Virtual Pointer Using Hand Gesture Recognition

A project presented to the National University in partial fulfillment of the requirement for the degree of Bachelor of Science (Hon's) in Computer Science & Engineering.

Supervised by

Prof.Dr.Mohammed Shakhawat Hossain

Principal,DIIT

Submitted by

Md. Fuad Alam

Registration No: 17502004999

Session: 2017-18

Department of Computer Science & Engineering

Daffodil Institute of IT, Dhaka

Under National University Bangladesh

Submission date: 30/08/23

APPROVAL

The Project "**Virtual Pointer Using Hand Gesture Recognition**" is submitted to the Department of Computer Science & Engineering, DIIT under the National University of Bangladesh in partial fulfillment of the requirements for the degree of Bachelor of Science (Hon's) in Computer Science and Engineering and approved as to its style and content.

Examiner

Examiner

.....

Prof. Dr. Mohammad Shakhawat Hossain
Supervisor,
Principal, Department of CSE, DIIT

.....

Md. Imran Hossain
Head
Department of CSE
Daffodil Institute of IT

DECLARATION

We affirm that the project work titled "**Virtual Pointer Using Hand Gesture Recognition**" is being submitted in partial fulfillment for the degree of B.Sc. (Hon's) in Computer Science & Engineering is the original work carried out by me. It has not formed the part of any other project work submitted for any degree or diploma, either in this or any other university

Submitted By:

Md. Fuad Alam
Registration No: 17502004999
Session: 2017-18

ACKNOWLEDGMENT

I would like to express my profound gratitude to Almighty Allah. With the blessing of Almighty Allah, I have successfully planned my project.

My sincere thanks to **Prof.Dr.Mohammed shakhawat Hossain**, Associate Professor and Principal, DIIT who has allowed me to do this project and encouragement given to me. Also, thanks for his valuable guidance and support to meet the successful completion of my project.

I would also like to thank my Project Supervisor **Prof.Dr.Mohammed shakhawat Hossain** Principal, Department of Computer Science & Engineering, DIIT, for her guidance and support to meet the successful completion of my Lecturer.

My heartiest thanks **Md. Imran Hossain**,Senior Lecturer & Head of Department, DIIT, Dhaka, for his patronage and giving me an opportunity to undertake this Project.

I express my gratitude to **Poly Bhoumik**, Senior Lecturer, DIIT, Dhaka, for having provided us the facilities to do the project successfully.

I express my gratitude to **Md. Saidur Rahman**,Senior Lecturer & Batch Co-Ordinator, DIIT, Dhaka, for having provided us the facilities to do the project successfully.

I express my gratitude to **Safrun Nesa Saira**, Lecturer, DIIT, Dhaka, for having provided us the facilities to do the project successfully.

I express my gratitude to **Nusrhat Jahan Sarkar**, Lecturer, DIIT, Dhaka, for having provided us the facilities to do the project successfully.

I express my gratitude to **Mizanur Rahman**, Lecturer, DIIT, Dhaka, for having provided us the facilities to do the project successfully.

I express my gratitude to **Moumita Akter**, Lecturer, DIIT, Dhaka, for having provided us the facilities to do the project successfully.

1 express my gratitude to **Md. Mushfiqur Rahaman**, Lecturer, DIIT, Dhaka, for having provided us the facilities to do the project successfully.

Last but not the least, I extend my sincere thanks to my family members and my friends for their constant support throughout this project.

ABSTRACT

The field of human-computer interaction has seen tremendous advancement in recent years. In this project, we used a Human Computer Interaction strategy in which we aimed to include the use of a hardware mouse and control the mouse points with hand gestures and colour recognition. With the use of a camera and a colour recognition technology, hand motions were captured. We have attempted to remove the boundaries of contact between humans and computers by replacing current hardware with gestures, motivated by the thought that we can communicate with the computer system. The goal is to move the mouse pointer on the screen without using any hardware, such as a mouse, and instead by utilising finger motions, i.e. the gesture recognition process. Different technologies have been explored in the development of virtual mice in recent years. Our project's suggested technology focuses on three main areas: object identification, picture processing, and colour recognition. We demonstrate an innovative method to Human Computer Interaction in this research, in which cursor movement is controlled by a real-time camera.

Virtual Pointer Using Hand Gesture Recognition

Table of contents

Contents

Chapter 1

Introduction

1.1 Introduction

With the development technologies in the areas of augmented reality and devices that we use in our daily life, these devices are becoming compact in the form of Bluetooth or wireless technologies. This paper proposes an AI virtual mouse system that makes use of the hand gestures and hand tip detection for performing mouse functions in the computer using computer vision. The main objective of the proposed system is to perform computer mouse cursor functions and scroll function using a web camera or a built-in camera in the computer instead of using a traditional mouse device. Hand gesture and hand tip detection by using computer vision is used as a HCI [1] with the computer. With the use of the AI virtual mouse system, we can track the fingertip of the hand gesture by using a built-in camera or web camera and perform the mouse cursor operations and scrolling function and also move the cursor with it.

While using a wireless or a Bluetooth mouse, some devices such as the mouse, the dongle to connect to the PC, and also, a battery to power the mouse to operate are used, but in this paper, the user uses his/her built-in camera or a webcam and uses his/her hand gestures to control the computer mouse operations. In the proposed system, the web camera captures and then processes the frames that have been captured and then recognizes the various hand gestures and hand tip gestures and then performs the particular mouse function.

Python programming language is used for developing the AI virtual mouse system also, OpenCV which is the library for computer vision is used in the AI virtual mouse system. In the proposed AI virtual mouse system, the model makes use of the MediaPipe package for the tracking of the hands and for tracking of the tip of the hands, and also, Pynput, Autopy, and PyAutoGUI packages were used for moving around the window screen of the computer for performing functions such as left click, right click, and scrolling functions. The results of the proposed model showed very high accuracy level, and the proposed model can work very well in real-world application with the use of a CPU without the use of a GPU. The results of the proposed model showed very high accuracy level, and the proposed model can work very well in real-world application with the use of a CPU without the use of a GPU

1.2. Objective

The main objective of the proposed AI virtual mouse system is to develop an alternative to the regular and traditional mouse system to perform and control the mouse functions, and this can be achieved with the help of a web camera that captures

the hand gestures and hand tip and then processes these frames to perform the particular mouse function such as left click, right click, and scrolling function. The purpose of this project is to develop a Virtual Mouse application that targets a few aspects of significant development. For starters, this project aims to eliminate the needs of having a physical mouse while able to interact with the computer system through webcam by using various image processing techniques. Other than that, this project aims to develop a Virtual Mouse application that can be operational on all kind of surfaces and environment. IA(HONS) Information System Engineering Faculty of Information and Communication Technology (Perak Campus), UTAR 7 The following describes the overall objectives of this project:

- To design to operate with the help of a webcam. The Virtual Mouse application will be operational with the help of a webcam, as the webcam are responsible to capture the images in real time. The application would not work if there are no webcam detected.
- To design a virtual input that can operate on all surface. The Virtual Mouse application will be operational on all surface and indoor environment, as long the users are facing the webcam while doing the motion gesture.
- To program the camera to continuously capturing the images, which the images will be analysed, by using various image processing techniques. As stated above, the Virtual Mouse application will be continuously capturing the images in real time, where the images will be undergo a series of process, this includes HSV conversion, Binary Image conversion, salt and pepper noise filtering, and more.
- To convert hand gesture/motion into mouse input that will be set to a particular screen position. The Virtual Mouse application will be programmed to detect the position of the defined colours where it will be set as the position of the mouse pointers. Furthermore, a combination of different colours may result in triggering different types of mouse events.

1.3 Project Scope

Virtual Mouse that will soon to be introduced to replace the physical computer mouse to promote convenience while still able to accurately interact and control the

computer system. To do that, the software requires to be fast enough to capture and process every image, in order to successfully track the user's gesture. Therefore, this project will develop a software application with the aid of the latest software coding technique and the open-source computer vision library also known as the OpenCV. The scope of the project is as below:

- Real time application.
- User friendly application.
- Removes the requirement of having a physical mouse.

The process of the application can be started when the user's gesture was captured in real time by the webcam, which the captured image will be processed for segmentation to identify which pixels values equals to the values of the defined colour. After the segmentation is completed, the overall image will be converted to Binary Image where the identified pixels will show as white, while the rest are black. The position of the white segment in the image will be recorded and set as the position of the mouse pointer, thus resulting in simulating the mouse pointer without using a physical computer mouse. The software application is compatible with the Windows platform. The functionality of the software will be coded with C++ programming language code with the integration of an external library that does the image processing known as the OpenCV. The process of the application can be started when the user's gesture was captured in real time by the webcam, which the captured image will be processed for segmentation to identify which pixels values equals to the values of the defined colour. After the segmentation is completed, the overall image will be converted to Binary Image where the identified pixels will show as white, while the rest are black.

1.4 Impact, Significance and Contribution

The Virtual Mouse application is expected to replace the current methods of utilizing a physical computer mouse where the mouse inputs and positions are done manually. This application offers a more effortless way to interact with the computer system, where every task can be done by gestures. Furthermore, the Virtual Mouse application

could assist the motor-impaired users where he/she could interact with the computer system by just showing the correct combination of colours to the webcam.

1.5 Problem Description and Overview

The proposed AI virtual mouse system can be used to overcome problems in the real world such as situations where there is no space to use a physical mouse and also for the persons who have problems in their hands and are not able to control a physical mouse. Also, amidst of the COVID-19 situation, it is not safe to use the devices by

touching them because it may result in a possible situation of spread of the virus by touching the devices, so the proposed AI virtual mouse can be used to overcome these problems since hand gesture and hand Tip detection is used to control the PC mouse functions by using a webcam or a built-in camera. touching them because it may result in a possible situation of spread of the virus by touching the devices, so the proposed AI virtual mouse can be used to overcome these problems since hand gesture and hand Tip detection is used to control the PC mouse functions by using a webcam or a built-in camera

1.6 Limitation of the Existing system

- To detect a face, the candidate must be in an area with light
- The accuracy of the system is not 100%
- Face detection and loading training data processes just a little bit slow
- The instructor and trainingSet manager still have to do some work manually
- It can only detect face from a limited distance.

1.7 Features

1.Gesture-Based Control: Users can manipulate a virtual pointer on-screen by performing hand gestures,enabling intuitive and natural interaction.

2.Real-Time Recognition: The system identifies and interprets hand gestures in real-time, providing instant responses to users' actions.

3.Diverse Gestures: The system supports a range of gestures, such as pointing, clicking, dragging, scrolling, and more, allowing for versatile interactions.

4.Accurate Tracking: Utilizing depth-sensing technology, the system accurately tracks hand movements in three dimensions, ensuring precise gesture recognition.

5.Customizable Actions: Users can map specific gestures to customizable actions, tailoring the interaction experience to their preferences and needs.

6.Gesture Learning: The system learns and adapts to users' individual gesture styles over time, optimizing recognition accuracy.

7.Accessibility: Individuals with mobility challenges can engage with digital content more easily, promoting inclusivity and equal access.

8.Multi-Modal Integration: The system seamlessly integrates with other input methods like voice commands or touch, expanding the range of interactions.

9.Immersive Experience: Gesture-based control immerses users in digital environments, enhancing engagement and interaction depth.

10.Applications Across Domains: From gaming and presentations to design and virtual reality, the system finds applications across a wide array of domains, enriching user experiences

Chapter 2
Background Study

2.1 Review Of The Physical Mouse

It is known that there are various types of physical computer mouse in the modern technology, the following will discuss about the types and differences about the physical mouse.

2.1.1 Mechanical Mouse

Known as the trackball mouse that is commonly used in the 1990s, the ball within the mouse are supported by two rotating rollers in order to detect the movement made by the ball itself. One roller detects the forward/backward motion while the other detects the left/right motion. The ball within the mouse are steel made that was covered with a layer of hard rubber, so that the detection are more precise. The common functions included are the left/right buttons and a scroll-wheel. However, due to the constant friction made between the mouse ball and the rollers itself, the mouse are prone to degradation, as overtime usage may cause the rollers to degrade, thus causing it to unable to detect the motion properly, rendering it useless. Furthermore, the switches in the mouse buttons are no different as well, as long term usage may cause the mechanics within to be loosed and will no longer detect any mouse clicks till it was disassembled and repaired. The results of the proposed model showed very high accuracy level, and the proposed model can work very well in real-world application with the use of a CPU without the use of a GPU

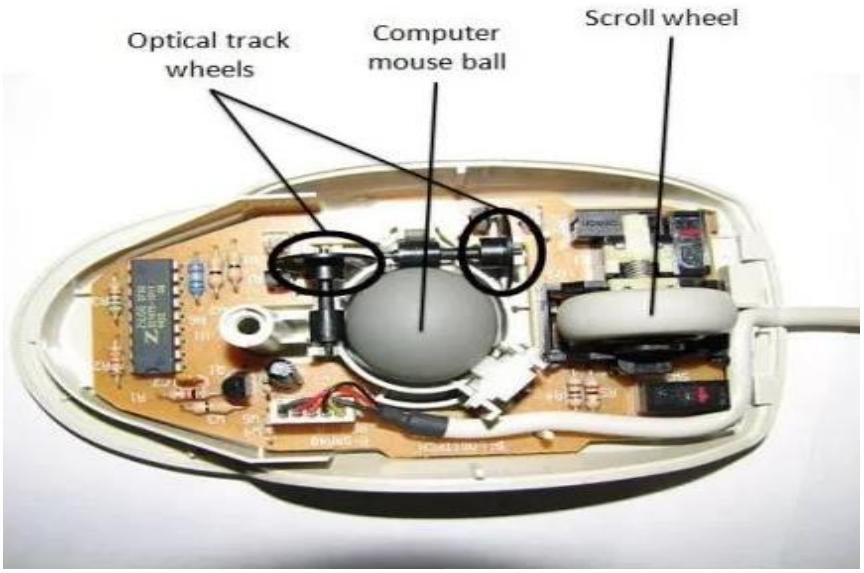


Figure 2.1.1 Mechanical mouse, with top cover removed

2.1.2 Optical And Laser Mouse

A mouse that commonly used in these days, the motions of optical mouse rely on the Light Emitting Diodes (LEDs) to detect movements relative to the underlying while the laser mouse is an optical mouse that uses coherent laser lights. Comparing to its predecessor, which is the mechanical mouse, the optical mouse no longer rely on the rollers to determine its movement, instead it uses an imaging array of photodiodes. The purpose of implementing this is to eliminate the limitations of degradation that plagues the current predecessor, giving it more durability while offers better resolution and precision. However, there's still some downside, even-though the optical mouse are functional on most opaque diffuse surface, it's unable to detect motions on the polished surface. Furthermore, long term usage without a proper cleaning or maintenance may leads to dust particles trap between the LEDs, which will cause both optical and laser mouse having surface detection difficulties. Other than that, it's still prone to degradation of the button switches, which again will cause the mouse to function improperly unless it was disassembled and repaired

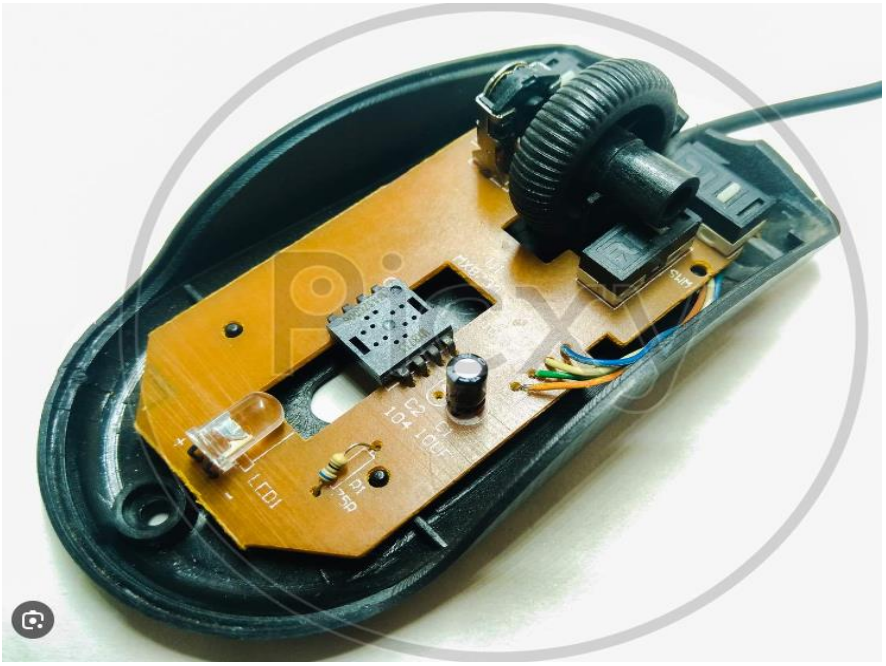


Figure 2.1.2 Optical Mouse, with top cover removed

2.2 Problem Statement

It's no surprised that every technological devices have its own limitations, especially when it comes to computer devices. After the review of various type of the physical mouse, the problems are identified and generalized. The following describes the general problem that the current physical mouse suffers:

- Physical mouse is subjected to mechanical wear and tear.
- Physical mouse is not easily adaptable to different environments and its performance varies depending on the environment.
- Mouse has limited functions even in present operational environments.
- All wired mouse and wireless mouse have its own lifespan.
- Physical mouse requires special hardware and surface to operate.

2.3 Motivation of Virtual Mouse

It is fair to say that the Virtual Mouse will soon to be substituting the traditional physical mouse in the near future, as people are aiming towards the lifestyle where that every technological devices can be controlled and interacted remotely without using any peripheral devices such as the remote, keyboards, etc. it doesn't just provides convenience, but it's cost effective as well.

2.3.1 Convenient

It is known in order to interact with the computer system, users are required to use an actual physical mouse, which also requires a certain area of surface to operate, not to mention that it suffers from cable length limitations. Virtual Mouse requires none of it, as it only a webcam to allow image capturing of user's hand position in order to determine the position of the pointers that the user want it to be.

2.3.2 Cost Effective

A quality physical mouse is normally cost from the range of 30 ringgit to a hefty 400 ringgit, depending on their functionality and features. Since the Virtual Mouse requires only a webcam, a physical mouse are no longer required, thus eliminating the need to purchase one, as a single webcam is sufficient enough to allow users to interact with the computer system through it, while some other portable computer system such as the laptop, are already supplied with a built-in webcam, could simply utilize the Virtual Mouse software without having any concerns about purchasing any external peripheral devices

Chapter 3
Literature Review

3.1 Introduction to literature

As modern technology of human computer interactions become important in our everyday lives, varieties of mouse with all kind of shapes and sizes were invented, from a casual office mouse to a hard-core gaming mouse. However, there are some limitations to these hardware as they are not as environmental friendly as it seems. For example, the physical mouse requires a flat surface to operate, not to mention that it requires a certain area to fully utilize the functions offered. Furthermore, some of these hardware are completely useless when it comes to interact with the computers remotely due to the cable lengths limitations, rendering it inaccessible

3.2 Visual Panel

To overcome the stated problems, Zhengyou et al. (2001), proposed an interface system named Visual Panel that utilize arbitrary quadrangle-shaped planar object as a panel to allow the user to use any tip-pointer tools to interact with the computer. The interaction movements will be captured, analysed and implement the positions of the tip-pointer, resulting accurate and robust interaction with the computer. The overall system consists of panel tracker, tip-pointer tracker, holography, calculation and update, and action detector and event generator as it can simulate both mouse and keyboard.

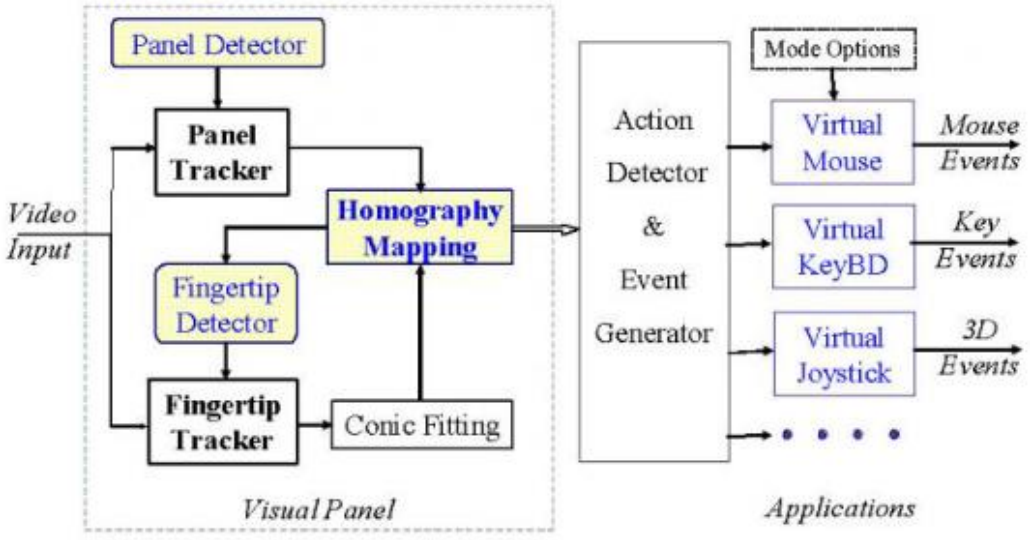


Figure 3.2: The system overview of Visual Panel (Zhengyou, Ying and Shafer, 2001)

3.3 Virtual Mouse Using a Webcam

Another colour detection method proposed by Kazim Sekeroglu (2010), the system requires three fingers with three colour pointers to simulate the click events. The proposed system are capable of detecting the pointers by referring the defined colour information, track the motion of the pointers, move the cursor according to the position of the pointer, and simulate the single and double left or/and right click event of the mouse

To detect the colours, they have utilized the MATLAB's built in "imsubtract" function, with the combination of the noise filtering by using median filter, which are effective in filtering out or at least reduce the "salt and pepper" noise. The captured image will be converted to Binary Scale Image by using MATLAB's built in "im2bw" function to differentiate the possible values for each pixel. When the conversion is done, the captured image will undergo another filtering process by using "bwareaopen" to remove the small areas in order to get an accurate number of the object detected in the image.

3.4 Portable Vision-Based Human Computer Interaction(HCI)

Another "Ubiquitous Computing" approach proposed by Chu-Feng Lien (2015), requires only finger-tips to control the mouse cursor and click events. The proposed system doesn't requires hand-gestures nor colour tracking in order to interact with the system, instead it utilize a feature name Motion History Images(MHI) , a method that used to identify movements with a row of images in time.

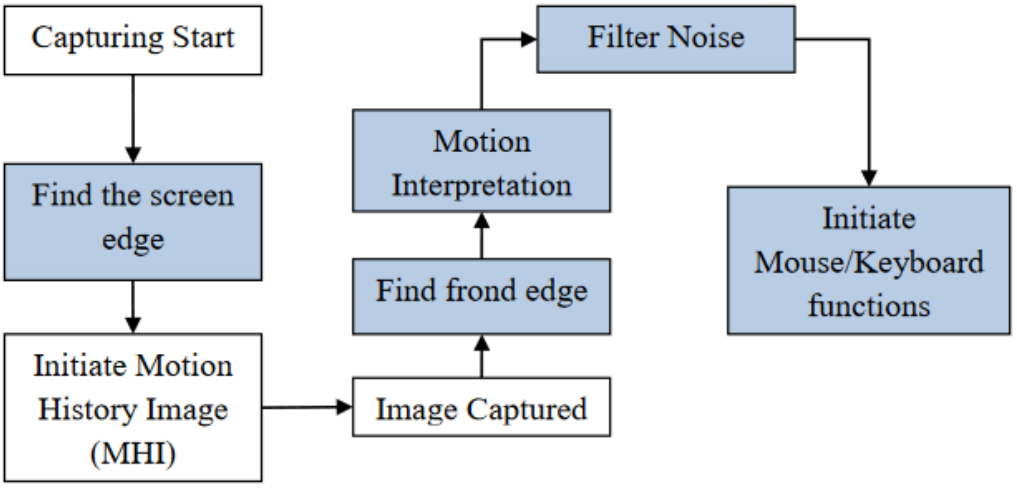


Figure 3.4: The Flow Chart of Portable Vision-Based Human Computer Interaction (Chu-Feng, 2008)

Even though the proposed system possess good accuracy in a well-controlled environment, it does has its own limitations. The proposed system are not capable to detect fast moving movements as the frame-rates are not able to keep up, thus leading to increase of error rate. Furthermore, due to the mouse click events occurred when the finger hold on a certain positions, this may lead to user constant finger movements to prevent false alarm, which may result inconvenience.

the system requires three fingers with three colour pointers to simulate the click events. The proposed system are capable of detecting the pointers by referring the defined colour information, track the motion of the pointers, move the cursor according to the position of the pointer, and simulate the single and double left or/and right click event of the mouse To detect the colours, they have utilized the MATLAB's built in "imsubtract" function, with the combination of the noise filtering by using median filter, which are effective in filtering out or at least reduce the "salt and pepper" noise. The captured image will be converted to Binary Scale Image by using MATLAB's built in "im2bw" function to differentiate the possible values for each pixel. When the conversion is done, the captured image will undergo another filtering process by using "bwareaopen" to remove the small areas in order to get an accurate number of the object detected in the image

Chapter 4
Requirements

4.1 Introduction to requirements

All computer softwere needs certain hardware component other sotfwere resources to be present on a computer to be used efficiently.These prerequisites are no as (computer) system requirements and are often used as a guidline as offsite to an absulate rule.

4.2 Hardware Requirement

The following describes the hardware needed in order to execute and develop the Virtual Mouse application:

- Computer Desktop or Laptop The computer desktop or a laptop will be utilized to run the visual software in order to display what webcam had captured. A notebook which is a small, lightweight and inexpensive laptop computer is proposed to increase mobility. System will be using
    1. Processor : Core2Duo
    2. Main Memory : 4GB RAM
    3. Hard Disk : 320GB
    4. Display : 14" Monitor
- Webcam

Webcam is utilized for image processing, the webcam will continuously taking image in order for the program to process the image and find pixel position

4.3 Software Requirement

The following describes the software needed in-order to develop the Virtual Mouse application:

- C++ Language

The coding technique on developing the Virtual Mouse application will be the C++ with the aid of the integrated development environment (IDE) that are used for developing computer programs, known as the Microsoft Visual Studio. A C++ library provides more than 35 operators, covering basic arithmetic, bit manipulation, indirection, comparisons, logical operations and others.

prioritized based on business value, and the development team works on the highest-priority user stories first.

- Open CV Library

OpenCV are also included in the making of this program. OpenCV (Open Source Computer Vision) is a library of programming functions for real time computer vision. OpenCV have the utility that can read image pixels value, it also have the ability to create real time eye tracking and blink detection. Software will be using:

OS : Window 7/10 Ultimate 64-bit

Language : C++

Tool Used : Open CV and Cmake

4.4 Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on.

4.5 Pycharm

PyCharm is a hybrid platform developed by JetBrains as an IDE for Python. It is commonly used for Python application development. Some of the unicorn organizations such as Twitter, Facebook, Amazon, and Pinterest use PyCharm as their Python IDE!

It supports two versions: v2.x and v3.x.

We can run PyCharm on Windows, Linux, or Mac OS. Additionally, it contains modules and packages that help programmers develop software using Python in less time and with minimal effort. Further, it can also be customized according to the requirements of developers. Nowadays, Python is in great demand. It is widely used in the software development idustry. There is ‘n’ number of reasons for this.

- **High-level object-oriented programming language:** Python includes effective symbolism.
- **Rapid application development:** Because of its concise code and literal syntax, the development of applications gets accelerated. The reason for its wide usability is its simple and easy-to-master syntax. The simplicity of the code helps reduce the time and cost of development.
- **Dynamic typescript:** Python has high-level incorporated data structures blended with dynamic typescript and powerful binding.

Chapter 5
Algorithm Used for Hand Tracking

5.1 MediaPipe

MediaPipe is a framework which is used for applying in a machine learning pipeline, and it is an opensource framework of Google. The MediaPipe framework is useful for cross platform development since the framework is built using the time series data. The MediaPipe framework is multimodal, where this framework can be applied to various audios and videos . The MediaPipe framework is used by the developer for building and analyzing the systems through graphs, and it also been used for developing the systems for the application purpose. The steps involved in the system that uses MediaPipe are carried out in the pipeline configuration. The pipeline created can run in various platforms allowing scalability in mobile and desktops. The MediaPipe framework is based on three fundamental parts; they are performance evaluation, framework for retrieving sensor data, and a collection of components which are called calculators , and they are reusable. A pipeline is a graph which consists of components called calculators, where each calculator is connected by streams in which the packets of data flow through. Developers are able to replace or define custom calculators anywhere in the graph creating their own application. The calculators and streams combined create a data-flow diagram; the graph (Figure 5.1.1) is created with MediaPipe where each node is a calculator and the nodes are connected by streams .

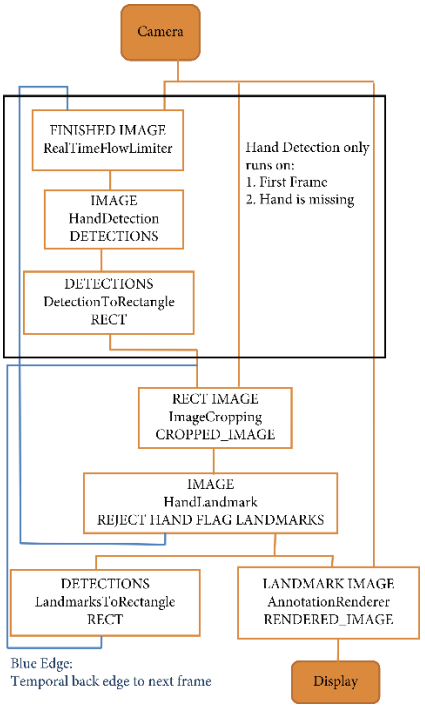
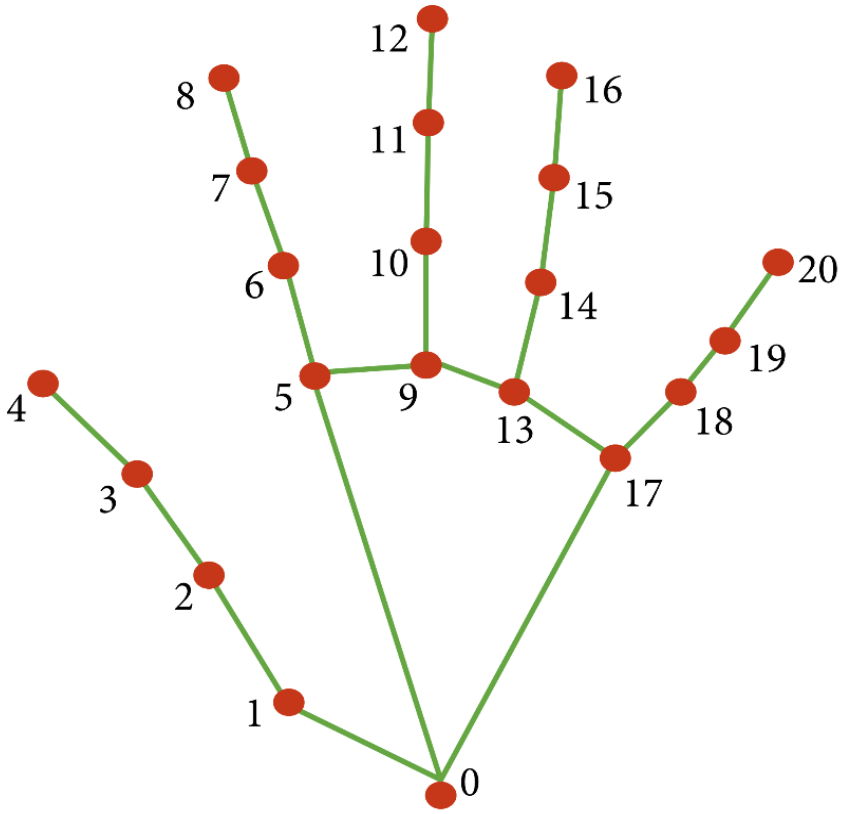


Figure 5.1.1: Mediapipe

Single-shot detector model is used for detecting and recognizing a hand or palm in real time. The single-shot detector model is used by the MediaPipe. First, in the hand detection module, it is first trained for a palm detection model because it is easier to train palms. Furthermore, the nonmaximum suppression works significantly better on small objects such as palms or fists . A model of hand landmark consists of locating 21 joint or knuckle co-ordinates in the hand region, as shown in Figure 5.1.2



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Figure 5.1.2

5.2 Gesture Recognition

The gesture recognition method used in the proposed design is a combination of two methods, the method proposed by Yeo and the method proposed by Balazs. The algorithm for the proposed gesture recognition method is described in the flow chart of Figure below. It can beseen from Figure above that the convexity defects for the hand contour must firstly be calculated. The convexity defects for the hand contour was calculated using the OpenCV inbuilt function

"cv Convexity

Defects"The parameters of the convexity defects (start point, end point and depth point) are stored in a sequence of arrays. After the convexity defects are obtained, there are twomain steps for gesture recognition:

i.Finger Tip Identificationi.

ii.Number Of Fingers

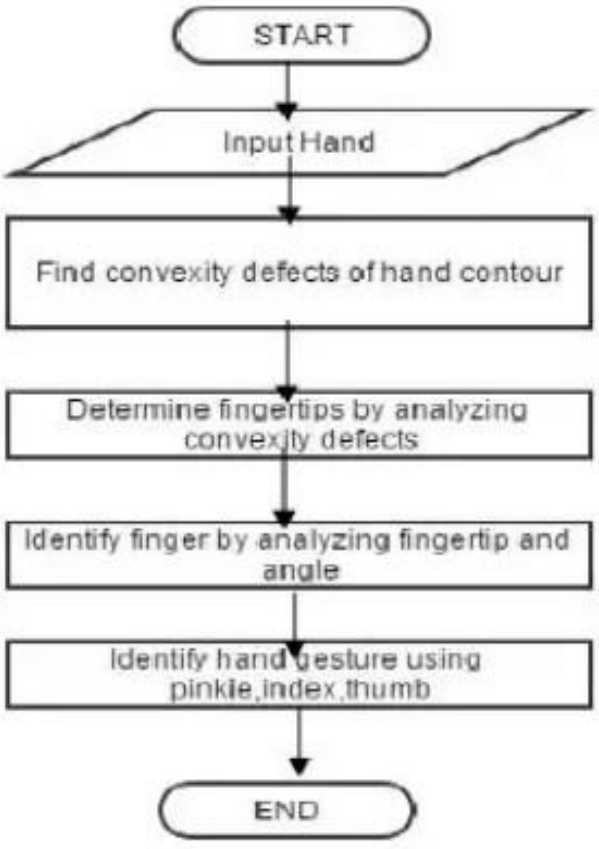
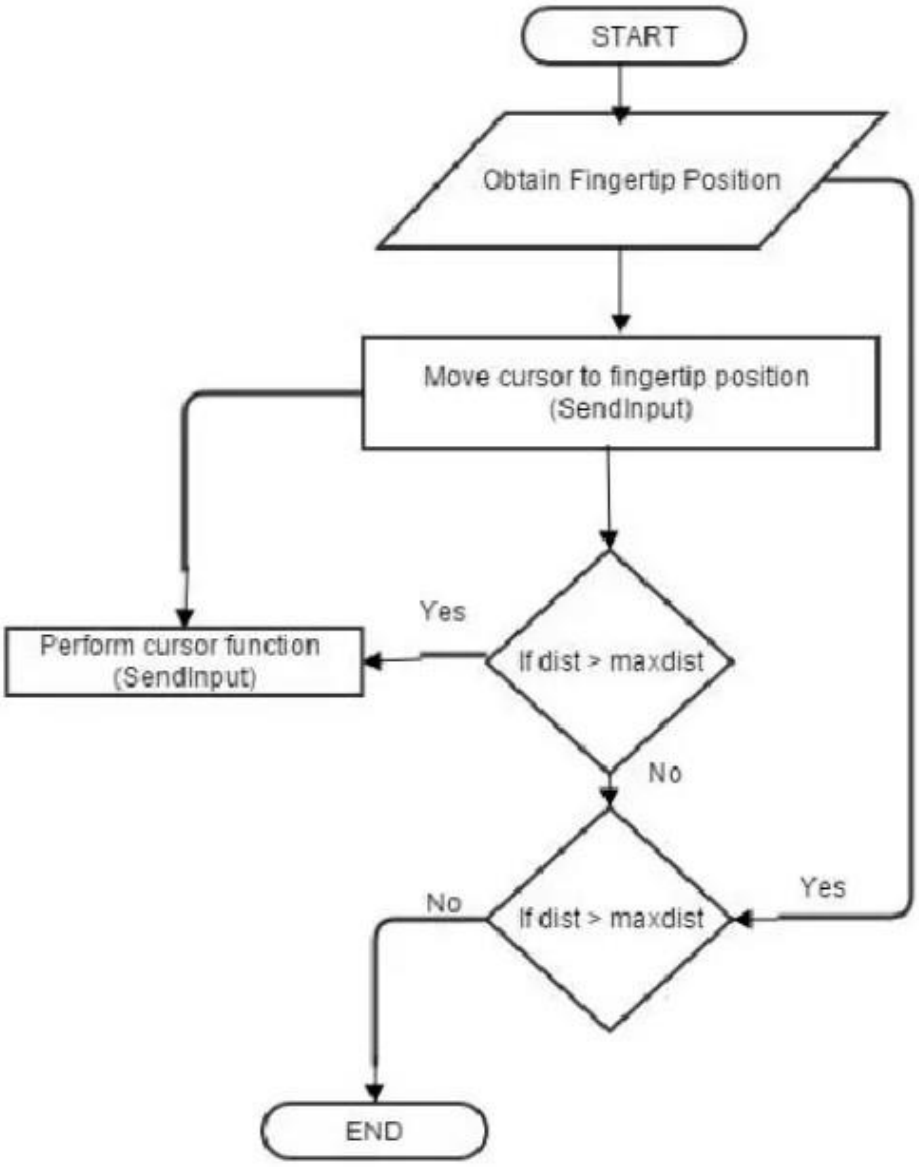


Figure 5.2: Gesture Recognition

5.3.Curso-Control

Once the hand gestures are recognized, it will be a simple matter of mapping different handgestures to specific mouse functions. It turns out that controlling the computer cursor, in theC/C++ programming language is relatively easy. By including the User.lib library into the program, the “SendInput” function will allow control of the computer cursor. Instructions on how to properly use this funcation,was obtained from the Microsoft Develops Network website.



Algorithm for Cursor Control

5.4 Flow Chart

The infinite loop is used so that the web camera captures the frames in every instance and is open In real-time hand gesture recognition, it is still challenging to determine when a gesture begins and when it ends from a hand trajectory. This session presents a new solution for hand gesture spotting to mark the start and end points of a gesture during the entire course of the program. We capture the live feed stream, frame by frame. Then we process each captured frame which is in RGB(default) color space to HSV color space. There are more than 150 color-space conversion methods available in OpenCV. But we will look into only two which are most widely used ones, BGR to Gray and BGR to HSV

.

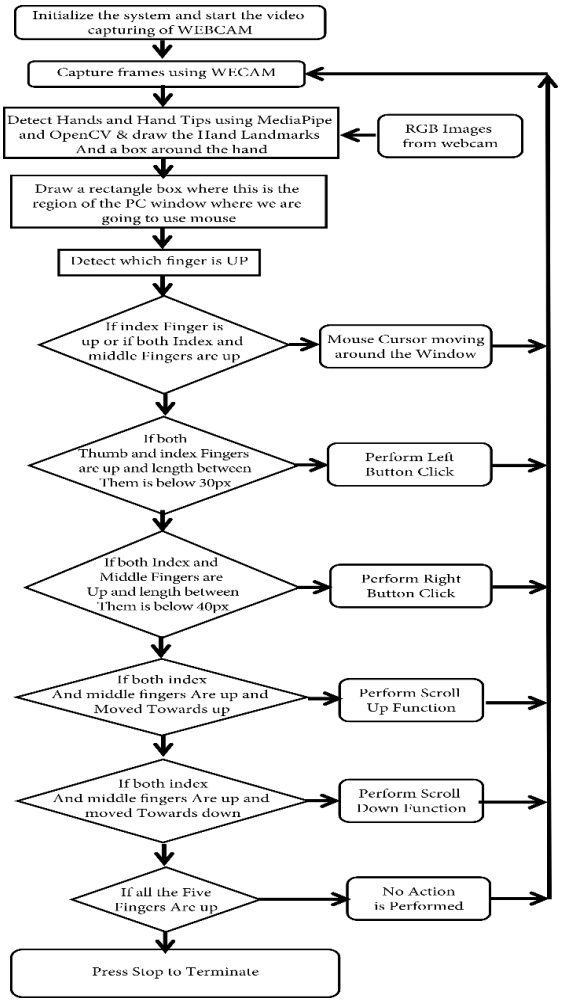
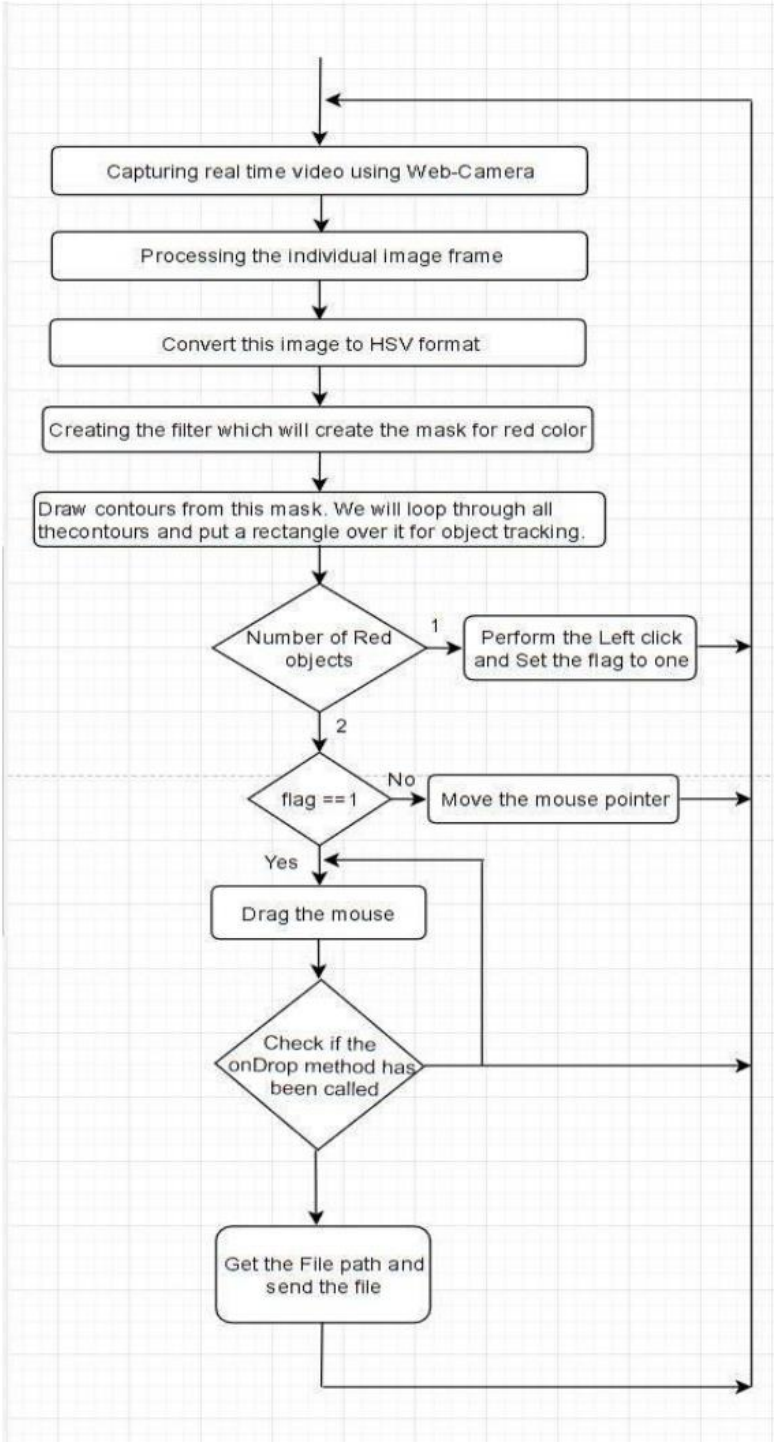


Figure 5.2.1: The Flow Chart

5.5 Capturing frames

The infinite loop is used so that the web camera captures the frames in every instance and is open during the entire course of the program. We capture the live feed stream, frame by frame. Then we process each captured frame which is in RGB(default) color space to HSV color space. There are more than 150 color-space conversion methods available in OpenCV. But we will look into only two which are most widely used ones, BGR to Gray and BGR to HSV.



5.6 Data Flow Diagram

We want to show through the application that a hand gesture-based system is available on behalf of a keyboard or mouse-based system. Thus, we develop practical application with hand gesture recognition as shown in Figure 5.3.1. When the user stands in front of the RGB-D camera, video frames are captured and the hand is detected. When the user interacts with a computer, a deep learning technique is used for hand gesture recognition internally in the system. Therefore, users can easily use the hand gestures to interact and control applications. In shows an example of how our hand gestures-based real-time HCI system controls the computer with

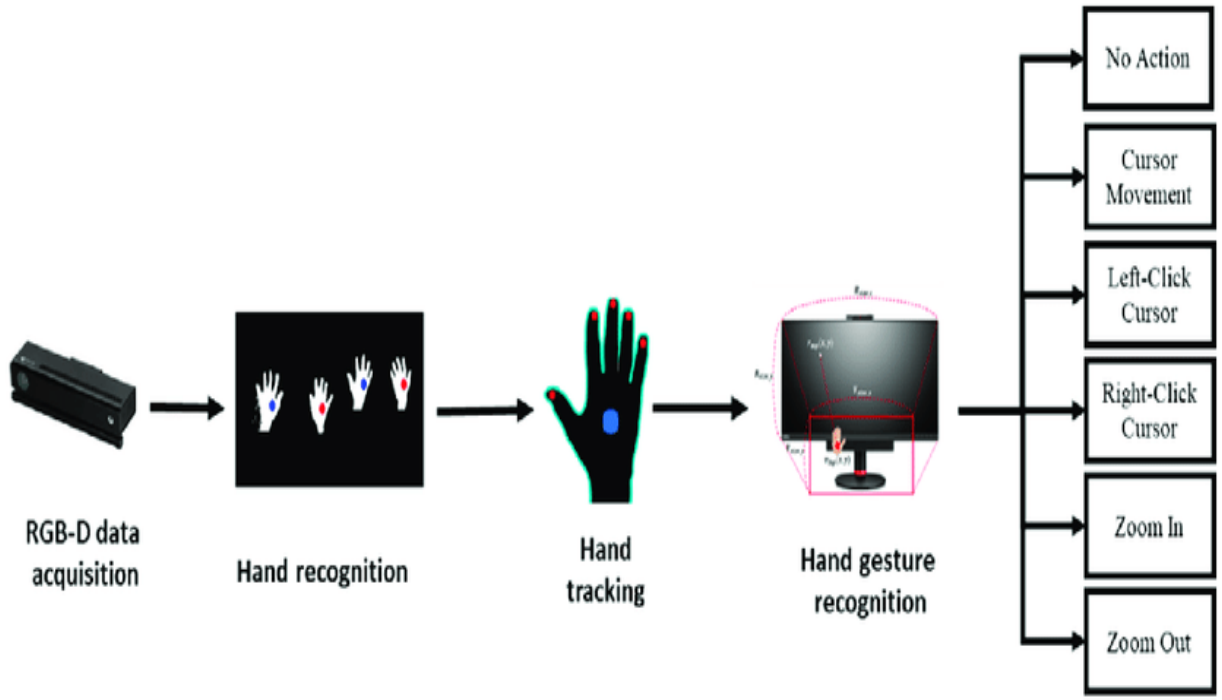


Figure 5.3.1 Data Flow Diagram

Chapter 6
Implementation

6.1 Camera Settings

The runtime operations are managed by the webcam of the connected laptop or desktop. To capture a video, we need to create a Video Capture object. Its argument can be either the device index or the name of a video file. Device index is just the number to specify which camera.Since we only use a single camera we pass it as '0'. We can add additional camera to the system and pass it as 1,2 and so on. After that, you can capture frame-by-frame. But at the end, don't forget to release the capture. We could also apply color detection techniques to any image by doing simple modifications in the code.

6.2 Display the frame

The imShow() is a function of HighGui and it is required to call the waitKey regulerly. The processing of the event loop of the imshow() function is done by calling waitKey. The function waitKey() waits for key event for a "delay" (here, 5 milliseconds). Windows events like redraw, resizing, input event etc. are processed by HighGui. So we call the waitKey function, even with a 1ms delay.

6.3. The Camera Used in the AI Virtual Mouse System

The proposed AI virtual mouse system is based on the frames that have been captured by the webcam in a laptop or PC. By using the Python computer vision library OpenCV, the video capture object is created and the web camera will start capturing video, as shown in Figure 6.3.1. The web camera captures and passes the frames to the AI virtual system.

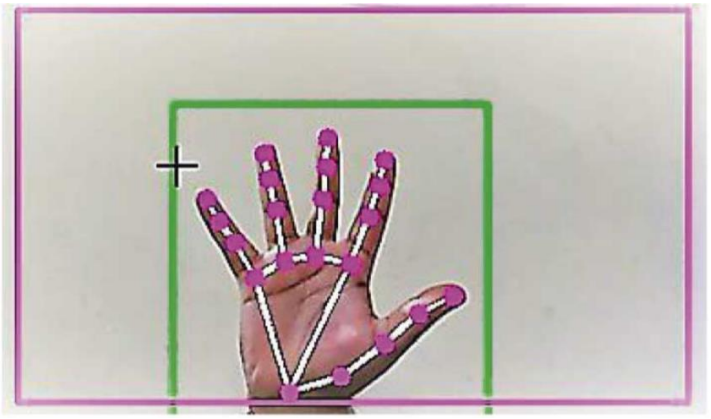


Figure 6.3.1

6.4. Capturing the Video and Processing

The AI virtual mouse system uses the webcam where each frame is captured till the termination of the program. The video frames are processed from BGR to RGB color space to find the hands in the video frame by frame as shown in the following code:

```
def findHands(self, img, draw = True):

imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

self.results = self.hands.process(imgRGB)
```

6.5 (Virtual Screen Matching) Rectangular Region for Moving through the Window

The AI virtual mouse system makes use of the transformational algorithm, and it converts the co-ordinates of fingertip from the webcam screen to the computer window full screen for controlling the mouse. When the hands are detected and when we find which finger is up for performing the specific mouse function, a rectangular box is drawn with respect to the computer window in the webcam region where we move throughout the window using the mouse cursor, as shown in Figure 6.5.1

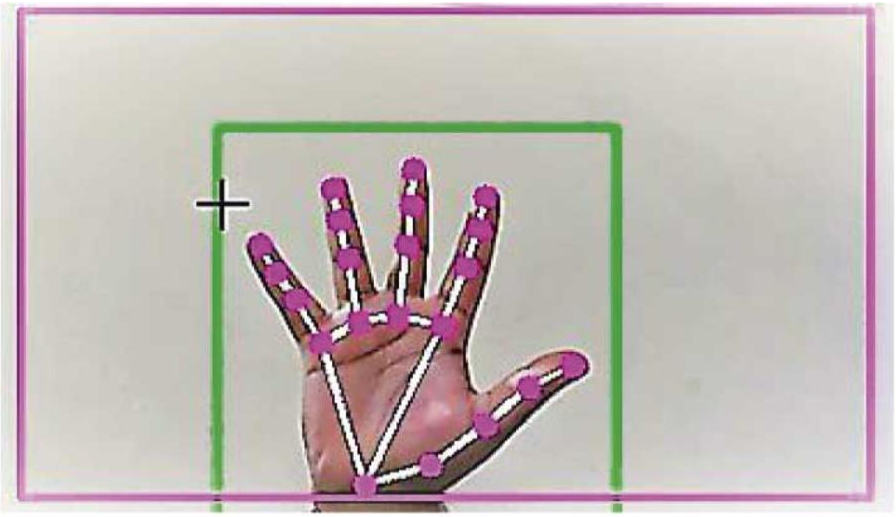


Figure 6.5.1

6.6 Detecting Which Finger Is Up and Performing the Particular Mouse Function

In this stage, we are detecting which finger is up using the tip Id of the respective finger that we found using the MediaPipe and the respective co-ordinates of the fingers that are up, as shown in Figure 6.6.1, and according to that, the particular mouse function is performed.

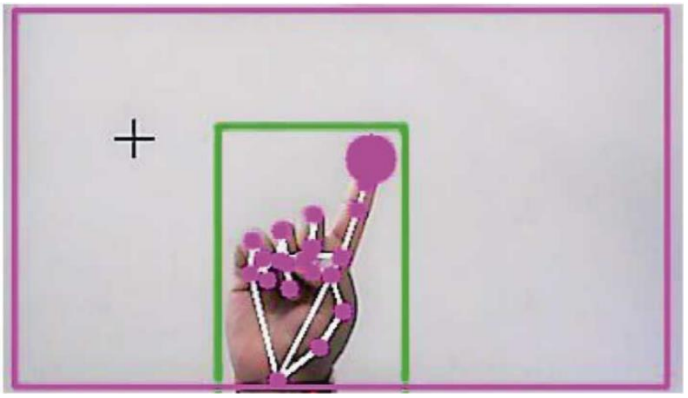


Figure 6.6.1

6.7. Mouse Functions Depending on the Hand Gestures and Hand Tip Detection Using Computer Vision

6.7.1. For the Mouse Cursor Moving around the Computer Window

If the index finger is up with tip Id = 1 or both the index finger with tip Id = 1 and the middle finger with tip Id = 2 are up, the mouse cursor made to move around window of the computer using the AutoPy package of Python, as shown in Figure 6.7.1

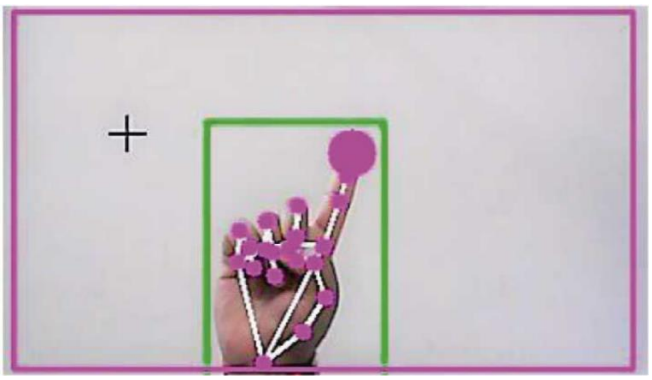


Figure 6.7.1

6.7.2. For the Mouse to Perform Left Button Click

If both the index finger with tip Id = 1 and the thumb finger with tip Id = 0 are up and the distance between the two fingers is lesser than 30px, the computer is made to perform the left mouse button click using the pynput Python package, as shown in Figures 6.7.2

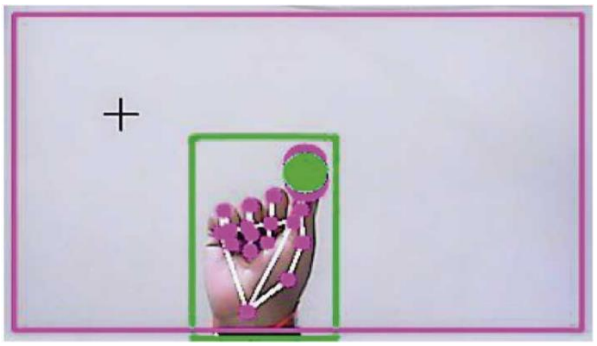


Figure 6.7.2

6.7.3. For the Mouse to Perform Right Button Click

If both the index finger with tip Id = 1 and the middle finger with tip Id = 2 are up and the distance between the two fingers is lesser than 40 px, the computer is made to perform the right mouse button click using the pynput Python package, as shown in Figure 6.7.3

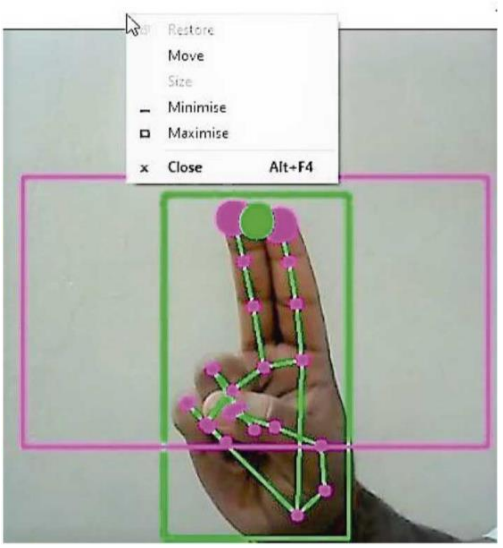


Figure 6.7.3

6.7.4. For the Mouse to Perform Scroll up Function

If both the index finger with tip Id = 1 and the middle finger with tip Id = 2 are up and the distance between the two fingers is greater than 40 px and if the two fingers are moved up the page, the computer is made to perform the scroll up mouse function using the PyAutoGUI Python package, as shown in Figure 6.7.4

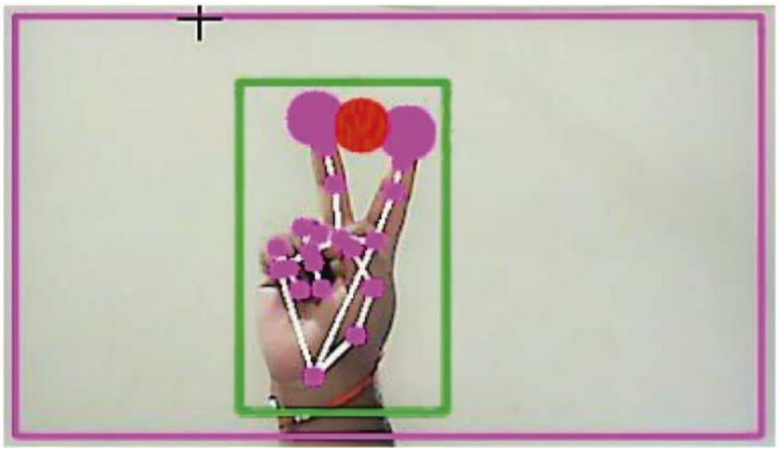


Figure 6.7.4

6.7.5. For the Mouse to Perform Scroll down Function

If both the index finger with tip Id = 1 and the middle finger with tip Id = 2 are up and the distance between the two fingers is greater than 40px and if the two fingers are moved down the page, the computer is made to perform the scroll down mouse function using the PyAutoGUI Python package, as shown in Figure 6.7.5

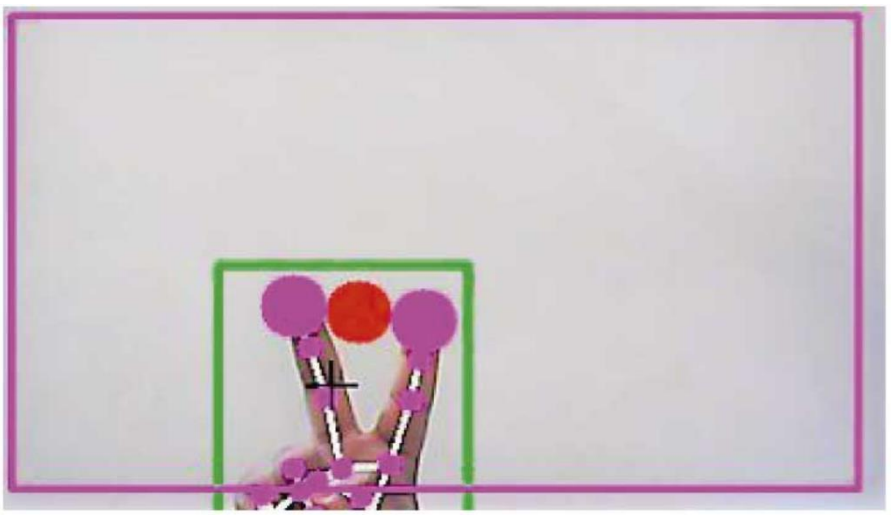


Figure 6.7.5

6.7.6. For No Action to be Performed on the Screen

If all the fingers are up with tip Id = 0, 1, 2, 3, and 4, the computer is made to not perform any mouse events in the screen, as shown in Figure 6.7.6

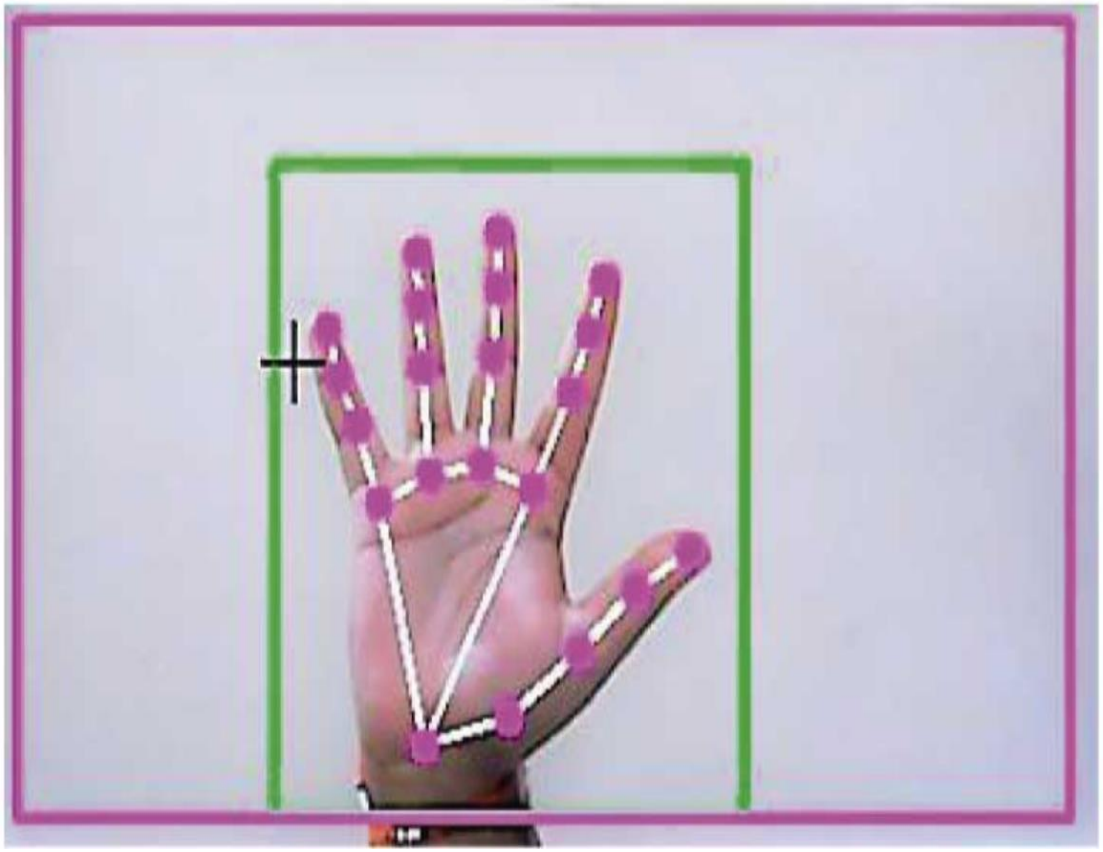


Figure 6.7.6

Chapter 7
Experimental Results and Evaluation

7.1 Result and Evaluation

In the proposed AI virtual mouse system, the concept of advancing the human-computer interaction using computer vision is given.

Cross comparison of the testing of the AI virtual mouse system is difficult because only limited numbers of datasets are available. The hand gestures and finger tip detection have been tested in various illumination conditions and also been tested with different distances from the webcam for tracking of the hand gesture and hand tip detection. An experimental test has been conducted to summarize the results shown in Table 1. The test was performed 25 times by 4 persons resulting in 600 gestures with manual labelling, and this test has been made in different light conditions and at different distances from the screen, and each person tested the AI virtual mouse system 10 times in normal light conditions, 5 times in faint light conditions, 5 times in close distance from the webcam, and 5 times in long distance from the webcam, and the experimental results are tabulated in Table

Table 1

Experimental results.

Hand tip gesture*	Mouse function performed	Success	Failure	Accuracy (%)
Tip ID 1 or both tip IDs 1 and 2 are up	Mouse movement	100	0	100
Tip IDs 0 and 1 are up and the distance between the fingers is <30	**Left button click**	99	1	99
Tip IDs 1 and 2 are up and the distance between the fingers is <40	**Right button click**	95	5	95
Tip IDs 1 and 2 are up and the distance between the fingers is >40 and both fingers are moved up the page	Scroll up function	100	0	100
Tip IDs 1 and 2 are up and the distancebetween the fingers is >40 and both fingers are moved down the page	Scroll down function	100	0	100
All five tip IDs 0, 1, 2, 3, and 4 are up	No action performed	100	0	100
Result		594	6	99

*Finger tip ID for respective fingers: tip Id 0: thumb finger; tip Id 1: index finger; tip Id 2: middle finger; tip Id 3: ring finger; tip Id 4: little finger.

From Table 1, it can be seen that the proposed AI virtual mouse system had achieved an accuracy of about 99%. From this 99% accuracy of the proposed AI virtual mouse system, we come to know that the system has performed well. As seen in Table 1, the accuracy is low for “Right Click” as this is the hardest gesture for the computer to

understand. The accuracy for right click is low because the gesture used for performing the particular mouse function is harder. Also, the accuracy is very good and high for all the other gestures. Compared to previous approaches for virtual mouse, our model worked very well with 99% accuracy. The graph of accuracy is shown in Figure 7.1

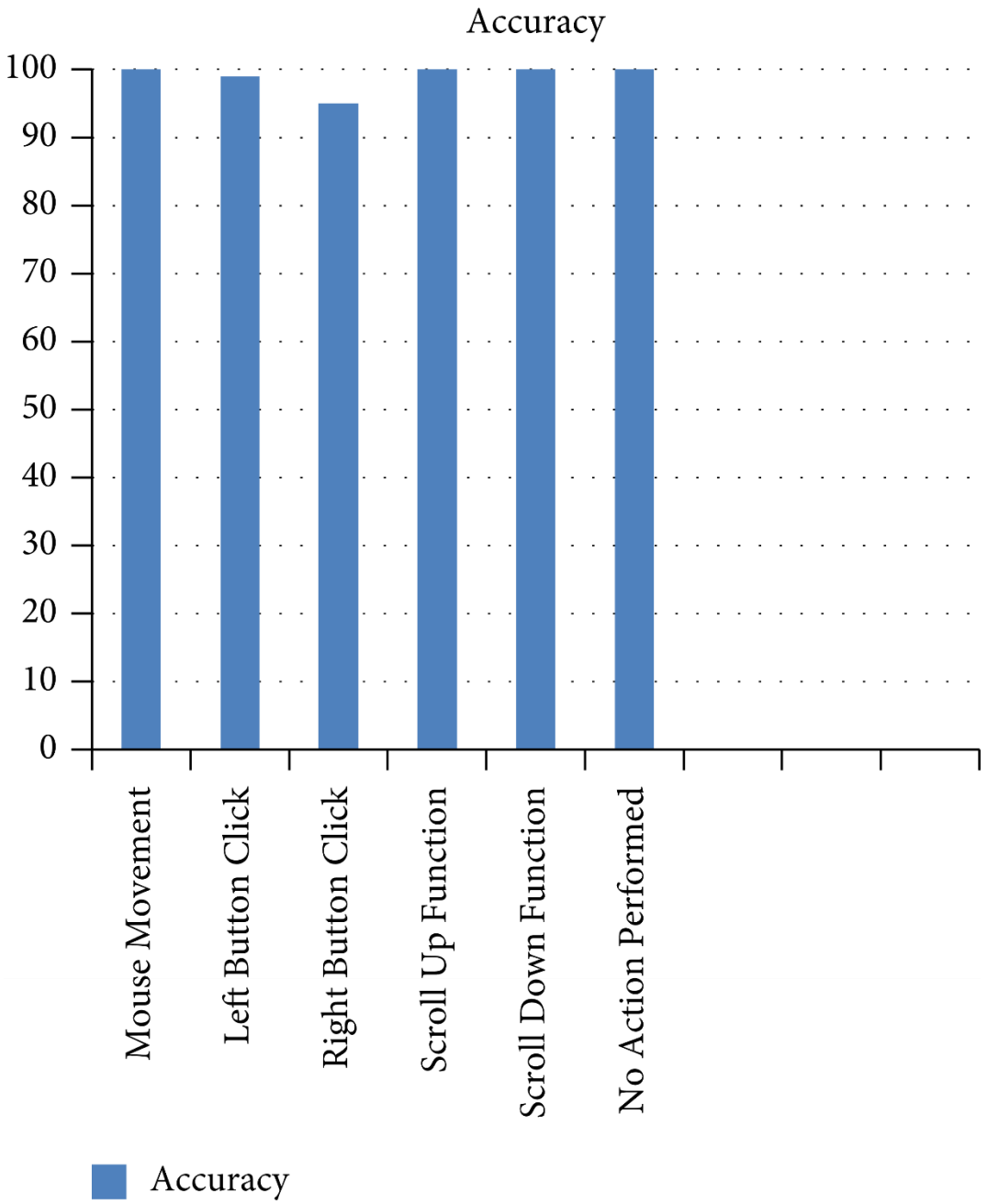


Figure 7.1

Chapter 8
Conclusions

8.1 Conclusion

Gesture recognition gives the best interaction between human and machine. Gesture recognition is also important for developing alternative human computer interaction modalities. It enables human to interface with machine in a more natural way. Gesture recognition can be used for many applications like sign language recognition for deaf and dumb people, robot control etc. This technology has wide applications in the fields of augmented reality, computer graphics, computer gaming, prosthetics, and biomedical instrumentation.

Digital Canvas is an extension of our system which is gaining popularity among artists, by which the artist could create 2D or 3D images using the Virtual Mouse technology using the hand as brush and a Virtual Reality kit or a monitor as display set. This technology can be used to help patients who don't have control of their limbs. In case of computer graphics and gaming this technology has been applied in modern gaming consoles to create interactive games where a person's motions are tracked and interpreted as commands.

The major extension to this work can be done to make system able to work at much complex background and compatible with different light conditions. It can be made as an effective user interface and which can include all mouse functionalities. And also, it would be ideal to research into advanced mathematical materials for image processing and investigate on different hardware solutions that would result in more accurate hand detections. Not only did this project show the different gesture operations that could be done by the users but it also demonstrated the potential in simplifying user interactions with personal computers and hardware systems. In conclusion, it's no surprised that the physical mouse will be replaced by a virtual non-physical mouse in the Human-Computer Interactions (HCI), where every mouse movements can be executed with a swift of your fingers everywhere and anytime without any environmental restrictions. This project had develop a colour recognition program with the purpose of replacing the generic physical mouse without sacrificing the accuracy and efficiency, it is able to recognize colour movements, combinations, and translate them into actual mouse functions. Due to accuracy and efficiency plays an important role in making the program as useful as an actual physical mouse, a few techniques had to be implemented. First and foremost, the coordinates of the colours that are in charge of handling the

cursor movements are averaged based on a collections of coordinates, the purpose of this technique is to reduce and stabilize the sensitivity of cursor movements, as slight movement might lead to unwanted cursor movements. Other than that, several colour combinations were implemented with the addition of distance calculations between two colours within the combination, as different distance triggers different mouse functions. The purpose of this implementation is to promote convenience in controlling the program without much of a hassle. Therefore, actual mouse functions can be triggered accurately with minimum trial and errors. Furthermore, to promote efficient and flexible tracking of colours, calibrations phase was implemented, this allows the users to choose their choices of colours on different mouse functions, as long the selected colours doesn't fall within the same/similar RGB values (e.g. blue and sky-blue). Other than that, adaptive calibrations were also implemented as well, it is basically allows the program to save different set of HSV values from different angles where it will be used during the recognition phase. In Overall, the modern technologies have come a long way in making the society life better in terms of productivity and lifestyle, not the other way around. Therefore, societies must not mingle on the past technologies while reluctant on IA(HONS) Information System Engineering Faculty of Information and Communication Technology (Perak Campus), UTAR 40 accepting changes of the newer one. Instead, it’s advisable that they should embrace changes to have a more efficient, and productive lifestyle.

8.2 Limitation

In this project, there are several existing problems that may hinder the results of colour recognitions. One of the problems is the environmental factor during the recognition phase takes place. The recognition process are highly sensitive on the intensity of brightness, as immense brightness or darkness may cause the targeted colours to be undetected within the captured frames. Besides that, distance is also the one of the problem that may affect the colour recognition results, as the current detection region can support up to 25cm radius, any display of colours exceed the mentioned distance will be considered as a noise and be filtered off. Furthermore, the performance of the program are highly dependent on the users' hardware, as processor speed and/or resolutions taken from the webcam could have an effect on performance load.

Therefore, the slower the processing speed and/or the higher the resolutions, the longer time are required to process a single frame.

8.3 Future Works

There are several features and improvements needed in order for the program to be more user friendly, accurate, and flexible in various environments. The following describes the improvements and the features required:

a) Smart Recognition Algorithm Due to the current recognition process are limited within 25cm radius, an adaptive zoom-in/out functions are required to improve the covered distance, where it can automatically adjust the focus rate based on the distance between the users and the webcam.

b) Better Performance The response time are heavily rely on the hardware of the machine, this includes the processing speed of the processor, the size of the available RAM, and the available features of webcam. Therefore, the program may have better performance when it's running on a decent machines with a webcam that performs better in different types of lightings.

The above-mentioned points are the enhancements that can be done to increase the applicability and usage of this project. We have left all the options open so that if there is any other future requirement in the system by the user or students for the enhancement of the application then it is possible to implement them. In the last, we would like to thank all the persons involved in the development of the application directly or indirectly. We hope that the project will serve the purpose for which it is developed thereby underlining the success of the process.

REFERENCES

1. J. Katona, “A review of human–computer interaction and virtual reality research fields in cognitive InfoCommunications,” *Applied Sciences*, vol. 11, no. 6, p. 2646, 2021.

2. D. L. Quam, “Gesture recognition with a DataGlove,” *IEEE Conference on Aerospace and Electronics*, vol. 2, pp. 755–760, 1990.

3. D.-H. Liou, D. Lee, and C.-C. Hsieh, “A real time hand gesture recognition system using motion history image,” in *Proceedings of the 2010 2nd International Conference on Signal Processing Systems*, IEEE, Dalian, China, July 2010.

4. S. U. Dudhane, “Cursor control system using hand gesture recognition,” *IJARCCE*, vol. 2, no. 5, 2013.

5. K. P. Vinay, “Cursor control using hand gestures,” *International Journal of Critical Accounting*, vol. 0975–8887, 2016.

6. L. Thomas, “Virtual mouse using hand gesture,” *International Research Journal of Engineering and Technology (IRJET*, vol. 5, no. 4, 2018.

7. P. Nandhini, J. Jaya, and J. George, “Computer vision system for food quality evaluation—a review,” in *Proceedings of the 2013 International Conference on Current Trends in Engineering and Technology (ICCTET)*, pp. 85–87, Coimbatore, India, July 2013.

8. J. Jaya and K. Thanushkodi, “Implementation of certain system for medical image diagnosis,” *European Journal of Scientific Research*, vol. 53, no. 4, pp. 561–567, 2011.

9. P. Nandhini and J. Jaya, “Image segmentation for food quality evaluation using computer vision system,” *International Journal of Engineering Research and Applications*, vol. 4, no. 2, pp. 1–3, 2014.

Appendix

```
import cv2
import mediapipe as mp
import time
import math
import numpy as np


class handDetector():
    def __init__(self, mode=False, maxHands=2, detectionCon=False, trackCon=0.5):
        self.mode = mode
        self.maxHands = maxHands
        self.detectionCon = detectionCon
        self.trackCon = trackCon

        self.mpHands = mp.solutions.hands
        self.hands = self.mpHands.Hands(self.mode, self.maxHands,
                                        self.detectionCon, self.trackCon)
        self.mpDraw = mp.solutions.drawing_utils
        self.tipIds = [4, 8, 12, 16, 20]

    def findHands(self, img, draw=True):    # Finds all hands in a frame
        imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        self.results = self.hands.process(imgRGB)

        if self.results.multi_hand_landmarks:
            for handLms in self.results.multi_hand_landmarks:
                if draw:
                    self.mpDraw.draw_landmarks(img, handLms,
                                               self.mpHands.HAND_CONNECTIONS)

        return img

    def findPosition(self, img, handNo=0, draw=True):   # Fetches the position of
hands
        xList = []
        yList = []
        bbox = []
        self.lmList = []
        if self.results.multi_hand_landmarks:
            myHand = self.results.multi_hand_landmarks[handNo]
            for id, lm in enumerate(myHand.landmark):
                h, w, c = img.shape
                cx, cy = int(lm.x * w), int(lm.y * h)
                xList.append(cx)
                yList.append(cy)
                self.lmList.append([id, cx, cy])
                if draw:
                    cv2.circle(img, (cx, cy), 7, (255, 0, 0), cv2.FILLED)
```

```
            xmin, xmax = min(xList), max(xList)
            ymin, ymax = min(yList), max(yList)
            bbox = xmin, ymin, xmax, ymax

            if draw:
                cv2.rectangle(img, (xmin - 20, ymin - 20), (xmax + 20, ymax + 20),
                              (0, 255, 0), 2)

        return self.lmList, bbox

    def fingersUp(self):    # Checks which fingers are up
        fingers = []
        # Thumb
        if self.lmList[self.tipIds[0]][1] > self.lmList[self.tipIds[0] - 1][1]:
            fingers.append(1)
        else:
            fingers.append(0)

        # Fingers
        for id in range(1, 5):

            if self.lmList[self.tipIds[id]][2] < self.lmList[self.tipIds[id] - 2][2]:
                fingers.append(1)
            else:
                fingers.append(0)

        # totalFingers = fingers.count(1)

        return fingers

    def findDistance(self, p1, p2, img, draw=True,r=15, t=3):   # Finds distance
between two fingers
        x1, y1 = self.lmList[p1][1:]
        x2, y2 = self.lmList[p2][1:]
        cx, cy = (x1 + x2) // 2, (y1 + y2) // 2

        if draw:
            cv2.line(img, (x1, y1), (x2, y2), (255, 0, 255), t)
            cv2.circle(img, (x1, y1), r, (255, 0, 255), cv2.FILLED)
            cv2.circle(img, (x2, y2), r, (255, 0, 255), cv2.FILLED)
            cv2.circle(img, (cx, cy), r, (0, 0, 255), cv2.FILLED)
        length = math.hypot(x2 - x1, y2 - y1)

        return length, img, [x1, y1, x2, y2, cx, cy]


def main():
    pTime = 0
    cTime = 0
```

```
    cap = cv2.VideoCapture(0)
    detector = handDetector()
    while True:
        success, img = cap.read()
        img = detector.findHands(img)
        lmList, bbox = detector.findPosition(img)
        if len(lmList) != 0:
            print(lmList[4])

        cTime = time.time()
        fps = 1 / (cTime - pTime)
        pTime = cTime

        cv2.putText(img, str(int(fps)), (10, 70), cv2.FONT_HERSHEY_PLAIN, 3,
                    (255, 0, 255), 3)

        cv2.imshow("Image", img)
        cv2.waitKey(1)


if __name__ == "__main__":
    main()
```