A

PROJECT REPORT

ON

ML Based Real-time Hand Gesture Recognition System and Its Application

Is submitted in partial fulfillment for the the award of degree of

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

BY

ESHAAN GUPTA (2018UEC1747)

VAIBHAV PRAKASH SARAF(2018UEC1773)

VAIBHAV BADGUZAR (2018UEC1779)

RUPESH KUMAR (2018UEC1773)

UNDER THE GUIDANCE OF

Sh. RAKESH BAIRATHI

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

MALVIYA NATIONAL INSTITUTE OF TECHNOLOGY, JAIPUR

MAY 2022

MALAVIYA NATIONAL INSTITUTE OF TECHNOLOGY
JAIPUR

DEPARTMENT OF ELECTRONICS AND
COMMUNICATION ENGINEERING

JAIPUR-302017, RAJASTHAN, INDIA

CERTIFICATE

This is to certify that the project report entitled **"ML based Real time Hand Gesture recognition System and its application"** submitted by **Eshaan Gupta (2018uec1747), Vaibhav Singh Badguzar (2018uec1779), Rupesh Kumar (2018uec1773) and Vaibhav Prakash Saraf (2018uec1504)** to the **Malaviya National Institute Of Technology, Jaipur**, in partial fulfillment for the award of the degree of **B. Tech in (Electronics And Communication Engineering)** is a *bona fide* record of project work carried out by them under my supervision. The contents of this report, in full or in parts, have not been submitted to any other Institution or University for the award of any degree or diploma.

Date:

Place:

Sh. Rakesh Bairathi

Supervisor

Department of Electronics and Communication Engineering

Malaviya National Institute of Technology Jaipur

MALAVIYA NATIONAL INSTITUTE OF TECHNOLOGY
JAIPUR DEPARTMENT OF ELECTRONICS AND
COMMUNICATION ENGINEERING
JAIPUR-302017, RAJASTHAN, INDIA

DECLARATION

We,

Eshaan Gupta (2018uec1747)

Vaibhav Singh Badguzar(2018uec1779)

Rupesh Kumar (2018uec1773)

Vaibhav Prakash Saraf (2018uec1504)

Declare that this project titled, "**ML based Real time Hand Gesture recognition system and its application"** and the work presented in it are our own. We confirm that:

- This project was done wholly while in the candidature for B.Tech. degree in the Department of Electronics and Communication Engineering at Malaviya National Institute of Technology Jaipur.
- Where any part of this project has previously been submitted for a degree or any other qualification at Malaviya National Institute of Technology Jaipur or any other institution, this has been clearly stated. Where we have consulted the published work of others, this is always clearly attributed. Where we have quoted from the work of others, the source is always given. Except for such quotations, this project is entirely our own work.
- We have acknowledged all main sources of help.

Date: **Sh. Rakesh Bairathi**

Place: Supervisor

Department of Electronics and Communication Engineering

Malaviya National Institute of Technology Jaipur

Table of Content

Chapter 1 **8**
Introduction 8
1.1 Background 8
1.2 Methods of hand gesture recognition [6] 9
1.3 HAND GESTURE TECHNOLOGIES 10
Chapter 2 **14**
TensorFlow 14
2.1 Basics of Tensorflow 14
2.1.1 Advantages of TensorFlow: 14
2.1.2 Disadvantages of TensorFlow 16
2.1.3 Object Detection using SSD Mobilenet and Tensorflow Object Detection 17
2.2 RCNN 18
2.2.1 Basics of RCNN 18
2.2.2 Applications for R-CNN object detectors include 18
2.2.3 Advantages 19
2.2.4 Disadvantages 19
Chapter 3 **20**
Preparation of Dataset 20
3.1 Step 1 : Collecting images 20
3.2 Step 2 : Installing LabelIng 22
3.3 Step 3 : Annotating dataset 23
Chapter 4 **25**
Background Elimination 25
Problem Statement 25
Segment the Hand Region 25
Contour Extraction 26
Chapter 5 **30**

Training and Testing the model 30
Artificial Neural Networks 30
Convolutional Neural Networks 31
The Deep Convolutional Neural Network 32
Conclusions 36

Chapter 6 **40**
Speaker Volume Control Using Gestures 40
OpenCV 40
Controlling Volume 41

Chapter 7 **43**
Applications 43
7.1 SmartCars 43
How it will work 43
7.2 Healthcare 44
7.3 Virtual reality 44
7.4 Consumer electronics 45

References **46**

Source Of Figures **48**

\

ACKNOWLEDGEMENTS

We, all the team members, would like to acknowledge and express gratitude to the Electronics and Communication Department, MNIT for giving us this opportunity to learn new things and grow by doing the project and for providing us the knowledge to reach this level.

We would also like to acknowledge my mentor Dr. Rakesh Bairathi, who was a great source of inspiration and support, and for his timely guidance and all his valuable assistance and guidance in the project work.

We wish to express my deep sense of gratitude to all team members for their immense support , hardwork and help in making this project a wonderful as well as insightful learning opportunity. The generosity and expertise of one and all have improved this study in innumerable ways and saved us from many errors.

ABSTRACT

From the ancient age, gestures were the first mode of communication, after the evolution of human civilization they developed verbal communication, but still non-verbal communication is equally significant. Such non-verbal communication is not only used for physically challenged persons but also it can be efficiently used for various applications such as 3D gaming, aviation, surveying, etc. This is one of the methods to interact with computers with least interactive peripheral devices. Many researchers are still developing robust and efficient new hand gesture recognition techniques. The major steps associated with the gesture mode of communication associated while designing the system are: data acquisition, segmentation and tracking, feature extraction and gesture recognition. There are different methodologies associated with several substeps present at each step.

This project presents the design of the CNN model to improve the accuracy of Hand gesture recognition along with discussion on its applications from use in cars to control its function to complete music and feature control of a computer.

Chapter 1

Introduction

1.1 Background

The idea of gesture recognition is to develop a technique which can identify particular human gestures and use them to transfer information. In the gesture recognition method, a camera displays the human body movements and communicates the data to a computer that uses the gestures as input to control gadgets or applications. The idea of developing hand gesture recognition procedures is to develop an interaction between human and computer and the recognized gestures are used to control the meaningful information. The principal components of a hand gesture recognition process are data acquisition, hand localization, hand characteristics recognition and gesture identification. Gesture is an indication of physical behavior or emotional expression. It consists of body gestures and hand gestures.

It is divided into two categories

- Static gesture
- Dynamic gesture.

Static hand gesture recognition performed without any additional devices which are used to give instructions to machines. A person commands the machine using his bare hands, the person's hand's images are captured. and analyzed to determine the meaning of hand gesture. To recognize static gestures a common classifier or template matcher is used. A dynamic movement is supposed as a route between the first phase and last phase.

Dynamic gesture recognition characterizes the hand movements. It has four features - velocity, shape, location and orientation. Gesture recognition has many uses in coming times like: - in medical application, entertainment application, automated systems, better life for the disabled people..

As the technology matures, gesture recognition as its application will move beyond infotainment and will allow drivers to control other systems within the vehicle, such as heating and cooling, and to connect with smart home systems which is one of the applications of the technology. For example, imagine being able to check home security camera as one drive home by simply making a hand gesture. Gestures could also be coupled with telematics systems, allowing the vehicle to provide information about nearby landmarks if it recognizes that an occupant is pointing at it.

Concept of smart car (application of hand gesture)

- Function control by hand gestures , facial gestures and climate gestures.
- Functions of the car like Volume controls , Map guidance , Music Playback , Call Assists can be controlled by Finger or hand gestures to a certain limit.
- Facial gestures will control the activeness of the driver , his/her concentration and mood recognition for safety(like drowsiness results in safe stopping of the car).

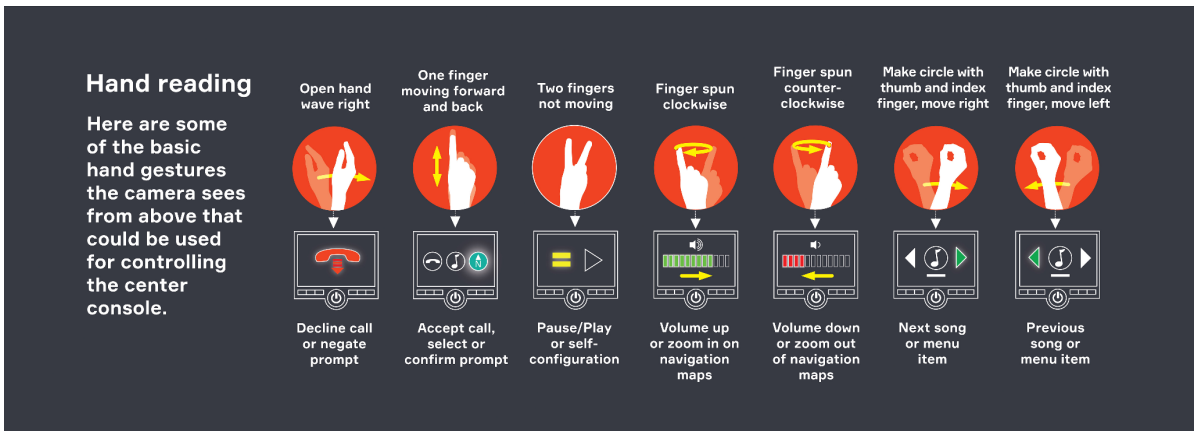


Figure 1.1 - List of hand gestures for different controls

1.2 Methods of hand gesture recognition [6]

- **Pixel to Pixel Comparison**: This method involves the pixel by pixel comparison of the frame captured and with the image database. This method is not so accurate but easy to implement. Prerequisite to the

comparison image should be segmented out from the virtual environment. This can be done by selecting the threshold using Otsu's method.

- **Edges Method**: The objective of this method is to find out the highest gradient in an image and this is found by applying the threshold in the gradients. Threshold will remove the low magnitude gradients. The magnitude of gradient is the sum of derivatives in x and y directions.
- **Using Orientation Histogram**: Orientation histogram is dependent on the feature vectors, resulting in formation of histogram based on the edges of the image. First the system is trained with the images. As images are captured by the webcam as the input these images are converted to the gray scale and from this the histogram is created which is used as the training pattern. There onwards the same steps are followed for recognition i.e capture of image, conversion to grayscale and histogram calculation. The advantage of this method is that it is very fast, robust and translation invariant but the disadvantage is dependent on rotation.
- **Thinning Method**: In order to find the histogram of image the center of image is taken as the reference. Assuming the window at the center of the image, which is in RGB format, needs to be converted to YCbCr. There is a range of Cb and Cr. If the pixel is in the specified range, then it should be converted to white and the rest of the pixel is turned to black. Results in the output image as a gray scale image; next step is to convert this gray scale image to binary which is done by selecting the threshold using the Otsu's method. Thus the binary is then thinned. While undergoing this procedure, noise and undesirable segments were added to it. So those noises should be removed.

1.3 HAND GESTURE TECHNOLOGIES

Gestures are most considered as the most natural and expensive way of communication between computer and human in a virtual environment and it is the only powerful way of communication among humans. The major constraint is interfacing gesture movement with the computer. For initializing the system, the first step is to collect the gesture data in order to accomplish the specific application. For gesture recognition systems and hand posture different technologies are discovered for acquiring the input data.

Vision based approaches: Recognition system only requires the camera to capture the image for natural interaction between humans and computers. That is more useful in real time applications. Although this technique is simple there are lots of challenges while implementing such as complex background, lighting conditions and variations, and other skin color objects with the hand object. Vision based technology deals with the characteristics of image such as texture and color that are required for the identification of gesture. There are many techniques evolved for detecting hand objects after the image is pre-processed, these methods are divided into two parts.

1) Appearance based approaches: In this approach, the visual appearance of the input hand image is modeled using the feature extraction, which are then compared to the feature extracted of the stored image. This method has the advantage of real time performance and is easier as compared to that of a 3D model based approach. The general method of this approach is to detect skin colored regions in an image. Recently there is a lot of research taking place that applies invariant features such as the Adaboost learning algorithm. Using this invariant features that enables the identification of points and regions on the hand, rather than modeling the complete hand. The positive point of this method is it overcomes the problem of occlusion.

2) 3D model based approaches: In this approach for modeling and analysis of hand shape the 3D model is used. While doing the 2D projection some data is lost, So intended a depth parameter is added in the 3D model to make it more accurate. 3D models are classified into volumetric and skeletal models. These two models have constraints while designing. Volumetric model is used in real time application as it deals with the 3D model base appearance of the human hand. The main constraint while designing this system is, it has the huge dimensionality parameters which cause the designer to work in 3 dimensional. Skeletal models overcome the volumetric model by limiting the parameters which are going to model the hand shape from 3D structure.Sparse coding is a complex feature optimization which will provide the high efficiency for high dimension description. It is a new technique which has first-tier precision as compared to HOG-DTF and DSR descriptors. After the sparse signal compressive sensing is obtained. Compressive sensing is used to recover the sparse signal from few observations. Because of this the resource consumption is reduced.

3) Gloved based approaches: Glove approach uses the sensors for capturing the hand position and motion. Detection of the hand is done by the sensors on hand and the correct coordinates of location of palm and fingers is to be found using the sensors on the gloves. This approach is cumbersome as the user needs to be connected to the computer physically. Even though the sensors required for this interaction are expensive.

4) Marked Colored glove approach: In this approach the gloves which are to be worn by the human hand are made marked by the colors to direct the process of tracking the hand and locating the palm and fingers, which will provide the exact geometric features resulting in formation of hand shape.

Techniques/ Methods	Learning theory	Remarks & Accuracy
MRS-CRF algorith	Probabilistic models	Accuracy:90.45% Efficient for dynamic hand gesture recognition system. Accurate as compared to CRF algorithm (85.3%)
Superpixel Earth movers distance (SP-EMD)	Novel distance metric to overcome the partial matching problem	Accuracy:98.8% Distance measurement between two hand gestures based on shape and texture
Random Fores	Machine learning model	Accuracy: 94.33% Decision making trees for training samples of the training sets
Fusing frame image	Hierarchical identification model	Accuracy: 90% Rough hand gesture recognition
Non-linear SVM and Bag-of-features(BOF)	Statistical learning theory	Accuracy:97% Does not need hand segmentation and tracking.
Support vector machine	Statistical learning theory	Accuracy:99.2%
Self-organizing Maps	Automatic learning theory	----
Hidden Markov model	Machine learning theory	Accuracy:93.7%

TABLE 1 Recognition Techniques [10]

Chapter 2

TensorFlow

2.1 Basics of Tensorflow

TensorFlow is designed to make building neural networks for machine learning easy. TensorFlow can Build and train ML models easily using intuitive high-level APIs like Keras with eager execution, which makes for immediate model iteration and easy debugging. It can also easily train and deploy models in the cloud, on-prem, in the browser, or on-device no matter what language one use. TensorFlow is a simple and flexible architecture to take new ideas from concept to code, to state-of-the-art models, and to publication faster.

2.1.1 Advantages of TensorFlow:

1. Open-source platform

It is an open-source platform that makes it available to all the users around and ready for the development of any system on it.

2. Data visualization

TensorFlow provides a better way of visualizing data with its graphical approach. It also allows easy debugging of nodes with the help of TensorBoard. This reduces the effort of visiting the whole code and effectively resolves the neural network.

3. Keras friendly

TensorFlow has compatibility with Keras, which allows its users to code some high-level functionality sections in it. Keras provides system-specific functionality to TensorFlow, such as pipelining, estimators, and eager execution.

The Keras functional API supports a variety of topologies with different combinations of inputs, output, and layers.

4. Scalable

Almost every operation can be performed using this platform. With its characteristic of being deployed on every machine and graphical representation of a model allows its users to develop any kind of system using TensorFlow.

Hence TensorFlow has been able to develop systems like Airbnb, Dropbox, Intel, Snapchat, etc.

5. Compatible

It is compatible with many languages such as C++, JavaScript, Python, C#, Ruby, and Swift. This allows a user to work in an environment they are comfortable in.

6. Parallelism

TensorFlow finds its use as a hardware acceleration library due to the parallelism of work models. It uses different distribution strategies in GPU and CPU systems.

A user can choose to run its code on either of the architectures based on the modeling rule. A system chooses a GPU if not specified. This process reduces the memory allocation to an extent.

7. Architectural support

TensorFlow also has its architecture TPU, which performs computations faster than GPU and CPU. Models built using TPU can be easily deployed on a cloud at a cheaper rate and executed at a faster rate.

8. Graphical support

Deep learning uses TensorFlow for its development as it allows building neural networks with the help of graphs that represent operations as nodes.

TensorFlow acts in multiple domains such as image recognition, voice detection, motion detection, time series, etc hence it suits the requirement of a user.

2.1.2 Disadvantages of TensorFlow

1. Frequent updates

TensorFlow releases different updates every 2-3 month, increasing the overhead for a user to install it and bind it with the existing system.

2. Inconsistent

TensorFlow provides homonyms that share similar names but different implementations, which makes it confusing to remember and use. For eg: tf.nn.conv2d, tf.nn.convolution, tf.layers.conv2d, tf.layers.Conv2d has varying meanings and often makes it inconsistent with its usability.

3. Architectural limitation

TensorFlow's architecture TPU only allows the execution of a model not to train it.

4. Dependency

Although TensorFlow reduces the length of code and makes it easier for a user to access it, it adds a level of complexity to its use. Every code needs to be executed using any platform for its support which increases the dependency for the execution.

5. Symbolic loops

TensorFlow lags at providing the symbolic loops for indefinite sequences. It has its usage for definite sequences, which makes it a usable system. Hence it is referred to as a low-level API.

6. GPU Support

TensorFlow has only NVIDIA support for GPU and python support for GPU programming. It does not have any other support.

7. Slow speed

TensorFlow has low speed with respect to its competitors. It has less usability in comparison to other frameworks.

8. Support for Windows

TensorFlow does not provide many features for the Windows Operating System users. It opens a wide range of features for the Linux users. But still, Windows users can download TensorFlow using the anaconda prompt or using the pip package.

2.1.3 Object Detection using SSD Mobilenet and Tensorflow Object Detection

The SSD architecture is a single convolution network that learns to predict bounding box locations and classify these locations in one pass. Hence, SSD can be trained end-to-end. The SSD network consists of base architecture (MobileNet in this case) followed by several convolution layers:

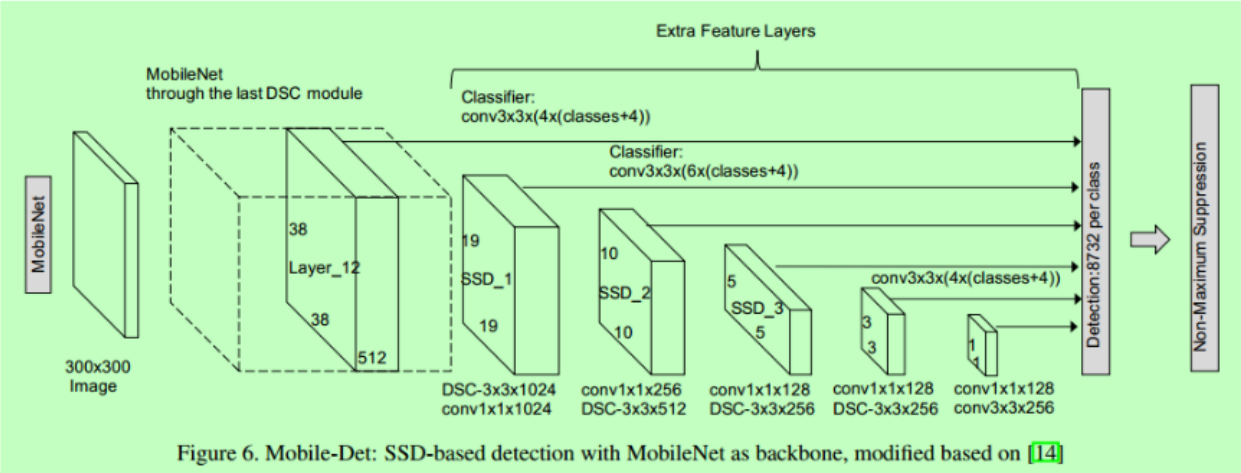


Figure 6. Mobile-Det: SSD-based detection with MobileNet as backbone, modified based on [14]

Figure 2.1 - Layers of SSD algorithms

By using SSD, we only need to take one single shot to detect multiple objects within the image, while regional proposal network (RPN) based approaches such as R-CNN series that need two shots, one for generating region proposals, one for detecting the object of each proposal. Thus, SSD is much faster compared with two-shot RPN-based approaches.

The TensorFlow object detection API is the framework for creating a deep learning network that solves object detection problems.
There are already pretrained models in their framework which they refer to as Model Zoo. This includes a collection of pretrained models trained on the COCO dataset, the KITTI dataset, and the Open Images Dataset.model

2.2 RCNN

2.2.1 Basics of RCNN

RCNN is one deep learning approach, **regions with convolutional neural networks** (R-CNN), combining rectangular region proposals with convolutional neural network features. R-CNN is a two-stage detection algorithm. The first stage identifies a subset of regions in an image that might contain an object. The second stage classifies the object in each region.

2.2.2 Applications for R-CNN object detectors include

- Autonomous driving
- Smart surveillance systems
- Facial recognition

Models for object detection using regions with CNNs are based on the following three processes:

- Find regions in the image that might contain an object. These regions are called region proposals.
- Extract CNN features from the region proposals.
- Classify the objects using the extracted features.

There are three variants of an R-CNN. Each variant attempts to optimize, speed up, or enhance the results of one or more of these processes.

The R-CNN detector first generates region proposals using an algorithm such as Edge Boxes. The proposal regions are cropped out of the image and resized. Then, the CNN classifies the cropped and resized regions. Finally, the region proposal bounding boxes are refined by a support vector machine (SVM) that is trained using CNN features.

2.2.3 Advantages

An RNN remembers each and every information through time. It is useful in time series prediction only because of the feature to remember previous inputs as well. This is called Long Short Term Memory.
Recurrent neural networks are even used with convolutional layers to extend the effective pixel neighborhood.

2.2.4 Disadvantages

Gradient vanishing and exploding problems.
Training an RNN is a very difficult task.
It cannot process very long sequences if using tanh or relu as an activation function.

Chapter 3

Preparation of Dataset

Object detection is one of the most fascinating aspects of computer vision as it allows to detect each individual object in images, locate their position and size as well relative to the rest of the image. Today's state-of-the-art object detection models are powered by **Deep Learning** and at the very soul of training deep learning networks is the training dataset. Training dataset are images collected as samples and annotated for training deep neural networks. For object detection, there are many formats for preparing and annotating our dataset for training. The most popular formats for annotating our datasets are - Pascal VOC and Microsoft Coco.

The thing is, all datasets has some flaws based on particular use case, environment or some other factors. That's why data preparation is such an important step in the machine learning process. In a nutshell, data preparation is a set of procedures that helps make our dataset more suitable for machine learning. In broader terms, the data prep also includes establishing the right data collection mechanism. And these procedures consume most of the time spent on machine learning.

3.1 Step 1 : Collecting images

At the beginning we want our model to train at four different gestures that are

- Swing
- Thumbs up
- Palm
- Right Indication
- Left Indication

To train models we are collecting images using openCV that enables us to collect and organize images in an efficient way.

```
labels = ['Right-Indicator',  'Left-Indicator', 'Call-Pick', 'Call-Decline']
number_images = 15
```

```
for label in labels:
    !mkdir{'.\Tensorflow\workspace\images\collectedimages\\'+label}
    cap = cv2.VideoCapture(0)
    time.sleep(5)
    print ('Capturing ' + label)
    for img_num in range(number_images):
        print('Image #' + str(img_num) + 'starting...')
        ret, frame = cap.read()
        image_name = os.path.join(IMAGES_PATH, label, label + '.'+'{}.jpg'.format(str(uuid.uuid1())))
        cv2.imwrite(image_name, frame)
        cv2.imshow('frame', frame)
        print('capture | 5 sec sleep')
        time.sleep(5)

        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    cap.release()
```

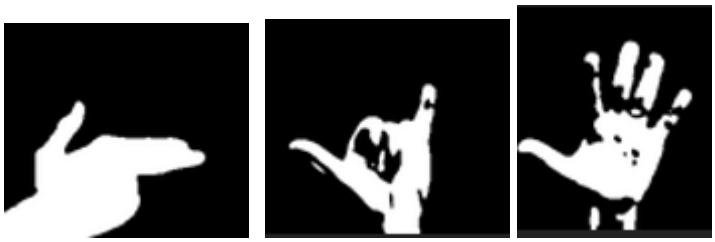
```
A subdirectory or file .\Tensorflow\workspace\images\collectedimages\Right-Indicator already exists.

Capturing Right-Indicator
Image #0starting...
capture | 5 sec sleep
Image #1starting...
capture | 5 sec sleep
Image #2starting...
capture | 5 sec sleep
```

Figure 3.1 - Python script to capture dataset using openCV

```
C:\Users\ajeya\Desktop\gesture\Dataset>tree
Folder PATH listing for volume OS
Volume serial number is 1EF5-CDC3
C:.
├───FistImages
├───FistTest
├───LeftImages
├───LeftTest
├───PalmImages
├───PalmTest
├───RightImages
├───RightTest
├───SwingImages
├───SwingTest
├───ThumbImages
└───ThumbTest
```

Figure 3.2 - Folder structures of stored gestures



(Right Indication) (Swing) (Palm)

Figure 3.3 - Sample dataset

3.2 Step 2 : Installing LabelIng

The Pascal VOC format uses XML files to store details of the objects in the individual images. To easily generate these XML files for the images, we will be using an easy to use tool by the name LabelImg that allows to

- draw visual boxes around our objects in the images
- and it automatically saves the XML files for our images

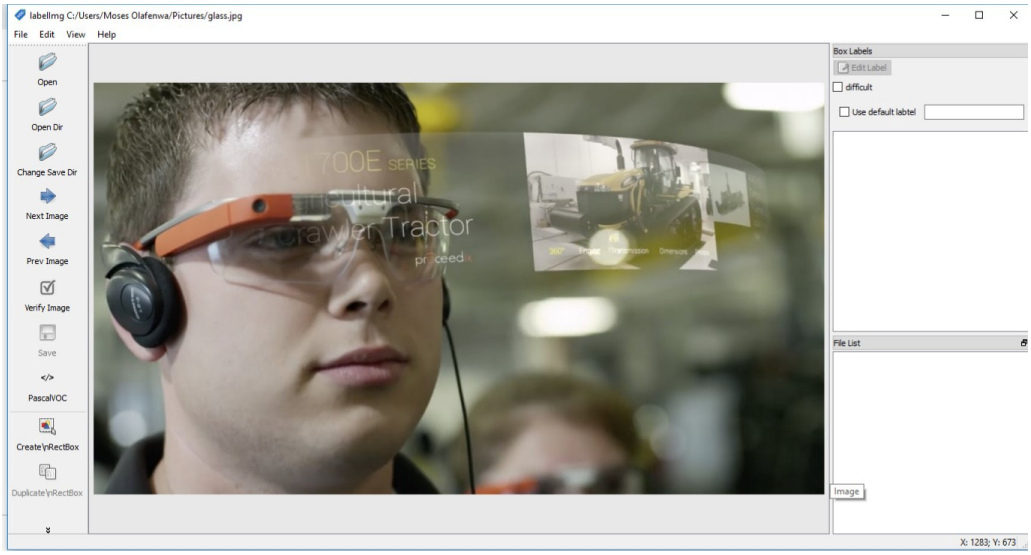


Figure 3.4 - LabelImg UI used for annotation

Installing labelIng in windows has following steps

- Install **PyQt5** and **lxml** via **PIP**

```
pip3 install pyqt5 lxml
```

- Download, unzip and run the **LabelImg.exe** file via https://www.dropbox.com/s/kqoxr10l3rkstqd/windows_v1.8.0.zip?dl=1

3.3 Step 3 : Annotating dataset

Now that we have loaded our images and set the save folder for the annotations. Start annotating our images by:

- Click on the “**Create RectBox**” button on the left-bottom and draw a box around the objects that need to annotated as seen in the images below.
- Then once drawn, enter the name of the object in the **pop-up** box and select the “**OK**” button as seen in the example below

- Click on the “**Create \nRectBox**” button again and annotate all the objects in the image.

- Once done, click the “**Next Image**” button on the middle-left to annotate another image.

As we are annotating our images, the XML file containing box annotations are saved for each image in the “annotations” folder. See the picture below for the corresponding annotation XML files generated.

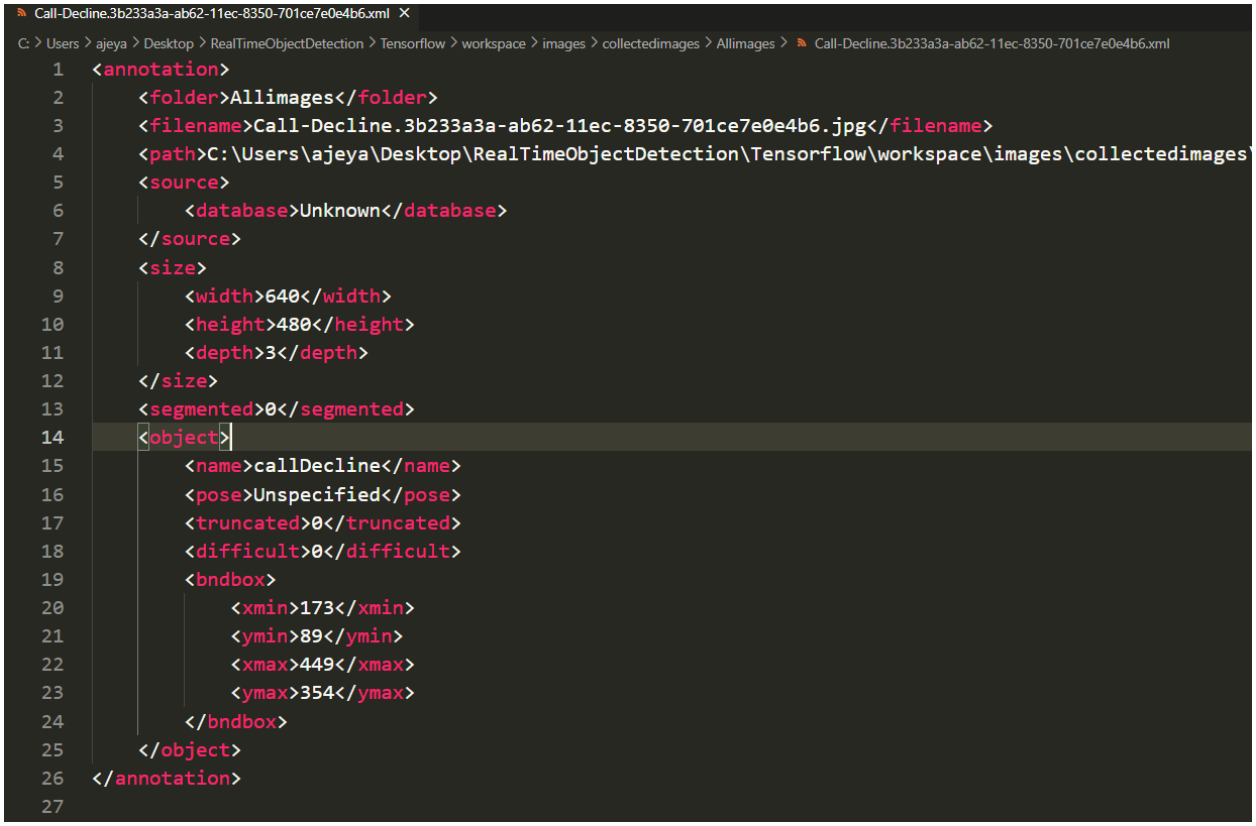


Fig 3.5 - XML file containing box annotations

Next, we'll set up the environment. The first thing is installing OpenCV on a Raspberry Pi 3 with all the dependencies.

The next step is to set up and enable the camera. We also need to install a python module named picamera[array].

The module provides an interface to represent images from the camera as NumPy arrays.

Chapter 4

Background Elimination

Problem Statement

We are going to recognize hand gestures from a video sequence. To recognize these gestures from a live video sequence, we first need to take out the hand region alone, removing all the unwanted portions in the video sequence. After segmenting the hand region, we then count the fingers shown in the video sequence to instruct a robot based on the finger count. Thus, the entire problem could be solved using 2 simple steps [5] :-

1. Find and segment the hand region from the video segment.
2. Count the number of fingers from the segmented hand region in the video sequence.

Segment the Hand Region

The first step is to find the hand region by eliminating all the other unwanted portions(regions) in the video sequence using Python and Tensorflow.

Background Subtraction

We use the concept of Running Averages to separate foreground from background. We make our system to look over a particular scene for 30 frames. During this period , we compute the running average over the current frame and the previous frames. By doing this, we essentially tell our system that the video sequence that started (running average of those 30 frames) is the background.

After figuring out the background , we bring in hand and make the system understand that our hand is a new entry in the background, i.e, it becomes foreground.

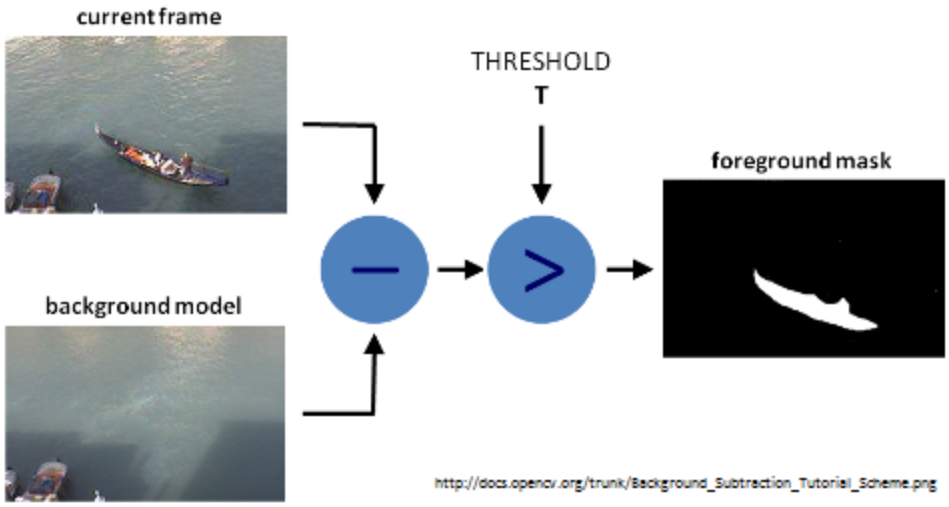


Figure 4.1 : Background Subtraction

After figuring out the background model using running averages, we will use the current frame with a foreground object (i.e. hand). Now the calculation of absolute difference between the background model and the current frame.

Motion Detection and Thresholding

Now to detect the hand region from the difference image, we need to threshold the difference image to make only the hand region visible and all other unwanted regions are painted black(eliminated).

Note: Thresholding is the assignment of pixel intensities to 0's and 1's based a particular threshold level so that our object of interest alone is captured from an image.

Contour Extraction

After thresholding the difference image, we find contours in the resulting image. The contour with the *largest area* is assumed to be our hand.

So, our first step to find the hand region from a video sequence involves three simple steps.

1. Background Subtraction

2. Motion Detection and Thresholding

3. Contour Extraction

We have our function that is used to compute the running average between the background model and the current frame. This function takes in two arguments - current frame and aWeight, which is like a threshold to perform running average over images. If the background model is None (i.e if it is the first frame), then initialize it with the current frame. Then, compute the running average over the background model and the current frame
using cv2.accumulateWeighted() function. Running average is calculated using the formula given below –

$$dst(x,y) = (1-a).dst(x,y) + a.src(x,y)dst(x,y)$$

- **src(x,y)** - Source image or input image (1 or 3 channel, 8-bit or 32-bit floating point)
- **dst(x,y)** - Destination image or output image (same channel as source image, 32-bit or 64-bit floating point)
- **a** - Weight of the source image (input image)

Our next function is used to segment the hand region from the video sequence. This function takes in two parameters - current
frame and threshold used for thresholding the difference image.

- First, we find the absolute difference between the background model and the current frame using cv2.absdiff() function.
- Next, we threshold the difference image to reveal only the hand region. Finally, we perform contour extraction over the thresholded image and take the contour with the largest area (which is our hand).
- We return the thresholded image as well as the segmented image as a tuple. The math behind thresholding is pretty simple. If x(n)x(n) represents the pixel intensity of an input image at a particular pixel coordinate, then threshold decides how nicely we are going to segment/threshold the image into a binary image.

$$x(n)=\{1,0\}$$

Instead of recognizing gestures from the overall video sequence, we will try to minimize the recognizing zone (or the area), where the system has to look for hand region. To highlight this region, we use cv2.rectangle() function which needs top, right, bottom and left pixel coordinates.

To keep track of frame count, we initialize a variable num_frames. Then, we start an infinite loop and read the frame from our webcam using camera.read() function. We then resize the input frame to a fixed width of 700 pixels maintaining the aspect ratio using imutils library and flip the frame to avoid mirror view.

Next, we take out only the region of interest (i.e the recognizing zone), using simple NumPy slicing. We then convert this ROI into a grayscale image and use gaussian blur to minimize the high frequency components in the image. Until we get past 30 frames, we keep on adding the input frame to our run_avg function and update our background model. Please note that, during this step, it is mandatory to keep the camera without any motion. Or else, the entire algorithm fails.

After updating the background model, the current input frame is passed into the segment function and the thresholded image and segmented image are returned. The segmented contour is drawn over the frame using cv2.drawContours() and the thresholded output is shown using cv2.imshow().

Finally, we display the segmented hand region in the current frame and wait for a keypress to exit the program. Notice that we maintain bg variable as a global variable here. This is important and must be taken care of.

Executing Code

Remember to update the background model by keeping the camera static without any motion. After 5-6 seconds, show our hand in the recognizing zone to reveal our hand region alone. Below realising how our system segments the hand region from the live video sequence effectively.

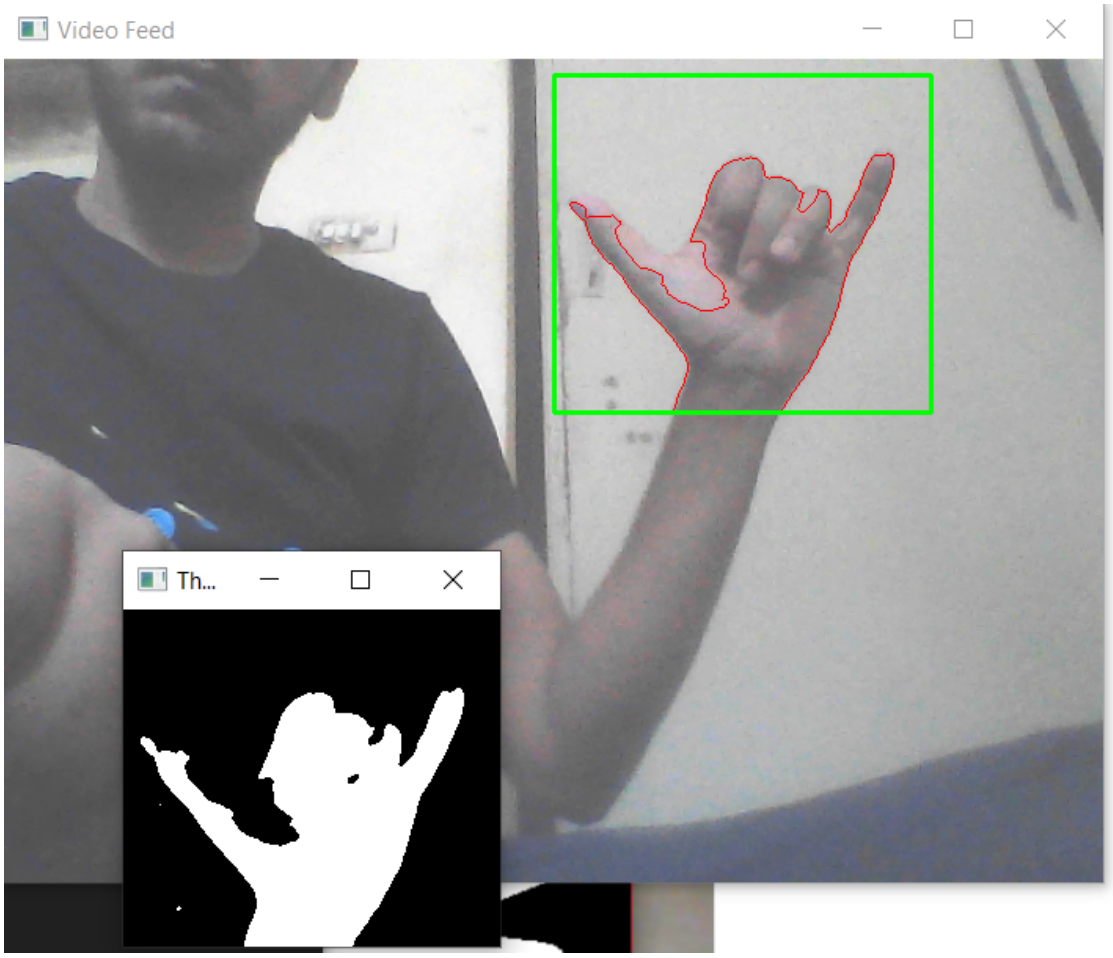


Figure 4.2 - Segmenting hand region

Chapter 5

Training and Testing the model

Artificial Neural Networks

Artificial neural networks are structures widely used for classification tasks. When using this mechanism, the object to be classified is presented to the network through the activation of artificial neurons in the input layer. These activations are processed in the inner layers, and the result emerges as a pattern in the output layer. A network that has a single layer is known as a simple perceptron but is only capable of solving linearly separable problems. In order to solve non linearly separable problems, it is necessary to use a multilayer perceptron (MLP) neural network. The MLP consists of an input layer, a number of hidden layers, and an output layer, as can be seen in Figure 1.

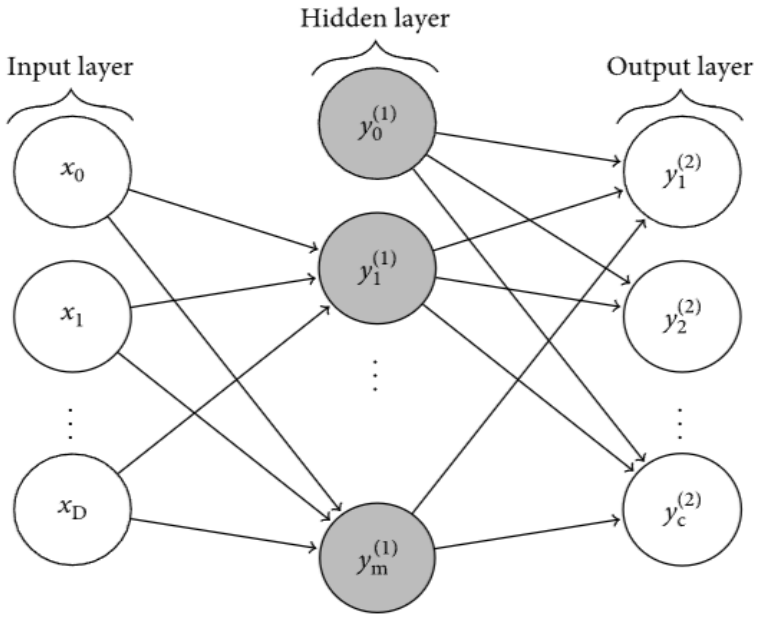


Figure 5.1

Example of MLP. Image adapted from Stutz

The adequacy of the neural network to a classification task is determined by its weights. The training of a neural network consists of adapting the weights of the artificial synapses to a specific problem. In order to train an MLP, one of the most common methods is to use the backpropagation algorithm, in which the connection weights in the inner layers are modified as the error is propagated in the reverse direction, with the purpose of adapting the neural network to the resolution of the problem.

Convolutional Neural Networks

Convolutional neural networks, or CNNs, are widely used for image classification, object recognition, and detection]. Three types of layers can summarize its structure: convolution, pooling, and classification, as shown in Figure 5.1. The CNN architecture must be defined according to the application and is usually defined by the number of alternate convolution and pooling players, number of neurons in each layer, and choice of activation function. [10]

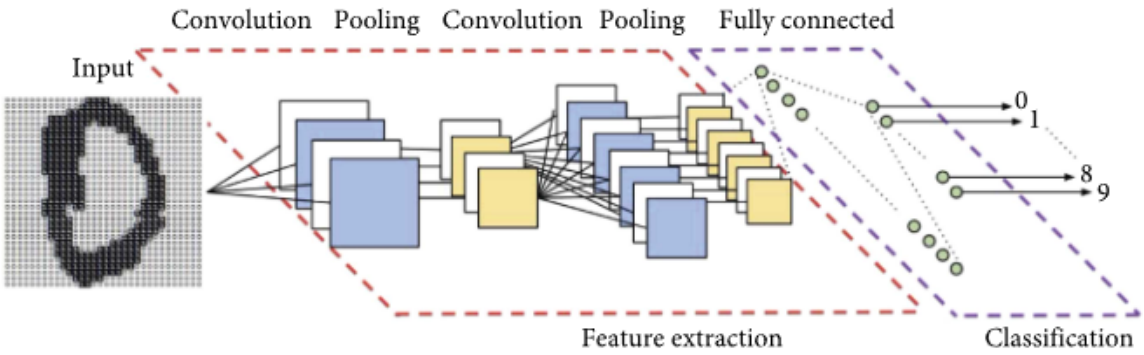


Figure 5.2

Example of CNN and its layers.

In the context of image classification, the input for a CNN is an image represented by an arbitrary color model. At the convolution layer, every neuron is associated with a kernel window that is convolved with the input image during CNN training and classification. This convolution kernel is composed of the weights of each associated neuron. The output of this convolution step is a set of *N* images, one for each of the *N* neurons. Because of convolution, these new images can contain negative values. In order to avoid this issue, a rectified linear

unit (ReLU) is used to replace negative values by zero. The outputs of this layer are called feature maps.

After a convolution layer, it is common to apply a pooling layer. This is important because pooling reduces the dimensionality of feature maps, which subsequently reduces the network training time. Some architectures alternate between convolution and pooling, for example, GoogLeNet [30] has five convolution layers followed by one pooling layer. At the end of the convolution and pooling architectures, there is a multilayer perceptron neural network that performs classification based on the feature maps computed by the previous layers.

Because of its large number of layers and successful applications, CNNs are one of the preferred techniques for deep learning. Its architecture allows automatic extraction of diverse image features, like edges, circles, lines, and texture. The extracted features are increasingly optimized in further layers. It is important to emphasize that the values of the kernel filters applied in the convolution layers are the result of backpropagation during CNN training.

The Deep Convolutional Neural Network

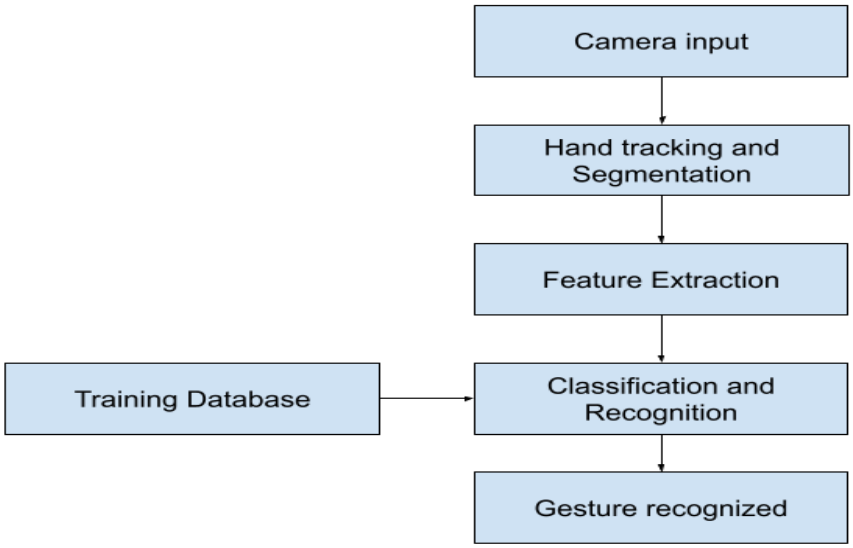


Figure 5.3 - Steps for training dataset

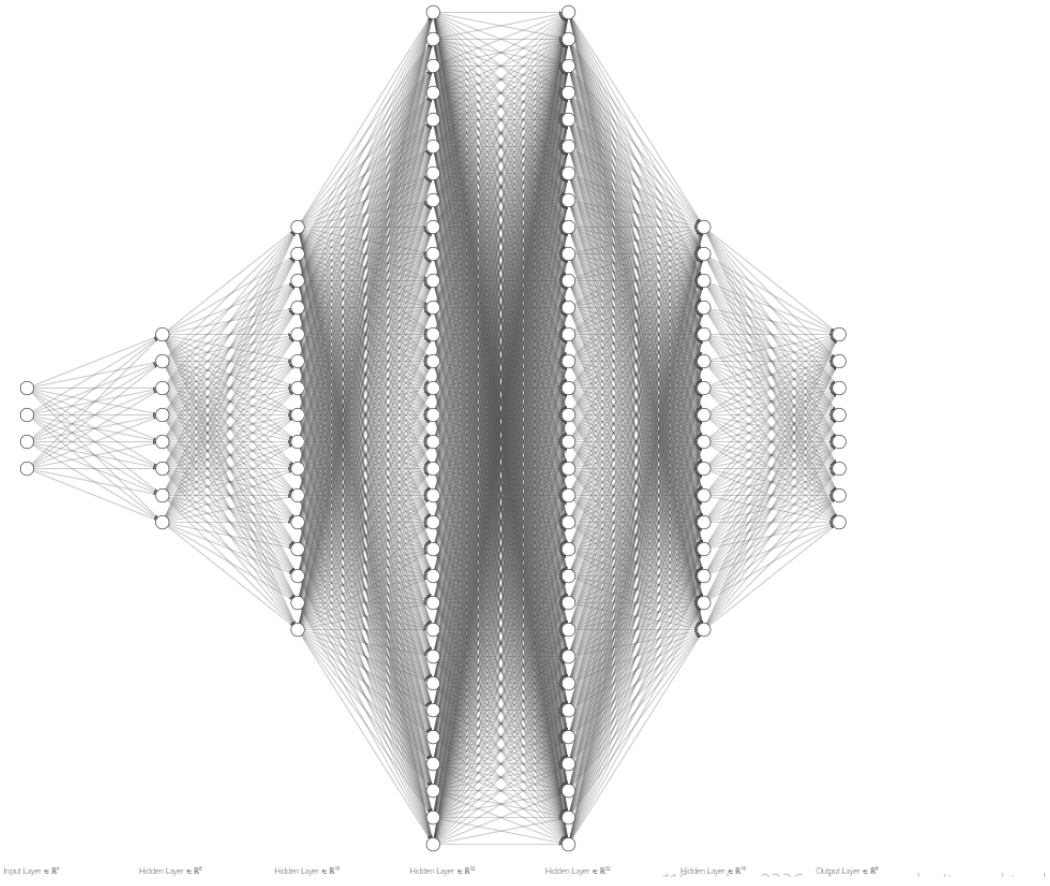
The network contains **7** hidden convolution layers with **Relu** as the activation function and **1** Fully connected layer.

The network is trained across **50** iterations with a batch size of **64**.

We saw that 50 iterations trains the model well and there is no increase in validation accuracy along the lines so that should be enough.

The model achieves an accuracy of 96.6% on the validation dataset.

The ratio of training set to validation set is 1000 : 100.



CNN Model (scaled down by factor of 4)

Figure 5.5 - Input to CNN model post background elimination

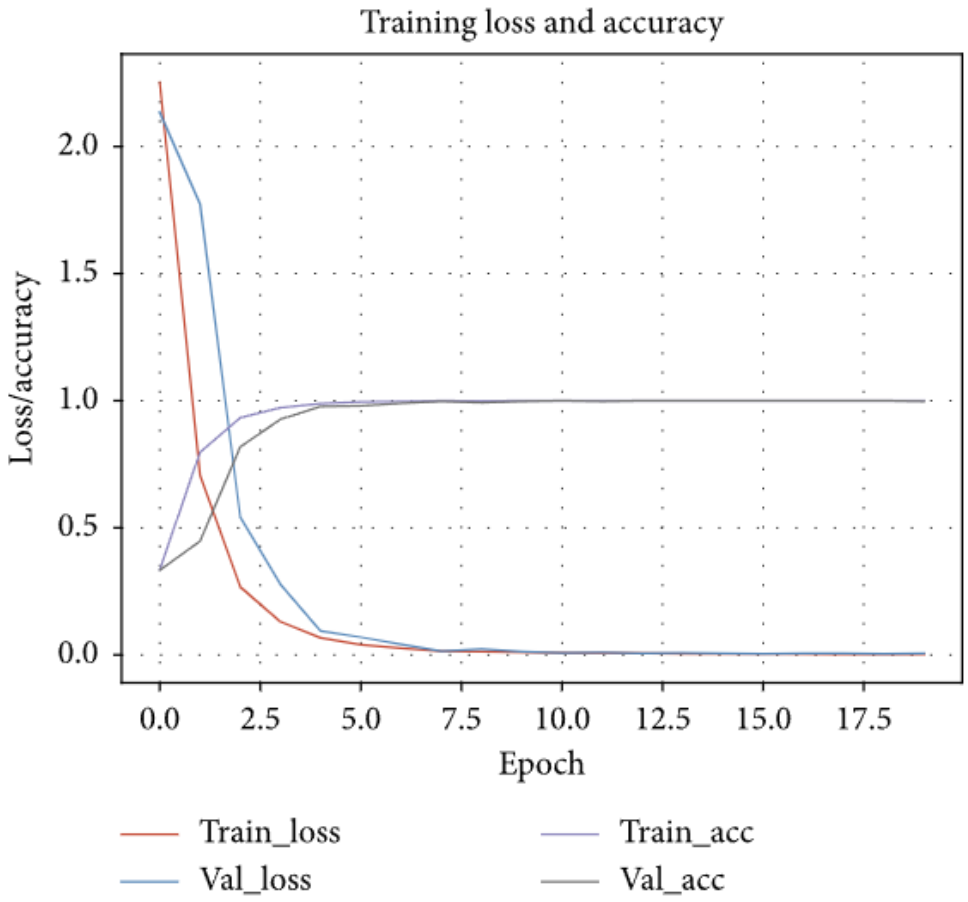


Figure 5.6 - Training statistics for model

Conclusions

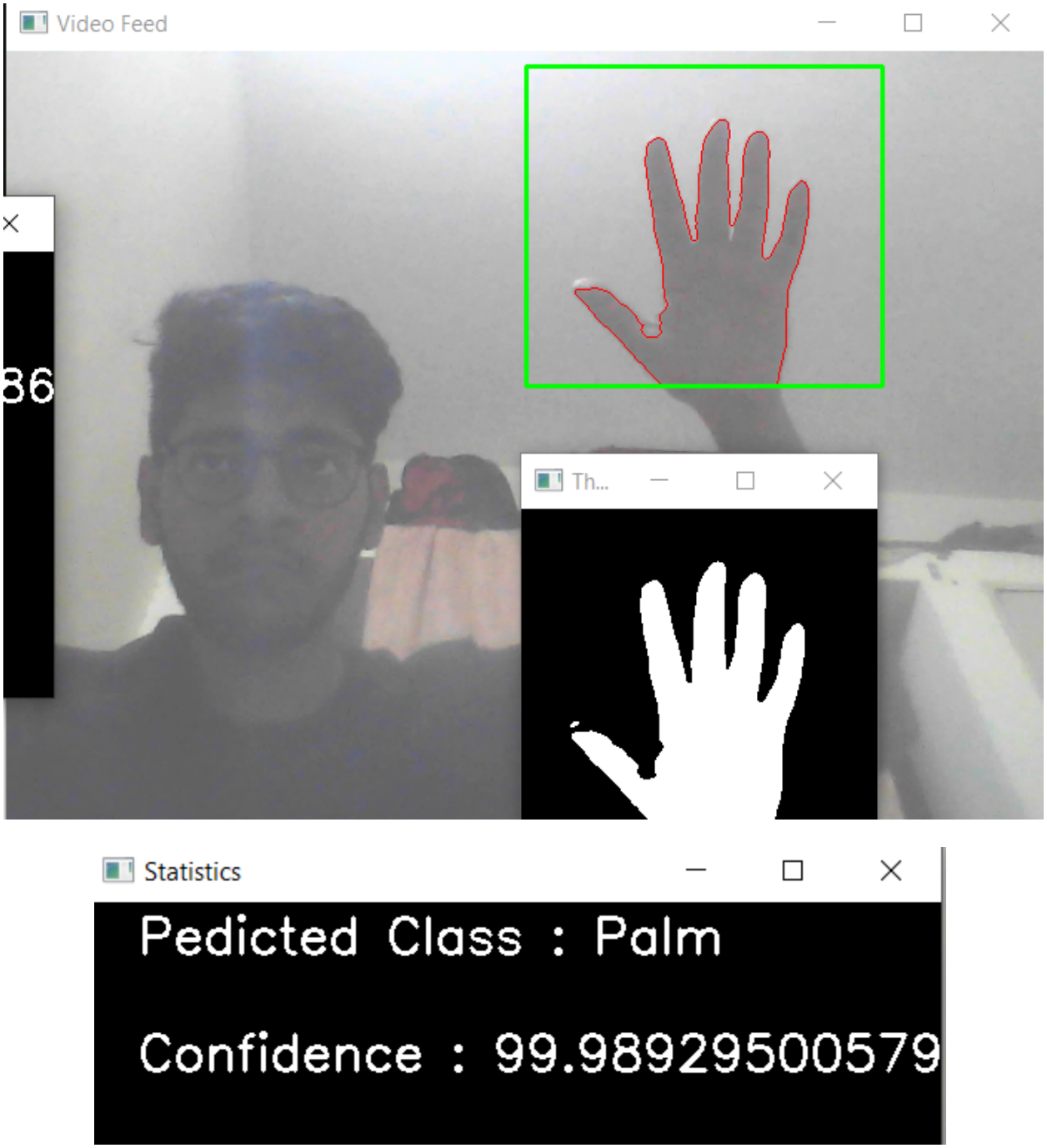


Figure 5.7 - Palm gesture detected correctly

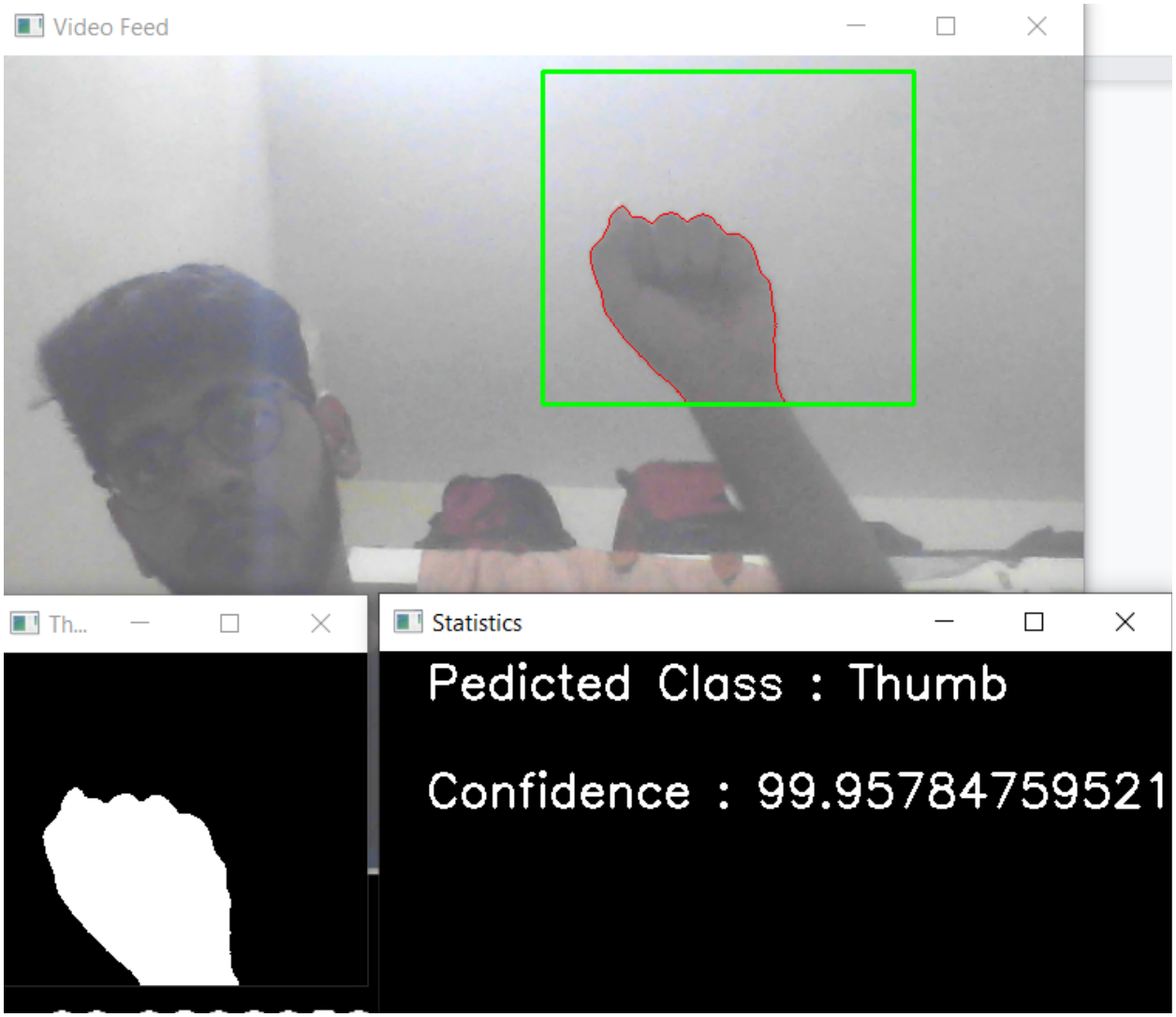


Figure 5.8 - Incorrectly identified gesture

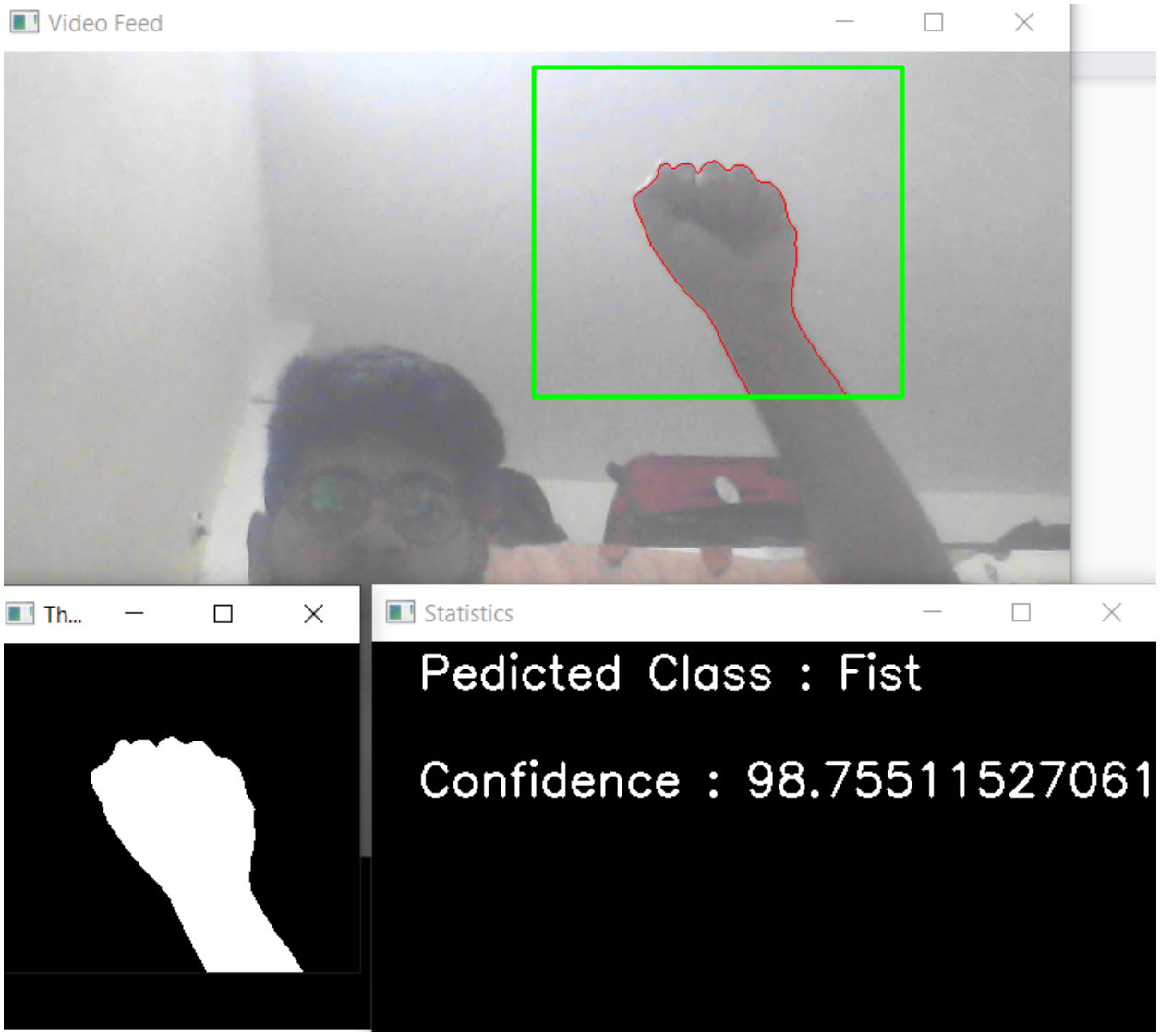


Figure 5.9 - Correctly identified fist

One of the problems in gesture recognition is dealing with the image background and the noise often present in the regions of interest, such as the hand region. The use of neural networks for color segmentation, followed by morphological operations and a polygonal approximation, presented excellent results as a way to separate the hand region from the background and to remove noise. This step is important because it removes image objects that are not relevant to the classification method, allowing the convolutional neural network to extract the most relevant gesture features through their convolution and pooling layers and, therefore, to increase network accuracy. The proposal to make a logical AND operation with the segmentation masks and the original images provided the relevant information of the palms and fingers. Thus, the proposed CNN architectures achieved high success rates at a relatively low computational cost.

It was superior to methodologies mentioned in related works, confirming the robustness of the presented method. In addition, the proposed architectures reached accuracies very similar to the architectures already defined in the literature, although they are much simpler and have a lower computational cost. This is possible due to the proposed image processing methodology, in which unnecessary information is removed, allowing improved feature extraction by the CNN. The proposed methodology and CNN architecture open the door to a future implementation of gesture recognition in embedded devices with hardware limitations. The proposed methodology approaches only cases of gestures present in static images, without hand detection and tracking or cases of hand occlusion. In the future, we intend to work on these particular cases in a new data preprocessing methodology, investigating other techniques of color segmentation] and deep learning architectures.

```
# Shuffle Training Data
loadedImages, outputVectors = shuffle(loadedImages, outputVectors, random_state=0)

# Train model
model.fit(loadedImages, outputVectors, n_epoch=50,
          validation_set = (testImages, testLabels),
          snapshot_step=100, show_metric=True, run_id='convnet_coursera')

model.save("TrainedModel/GestureRecogModel.tfl")
```

```
Training Step: 120  | total loss: 0.06086 | time: 19.870s
| Adam | epoch: 003 | loss: 0.06086 - acc: 0.9971 -- iter: 2944/3000
Training Step: 121  | total loss: 0.05477 | time: 21.241s
| Adam | epoch: 003 | loss: 0.05477 - acc: 0.9974 | val_loss: 0.27714 - val_acc: 0.9267 -- iter: 3000/3000
```

Figure 5.10 - Achieved accuracy

Chapter 6

Speaker Volume Control Using Gestures

Computer vision is becoming a more and more in-demand field as many different fields are currently using this technology such as self-driving cars and industrial automation. We have created and integrated a music controller which uses the webcam to change the volume of the device and also play/pause and skip/repeat songs.

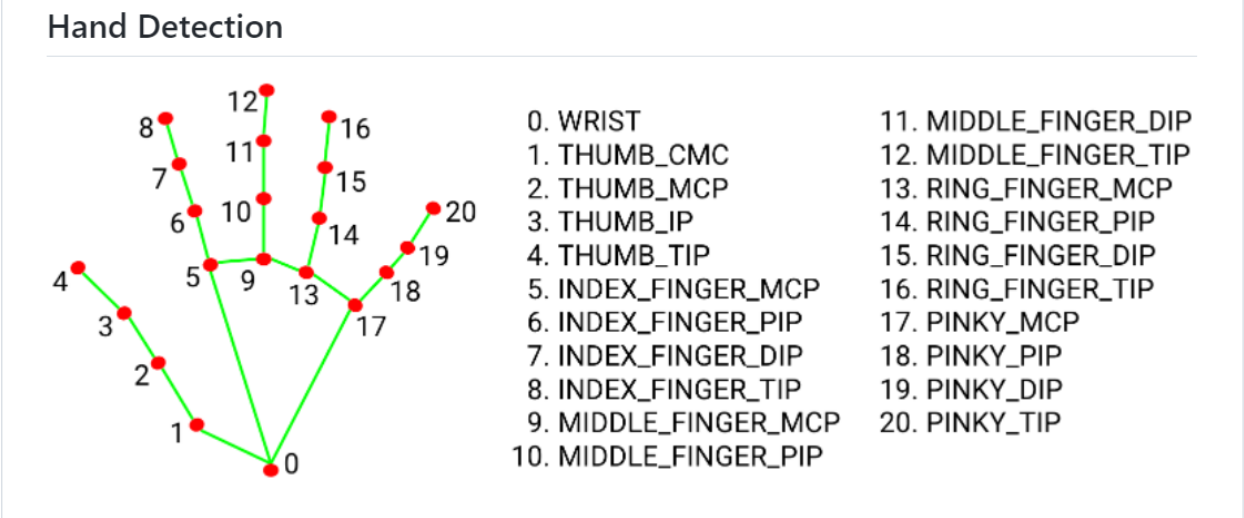


Figure 6.1 - Hand Detection

OpenCV

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code. What can OpenCV do?

OpenCV is a great tool for **image processing and performing computer vision tasks**. It is an open-source library that can be used to perform tasks like face detection, objection tracking, landmark detection, and much more.OpenCV is an essential part of the computer vision community and using it we can build thousands of amazing applications. We are using OpenCv to take input of hand gestures.

Demo

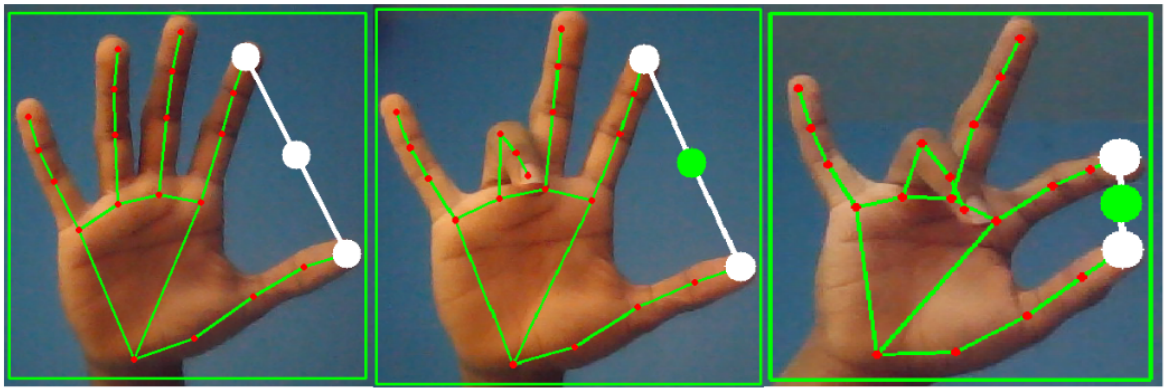


Figure 6.2 - Input demo

Controlling Volume

Step 1:

- Put a hand in the frame until the thumb and index finger connect with a line

Step 2:

- While in this mode volume change will be shown but not executed

Step 3:

- Put down the ring ringer, the middle circle should turn green

Step 4:

- Moving finger and thumb further and closer together will change the system volume

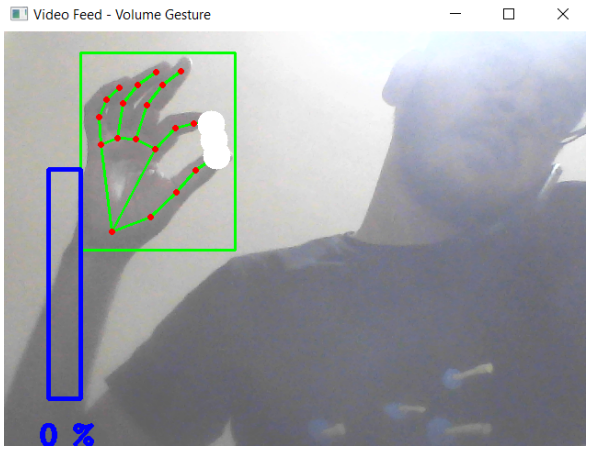


Figure 6.3 (a)

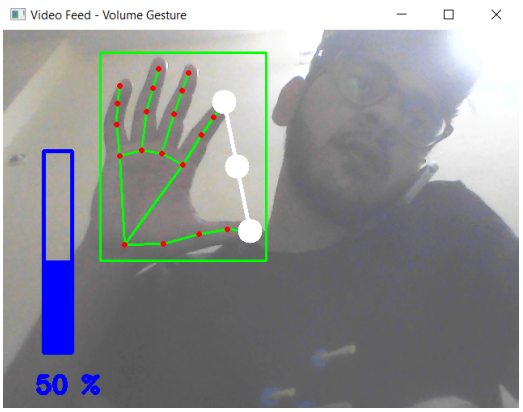


Figure 6.3 (b)

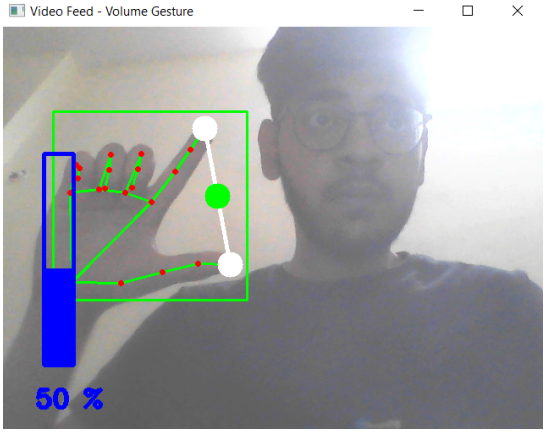


Figure 6.3 (c)

Figure 6.3 - Working demo

[Image a - Volume initialized to 0 as the distance between fingers are 0]

[Image b - Volume increased to 50% based on finger distance]

[Image c - Volume set to 50% based on remaining three fingers]

Chapter 7

Applications

7.1 SmartCars

- Functions of the car like Volume controls, Map guidance, Music Playback , Call Assists can be controlled by Finger or hand gestures to a certain limit.
- Facial gestures will control the activeness of the driver , his/her concentration and mood recognition for safety(like drowsiness results in safe stopping of the car).

How it will work

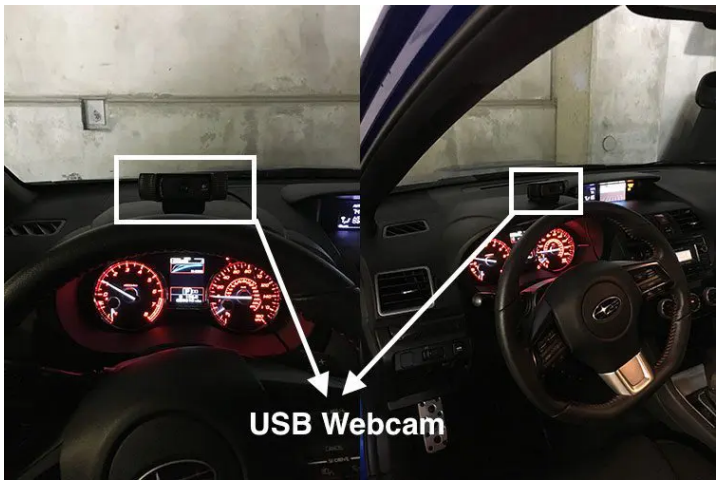


Figure 7.1 - Camera installed in car dashboard

A gesture recognition system starts with a camera pointed at a specific three-dimensional zone within the vehicle, capturing frame-by-frame images of hand positions and motions. This camera is typically mounted in the roof module or other vantage point as shown in Fi1.1 that is unlikely to be obstructed. The system illuminates the area with infrared LEDs or lasers for a clear image even when there is not much natural light.

These images are analyzed in real time by computer vision and machine learning technologies, which translate the palm hand postures and motions into commands, based on a predetermined library of signs. Commands generated by the gesture recognition software become just another type of input, similar to turning a dial, pressing a button or touching a screen. Additionally, as the quantity and quality of cabin cameras improves, other passengers in the vehicle could eventually get on the act.

7.2 Healthcare

Emergency rooms and operating rooms may be chaotic, with lots of noise from personnel and machines. In such environments, voice commands are less effective than gestures. Touchscreens are not an option either, since there's a strict boundary between what is and is not sterile. But accessing information and imaging during surgery or another manipulation is possible with HGR tech, as proven by Microsoft. GestSure provides doctors with the ability to check MRI, CT, and other imagery with simple gestures without scrubbing out.

7.3 Virtual reality

In 2016, Leap Motion (acquired by Ultrahaptics in 2019) presented updated HGR software that allows users, in addition to controlling a PC, to track gestures in virtual reality. The Leap Motion controller is a USB device that observes the area of about one meter with the help of two IR cameras and three infrared LEDs. This controller is used for applications in the medical, automotive, and other fields.

A hand tracking application from ManoMotion recognizes gestures in three dimensions using a smartphone camera (on both Android and iOS) and can be applied in AR and VR environments. The use cases for this technology include gaming, IoT devices, consumer electronics, and robots.

7.4 Consumer electronics

The size of the global gesture recognition market is predicted to grow by $624 million from 2018 to 2022 [19], and companies are trying to catch the opportunities. The Italian startup Limix uses a combination of IoT and dynamic hand gesture recognition to record sign language, translate it into words, and then play them on a smartphone via a voice synthesizer.

Home automation is another broad field within the consumer electronics domain in which gesture recognition is being employed. uSens develops hardware and software to make smart TVs sense finger movements and hand gestures. Gestoos' AI platform with gesture recognition technology offers touchless control over lighting and audio systems. With Gestoo, gestures can be created and assigned via a smartphone or another device, and one gesture can be used to enable several commands.

These days, the consumer market is open for new experiences in HMI, and hand gesture recognition technology is a natural evolution from touchscreens. Demand for smoother and more hygienic means of interaction with devices as well as a concern for driver safety are pushing the adoption of HGR in industries from healthcare to automotive and robotics. And while software development for gesture recognition systems is quite challenging, expertise in AI, deep learning, computer vision, and innovative hardware from top tech providers make HGR solutions more affordable than they were even a few years ago.

References

[1] TensorFlow 2.8.0. TensorFlow February 2, 2022 release GitHub documentation.

[2] .Kobylarz, Jhonatan; Bird, Jordan J.; Faria, Diego R.; Ribeiro, Eduardo Parente; Ekárt, Anikó (2020-03-07). "Thumbs up, thumbs down: non-verbal human-robot interaction through real-time EMG classification via inductive and supervised transductive transfer learning". *Journal of Ambient Intelligence and Humanized Computing*. Springer Science and Business Media LLC.

[3] Object Detection using Tensorflow Colab demonstration using a TF-Hub module trained to perform object detection.

[4] Wei Liu1 , Dragomir Anguelov , Dumitru Erhan SSD: Single Shot MultiBox Detector. Method for detecting objects in images using a single deep neural network

[5] Hand Gesture Recognition using Python and OpenCV (06-04-2020) https://gogul.dev/software/hand-gesture-recognition-p1

[6] Meenakshi Panwar and Pawan Singh Mehra , “Hand Gesture Recognition for Human Computer Interaction”, in Proceedings of IEEE International Conference on Image Information Processing(ICIIP 2011), Waknaghat, India, November 2011.

[7] Amornched Jinda-apiraksa, Warong Pongstiensak, and Toshiaki Kondo, ”A Simple Shape-Based Approach to Hand Gesture Recognition”, in Proceedings of IEEE International Conference on Electrical Engineering/Electronics Computer Telecommunications and Information Technology (ECTI-CON), Pathumthani, Thailand , pages 851-855, May 2010

[8] A. Jinda-Apiraksa, W. Pongstiensak, and T. Kondo, “Shape-Based Finger Pattern Recognition using Compactness and Radial Distance,” The 3rd International Conference on Embedded Systems and Intelligent Technology(ICESIT 2010), Chiang Mai, Thailand, February 2010.

[9] Rajeshree Rokade , Dharmpal Doye, Manesh Kokare, “Hand Gesture Recognition by Thinning Method”, in Proceedings of IEEE International Conference on Digital Image Processing (ICDIP), Nanded India, pages 284 – 287, March 2009.

[10] M. Panwar, "Hand gesture recognition based on shape parameters," 2012 International Conference on Computing, Communication and Applications, 2012, pp. 1-6, doi: 10.1109/ICCCA.2012.6179213.

[11] B. Nandwana, S. Tazi, S. Trivedi, D. Kumar and S. K. Vipparthi, "A survey paper on hand gesture recognition," 2017 7th International Conference on Communication Systems and Network Technologies (CSNT), 2017, pp. 147-152, doi: 10.1109/CSNT.2017.8418527.

[12] J. S. Sonkusare, N. B. Chopade, R. Sor and S. L. Tade, "A Review on Hand Gesture Recognition System," 2015 International Conference on Computing Communication Control and Automation, 2015, pp. 790-794, doi: 10.1109/ICCUBEA.2015.158.

[13] Liling Ma, Jing Zhang, Junzheng Wang ; “Modified CRF Algorithm for Dynamic Hand Gesture Recognition”; Proceedings of the 33rd Chinese Control Conference; IEEE 2014.

[14] C. Wang and S. C. Chan; “A new hand gesture recognition algorithm based on joint color-depth Super-pixel Earth Mover’s Distance”; International Workshop on Cognitive Information Processing; IEEE 2014.

[15] Weihua Liu, Yangyu Fan, TaoLei, ZhongZhang; ”Human Gesture Recognition Using Orientation Segmentation Feature on Random Forest "; IEEE 2014.

[16] Tingfang Zhang, Zhiquan Feng;“Dynamic Gesture Recognition Based on Fusing Frame Images"; International Conference on Intelligent Systems Design and Engineering Applications; IEEE 2013.

[17] Deepika Tewari, Sanjay Kumar Srivastava , “A Visual Recognition of Static Hand Gestures in Indian Sign Language based on Kohonen Self Organizing Map Algorithm”, International Journal of Engineering and Advanced Technology (IJEAT), Vol.2, Dec 2012, pp. 165-170.

[18] Rajat Shrivastava; “A Hidden Markov Model based Dynamic Hand Gesture Recognition System using OpenCV "; International Advance Computing Conference(IACC); IEEE 2013.

[19] Hand Tracking and Gesture Recognition Using AI: Applications and Challenges https://intellias.com/hand-tracking-and-gesture-recognition-using-ai-applications-and-challenges/

Source Of Figures

Figure 1.1 - List of hand gestures

https://analyticsindiamag.com/how-i-created-a-ml-model-that-identifies-hand-gestures/

Figure 2.1 - Layers of SSD algorithms

https://cdn-images-1.medium.com/max/1000/1*GmJiirxTSuSVrh-r7gtJdA.png

Figure 3.1 - Python script to capture dataset using openCV

Figure 3.2 - Folder structures of stored gestures

Figure 3.3 - Sample dataset (Right indicator, swing, palm)

Figure 3.4 - LabelImg UI used for annotation

https://github.com/tzutalin/labelImg

Figure 3.5 - XML file containing box annotations

Figure 4.1 : Background Subtraction

https://gogul.dev/software/hand-gesture-recognition-p1

Figure 4.2 : Segmenting hand region

Figure 5.1 : Example of CNN and its layers.

https://towardsdatascience.com/covolutional-neural-network-cb0883dd6529

Figure 5.2 - Convolution Neural Network

Figure 5.3 - Steps of training dataset

Figure 5.4 - Our CNN model

Figure 5.5 - Input to CNN model post BE

Figure 5.6 - Training statistics for model

Figure 5.8 - Figure 5.9 - Demo

Figure 5.10 - Result

Figure 6.1 - Hand Detection

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6982909

Figure 6.2 - Input demo

Figure 6.3 - Working demo with volume control

Figure 7.1 - Camera installed in car dashboard

https://www.thedashcamstore.com/advanced-dashcam-installation/