Hand Gesture controlled Mouse Pointer

A Report Submitted

in Partial Fulfillment of the Requirements

for the Degree of

Bachelor of Technology

in

Computer Science & Engineering

by

Siddharth Majumdar(20164044), Sunil Kumar(20164012),

Saurabh(20164104) and Pradyumna Pandey(20164159)

Group: **CS-26**

to the

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

MOTILAL NEHRU NATIONAL INSTITUTE OF TECHNOLOGY

ALLAHABAD, PRAYAGRAJ

May 01, 2019

UNDERTAKING

I declare that the work presented in this report titled "*Hand Gesture controlled Mouse Pointer*", submitted to the Computer Science and Engineering Department, Motilal Nehru National Institute of Technology Allahabad, Prayagraj, for the award of the ***Bachelor of Technology*** degree in ***Computer Science & Engineering***, is my original work. I have not plagiarized or submitted the same work for the award of any other degree. In case this undertaking is found incorrect, I accept that my degree may be unconditionally withdrawn.

May 01, 2019
Allahabad

(Siddharth Majumdar)

(Sunil Kumar)

(Saurabh)

(Pradyumna Pandey)

CERTIFICATE

Certified that the work contained in the report titled "*Hand Gesture controlled Mouse Pointer*", by *Siddharth Majumdar, Sunil Kumar, Saurabh and Pradyumna Pandey*, has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

(Er. Rajesh Tripathi)
Computer Science and Engineering Dept.
M.N.N.I.T. Allahabad

May 01, 2019

Preface

In the domain of human computer interaction, the terms 'gesture' and 'gesture recognition' are heavily encountered. In today's world, where computer vision plays a big role in shaping reality, there is great emphasis on the usage of hand gestures in place of input modality in broad range applications. It is known that hand gestures have the natural ability to represent ideas and actions quite easily, thus giving rise to the idea of interpreting gestures to generate corresponding events. In today's world, this is a domain that is highly regarded as something that will shape our future, and hence, extensive research is going on, to ease the access of user and provide a more user-friendly environment. Hence, in this project, we shall try to create a human-computer interaction system based on hand gesture recognition.

Acknowledgments

We would like to thank everyone who contributed in some way in helping us achieve our goals with regards to our project. We are especially grateful to our mentor Er. Rajesh Tripathi for assisting us in our project. Without his expertise and suggestions we would never have been able to complete our work.

We would also like to thank Dr. Shashwati Banerjea for the formation of groups for the project which enabled us to work on this project.

Abstract

Gesture recognition is a sort of perceptual figuring user interface that enables computers to catch and translate human motions as directions. The general meaning of gesture recognition is the capacity of a computer to get motions and execute directions dependent on those motions. In order to understand how gesture recognition works, it is imperative to see how "gesture" is characterized. In its most broad sense, the word gesture can allude to any non-verbal correspondence that is proposed to convey a particular message. In the realm of gesture recognition, a gesture is characterized as any physical development, substantial or little, that can be deciphered by a movement sensor. It might incorporate anything from the clinching of a fist to a football kick or a nod of the head to a peace symbol. Gestures can be wide and clearing or little and contained. Sometimes, the meaning of "gesture" may likewise incorporates voice or verbal directions.

For the purpose of this project, we have created an interface where we use various hand gestures like clinching of a fist, pointing of a finger, peace symbol, raising of thumb et al, to act as an input for the cursor of the computer system.

Contents

Chapter 1

Introduction

In recent years, with the rapid progress in the domain of computer vision, specifically the domains of image processing and recognition technology, the focus is no longer restricted to the improvement of traditional input methods of interaction between humans and computers, the usage of biological characteristics will help in studying interaction technologies, which come across as more natural, to make this interaction more direct, is becoming the current research focus of human-computer interaction.

Gesture recognition was first proposed by Myron W. Krueger as a new form of interaction between human and computer in the middle of seventies [3]. It has become a highly significant research area with the rapid advancements in computer hardware and vision systems. The main problem in gesture interaction is how to make hand gestures understood by computers. A lot of techniques related to hand gesture recognition have been proposed recently.

Out of all the human centered human-computer interaction technologies, hand gesture recognition is one domain with a significant amount of research being done on it. Hand gesture recognition, is simply the recognition of the hand motions and gestures. It is broadly divided into two categories:

- Static Hand Gesture Recognition: This type of hand gesture recognition involves the interpretation of static hand shapes by the system, i.e. gestures that do not involve any motion with them. For example, pointing a finger, clinching a fist etc.

- Dynamic Hand Gesture Recognition: This type of hand gesture recognition involves the detection of gestures which involve motion. Here, we not only, interpret the gesture, but also the motion that is associated with them and some attributes associated with the motion.

1.1 Motivation

The most basic expectation from any human-centered human-computer interaction technology is the ease of access for the user, when compared to any conventional input modality. The mouse in an input device that controls much of the operations/tasks of a computer system, hence, by gaining control of the mouse pointer using hand gestures, we can significantly enhance user experience, and take a step towards building a future with augmented reality.

Chapter 2

Related Work

The domain of our project, i.e gesture recognition, or more specifically hand gesture recognition has a significant amount of literature associated with it. If you talk about the 20th century, initial research and progress in this field was done by many computer scientists. One of the first attempts can be attributed to Zimmerman et al[8]. In this case, the scientists tried to develop a hand to machine interface device that provided real time gesture, orientation and position information. Many more attempts were made and research work was done in this domain, which was being considered as the key to the future.

Although pattern recognition and image processing were concepts that were identified as potential market disruptors long back, and research had been started on them as early as 1970s, it was about the 1980s, that the idea of combining the two concepts and human biological information to create more natural systems arose. An attempt was made by Sun Fu[4] in 1976 to study about Pattern Recognition and Image Processing, but this did not include any portion related to hand gesture recognition or any other biological feature extraction.

There have been many research papers citing the positives and negatives of this technology, and one of them by Khan and Ibraheem[5], listed out the literature review of the technology domain. This paper served as the basis for the theoretical knowledge that helped us in the project.

The recognition part has not just been limited to webcams and small cameras.

There have been attempts to use other devices and sensors for this purpose. One such attempt was taken by Chen et al[2], where they used a Microsoft Kinect sensor for the purpose of recognition and used Support Vector Machines(SVM) as the recognition algorithm. They were able to recognize the gestures with a recognition rate of 95.42%. This high recognition rate results from the fact that the model was trained correctly and efficiently, and all outliers were treated and then eliminated to enhance the success rate of prediction.

Chapter 3

Proposed Work

3.1 Objective

Mouse pointer (or cursor) control has been the key to getting tasks done in a computer system. Using hand gestures to control the cursor will result in a better user interface and experience. Hence, we aim to control various mouse pointer actions such as cursor movement, clicking etc. using hand gestures and hand gesture recognition.

3.2 Software Requirement Specification

3.2.1 Introduction

- *Purpose*

The aim of this document[1] is to provide a detailed description of the Hand Gesture controlled Mouse Pointer. It will cover the applications and features of the system, the interfaces of the system, what the system is expected to do, the constraints that the project will work under and how it behaves in response to external stimuli. This is intended for both the developers and the users of this system.

- *Scope*

The system is intended for making a hand gesture controlled mouse pointer, which covers all the actions performed by the mouse pointer. Although this project is only limited to hand gestures, we can create a similar interface using eye gazing and blinking as well. Also, mouse pointer control can be extended to control of other input devices such as keyboard, joystick etc.

- *Glossary*

  - Epoch : A complete presentation of the training dataset to the learning machine.
  - Accuracy : The correctness of the model in predicting a particular gesture made by the user.
  - Socket : A mode of communication between two interfaces.

- *Overview of Document*

The next section, the Overall Description section, of this document gives an overview of the functionality of the project. It describes the informal requirements and is used to establish a context for the technical requirements specification in the next section. The third section, Requirements Specification section, of this document is written primarily for the developers and describes in technical terms the details of the functionality of the product. Both sections of the document describe the same software product in its entirety, but are intended for different audiences and thus use different language.

3.2.2 Overall Description

- *System Architecture*

The primary purpose of the software is to detect a hand gesture made by the user and perform the corresponding event/action assigned to it. Now, the software has two components, a front-end GUI that captures the gestures using a web-cam (built-in or external) and a back-end which generates the results after running the model on the incoming webcam capture of the hand gesture. To maintain connectivity between the front end and the back end, a socket is maintained. The socket is responsible for transferring data to the back end for processing and returning the output to the front end for the user to witness the corresponding actions. The front-end is basically a software built on Java, whereas the back-end has been created using Python, so as to take help of the various libraries associated with image detection and processing.

3.3 Platforms Used

3.3.1 IntelliJ IDEA

IntelliJ IDEA is a Java integrated development environment developed by JetBrains, and serves the purpose of developing computer software. It provides support to a host of other languages through its software suite like Python(PyCharm), JavaScript(WebStorm), SQL(DataGrip) etc. The interface of the software has been illustrated in Figure 1.

In our project, IntelliJ IDEA has been used to create the front-end interface of the project where we capture the images using the webcam and Sarxos API and send the required data to the Python client (backend) using a socket.

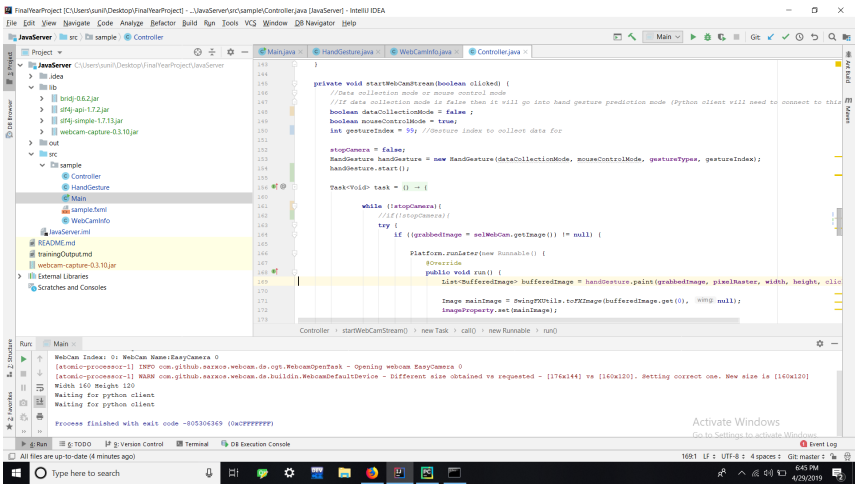

Figure 1: IntelliJ IDEA user interface

3.3.2 PyCharm

PyCharm is a Python integrated development software developed by JetBrains, and is used to develop codes and projects based on Python. It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems (VCS), and supports web development with Django. The interface of the software has been illustrated in Figure 2.

In our project, PyCharm has been used to create the Python client wherein the entire programming for the image detection and processing has been done. It communicates with the Java GUI using a socket and transfers data through it.

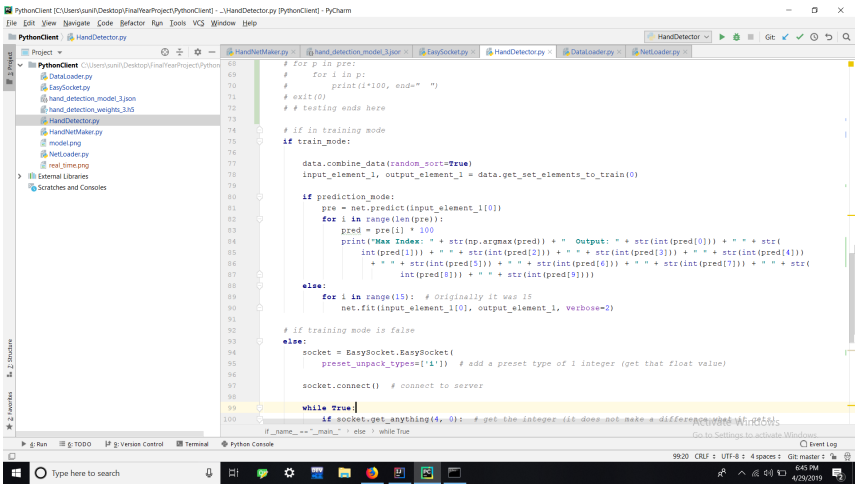

Figure 2: PyCharm user interface

3.4 Overview of Approach

For the purpose of creating the software, we started by collecting pre-existing data for the initial training of our model. The data basically consisted of various snapshots of different hand gestures. We classified our dataset into 10 parts for identifying 10 different hand gestures, and planned on assigning corresponding actions to each of them.

In order to classify the images and data, we created a model, fit the collected data into the model, trained it using the Convolutional Neural Network algorithm and received the predictions. We then sent the predicted data into the Java client using a socket and hence, each identified gesture got associated with a particular mouse action that was already assigned to its category.

3.5 Description of the Model

The model type we are using for the project is Sequential, as it allows you to build a model layer by layer. A visual representation of the model has been illustrated in Figure 3. The detailed flow of the sequential model has been illustrated in Figure 4 and Figure 5. Every layer used in the model can be either of the six types:

3.5.1 Zero Padding

This layer adds rows and columns of zeroes at the top, bottom, left and right side of an image tensor. It prevents data loss by keeping track of the outliers and edgees, thus leading to a loss in the process of feature extraction. It takes 2 parameters which symbolise the padding values, i.e. the number of padding rows and colums to be added. A single image tensor is processed by sending tuples of a given size to the neural network, and the number is determined by the batch size parameter.

3.5.2 Two-Dimensional Convolution

This layer extracts a matrix of a given size (specified in the parameters), and multiplies it with the kernel matrix, and gives a corresponding output matrix. The kernel matrix is a basically a weight matrix, whose weights are enhanced as the training of the model progresses with each epoch. The activation function used in this layer is called Rectified Linear Unit, which basically has an output range from 0 to the summation of the products of extracted matrix and kernel matrix. The purpose of this layer is to set a threshold for the features, i.e. only those features are counted for further training which cross the threshold value.

3.5.3 Max Pooling

This main purpose of this layer is to extract the maximum value from a given matrix. It simply helps in reducing the dimensionality and down-sampling an input image, thus allowing for certain assumptions to be made about the features. It also reduces the cost of computing by decreasing the count of parameters involved in learning.

3.5.4 Dropout

The basic purpose of this layer/technique to solve the problem of overfitting in a neural network. It is done by dropping random units and their connections from the network during training, and thus preventing them from co-adapting.[7]

3.5.5 Flatten

This layer serves the function of reshaping a tensor to give a shape equal to the count of members of that particular tensor. It basically unrolls the values that begin at the last dimension.

3.5.6 Dense

This layer performs classification on the various features that are extracted by the convolutional layers and downsampled by the pooling layers. Every node of a particular dense layer is connected to every node of the preceding layer.

- *Activation Function*

This is a function that is embedded into every node (or neuron) of a dense layer. It can be a function of any type like boolean, integer etc. The most common activation function is Rectified Linear Unit (ReLU), which is the one we have used in our project.

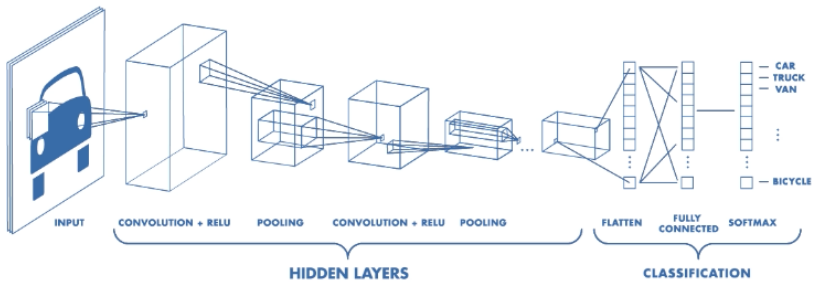

Figure 3: Visual representation of different layers of a CNN from [6]

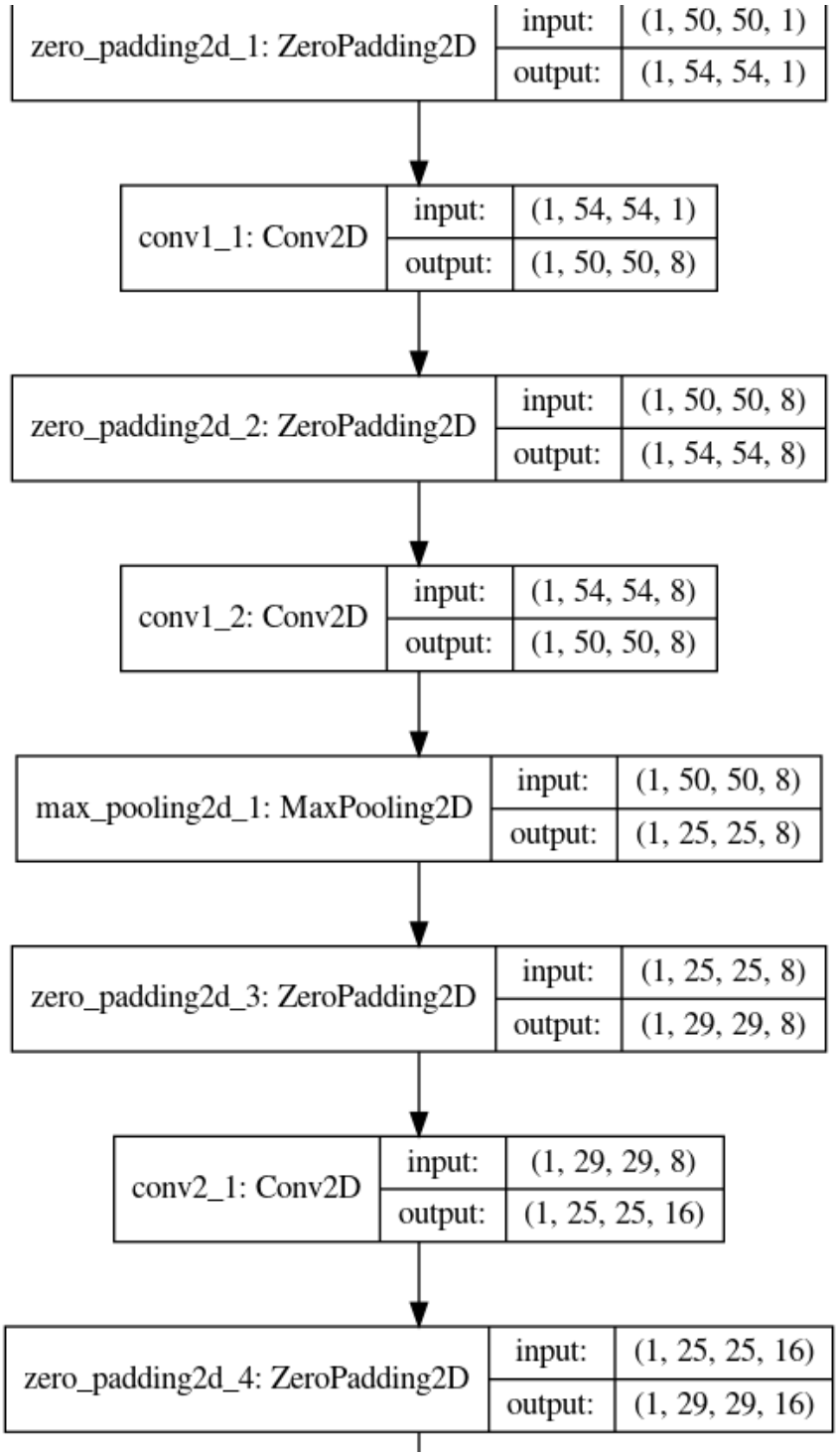

Figure 4: Layer description for initial layers of CNN

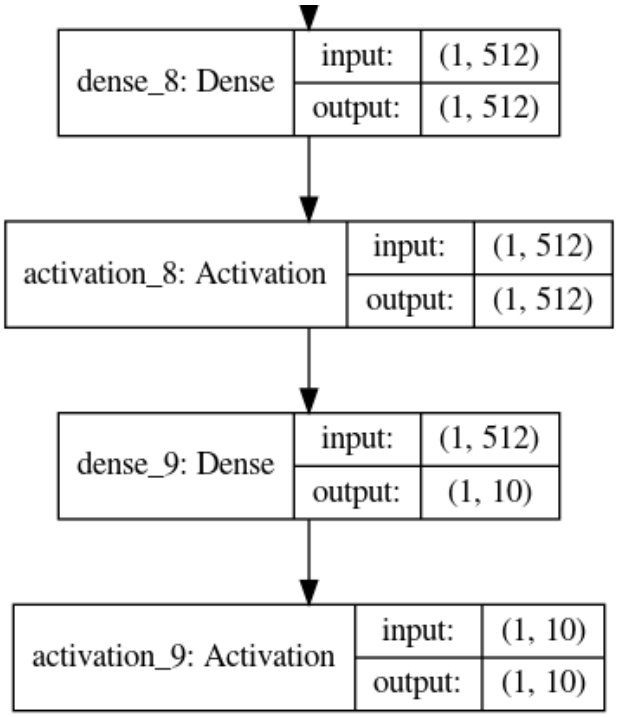

Figure 5: The final layer of CNN shows it to be reduced to 10 possible outputs

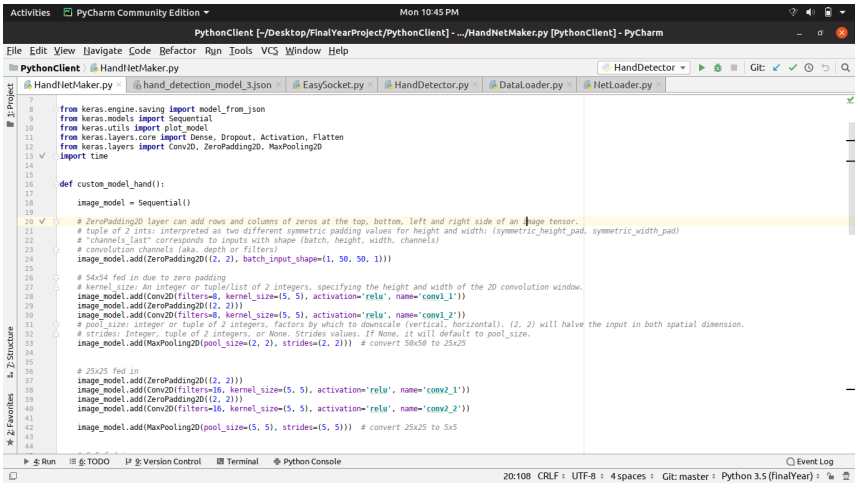

Figure 6: Snapshot of the Python code written in PyCharm

3.6 The Software Interface

As mentioned in Section 3.3.1, the interface is built on IntelliJ IDEA. The graphic captured by the webcam is displayed on the screen, along with two other visuals of black and white. The program running in the front end detects the skin colour by specifying its RGB value and converts it into white and keeping all the other contents of the image black. It also has a panel which shows the percentage prediction of each category. The complete interface of the software can be seen in Figure 7.

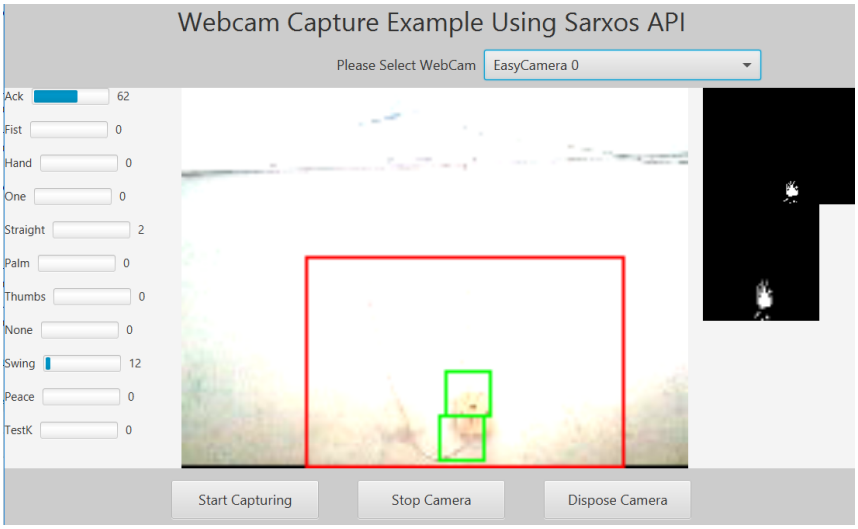

Figure 7: Java based user interface

3.6.1 Formatting of Image Captured By Webcam

The image captured by the webcam, will be processed by a number of functions in the Java program. We will first convert the RGB value of each pixel to its corresponding HSB (Hue, Saturation and Brightness) value. Then we will check whether the HSB value of the pixel in question falls in range of the HSB value of skin colour or not. Now depending on whether the pixel falls in the desired range, the value of the pixel in the PixelRaster 2-D array will be set to either 0xFFFFFFFF(white) or 0xFF000000(black).

Now, we will consider another 2-D array called DensityRastor, which keeps track of the surrounding density of the pixel. It will consider a block of arbitrary size around a pixel, then it will increment the DensityRastor value for every point inside that block, if the pixel around which the block is conceived is of skin color. Now, if the DensityRastor value of a particular pixel is greater than a threshold value, it will automatically set the PixelRastor value of that pixel to white irrespective of its original value. This process will be applied on every pixel of the capture. This is done to eliminate the effect of outliers and other irregular points in the capture, so that there is no practical discrepancy in the black and white image so formed after the entire process. The black and white image so formed has been illustrated in Figure 8.

The last part of the image processing is about creating a definite boundary for the final image that has to be sent to the backend via the socket. Now, using a vector of rectangles, we will store all the rectangles formed around each pixel, and then filter out some of them using the intersection function. After all the non-overlapping rectangles are entered into the vector, the minimum and maximum x-coordinates and y-coordinates are calculated to form the overall boundary of the image that has to be sent via the socket. The maximum limit that we have kept for the image boundary is 100x100 (in pixels). If the calculated boundary exceeds the limit, the boundaries are pushed from both sides (up and down and/or left and right) equally by half the difference between the boundary size and 100.

The following formula is used:

$$diff = maxY - minY \tag{1}$$

$$minY = minY + diff/2 \tag{2}$$

$$maxY = maxY - diff/2 \tag{3}$$

Here, maxY and minY are the maximum and minimum values of the Y-coordinate fetched from the vector of rectangles.

The same formula is applied for cropping the horizontal boundary of the image if it exceeds 100. This is done to equalize the effect of cropping on all sides of the image, so that no portion gets unjustly removed from the final image, and each part is given equal importance.

After the final image is processed, one last operation is performed on it, wherein , the image is scaled down from 100x100 to 50x50 to send it to the back end client. This is done because the trained model requires a fixed size image to be fed into it so that it can produce results. In our case, the size that we have chosen for our model is 50x50, hence all images that are sent to the Python client must be of the size 50x50 pixels.

Figure 8: Sample photo that is sent to the Python client

Chapter 4

Results And Analysis

4.1 Analysis

4.1.1 The Testing Process

The software testing is a very important part of the software engineering for this project. For testing purposes, one of our team members volunteered to perform the hand gestures to control the mouse pointer, and check for the accuracy of our trained model using CNN, by testing each gesture and seeing how accurately can it be identified by the model.

4.2 Results

The results of testing the software, as illustrated in the Figure 9 and Figure 10 below, were positive. All gestures were identified with near-perfect accuracy. The small errors in accuracy can be attributed to the socket transferring of data, which is a potential source of leakage of data, hence causing a small amount of disorientation of features.

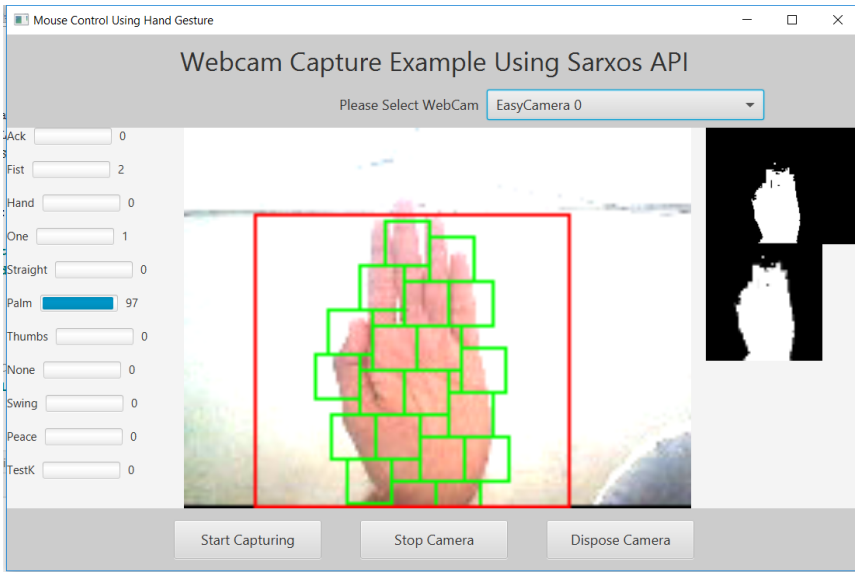

Figure 9: Palm gesture being successfully identified by the software

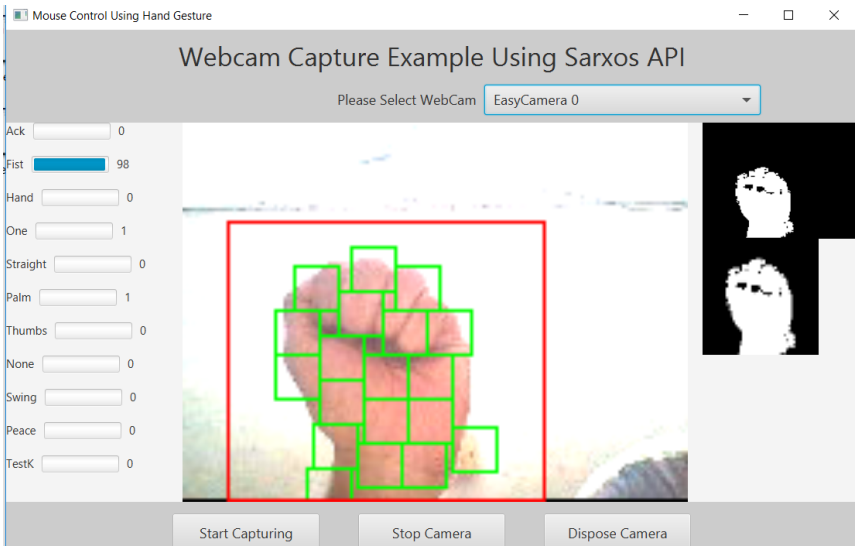

Figure 10: Fist gesture being successfully identified by the software

4.3 Shortcomings

As mentioned in the results, potential data leakage is being caused while transferring data using the socket, which is denying perfect accuracy of identifying gestures. Control over mouse pointer, although achieved, has still scope for improvement as maintaining the speed of movement remains a cause of concern.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

The first step taken by us while starting the project was the development of a model which would serve as the basis of the entire software. After working on the constructive criticism of the project given by our project mentor, a lot of steps were taken and a number of modifications were done on the project that enhanced its working. We have attempted to create a software to control the mouse pointer using hand gestures and have presented our results on it. We have used a sequential model trained using Convolutional Neural Network algorithm to predict the data, and data has been sourced using the in-built webcam of the personal computers, after some required modifications done on the capture. Hence, on the basis of our research and results, we can conclude that hand gesture recognition is an effective input method, that surely has the potential to replace conventional input modality in the future.

5.2 Future Work

The project was aimed at finding a way to enhance the user experience and we worked on hand gesture recognition. While we used hand gesture recognition, there are a host of other options that can serve the purpose of a new input modality, and can prove to be even better than hand gesture recognition. As the world progresses towards a future where augmented reality will play an important part, technologies like these are bound to play a critical role in such environments.

References

[1] Recommended practice for software requirement specification. Standard IEEE 830-1998, IEEE, 1998.

[2] Chen, Y., Luo, B., Chen, Y.-L., Liang, G., and Wu, X. A real-time dynamic hand gesture recognition system using kinect sensor. In *Proceedings of 2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)* (Zhuhai, China, Dec. 2015), IEEE.

[3] Dong, G., Y. Y., and Xie, M. Vision-based hand gesture recognition for human-vehicle interaction. Citeseer, 1: 151-155.

[4] Fu, K. S. Pattern recognition and image processing. *IEEE Transactions on Computers C-25* (Dec. 1976), 1336–1346.

[5] Khan, R. Z., and Ibraheem, N. A. Hand gesture recognition: A literature review. *International Journal of Artificial Intelligence and Applications (IJAIA) 3*, 4 (July 2012).

[6] Medium. Convolutional neural network, 2016. [Online; accessed April 28, 2019].

[7] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning 15* (2014).

[8] Zimmerman, T. G., Lanier, J., Blanchard, C., Bryson, S., and Harvill, Y. A hand gesture interface. *ACM SIGCHI Bulletin 17* (1987).