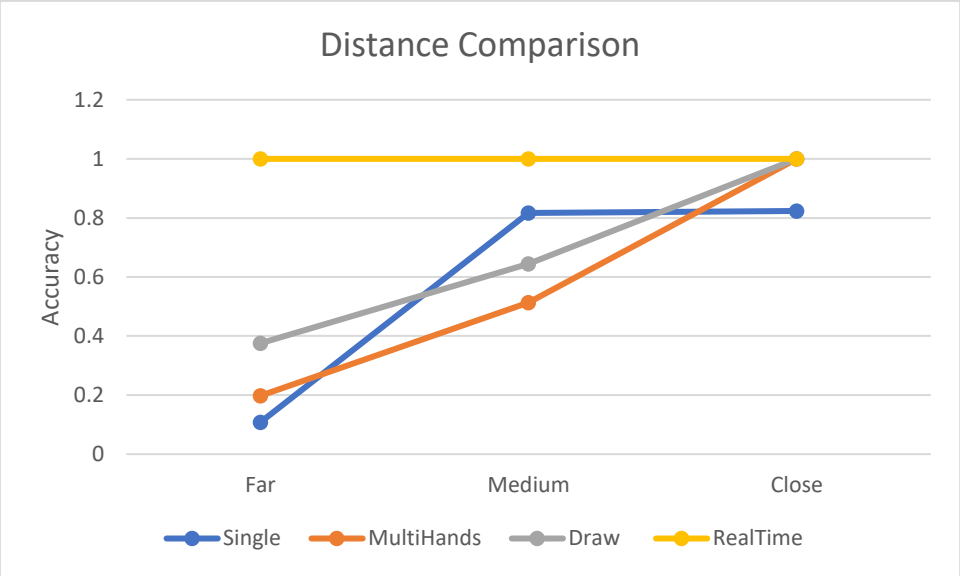

Figure 3.6.2. Influence of Distance

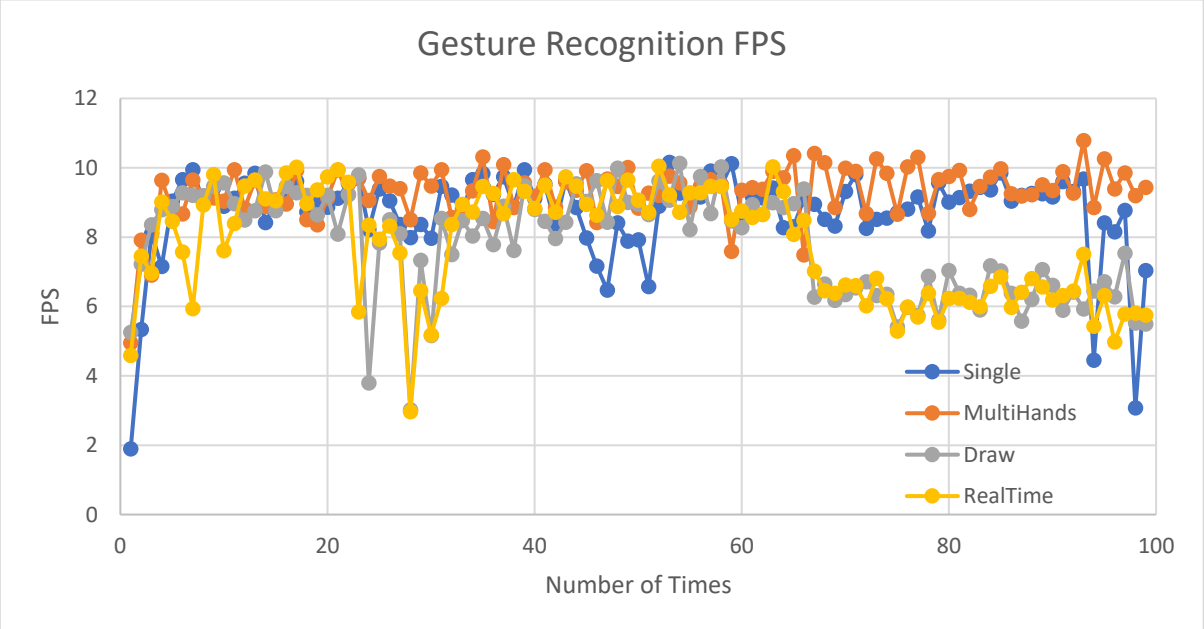

Figure 3.6.3. FPS Testing

The FPS testing was based on eqn. 3.2.1. In Figure 3.6.3, the tendency indicated that the closer the distance, the more accurate the gesture recognition. The FPS in four gesture recognition was roughly stable, only the FPS in “RealTime” gesture recognition was slightly unstable.

3.7 Multi Hand Recognition

Table 3.7.1. Recognition Accuracy

	HandDistance	FingerCounting(2)
Accuracy	100	83.86
Samples	267	276

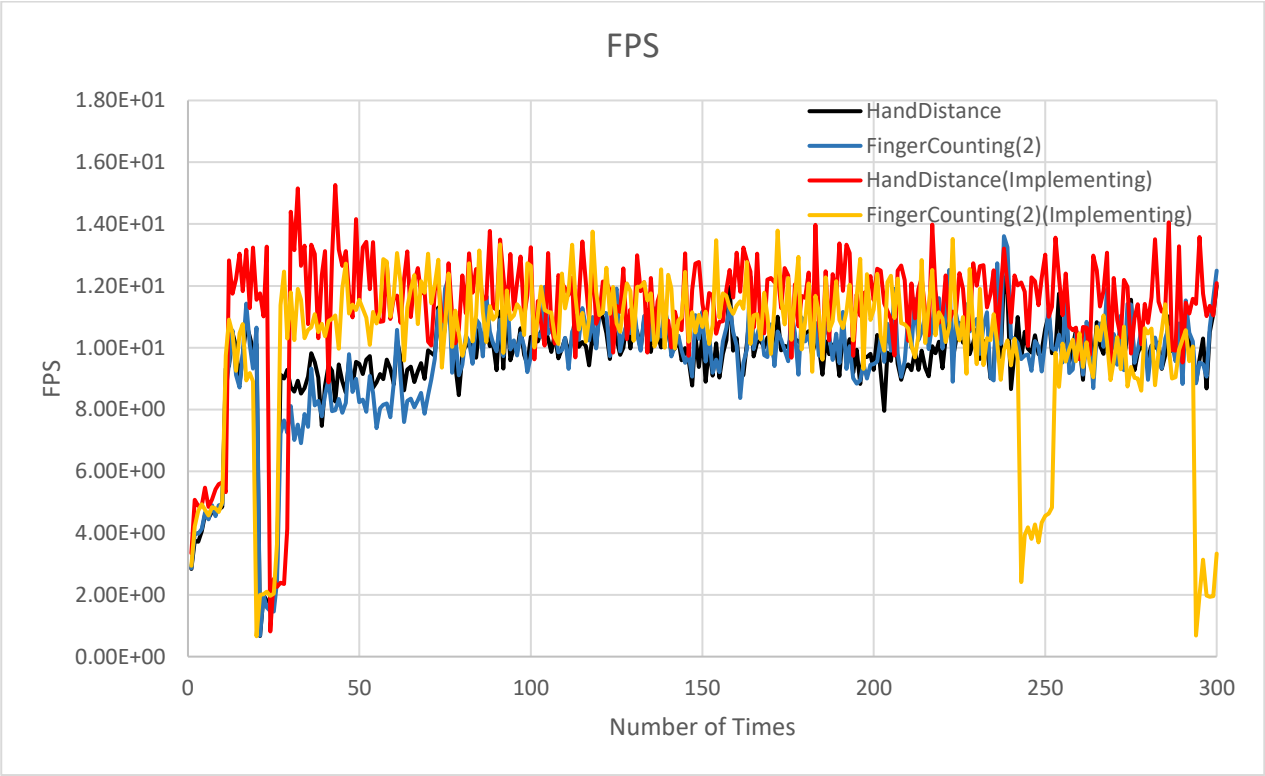

Figure 3.7.1. FPS Testing

The FPS testing was based on eqn. 3.2.1. The data in Table 3.7.1 was tested under the well-lit and clear background to verify the recognition accuracy. As shown in Figure 3.7.1, except for the initial setup, the overall FPS was in the range of 8 to 15 so that the frame was displayed smoothly during the major execution time.

3.7.1 Hand Distance

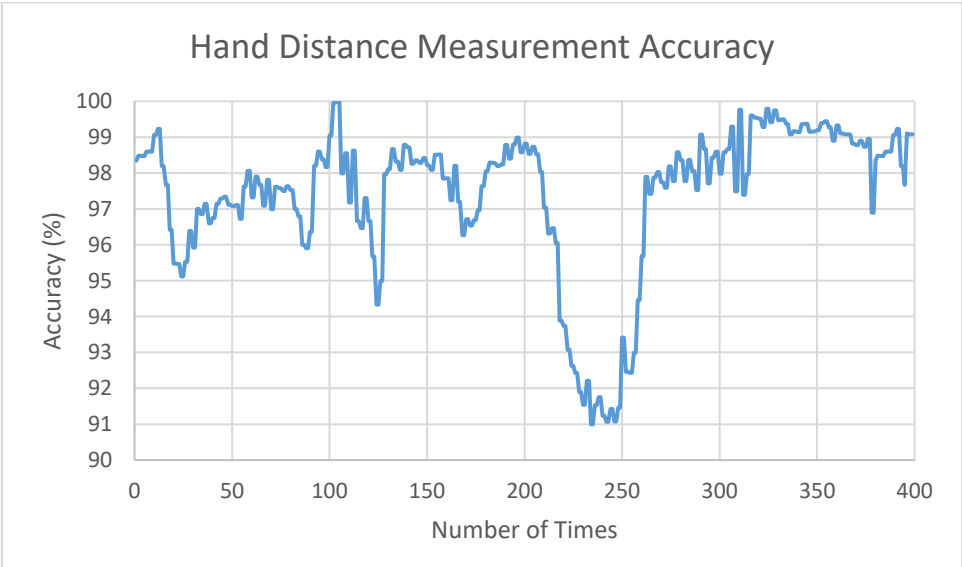

Figure 3.7.1.1. Measurement Accuracy

The testing method was to obtain the coordinates of middle finger MCP (ID:9) and calculate its length in three- dimensional (3D) space to compare with the calculated distance measured by the system. In Figure 3.7.1.1, the results indicated that the hand distance calculated by the system was in conformity with the hand distance in 3D with more than 91% accuracy.

3.7.2 Finger Counting (Multi-hands)

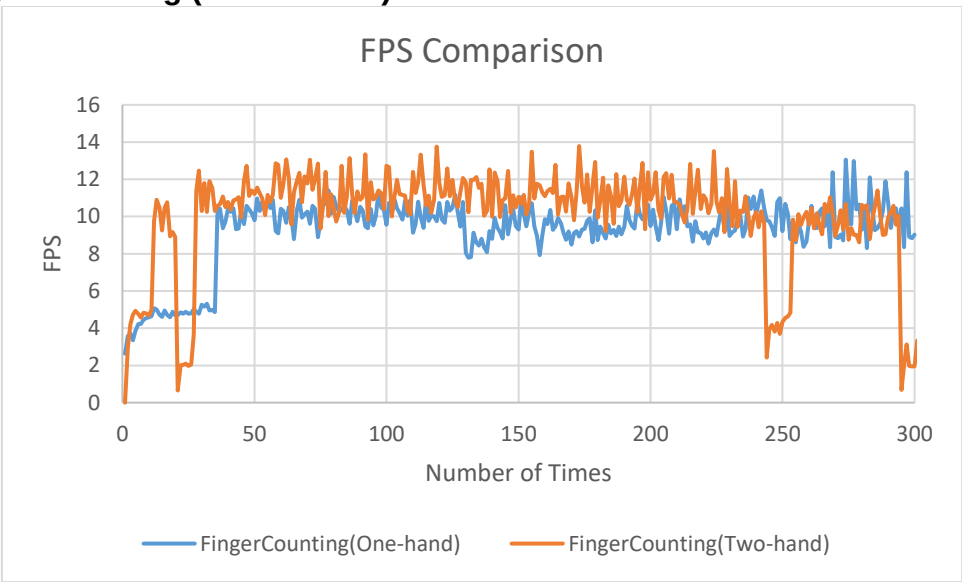

Figure 3.7.2.1. FPS Comparison

The FPS testing was based on eqn. 3.2.1. The verified method was to compare the FPS during the execution of FingerCounting (One-hand) in section 2.2.2.8 and FingerCounting (Two-hand). Based on Figure 3.7.2.1, the FPS during the execution of both tasks was approximately the same, and the phenomenon that the FPS suddenly descended significantly only happened in FingerCounting (Two-hand) as the system was needed to recognise two hands simultaneously.

3.8 Virtual Drawing

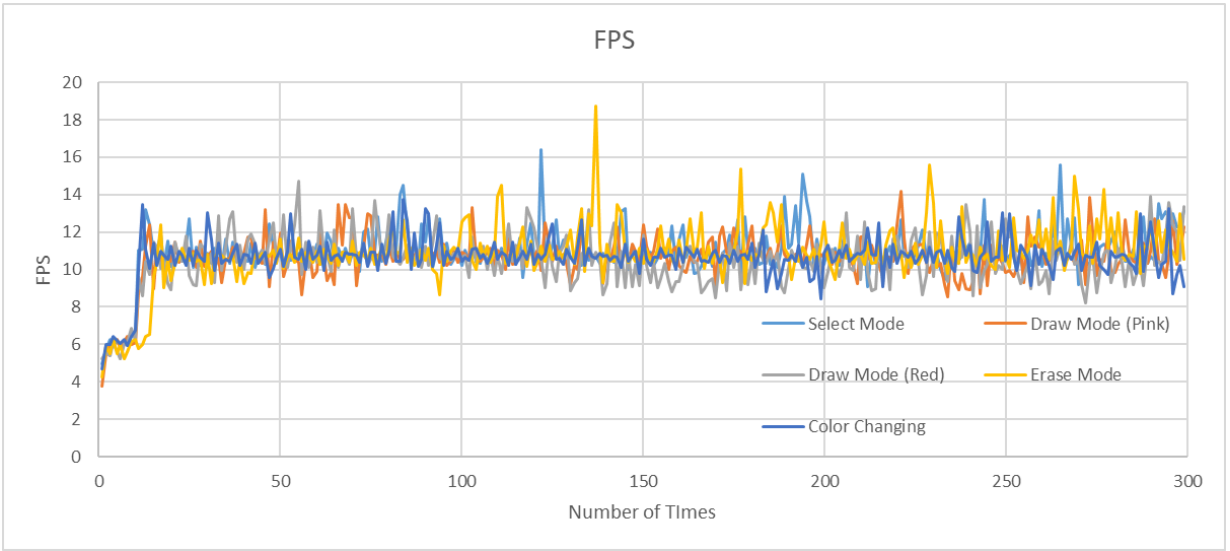

Figure 3.8.1. FPS Testing

The FPS testing was based on eqn. 3.2.1. As shown in Figure 3.8.1, after the initial setup, the FPS of all tasks was stable. In other words, the system could provide a smooth display for the user during the implementation of all tasks.

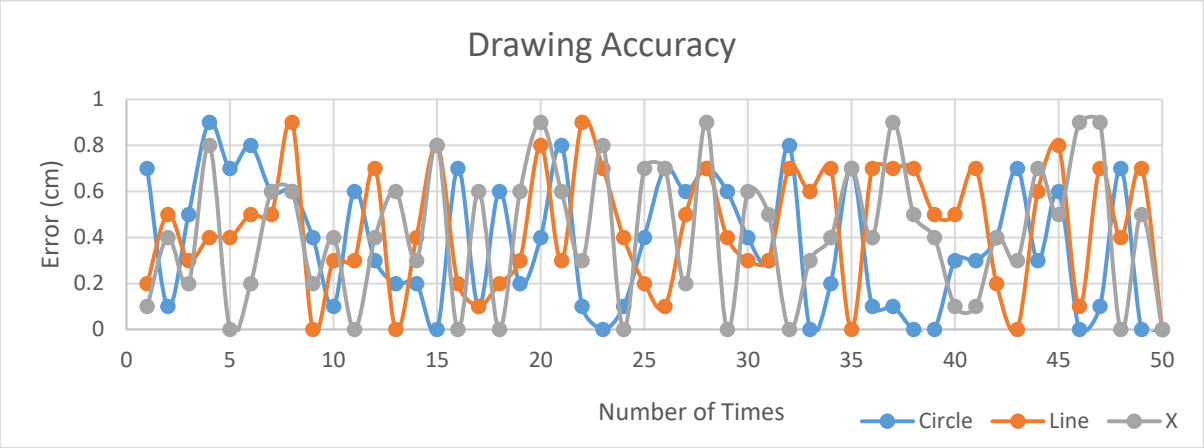

Figure 3.8.2. Drawing Accuracy

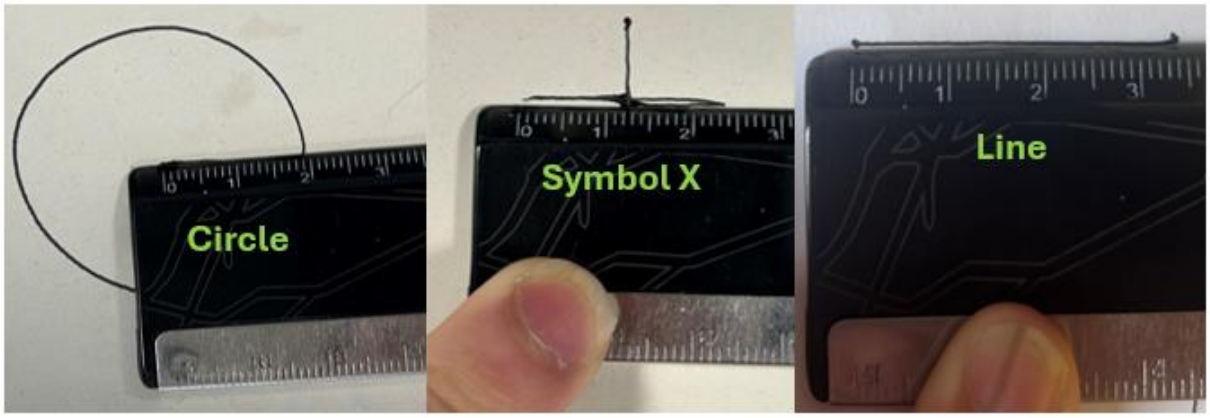

Figure 3.8.3. Verified Method

The verified approach was to use the rule in Figure 3.8.3 to measure the distance drawn by Dobot Magician on paper to compare with the corresponding distance measured by the system. In Figure 3.8.2, the maximum error was approximately 0.9 cm, and the minimum error was 0 cm. Consequently, the system design could work precise with less than 1 cm error.

3.9 Real-Time Gesture Control

As Dobot Magician was only able to use in the project lab, the influence of light would not be tested in this section. The method to test Real-Time Gesture Control with Dobot Magician was to obtain the pose of Dobot Magician such as x, y and z coordinates, joint angle 1, joint angle 2, joint angle 3 and joint angle 4 using Dobot Studio to compare to calculated joint angles. Using Inverse Kinematics to calculate the corresponding joint angles from the coordinates obtained from Dobot Studio. The principle of Inverse Kinematics was learned from [62]. The following diagrams of Inverse Kinematics was based on the theory [62] and redrew as follows:

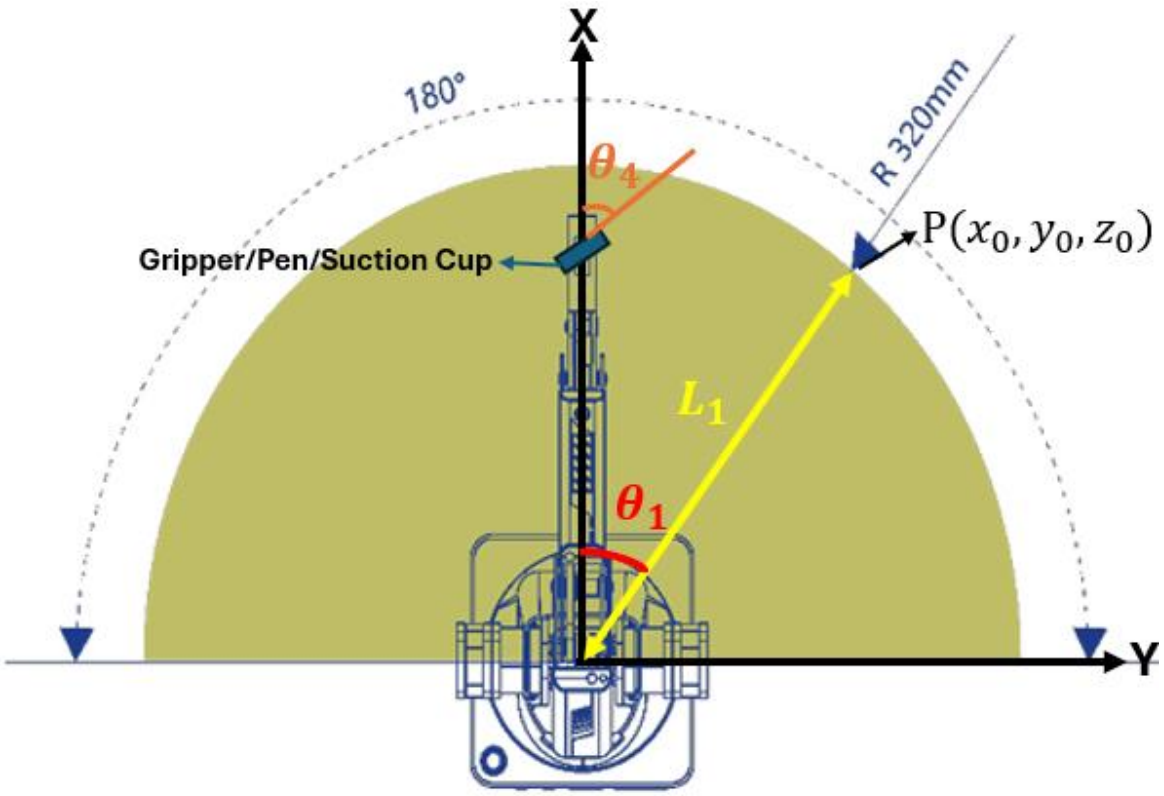

Figure 3.9.1. Joint Angle 1 and Joint Angle 4Calculation

Based on Figure 3.9.1, the joint angle 1 could be calculated:

$$L_1 = \sqrt{x_0^2 + y_0^2} \qquad \text{eqn. 3.9.1}$$

$$\theta_1(Joint\ Angle\ 1) = \tan^{-1}\left(\frac{y_0}{x_0}\right)[Degrees] \qquad \text{eqn.3.9.2}$$

$$\theta_4(Joint\ Angle\ 4) = Rotation\ (R) - \theta_1 \qquad \text{eqn. 3.9.3}$$

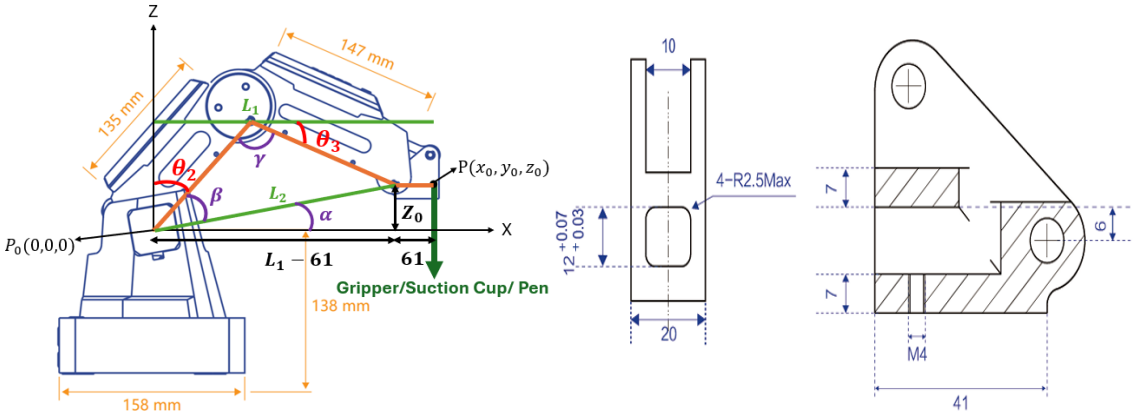

Figure 3.9.2. Joint Angle 2 and Joint Angle 3 Calculation

In Figure 3.9.2, the value of 61 was calculated from the left (41+20=61).

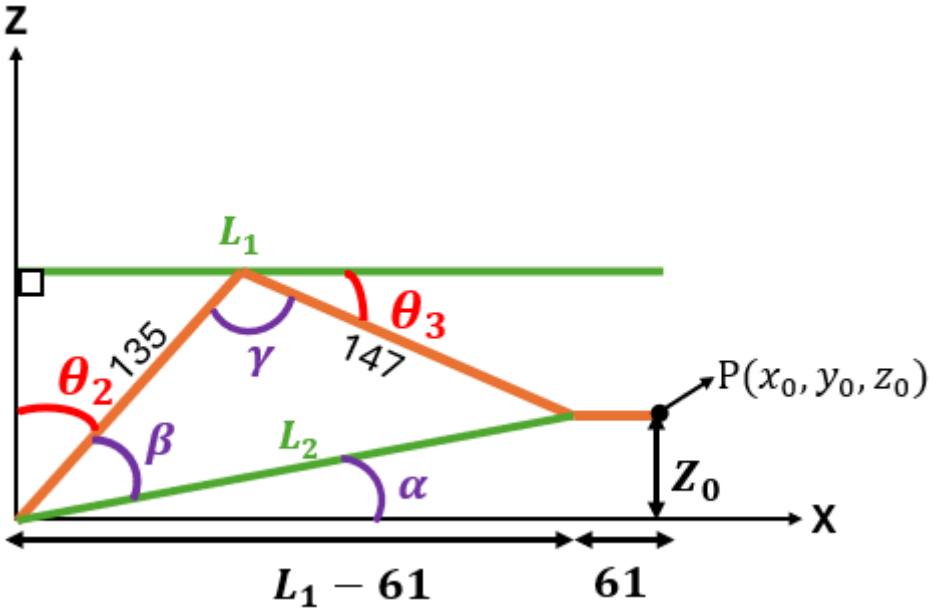

Figure 3.9.3. Simplified Diagram from Figure 3.9.2

Based on Figure 3.9.3, the joint angle 2 and the joint angle 3 could be calculated:

$$L_1 = \sqrt{x_0^2 + y_0^2} \quad \text{eqn. 3.9.4}$$

$$L_2 = \sqrt{(x_0 - 61)^2 + z_0^2} \quad \text{eqn.3.9.5}$$

$$\alpha = \tan^{-1}\left(\frac{z_0}{x_0 - 61}\right) \ [Degrees] \quad \text{eqn.3.9.6}$$

$$147^2 = 135^2 + (L_2)^2 - 2 \times 135 \times L_2 \times \cos(\beta) \quad \text{eqn.3.9.7}$$

$$\beta = \cos^{-1}\left(\frac{135^2 + (L_2)^2 - 147^2}{2 \times 135 \times L_2}\right) = \cos^{-1}\left(\frac{(L_2)^2 - 3384}{270 \times L_2}\right)[Degrees] \quad \text{eqn.3.9.8}$$

$$\theta_2(Joint\ Angle\ 2) = 90^\circ - \alpha - \beta\ [Degrees] \quad \text{eqn.3.9.9}$$

$$(L_2)^2 = 135^2 + 147^2 - 2 \times 135 \times 147 \times \cos(\gamma) \quad \text{eqn.3.9.10}$$

$$\gamma = \cos^{-1}\left(\frac{135^2 + 147^2 - (L_2)^2}{2 \times 135 \times 147}\right) = \cos^{-1}\left(\frac{39834 - (L_2)^2}{39690}\right)[Degrees] \quad \text{eqn.3.9.11}$$

$$\theta_3(Joint\ Angle\ 3) = 180^\circ - \gamma - (90^\circ - \theta_2) = 90^\circ - \gamma + \theta_2\ [Degrees] \quad \text{eqn.3.9.12}$$

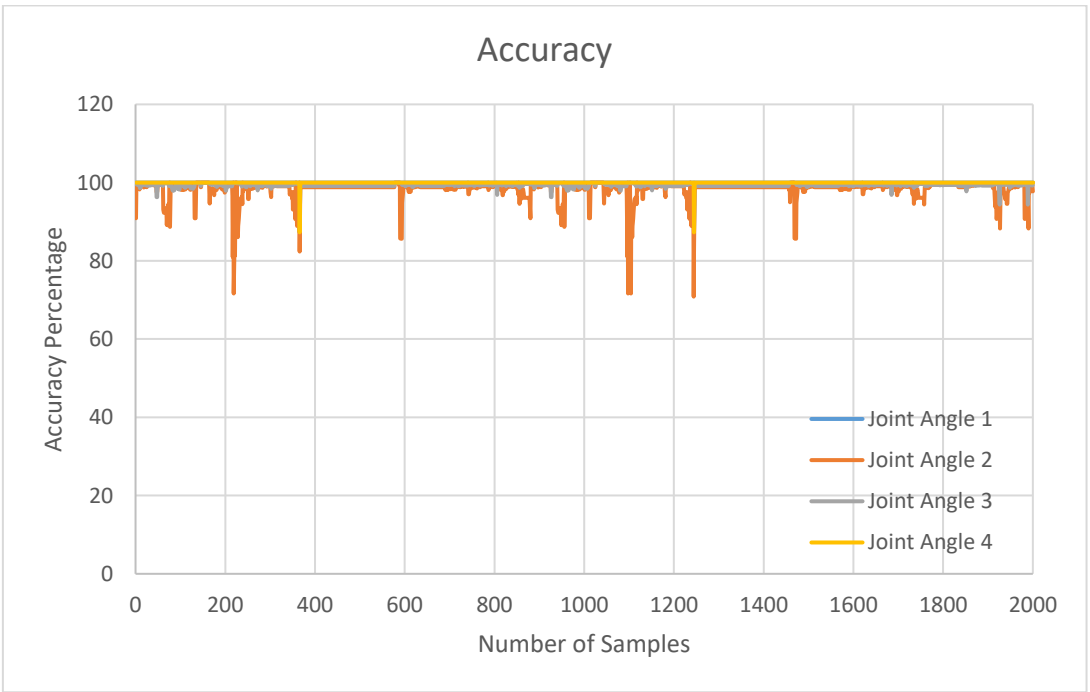

Figure 3.9.4. Calculated angle and Actual Angle (Dobot) Comparison

The testing method was to obtain the coordinates and rotation of Dobot Magician by recording current pose of Dobot Magician, and then the Inverse Kinematics was used to calculate the corresponding joint angles to compare the corresponding joint angles with the calculated joint angles through the calculation of joint angles in section 2.2.2.11. The joint angle 4 was verified by measuring the difference of rotation and joint angle 1. As shown in Figure 3.9.4, the accuracy of joint angle 1 calculation was extremely precise with 100% accuracy, the calculation of joint angle 3 was also with high accuracy, and only the accuracy of joint angle 2 calculation was not stable, but the overall accuracy was still above 75%. Overall, the calculations of three joint angles were significantly consistent with the expected angles (Actual joint angles of Dobot Magician).

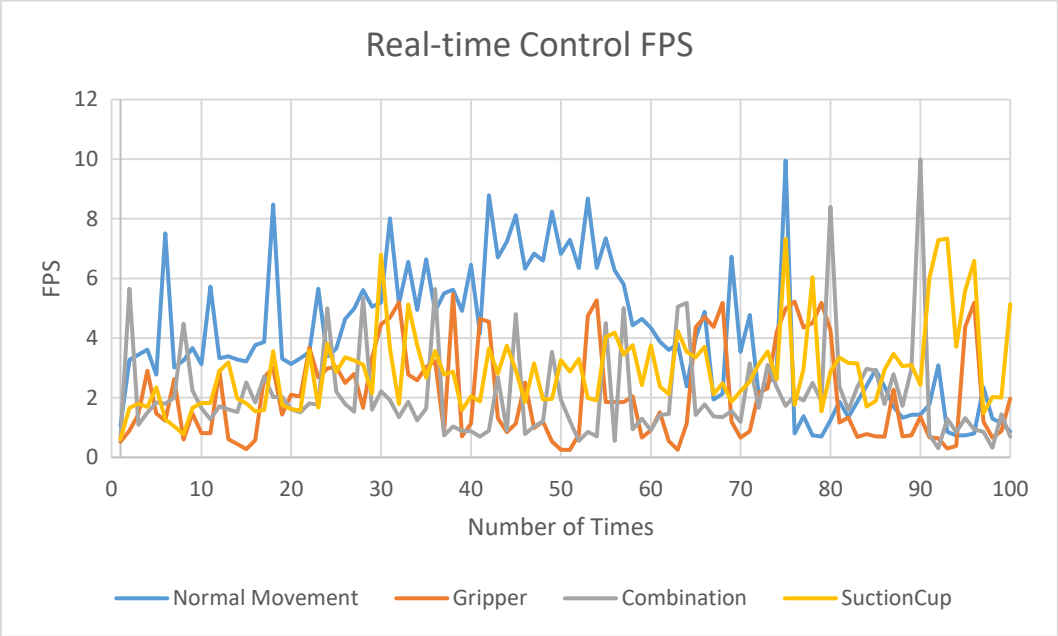

Figure 3.9.5. Implementation FPS

The FPS testing was based on eqn. 3.2.1. In Figure 3.9.5, the legends meanings were as follows: Normal Movement – Ordinary movement (No control of Gripper or Suction Cup), Gripper – Only control Gripper, Combination (Normal Movement+ Gripper/SuctionCup), SuctionCup - Only control suction cup
Based on Figure 3.9.5, the FPS of all movements was in the range of 0 to 10, and the normal movement could be executed under the highest FPS value. In other word, the normal movement was the most potential to achieve real-time control with gestures. The type of Combination was to test its FPS during the comprehensive implementation of gripping the object using a gripper or a suction cup on the desk.

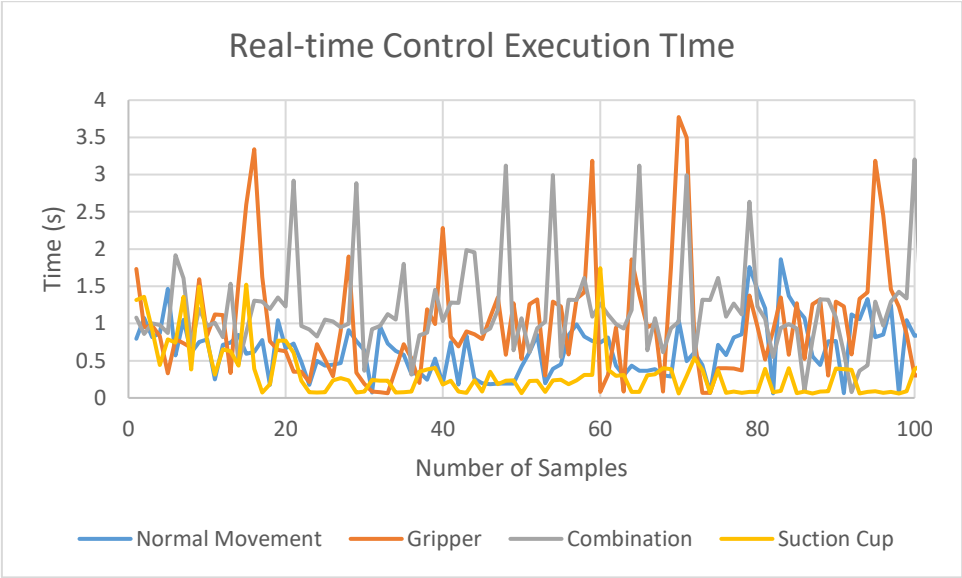

Figure 3.9.6. Execution Time

The testing of execution time was to measure the duration from the completion of previous movement to the completion of next movement to test the delay time. The testing results in Figure 3.9.6 manifested that as the increase of the run time, the Dobot Magician would be more potential to need longer execution time. As mentioned in section 2.2.2.11, the testing results could demonstrate the need of the design that the joint angles set to Dobot Magician were the calculated angles rounded to an integer, and if the joint angles were consistent with the previous joint angles, the joint angles were not changed to reduce the possibility of the increase of the delay time.

3.10 Overall System Testing

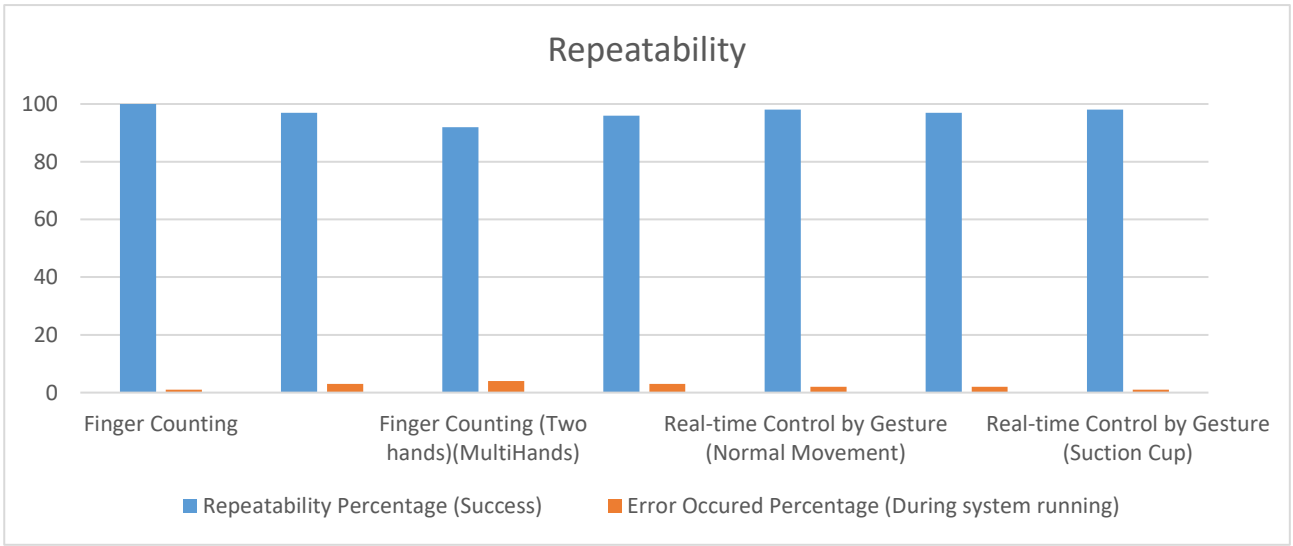

Figure 3.10.1 Overall System Repeatability

The tests of system's repeatability were performed by restarting the program fifty times to determine how many times the system could implement correctly and successfully. The error occur percentage was to indicate how many times the error occurred during the system running. As shown in Figure 3.10.1, the overall system was able to provide the high repeatability with low error occurred percentage for the user. Moreover, in the future work, the error occurred percentage could be improved to 0%, and the repeatability could also be enhanced to be 100%.

4 Conclusion

The project is to realize a gesture-controlled robot arm. The thesis is to present the design of the program system, methodology, testing and validation. The methodology is focused on computer vision and the choice of robot arm is Dobot Magician. The project initially used C++ to fulfil gesture detection and gesture recognition. However, as the consideration of connection with Dobot Magician on the same interface, the coding language was changed to be Python based on its diverse advantages such as providing extensive libraries and accomplishment of connecting Dobot Magician with the laptop and the camera on the same interface.

For computer vision, Xbox Kinect Motion Sensor is used to achieve gesture recognition to perform tasks such as finger counting, multi-hand recognition, hand distance measurement, two-hand finger counting, virtual painting, and real-time gesture recognition to control Dobot Magician in real-time and so on, and the coordinate of the hand is in 3D. As mentioned before, the depth camera is more appropriate than the webcam in this project. Nevertheless, due to the conflict of the Python version, the superiority of depth measurement of Xbox One Kinect Motion Sensor is unable to be utilized directly in the specific Python version, and thus the depth measurement was realized by the combination of calculations and hand information obtained by MediaPipe. Consequently, other depth cameras can be tried in the future. Since the time is limited and limited budget, the tests of the comparison of other cameras cannot be achieved.

For the choice of robot arm, Dobot Magician is a suitable choice in this project based on the consideration of cost-effectiveness, compatibility with microcontrollers, high precision and high accuracy. Most important of all, the workshop in the University of Nottingham can provide Dobot Magician directly without online purchase. Moreover, friendly interface and diverse accessibility mentioned in section 2.1.1 can help understanding Dobot Magician rapidly and provide assistance in data recording, validation and testing. After the completion of the project, the results prove that the capability of Dobot Magician is versatile and sufficient in building a gesture-controlled robot arm, and as Dobot Magician is an industry-standard robot arm mentioned in section 2.1.1, it can help to understand industrial robot arms in advance. As a result, the more powerful functionality robot arm (such as industrial robot arm) can be used to achieve more complicated tasks.

Based on the results in final system testing and validation, all tasks can be performed with high accuracy and high precision. The improvement of complex gesture recognition in left hand and recognition accuracy in messy background and far distance is needed to adjust in the future work. Moreover, the FPS of the implementation of the tasks, particularly in real-time control is required to enhance to provide smoother display. Hence, the future work can also focus on the coding quality and improving execution time.

Overall, all the project specifications are achieved and tested to prove its accomplishments. Moreover, the repeatability of overall system was also tested, and the results manifest that the overall system can provide high repeatability with low error occurred percentage. A gesture-controlled robot arm is built with versatile functionalities. The high precision and high repeatability underscore the superior quality of the system and Dobot Magician constructed in this project and robustness of the comprehensive system created in this project. The accomplishments in this project are potential to achieve the aim of this project that realizing gesture-controlled robot arm for the

healthcare application. The combination of the system and Dobot Magician in this project is possible to help rehabilitation therapists to precisely control patient's range of motion to prevent overextension and ensures that they perform exercises correctly.

4.1 Future Work

Although the robust system and Dobot Magician is built and can perform various complicated tasks under high accuracy, high precision, high repeatability and high efficiency, some adjustments are also needed in the future work. Since the project is required to be finished less than one year and the balance between the coursework and the project is also necessary to be considered, some weakness in the project such as the FPS and the enhancement accuracy in specific tasks cannot be adjusted in time.

Thus, if an extra one or two months are provided, the prior task is to adjust and correct the error and improve the FPS to enhance the system's performance. Moreover, the face depth measurement can be the secondary task to integrate into the system, and the task of advanced virtual painting the user draws on the frame and Dobot Magician drawing the same pattern on paper in real-time can be the secondary task.

Furthermore, if an extra one or two years even more are provided and more budget is given, the tests of the different depth camera can be performed to test the system's ability, and the more advanced robot arm such as industrial robot arms to implement the more complicated and advanced tasks to compare its performance with the Dobot Magician. Additionally, the integration of a robot arm created by 3D printing on an electric vehicle can also be a future task to deal with to fulfil tasks such as automatic warehouse sorting robots or automatic delivery robots to offer a cost-effective alternative for the enterprise or industries to reduce labour expenses.

4.2 Reflection

The planned timeline and the actual timeline are shown in the appendix 6.3 (Figure 6.3.1 and Figure 6.3.2). Five project review meetings are completed, and the project review meetings are shown in the appendix 6.2. The table of risk management is also shown in the appendix. As mentioned in the first project review meeting, the progress of the project followed the timeline to complete the milestone research of Hand Detection in the objective, and the issues happened that the expected Xbox One Kinect Motion Sensor couldn't be purchased from the workshop. Therefore, the completed tasks were tested by the webcam on the laptop rather than Xbox One Kinect Motion Sensor. To prevent the delay of the progress, the setup of Dobot Magician was started to keep the tasks of computer vision and the robot arm could be in progress in parallel. Furthermore, another serious issue was that my laptop was not compatible with the camera's adapter as it required USB 3.0 port. It was solved by loaning a laptop from the IT service in the university. In autumn semester, the majority of research milestone was finished, and the new methods were found and utilized to replace the original method. Before the Christmas break, the hand detection (MediaPipe) in the computer vision tasks was completed, the setup of Xbox One Kinect Motion Sensor and Dobot Magician were also started and almost completed, and the task design and thesis writing (Draft) were started.

In spring semester, the begin progress was to finish the task design, gesture recognition and the adjustment of the codes in the tasks. In the second project review meeting, the basic control of Dobot Magician and gesture recognition were completed,

so the progress followed the timeline. Moreover, the new challenging of real-time gesture control was added to the project, and the prior task was to deal with the data transaction to connect the camera and Dobot Magician with the laptop. In the third project review meeting, all tasks were completed except for real-time gesture control so that the time was spent on code adjustment and real-time gesture control. In the fourth and the fifth project review meeting, the progress was to work on thesis writing, the final code adjustment and testing and validation of the whole system.

Compared the actual timeline to the planned timeline, the tasks and the methods are changed as the conflict and limitation of coding language. Initially, the progress of the computer vision was a bit left behind on account of the lack of the camera. Thus, the progress of robot arms was started in advanced. The overall progress followed the timeline, and the tasks were completed in advanced in spring semester. In the future engineering project planning, the component order should be the priority and the tasks are supposed to be in progress in parallel to prevent that if one of the tasks get stuck, the overall progress will be affected.

For the risk management, as the advice mentioned in the lecture always takes into consideration, the majority of the risks didn’t happen in the process, only the delay of ordering (component), failure in camera (unable to connect with the camera) and laptop malfunction (Compatibility with the camera) happened. In the future, the delay of ordering and compatibility with the system should be considered more, and the conflict of software platform should also be considered.

5 Reference

[1] "Robotic Arm Market Demand, Prize, Scenario, & Forecast Analysis By 2029." https://www.databridgemarketresearch.com/reports/global-robotic-arm-market (accessed April 8, 2024).

[2] J. Ni and V. Balyan, "Research on Mobile User Interface for Robot Arm Remote Control in Industrial Application," *Scalable Computing: Practice and Experience,* vol. 22, 10/24 2021, doi: 10.12694/scpe.v22i2.1900.

[3] M. Sathiyanarayanan and S. Rajan, "MYO Armband for physiotherapy healthcare: A case study using gesture recognition application," in *2016 8th International Conference on Communication Systems and Networks (COMSNETS)*, 5-10 Jan. 2016 2016, pp. 1-6, doi: 10.1109/COMSNETS.2016.7439933.

[4] S. Gokul, R. Dhiksith, S. A. Sundaresh, and M. Gopinath, "Gesture Controlled Wireless Agricultural Weeding Robot," in *2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS)*, 15-16 March 2019 2019, pp. 926-929, doi: 10.1109/ICACCS.2019.8728429.

[5] "Dobot Robotics | Official website." https://www.dobot-robots.com/?utm_source=google&utm_medium=cpc&utm_campaign=fj+dp&gad_source=1&gclid=CjwKCAjwuJ2xBhA3EiwAMVjkVLdIOQ9ht2TLrUSw1Zr2z7OTTG8q4RM80jHAn-sk8aTn_PNFtfN6HBoCsg0QAvD_BwE (accessed April 10, 2024).

[6] A. D. Segal, M. C. Lesak, A. K. Silverman, and A. J. Petruska, "A Gesture-Controlled Rehabilitation Robot to Improve Engagement and Quantify Movement Performance," *Sensors*, vol. 20, no. 15, doi: 10.3390/s20154269.

[7] M. Çoban and G. Gelen, "Wireless teleoperation of an industrial robot by using myo arm band," in *2018 International Conference on Artificial Intelligence and Data Processing (IDAP)*, 28-30 Sept. 2018 2018, pp. 1-6, doi: 10.1109/IDAP.2018.8620789.

[8] D. G. Schneider, L. L. d. Silva, P. Diehl, A. H. R. Leite, and G. S. Bastos, "Robot Navigation by Gesture Recognition with ROS and Kinect," in *2015 12th Latin American Robotics Symposium and 2015 3rd Brazilian Symposium on Robotics (LARS-SBR)*, 29-31 Oct. 2015 2015, pp. 145-150, doi: 10.1109/LARS-SBR.2015.21.

[9] E. Aksoy, A. D. Çakır, B. A. Erol, and A. Gumus, "Real Time Computer Vision Based Robotic Arm Controller with ROS and Gazebo Simulation Environment," in *2023 14th International Conference on Electrical and Electronics Engineering (ELECO)*, 30 Nov.-2 Dec. 2023 2023, pp. 1-5, doi: 10.1109/ELECO60389.2023.10416078.

[10] A. A. Farouq and D. M. B. Santoso, "Revolutionizing Robot Sorting: Unleashing the Power of Dobot Magician Arm's Kinematics Inverse Transformation System with Depth Camera," in *2023 IEEE International Conference on Communication, Networks and Satellite (COMNETSAT)*, 23-25 Nov. 2023 2023, pp. 428-433, doi: 10.1109/COMNETSAT59769.2023.10420575.

[11] "Dobot Magician Interface Description (V2)," ed, 2019.

[12] "Dobot Magician User Guide," ed, 2020.

[13] M. M. F. M. Fareed, Q. I. Akram, S. B. A. Anees, and A. H. Fakih, "Gesture Based Wireless Single-Armed Robot in Cartesian 3D Space Using Kinect," in *2015 Fifth International Conference on Communication Systems and Network Technologies*, 4-6 April 2015 2015, pp. 1210-1215, doi: 10.1109/CSNT.2015.86.

[14] K. Suphalak, N. Klanpet, N. Sikaressakul, and S. Prongnuch, "Robot Arm Control System via Ethernet with Kinect V2 Camera for use in Hazardous Areas," in *2024 1st International Conference on Robotics, Engineering, Science, and Technology (RESTCON)*, 16-18 Feb. 2024 2024, pp. 175-180, doi: 10.1109/RESTCON60981.2024.10463582.

[15] A. Al-Naji, K. Gibson, S. H. Lee, and J. Chahl, "Real Time Apnoea Monitoring of Children Using the Microsoft Kinect Sensor: A Pilot Study," (in eng), *Sensors (Basel),* vol. 17, no. 2, Feb 3 2017, doi: 10.3390/s17020286.

[16] D. Pagliari and L. Pinto, "Calibration of Kinect for Xbox One and Comparison between the Two Generations of Microsoft Sensors," *Sensors*, vol. 15, no. 11, pp. 27569-27589doi: 10.3390/s151127569.

[17] K. H. Chen, Y. T. Hwang, and C. P. Fan, "Sensor Data Fusion of Intelligent Autonomous Mover for Object Detection and Collision Avoidance in Environments with Surrounding

Crowds," in *2021 IEEE International Conference on Consumer Electronics (ICCE)*, 10-12 Jan. 2021 2021, pp. 1-4, doi: 10.1109/ICCE50685.2021.9427728.
[18] P. A. Parikh, K. D. Joshi, and R. Trivedi, "Face Detection-Based Depth Estimation by 2D and 3D Cameras: A Comparison," in *2022 28th International Conference on Mechatronics and Machine Vision in Practice (M2VIP)*, 16-18 Nov. 2022 2022, pp. 1-4, doi: 10.1109/M2VIP55626.2022.10041072.
[19] A. D. Calin, "Gesture Recognition on Kinect Time Series Data Using Dynamic Time Warping and Hidden Markov Models," in *2016 18th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, 24-27 Sept. 2016 2016, pp. 264-271, doi: 10.1109/SYNASC.2016.049.
[20] S. G. Heng, R. Samad, M. Mustafa, N. R. H. Abdullah, and D. Pebrianti, "Analysis of Performance between Kinect v1 and Kinect v2 for Various Facial Part Movements," in *2019 IEEE 9th International Conference on System Engineering and Technology (ICSET)*, 7-7 Oct. 2019 2019, pp. 17-22, doi: 10.1109/ICSEngT.2019.8906419.
[21] J. Mistry and B. Inden, "An Approach to Sign Language Translation using the Intel RealSense Camera," in *2018 10th Computer Science and Electronic Engineering (CEEC)*, 19-21 Sept. 2018 2018, pp. 219-224, doi: 10.1109/CEEC.2018.8674227.
[22] H. Alaoui, M. T. Moutacalli, and M. Adda, "AI-Enabled High-Level Layer for Posture Recognition Using The Azure Kinect in Unity3D," in *2020 IEEE 4th International Conference on Image Processing, Applications and Systems (IPAS)*, 9-11 Dec. 2020 2020, pp. 155-161, doi: 10.1109/IPAS50080.2020.9334945.
[23] "Official Xbox One Kinect Sensor: Amazon.co.uk: PC & Video Games." Amazon.co.uk. https://www.amazon.co.uk/Official-Xbox-One-Kinect-Sensor/dp/B00NABN4VS (accessed April 16, 2024).
[24] "Official Xbox 360 Kinect Sensor with Kinect Adventures (Xbox 360) : Amazon.co.uk: PC & Video Games." www.amazon.co.uk. https://www.amazon.co.uk/Official-Xbox-Kinect-Sensor-Adventures/dp/B0036DDW2G (accessed April 18, 2024).
[25] "Intel® RealSense™ Depth Camera D456." store.intelrealsense.com. https://store.intelrealsense.com/buy-intel-realsense-depth-camera-d456.html (accessed April 20, 2024).
[26] "Buy the Azure Kinect developer kit – Microsoft." Microsoft Store. https://www.microsoft.com/en-us/d/azure-kinect-dk/8pp5vxmd9nhq?activetab=pivot:overviewtab (accessed April 19, 2024).
[27] A. Krishan Kumar, A. Kaushal Kumar, and S. Guo, "Two viewpoints based real-time recognition for hand gestures," *IET Image Processing,* vol. 14, no. 17, pp. 4606-4613, 2020/12/01 2020, doi: https://doi.org/10.1049/iet-ipr.2019.1458.
[28] P. Carbonnelle. "PYPL PopularitY of Programming Language index." Github.io. https://pypl.github.io/PYPL.html (accessed April 22, 2024).
[29] "Dobot Magician Demo Description," ed: DOBOT, 2022.
[30] R. Shrivastava, "A hidden Markov model based dynamic hand gesture recognition system using OpenCV," in *2013 3rd IEEE International Advance Computing Conference (IACC)*, 22-23 Feb. 2013 2013, pp. 947-950, doi: 10.1109/IAdCC.2013.6514354.
[31] OpenCV. "About OpenCV." OpenCV. https://opencv.org/about/ (accessed April 25, 2024).
[32] A. Y. Bahar, S. M. Shorman, M. A. Khder, A. M. Quadir, and S. A. Almosawi, "Survey on Features and Comparisons of Programming Languages (PYTHON, JAVA, AND C#)," in *2022 ASU International Conference in Emerging Technologies for Sustainability and Intelligent Systems (ICETSIS)*, 22-23 June 2022 2022, pp. 154-163, doi: 10.1109/ICETSIS55481.2022.9888839.
[33] J. Qi, L. Ma, Z. Cui, and Y. Yu, "Computer vision-based hand gesture recognition for human-robot interaction: a review," *Complex & Intelligent Systems,* vol. 10, no. 1, pp. 1581-1606, 2024/02/01 2024, doi: 10.1007/s40747-023-01173-6.
[34] S. Salim, M. M. A. Jamil, R. Ambar, W. S. W. Zaki, and S. Mohammad, "Learning Rate Optimization for Enhanced Hand Gesture Recognition using Google Teachable Machine," in *2023 IEEE 13th International Conference on Control System, Computing and Engineering (ICCSCE)*, 25-26 Aug. 2023 2023, pp. 332-337, doi: 10.1109/ICCSCE58721.2023.10237148.
[35] S. Prakash, S. S. Kumar, M. Abhijith, and R. J. Menezes, "Action Recognition to Sign Language Recognition using Machine Learning Techniques," in *2022 International*

Conference on Computing, Communication, Security and Intelligent Systems (IC3SIS), 23-25 June 2022 2022, pp. 1-6, doi: 10.1109/IC3SIS54991.2022.9885720.
[36] S. Salim, M. M. A. Jamil, R. Ambar, R. Roslan, and M. G. Kamardan, "Sign Language Digit Detection with MediaPipe and Machine Learning Algorithm," in *2022 IEEE 12th International Conference on Control System, Computing and Engineering (ICCSCE)*, 21-22 Oct. 2022 2022, pp. 180-184, doi: 10.1109/ICCSCE54767.2022.9935659.
[37] "Teachable Machine." teachablemachine.withgoogle.com. https://teachablemachine.withgoogle.com/train/image (accessed April 17, 2024).
[38] "OpenCV: OpenCV modules." docs.opencv.org. https://docs.opencv.org/4.x/index.html (accessed April 24, 2024).
[39] "Hand landmarks detection guide | MediaPipe." Google Developers. https://developers.google.com/mediapipe/solutions/vision/hand_landmarker (accessed April 24, 2024).
[40] CVZone. "CVZone." https://github.com/cvzone/cvzone (accessed April 16, 2024).
[41] P. S. Foundation. "Math — Mathematical Functions — Python 3.8.3rc1 Documentation." docs.python.org. https://docs.python.org/3/library/math.html (accessed April 17, 2024).
[42] "numbers — Numeric abstract base classes." Python documentation. https://docs.python.org/3/library/numbers.html (accessed May 1, 2024).
[43] "numpy/numpy," ed: GitHub, 2021.
[44] N. Rajule, S. Pawar, S. Jadhav, U. Jambhulkar, D. Kapse, and O. Kanthale, "Real-Time Hand Gesture Recognition with Voice Conversion for Deaf and Dumb," in *2023 7th International Conference On Computing, Communication, Control And Automation (ICCUBEA)*, 18-19 Aug. 2023 2023, pp. 1-7, doi: 10.1109/ICCUBEA58933.2023.10392183.
[45] "TensorFlow." TensorFlow. https://www.tensorflow.org/?hl=zh-tw (accessed April 18, 2024).
[46] tensorflow, "tensorflow/tensorflow," ed: GitHub, 2019.
[47] Python. "os — Miscellaneous operating system interfaces — Python 3.8.0 documentation." Python.org. https://docs.python.org/3/library/os.html (accessed April 23, 2024).
[48] P. S. Foundation. "time — Time access and conversions — Python 3.7.2 documentation." Python.org. https://docs.python.org/3/library/time.html (accessed.
[49] M. Bensaadallah, N. Ghoggali, L. Saidi, and W. Ghoggali, "Deep Learning-Based Real-Time Hand Landmark Recognition with MediaPipe for R12 Robot Control," in *2023 International Conference on Electrical Engineering and Advanced Technology (ICEEAT)*, 5-7 Nov. 2023 2023, vol. 1, pp. 1-6, doi: 10.1109/ICEEAT60471.2023.10426264.
[50] "layout: forward target: https://developers.google.com/mediapipe/solutions/vision/hand_landmarker title: Hands parent: MediaPipe Legacy Solutions nav_order: 4 — MediaPipe v0.7.5 documentation." mediapipe.readthedocs.io. https://mediapipe.readthedocs.io/en/latest/solutions/hands.html (accessed.
[51] L. Ge, H. Liang, J. Yuan, and D. Thalmann, "Robust 3D Hand Pose Estimation in Single Depth Images: from Single-View CNN to Multi-View CNNs," *arXiv (Cornell University),* 2016, doi: 10.48550/arxiv.1606.07253.
[52] F. Zhang *et al.*, "MediaPipe Hands: On-device Real-time Hand Tracking," *arXiv:2006.10214 [cs],* 2020, doi: https://doi.org/10.48550/arXiv.2006.10214
[53] M. s. Workshop. "Hand Sign Detection (ASL)." Computer Vision Zone. https://www.computervision.zone/courses/hand-sign-detection-asl/ (accessed.
[54] A. S. S. Urs, V. B. Raj, S. P, P. K. K, B. R. M, and V. K. S, "Action Detection for Sign Language Using Machine Learning," in *2023 International Conference on Network, Multimedia and Information Technology (NMITCON)*, 1-2 Sept. 2023 2023, pp. 1-6, doi: 10.1109/NMITCON58196.2023.10275950.
[55] M. s. Workshop. "Finger Counter." Computer Vision Zone. https://www.computervision.zone/courses/finger-counter/ (accessed March 8, 2024).
[56] M. s. Workshop. "Multiple Hand Gesture Control." Computer Vision Zone. https://www.computervision.zone/courses/multiple-hand-gesture-control/ (accessed April 9, 2024).
[57] M. s. Workshop. "AI Virtual Painter." Computer Vision Zone. https://www.computervision.zone/courses/ai-virtual-painter/ (accessed April 20, 2024).

[58] T. Sravya, S. N. Bhargava, S. S, R. Bodke, and N. Kulkarni, "AI Virtual Hardware," in *2022 IEEE Pune Section International Conference (PuneCon)*, 15-17 Dec. 2022 2022, pp. 1-6, doi: 10.1109/PuneCon55413.2022.10014775.

[59] H. Li, C. Dong, X. Jia, S. Xiang, and Z. Hu, "Design of Automatic Sorting and Transportation System for Fruits and Vegetables Based on Dobot Magician Robotic Arm," in *2023 2nd International Symposium on Control Engineering and Robotics (ISCER)*, 17-19 Feb. 2023 2023, pp. 229-234, doi: 10.1109/ISCER58777.2023.00045.

[60] Sam. "Crazycurly/gesture_MeArm." https://github.com/Crazycurly/gesture_MeArm (accessed April 28, 2024).

[61] "google/mediapipe." GitHub. https://github.com/google/mediapipe/blob/master/docs/solutions/hands.md (accessed April 17, 2024).

[62] C. Cheng-Hung and N. Desineni Subbaram, "Kinematics and Trajectory Planning," in *Fusion of Hard and Soft Control Strategies for the Robotic Hand*: IEEE, 2018, pp. 47-91.

6 Appendix

6.1 Reflection Meeting

6.1.1 Project Review Meeting 1: 24/11/2023

Summarised Planned State of Project: *[This section should summarise where you expected to be with the project at this review stage according to your Time Plan.]* *The progress of the project follows the Time Plan.*	**Actual Progress Since Last Review** *[Summarise the actual progress you have made on the project since previous Progress Review (or start of project). You should consider relevant deliverables, milestones, and Gannt chart tasks.]* *I have done the health and safety paperwork, and have submitted the project planning report. For the milestone of the research, I have determined the methods of each part of the tasks and the methods for each section and the reasons why I choose the methods.* *According to Time Plan, the milestone of the research should be done.*
Next Steps and Supervisor Feedback *[In this section you should consider the differences between where you planned to be and where you are. How will this affect the project? What are you going to do to get back on track? Are there on-going problems that will have a knock-on effect to future tasks? Is there additional support or resource that is required? You should include feedback from your supervisor as discussed in the meeting and their approval of changes.]* *The progress is following the time plan. Next, I need to start to do developing computer vision to hand tracking.* *From the supervisor feedback: Hopefully the components can arrive as soon as possible, and the progress of the gesture recognition can go through well.* *After the components are ready, the first part of image processing of filtering will be accomplished before Christmas Break.*	

6.1.2 Project Review Meeting 2: 8/2/2024

Summarised Planned State of Project: *[This section should summarise where you expected to be with the project at this review stage according to your Time Plan.]* • Completion of Research Milestone (detail which) • Completion of all subtasks in Task Filtering (indicate overall position so give this context) • Completion of edge detection and evaluation in Task Isolation (same as previous, give it some context)	**Actual Progress Since Last Review** *[Summarise the actual progress you have made on the project since previous Progress Review (or start of project). You should consider relevant deliverables, milestones, and Gannt chart tasks.]* • Complete Research Milestone • Complete all subtasks in Task Filtering • Complete all subtasks in Task Isolation • Complete the basic control of Dobot Magician • Gesture Recognition and Robot Arm control are synchronized (Parallel) (indicate what this means – is it ahead of schedule, how much more to do, what next for example).

Next Steps and Supervisor Feedback

[In this section you should consider the differences between where you planned to be and where you are. How will this affect the project? What are you going to do to get back on track? Are there on-going problems that will have a knock-on effect to future tasks? Is there additional support or resource that is required? You should include feedback from your supervisor as discussed in the meeting and their approval of changes.]

- The progress is a bit ahead of schedule so that the project is expected to be completed in advanced.
- The on-going problem is to solve the integration of robot arm and gesture recognition such as data transaction and communication.
- Some good progress has been made and it is looking to get ahead and do more challenging things.

6.1.3 Project Review Meeting 3: 8/3/2024

<table>
<tr>
<td>Summarised Planned State of Project:
[This section should summarise where you expected to be with the project at this review stage according to your Time Plan.]
Expectation (Time Plan):
• Completion of Research Milestone of gesture detection and gesture recognition
• Completion of Developing Computer Vision to Hand Tracking Milestone of objective gesture detection</td>
<td>Actual Progress Since Last Review
[Summarise the actual progress you have made on the project since previous Progress Review (or start of project). You should consider relevant deliverables, milestones, and Gannt chart tasks.]
• Research Milestone is completed
• Developing Computer Vision to Hand Tracking Milestone is completed
• Communication to Robot Arm Controller Milestone is completed
• Subtasks of command generation in Interface and Control Movement with Robot Arm is completed
• Working on the adjustment of system efficiency and real time control of Dobot Magician</td>
</tr>
<tr>
<td colspan="2">Next Steps and Supervisor Feedback
[In this section you should consider the differences between where you planned to be and where you are. How will this affect the project? What are you going to do to get back on track? Are there on-going problems that will have a knock-on effect to future tasks? Is there additional support or resource that is required? You should include feedback from your supervisor as discussed in the meeting and their approval of changes.]
The majority of milestones are completed, and the thesis writing can be started. Moreover, the real time control of Dobot Magician can be in progress in parallel.
Supervisor Feedback: Good progress but time to get on to writing. Need to think about and just finish off 'testing and results' but otherwise good work that just needs to be written down.</td>
</tr>
</table>

6.1.4 Project Review Meeting 4: 21/3/2024

<table>
<tr>
<td>Summarised Planned State of Project:
[This section should summarise where you expected to be with the project at this review stage according to your Time Plan.]
Expectation (Time Plan):
Completion of Research Milestone of gesture detection and gesture recognition
Completion of Developing Computer Vision to Hand Tracking Milestone of objective gesture detection
Completion of objective gesture recognition of quadrant control and hand pose detection</td>
<td>Actual Progress Since Last Review
[Summarise the actual progress you have made on the project since previous Progress Review (or start of project). You should consider relevant deliverables, milestones, and Gannt chart tasks.]
Research Milestone is completed
Developing Computer Vision to Hand Tracking Milestone is completed
Communication to Robot Arm Controller Milestone is completed
Subtasks of command generation in Interface and Control Movement with Robot Arm is completed
Working on the adjustment of system efficiency and real time control of Dobot Magician
Working on the draft of thesis</td>
</tr>
<tr>
<td colspan="2">Next Steps and Supervisor Feedback
[In this section you should consider the differences between where you planned to be and where you are. How will this affect the project? What are you going to do to get back on track? Are there on-going problems that will have a knock-on effect to future tasks? Is there additional support or resource that is required? You should include feedback from your supervisor as discussed in the meeting and their approval of changes.]
The majority of milestones are completed, and the draft of thesis writing can be started. Additionally, the real time control of Dobot Magician can be in progress in parallel.
Supervisor Feedback: Good progress. Comes to meetings with good questions about ongoing work/writing. Should focus on writing and tests but if there is time can stretch to additional functionality – with time.</td>
</tr>
</table>

6.1.5 Project Review Meeting 5: 28/3/2024

Summarised Planned State of Project: *[This section should summarise where you expected to be with the project at this review stage according to your Time Plan.]* *Expectation (Time Plan):* • Completion of Research Milestone of gesture detection and gesture recognition • *Completion of Developing Computer Vision to Hand Tracking Milestone of objective gesture detection* • *Completion of Communication to Robot Arm Controller* • *Completion of Interface and Control Movement with Robot Arm* • *Completion of Emergency Stop Mechanisms* • *Completion of Draft*	**Actual Progress Since Last Review** *[Summarise the actual progress you have made on the project since previous Progress Review (or start of project). You should consider relevant deliverables, milestones, and Gannt chart tasks.]* • *Research Milestone is completed* • *Developing Computer Vision to Hand Tracking Milestone is completed* • *Communication to Robot Arm Controller Milestone is completed* • *Interface and Control Movement with Robot Arm Milestone is completed* • *Completion of Draft* • *Completion of Emergency Stop Mechanism* • *Working on the adjustment of system efficiency and real time control of Dobot Magician* • *Working on the thesis* • *Working on Real Time Drawing with Dobot Magician*
Next Steps and Supervisor Feedback *[In this section you should consider the differences between where you planned to be and where you are. How will this affect the project? What are you going to do to get back on track? Are there on-going problems that will have a knock-on effect to future tasks? Is there additional support or resource that is required? You should include feedback from your supervisor as discussed in the meeting and their approval of changes.]* *All milestones are completed, and the draft of thesis is also completed. Additionally, the real time control of Dobot Magician and real-time drawing with Dobot Magician can be in progress in parallel. The priority task is the completion of the project thesis.* *Supervisor Feedback: G*ood progress made and now writing. Priority is to finish the thesis and use feedback for the last bits.	

6.2 Risk Management

Table 6.2.1. Risk Management

Risk	Level	Effect	Solution
Saving Lost	Low	Data Lost	Save the data to OneDrive or email to personal email
Ordering Delay	Medium	Progression Delay	Communicate with the customer service to resolve the problem
Failure in Equipment such as ESP32, Camera	Medium	Progression Delay	Within budget, purchase the backup equipment
Robot arm out of control	High	Causing injury to the user	Developing emergency stop mechanisms
Laptop Malfunction	High	Progression Delay	Borrow a laptop from IT Service and ask the relevant personnel to repair it
Failure in Robot Arm	High	Progression Delay	Repair the robot arm by ourselves then ask the supervisor to provide the assistance if necessary

6.3 Time Plan

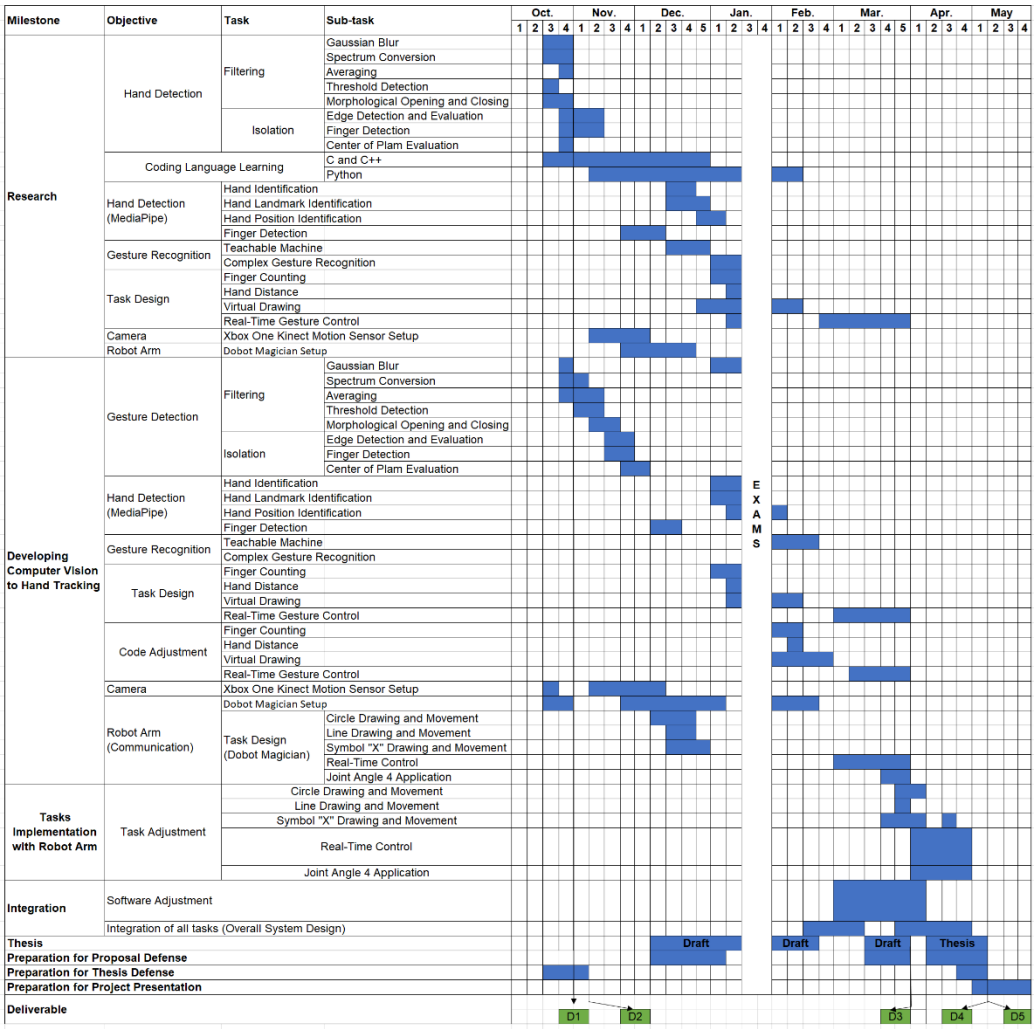

Figure 6.3.1 Actual Timeline

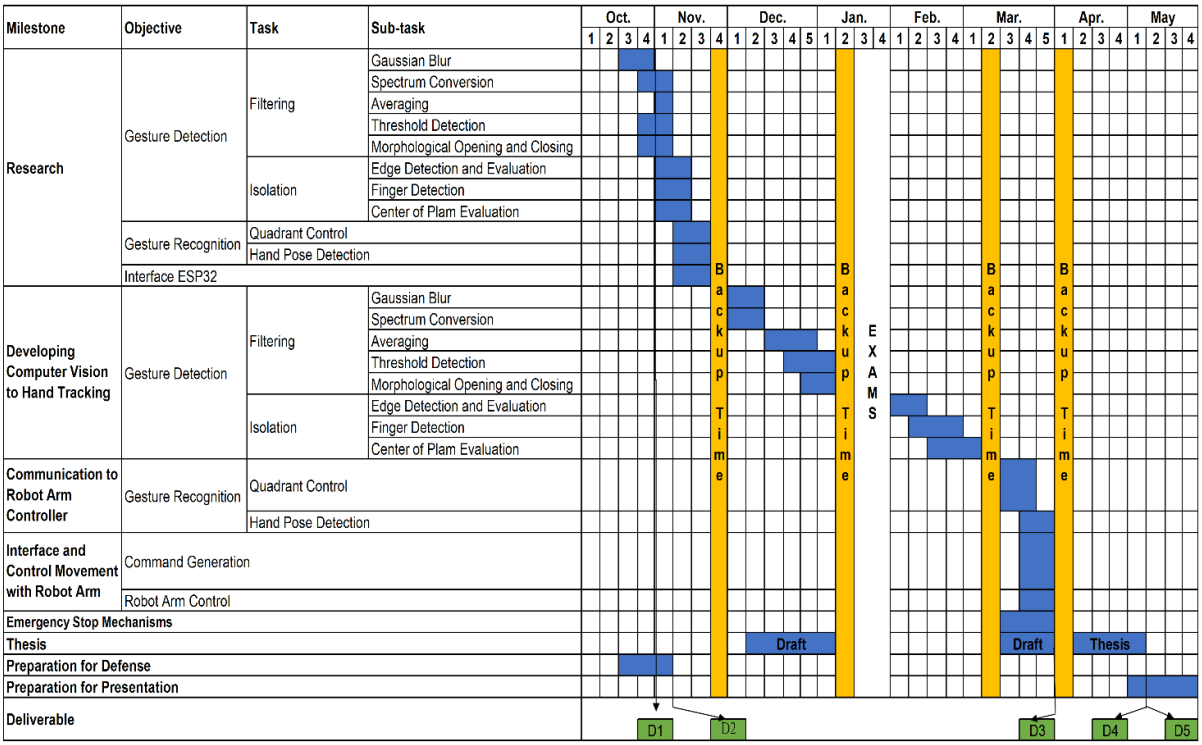

Figure 6.3.2 Original Timeline