



“VIRTUAL MOUSE”

A MINI PROJECT REPORT

NANDHINI C 19CECS021

NARAYANAN M. S 19CECS022

POZHIL.K 19CECS024

PRAGADEESHWAR.R 19CECS025

Submitted by

**in partial fulfilment for the award of the degree of
BACHELOR OF ENGINEERING**

in

COMPUTER SCIENCE AND ENGINEERING

SNS COLLEGE OF ENGINEERING, COIMBATORE-107

ANNA UNIVERSITY: CHENNAI 600025

October - 2021

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report “**VIRTUAL MOUSE**” is the bonafide work of “**NANDHINI.C (19CECS021), NARAYANAN M.S (19CECS022), POZHIL.K (19CECS024), PRAGADEESHWAR.R (19CECS025)**” who carried out the project work under my supervision.

SIGNATURE

Dr. K. Periyakaruppan

HEAD OF THE DEPARTMENT

Professor and Head

Computer Science and Engineering

SNS College of Engineering

Coimbatore – 641 107

SIGNATURE

Mr. K. Karthikeyan

SUPERVISOR

Assistant Professor

Computer Science and Engineering

SNS College of Engineering

Coimbatore – 641 107

Submitted for the Project Viva Voce Examination held on _____.

Internal examiner

External examiner

TABLE OF CONTENTS

CHAPTER NO:	TITLE	PAGE NO:
1.	ABSTRACT	1
2.	INTRODUCTION	2
3.	AIM AND OBJECTIVE	3
4.	PROJECT FEATURE	3
5.	AUTOMATION	4
6.	BLOCK DIAGRAM	5
7.	DATA FORMATS	5
8.	EXTENDED MODE	6
9.	EXISITING SYSTEM	6
10.	PHYSICAL MOUSE	7
11.	LIGHT PEN	7
12.	OPTICAL MOUSE	8
13.	HOLOGRAM	9
14.	PROJECT SCOPE	10
15.	PROJECT OBJECTIVE	10
16.	IMPACT AND CONTRIBUTION	11
17.	WORKING PRINCIPLE	12
18.	SYSTEM APPROACH	12
19.	TRACKING OF MOUSE POINTER	16
20.	PROBLEMS AND DRAWBACK	17
21.	EXPERIMENT AND RESULT	17
22.	RECOGNISATION PHASE	18
23.	HARDWARE REQUIREMENTS	21
24.	SOFTWARE REQUIREMENTS	22
25.	VERIFICATION PLAN	23
26.	IMPLEMENTATION AND TESTING	25
27.	SOURCE CODE AND HAND TRACKING	27
28.	VIRTUAL MOUSE SOURCE CODE	29
29.	VOLUME GESTURE SOURCE CODE	32
30.	LIMITATIONS	34
31.	FUTURE WORK	35
32.	CONCLUSION	35
33.	REFERENCES	36

ABSTRACT

In this modern world technologies are improving day by day people are moving to the advanced technologies many people are struggling with the physical mouse by the limitation in surfaces

The project provides a new approach for controlling mouse movement using a real time web camera. Major approaches consist of adding more buttons are changing the position of the tracking ball of the mouse. Instead, we decide to changer the design of the hardware. Our concept is to use a web camera and computer vision technology, As image segmentation and hand tracking recognition to control mouse task. In virtual mouse can provide a high performance than the physical mouse. The goal is to manage computers and their devices with gestures rather than pointing and clicking or mouse or touching a display directly. Increasingly we are recognizing the importance of human computing interacting (HCL), and in particular vision-based gesture and object recognition

This project proposes a way to control the position of the cursor with the bare hands without using any electronic device. While the operations like clicking and dragging of objects will be performed with different hand gestures. The proposed system will only require a webcam as an input device. The software's that will be required to implement the proposed system are OpenCV and python. The output of the camera will be displayed on the system's screen so that it can be further calibrated by the user. The python dependencies that will be used for implementing this system are AutoPy, math and mouse.

Packages used- OpenCV, Mediapipe.

CHAPTER - 1

INTRODUCTION:

A mouse, in computing terms is a pointing device that detects two-dimensional movements relative to a surface. This movement is converted into the movement of a pointer on a display that allows to control the Graphical User Interface (GUI) on a computer platform. There are a lot of different types of mouse that have already existed in the modern day's technology, there's the mechanical mouse that determines the movements by a hard rubber ball that rolls around as the mouse is moved. Years later, the optical mouse was introduced that replace the hard rubber ball to a LED sensor to detects table top movement and then sends off the information to the computer for processing. On the year 2004, the laser mouse was then introduced to improve the accuracy movement with the slightest hand movement, it overcome the limitations of the optical mouse which is the difficulties to track high-gloss surfaces. However, no matter how accurate can it be, there are still limitations exist within the mouse itself in both physical and technical terms. For example, a computer mouse is a consumable hardware device as it requires replacement in the long run, either the mouse buttons were degraded that causes inappropriate clicks, or the whole mouse was no longer detected by the computer itself. It has been generations since we have been using hand gestures for communicating in human society. The shaking of hands, thumbs up and Thumbs down signs have been ever existing in the environment. It is believed that gestures are the easiest way of interaction with anyone. So then why not apply it to the machines that we are using. In this work, we are demonstrating, real- gesture. The initial setup includes a low-cost USB web camera that can be used for providing the input to the system. The complete process is divided into 4 steps which are frame-capturing, image-processing, region extraction, feature-matching. To the extreme, it can also be called as hardware because it uses a camera for tracking hands. Despite the limitations, the computer technology still continues to grow, so does the importance of the human computer interactions. Ever since the introduction of a mobile device that can be interact with touch screen technology, the world is starting to demand the same technology to be applied on every technological device, this includes the desktop system. However, even though the touch screen technology for the desktop system is already exist, the price can be very steep. Therefore, a virtual human computer interaction device that replaces

the physical mouse or keyboard by using a webcam or any other image capturing devices can be an alternative way for the touch screen. This device which is the webcam will be constantly utilized by a software that monitors the gestures given by the user in order to process it and translate to motion of a pointer, as similar to a physical mouse

Aim and objective of research work include:

- For most laptop touchpad is not the most comfortable and convenient.
- Main objective pre-processing is to represent the data in such a way that it can be easily interpreted and processed by the system.
- Reduce cost of hardware. It focuses on extracting the features over the human hands and then matching their features to recognize the movement of the hand.

Project essential feature-

- User friendly.
- Portable.
- Handle simple operation left- click dragging, minimizing.
- No hardware.

A virtual mouse is a concept device similar to a virtual keyboard, allowing usage of a computer with a minimal amount of interactive physical peripherals. For the various methods of implementing a virtual mouse design, this particular project focuses on moving a computer mouse cursor by implementing an accelerometer-based wireless glove. This glove, which is powered by a standard battery pack, will be attached with an accelerometer and microcontroller that has the capability to wireless send data to a nearby wireless receiver that is interfaced directly with a computer PC. Software will then be incorporated to collect and analyse the user movement data for precise cursor readjustment.

An example would be a customer who has limited physical mobility in a part of their arm and wishes to minimize the work effort. An application example would be changing the

interaction experience between a user and CAD software modelling, where the user can obtain a virtual experience of positioning an object without the use of a peripheral mouse. The intent of this project serves for various applications or customers with specific requirements, making this an attractive and potentially marketable product in a wide variety of markets (such as TV, mobile, gaming, health care, and so forth) in the near future when virtual experience becomes more prevalent.

AUTOMATION:

Automation is a term for technology applications where human input is minimized. This includes business process automation (BPA), IT automation, personal applications such as home automation and more.

Types of automation:

Basic automation:

Basic automation takes simple, rudimentary tasks and automates them. This level of automation is about digitizing work by using tools to streamline and centralize routine tasks, such as using a shared messaging system instead of having information in disconnected silos. Business process management (BPM) and robotic process automation (RPA) are types of basic automation.

Process automation:

Process automation manages business processes for uniformity and transparency. It is typically handled by dedicated software and business apps. Using process automation can increase productivity and efficiency within your business. It can also deliver new insights into business challenges and suggest solutions. Process mining and workflow automation are types of process automation.

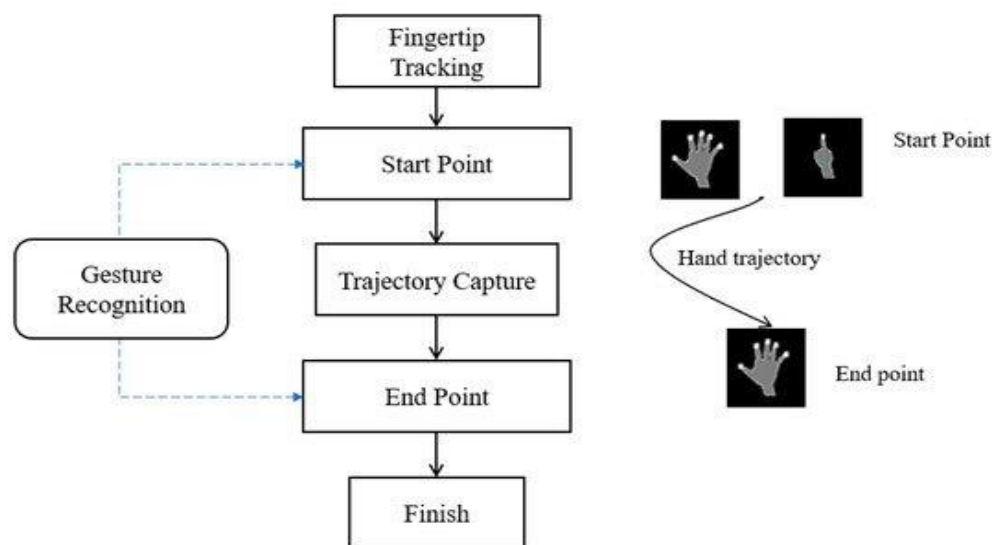
Integration automation:

Integration automation is where machines can mimic human tasks and repeat the actions once humans define the machine rules. One example is the “digital worker.” In recent years, people have defined digital workers as software robots that are trained to work with humans to perform specific tasks. They have a specific set of skills, and they can be “hired” to work on teams.

Artificial intelligence (AI) automation:

The most complex level of automation is artificial intelligence (AI) automation. The addition of AI means that machines can “learn” and make decisions based on past situations they have encountered and analysed. For example, in customer service, virtual assistants powered can reduce costs while empowering both customers and human agents, creating an optimal customer service experience.

BLOCK DIAGRAM FOR VIRTUAL MOUSE:



DATA FORMATS:

The IR Camera can return different sets of data describing the objects it is tracking. When the IR camera identifies an object, it assigns it to the first available object slot. If an object moves out of view, its slot is marked as empty (returns 0xFF data), but other objects retain their slots. For example, if the camera is tracking two objects and the first moves out of view, the data returned will be [empty, second object, empty, empty]. With more than four

objects visible, the camera is prone to rapidly switching between some of them. This could allow perception of more than four objects, at a reduced response speed and reliability. IR Camera works in different operating modes which are as follows: [2] 1) Basic Mode 2) Extended Mode 3) Full Mode Each mode has its own data format which is explained below The data format MUST match the number of bytes available in the Reporting Mode selected. Even choosing a mode with space for more bytes than necessary will not work, it has to be an exact match. In this project we have chosen extended mode.

EXTENDED MODE:

In Extended Mode, the IR Camera returns the same data as it does in Basic Mode, plus a rough size value for each object. The data is returned as 12 bytes, three bytes per object. Size has a range of 0-15. The data format for each object is shown in

Byte	Bit							
	7	6	5	4	3	2	1	0
0	X<7:0>							
1	Y<7:0>							
2	Y<9:8>		X<9:8>		S<3:0>			

EXISTING SYSTEM:

- Physical Mouse
- Light Pen
- Optical Mouse
- Hologram

PHYSICAL MOUSE:

A **computer mouse** plural **mice**, rarely **mouse** is a hand-held pointing device that detects two-dimensional motion relative to a surface. This motion is typically translated into the motion of a pointer on a display, which allows a smooth control of the graphical user interface of a computer.

The first public demonstration of a mouse controlling a computer system was in 1968. Mice originally used two separate wheels to track movement across a surface; one in the X-dimension, and one in the Y. Later, the standard design shifted to utilise a ball rolling on a surface to detect motion. Most modern mice use optical sensors that have no moving parts. Though originally all mice were connected to a computer by a cable, some modern mice are cordless, relying on short-range radio communication with the connected system.

In addition to moving a cursor, computer mice have one or more buttons to allow operations such as selection of a menu item on a display. Mice often also feature other elements, such as touch surfaces and scroll wheels, which enable additional control and dimensional input.

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. .
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.

LIGHT PEN:

A **light pen** is a computer input device in the form of a light-sensitive wand used in conjunction with a computer's cathode-ray tube (CRT) display.

It allows the user to point to displayed objects or draw on the screen in a similar way to a touchscreen but with greater positional accuracy. A light pen can work with any CRT-based display, but its ability to be used with LCDs was unclear (though Toshiba and Hitachi displayed a similar idea at the "Display 2006" show in Japan).

A light pen detects changes in brightness of nearby screen pixels when scanned by cathode-ray tube electron beam and communicates the timing of this event to the computer. Since a CRT scans the entire screen one pixel at a time, the computer can keep track of the expected time of scanning various locations on screen by the beam and infer the pen's position from the latest timestamp.

OPTICAL MOUSE:

An optical mouse is an advanced computer pointing device that uses a light-emitting diode (LED), an optical sensor, and digital signal processing (DSP) in place of the traditional mouse ball and electromechanical transducer. Movement is detected by sensing changes in reflected light, rather than by interpreting the motion of a rolling sphere.

The optical mouse takes microscopic snapshots of the working surface at a rate of more than 1,000 images per second. If the mouse is moved, the image changes. The tiniest irregularities in the surface can produce images good enough for the sensor and DSP to generate usable movement data. The best surfaces reflect but scatter light; an example is a blank sheet of white drawing paper. Some surfaces do not allow the sensor and DSP to function properly because the irregularities are too small to be detected. An example of a poor optical-mousing surface is unfrosted glass.

In practice, an optical mouse does not need cleaning, because it has no moving parts. This all-electronic feature also eliminates mechanical fatigue and failure. If the device is used with the

proper surface, sensing is more precise than is possible with any pointing device using the old electromechanical design. This is an asset in graphics applications, and it makes computer operation easier in general.

HOLOGRAME:

Holography is a technique that enables a wavefront to be recorded and later re-constructed. Holography is best known as a method of generating three-dimensional images, but it also has a wide range of other applications. In principle, it is possible to make a hologram for any type of wave.

A **hologram** is made by superimposing a second wavefront (normally called the reference beam) on the wavefront of interest, thereby generating an interference patterns which is recorded on a physical medium. When only the second wavefront illuminates the interference pattern, it is diffracted to recreate the original wavefront. Holograms can also be computer generated by modelling the two wavefronts and adding them together digitally. The resulting digital image is then printed onto a suitable mask or film and illuminated by a suitable source to reconstruct the wavefront of interest.

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

PROJECT OBJECTIVE:

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, such as the right/left clicks, scroll up/down, and more

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.

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 is 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.

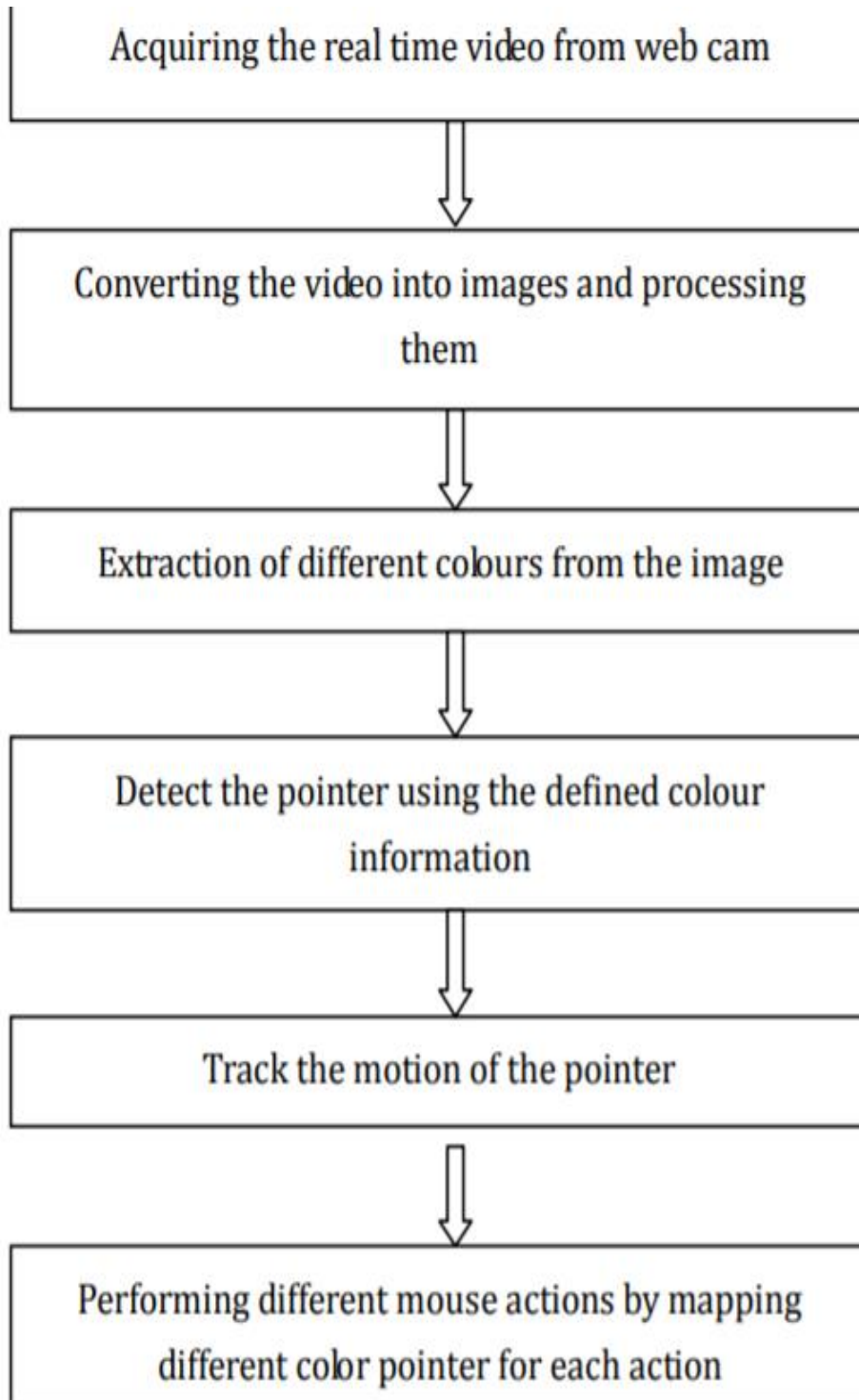
To detect the colours, they have utilized the MATLAB's built in "imsubtract" function, with the combination of the noise filtering by using median filter, which are effective in filtering out or at least reduce the "salt and pepper" noise.

The captured image will be converted to Binary Scale Image by using MATLABs 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 - 2

WORKING PRINCIPAL:

1) SYSTEM FLOW



SYSTEM APPROACH:

- Capturing real time video using Web-Camera.
- Processing the individual image frame.
- Flipping of each image frame.
- Conversion of each frame to a grey scale image.
- Colour detection and extraction of the different colours (RGB) from flipped grey scale image
- Conversion of the detected image into a binary image.
- Finding the region of the image and calculating its centroid.
- Tracking the mouse pointer using the coordinates obtained from the centroid.
- Simulating the left click and the right click events of the mouse by assigning different colour pointers.

CAPTURING THE REAL TIME VIDEO:

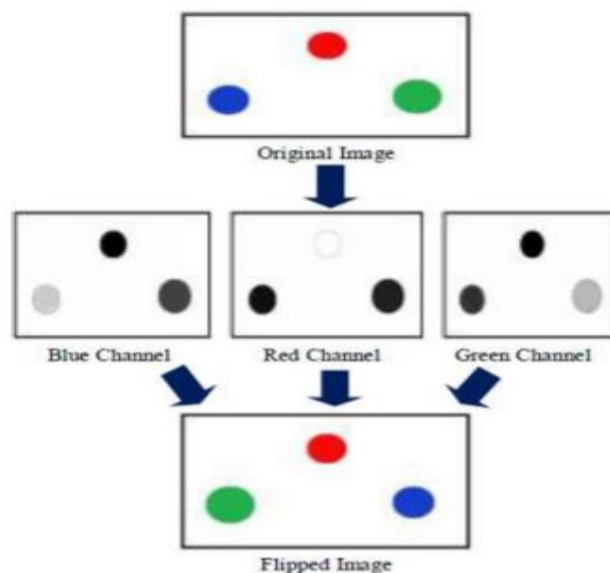
For the system to work we need a sensor to detect the hand movements of the user. the webcam of the computer is used as a sensor. the webcam captures the real time video at a fixed frame rate and resolution which is determined by the hardware of the camera. the frame rate and resolution can be changed in the system if required.

- 1) Computer webcam is used to capture the real time video
- 2) Video is divided into image frames based on the fps (frames per second) of the camera



FLIPPING OF IMAGES:

When the camera captures an image, it is inverted. This means that if we move the color pointer towards the left, the image of the pointer moves towards the right and vice-versa. It's similar to an image obtained when we stand in front of a mirror (Left is detected as right and right is detected as left). To avoid this problem, we need to vertically flip the image. The image captured is an RGB image and flipping actions cannot be directly performed on it. So the individual colour channels of the image are separated and then they are flipped individually. After flipping the red, blue and green coloured channels individually, they are concatenated and a flipped RGB image is obtained.



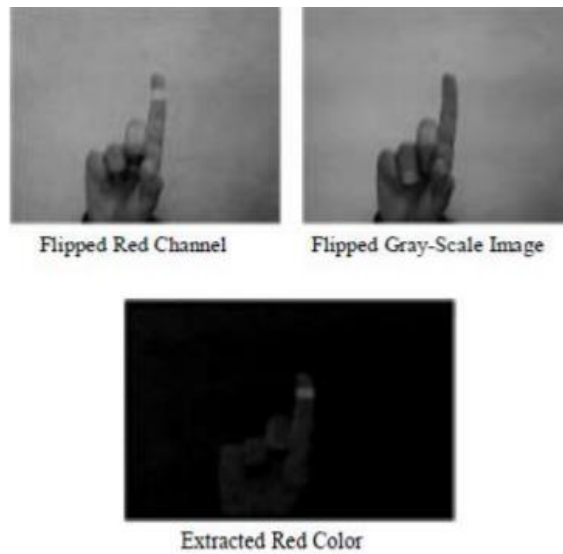
CONVERSION OF FLIPPED IMAGE INTO GRAY SCALE IMAGE:

As compared to a coloured image, computational complexity is reduced in a Gray scale image. Thus, the flipped image is converted into a Gray scale image. All the necessary operations were performed after converting the image into Gray scale.

COLOUR DETECTION:

This is the most important step in the whole process. The red, green and blue color object is detected by subtracting the flipped color suppressed channel from the flipped Gray-

Scale Image. This creates an image which contains the detected object as a patch of grey surrounded by black space.



Conversion of Gray scale Image into Binary scale Image:

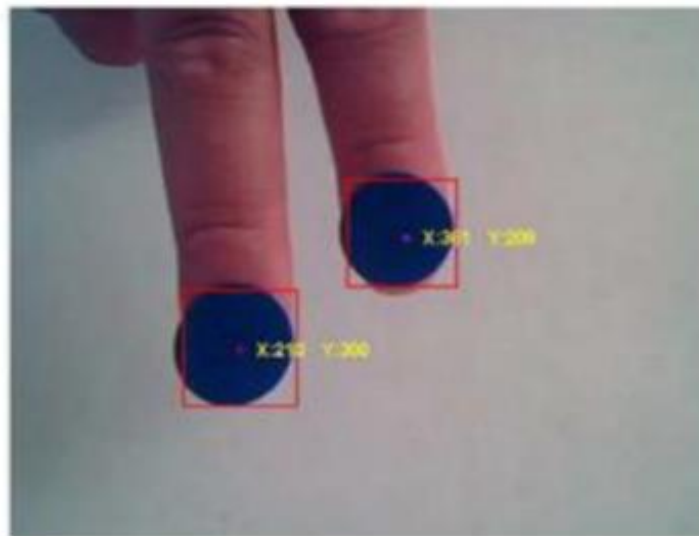
The grey region of the image obtained after subtraction needs to be converted to a binary image for finding the region of the detected object. A grayscale image consists of a matrix containing the values of each pixel. The pixel values lay between the ranges 0 to 255 where 0 represents pure black and 255 represents pure white color. We use a threshold value to convert the image to a binary image. This means that all the pixel values lying below threshold value is converted to pure black that is 0 and the rest is converted to white that is 255. Thus the resultant image obtained is a monochromatic image consisting of only black and white colors. The conversion to binary is required because MATLAB can only find the properties of a monochromatic image.



FINDING CENTROID OF AN OBJECT AND PLOTTING BOUNDING BOX:

For the user to control the mouse pointer it is necessary to determine a point whose coordinates can be sent to the cursor. With these coordinates, the system can control the cursor movement. An inbuilt function in MATLAB is used to find the centroid of the detected region. The output of function is a matrix consisting of the X (horizontal) and Y (vertical) coordinates of the centroid. These coordinates change with time as the object moves across the screen.

- Centroid of the image is detected and a bounding box is drawn around it.
- Its co-ordinates are located and stored in a variable



TRACKING THE MOUSE POINTER:

Once the coordinates have been determined, the mouse driver is accessed and the coordinates are sent to the cursor. With these coordinates, the cursor places itself in the required position. It is assumed that the object moves continuously, each time a new centroid is determined and for each frame the cursor obtains a new position, thus creating an effect of tracking. So as the user moves his hands across the field of view of the camera, the mouse moves proportionally across the screen. There is no inbuilt function in MATLAB which can directly access the mouse drivers of the computer. But MATLAB code supports integration with other languages like C, C++, and JAVA. Since java is a machine independent language so it is preferred over the others. A java object is created and it is linked with the mouse drivers. Based on the detection of other colours along with red the system performs the clicking events of the mouse. These colour codes can be customized based on the requirements.

Performing Clicking Actions

The control actions of the mouse are performed by controlling the flags associated with the mouse buttons. JAVA is used to access these flags. The user has to perform hand gestures in order to create the control actions. Due to the use of colour pointers, the computation time required is reduced. Furthermore, the system becomes resistant to background noise and low illumination conditions. The detection of green and blue colours follows the same procedure discussed above.

CLICKING ACTION IS BASED ON SIMULTANEOUS DETECTION OF TWO COLOURS:

- If Red along with single blue colour is detected, Left clicking action is performed.
- If Red along with double blue colour is detected, Right clicking action is performed.

PROBLEMS AND DRAWBACKS:

Since the system is based on image capture through a webcam, it is dependent on illumination to a certain extent. Furthermore, the presence of other coloured objects in the background might cause the system to give an erroneous response. Although by configuring the threshold values and other parameters of the system this problem can be reduced but still it is advised that the operating background be light and no bright coloured objects be present. The system might run slower on certain computers with low computational capabilities because it involves a lot of complex calculations in a very small amount of time. However, a standard pc or laptop has the required computational power for optimum performance of the system. Another fact is that if the resolution of the camera is too high then the system might run slow. However, this problem can be solved by reducing the resolution of the image by making changes in the system.

EXPERIMENTAL RESULTS:

The necessary steps to be followed are as follows:

- Firstly, we need to interface the camera using MATLAB.
- Start the camera to capture the images continuously.

- Capturing the real time video using the started-up camera.
- Processing the individual image frame obtained from the video.
- Flipping of each image frame.
- Conversion of each frame to a GRAY-SCALE image.
- Colour detection and extraction of different colours.
- Conversion of each frame into BINARY-SCALE image.
- Find the centre and draw the bounding box.
- Track mouse pointer using coordinates of centroid.

DIFFERENT MOUSE ACTIONS ARE PERFORMED BASED ON THE STEPS BELOW:

- a. First, we need to detect red colour in the image.
- b. If Red colour pointer is detected, we need to track the mouse pointer.
- c. If a single blue colour pointer is detected, we need to perform Left click.
- d. If two blue pointers are detected, we need to perform Right clicking action.
- e. If three blue pointers are detected, we need to perform Double clicking action.
- f. If none of the colour has been detected again we need to try tracking different colours from the image.

RECOGNITION PHASE:

a) Webcam & Variables Initialization

On the early stage of the recognition phase, the program will initialize the required variables which will be used to hold different types of frames and values where each are will be used to carry out certain task. Furthermore, this is the part where the program collects the calibrated HSV values and settings where it will be used later during the transitions of Binary Threshold.

b) Real Time Image Acquisition

The real time image is captured by using the webcam by using (cv:Video Capture cap(0);), where every image captured are stored into a frame variable (cv::Mat), which will be flipped and compressed to a reasonable size to reduce process load.

c) Frame Noise Filtering

Similar to the noise filtering during the calibration phase, Gaussian filters will be applied to reduce the existing noise of the captured frames. This can be done by using Gaussian Blur (Input Array src, Output Array dst, Size ksize, double sigmaX, double sigmaY=0, intborderType=BORDER_DEFAULT).

d) HSV Frame Transition

The captured frame requires to be converted from a BGR format to a HSV format. Which can be done by using cvtColor (src, dst, CV_BGR2HSV).

e) Binary Threshold Transition

The converted HSV frame will undergo a range check to check if the HSV values of the converted frame lies between the values of the HSV variables gathered during the calibration phase. The result of the range check will convert the frame into a Binary Threshold, where a part of the frame will set to 255 (1 bit) if the said frame lies within the specified HSV values, the frame will set to 0 (0 bit) if otherwise.

f) Binary Threshold Morphological Transformation

After the binary threshold is obtained, the frame will undergo a process called Morphological Transformation, which is a structuring operation to eliminate any holes and small object lurking around the foreground. The transformation consists of two morphological operators, known as Erosion and Dilation. The Erosion operator are responsible for eroding the boundaries of the foreground object, decreasing the region of the binary threshold, which is useful for removing small noises. As for Dilation, it is an opposite of erosion, it increases the region of the binary threshold, allowing eroded object to return to its original form. For the current project, both operators were used for morphological Opening and Closing, where Opening consists of combination of erosion followed by dilation, which is very useful in removing noise, whereas Closing is the opposite of Opening, which is useful in closing small holes inside the foreground object.

g) Colour Combination Comparison

After obtaining results from Morphological Transformation process, the program will calculate the remaining number of objects by highlighting it as blobs, this process requires cvBlob library, which is an add-on to OpenCV. The results of the calculation will then send for comparison to determine the mouse functions based on the colour combinations found within the captured frames.

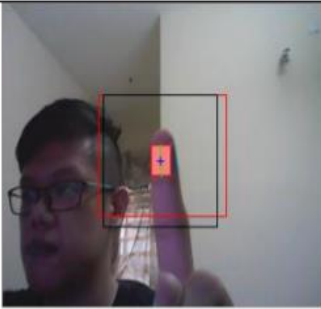

h) Colours' Coordinates Acquisition


For every object within the binary threshold, the program will highlight the overall shape of the object (cvRenderBlobs (const IplImage *imgLabel, CvBlobs &blobs, IplImage *imgSource, IplImage *imgDest, unsigned short mode=0x000f, double alpha=1.) ;), where it will calculate the area of the shape and the coordinates of midpoint of the shapes. The coordinates will be saved and used later in either setting cursor positions, or to calculate the distance between two points to execute various mouse functions based on the result collected.

I. Execution of Mouse Action

The program will execute the mouse actions based on the colour's combinations exist in the processed frame. The mouse actions will perform according to the coordinates provided by the program, and the program will continue on acquire and process the next real-time image until the users exit from the program.

The following describes the colour combinations to trigger and execute the mouse functions:

Sample Layout	Mouse Functions	Colour Combinations
	Move Cursor	First Colour (e.g. Blue)
	Left Click Up	First Colour + Second Colour (Un-pinch Gesture) (e.g. Blue + Green)

	Left Click Down	First Colour + Second Colour (Pinch Gesture) (e.g. Blue + Green)
	Right Click	First Colour + Third Colour (e.g. Blue + Red)
	Enable/Disable Scroll Mode	First Colour + Second Colour + Third Colour (e.g. Blue + Green + Red)
	Scroll Down	(During Scroll Mode) First Colour + Second Colour (Un-pinch Gesture) (e.g. Blue + Green)
	Scroll Up	(During Scroll Mode) First Colour + Second Colour (Pinch Gesture) (e.g. Blue + Green)

HARDWARE AND SOFTWARE REQUIREMENTS:

HARDWARE REQUIREMENTS:

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

Processor: Core2Duo

Main Memory: 4GB RAM

Hard Disk: 320GB

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.
 - Intel Pentium D processor 1.8 GHz or AMD Athlon X2 processor 1.8 GHz or higher
 - 3 GB RAM
 - 5 GB HDD space.
 - Peripheral Webcam at least 30 Frames /sec, 640 x 480 resolution.

SOFTWARE REQUIREMENTS:

- Windows XP x 86 or higher (for x 86 environment)
- Windows XP professional x 64 or higher (for x 64 environment)
- .NET framework 3.5 or higher
- Visual studio 2008 professional
- EmguCV library (Wrapper of OpenCV library for .NET framework)
- EmguCV library 64-bit binaries (for developing on x 64 environment)
- Webcam drives (device specific)

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.

- 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 has the ability to create real time eye tracking and blink detection.

Software will be using:

- OS: Window 7 Ultimate 64-bit
- Language: C++
- Tool Used: Open CV and CMake.

VERIFICATION PLAN:

The Virtual Mouse Colour Recognition requires being able to recognize most of the colours provided by the users with high accuracy, consistency, and minimal performance impact on other processes. However, the recognition results may varies whenever the qualities of the captured frames have changed, as it may be affected by different situation in terms of environment, brightness, and the weather. The following describes the situations which may result in false detection and/or any other problem that may occur during recognition phase:

- a) The real-time images are taken under dark or bright environment condition.
- b) The real-time images are taken in a colour conflicts background.
- c) The users interact with the program in near or far distance.
- d) The real-time images are rotated in a clockwise or anti-clockwise rotation. In order to achieve greater accuracy and consistency throughout the whole recognition cycle, verification plan is required to be implemented in order for the program to perform flawlessly.

The verification plans is as follows:

a) Dark or bright environment condition

Procedure Number	P1
Method	Testing
Applicable Requirements	Recognize colours input in a dark environment
Purpose / Scope	To recognize colours input in a dark environment
Item Under Test	Real-time image captured from webcam
Precautions	The targeted colours must be calibrated first
Special Conditions/Limitations	If the colours are not calibrated, the program will not be able to detect the targeted colours accurately.
Equipment/Facilities	Laptop
Data Recording	None
Acceptance Criteria	The colours are detected successfully
Procedure	<ul style="list-style-type: none"> • Run the program in the dark room. • Calibrate the desired colours. • Performs colour recognition by attempting to execute mouse functions by interacting with the

	program.
Troubleshooting	Modify the HSV track-bars
Post-Test Activities	None

b) Colour conflicts condition

Procedure Number	P2
Method	Testing
Applicable Requirements	Recognize colours input in a colour conflicts background.
Purpose / Scope	To recognize colours input in a colour conflicts background.
Item Under Test	Real-time image captured from webcam
Precautions	The targeted colours must be calibrated first
Special Conditions/Limitations	If the colours are not calibrated, the program will not be able to detect the targeted colours accurately.
Equipment/Facilities	Laptop
Data Recording	None
Acceptance Criteria	Program ignores inputs to avoid unnecessary result.
Procedure	<ul style="list-style-type: none"> • Run the program with more than one colour that has similar RGB values (e.g. blue and sky-blue). • Calibrate the desired colours. • Performs colour recognition by attempting to execute mouse functions by interacting with the program.
Troubleshooting	Modify the HSV track-bars
Post-Test Activities	None

CHAPTER - 3

IMPLEMENTATION AND TESTING


Overview In order to achieve accuracy, and consistency of the Virtual Mouse colour recognition, testing phase have been conducted on various scenarios.

- Performance in various environments
 - Brightness
 - Colour Conflicts
 - Tilt and Rotation

Distance. The purpose of testing phase is to ensure that the final deliverable is able to perform flawlessly in terms of accuracy, consistency, and performance. To achieve that, the program has to able to recognize the colours input provided by the users with minimal adjustment, provide that the colours are thoroughly calibrated at first hand. Furthermore, the program is required to be able to execute the mouse functions efficiently and accurately as well.

PERFORMANCE IN VARIOUS ENVIRONMENTS:

The following describes the outcome of the program testing in various environments:

Brightness	
Normal Environment	
	<p>Results:</p> <p>All colours are successfully recognized. The three highlighted squares indicate that the targeted colours are identified, compared, and execute accordingly.</p>

Colour Conflicts



Results:

Colours input are successfully ignored, the conflicts were detected, causing it to stop executing unwanted mouse functions until the colour conflicts are no longer within the frame.

Table 5.2: Colour conflicts testing results

Tilt and Rotation



Results:

All colours are successfully recognized. The three highlighted squares indicate that the targeted colours are identified, compared, and execute accordingly.

	Left Click Down	First Colour + Second Colour (Pinch Gesture) (e.g. Blue + Green)
	Right Click	First Colour + Third Colour (e.g. Blue + Red)
	Enable/Disable Scroll Mode	First Colour + Second Colour + Third Colour (e.g. Blue + Green + Red)
	Scroll Down	(During Scroll Mode) First Colour + Second Colour (Un-pinch Gesture) (e.g. Blue + Green)
	Scroll Up	(During Scroll Mode) First Colour + Second Colour (Pinch Gesture) (e.g. Blue + Green)

SOURCE CODE:

Hand Tracking Module:

```
import cv2
import mediapipe as mp
import time

cap = cv2.VideoCapture(0)

mpHands = mp.solutions.hands
hands = mpHands.Hands()
mpDraw = mp.solutions.drawing_utils

pTime = 0
cTime = 0

while True:
    success, img = cap.read()
    imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    results = hands.process(imgRGB)
    # print(results.multi_hand_landmarks)

    if results.multi_hand_landmarks:
        for handLms in results.multi_hand_landmarks:
            for id, lm in enumerate(handLms.landmark):
                # print(id, lm)
```

```
h, w, c = img.shape
```

```
cx, cy = int(lm.x * w), int(lm.y * h)
```

```
print(id, cx, cy)
```

```
# if id == 4:
```

```
cv2.circle(img, (cx, cy), 15, (255, 0, 255), cv2.FILLED)
```

```
mpDraw.draw_landmarks(img, handLms, mpHands.HAND_CONNECTIONS)
```

```
cTime = time.time()
```

```
fps = 1 / (cTime - pTime)
```

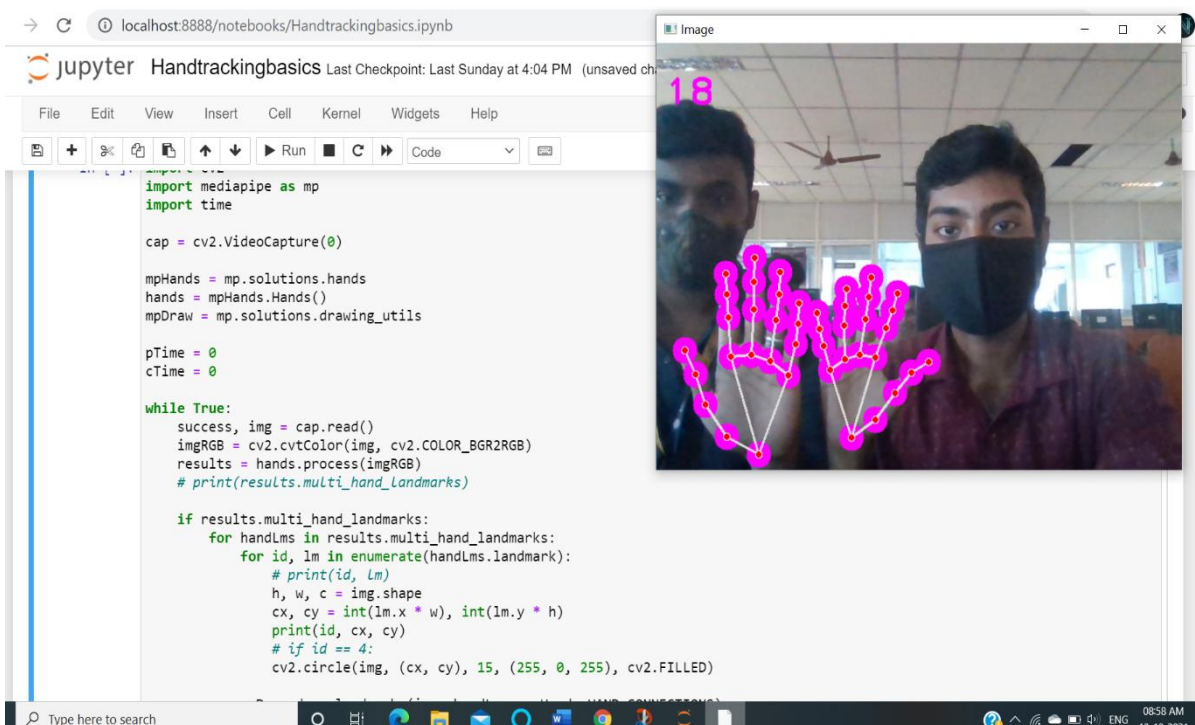
```
pTime = cTime
```

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

```
cv2.imshow("Image", img)
```

```
cv2.waitKey(1)
```

RESULT:



VIRTUAL MOUSE CODE:

```
import cv2
import numpy as np
import HandTrackingModule as htm
import time
import autopy

#####

wCam, hCam = 640, 480
frameR = 100    #Frame Reduction
smoothing = 7   #random value
#####

pTime = 0
plocX, plocY = 0, 0
clocX, clocY = 0, 0
cap = cv2.VideoCapture(0)
cap.set(3, wCam)
cap.set(4, hCam)

detector = htm.handDetector(maxHands=1)
wScr, hScr = autopy.screen.size()

# print(wScr, hScr)

while True:
```


Step1: Find the landmarks

```
success, img = cap.read()
```

```
img = detector.findHands(img)
```

```
lmList, bbox = detector.findPosition(img)
```

```
if len(lmList) != 0:
```

```
    x1, y1 = lmList[8][1:]
```

```
    x2, y2 = lmList[12][10:]
```

Step3: Check which fingers are up

```
fingers = detector.fingersUp()
```

```
cv2.rectangle(img, (frameR, frameR), (wCam - frameR, hCam - frameR),  
              (255, 0, 255), 2)
```

Step4: Only Index Finger: Moving Mode

```
if fingers[1] == 1 and fingers[2] == 0:
```

Step5: Convert the coordinates

```
x3 = np.interp(x1, (frameR, wCam-frameR), (0, wScr))
```

```
y3 = np.interp(y1, (frameR, hCam-frameR), (0, hScr))
```

Step6: Smooth Values

```
clocX = plocX + (x3 - plocX) / smoothening
```

```
clocY = plocY + (y3 - plocY) / smoothening
```

Step7: Move Mouse

```
autopy.mouse.move(wScr - clocX, clocY)
```

```
cv2.circle(img, (x1, y1), 15, (255, 0, 255), cv2.FILLED)
```

```
plocX, plocY = clocX, clocY
```

Step8: Both Index and middle are up: Clicking Mode

if fingers[1] == 1 and fingers[2] == 1:

Step9: Find distance between fingers

length, img, lineInfo = detector.findDistance(8, 12, img)

Step10: Click mouse if distance short

if length < 40:

cv2.circle(img, (lineInfo[4], lineInfo[5]), 15, (0, 255, 0), cv2.FILLED)

autopy.mouse.click()

Step11: Frame rate

cTime = time.time()

fps = 1/(cTime-pTime)

pTime = cTime

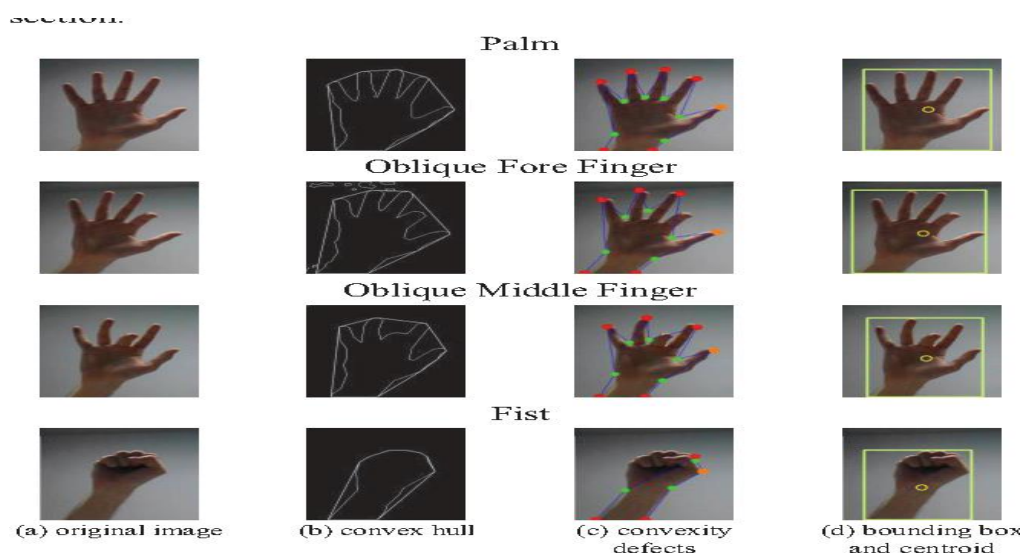
cv2.putText(img, str(int(fps)), (28, 58), cv2.FONT_HERSHEY_PLAIN, 3, (255, 8, 8), 3)

Step12: Display

cv2.imshow("Image", img)

cv2.waitKey(1)

SAMPLE REOPRT:



VOLUME GESTURE SOURCE CODE:

```
import cv2
import mediapipe as mp
from math import hypot
from ctypes import cast, POINTER
from comtypes import CLSCTX_ALL
from pycaw.pycaw import AudioUtilities, IAudioEndpointVolume
import numpy as np

cap = cv2.VideoCapture(0)

mpHands = mp.solutions.hands
hands = mpHands.Hands()
mpDraw = mp.solutions.drawing_utils

devices = AudioUtilities.GetSpeakers()
interface = devices.Activate(IAudioEndpointVolume._iid_, CLSCTX_ALL, None)
volume = cast(interface, POINTER(IAudioEndpointVolume))

volMin,volMax = volume.GetVolumeRange()[:2]

while True:
    success,img = cap.read()
    imgRGB = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
```

```

results = hands.process(imgRGB)

lmList = []
if results.multi_hand_landmarks:
    for handlandmark in results.multi_hand_landmarks:
        for id,lm in enumerate(handlandmark.landmark):
            h,w,_ = img.shape
            cx,cy = int(lm.x*w),int(lm.y*h)
            lmList.append([id,cx,cy])
        mpDraw.draw_landmarks(img,handlandmark,mpHands.HAND_CONNECTIONS)

if lmList != []:
    x1,y1 = lmList[4][1],lmList[4][2]
    x2,y2 = lmList[8][1],lmList[8][2]

    cv2.circle(img,(x1,y1),4,(255,0,0),cv2.FILLED)
    cv2.circle(img,(x2,y2),4,(255,0,0),cv2.FILLED)
    cv2.line(img,(x1,y1),(x2,y2),(255,0,0),3)

    length = hypot(x2-x1,y2-y1)

    vol = np.interp(length,[15,220],[volMin,volMax])
    print(vol,length)
    volume.SetMasterVolumeLevel(vol, None)

    # Hand range 15 - 220
    # Volume range -63.5 - 0.0

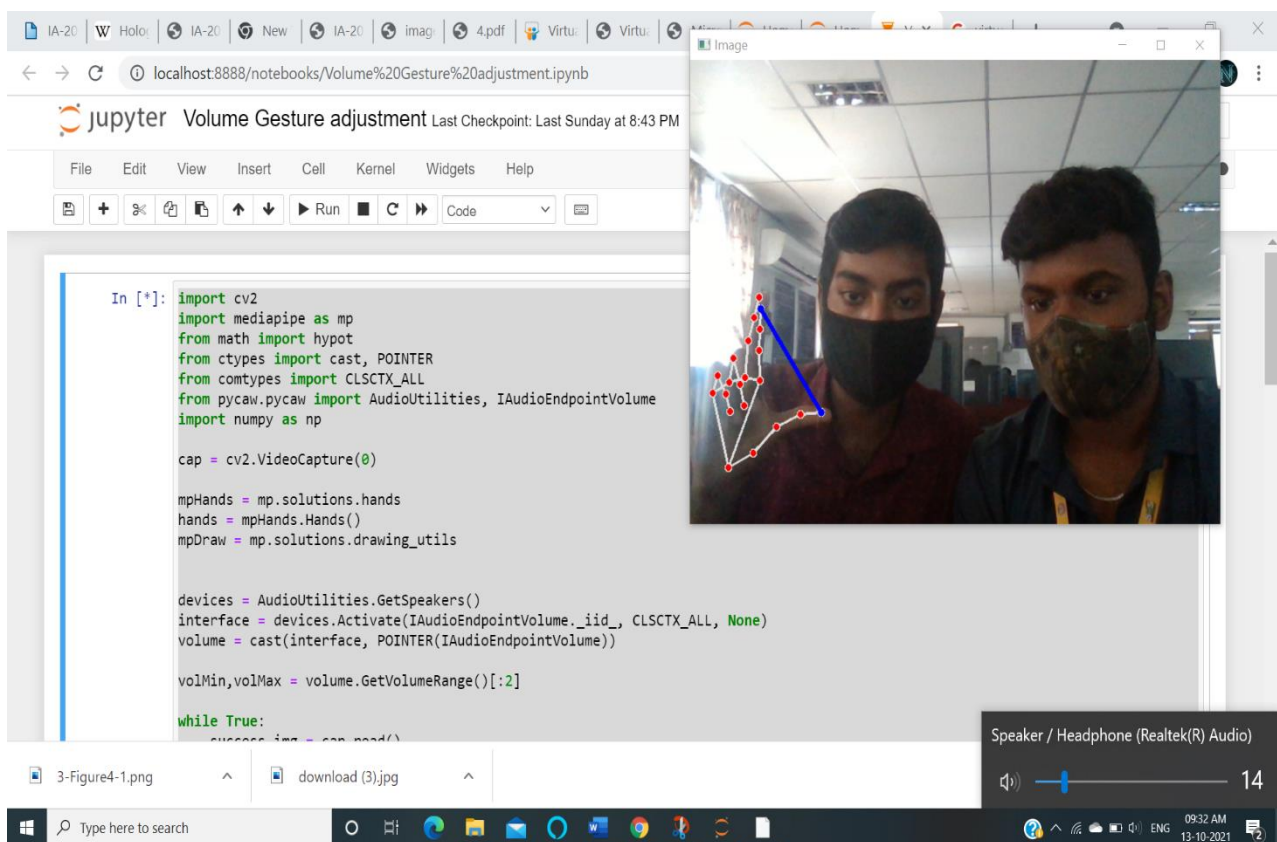
cv2.imshow('Image',img)

```

```
if cv2.waitKey(1) & 0xff==ord('q'):
```

```
    break
```

OUTPUT:



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.

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 machine with a webcam that performs better in different types of lightings.

CONCLUSION:

Overview 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 accepting changes of the newer one. Instead, it's advisable that they should embrace changes to have a more efficient, and productive lifestyle.

REFERENCES:

- [1] Amardip Ghodichor, Binitha Chirakattu “Virtual Mouse using Hand Gesture and Colour Detection”, Volume 128 – No.11, October 2015.
- [2] Chhoriya P., Paliwal G., Badhan P., 2013, “Image Processing Based Color Detection”, International Journal of Emerging Technology and Advanced Engineering, Volume 3, Issue 4, pp. 410-415
- [3] Rhitivij Parasher, Preksha Pareek ,”Event triggering Using hand gesture using open cv”, volume -02-february,2016 page No.15673-15676.
- [4] AhemadSiddique, Abhishek Kommera, DivyaVarma, ” Simulation of Mouse using Image Processing Via Convex Hull Method ”, Vol. 4, Issue 3, March 2016.
- [5] Student, Department of Information Technology, PSG College of Technology, Coimbatore, Tamilnadu, India,”Virtual Mouse Using Hand Gesture Recognition ”,Volume 5 Issue VII, July 2017.
- [6] Kalyani Pendke¹ , Prasanna Khuje² , Smita Narnaware³ , Shweta Thool⁴ , Sachin Nimje⁵ ,”International Journal of Computer Science and Mobile Computing ”,IJCSMC, Vol. 4, Issue. 3, March 2015.
- [7] [. Abhilash S S¹, Lisho Thomas², Naveen Wilson³, Chaithanya C⁴,”VIRTUAL MOUSE USING HAND GESTURE”, Volume: 05 Issue: 04 | Apr-2018.
- [8] Abdul Khaliq and A. Shahid Khan, “Virtual Mouse Implementation Using Color Pointer Detection”, International Journal of Electrical Electronics & Computer Science Engineering, Volume 2, Issue 4, August, 2015, pp. 63-66
- [9] Erdem, E. Yardimci, Y. Atalay, V. Cetin, A. E., “Computer vision based mouse”, Acoustics, Speech, and Signal Processing, Proceedings (ICASS), IEEE International Conference, 2002.
- [10] Chu-Feng Lien, “Portable Vision-Based HCI – A Realtime Hand Mouse System on Handheld Devices”, National Taiwan University, Computer Science and Information Engineering Department

- [11] Hojoon Park, "A Method for Controlling the Mouse Movement using a Real Time Camera", Brown University, Providence, RI, USA, Department of Computer Science, 2008.
- [12] AsanterabiMalima, Erol Ozgur, and Mujdat Cetin, "A FastAlgorithm for Vision-Based Hand Gesture Recognition for Robot Control"
- [13] using Hand Gesture Recognition", InternationalJournal of Engineering Sciences & Research Technology,ISSN:2277- 9655,March 2014.
- [14] ShanyJophin, Sheethal M.S, Priya Philip, T M Bhruguram, "Gesture Based Interface Using Motion and Image Comparison", International Journal of Advanced Information Technology (IJAIT) Vol. 2, No.3, June 2012.
- [15] Abhik Banerjee, Abhirup Ghosh, Koustuvmoni Bharadwaj, HemantaSaik, Mouse Control using a Web Camera based on ColourDetection", International Journal of Computer Trends and Technology (IJCTT) –volume 9 number 1, ISSN:2231-2803, March 2014.
- [16] S.Sadhana Rao, "Sixth Sense Technology", Proceedings of the International Conference on Communication and Computational Intelligence– 2010, pp.336-339.
- [17] Game P. M., Mahajan A.R,"A gestural user interface to Interact with computer system ", International Journal on Science and Technology (IJSAT) Volume II, Issue I, (Jan.- Mar.) 2011, pp.018 – 027
- [18] Abhik Banerjee, Abhirup Ghosh, Koustuvmoni Bharadwaj," Mouse Control using a Web Camera based on Color Detection",IJCTT,vol.9, Mar 2014.
- [19] Angel, Neethu.P.S, "Real Time Static & Dynamic Hand Gesture Recognition", International Journal of Scientific & Engineering Research Volume 4, Issue3, March-2013.
- [20] Q. Y. Zhang, F. Chen and X. W. Liu, "Hand Gesture Detection and Segmentation Based on Difference Back-ground Image with Complex Background," Proceedings of the 2008 International Conference on Embedded Soft-ware and Systems, Sichuan, 29- 31 July 2008, pp. 338-343

- [21] A. Erdem, E. Yardimci, Y. Atalay, V. Cetin, A. E. “Computer vision based mouse”, Acoustics, Speech, and Signal Processing, Proceedings. (ICASS). IEEE International Conference, 2002
- [22] Hojoon Park, “A Method for Controlling the Mouse Movement using a Real Time Camera”, Brown University, Providence, RI, USA, Department of computer science, 2008
- [23] Chu-Feng Lien, “Portable Vision-Based HCI –A Real-time Hand Mouse System on Handheld Devices”, National Taiwan University, Computer Science and Information Engineering Department
- [24] Kamran Niyazi, Vikram Kumar, Swapnil Mahe, Swapnil Vyawahare, “Mouse Simulation Using Two Coloured Tapes”, Department of Computer Science, University of Pune, India, International Journal of Information Sciences and Techniques (IJIST) Vol.2, No.2, March 2012
- [25] K N. Shah, K R. Rathod and S. J. Agravat, “A survey on Human Computer Interaction Mechanism Using Finger Tracking”,