GESTURE CONTROLLED MOUSE POINTER

A PROJECT REPORT

Submitted by

Ayush Roy [Reg No: RA1611008010641]
Saumya Awasthi [Reg No: RA1611008010414]
Ansh Vinod Motwani [Reg No: RA1611008010661]

Under the Guidance of

Ms. S. Deepanjali

(Assistant Professor, Department of Information Technology)

In partial fulfillment of the Requirements for the Degree of

BACHELOR OF TECHNOLOGY

DEPARTMENT OF INFORMATION TECHNOLOGY

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR

APRIL 2020

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY KATTANKULATHUR-603203

BONAFIDE CERTIFICATE

Certified that this project report titled "**Gesture Controlled Mouse Pointer"** is the bonafide work of **"Ansh Vinod Motwani [Reg No: RA1611008010661], Ayush Roy [Reg No: RA1611008010641] and Saumya Awasthi [Reg No: RA1611008010414]** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion for this or any other candidate.

Ms. S. Deepanjali
GUIDE
Assistant Professor
Technology Dept. of Information Technology

Dr. G. Vadivu
HEAD OF THE DEPARTMENT
Dept. of Information

Signature of Internal Examiner

Signature of External Examiner

ABSTRACT

Our project deals with controlling the mouse pointer and carrying out the functions of a trackpad with the help of hand gestures. Moving our hand would allow us to change tabs, double click an item, drag objects and carry out other simple tasks. In existing research, an external camera is used for better image capture and detection but that makes is uneasy to use in everyday life. Using an external camera also requires extra battery usage and has an added hardware cost. Since our software will be using the computer webcam, there would not be any extra cost. Our software can make everyday life very easy. If the computer is kept at a distance, the person would not have to get up but instead can interact with the computer by just gesturing with their hands. It could greatly help partially paralyzed or bedridden people who cannot walk or get up to use their computer kept at a distance. There would no longer be any need to keep laptops on the bed while using it as it is harmful to keep the laptop on bed. This problem faces lots of issues as mostly those solutions cater to only a few gestures and for the remaining gestures there is a need to use the mousepad. But we plan on making a complete replacement for a mouse which will not require any mouse input. The one thing which would be unique about our project would be its precision. We will design it in such a way that the user can easily point to buttons and select anything on the screen without having trouble in moving the pointer to a very small distance. Our pointer will also be having mouse assist which will be helping the user to effortlessly maneuver the pointer. We aim to create a robust software which could be used in everyday life and does not remain as just a project. It would be as user friendly as possible and we will make it in a way that people can use it.

ACKNOWLEDGMENT

The success and the final outcome of this project required guidance and assistance from different sources, and we feel extremely fortunate to have got this all along with the completion of our project. We express our sincere thanks to the Head of the Department, Department of Information Technology, **Dr. G. Vadivu**, for all the help and infrastructure provided us to complete this project successfully. We owe our profound gratitude to our project guide **Ms. S. Deepanjali**, who took a keen interest in our project work and guided us all along, till the completion of our project work by providing all the necessary information for developing a good system. We are thankful and fortunate enough to get constant encouragement, support and guidance from all the teaching staff of the Department of Information Technology who helped us in successfully completing our major project. Also, we would like to extend our sincere regards to all the non-teaching staff of the Department of Information Technology for their timely support.

Ansh Vinod Motwani [Reg No: RA1611008010661]

Ayush Roy [Reg No: RA1611008010641]

Saumya Awasthi [Reg No: RA1611008010414]

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

HCI	Human Computer Interaction
OpenCV	Open Source Computer Vision Library
GUI	Graphical User Interface
FPS	Frames per second
IDE	Integrated Development Environment
RGB	Red Green Blue
CNN	Convolutional Neural Network
DNN	Deep Neural Network
ROI	Region of Interest

CHAPTER 1
INTRODUCTION

1.1 Overview

PC innovation has massively grown over the previous decade and has turned into a vital piece of ordinary life. The essential PC hardware for Human Computer Interaction (HCI) is the mouse. The mouse isn't appropriate for HCI in some genuine circumstances, for example, with Human Robot Interaction (HRI). There have been numerous explores on elective strategies to the PC mouse for HCI. The most characteristic and natural strategy for HCI, that is a reasonable swap for the PC mouse is with the utilization of hand motions. This task is subsequently gone for examining and building up a Computer Control (CC) framework utilizing hand motions. Most workstations today are outfitted with webcams, which have as of late been utilized instability applications using face acknowledgment. So as to tackle the maximum capacity of a webcam, it tends to be utilized for vision-based CC, which would adequately dispose of the requirement for a PC mouse or mouse cushion. The handiness of a webcam can likewise be extraordinarily stretched out to other HCI application, for example, a communication via gestures database or movement controller. HCI utilizing hand motions is instinctive and powerful for balanced connection with PCs and it gives a Natural User Interface (NUI). There has been broad research towards novel gadgets and strategies for cursor control utilizing hand signals. Other than HCI, hand motion acknowledgment is additionally utilized in communication via gestures acknowledgment, which makes hand motion acknowledgment much increasingly noteworthy. Human Computer Interface mainly focus on the development of an efficient and easy to use interfaces. The personal computers have a variety of options to interact with different applications efficiently with the use of mouse, track-pad, Joystick etc. Nowadays touchscreen technology is available for devices like mobile phones. But this technology is still costly when used in the personal computers. And the devices currently used to

interact are frequently breakable. Our objective was to make an alternative technology to interact with the computer which not as costly as the touchscreen technology and is easy to operate with.

1.2 Problem Statement

For personal use in computers, we have been using the mouse for a long time. Using the mouse has become obsolete since touchpads and touch screens have been introduced. Now it is time to use the most natural way of communication which is through gesturing the computer. Our software is targeted to people who have very limited physical movement like bedridden people. It could also be used by people using their personal computers in flights or in a car.

1.3 Goals of the System

- Capability to control the mouse without physically touching the system.
- A system that ensures vision-based machine.
- Capability of controlling mouse functions using hand gestures.
- To eliminate the limitations of stationary hand.
- To provide real time gesture-based tracking system.

CHAPTER 2

LITERATURE STUDY

2.1 Mouse Cursor Control System Based on Hand Gesture

Authors: Horatiu-Stefan Grif, Cornel Cristian Farcas

The paper presents a human computer interaction interface for mouse cursor control. A difference between colour levels between different fingers was used to differentiate different gestures. Along with this the hand was moved on the plain surface and its movement was captured by the external webcam. The hand gesture was recognized by the combination of colours detected. The system showed expected output even in low lighting conditions. The increased illumination can cause an effect on the detection of colour strips.

Pros	Cons
1. Works well in low lighting. 2. It gives better precision than other similar applications.	1. Use of external webcam. 2. Requires a stable background. 3. The gesture recognition task is influenced by the rotation of the hand on the plane of the pad.

2.2 Controlling Mouse Pointer Using Web Cam

Authors: Gaurav Sahu, Sonam Mittal

This paper deals with the management and enhancement of human interaction with the digital world. This paper has proposed a way to control the mouse pointer using webcam. Here different colour strips are used on different fingers. Just the colour strip part is extracted from the image captured on basis of pre-defined colour information. The mouse tracking is done based on its movement. The different mouse pointer actions are performed based on the mappings of different colour strips for each action.

Pros	Cons
1. All basic mouse functions can easily be carried out. 2. It can be applied to areas of Augmented Reality, Computer Gaming etc.	1. Use of colour strips instead of actual finger detection. 2. Calculations used are complex which makes the system difficult to understand and extend forward.

2.3 Finger Gesture Control Computer Mouse with Image Processing

Author: Heera Lal Bhadrecha

In this paper, the mouse functions like left click, right click are carried out. The motion of the fingers along with skin detection algorithm is used. Here the binary image of the hand is extracted from the captured image. Then the centroid and the extreme points are found out along with the angle made at the centroid. And based on these angles, the mouse cursor is controlled and different click events are been performed.

Pros	Cons
1. Cost Effective 2. Vast range of applications.	1. Limited functionality. 2. Output is fluctuated due to change in lighting. 3. Pc requires high computations capacity to run the complex functions.

2.4 Human hand gesture-based system for mouse cursor control

Authors: Horatiu-Stefan Grif, Trian Turc

This paper intends to improve hand postures in gesture based HCI systems. They have used the hand angle to determine different hand gestures. These hand angles are been captured by the external web-camera which is focused on the blue background on which the hand moves. This is comfortable for the user because the most user friendly and appropriate hand gesture is detected. The system has very good results under both low as well as high level lighting.

Pros	Cons
1. Uses natural and more relaxed postures. 2. No additional noise removal filter is used.	1. Works well only in well-lit environments. 2. A single colour background is required to detect the gestures.

2.5 Simulation of Mouse using Image Processing Via Convex Hull Method

Authors: Ahemad Siddique, Abhishek Kommera ,Divya Varma

This paper uses the convex hull method for gesture detection. Using this method, gesture detection becomes very easy. The frame of reference is extracted from the captured image after which the finger tips are found. Based on these finger points and the area covered by a gesture, different hand gestures are found out.

Pros	Cons
It uses Convex Hull Algorithm, which makes hand detection easy.	1. Heavy Software. 2. Requires latest technology for smooth operation.

2.6 Design and Development of Hand Gesture Based Virtual Mouse

Authors: Kabid Hassan Shibly, Samrat Kumar Dey, Md. Aminul Islam, Shahriar Iftekhar Showrav

In this paper virtual mouse computer using HCI has been implemented. Gestures are detected using an external webcam and processed using colour segmentation. The user will be allowed to control some of the computer cursor functions with their hands which bear coloured caps on fingertips. This system removes the device dependency in using the mouse as the external mouse will not be used.

Pros	Cons
1. Most mouse functions can be carried out. 2. It can be used for patients with very limited limb control. 3. It is also usable in sign language.	1. External camera is required for the software to correctly detect the gestures. 2. The background highly influences the output generated. 3. Colour strips are used to detect different fingers.

2.7 Gesture Based Computing as an Alternative to Mouse by Calibrating Principal Contour Process Actions

Authors: Chinnu Thomas, D. Lakshmi

In this paper, gesture-based computing is carried out instead of using the mouse by calibrating the Principal Contour Process actions. It builds a rich bridge between users and computer than previous primitive text user interface. This is required as we are looking for better improved methods to interact with our PC's. The traditional methods are becoming obsolete and touch will no longer be used in a few years from now.

Pros	Cons
1. High Accuracy. 2. Recognition rate – 90.45%	1. Use of external gloves for hand detection. 2. Time consuming. 3. Confusion between gestures.

2.8 Virtual Mouse Using Hand Gesture

Authors: Abhilash S, Lisho Thomas, Naveen Wilson, Chaithanya C

In this paper, vision-based cursor control system is carried out to transfer files between 2 different computers on the same network. It uses color-based gesture detection to detect the shape and size of the gesture made. Since hand gesture is the most effortless and natural method of communication, the same is used in this paper. It uses python for all its coding ad uses the OpenCV library for the computer vision libraries.

Pros	Cons
1. Its main aim is to focus on humans that don't have control of their limbs.	1. Very limited functionalities. 2. Lighting affects the results.

2.9 Hand Recognition and Gesture Control Using a Laptop Web-camera

Authors: Zi Xian, Justin Yeo

Hand gesture recognition is increasingly becoming very common these days especially in the field of Augmented Reality. It is one key aspect to Human Computer Interface, allowing for two-way interaction in virtual spaces. But many of these actions of augmented reality come at a very high price. With this method, Augmented Reality could be carried out using a very cost effective method. Here the edge detection is done on the captured image, then the background is elimination algorithm is applied to it and the threshold of resultant is calculated and from the resultant hand image the different hand gestures are been recognized.

Pros	Cons
1. No external equipment. 2. Use of canny edge detection.	1. Limited functionalities. 2. Low Accuracy. 3. Limited research performed.

2.10 Efficient Fingertip Tracking and Mouse Pointer Control for a Human Mouse

Authors: Jiyoung Park, Juneho Yi

This paper works on visually recognizing hand gestures. The method of fingertip tracking is used for the gesture detection. This method is based on the Camshift algorithm which tracks the particular hand poses. The location of the fingertip is linked to a point on the monitor screen. This method works in realtime while not occupying a lot of system memory. It ensures that other applications can run smoothly run alongside it.

Pros	Cons
1. Based on CAMSHIFT. 2. Easily able to follow the movement of hand.	1. Research and Development is expensive and consumes a lot of data.

CHAPTER 3

PROPOSED WORK

3.1 Dataset Gathering

To train the system, a dataset is required comprising of testing and training data. The dataset used in this project is self-generated. The testing dataset comprises of 6 sets of 1000 images each in which each set denotes a different gesture. The training data comprises of 6 sets of 100 images each in which each set denotes a different gesture. For capturing the data, the camera clicks the image in a loop depending on which type of image it is. Every image gets automatically numbered based on the iteration value.

The hand images are identified by background elimination technique. First a still background is chosen. And after a few frames, any movement on the screen is considered as the foreground images. The largest foreground image contour is considered as the main hand image so that any small movement does not create any external noise. The hand contour is converted to white colour and the entire background is made black in colour.

Figure 3.1.1 Background eliminated image

Once the image is converted to the required format, a keypress begins the recording of the images. One image is recorded in every iteration. Once the required number of images is reached, the loop automatically ends storing every image captured in the hard drive of the local machine.

```
def main():
    aWeight = 0.5
    camera = cv2.VideoCapture(0)
    top, right, bottom, left = 100, 400, 300, 600
    num_frames = 0
    image_num = 0
    start_recording = False

    while(True):
        (grabbed, frame) = camera.read()
        if (grabbed == True):
            frame = imutils.resize(frame, width=700)
            frame = cv2.flip(frame, 1)
            clone = frame.copy()
            (height, width) = frame.shape[:2]
            roi = frame[top:bottom, right:left]
            gray = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)
            gray = cv2.GaussianBlur(gray, (7, 7), 0)

            if num_frames < 30:
                run_avg(gray, aWeight)
                print(num_frames)
            else:
                hand = segment(gray)
                if hand is not None:
                    (thresholded, segmented) = hand
                    cv2.drawContours(clone, [segmented + (right, top)], -1, (0, 0, 255))
                    if start_recording:
                        cv2.imwrite("Dataset/YoImages/yo_" + str(image_num) + '.png', thresholded)
                        image_num += 1
                    cv2.imshow("Thesholded", thresholded)
            cv2.rectangle(clone, (left, top), (right, bottom), (0, 255, 0), 2)
            num_frames += 1
            cv2.imshow("Video Feed", clone)
            keypress = cv2.waitKey(1) & 0xFF
            print(image_num)

            if keypress == ord("q") or image_num > 999:
                break
            if keypress == ord("s"):
                start_recording = True

        else:
            print("Error, Check Camera")
            break

main()
#camera.release()
cv2.destroyAllWindows()
```

Figure 3.1.2 Code for dataset generation

Two additional functions help in background elimination and detecting the required contour of the image. This helps in reducing the noise in the images and provide better image quality.

```python
import camera as camera
import cv2
import imutils
import numpy as np

bg = None

def run_avg(image, aWeight):
    global bg
    if bg is None:
        bg = image.copy().astype("float")
        return
    cv2.accumulateWeighted(image, bg, aWeight)

def segment(image, threshold=25):
    global bg
    diff = cv2.absdiff(bg.astype("uint8"), image)
    thresholded = cv2.threshold(diff, threshold, 255, cv2.THRESH_BINARY)[1]
    cnts, hierarchy = cv2.findContours(thresholded.copy(), cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

    if len(cnts) == 0:
        return
    else:
        segmented = max(cnts, key=cv2.contourArea)
        return (thresholded, segmented)

```

Figure 3.1.3 Functions for dataset generation

3.2 Preprocessing Dataset

Once the dataset is generated, all the images must be resized to a fixed size for all the computations to take place. Same sized images are required for all the computations to take place. It eases the computation process if the size of the images is known. Every image is converted to 100 x 89 pixels and the previous image is overwritten with the new resized image.

```
from PIL import Image

def resizeImage(imageName):
    basewidth = 100
    img = Image.open(imageName)
    wpercent = (basewidth/float(img.size[0]))
    hsize = int((float(img.size[1])*float(wpercent)))
    img = img.resize((basewidth,89), Image.ANTIALIAS)
    img.save(imageName)

for i in range(0, 100):
    resizeImage("C:/Users/ayroy/Downloads/Gesture Controlled Mouse Pointer/Dataset/YoTest/yo_" + str(i) + '.png')
```

Figure 3.2 Code for resizing images

3.3 Model Training using Neural Network

In Machine Learning, Deep Neural Networks (DNN) is a supervised learning algorithm where the machine recurrently performs the same task on every element of the sequence of where the output of each task is dependent on every previous calculation. This method learns in a sequence called feature hierarchy. Where the features on the top of the hierarchy are computed by the help of features at the bottom of the hierarchy. DNN's are formulated of multiple layers. The computations in each layer are hidden hence the layers are termed as "hidden layers ". Epoch is a parameter that determines the number of times the algorithm will work on the entire training dataset. In this case, the Epoch is set to 50 meaning the system will repeat the same process 50 times.

```
convnet = input_data(shape=[None, 89, 100, 1], name='input')
convnet = conv_2d(convnet, 32, 2, activation='relu')
convnet = max_pool_2d(convnet, 2)
convnet = conv_2d(convnet, 64, 2, activation='relu')
convnet = max_pool_2d(convnet, 2)
convnet = conv_2d(convnet, 128, 2, activation='relu')
convnet = max_pool_2d(convnet, 2)
convnet = conv_2d(convnet, 256, 2, activation='relu')
convnet = max_pool_2d(convnet, 2)
convnet = conv_2d(convnet, 256, 2, activation='relu')
convnet = max_pool_2d(convnet, 2)
convnet = conv_2d(convnet, 128, 2, activation='relu')
convnet = max_pool_2d(convnet, 2)
convnet = conv_2d(convnet, 64, 2, activation='relu')
convnet = max_pool_2d(convnet, 2)
convnet = fully_connected(convnet, 1000, activation='relu')
convnet = dropout(convnet, 0.75)
convnet = fully_connected(convnet, 6, activation='softmax')
convnet = regression(convnet, optimizer='adam', learning_rate=0.001, loss='categorical_crossentropy', name='regression')
model = tflearn.DNN(convnet, tensorboard_verbose=0)
loadedImages, outputVectors = shuffle(loadedImages, outputVectors, random_state=0)
model.fit(loadedImages, outputVectors, n_epoch=50, validation_set=(testImages, testLabels),snapshot_step=100, show_metric=True, run_id='conv
model.save("TrainedNewModel/GestureRecogModel.tfl")
```

Figure 3.3 Code for convolutional neural networks

CHAPTER 4

IMPLEMENTATION

To run the program, the Main.py file is executed. It executes the following processes in order.

- Capture the real time video.
- Extract the individual frames.
- Resize the frame.
- Flip the frames.
- Get the ROI (Region of Interest).
- Convert the ROI to grayscale.
- Blur the ROI.
- Eliminate the background.
- Predict the gesture.
- Draw the contours and calculate the center point.
- Track pointer according to the centroid movement.
- Perform different actions with the different contours extracted.

4.1 Capturing the real time video

For the system to sense the movement of hand we need a sensor. We used the web camera in our work for that sensing. The web camera captures the real time video that is it gives us the real time news feed. The captured image is at a particular resolution and at a particular frame rate. We can change the frame rate and resolution according to our needs. The web camera captures the real time video. This is divided into different frames. These frames are then processed further.

Figure 4.1 Actual image captured by webcam

4.2 Extract the individual frames.

In this step, we extract every frame captured by the camera. Since the camera takes the input in the form of a video, we need to convert that video into multiple individual frames. This is necessary because all the computations are done on individual frames and not on the entire video. Once every frame is extracted, the images can be processed.

```
52      camera = cv2.VideoCapture(0)
53      top, right, bottom, left = 110, 350, 325, 590
54      num_frames = 0
55      start_recording = False
56      n=0
57      while (True):
58          (grabbed, frame) = camera.read()
59          frame = imutils.resize(frame, width=700)
```

Figure 4.2 Code for extracting individual frames

4.3 Resizing the image

When the camera captures the real time feed and converts it into the frames then the frame obtained might be of different size. So, we need to resize the image to fit our need. Now this image is inverted. It is like the image we get when we stand in front of the mirror. So, if we move our coloured strip to right it will in turn move to left and so will our mouse pointer. This will create a lot of confusion in operating the mouse pointer. Hence, we need to flip each frame. This is done by vertically inverting the frame.

```
21  def resizeImage(imageName):
22      basewidth = 100
23      img = Image.open(imageName)
24      wpercent = (basewidth / float(img.size[0]))
25      hsize = int((float(img.size[1]) * float(wpercent)))
26      img = img.resize((basewidth, hsize), Image.ANTIALIAS)
27      img.save(imageName)
28
```

Figure 4.3 Function for resizing images

4.4 Flipping the Image

Now this image is inverted. It is like the image we get when we stand in front of the mirror. So, if we move our coloured strip to right it will in turn move to left and so will our mouse pointer. This will create a lot of confusion in operating the mouse pointer. Hence, we need to flip each frame. This is done by vertically inverting the frame.

Figure 4.4 Flipped actual image

4.5 Getting the Region of Interest

The Region of Interest is a small part of the entire frame which is the actual area where all the processing will be done. The remaining part of the frame is ignored. Only the image inside this region is processed. This is done to remove any unnecessary objects in the background which can cause noise.

Figure 4.5 Region of interest

4.6 Convert the Region of Interest to grayscale

The image is converted from an RGB image into a grayscale image so that the foreground and background images can be properly differentiated. This helps in easier computation as there is a clear distinction between the hand and the background. This also helps to remove any noise from the image.

```
64          roi = frame[top:bottom, right:left]
65          gray = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)
```

Figure 4.6 Code to convert from RGB to grayscale

4.7 Eliminate the Background

In this step, all the background images are removed and only the image of the hand is shown. This is important as based on this image, the different gestures made by the hand will be predicted. The hand is shown in white colour and the remaining background is shown as black colour.

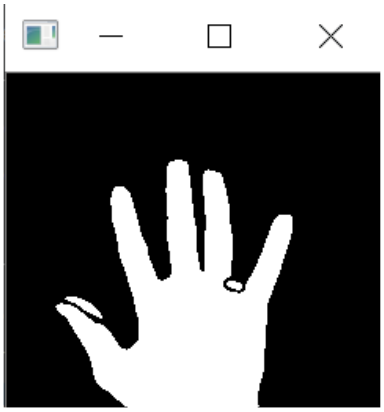

Figure 4.7 Background eliminated image

4.8 Predict the gesture

The image then passes through our convolutional neural network. The gesture is predicted on basis of how the neural network is trained previously. This is an important step as a major part of the accuracy of our system depends on it.

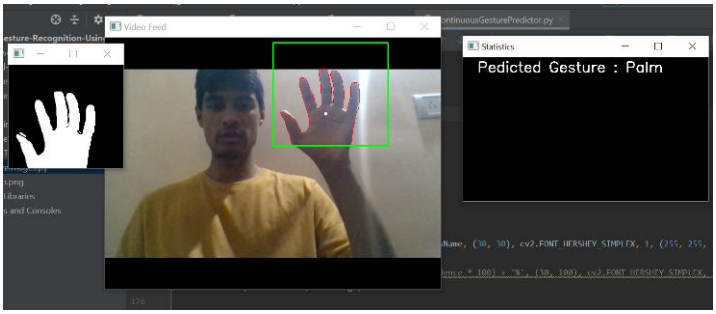

Figure 4.8.1 Palm gesture detected

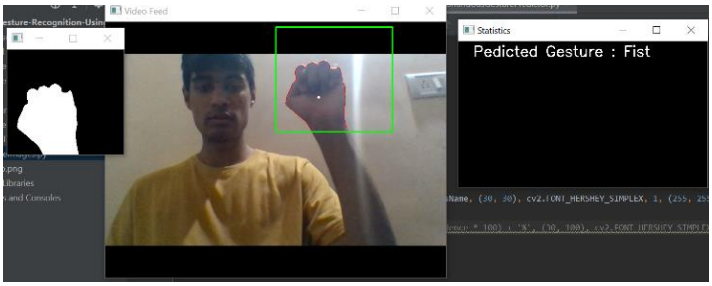

Figure 4.8.2 Fist gesture detected

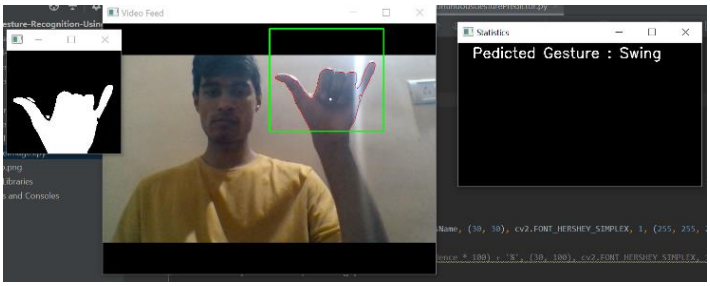

Figure 4.8.3 Swing gesture detected

4.9 Draw the contours and its centroid

The contours are drawn around the hand gesture. Contour is the outer line around a certain area. After this we calculate the centroid. It is the center point of the area surrounded by the contour. We control our mouse pointer movement with respect to the calculated centroid.

```
37  def segment(image, threshold=25):
38      global bg
39      diff = cv2.absdiff(bg.astype("uint8"), image)
40      thresholded = cv2.threshold(diff, threshold, 255, cv2.THRESH_BINARY)[1]
41      (cnts, _) = cv2.findContours(thresholded.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
42
43      if len(cnts) == 0:
44          return
45      else:
46          segmented = max(cnts, key=cv2.contourArea)
47          return (thresholded, segmented)
48
```

Figure 4.9 Code for drawing contours

4.10 Tracking and performing different functions

The tracking and different actions performed are done using the inbuilt library in python named PyAutoGUI. This library has many functions for left click, right click or just moving the pointer and many more.

The following are the different functions which are performed

1. Mouse Movement
2. Left Click
3. Right Click
4. Double Click
5. Scroll Up
6. Scroll Down

4.10.1 Mouse Movement

This function helps the mouse cursor to move in accordance with the movement of the hand while a fist is made. When the fist gesture is detected, the mouse movement mode is selected. When the mouse is kept still for more than 8 frames, then the mouse gets fixed in that position and any movement made by the hand does not make any changes. A specific gesture has to be made in order to make the mouse move again.

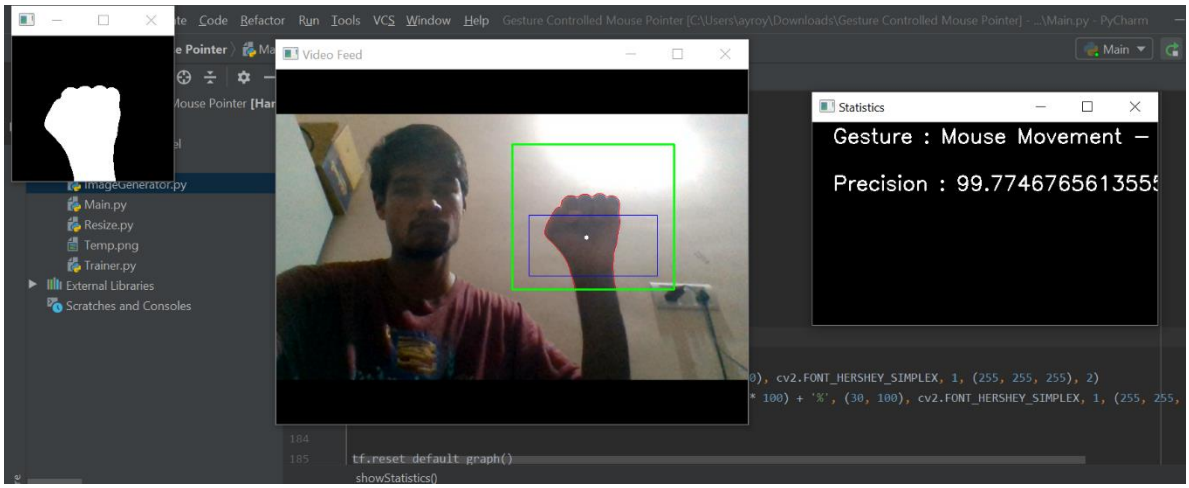

Figure 4.10.1 Mouse movement gesture

4.10.2 Left Click

The left click function is made when the system detects the index and middle finger raised. The left click gesture is run only once. After that a separate gesture needs to be made in order to run the left click function again.

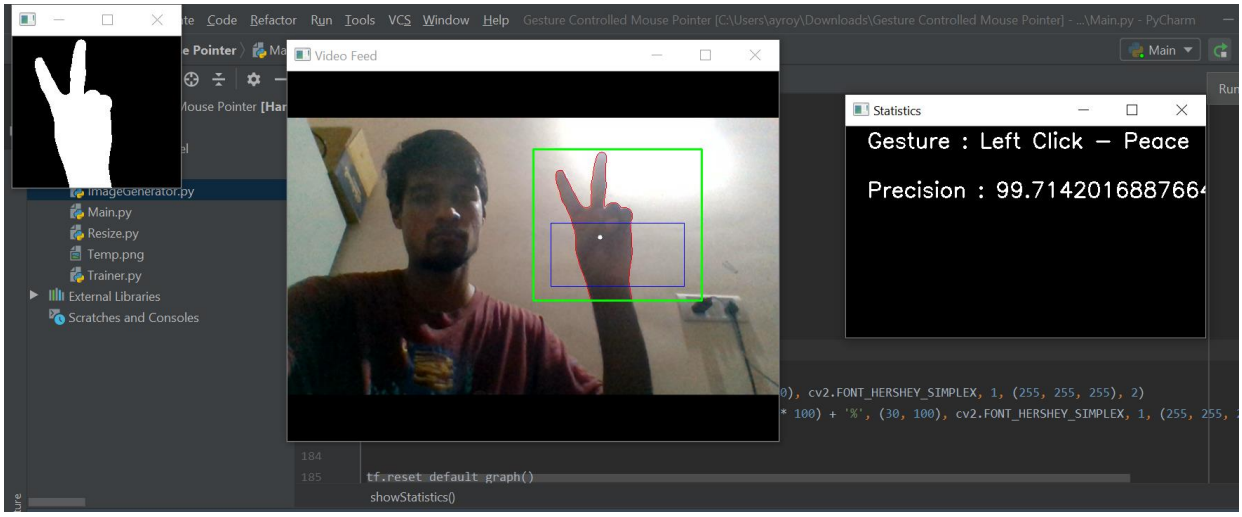

Figure 4.10.2 Left Click gesture

4.10.3 Right Click

The right click is made when the entire palm is shown. Similar to the left click function, this function also runs only once post which it awaits a separate gesture to be detected.

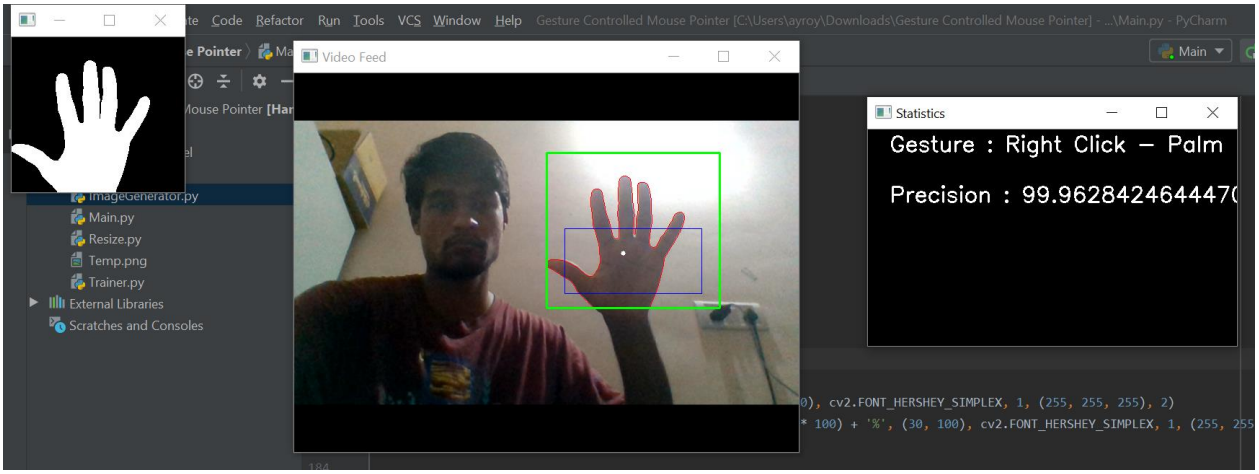

Figure 4.10.3 Right click gesture

4.10.4 Double Click

This gesture is detected when 3 fingers are raised up which trigger the double click function. The double click is the equivalent of 2 left click actions simultaneously. But PyAutoGUI has an inbuilt function for double click.

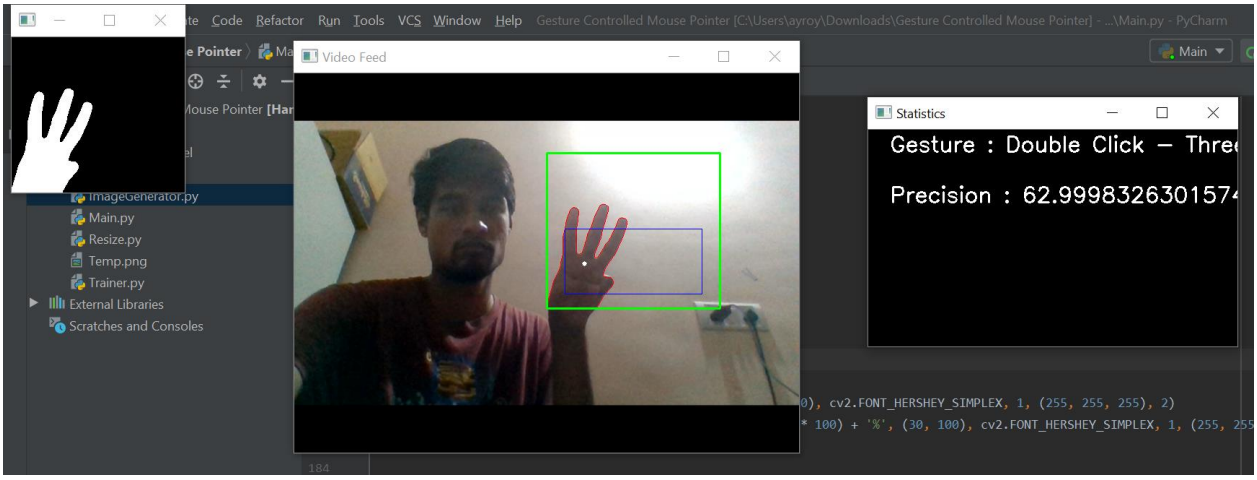

Figure 4.10.4 Double click gesture

4.10.5 Scroll Up

This function is run when the thumb, index and little finger are put up together. It runs repeatedly as long as the gesture is made. It scrolls the current screen up 10 units.

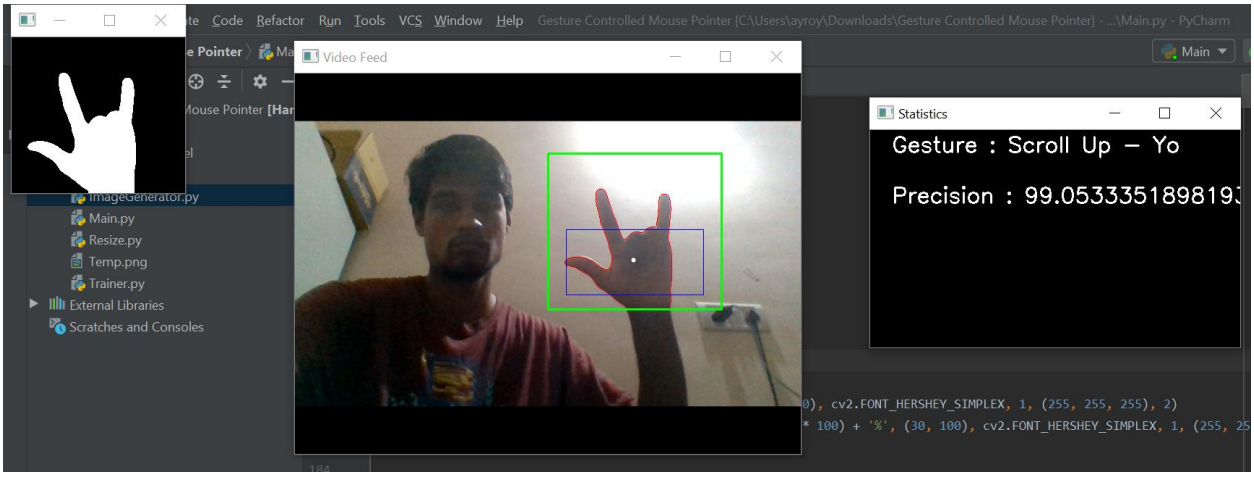

Figure 4.10.5 Scroll up gesture

4.10.6 Scroll Down

This function scrolls the screen down 10 units. It is detected when the thumb and the little finger are held up.

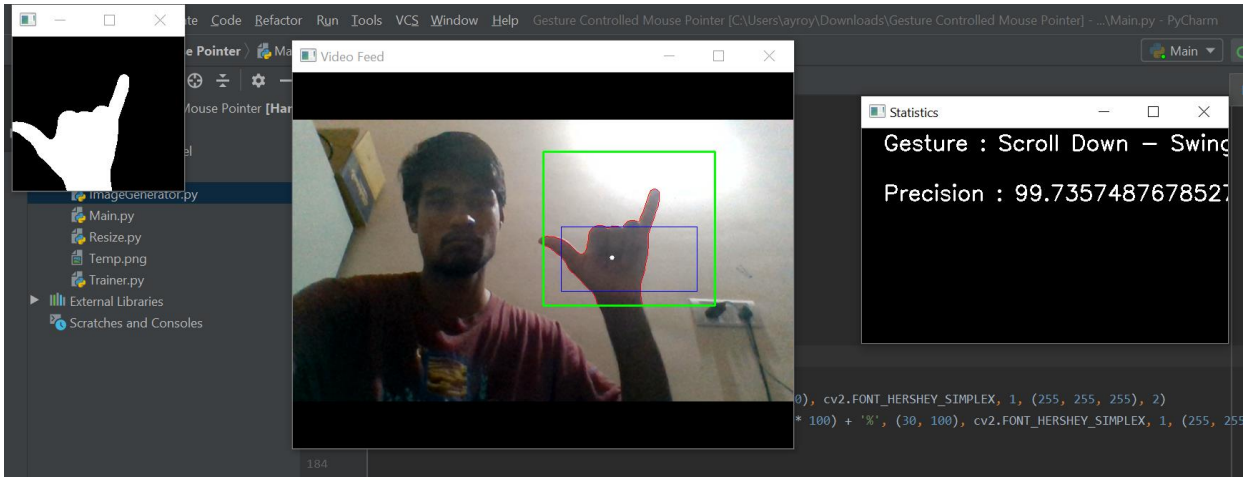

Figure 4.10.6 Scroll down gesture

4.11 Architecture Diagram

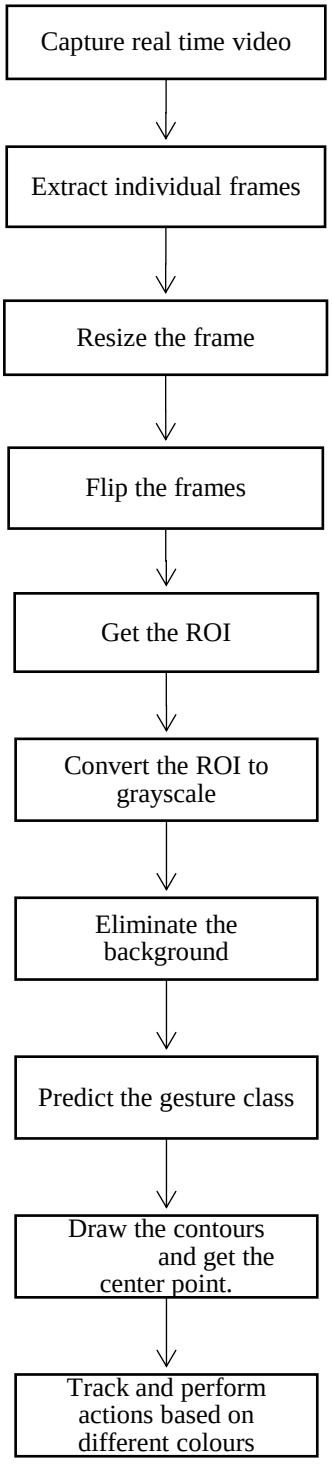

Figure 4.11 Architecture Diagram

CHAPTER 5

DRAWBACKS

Since we have taken a region of reference so the hand needs to be in that region of reference for this to work otherwise it will not be able to predict the gesture. There should be no moving objects in the background as we have used the background elimination technique, so if there is a moving object in back it will consider it too. The hand gestures need to be done appropriately for the system to guess it more properly.

The framework may run slower on certain PCs with low computational abilities since a lot of complex calculations are done in a very short span of time. Although a standard personal computer has the required computational power for ideal execution of the framework. Another problem is that if the resolution of the camera is too high then the project may run at a slower speed. This issue can be resolved by decreasing the resolution of the captured picture by making changes in the framework.

CHAPTER 6

CONCLUSION

We have developed a hand gesture-controlled mouse pointer by using web camera. We have used OpenCV and Tensorflow software to implement our project. Along with OpenCV we coded in Python language, we used convolutional neural network to recognize different gestures and we have used PyAutoGUI, an inbuilt library in python to programmatically control the mouse and keyboard. This technology has great applications in the fields of computer graphics, gaming, prosthetics, and many more. This technology can be used to help patients who are physically challenged. Most of the earlier applications developed required additional hardware which is often very costly. Our aim was to create this technology in the cheapest all possible way and to create it on a standardized operating system. Various application can be developed using this technology with the minimum requirement of resources.

6.1 Results

Most of the gestures have a precision of above 95%. The system can easily distinguish between each of the different gestures. The system provides good results in both bright and low lighting conditions. The system needs a stable background to run as it uses background elimination technique to detect the hand. Also, it is preferred to have a solid background colour.

CHAPTER 7

FUTURE ENHANCEMENTS

We would improve the execution of our project particularly the gesture controlling. Moreover, we additionally need to reduce the reaction time of the project for cursor development so that it can totally be utilized to supplant our regular mouse. We are also intending to structure an equipment execution for the equivalent in order to improve exactness and increase the usefulness to different spaces, for example, a gaming controller or as a universally useful PC controller. We should be able to detect the hand an any part of the screen and it could be at any distance from the camera. Some transitions could be made smoother. Since this will work as a substitute to the actual mouse, we should be able to carry out every function which is done by the mouse like drag, pinch zoom etc. Also, there should be a better background elimination technique to recognize moving backgrounds and eliminate them. Sometimes the gesture is not accurately detected, and fluctuation occurs which can cause wrong gestures to be detected. This can be improved by training the system more intensively which would require a very high-performance processor.

CHAPTER 8
REFERENCES

[1] Horatiu-Stefan Grif, Cornel Cristian Farcas, “Mouse Cursor Control System Based on Hand Gesture”

[2] Gaurav Sahu, Sonam Mittal, “Controlling Mouse Pointer Using Web Cam”

[3] Heera Lal Bhadrecha, “Finger Gesture Control Computer Mouse with Image Processing”

[4] Horatiu-Stefan Grif, Trian Turc, “Human hand gesture-based system for mouse cursor control”

[5] Kabeer Manchanda, Benny Bing, “Advanced mouse pointer control using trajectory-based gesture recognition”

[6] Kabid Hassan Shibly, Samrat Kumar Dey, Md. Aminul Islam, Shahriar Iftekhar Showrav, “Design and Development of Hand Gesture Based Virtual Mouse”

[7] Rachit Puri, “Gesture Recognition Based Mouse Events”

[8] Onkar Yadav, Sagar Makwana, Pandhari Yadav, Prof. Leena Raut, “Cursor Movement by Hand Gesture”

[9] Kalyani Pendke, Prasanna Khuje, Smita Narnaware, Shweta Thool, Sachin Nimje, “Computer Cursor Control Mechanism by Using Hand Gesture Recognition”

[10] Abhilash S, Lisho Thomas, Naveen Wilson, Chaithanya C, “Virtual Mouse Using Hand Gesture”

[11] Jiyoung Park, Juneho Yi, “Efficient Fingertip Tracking and Mouse Pointer Control for a Human Mouse”

[12] Nalini Jungare, “Review on Virtual Mouse Using Hand Gesture and Color Detection”

[13] Ahemad Siddique, Abhishek Kommera ,Divya Varma, “Simulation of Mouse using Image Processing Via Convex Hull Method”

[14] Akshay Ishwar Pawar, Sahil Prakash Tumbare , Tejal Avinash Godse , Tejaswini Wagh, “Mouse Control using a Web Camera and Hand Gestures with Colour Tapes”

[15] Sai Prasanth, Aswathy Gopalakrishnan, Oviya Sivakumar, A. Aruna, “Enhancing User Experience Using Hand - Gesture Control”

[16] Hojoon Park, “A Method for Controlling Mouse Movement using a Realtime Camera”

[17] Shany Jophin, Sheethal M.S, Priya Philip, T M Bhruguram, “Gesture Based Interface Using Motion and Image Comparison”

[18] Chinnu Thomas, D.Lakshmi, “Gesture Based Computing as an Alternative to Mouse by Calibrating Principal Contour Process Actions”

[19] Suneeta V.Budihal, Anant R Choudhari, Kripa S Patil, Nayana P Desai, “Design of Gesture Controlled Wireless Mouse Using Microcontroller”

[20] Athiya Marium, Deepthi Rao, Divina Riya Crasta, Kavya Acharya, Rio D’Souza, “Hand Gesture Recognition using Webcam”

APPENDIX

ImageGenerator.py

```
import camera as camera
import cv2
import imutils
import numpy as np

bg = None

def run_avg(image, aWeight):
    global bg
    if bg is None:
        bg = image.copy().astype("float")
        return
    cv2.accumulateWeighted(image, bg, aWeight)

def segment(image, threshold=25):
    global bg
    diff = cv2.absdiff(bg.astype("uint8"), image)
    thresholded = cv2.threshold(diff, threshold, 255, cv2.THRESH_BINARY)[1]
    cnts, hierarchy = cv2.findContours(thresholded.copy(), cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)

    if len(cnts) == 0:
        return
    else:
        segmented = max(cnts, key=cv2.contourArea)
        return (thresholded, segmented)
```

```
def main():
    aWeight = 0.5
    camera = cv2.VideoCapture(0)
    top, right, bottom, left = 100, 400, 300, 600
    num_frames = 0
    image_num = 0
    start_recording = False

    while(True):
        (grabbed, frame) = camera.read()
        if (grabbed == True):
            frame = imutils.resize(frame, width=700)
            frame = cv2.flip(frame, 1)
            clone = frame.copy()
            (height, width) = frame.shape[:2]
            roi = frame[top:bottom, right:left]
            gray = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)
            gray = cv2.GaussianBlur(gray, (7, 7), 0)

            if num_frames < 30:
                run_avg(gray, aWeight)
                print(num_frames)
            else:
                hand = segment(gray)
                if hand is not None:
                    (thresholded, segmented) = hand
                    cv2.drawContours (clone, [segmented + (right, top)], -1, (0, 0, 255))
                    if start_recording:
                        cv2.imwrite("Dataset/YoImages/yo_" + str(image_num) + '.png', thresholded)
                        image_num += 1
                    cv2.imshow("Thesholded", thresholded)
```

```
            cv2.rectangle (clone, (left, top), (right, bottom), (0, 255, 0), 2)
            num_frames += 1
            cv2.imshow("Video Feed", clone)
            keypress = cv2.waitKey(1) & 0xFF
            print(image_num)

            if keypress == ord("q") or image_num > 999:
                break
            if keypress == ord("s"):
                start_recording = True

        else:
            print("Error, Check Camera")
            break

main()
#camera.release()
cv2.destroyAllWindows()
```

Resize.py

```
from PIL import Image

def resizeImage(imageName):
    basewidth = 100
    img = Image.open(imageName)
    wpercent = (basewidth/float(img.size[0]))
    hsize = int((float(img.size[1])*float(wpercent)))
    img = img.resize((basewidth,89), Image.ANTIALIAS)
    img.save(imageName)

for i in range(0, 100):
    resizeImage("C:/Users/ayroy/Downloads/Gesture Controlled Mouse
Pointer/Dataset/YoTest/yo_" + str(i) + '.png')
```

Trainer.py

```
import tensorflow as tf
import tflearn
from tflearn.layers.conv import conv_2d, max_pool_2d
from tflearn.layers.core import input_data, dropout, fully_connected
from tflearn.layers.estimator import regression
import numpy as np
import cv2
from sklearn.utils import shuffle

# Load Images from Swing
loadedImages = []
for i in range(0, 1000):
    image = cv2.imread('Dataset/SwingImages/swing_' + str(i) + '.png')
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    loadedImages.append(gray_image.reshape(89, 100, 1))

# Load Images From Palm
for i in range(0, 1000):
    image = cv2.imread('Dataset/PalmImages/palm_' + str(i) + '.png')
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    loadedImages.append(gray_image.reshape(89, 100, 1))

# Load Images From Fist
for i in range(0, 1000):
    image = cv2.imread( 'Dataset/FistImages/fist_' + str(i) + '.png')
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    loadedImages.append(gray_image.reshape(89, 100, 1))

#Load Images From Peace
```

```
for i in range(0, 1000):
    image = cv2.imread('Dataset/PeaceImages/peace_' + str(i) + '.png')
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    loadedImages.append(gray_image.reshape(89, 100, 1))

#Load Images From ThumbsUp
for i in range(0, 1000):
    image = cv2.imread('Dataset/TriImages/tri_' + str(i) + '.png')
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    loadedImages.append(gray_image.reshape(89, 100, 1))

#Load Images From ThumbsDown
for i in range(0, 1000):
    image = cv2.imread('Dataset/YoImages/yo_' + str(i) + '.png')
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    loadedImages.append(gray_image.reshape(89, 100, 1))

# Create OutputVector
outputVectors = []
for i in range(0, 1000):
    outputVectors.append([1, 0, 0, 0, 0, 0])

for i in range(0, 1000):
    outputVectors.append([0, 1, 0, 0, 0, 0])

for i in range(0, 1000):
    outputVectors.append([0, 0, 1, 0, 0, 0])

for i in range(0, 1000):
    outputVectors.append([0, 0, 0, 1, 0, 0])
```

```
for i in range(0, 1000):
    outputVectors.append([0, 0, 0, 0, 1, 0])

for i in range(0, 1000):
    outputVectors.append([0, 0, 0, 0, 0, 1])

testImages = []

for i in range(0, 100):
    image = cv2.imread('Dataset/SwingTest/swing_' + str(i) + '.png')
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    testImages.append(gray_image.reshape(89, 100, 1))

for i in range(0, 100):
    image = cv2.imread('Dataset/PalmTest/palm_' + str(i) + '.png')
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    testImages.append(gray_image.reshape(89, 100, 1))

for i in range(0, 100):
    image  =  cv2.imread( 'Dataset/FistTest/fist_' + str(i) + '.png')
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    testImages.append(gray_image.reshape(89, 100, 1))

for i in range(0, 100):
    image = cv2.imread('Dataset/PeaceTest/peace_' + str(i) + '.png')
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    testImages.append(gray_image.reshape(89, 100, 1))

for i in range(0, 100):
    image = cv2.imread('Dataset/TriTest/tri_' + str(i) + '.png')
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

```
    testImages.append(gray_image.reshape(89, 100, 1))

for i in range(0, 100):
    image = cv2.imread('Dataset/YoTest/yo_' + str(i) + '.png')
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    testImages.append(gray_image.reshape(89, 100, 1))

testLabels = []

for i in range(0, 100):
    testLabels.append([1, 0, 0, 0, 0, 0])

for i in range(0, 100):
    testLabels.append([0, 1, 0, 0, 0, 0])

for i in range(0, 100):
    testLabels.append([0, 0, 1, 0, 0, 0])

for i in range(0, 100):
    testLabels.append([0, 0, 0, 1, 0, 0])

for i in range(0, 100):
    testLabels.append([0, 0, 0, 0, 1, 0])

for i in range(0, 100):
    testLabels.append([0, 0, 0, 0, 1, 0])

tf.reset_default_graph()

convnet = input_data(shape=[None, 89, 100, 1], name='input')
convnet = conv_2d(convnet, 32, 2, activation='relu')
```

```
convnet = max_pool_2d(convnet, 2)
convnet = conv_2d(convnet, 64, 2, activation='relu')
convnet = max_pool_2d(convnet, 2)
convnet = conv_2d(convnet, 128, 2, activation='relu')
convnet = max_pool_2d(convnet, 2)
convnet = conv_2d(convnet, 256, 2, activation='relu')
convnet = max_pool_2d(convnet, 2)
convnet = conv_2d(convnet, 256, 2, activation='relu')
convnet = max_pool_2d(convnet, 2)
convnet = conv_2d(convnet, 128, 2, activation='relu')
convnet = max_pool_2d(convnet, 2)
convnet = conv_2d(convnet, 64, 2, activation='relu')
convnet = max_pool_2d(convnet, 2)
convnet = fully_connected(convnet, 1000, activation='relu')
convnet = dropout(convnet, 0.75)
convnet = fully_connected(convnet, 6, activation='softmax')
convnet = regression(convnet, optimizer='adam', learning_rate=0.001,
loss='categorical_crossentropy', name='regression')
model = tflearn.DNN(convnet, tensorboard_verbose=0)
loadedImages, outputVectors = shuffle(loadedImages, outputVectors, random_state=0)
model.fit(loadedImages, outputVectors, n_epoch=50, validation_set=(testImages,
testLabels),snapshot_step=100, show_metric=True, run_id='convnet_coursera')
model.save("TrainedNewModel/GestureRecogModel.tfl")
```

Main.py

```
import tensorflow as tf
import tflearn
from tflearn.layers.conv import conv_2d, max_pool_2d
from tflearn.layers.core import input_data, dropout, fully_connected
from tflearn.layers.estimator import regression
import numpy as np
from PIL import Image
import cv2
import imutils
import pyautogui


bg = None
n=0
cX=0
cY=0
nX=0
nY=0
i=0

def resizeImage(imageName):
    basewidth = 100
    img = Image.open(imageName)
    wpercent = (basewidth / float(img.size[0]))
    hsize = int((float(img.size[1]) * float(wpercent)))
    img = img.resize((basewidth, hsize), Image.ANTIALIAS)
    img.save(imageName)
```

```
def run_avg(image, aWeight):
    global bg
    if bg is None:
        bg = image.copy().astype("float")
        return
    cv2.accumulateWeighted(image, bg, aWeight)

def segment(image, threshold=25):
    global bg
    diff = cv2.absdiff(bg.astype("uint8"), image)
    thresholded = cv2.threshold(diff, threshold, 255, cv2.THRESH_BINARY)[1]
    (cnts, _) = cv2.findContours(thresholded.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)

    if len(cnts) == 0:
        return
    else:
        segmented = max(cnts, key=cv2.contourArea)
        return (thresholded, segmented)

def main():
    global cX,cY,nX,nY
    aWeight = 0.5
    camera = cv2.VideoCapture(0)
    top, right, bottom, left = 110, 350, 325, 590
    num_frames = 0
    start_recording = False
    n=0
    while (True):
        (grabbed, frame) = camera.read()
        frame = imutils.resize(frame, width=700)
```

```
frame = cv2.flip(frame, 1)
clone = frame.copy()
(height, width) = frame.shape[:2]

roi = frame[top:bottom, right:left]
gray = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)

gray = cv2.GaussianBlur(gray, (7, 7), 0)

if num_frames < 30:
   run_avg(gray, aWeight)
else:
   hand = segment(gray)

   if hand is not None:
      (thresholded, segmented) = hand
      cv2.drawContours(clone, [segmented + (right, top)], -1, (0, 0, 255))
      try:
         M = cv2.moments(segmented + (right, top))
         cX = int(M["m10"] / M["m00"])
         cY = int(M["m01"] / M["m00"])
         if nX == 0 and nY == 0 :
            nX=cX
            nY=cY
         cv2.circle(clone, (cX, cY), 3, (255, 255, 255), -1)
         #print(str(cX) + " " + str(cY))

      except:
         print("Empty")

      if start_recording:
```

```
                cv2.imwrite('Temp.png', thresholded)
                resizeImage('Temp.png')
                predictedClass, confidence = getPredictedClass()
                showStatistics(predictedClass, confidence)
            cv2.imshow("Thesholded", thresholded)

        cv2.rectangle(clone, (left, top), (right, bottom), (0, 255, 0), 2)
        cv2.rectangle(clone, (375,215), (565,305), (255,0,0), 1)
        num_frames += 1
        cv2.imshow("Video Feed", clone)
        keypress = cv2.waitKey(1) & 0xFF

        if keypress == ord("q"):
            break
        if keypress == ord("s"):
            start_recording = True

def getPredictedClass():
    image = cv2.imread('Temp.png')
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    prediction = model.predict([gray_image.reshape(89, 100, 1)])
    return np.argmax(prediction), (np.amax(prediction) / (prediction[0][0] + prediction[0][1] +
prediction[0][2] + prediction[0][3] + prediction[0][4]  + prediction[0][5]))


def showStatistics(predictedClass, confidence):
    global n,cX,cY,nX,nY,i
    pyautogui.FAILSAFE = False
    textImage = np.zeros((300, 512, 3), np.uint8)
    className = ""
```

```
if predictedClass == 0:
    className = "Scroll Down - Swing"
    if i==8:
        i = 0
    else:
        pyautogui.scroll(-10)
        n = 1
        i=0

elif predictedClass == 1:
    className = "Right Click - Palm"
    if n != 2:
        pyautogui.click(button='right')
        n = 2
    i = 0

elif predictedClass == 2:
    className = "Mouse Movement - Fist"

    if i<8:
        pyautogui.move((cX-nX)*10,(cY-nY)*10)
        if (nX < 375):
            pyautogui.move(-30, 0)
        if (nX > 565):
            pyautogui.move(30, 0)
        if (nY < 215):
            pyautogui.move(0, -30)
        if (nY > 305):
            pyautogui.move(0, 30)

        if abs(cX-nX)<2 and abs(cY-nY)<5:
```

```
            i=i+1
        else:
            i=0
    n = 3

elif predictedClass == 3:
    className = "Left Click - Peace"
    if n != 4:
        pyautogui.click()
        n = 4
        i = 0

elif predictedClass == 4:
    className = "Double Click - Three Finger"
    if n!=5:
        pyautogui.doubleClick()
        n = 5
    i = 0

elif predictedClass == 5:
    className = "Scroll Up - Yo"
    pyautogui.scroll(10)
    n = 6
    i = 0
nX=cX
nY=cY
print(className)

cv2.putText(textImage, "Gesture : " + className, (30, 30),
cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2)
cv2.putText(textImage, "Precision : " + str(confidence * 100) + '%', (30, 100),
```

```
cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2)
    cv2.imshow("Statistics", textImage)

tf.reset_default_graph()

convnet = input_data(shape=[None, 89, 100, 1], name='input')
convnet = conv_2d(convnet, 32, 2, activation='relu')
convnet = max_pool_2d(convnet, 2)
convnet = conv_2d(convnet, 64, 2, activation='relu')
convnet = max_pool_2d(convnet, 2)
convnet = conv_2d(convnet, 128, 2, activation='relu')
convnet = max_pool_2d(convnet, 2)
convnet = conv_2d(convnet, 256, 2, activation='relu')
convnet = max_pool_2d(convnet, 2)
convnet = conv_2d(convnet, 256, 2, activation='relu')
convnet = max_pool_2d(convnet, 2)
convnet = conv_2d(convnet, 128, 2, activation='relu')
convnet = max_pool_2d(convnet, 2)
convnet = conv_2d(convnet, 64, 2, activation='relu')
convnet = max_pool_2d(convnet, 2)
convnet = fully_connected(convnet, 1000, activation='relu')
convnet = dropout(convnet, 0.75)
convnet = fully_connected(convnet, 6, activation='softmax')
convnet = regression(convnet, optimizer='adam', learning_rate=0.001,
loss='categorical_crossentropy', name='regression')
model = tflearn.DNN(convnet, tensorboard_verbose=0)
model.load("TrainedNewModel/GestureRecogModel.tfl")

main()
```