

# **AI Virtual Mouse**

**A PROJECT REPORT**

**for**

**Mini Project (KCA353)**

**Session (2023-24)**

**Submitted by**

**Aastha Gupta**

**University Roll No-2200290140001**

**Abhishek kumar**

**University Roll No- 2200290140008**

**Submitted in partial fulfillment of  
the Requirements for the Degree of**

## **MASTER OF COMPUTER APPLICATION**

**Under the Supervision of**

**Ms. Neelam Rawat**

**(Associate Professor)**



**Submitted to**

**DEPARTMENT OF COMPUTER APPLICATIONS**

**KIET Group of Institutions, Ghaziabad Uttar Pradesh-201206**

**(FEB 2024)**

# CERTIFICATE

Certified that **Aastha Gupta 2200290140001, Abhishek Kumar 2200290140008** has/have carried out the project work having “**AI Virtual Mouse**” (**Mini Project-KCA353**) for **Master of Computer Application** from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU), Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself/herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

**Date:**

**Aashta Gupta (2100290140001)**

**Abhishek Kumar (2100290140008)**

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

**Date:**

**Ms. Neelam Rawat**  
**Associate Professor**  
**Department of Computer Applications**  
**KIET Group of Institutions, Ghaziabad**

**Dr. Arun Tripathi**  
**Head**  
**Department of Computer Applications**  
**KIET Group of Institutions, Ghaziabad**

# **ABSTRACT**

This project promotes an approach for the Human-Computer Interaction (HCI) where cursor movement can be controlled using a real-time camera, it is an alternative to the current methods including manual input of buttons or changing the positions of a physical computer mouse. Instead, it utilizes a camera and computer vision technology to control various mouse events and can perform every task that the physical computer mouse can

The Virtual Mouse color recognition program will constantly be acquiring real-time images where the images will undergo a series of filtration and conversion. Whenever the process is complete, the program will apply the image processing technique to obtain the coordinates of the position of the targeted colors from the converted frames. After that, it will proceed to compare the existing colors within the frames with a list of color combinations, where different combinations consist of different mouse functions. If the current colors combination found a match, the program will execute the mouse function, which will be translated into an actual mouse function for the users' machine

## **ACKNOWLEDGEMENTS**

Success in life is never attained single-handedly. My deepest gratitude goes to my project supervisor Ms. Neelam Rawat for his/ her guidance, help, and encouragement throughout my project work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to Dr. Arun Kumar Tripathi, Professor and Head, Department of Computer Applications, for his insightful comments and administrative help on various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me with moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

**Aastha Gupta (2200290140001)**

**Abhishek Kumar (2200290140008)**

# CONTENTS

CERTIFICATE	i
ABSTRACT	ii
ACKNOWLEDGEMENT	iii

## CHAPTER: 1 PROJECT AND BACKGROUND

1.1 Introduction.....	
1.2 Literature overview.....	
1.3 Objectives and issues.....	
1.4 Scope.....	

## CHAPTER: 2 CONCEPT AND TECHNIQUES

2.1 Methodology.....	
2.2 Conceptual Review .....	
2.3 Feature and Scope.....	

## CHAPTER: 3 SYSTEM DESIGN AND ARCHITECTURE

3.1 Package Overview.....	
3.2 Architecture of Programming.....	
1.3 Working.....	

## CHAPTER: 4 TESTING AND RESULT ANALYSIS

1.1 Objective.....	
1.2 Recognition results .....	

## CHAPTER: 5 CONCLUSION AND FUTURE WORK

11-20

1.1 Overview.....	
1.2 Literature perspective of future work .....	

1.3 Advantages.....

## REFERENCES

# PROJECT BACKGROUND

## Introduction

With the development of 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 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 functions 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 an HCI 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 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 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 mouse function.

Python programming language is used for developing the AI virtual mouse system, and 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 the tip of the hands, 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 a very high accuracy level, and the proposed model can

work very well in real-world applications with the use of a CPU without the use of a GPU.

## **Literature 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 for the persons who have problems with their hands and are not able to control a physical mouse. Also, amidst the COVID-19 situation, it is not safe to use the devices by touching them because it may result in a possible situation of the spread of the virus by touching the devices, so the proposed AI virtual mouse can be used to overcome these problems since hand gestures and hand Tip detection is used to control the PC mouse functions by using a webcam or a built-in camera.

- To design a virtual mouse that detects hand gesture patterns instead of a physical mouse.
- We use colored tips for detection which are captured by webcam.
- Here, the colored fingertip acts as an object that the webcam senses.
- The camera is positioned such that it recognizes the moment of fingertips and performs the operations of the mouse.
- The utilization of virtual mouse appears in space-saving situations or in movement situations.

## **Objective and issues**

- 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 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 need needs of have a physical mouse while being able to interact with the computer system through a webcam by using various image processing techniques. Other than that, this project aims to develop a Virtual Mouse application that can be operated operate kinds of surfaces and d environments. 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 were no webcam detected.
- To design a virtual input that can operate on all surfaces. 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 continue capturing the images, which the images will be analyzed, by using various image processing techniques. As set above, the Virtual Mouse application will be continuously capturing the images in real-time, where the images will be undergoing a series of processes, this includes HSV conversion, Binary Image conversion, It 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 colors may result in triggering diverse types of mouse events, such as the right/left clicks, scroll up/down, and more.

## Scope

Virtual Mouse will soon be introduced to replace the physical computer mouse to promote convenience while still being able to accurately interact with and control the computer system. To do that, the software requires to be fast enough to capture and process every image, 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:

- For most laptop touch pad is not the most comfortable and convenient.
- Virtual mouse, known as Virtual Multitask Mouse.
- This is real time application.

- User friendly application.
- This project removes the requirement of having a physical.

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 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.

# CONCEPT AND TECHNIQUE

## Methodology

For this project, we'll be using the Agile Software Development methodology approach in developing the application. The stated approach is an alternative to the traditional waterfall model that helps the project team respond to unpredictability through incremental and iterative work. It promotes adaptive planning, evolutionary development, early delivery, and continuous improvement, and encourages rapid and flexible response to change.

The following describes the principles of the Agile Software Development methodology:

- Satisfy the customer by early and continuous delivery of workable software.
- Encourage changes in requirements.
- Workable software is delivered frequently.
- Continuous collaboration between the stakeholders and the developers.
- Projects are developed around motivated individuals.
- Encourage informal meetings.
- Operational software is the principal measure of progress.
- Sustainable development, able to maintain a constant pace.
- Continuous attention to technical excellence and good design
- Simplicity
- Self-organizing teams

- Regular adaption to changing circumstance

The reason for choosing this methodology is that the Virtual Mouse is still considered to be at the introduction stage, which means it still requires a great deal of extensive research and development before it could make it into the market. Therefore, this project requires a thorough yet iterative planning and requirements gathering where the lifecycle will be continually revisited to re-evaluate the direction of the project and eliminate the ambiguities in the process of development, and at the same time welcome changes of requirements, which promotes adaptability and flexibility. Furthermore, due to the Virtual Mouse application being more towards serving the users, this project requires continuous customer collaboration, as, as they are essential for gathering the proper requirements in all aspects. Therefore, the agile methodology is the ideal approach for developing the project.

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 vary whenever the qualities of the captured frames have changed, as they may be affected by the different situations in terms of environment, brightness, and weather. The following describes the situations which may result in false detection and/or any other problem that may occur during the recognition phase:

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

The aim of this paper is to implement a computer application that uses alternative methods to control keyboard and mouse cursors for the rehabilitation of people who are suffered from a stroke so that they can recover from the side effects. Therefore, we propose a new keyboard and mouse cursor control system based on the vision and recognition technique, utilizing hand gestures recorded from a webcam.

## **Conceptual review**

As modern technology of human-computer interactions become important in our everyday lives, varieties of mouse of all kinds of shapes, and sizes were invented, from a casual office mouse to a hard-core gaming mouse.

However, there are some limitations to this hardware as they are not as environmentally 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 this hardware is completely useless when it comes to interact with the computers remotely due to the cable length limitations, rendering it inaccessible.

There are traditional approaches for virtual keyboard and mouse systems which are usually based on eye gestures. Our literature review focuses on the research works on virtual keyboard and virtual mouse which were published in Elsevier, Springer, ACM (AWS Certificate Manager) Digital Library, IEEE Digital Library etc.

In 2016, S. Shetty et al. constructed a virtual mouse system using color detection. They used a webcam for detecting mouse cursor movement and click events using OpenCV built-in functions. A mouse driver, written in java, is quired as well. This system fails to perform well in the rough background. P. C. dhe et al. expanded a method for mouse-free cursor control where mouse cursor operations are controlled by using hand fingers. They have collected hand gestures via webcam using color detection principles. The built-in function of the Image Processing Toolbox in MATLAB and a mouse driver, written in java, is used in this approach. The pointer was not too efficient on the air as the cursor was very sensitive to the motion.

G. Sahu et al. built a system for controlling a mouse pointer using a webcam that controls the volume of the media player, and PowerPoint slides and can make or end a call. They used RGB color tapes to recognize the user's finger. In 2019, K. Hassan et al. presented a system to design and develop a hand gesture-based virtual mouse. They captured different gestures via webcam and performed mouse functions according to the gestures. This system achieved 78%-90% accuracy. The system does not work efficiently in a complex or rough background. As we can see from the reviewed literature, previous systems include either a virtual keyboard or a virtual mouse.

Those systems can't fully eliminate the need for a mouse and keyboard completely. This work aims to build an interactive computer system that can be operated without any physical mouse.

The current system is comprised of a generic mouse and trackpad monitor control system, as well as the absence of a hand gesture control system. The use of a hand gesture to access the monitor screen from a distance is not possible. Even though it is primarily attempting to implement, the scope is simply limited in the virtual mouse field. The existing virtual mouse control system consists of simple mouse operations using a hand recognition system, in which we can control the mouse pointer, left-click, right-click, drag, and so on. The use of hand recognition in the future will not be used. Even though there are a variety of systems for hand recognition, the system they used is static hand recognition, which is simply a recognition of the shape made by the hand and the definition of action for each shape made, which is limited to a few defined actions and causes a lot of confusion.

As technology advances, there are more alternatives to using a mouse.

## **FEATURES AND SCOPE**

- Features such as enlarging and shrinking windows, closing window, etc. by using the palm and multiple fingers are our future scope for this project.
- In the future, we plan to add more features such as enlarging and shrinking windows, closing windows, etc by using the palm and multiple fingers.

- We can also open the browser or any drives (C: /D:/E: etc) with the help of hand gestures instead of moving the cursor.

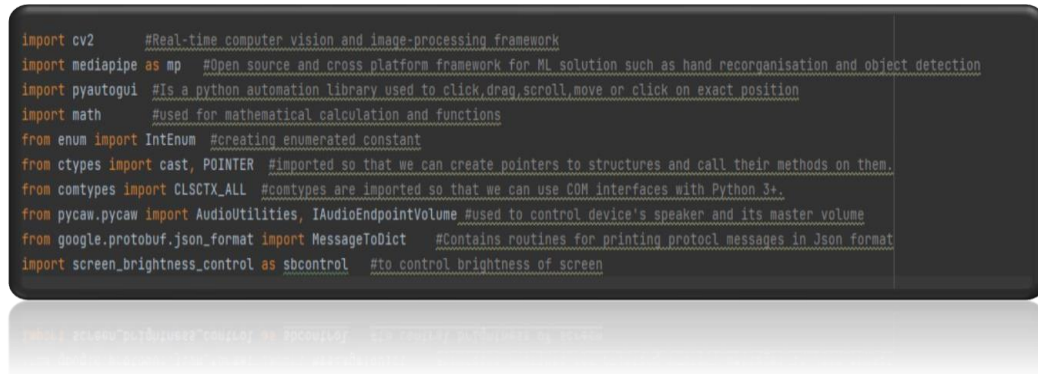
The Virtual Mouse application is expected to replace the current methods of utilizing a physical computer mouse where the mouse inputs and positions are done manually. This application offers a more effortless way to interact with the computer system, where every task can be done by gestures. Furthermore, the Virtual Mouse application could assist the motor-impaired users where he/she could interact with the computer system by just showing the correct pattern of fingers to the webcam.

The code is written in Python, and it employs the cross-platform image processing module OpenCV as well as the Python-specific library PyAutoGUI to implement mouse actions.

Skin detection can be defined as detecting the skin colour pixels in an image. It is a fundamental step in a wide range of image processing applications such as face detection, hand tracking and d gesture recognition. Skin detection using colour information has recently gained a lot of attention since it is computationally effective and provides robust information against scaling, rotation and partial occlusion. Skin detection using colour information can be a challenging task, since skin appearance in images is affected by illumination, camera characteristics, background and ethnicity. In order to reduce the effects of illumination, the image can be converted to a chrominance colour space, which is less sensitive to illumination changes. A chrominance colour space is one where the intensity information (luminance), is separated from the colour information.

# SYSTEM DESIGN AND PROGRAMMING

## Packages Overview



```
import cv2 #Real-time computer vision and image-processing framework
import mediapipe as mp #Open source and cross platform framework for ML solution such as hand recognisation and object detection
import pyautogui #Is a python automation library used to click,drag,scroll,move or click on exact position
import math #used for mathematical calculation and functions
from enum import IntEnum #creating enumerated constant
from ctypes import cast, POINTER #imported so that we can create pointers to structures and call their methods on them.
from ctypes import CLSCTX_ALL #ctypes are imported so that we can use COM interfaces with Python 3+.
from pycaw.pycaw import AudioUtilities, IAudioEndpointVolume #used to control device's speaker and its master volume
from google.protobuf.json_format import MessageToDict #Contains routines for printing protocol messages in Json format
import screen_brightness_control as sbcontrol #to control brightness of screen
```

Fig3.1:Imported packages

- The code starts by importing the necessary libraries.
- MediaPipe offers cross-platform, customizable ML solutions for live and streaming media. End-to-End acceleration: Built-in fast ML inference and processing accelerated even on common hardware. Build once, deploy anywhere: Unified solution works across Android, iOS, desktop/cloud, web and IoT.
- The cv2 library is used to import the functions for creating and manipulating images, while pyautogui is used to control a mouse cursor.
- The next line imports math which will be needed later in this code.
- Next, from enum import IntEnum is imported so that we can use enumerations instead of integers when defining colors.
- Then from ctypes import cast, POINTER are imported so that we can create pointers to structures and call their methods on them.



- Finally, ctypes are imported so that we can use COM interfaces with Python 3+.
- Next comes an if statement which checks whether or not FAILSAFE has been set as False before continuing with the rest of the code block.
- If it has been set as True then some error handling occurs because there's no point in trying to draw something if it won't work properly without crashing due to memory issues or other errors occurring during execution of this program.
- If FAILSAFE hasn't been set as False then a function called mp\_drawing() is defined which draws shapes onto an image using drawing utils provided by MP Solutions (a company who provides solutions for multimedia applications).
- This function takes two parameters: x
- The code is a part of the PyCaw library.
- The code above attempts to be used in conjunction with the PyCaw library.

## **Architecture Of Programming**

The basic flow is described as shown in the image below-

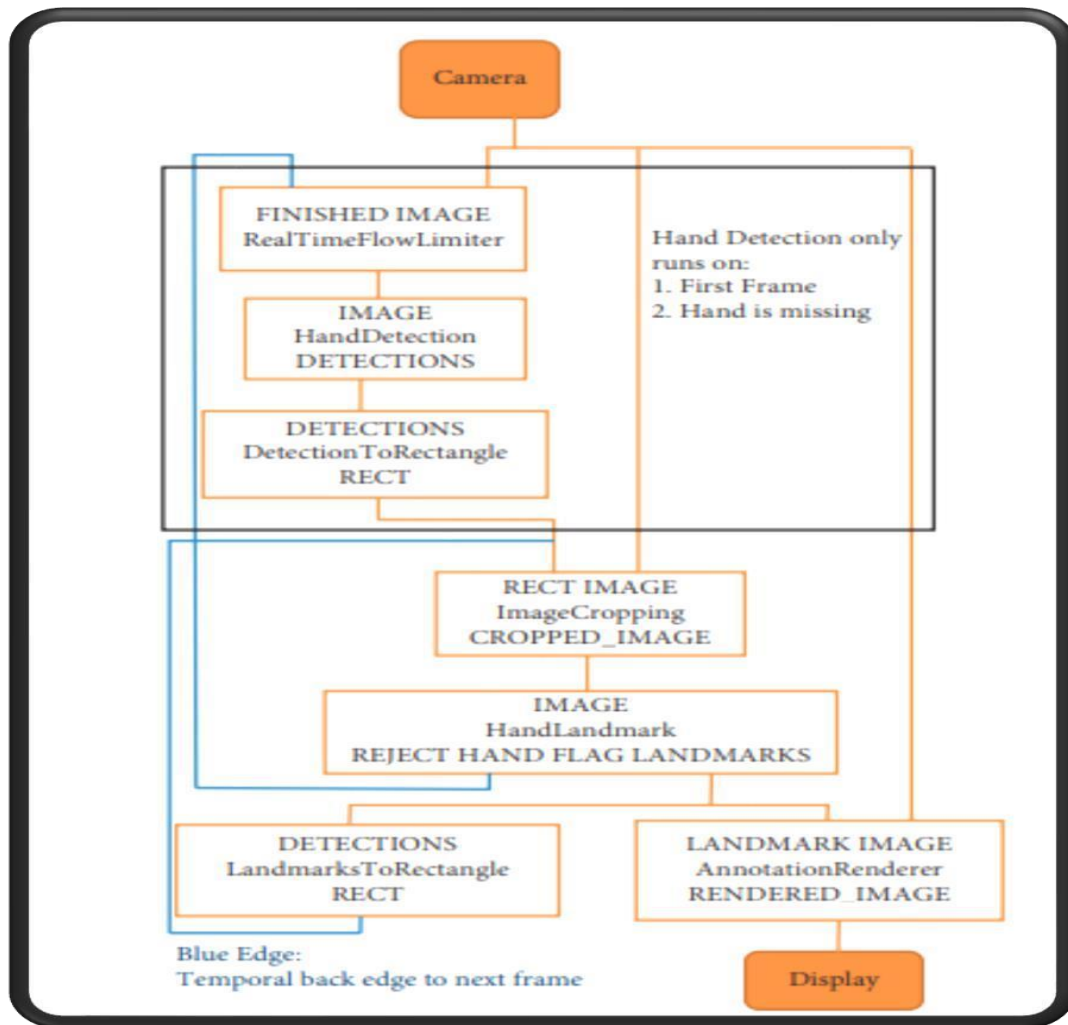


Fig 3.3: Basic Flow Of Project

There are total five classes their name as follows:

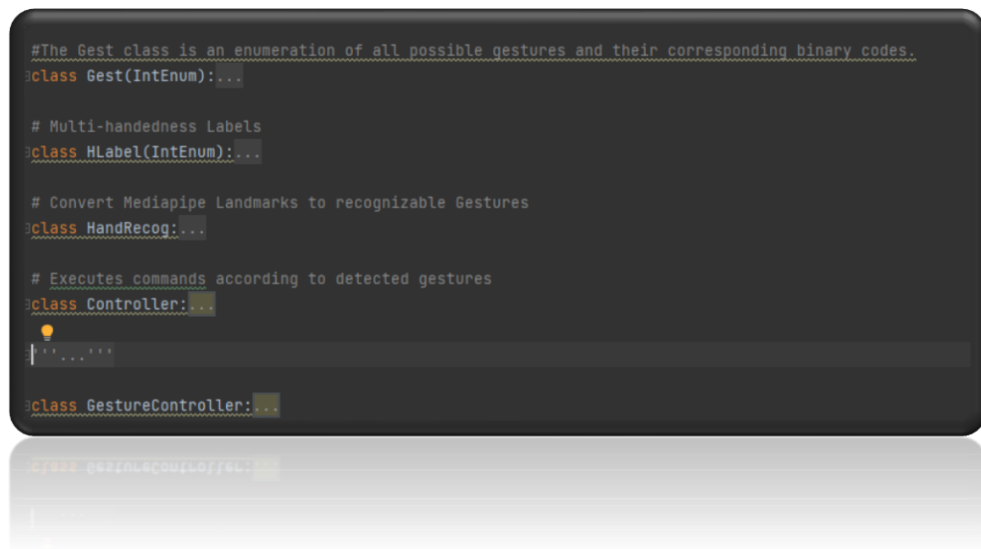


Fig 3.4: Classes Declaration

- **Class Gest**

The Gest class is an enumeration of all possible gestures and their corresponding binary codes. It is taking **IntEnum** as argument which is imported from enum so that we can use enumerations instead of integers when defining colors.

**The binary encoding for each gesture:**

**Cassss** Gest(IntEnum): # Binary Encoded

```
FIST = 0
PINKY = 1
RING = 2
MID = 4
LAST3 = 7
INDEX = 8
FIRST2 = 12
LAST4 = 15
THUMB = 16
PALM = 31

# Extra Mappings
```

```
V_GEST = 33
TWO_FINGER_CLOSED = 34
PINCH_MAJOR = 35
PINCH_MINOR = 36
```

- **Class HLabel**

The HLabel class has two members, MINOR and MAJOR. These are integers that can be used to identify which type of label it is.

```
class HLabel(IntEnum):
    MINOR = 0
    MAJOR = 1
```

- **Class HandRecog**

Convert Medeapipe Landmarks to recognizable gestures class HandRecog

The code is calculating the distance between two points on a chessboard. The first point is at coordinate (0, 0) and the second point is at coordinate (1, 1). The code starts by checking if `self.hand_result.landmark[point[0]].y < self.hand_result.landmark[point[1]].y` to see if the finger is closer to one of these points than it is to the other one; in this case, it will be true because they are both below each other so there would be no need for a signed variable or anything else that might indicate which way the player should move their piece next time around.

If this condition evaluates as true then we know that our fingers have moved their points towards one of these two points and we can use math functions like `dist = math.sqrt(dist)` to calculate how far away from them they have moved their point since it was last placed on its current point; in this case, `dist = 2*math.sqrt(2)`, which equals 4 units away from where they were originally pointing when pointing their point down on its current coordinates (0, 0).

The code is a function that calculates the distance between two points on the screen.

The function starts by defining two variables, **point** and **self.hand\_result.landmark** which is used to store the coordinates of each point in x and y-coordinates respectively.

Next, it defines a variable **dist** which stores the distance between each point in meters.

Then, it checks if there is a landmark at both points using **landmark[point[0]].y < landmark[point]**. If so, then **sign = 1**; otherwise, **sign = 0**.

The next step is to calculate the distance between each point in meters with **dist = math.sqrt(dist)** where dist stores the square root of dist. The code is trying to find the gesture encoding of a point on the map.

The code starts by checking if there is already an existing landmark at that point, and if so it returns its **z coordinate**.

If not, then it creates a **new landmark** at that point and stores its coordinates in **self.hand\_result**.

The next line calculates the distance between two landmarks using their x and y coordinates as well as their z coordinate: **def get\_dz(self,point): return abs(self.hand\_result.landmark[point[0]].z - self .hand\_result. land mark [point[1]].z)** This function takes in two points (the first being the current finger state's landmark), subtracts one from another, and returns this difference as an absolute value with respect to Z-axis (in other words how far away they are).

Next comes **set\_finger\_state()**, which sets up variables for when you want to use your fingers on the screen or not based on whether there is any data stored in **self.hand\_result** yet: **if self . hand \_ result == None : return**

The code is the function that will be used to find the gesture encoding.

The function **set\_finger\_state()** will be called in order to get the current finger state of the user. If it is not None, then it means that there was a previous gesture encoding. The code starts by declaring the variables that will be used in the program. The first variable is **self**, which is a reference to the current object being executed.

The second variable is **finger**, which stores a value from 0 to 5 representing how many fingers are on each hand. The next line of code declares an enumerate function that iterates through all the points in a list and assigns them to **idx** and **point** respectively.

Next, it calculates two signed distances using Pythagorean theorem: **dist1 = sqrt(self.finger)** and **dist2 = sqrt(self.finger)**.

It then uses these values as ratios for assigning 1 or 0 depending on whether they are greater than or less than .5 respectively (ratio > 0.5). Finally, it sets self's finger variable based on this ratio calculation with either one or zero added at the end (self >> 1 if ratio > 0.5; self << 1 if ratio < 0.5).

The code attempts to calculate the ratio of the distance between two points on a line, which can be used to determine if it is closer to one point or another.

The code calculates this ratio by dividing the distance between two points by their difference in distances. The first part of the code sets up an enumerate loop that iterates through all of the points on a line and assigns them values for **idx,point** in `enumerate(points): 0 #thumb dist = self.get_signed_dist(point[:2]) dist2 = self.get_signed_dist(point[1:])` . The code is trying to detect if the user has their hand in a pinch gesture.

If they do, it will change the gesture to a pinch-minor gesture.

Otherwise, it will keep the current gesture as a pinch-major one.

The code is checking for fingers that are either last 3 or last 4 and then checks if the distance between them is less than 0.05 meters (5 centimeters).

The code is the code that makes use of a switch statement to decide which gesture to make and handles any fluctuation in the input due to noise.

## • Class Controller

```
class Controller:
    tx_old = 0
    ty_old = 0
    trial = True
    flag = False
    grabflag = False
    pinchmajorflag = False
    pinchminorflag = False
    pinchstartxcoord = None
    pinchstartycoord = None
```

```
pinchdirectionflag = None
prevpinchlv = 0
pinchlv = 0
framecount = 0
prev_hand = None
pinch_threshold = 0.3
```

The first variable is called "Controller" and it is a class that represents the controller of the game. Next, there are two variables for tracking how many times we have been hit with our hand (**tx\_old**) and how many times we have touched our opponent's screen (**ty\_old**).

These values start at 0 because they represent when the program starts. The next variable is called "**trial**" which tracks whether you are currently showing hands game or not. If trial equals True then **grabflag** will also equal True, otherwise **grabflag** will equal False.

Next, there are three flags: **flag**, **grabflag**, and **pinchmajorflag** which track if your character has grabbed their opponent's screen yet during this round of play; these flags start off as False because they represent when the program starts.

There are also two more flags: **pinchminorflag** and **pinchstartxcoord** which track where your character started to touch their opponent's screen on their left side; these values start out as None because they represent when the program starts. Lastly, there is one last flag called "pinchdirectionflag".

- The get pinchily calculates the distance between the controller's pinchstartxcoord and hand\_result.landmark[8].x coordinate, which is then converted to a value in pixels.
- The get pinchily calculates the distance between the controller's pinchstartycoord and hand\_result.landmark[8].y coordinate, which is then converted to a value in pixels.
- The scroll Vertical() function scrolls up or down depending on whether pinchlv>0.0 or not respectively.
- The scroll Horizontal() function scrolls left or right depending on whether pinchlv<0.0 or not respectively

- The function **get position** will return the position of the controller's hand, with respect to its previous position. It will first calculate the `delta_x` and `delta_y` for both hands. Then it calculates the ratio of a new hand's x coordinate to its old hand's x coordinate.
- The `pinch_control_init()` function initialize the pinch control that sets the x coordinate for where to start pinching, and then the second line sets the y coordinate. Afterwards it sets the initial coordinates of the controller's pinch gesture, controller's pre-pinch gesture and frame count.
- The function `handle controls ()`, check If the user performs a fist, then it will move the mouse cursor to where they are pointing and if they perform an index finger, then it will click on whatever button was pressed.
- The code is checking to see if the user is holding their hand in a two-finger closed position and then if they are, it will double click.

## • **Class GestureController**

The code starts by declaring a class called **GestureController**. This is the class that will be used to control the camera and gesture recognition. The next line declares a class variables `gc_mode`, `cap`, `CAM_HEIGHT`, `CAM_WIDTH` (variables are declared as `None` because they are not needed until we connect our first camera later in this program), `hr_major`, `hr_minor`(for right and left hand by default), `dom_hand` set to be `True`.

Next, `__init__()` method of the class is defined with `self` being assigned to it so that when you call an instance of this class from Python code, you can access its attributes like `self._cap` for example (which would be `cv2.VideoCapture(0)` if you were calling it from Python).

The next line of the code is used to initialize the `GestureController` class. The first line of code sets the mode for the gesture controller to 0, which means that it will not be capturing any video frames. The next line of code initializes a variable called `cap` with a reference to `cv2.VideoCapture()`.



The **VideoCapture()** function returns a reference to an open video capture device. This captures all frames from the camera and stores them in memory until we close it or call **stop ()** on it.

If you want to use this as your main camera, then you can set up your own instance of **cv2.VideoCapture()** and pass in its address as the argument when creating your object (see below).

The next function is **classify\_hands()**. The code starts by defining the variables **left** and **right**. These are used to store the results of classification for each hand. The code then defines two dictionaries, one for each hand. Each dictionary has an entry with a label that is either **Right** or **Left** depending on which hand was classified as such.

The next line in the code starts by using **MessageToDict()** to convert the **multi\_handedness** list into dictionaries with labels corresponding to their respective hands.

If **GestureController** is set up so that it can recognize both hands, then **hr\_major** will be set equal to **right** and **hr\_minor** will be set equal to **left**. Otherwise, if only one hand is recognized, then **hr\_major** will be set equal to **left** and **hr\_minor** will be set equal to **right**.

Next comes **start()**, which begins analyzing data from gestures made by users in order to classify them according. The code is meant to classify the hands of a user. The code will first classify the left and right hands, then determine which hand is dominant based on the classification that was returned.

The code starts by creating a new image with the **cv2.cvtColor** function and then using the **flip ()** method to change it from **RGB** to **BGR**. Next, we create a hand variable and assign an empty list to it as well so that we can wait until there is something in this list before continuing on with our program as well.

We also set both lists' flags so they are not saved at all because if they were saved, then when we process them later, they would still be loaded up from memory instead of being processed again like what happens when you run your program multiple times without

saving anything first (which would cause errors). The code is a snippet of code that attempts to classify the hands in an image.

The snippet will first check if there are any multi-hand landmarks present in the image and then process the hand landmark results accordingly.

If so, it proceeds to call **GestureController.classify\_hands()** with all the results from **HandsMultiLandmark()** as input. This function returns two values, one for each hand's result, which will be stored into **HandMajor** and **HandMinor** respectively.

## Working

Following are the steps in working on our project:

- 1) Capturing real time video using Web-Camera: We will need a sensor to detect the user's hand movements in order for the system to work. As a sensor, the computer's webcam is used. The webcam records real-time video at a fixed frame rate and resolution determined by the camera's hardware. If necessary, the system's frame rate and resolution can be changed.
- 2) Converting the video captured into HSV format: The video has also been converted into HSV (hue, saturation, meaning, also called HSB), an alternative representation of the RGB color model created by computer graphics researchers to better reflect the perception of colored characteristics by human vision.

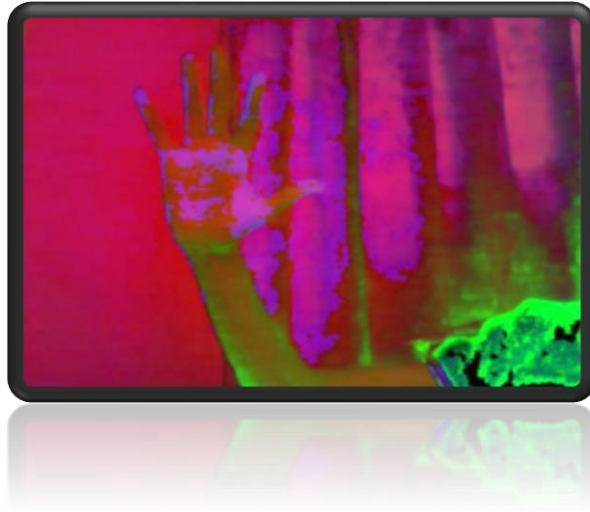


Fig 3.5: HSV Image

- 3) Each image frame is processed separately: Following the capture of the video, it goes through a brief pre- processing stage before being processed one frame at a time.
- 4) Conversion of each frame to a greyscale image: A grayscale image has lower computational complexity than a coloured image. It also aids in faster colour calibration without the use of external noise. All the necessary operations were carried out after the image was converted to grayscale.
- 5) Calibrate the color ranges: The device enters the calibration mode after the above steps, which assigns each colour according to the HSV rule to its colour hue, saturation or value values. Every colour already has its predetermined values. For accurate colour detection, the user can adjust the ranges. In the diagram below you can clearly see the variety of values used to detect every colour cap.
- 6) Calculate the image's centroid by locating the image's region. To guide the mouse pointer, the user must first choose a point whose coordinates can be sent to the cursor. The device can monitor cursor movement using these coordinates. As the object travels around the frame, these coordinates change over time

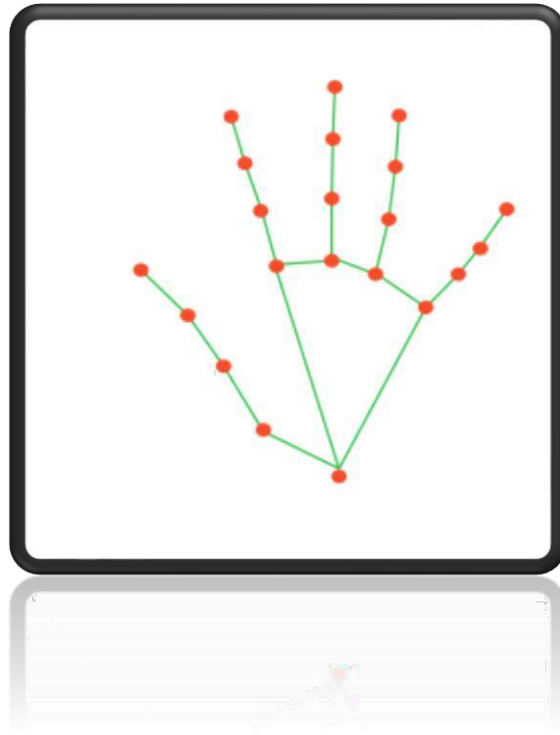


Fig 3.6: Co-ordinates

- 7) Tracking the mouse pointer. After determining the coordinates, the mouse driver is accessed, and the coordinates are sent to the cursor. The cursor positions itself in the required position using these coordinates. As a result, the mouse moves proportionally across the screen as the user moves his hands across the camera's field of view.
- 8) Simulating the mouse actions. To create the control actions in simulation mode, the user must make hand gestures. The computation time is reduced due to the use of colour pointers

## Testing And Result Analysis

### Overview

In order to achieve accuracy, and consistency of the Virtual Mouse colour recognition, testing phases have been conducted on various scenarios. The purpose of the 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 be able to recognize the colours input provided by the users with minimal adjustment, providing 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.

### Recognition on results

**Gesture Recognition:** Following are the results of various hand gestures that result into mouse operations:

- **Neutral Gesture:** Palm: Used to halt/stop execution of current gesture.



Fig 4.1:Neutral Gesture

- **Move Cursor**

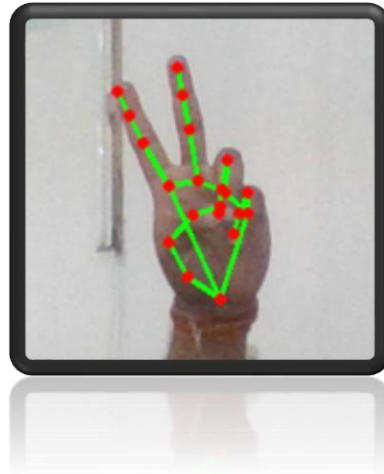


Fig 4.2: Move Curser Gesture

Cursor is assigned to the midpoint of index and middle fingertips. This gesture moves the cursor to the desired location. Speed of the cursor movement is proportional to the speed of hand

- **Left Click**



Fig 4.3: Left Click Gesture

- **Right Click**

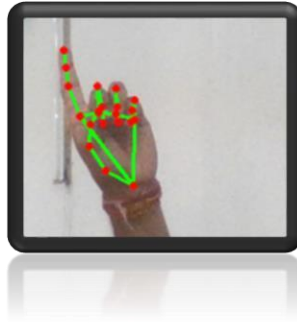


Fig 4.4: Right Click Gesture

- **Double Click**

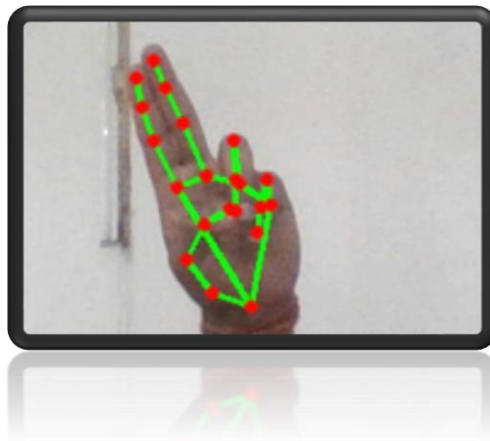


Fig 4.5: Double Click Gesture

- **Scrolling**



Fig 4.6: Scrolling Gesture

Dynamic Gestures for horizontal and vertical scroll. The speed of scroll is proportional to the distance moved by pinch gesture from start point. Vertical and Horizontal scrolls are controlled by vertical and horizontal pinch movements respectively.

The below two are pinch gestures similar to scrolling gestures:

### **Volume Control**

Dynamic Gestures for Volume control. The rate of increase/decrease of volume is proportional to the distance moved by pinch gesture from start point.





Fig 4.7: Volume Control gesture

### **Brightness Control**



Fig 4.8: Brightness Control gesture

Dynamic Gestures for Brightness control. The rate of increase/decrease of brightness is proportional to the distance moved by the pinch gesture from the start point.

## Drag and Drop

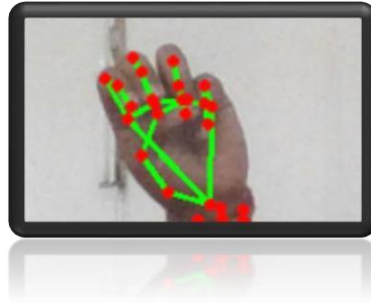


Fig 4.9: Drag And Drop Gesture

Gesture for drag and drop functionality. Can be used to move/transfer files from one directory to other.

## Multiple Item Selection



Fig 4.10: Multiple Item Selection Gesture

## **Conclusion and Future work**

### **Overview**

In conclusion, it is no surprise that the physical mouse will be replaced by a virtual non-physical mouse in the Human-Computer Interactions (HCI), where every mouse movement can be executed with a swift of your fingers everywhere and anytime without any environmental restrictions. This project had developed 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, and combinations, and translate them into actual mouse functions. Due to accuracy and efficiency playing a significant role in making the program as useful as an actual physical mouse, a few techniques had to be implemented. Primarily, the coordinates of the colours that are in charge of handling the cursor movements are averaged based on a collection 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 error.

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.

A Web camera is running on the mouse cursor. This will also lead to new levels of human-computer interaction (HCI), which does not require physical contact with the device. This machine can perform all mouse tasks centered on color recognition. This device is capable of being useful for interacting with contactless input modes. For people who don't use a touchpad, it's helpful. The architecture of the device proposed would dramatically change people's interactions with computers. Everyone is compatible with the Webcam, the microphone, and the mouse. It would eliminate the need for a mouse

completely. It can also be used in gaming or any other independent application. Free movement, left-click, right-click, drag/select, scroll-up, and scroll-down are all operations that can be performed using only gestures in this Multi-Functional system. The majority of the applications necessitate additional hardware, which can be quite costly. The goal was to develop this technology as cheaply as possible while also using a standardized operating system. Various application programs can be written specifically for this technology in order to create a wide range of applications with minimal resources.

Most of the applications require additional hardware which is often expensive. The motive of this work is to create this technology as cheaply as possible and to create it under a standardized operating system as well. Though our system can be used as an alternative to a physical keyboard and mouse, it still may perform less accurately in low light conditions. This is a concern for further research. Moreover, the work can be extended to a wide variety of environments and can be tested using sophisticated existing models. .

## **Literature perspective of future work**

The mouse actually forms an integral part of the computer system. Our system architecture can facilitate the use of computers for paralyzed people. We have developed a virtual system where people can communicate with the computer without using a physical mouse. This could lead to a new age of Human-Computer Interaction in which physical contact with the computer would not be necessary at all. The use of object detection and image processing in OpenCV for the implementation of our work has proved to be practically successful and the task of the mouse is achieved with good precision. This system can be beneficial to certain people who have no control over their limbs.

- The present application though seems to be feasible and more user friendly.
- An attempt to make the input modes less constraints dependent for the users hand gestures has been preferred.

- Another important aspect for the related development could be design of an independent gesture vocabulary framework.
- The colour detection algorithm can cause detection problem if another coloured rubber in working domain of webcam.

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 being limited to a 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 heavily relies 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 the webcam. Therefore, the program may have better performance when it is running on a decent machine with a webcam that performs better in different types of lighting.

## **Advantages**

- The main advantage of using hand gestures is to interact with the computer OS a non-contact human-computer input modality.
- Reduce hardware costs by eliminating the use of the mouse.
- Convenient for users not comfortable with the touchpad.
- The framework may be useful for controlling different types of games and other applications dependent on the control through user-defined gestures.
- We are developing a system to control the mouse cursor using a real-time camera.

- This system is based on computer vision algorithms and can do all mouse tasks.
- However, it is difficult to get stable results because of the variety of lighting and skin colors of human races.
- This system could be useful in presentations and to reduce work space.
- Features such as enlarging and shrinking windows, closing window, etc. by using the palm and multiple fingers.

## References

Angel, Neethu.P.S,”Real Time Static & Dynamic Hand Gesture Recognition”, International Journal of Scientific & Engineering Research Volume 4, Issue3, March-2013.

Chen-Chiung Hsieh and Dung-Hua Liou,” A Real Time Hand Gesture Recognition System Using Motion History Image”icsps, 2010

Hojoon Park, “A Method for Controlling the Mouse Movement using a Real Time Camera”, Brown University, Providence, RI, USA, Department of computer science, 2008.

Erdem, E. Yardimci, Y. Atalay, V. Cetin, A. E. “Computer vision based mouse”, Acoustics, Speech, and Signal Processing, 2002. Proceedings. (ICASS). IEEE International Conference

Park, H. (2008). A method for controlling mouse movement

using a real-time camera. Brown University, Providence, RI,

USA, Department of computer science.

Kumar N, M. (2011). Manual Testing: Agile software development. [online]

Manojforqa.blogspot.com. Available at:

<http://manojforqa.blogspot.com/2011/09/agile-software->

[development.html](http://manojforqa.blogspot.com/2011/09/agile-software-development.html) [Accessed 27 Aug. 2015].

Niyazi, K. (2012). Mouse Simulation Using Two Coloured Tapes. IJIST, 2(2), pp.57-63.

Sekeroglu, K. (2010). Virtual Mouse Using a Webcam. [online] Available at:

[http://www.ece.lsu.edu/ip1/SampleStudentProjects/ProjectKazim/Virtual%20Mouse%](http://www.ece.lsu.edu/ip1/SampleStudentProjects/ProjectKazim/Virtual%20Mouse%20Using%20a%20Webcam_Kazim_Sekeroglu.pdf)

[20Using%20a%20Webcam\\_Kazim\\_Sekeroglu.pdf](http://www.ece.lsu.edu/ip1/SampleStudentProjects/ProjectKazim/Virtual%20Mouse%20Using%20a%20Webcam_Kazim_Sekeroglu.pdf) [Accessed 29 Aug. 2015].