DAYANANDA SAGAR UNIVERSITY

KUDLU GATE, BANGALORE - 560068

Bachelor of Technology
in
COMPUTER SCIENCE AND ENGINEERING

Major Project Report

"HAND GESTURE RECOGNITION USING MACHINE LEARNING"

By

NAME	USN
A S ANILKUMAR	ENG18CS0004
ABHISHEK GOWDA G A	ENG18CS0014
AKHILESHKUMAR	ENG18CS0026
DEVASHISH MANDAL	ENG18CS0086
AKSHAYKUMAR SINNUR	ENG18CS0028

Under the supervision of

Dr. KIRAN B MALAGI

ASSOCIATE PROFESSOR, DEPARTMENT OF CSE

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING,
SCHOOL OF ENGINEERING
DAYANANDA SAGAR UNIVERSITY,
BANGALORE
(2021-2022)

DAYANANDA SAGAR UNIVERSITY

School of Engineering
Department of Computer Science & Engineering
Kudlu Gate, Bangalore – 560068
Karnataka, India

CERTIFICATE

This is to certify that the Phase-II project work titled **"HAND GESTURE RECOGNITION USING MACHINE LEARNING"** is carried out by **A S ANIL KUMAR(ENG18CS0004), ABHISHEK GOWDA G A (ENG18CS0014), AKHILESH KUMAR(ENG18CS0026), AKSHAYKUMAR SINNUR (ENG18CS0028), DEVASHISH MANDAL(ENG18CS0086),** the bonafide students of Bachelor of Technology in Computer Science and Engineering at the School of Engineering, Dayananda Sagar University, Bangalore in partial fulfillment for the award of degree in Bachelor of Technology in Computer Science and Engineering, during the year **2021-2022**.

Signature of The Guide **Dr. Kiran B Malagi** Associate Professor Dept. of CS&E, School of Engineering Dayananda Sagar University Date:	Signature of The Chairman **Dr. Girisha G S** Chairman CSE School of Engineering Dayananda Sagar University Date:	Signature of The Dean **Dr. A Srinivas** Dean School of Engineering Dayananda Sagar University Date:

Name of the Examiner	**Signature of Examiner**
1.	
2.	

DECLARATION

We, **A S ANIL KUMAR(ENG18CS0004), ABHISHEK GOWDA G A(ENG18CS0014), AKHILESH KUMAR(ENG18CS0026), AKSHAYKUMAR SINNUR(ENG18CS0028) DEVASHISH MANDAL(ENG18CS0086),** are student's of the eighth semester B.Tech in **Computer Science and Engineering**, at School of Engineering, **Dayananda Sagar University**, hereby declare that the phase-II project titled **"HAND GESTURE RECOGNITION USING MACHINE LEARNING"** has been carried out by us and submitted in partial fulfillment for the award of degree in **Bachelor of Technology in Computer Science and Engineering** during the academic year **2021-2022**.

Students	**Signature**
Name 1: Mr. A S ANIL KUMAR **USN: ENG18CS0004**	
Name 2: Mr. ABHISHEK GOWDA G A **USN: ENG18CS0014**	
Name 3: Mr. AKHILESH KUMAR **USN: ENG18CS0026**	
Name 4: Mr. AKSHAY KUMAR SINNUR **USN: ENG18CS0028**	
Name 5: Mr. DEVASHISH MANDAL **USN: ENG18CS0086**	

Place: Bangalore

Date :

ACKNOWLEDGEMENT

It is a great pleasure for us to acknowledge the assistance and support of many individuals who have been responsible for the successful completion of this project work.

First, we take this opportunity to express our sincere gratitude to the management of School of Engineering & Technology, Dayananda Sagar University for providing us with a great opportunity to pursue our Bachelor's degree in this institution.

We would like to thank **Dr. A Srinivas. Dean**, **School of Engineering & Technology**, **Dayananda Sagar University** for his constant encouragement and expert advice. It is a matter of immense pleasure to express our sincere thanks to **Dr. Girisha G S, Department Chairman**, **Computer Science, and Engineering**, **Dayananda Sagar University,** for providing the right academic guidance that made our task possible.

We would like to thank our guide **Dr. Kiran B Malagi**, **Associate Professor**, **Department of Computer Science and Engineering**, **Dayananda Sagar University**, for sparing his/her valuable time to extend help in every step of our project work, which paved the way for smooth

progress and the fruitful culmination of the project.

We would like to thank our Project Coordinators **Dr. Meenakshi Malhotra** and **Dr.Bharanidharan N**, and all the staff members of Computer Science and Engineering for their support.

We are also grateful to our family and friends who provided us with every requirement

throughout the course. We would like to thank one and all who directly or indirectly helped us in the Project work.

ABSTRACT

Machine learning is the branch of AI which is focused on building applications which can learn from user input and increase their accuracy over a period of time without being programmed. In this project we train a machine learning model such that, it allows us to capture the gestures made by the human or users and perform the desired task.

Hand gesture recognition for human-computer interaction is an area of active research in computer vision and machine learning. The primary goal of gesture recognition research is to create a system, which can identify specific human gestures and use them to convey information or for device control.

Hand Gesture recognition, although has been exploring for many years, is still a challenging problem. Complex background, camera angles and illumination conditions make the problem more difficult. Thus, this paper presents a fast and robust method for hand gesture recognition based on RGB video. First we detect the skin based on their color. Then we extract the contour and segment the hand region. Finally we recognize the gesture. The results of experiment demonstrate that the proposed method are efficient to recognize gesture with a higher accuracy than the state of the art.

CONTENTS

LIST OF ABBREVIATIONS

These are the some abbreviations we used in our project.

ML	Machine Learning
AI	Artificial Intelligence
LSTM	Long Short Term Memory
DL	Deep Learning
API	Application Programming Interface
Open CV	Open Source Computer Vision
ANN	Artificial Neural Network

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Chapter 1
Introduction

Hand gestures are an aspect of body language that can be conveyed through the center of the palm, the finger position and the shape constructed by the hand. Hand gestures can be classified into static and dynamic. As its name implies, the static gesture refers to the stable shape of the hand, whereas the dynamic gesture comprises a series of hand movements such as waving. There are a variety of hand movements within a gesture; for example, a handshake varies from one person to another and changes according to time and place. The main difference between posture and gesture is that posture focuses more on the shape of the hand whereas gesture focuses on the hand movement. The main approaches to hand gesture research can be classified into the wearable glove-based sensor approach and the camera vision-based sensor approach.

Gesture recognition is a technique which is used to understand and analyze the human body language and interact with the user accordingly. This in turn helps in building a bridge between the machine and the user to communicate with each other. Gesture recognition is useful in processing the information which cannot be conveyed through speech or text. Gestures are the simplest means of communicating something that is meaningful. This paper involves implementation of the system that aims to design a vision-based hand gesture recognition system with a high correct detection rate along with a high-performance criterion, which can work in a real time Human Computer Interaction system without having any of the limitations (gloves, uniform background etc.) on the user environment.

There has been great emphasis on Human-Computer-Interaction research to create easy-to-use interfaces by directly employing natural communication and manipulation skills of humans. As an important part of the body, recognizing hand gesture is very important for Human- Computer-Interaction. In recent years, there has been a tremendous amount of research on hand gesture recognition.

While there are numerous researches focused on this topic, there are still several problems to be solved. The speed and accuracy are two main characteristics of the algorithm, thus a robust and fast method is needed to improve user experiences. A contour based method for recognizing hand gesture using depth image data is shown in [3]. Using depth data, these method can distinguish the hand from background easily without getting confused by background color. However, depth data are not common and easily available while RGB data solves the problem. In a hierarchical method of static hand gesture recognition that combines finger detection and histogram of oriented gradient (HOG) feature is proposed. An algorithm applied for locating fingertips in hand region extracted by Bayesian rule based skin color segmentation is proposed. In the continuous gesture recognition problem is tackled with a two streams Recurrent Neural Networks for the RGB-D data input. These methods all get good effect, but they are not that efficient. This paper provides a more efficient way based on deep learning.

1.1 Purpose:

The gestures are chosen because they are commonly used to communicate and can thus be used in various applications such as, media player, stock management. We have assumed that the trained model should have greater accuracy to avoid misclassification. The model selection is totally depending upon model performance and prediction score. Purpose is to control media player using Hand Gestures.

1.2 Scope: The scope of this project is to create a method to recognize hand gestures based on a machine learning technique, employing histograms of local orientation, the orientation histogram will be used as a feature vector for gesture classification. And interpolation. High priority for the system is to be simple without making use of any special hardware. All the competition should occur on a workstation or PC. Special hardware would be used only to digitize the image (Scanner or digital camera).

1.3 Problem Statement: “Controlling media player using hand gestures, with the help of Machine learning and open cv”

1.4 Objectives:

To solve the problem identified we are coming up with following objectives:

- Recognise the hand gesture of the user
- Communicating with the keyboard controls with hand gestures

1.5 Controls and Respective Gestures:

Medial player controls are Play, Pause, Volume up, Volume down Forward, backward. Next, previous and mute. In this project we are using certain hand gestures to control the media player. Like, 2 finger gesture Represents play or pause functions three finger hand gesture represents volume up control, whereas four finger and gesture represents volume down in control, five finger represents forward fist represents the backward thumbs up will be the next video in the list and thumbs down will be previous video in the list, and one finger represents the mute.

Table 1: Controls and respective gestures

CONTROLS	GESTURES
Play / pause	2 finger
Volume up	3 fingers
Volume down	4 fingers
Forward	5 fingers
Backward	Fist
Next	Thumbs up
Previous	Thumbs down
Mute	1 finger

These above gestures have implemented, by showing these gestures to camera while the model is running we can handle the media player.

1.6 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.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 18 million. The library is used extensively in companies, research groups and by governmental bodies.

Along with well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that employ the library, there are many startups such as Applied Minds, VideoSurf, and Zeitera, that make extensive use of OpenCV. OpenCV's deployed uses span the range from stitching streetview images together, detecting intrusions in surveillance video in Israel, monitoring mine equipment in China, helping robots navigate and pick up objects at Willow Garage, detection of swimming pool drowning accidents in Europe, running interactive art in Spain and New York, checking runways for debris in Turkey, inspecting labels on products in factories around the world on to rapid face detection in Japan.

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full - featured and interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers.

OpenCV provides a module called ml that has many machine learning algorithms bundled into it. Some of the algorithms include Bayes Classifier, K-Nearest Neighbors, Support Vector Machines, Decision Trees, Neural Networks.

1.7 Media player:

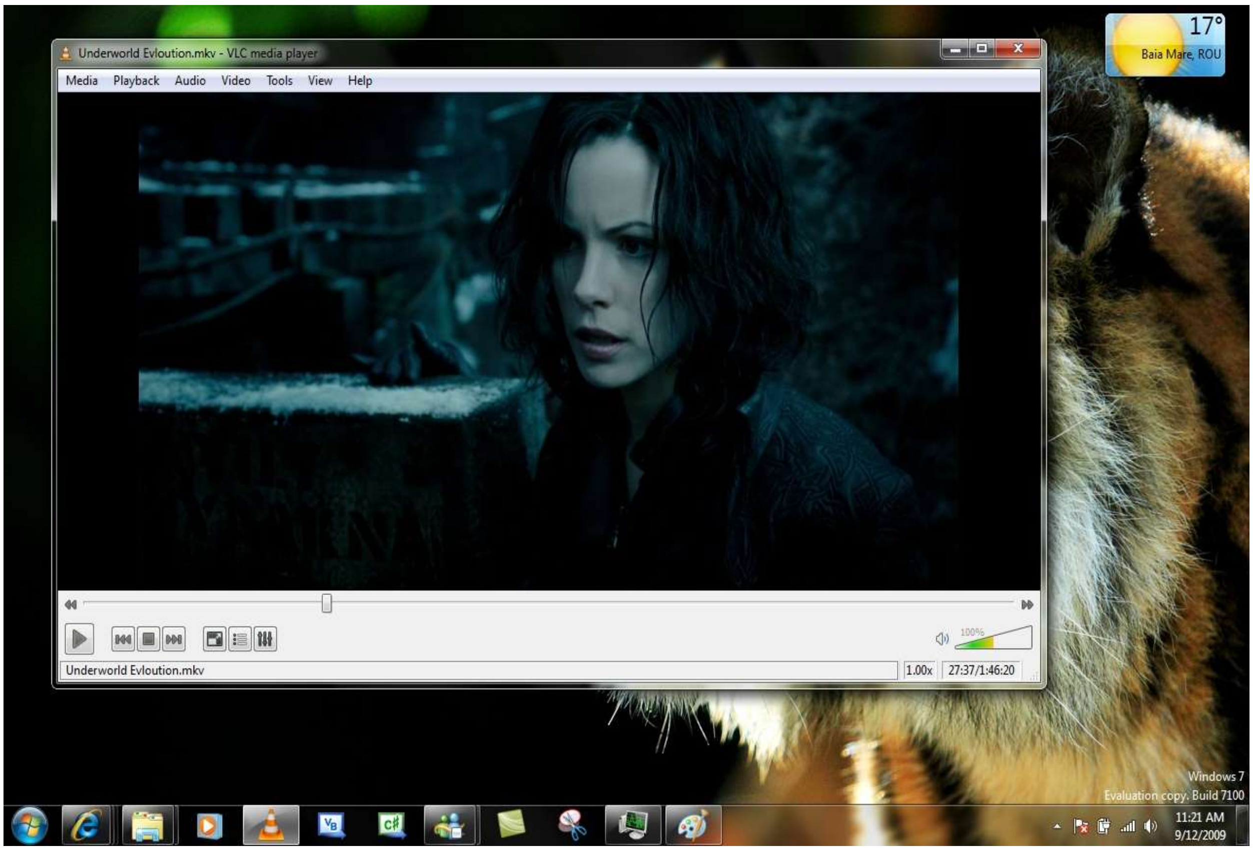
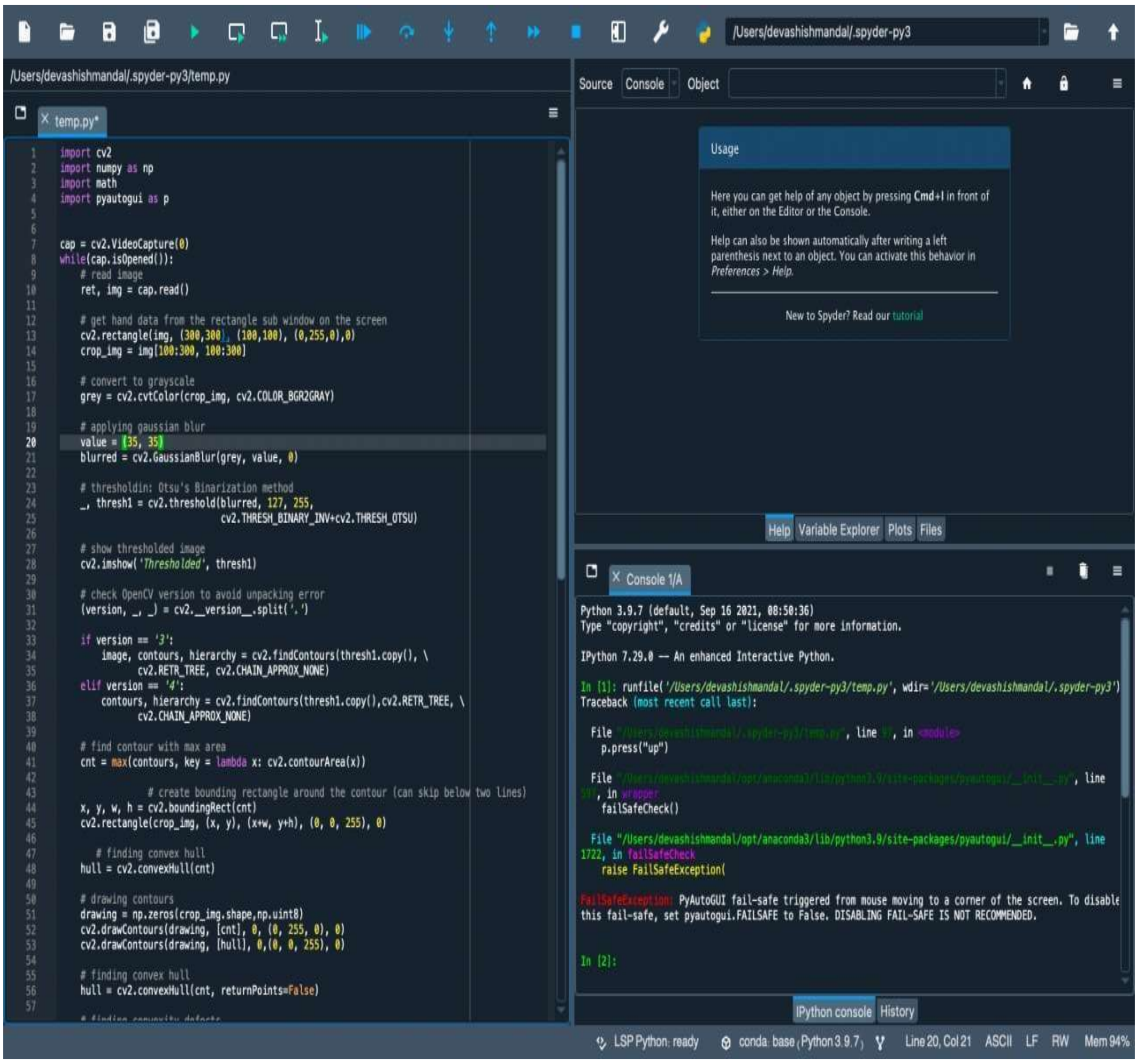

Underworld Evloution.mkv - VLC media player
Media Playback Audio Video Tools View Help
Underworld Evloution.mkv
1.00x
27:37/1:46:20
17°
Baia Mare, ROU
Windows 7
Evaluation copy. Build 7100
11:21 AM
9/12/2009

Fig 1.1 – Media Player

In this project we are using VLC media player which is very much feasible to control with the keyboard. While running this model we recommend to use this media player or any other media player which have same specifications. VLC is a free and open source cross-platform multimedia player and framework that plays most multimedia files, and various streaming purpose .

VLC media player (previously the Video LAN Client and commonly known as simply VLC) is a free and open-source, portable, cross-platform media player software and streaming media server developed by the Video LAN project. VLC is available for desktop operating systems and mobile platforms, such as Android, iOS and iPadOS.

1.8 Interpreter



Here you can get help of any object by pressing Cmd+I in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

```python
import cv2
import numpy as np
import math
import pyautogui as p


cap = cv2.VideoCapture(0)
while(cap.isOpened()):
    # read image
    ret, img = cap.read()

    # get hand data from the rectangle sub window on the screen
    cv2.rectangle(img, (300,300), (100,100), (0,255,0),0)
    crop_img = img[100:300, 100:300]

    # convert to grayscale
    grey = cv2.cvtColor(crop_img, cv2.COLOR_BGR2GRAY)

    # applying gaussian blur
    value = (35, 35)
    blurred = cv2.GaussianBlur(grey, value, 0)

    # thresholdin: Otsu's Binarization method
    _, thresh1 = cv2.threshold(blurred, 127, 255,
                               cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)

    # show thresholded image
    cv2.imshow('Thresholded', thresh1)

    # check OpenCV version to avoid unpacking error
    (version, _, _) = cv2.__version__.split('.')

    if version == '3':
        image, contours, hierarchy = cv2.findContours(thresh1.copy(), \
               cv2.RETR_TREE, cv2.CHAIN_APPROX_NONE)
    elif version == '4':
        contours, hierarchy = cv2.findContours(thresh1.copy(),cv2.RETR_TREE, \
               cv2.CHAIN_APPROX_NONE)

    # find contour with max area
    cnt = max(contours, key = lambda x: cv2.contourArea(x))

                 # create bounding rectangle around the contour (can skip below two lines)
    x, y, w, h = cv2.boundingRect(cnt)
    cv2.rectangle(crop_img, (x, y), (x+w, y+h), (0, 0, 255), 0)

       # finding convex hull
    hull = cv2.convexHull(cnt)

    # drawing contours
    drawing = np.zeros(crop_img.shape,np.uint8)
    cv2.drawContours(drawing, [cnt], 0, (0, 255, 0), 0)
    cv2.drawContours(drawing, [hull], 0,(0, 0, 255), 0)

    # finding convex hull
    hull = cv2.convexHull(cnt, returnPoints=False)

    # finding convexity defects
```

```
Python 3.9.7 (default, Sep 16 2021, 08:50:36)
Type "copyright", "credits" or "license" for more information.

IPython 7.29.0 -- An enhanced Interactive Python.

In [1]: runfile('/Users/devashishmandal/.spyder-py3/temp.py', wdir='/Users/devashishmandal/.spyder-py3')
Traceback (most recent call last):

  File "/Users/devashishmandal/.spyder-py3/temp.py", line [illegible], in <module>
    p.press("up")

  File "/Users/devashishmandal/opt/anaconda3/lib/python3.9/site-packages/pyautogui/__init__.py", line
[illegible], in wrapper
    failSafeCheck()

  File "/Users/devashishmandal/opt/anaconda3/lib/python3.9/site-packages/pyautogui/__init__.py", line
1722, in failSafeCheck
    raise FailSafeException(

FailSafeException: PyAutoGUI fail-safe triggered from mouse moving to a corner of the screen. To disable
this fail-safe, set pyautogui.FAILSAFE to False. DISABLING FAIL-SAFE IS NOT RECOMMENDED.


In [2]:
```

Fig 1.2– Interpreter

In this project we are using **SPYDER** interpreter, Spyder is a free and open source scientific environment written in Python, for Python, and designed by and for scientists, engineers and data analysts. It features a unique combination of the advanced editing, analysis, debugging, and profiling functionality of a comprehensive development tool with the data exploration, interactive execution, deep inspection, and beautiful visualization capabilities of a scientific package.

CHAPTER 2

LITERATURE REVIEW

Chapter 2
Literature Review

To understand the state of the art work in the area of hand gesture recognition using machine learning, we conducted a literature survey. The summary has been given in the table.

Table: 2.1 : Summary of Literature survey

Author's Name/ Paper Title	**Year**	**Technology / Design**	**Result shared by author**	**What you infer?**
Suharjito, H. Gunawan, N. Thiracitta and A. Nugroho.	'2018'	Video-based Sign Language Recognition	Recognition of a sign not only by the shape but also by the action the signer does	Model that classifies video clips based on sequence of frames.
Vijay Shinde, Tushar Bacchav, Jitendra Pawar and Mangesh Sanap.	'2018'	Human Computer Interaction (HCI)	Using library 'pyautogui' which works as an API, between ML model and the pc keyboard.	API that connects the keyboard and ML model.
Machine Learning Techniques for Indian Sign Language Recognition,	'2017'	MATLAB	Classification of single and double handed Indian sign language recognition using machine learning algorithms with the help of MATLAB with 92-100% of accuracy.	Recognizing different sign languages using machine learning.
Rohini M, Abhishek Leo Kingston. j, Shriram G S, Siva Sankaran & Vasuki G,	'2018'	Open CV	Open CV is the place where one can get pre processed data, Training machine learning model is not necessary.	Using Open-cv makes us to prepare machine learning model easily since, we get pre processed data.

With the help of literature survey, we came across certain libraries like PyAutoGUI which acts as an API that controls the keyboard and the mouse. Along with that we came to the conclusion that designing machine learning model using Open CV, training the model is not necessary.

CHAPTER 3

REQUIREMENTS

Chapter 3
Requirements

In this chapter we are discussing about requirements like functional, non-functional, hardware and software requirements

3.1 Functional Requirements:

Gesture analysis using machine learning has various applications, this project deals with the controlling of media player using hand gestures.

- The camera used will be able to capture user images from the real time video sequences.
- The media player used will be able to handle it's controls with the keyboard (eg: VLC Media player).
- The interpreter used will be able to have access for controlling camera and keyboard (eg: spider)

3.2 Non Functional Requirements:

- The distance with which we show the hand gestures should be less than 2 meters.
- Background should be plain whenever hand gesture is shown.

3.3 Software and Hardware requirements:

Hardware Requirements:

Processor : Any Processor above 500 MHz

Ram : 4GB Hard Disk : 512GB

Input Device : Standard Keyboard AND Mouse.

Output Device : High display monitor.

Software Requirements:

Microsoft Windows XP or later / Ubuntu 12.0 LTS or later /MAC OS 10.1 or later.

Python Interpreter (3.6).Python -IDLE(Python 3.4 64 bit) or any python

softwareTensorFlow framework, Keras API , AND Open CV.

CHAPTER 4

METHODOLOGY

Chapter 4
Methodology

Whenever we provide hand gesture to the camera while model is running, It captures the image and convert it to the grey scale image for the image preprocessing. Wherein image pre processing feature extraction will happen. That extracted data is used to recognize the gesture given by the user respective task will be performed.

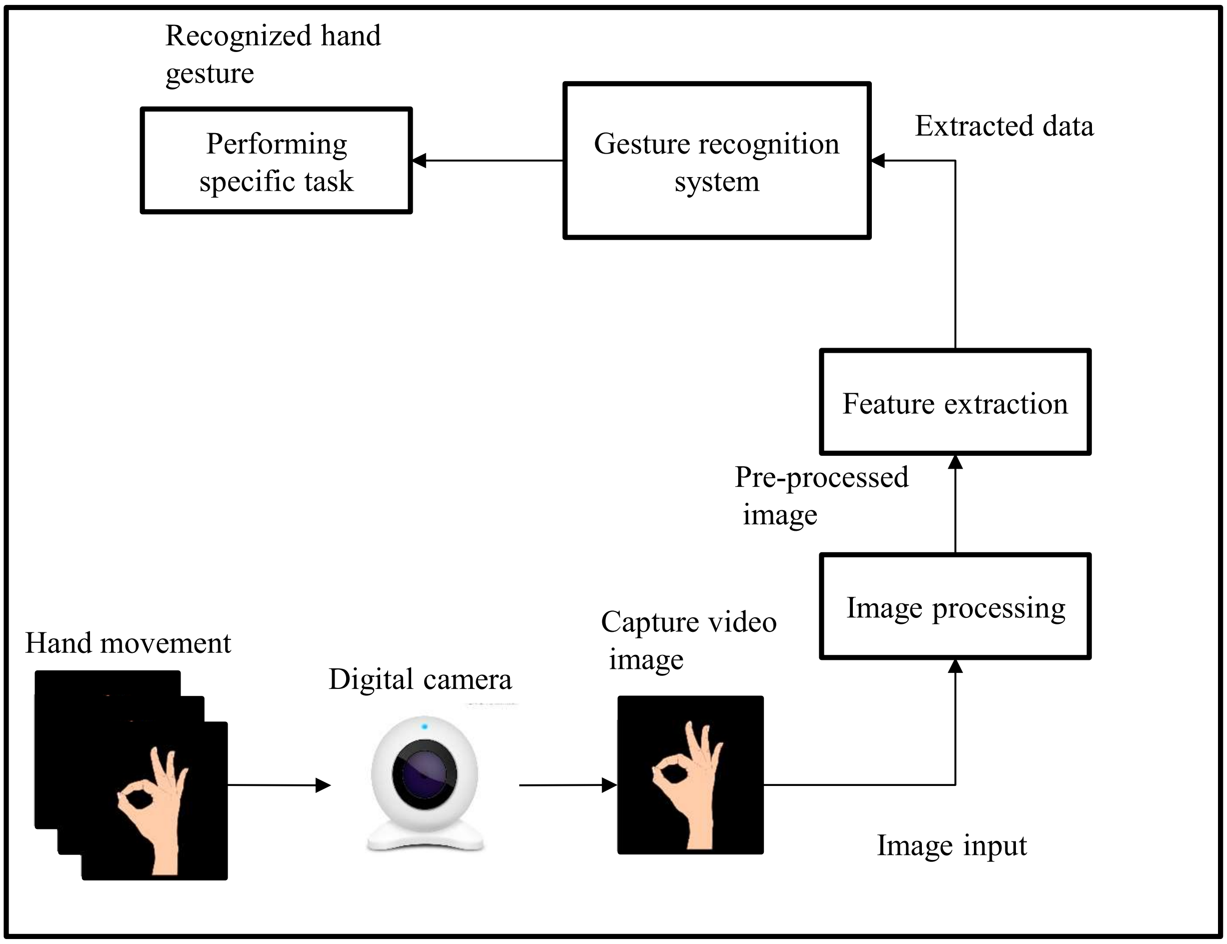


Figure 4.1 - Design

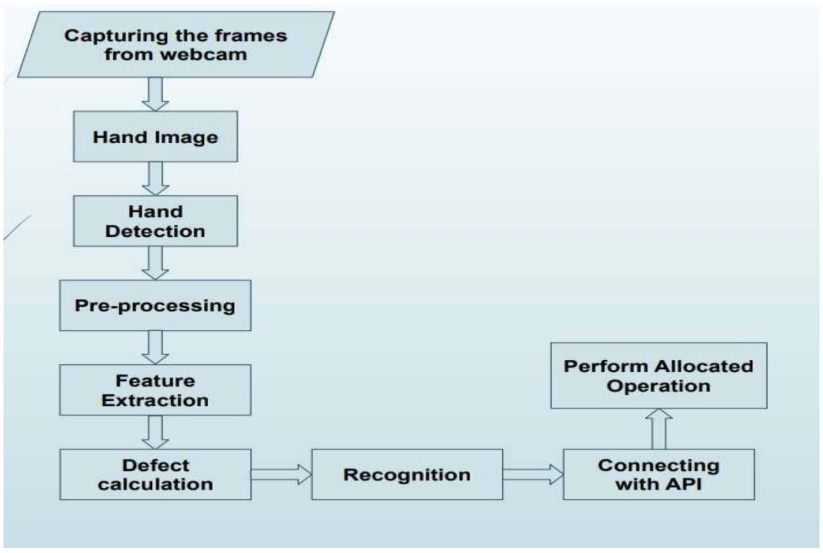


Figure 4.2 - System Data Flow Diagram

Once the hand is subjected to the camera, then it captures the frames of real time images. Then it converts to grayscale image. After the detection phase pre-processing happens, the grayscale image is subjected into thresholding, wherein only the specified image pixels from the range 127 - 255 remain. And then the other pixels are converted to black.

From The convexity hull we would be trying to find out contours that is exactly the contours formed between finger angles.

If the angle between the fingers is <= 90 degrees we will be incrementing the value of contours in the count variable, according to the number of count value captured we would be bringing out actions onto the media player.

Action to the media player from gestures is done by the python library PyAutoGUI, where it acts as an API between the model and the keyboard.

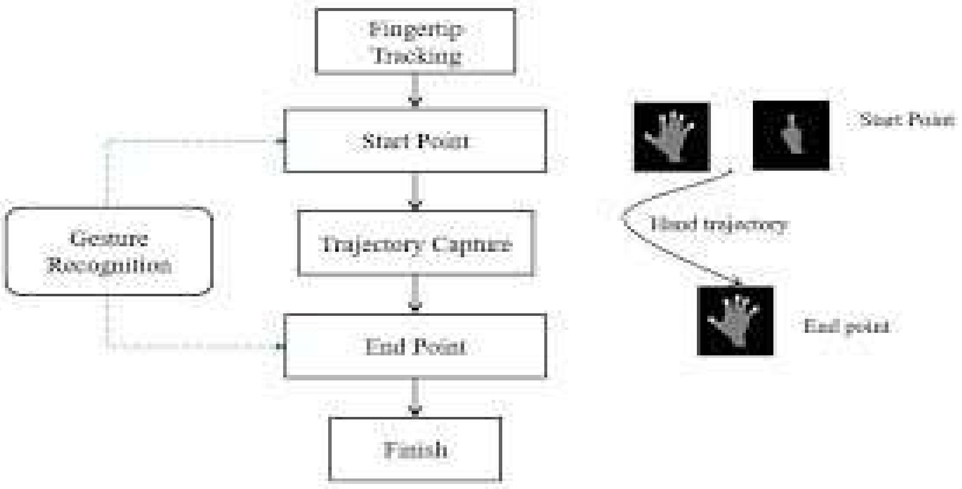


Figure 4.3 - Proposed scheme for gesture spotting.

A hand gesture recognition system was developed to capture the hand gestures being performed by the user and to control a computer system based on the incoming information. Many of the existing systems in literature have implemented gesture recognition using only spatial modelling, i.e. recognition of a single gesture and not temporal modelling i.e. recognition of motion of gestures. Also, the existing systems have not been implemented in real time, they use a pre captured image as an input for gesture recognition.

CHAPTER 5

IMPLEMENTATION

Chapter 5
Implementation

5.1 Built in modules:

import cv2: Importing open cv library.

import numpy: Importing numpy library.

import math: Importing mathematics library.

import pyautogui: Importing PyAutoGUI which acts as an API.

5.2 User defined modules:

> Reads image

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

> Get hand data from the rectangle sub window on the screen

```
cv2.rectangle(img, (300,300), (100,100), (0,255,0),0)
crop_img = img[100:300, 100:300]
```

> Convert to grayscale

```
grey = cv2.cvtColor(crop_img, cv2.COLOR_BGR2GRAY)
```

> Applying gaussian blur

```
value = (35, 35)
blurred = cv2.GaussianBlur(grey, value, 0)
```

> Thresholding: Otsu's Binarization method

```
_, thresh1 = cv2.threshold(blurred, 127, 255,

cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
```

> Show thresholded image

```
cv2.imshow('Thresholded', thresh1)
```

> Check OpenCV version to avoid unpacking error

```
(version, _, _) = cv2.__version__.split('.')
```

> Find contour with max area

```
cnt = max(contours, key = lambda x: cv2.contourArea(x))
```

> Create bounding rectangle around the contour (can skip below two lines)

```
x, y, w, h = cv2.boundingRect(cnt)
cv2.rectangle(crop_img, (x, y), (x+w, y+h), (0, 0, 255), 0)
```

> Finding convex hull

```
hull = cv2.convexHull(cnt)
```

> Drawing contours

```
drawing = np.zeros(crop_img.shape,np.uint8)
cv2.drawContours(drawing, [cnt], 0, (0, 255, 0), 0)
cv2.drawContours(drawing, [hull], 0,(0, 0, 255), 0)
```

> Finding convexity defects

```
defects = cv2.convexityDefects(cnt, hull)
count_defects = 0
cv2.drawContours(thresh1, contours, -1, (0, 255, 0), 3)
```

> Applying Cosine Rule to find angle for all defects (between fingers) with angle > 90 degrees and ignore defects

```
for i in range(defects.shape[0]):
    s,e,f,d = defects[i,0]

    start = tuple(cnt[s][0])
    end = tuple(cnt[e][0])
    far = tuple(cnt[f][0])
```

> Find length of all sides of triangle

```
    a = math.sqrt((end[0] - start[0])**2 + (end[1] - start[1])**2)
    b = math.sqrt((far[0] - start[0])**2 + (far[1] - start[1])**2)
    c = math.sqrt((end[0] - far[0])**2 + (end[1] - far[1])**2)
```

> Apply cosine rule here

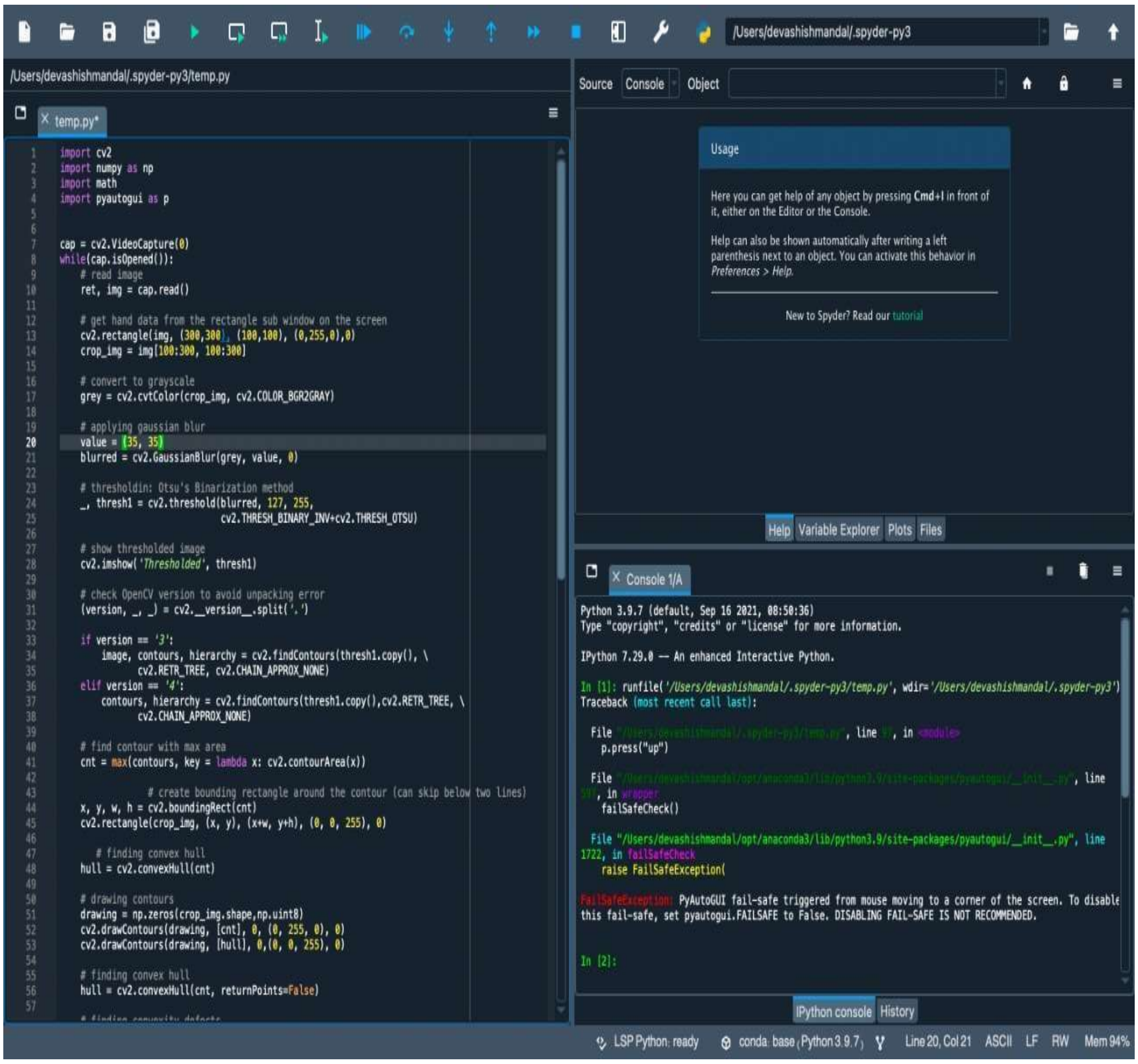
```
    angle = math.acos((b**2 + c**2 - a**2)/(2*b*c)) * 57
```

> Ignore angles > 90 and highlight rest with red dots

```
    if angle <= 90:
        count_defects += 1
        cv2.circle(crop_img, far, 1, [0,0,255], -1)
    #dist = cv2.pointPolygonTest(cnt,far,True)
```

> Define actions required

```
if count_defects == 1:
    p.press("space")
    cv2.putText(img,"2 finger,Space", (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 2, 2)
elif count_defects == 2:
    str = "3 fingers,VOLUP"
    p.press("up")
    cv2.putText(img, str, (5, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, 2)
elif count_defects == 3:
    cv2.putText(img,"4 fingers,VOLDWN", (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 2, 2
elif count_defects == 4:
    cv2.putText(img,"5 fingers,FWD", (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 2, 2)
else:
    cv2.putText(img,"entire hand", (50, 50),\
                cv2.FONT_HERSHEY_SIMPLEX, 2, 2).
```



/Users/devashishmandal/.spyder-py3/temp.py

temp.py*

```
import cv2
import numpy as np
import math
import pyautogui as p


cap = cv2.VideoCapture(0)
while(cap.isOpened()):
    # read image
    ret, img = cap.read()

    # get hand data from the rectangle sub window on the screen
    cv2.rectangle(img, (300,300), (100,100), (0,255,0),0)
    crop_img = img[100:300, 100:300]

    # convert to grayscale
    grey = cv2.cvtColor(crop_img, cv2.COLOR_BGR2GRAY)

    # applying gaussian blur
    value = (35, 35)
    blurred = cv2.GaussianBlur(grey, value, 0)

    # thresholdin: Otsu's Binarization method
    _, thresh1 = cv2.threshold(blurred, 127, 255,
                               cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)

    # show thresholded image
    cv2.imshow('Thresholded', thresh1)

    # check OpenCV version to avoid unpacking error
    (version, _, _) = cv2.__version__.split('.')

    if version == '3':
        image, contours, hierarchy = cv2.findContours(thresh1.copy(), \
               cv2.RETR_TREE, cv2.CHAIN_APPROX_NONE)
    elif version == '4':
        contours, hierarchy = cv2.findContours(thresh1.copy(),cv2.RETR_TREE, \
               cv2.CHAIN_APPROX_NONE)

    # find contour with max area
    cnt = max(contours, key = lambda x: cv2.contourArea(x))

             # create bounding rectangle around the contour (can skip below two lines)
    x, y, w, h = cv2.boundingRect(cnt)
    cv2.rectangle(crop_img, (x, y), (x+w, y+h), (0, 0, 255), 0)

       # finding convex hull
    hull = cv2.convexHull(cnt)

    # drawing contours
    drawing = np.zeros(crop_img.shape,np.uint8)
    cv2.drawContours(drawing, [cnt], 0, (0, 255, 0), 0)
    cv2.drawContours(drawing, [hull], 0,(0, 0, 255), 0)

    # finding convex hull
    hull = cv2.convexHull(cnt, returnPoints=False)

    # finding convexity defects
```

Usage

Here you can get help of any object by pressing Cmd+I in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in Preferences > Help.

New to Spyder? Read our tutorial

Console 1/A

```
Python 3.9.7 (default, Sep 16 2021, 08:50:36)
Type "copyright", "credits" or "license" for more information.

IPython 7.29.0 -- An enhanced Interactive Python.

In [1]: runfile('/Users/devashishmandal/.spyder-py3/temp.py', wdir='/Users/devashishmandal/.spyder-py3')
Traceback (most recent call last):

  File "/Users/devashishmandal/.spyder-py3/temp.py", line [illegible], in <module>
    p.press("up")

  File "/Users/devashishmandal/opt/anaconda3/lib/python3.9/site-packages/pyautogui/__init__.py", line
[illegible], in wrapper
    failSafeCheck()

  File "/Users/devashishmandal/opt/anaconda3/lib/python3.9/site-packages/pyautogui/__init__.py", line
1722, in failSafeCheck
    raise FailSafeException(

FailSafeException: PyAutoGUI fail-safe triggered from mouse moving to a corner of the screen. To disable
this fail-safe, set pyautogui.FAILSAFE to False. DISABLING FAIL-SAFE IS NOT RECOMMENDED.


In [2]:
```

Fig 5.1– Code

CHAPTER 6
TESTING AND RESULTS

Chapter 6
Testing and Results

Testing and Results table

The following diagram represents the testing and result cases, We used certain controls like play, pause, volume up, volume down, forward, backward, and mute. For these controls we have used hand gestures like 2 fingers, 3 fingers, 4 fingers, 5 fingers, fist, 1 finger respectively.

Table: 6.1: Sample test case

Controls	**Gestures**	**Keyboard key**	**Expected output**	**Output we got**	**Pass/fail**
Play / pause	2 Fingers	Space	Pause / Play	Pause / play	Pass
Volume up	3 fingers	Up arrow	Vol-up	Vol-up	Pass
Volume Down	4 Fingers	Down arrow	Vol-down	Vol-down	Pass
Forward	5 Fingers	Right arrow	Fwd	Fwd	Pass
Backward	Fist	Left arrow	Bwd	Bwd	Fail
Mute	1 Finger	M key	Mute	Mute	Fail

These are the some screenshots taken while testing and running our project in SPYDER interpreter.

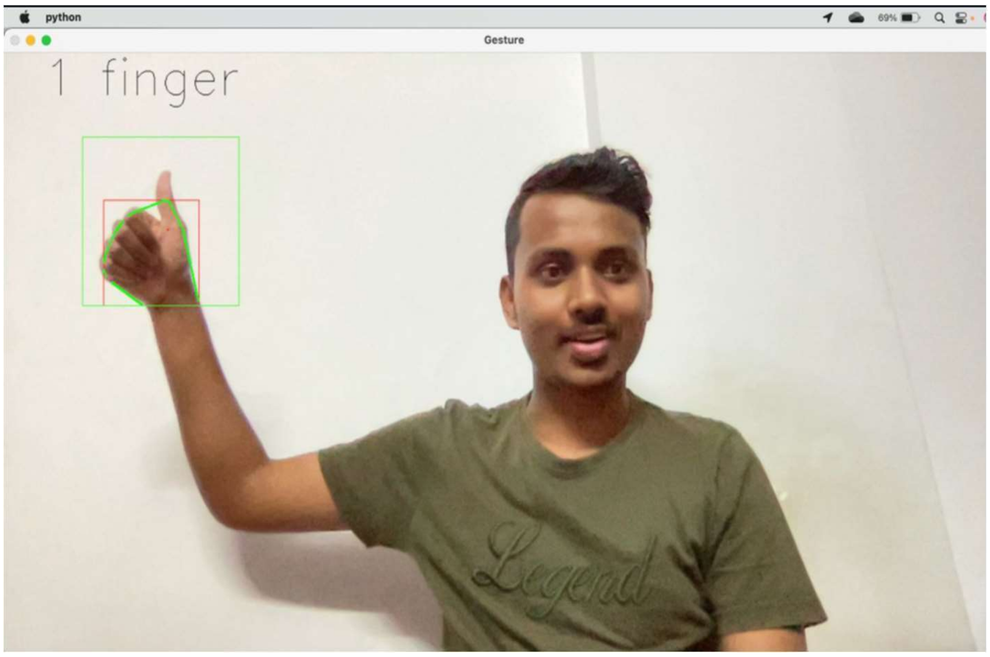


Figure 6.1 – 1 finger (MUTE)

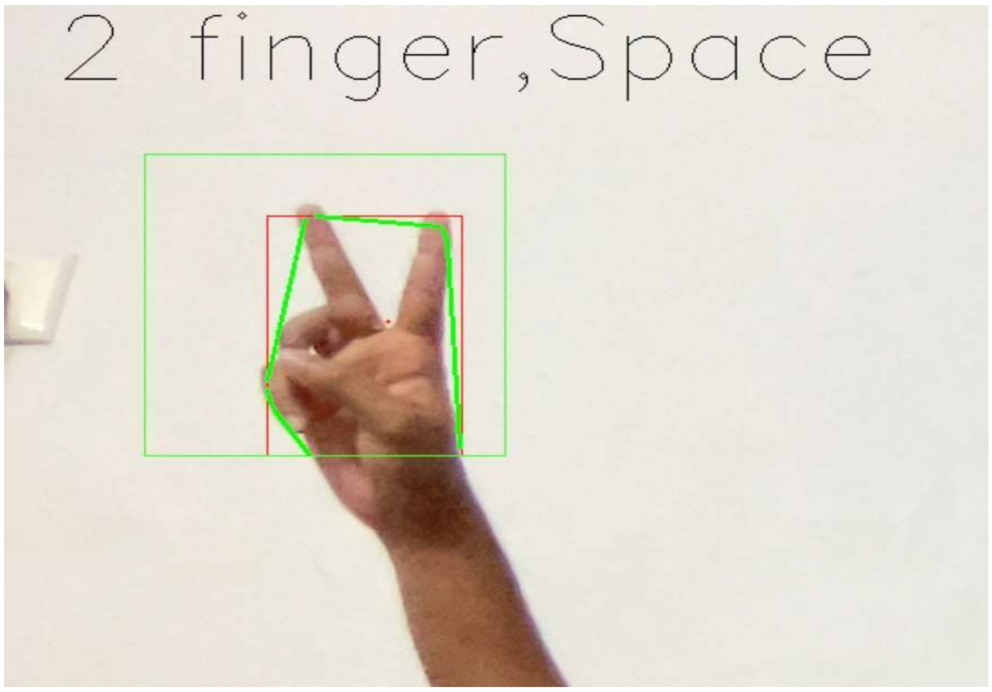


Figure 6.2 - 2 finger (PLAY / PAUSE)

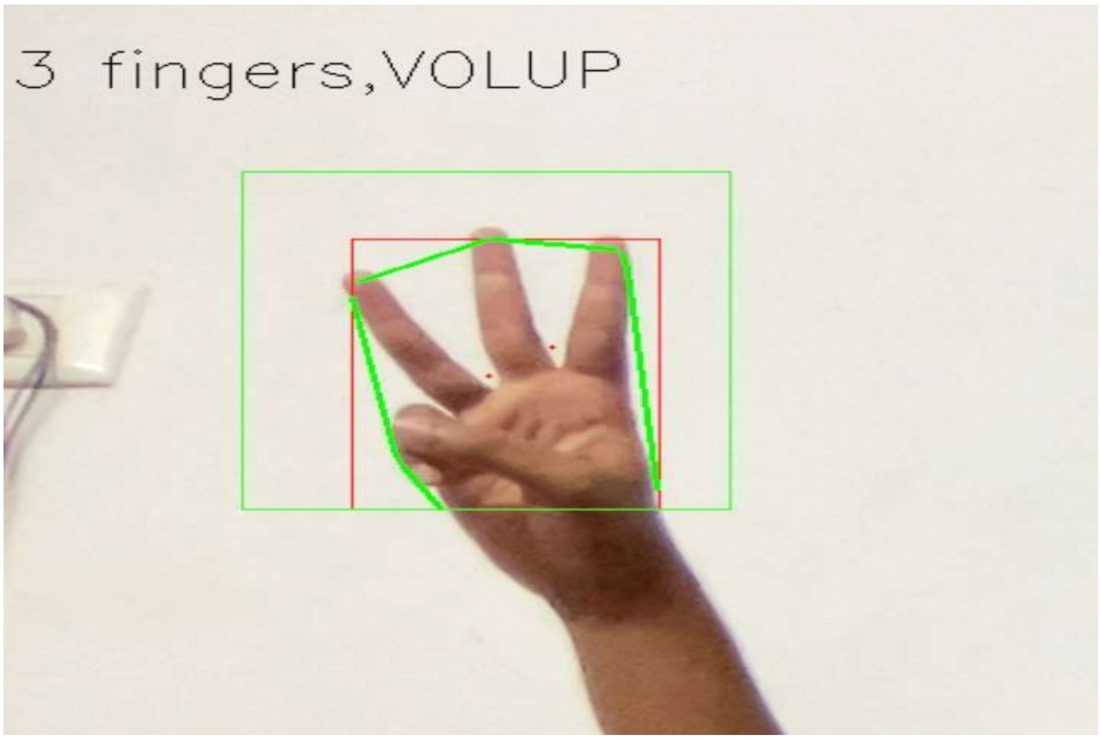


Figure 6.3 – 3 finger (VOLUME UP)

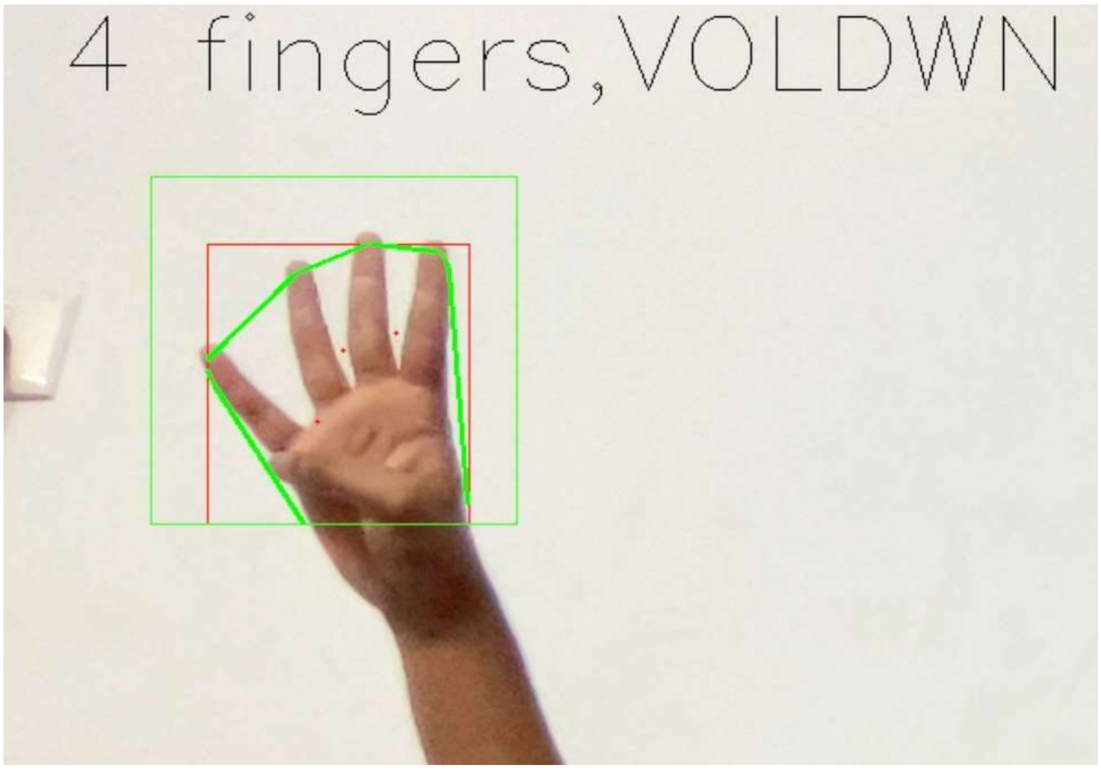


Figure 6.4 – 4 finger (VOLUME DOWN)

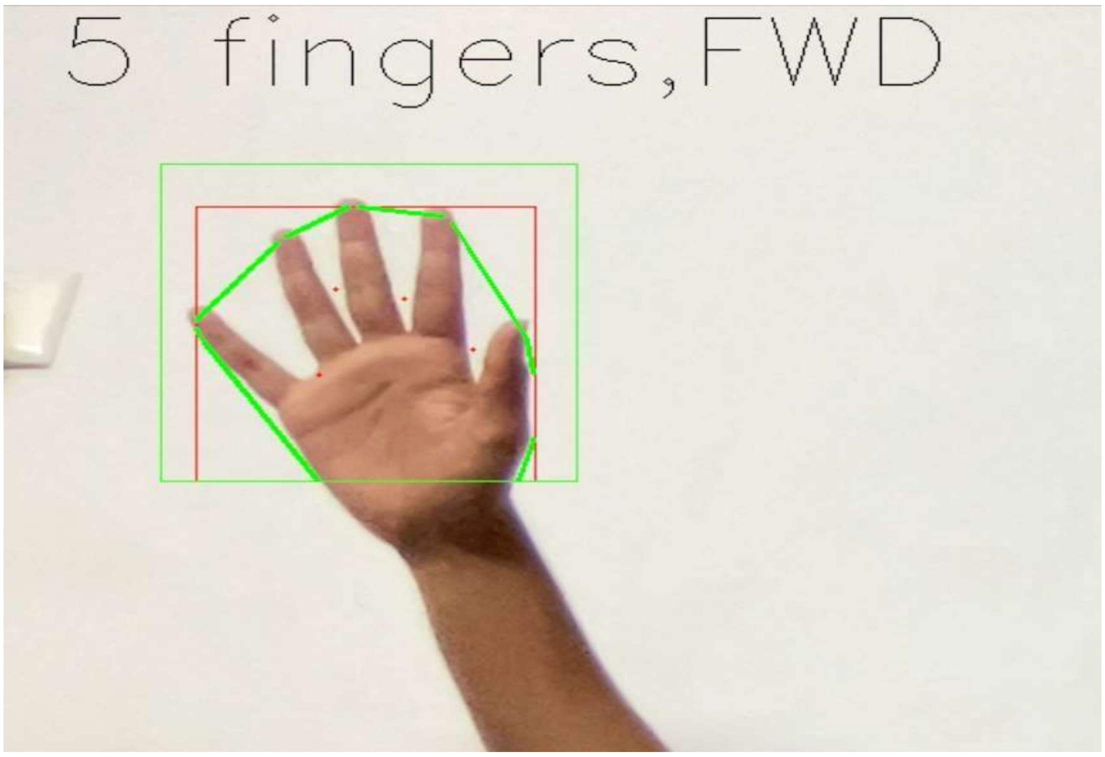


Figure 6.5 – 5 finger (FORWARD)

CHAPTER 7

CONCLUSION AND FUTURE WORK

Chapter 7
Conclusion and Future Work

This project is designed to analyze the hand gestures made by the user and produce the output and here we are using different gestures in order to control the media player. Here the machine learning model will be analyzing the gesture made by the user and to produce the required output, as gesture analysis and machine learning has lot more applications and future scopes which can be applied in wide range of applications.

- This project can be scaled by adding few more gestures and used conveniently.
- We can use this project for keyboard and mouse automation.
- The same concept can also be used for stock management.
- It can also be used to convert voice messages to texts.
- It can also be used to convey the useful information's to the robots.

References:

1. Suharjito, H. Gunawan, N. Thiracitta and A. Nugroho, "Sign Language Recognition Using Modified Convolutional Neural Network Model," 2018 Indonesian Association for Pattern Recognition International Conference (INAPR), Jakarta, Indonesia, 2018, pp. 1-5, DoI: 10.1109/INAPR.2018.8627014.

2. G. A. Rao, K. Syamala, P. V. V. Kishore and A. S. C. S. Sastry, "Deep convolutional neural networks for sign language recognition," 2018 Conference on Signal Processing And Communication Engineering Systems (SPACES), Vijayawada, 2018, pp. 194-197, DoI: 10.1109/SPACES.2018.8316344.

3. Sign Language Translator Application Using OpenCV Published under licence by IOP Publishing Ltd IOP Conference Series: Materials Science and Engineering, Volume 333, International Conference on Advanced Materials for Better Future 2017 4–5 September 2017, Surakarta, Indonesia

4. K. K. Dutta and S. A. S. Bellary, "Machine Learning Techniques for Indian Sign Language Recognition," 2017 International Conference on Current Trends in Computer, Electrical, Electronics and Communication (CTCEEC), Mysore, 2017, pp. 333-336, DoI: 10.1109/CTCEEC.2017.8454988

5. Vijay Shinde, Tushar Bacchav, Jitendra Pawar and Mangesh Sanap. 2014. International Journal of Engineering Research & Technology (IJERT). ISSN NO: 2278-0181 Vol. 3 Issue 1, January-2014.

6. Rohini M, Abhishek Leo Kingston. j, Shriram G S, Siva Sankaran & Vasuki G, "Hand Gesture Recognition Using OpenCV", International Journal of Scientific Research in Science Technology (IJSRST). ISSN NO:2395-602X, DoI:10.32628, March-April 2021.

7. Mr. Deepak K. Ray, Mayank Soni –Hand Gesture Recognition using Python / International Journal on Computer Science and Engineering (IJCSE) . Assistant Professor: MCA Dept , ICEM, Parandwadi Pune, India.

APPENDIX A

OPEN CV

OpenCV is the huge open-source library for computer vision. By using it, one can process images and videos to identify objects, faces, or even the handwriting of a human. When integrated with various libraries, such as NumPy, python is capable of processing the OpenCV array structure for analysis. To Identify image pattern and its various features we use vector space and perform mathematical operations on these features.

APPENDIX B

PyAutoGUI

PyAutoGUI is a Python module which can automate your GUI and programmatically control your keyboard and mouse. This article illustrates the GUI functions to create display boxes. PyAutoGUI has ability to control Mouse and Keyboard Automation. PyAutoGUI does not come with python, so go to command prompt and type the following:

pip3 install PyAutoGUI.

GITHUB LINK:

https://github.com/anilangdi01/HAND-GESTURE-RECOGNITION.git