# GESTURE BASED INPUT SYSTEM

*A project report submitted in partial fulfillment of the Academic requirements for the award of the Degree of*

**BACHELOR OF TECHNOLOGY**

**IN**

**ELECTRONICS AND COMMUNICATION ENGINEERING**

**By**

**Batch Code: 18MP01A02**

| | |
|---|---|
| **B.V.S.Satya Varun** | **18E11A0403** |
| **G.Bhanu Shankar** | **18E11A0412** |
| **Vamshi Jella** | **18E11A0416** |
| **K.Sai Krishna** | **18E11A0419** |

***Under the guidance of***

**Dr. Surendar Reddy**

**(Assistant Professor)**

**Department of Electronics and Communication Engineering**

**BHARAT INSTITUTE OF ENGINEERING AND TECHNOLOGY**

(Affiliated to JNTUH Hyderabad, Approved by AICTE & Accredited by NAAC)
Ibrahimpatnam - 501 510, Hyderabad

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

**BHARAT INSTITUTE OF ENGINEERING AND TECHNOLOGY**

(Affiliated to JNTUH Hyderabad, Approved by AICTE and Accredited by NBA)
Ibrahimpatnam - 501 510, Hyderabad

# Certificate

This is to certify that the project work entitled “Gesture Based Input System” is the bonafide work done

**By**

| | |
|---|---|
| **B.V.S.Satya Varun** | **18E11A0403** |
| **G.Bhanu Shankar** | **18E11A0412** |
| **Vamshi Jella** | **18E11A0416** |
| **K.Sai Krishna** | **18E11A0419** |

in the department of Electronics and Communication Engineering, **BHARAT INSTITUTE OF ENGINEERING AND TECHNOLOGY**, Ibrahimpatnam is submitted to **Jawaharlal Nehru Technological University,**Hyderabad in partial fulfillment of the requirements for the award of **B.TECH** degree in **ELECTRONICS AND COMMUNICATION ENGINEERING** during 2021-2022

**Guide:**

**Name : Dr. Surendar Reddy**

**(Assistant Professor)**

Dept of ECE
Bharat Institute of Engineering and Technology,
Ibrahimpatnam – 501 510, Hyderabad.

**Head of the Department:**

**Mr.D.Sankara Reddy**

Dept of ECE
Bharat Institute of Engineering and Technology,
Ibrahimpatnam – 501 510, Hyderabad.

Viva-Voce held on…………………………………………

**Internal Examiner**

**External Examiner**

**DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING**

**BHARAT INSTITUTE OF ENGINEERING AND TECHNOLOGY**

**(Affiliated to JNTUH Hyderabad, Approved by AICTE & Accredited by NAAC and NBA)**
**(UG Programmes: EEE, ECE, ME & CSE)**
**Ibrahimpatnam - 501 510, Hyderabad, Telangana.**

# DECLARATION

We hereby declare that this Major Project Report is titled **“GESTURE BASED INPUT SYSTEM”** is a genuine work carried out by us in **B.Tech (Electronics and Communication Engineering)** degree course of **Jawaharlal Nehru Technology University Hyderabad,** and has not been submitted to any other course or university for the award of the degree byus.

| **Name** | **Roll No.** | **Signature of the Student** |
|---|---|---|

# BHARAT INSTITUTE OF ENGINEERING AND TECHNOLOGY

**Mangalpally (V), Ibrahimpatnam (M), Ranga Reddy (Dist),**

**HYDERABAD-501 510**

**(Affiliated to JNTUH Hyderabad, Approved by AICTE & Accredited by NAAC)**

***Vision of the Institution:*** To achieve autonomous & university status and spread universal education by inculcating discipline, character and knowledge into the young minds and mould them into enlightened citizens.

***Mission of the Institution:*** Our mission is to impart education, in a conducive ambience, as comprehensive as possible, with the support of all the modern technologies and make the students acquire the ability and passion to work wisely, creatively and effectively for the betterment of our society.

***Vision of ECE department:*** The vision of the Department of Electronics and Communication is to effectively serve the educational needs of local and rural students within the core areas of Electronics and Communication Engineering and develop high quality engineers and responsible citizens.

***Mission of ECE department:*** The Mission of the department of Electronics and Communication Engineering is to work closely with industry and research organizations to provide high quality education in both theoretical and practical applications of Electronics and Communication Engineering.

## PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

**Program Educational Objective 1: (PEO1)**
Graduates will be able to synthesize mathematics, science, engineering fundamentals, laboratory and work-based experiences to formulate and solve engineering problems in Electronics and Communication engineering domains and shall have proficiency in Computer-based engineering and the use of computational tools design of electronics systems.

**Program Educational Objective 2: (PEO2)**
Graduates will succeed in entry-level engineering positions within the core Electronics and Communication Engineering, computational or manufacturing firms in regional, national, or international industries and with government agencies.

**Program Educational Objective 3: (PEO3)**
Graduates will succeed in the pursuit of advanced degrees in Engineering or other fields where a solid foundation in mathematics, basic science, and engineering fundamentals is required.

**Program Educational Objective 4: (PEO4)**
Graduates will be prepared to communicate and work effectively on team based engineering projects and will practice the ethics of their profession consistent with a sense of social responsibility.

**Program Educational Objective 5: (PEO5)**
Graduates will be prepared to undertake Research and Development works in the areas of Electronics and Communication fields.

## PROGRAM OUTCOMES (POs)

| | |
|---|---|
| **PO1:** | **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems. |
| **PO2:** | **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences. |
| **PO3:** | **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations. |

| | |
|---|---|
| PO4: | **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions. |
| PO5: | **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations. |
| PO6: | **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice. |
| PO7: | **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development. |
| PO8: | **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice. |
| PO9: | **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings. |
| PO10: | **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions. |
| PO11: | **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments. |
| PO12: | **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change. |

## PROGRAM SPECIFIC OUTCOMES (PSOs)

| | |
|---|---|
| PSO1: | **Professional Skills:** An ability to understand the basic concepts in Electronics & Communication Engineering and to apply them to various areas, like Electronics, Communications, Signal processing, VLSI, Embedded systems etc., in the design and implementation of complex systems. |
| PSO2: | **Problem-Solving Skills:** An ability to solve complex Electronics and communication Engineering problems, using the latest hardware and software tools, along with analytical skills to arrive at cost effective and appropriate solutions. |
| PSO3: | **Successful Career and Entrepreneurship:** An understanding of social-awareness & environmental-wisdom along with ethical responsibility to have a successful career and to sustain passion and zeal for real-world applications using optimal resources as an Entrepreneur. |

# ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of the task would be incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crown all the efforts with success.

We avail this opportunity to express our deep sense of gratitude and hearty thanks to Sri CH.Venugopal Reddy, Chairman & Secretary of BIET, for providing congenial atmosphere and encouragement

We would like to thank Prof G. Kumaraswamy Rao, Former Director & O.S of DLRL Ministry of Defense St. Director, BIET, Dr. CH. Naga Prasad, Principal and college Dean Admin for having provided all the facilities and support.

We would like to thank our Department Coordinator Mr D. Sankara Reddy. Admin In charge and Mr. IV. Ravi Kumar, Academic In charge of their expert guidance and encouragement at various levels of our Project.

We are thankful to our Project Coordinator Dr. Anil Kumar Sahu, and Dr. Ashish Mishra Asst. Professor, Electronics and Communication Engineering for his support and cooperation throughout the process of this project.

We are thankful to our Guide Dr. Surendar Reddy , Assistance Professor, Electronics and Communication Engineering Dept, for his sustained inspiring Guidance and cooperation.

# TABLE OF CONTENTS

# ABSTRACT

Gesture based input systems are a new way to interact with computer system unlike traditional input devices (Keyboard and mouse). Webcam is used to recognize users hand based on his/her gestures. Necessary system operations are performed like play/pause music, play games, virtual keyboard etc,.

Gesture Based Input System recognizes hand gestures in real-time using the power of Neural Networks**.** Through this System a user can interact with the computer easily without any traditional devices like Keyboard, Mouse and etc. future development can be made to recognize face gestures so even disabled people can easily interact with computers.

In our system we propose some non-complex algorithm and hand gestures to decrease the hand gesture recognition complexity and would be more easy and simple to control real-time computer systems.

# INTRODUCTION

With the development in Computer Vision and Human Machine Interaction the Computer holds the most important role in our daily life. Human Computer Interaction can provide several advantages by introducing the different natural forms of device free communication. Gesture Based Input System is one of the several types of them to interact with humans. gestures are the natural form of action which we often used in our day to day life. But in computer applications to interact humans with machines the interaction with devices like keyboard, mouse etc. must be required. As the various hand gestures are frequently used by humans, the aim of this project is to reduce external hardware interaction which is required for computer application, and hence this makes the system more reliable for use with ease.

This paper implements gesture based input system recognition technique to handle multimedia applications. In this system, a gesture recognition scheme has been proposed as an interface between human and machine. In our system we represent some low-complexity algorithm and some hand gestures to decrease the gesture recognition complexity and which becomes easier to control real-time systems.

# SYSTEM ANALYSIS

## Objectives:42

Gesture based input systems are a new way to interact with computer systems unlike traditional input devices (Keyboard and mouse).

Webcam is used to recognize a user's hand based on his/her gestures. Necessary system operations are performed like play/pause music, play games, virtual keyboard etc.

## Problem Specification:

Gesture Based Input System implements technique to handle multimedia applications, a gesture recognition scheme has been proposed as an interface between human and machine.

In our system we represent some low-complexity algorithm and some hand gestures to decrease the gesture recognition complexity and which becomes easier to control real-time systems.

## Proposed System:

Using the Hand Gesture Recognition Database dataset from Kaggle (7 hand gestures), we follow a series of steps to eliminate noise, unnecessary space and background.

## Applications:

A VLC media player system that has been controlled by various hand gestures consists of play, pause, Full screen, and stop, increase volume, and decrease volume features.

Gesture control is the ability to recognize and interpret movements of the human body in order to interact with and control a computer system without direct physical contact.

## Modules and their functionalities:

The various modules of the project and their functionalities are described as below:

**Data Collection Mode:** Allows the user to collect train, test, or validation data on a variety of hand gestures

**Model Testing Mode:** Test the model's ability to discern between different gestures through real-time visualizations

**Music-Player/ Gesture Mode:** Use gestures to play music, pause music, and change the volume of music.

1. `Rad:` Play the song.
2. `Fist`: Play the song.
3. `Five`: Pauses the song
4. `Okay`: Increases the volume. Hold the pose to continue increasing the volume.
5. `Peace`: Decreases the volume. Hold the pose to continue decreasing the volume.
6. `Straight`: Stop the song.
7. `None`: Does nothing.

# SOFTWARE AND HARDWARE REQUIREMENTS

**Languages used:**

- Python.
- Machine Learning.

### HARDWARE REQUIREMENTS

Requires at least 8GB of RAM and a good processor.

## SOFTWARE REQUIREMENTS

## Packages used:

- OpenCV.
- Keras.
- PyGame.
- NumPy.
- TensorFlow.

## Plugins:

- my_model_weights.h5.

# LITERATURE SURVEY

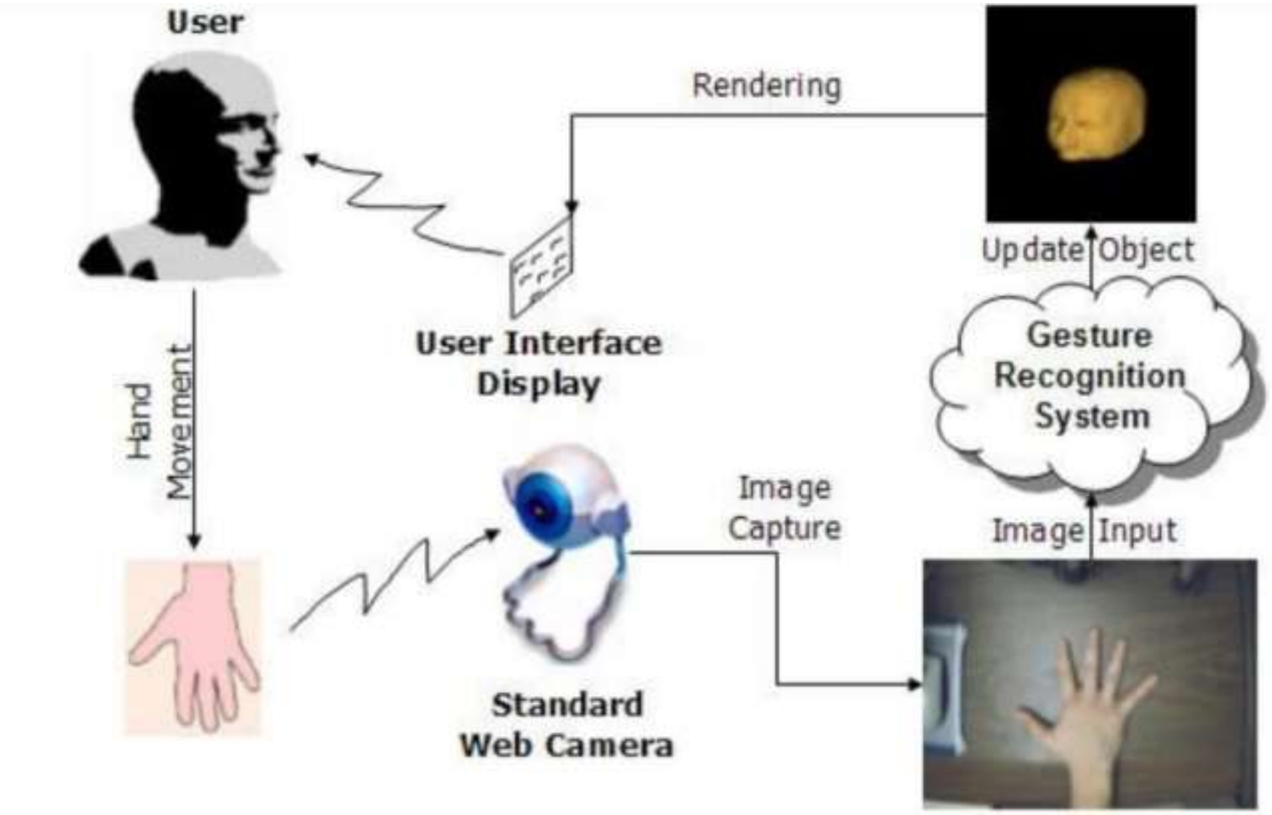
“Hand Gesture Recognition System” proposed a system which recognizes the unknown input gestures by using hand masking. This system is applied to recognize the single gesture. There is assumption of stationary background so that the system will have a smaller search region for tracking. This system only controls the finger using it on a webcam.

Controlling of media using hand gesture recognition this system uses various hand gestures as input to operate the media. This system uses single hand gestures and its directional motion which defines a particular gesture for the above mentioned application. In this system convolutional neural networks are used for classification. This system only supports media players and not any others. This system uses a database that consists of various hand gestures and then the input was compared with this stored image and accordingly media was controlled.

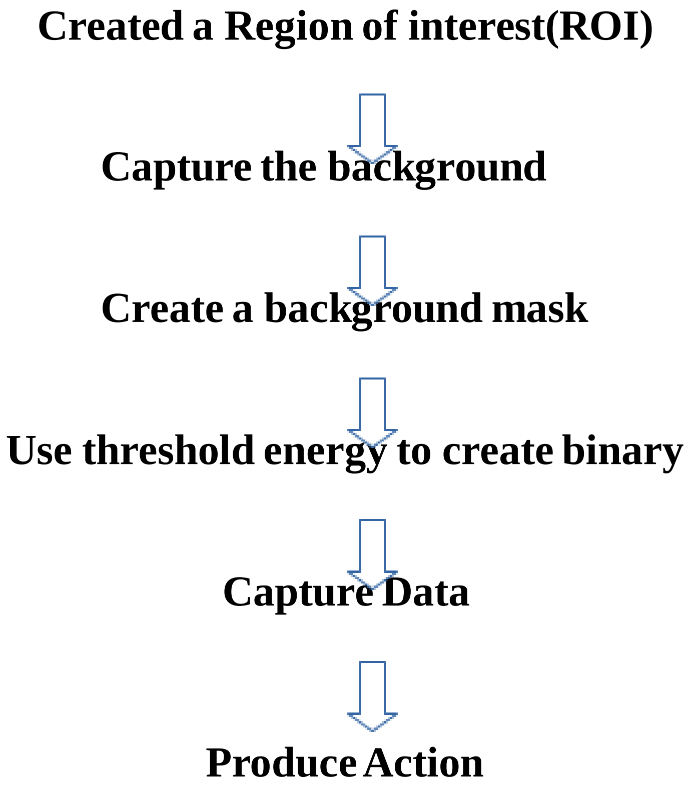
In 2014, Viraj Shinde, Tushar Bacchav, Jitendra Pawar and Mangesh Sanap developed “Hand Gesture Recognition System Using Camera”. They focus on using pointing behaviors for a natural interface to classify the dynamic hand gesture, they developed a simple and fast motion history image based method. This paper presents low complexity algorithms and gestures recognition complexity and is more suitable for controlling real time computer systems. It is applicable only for the application Of powerpoint presentation.

# DESIGN

## System Architecture:



# FLOW CHART



Created a Region of interest(ROI)

Capture the background

Create a background mask

Use threshold energy to create binary

Capture Data

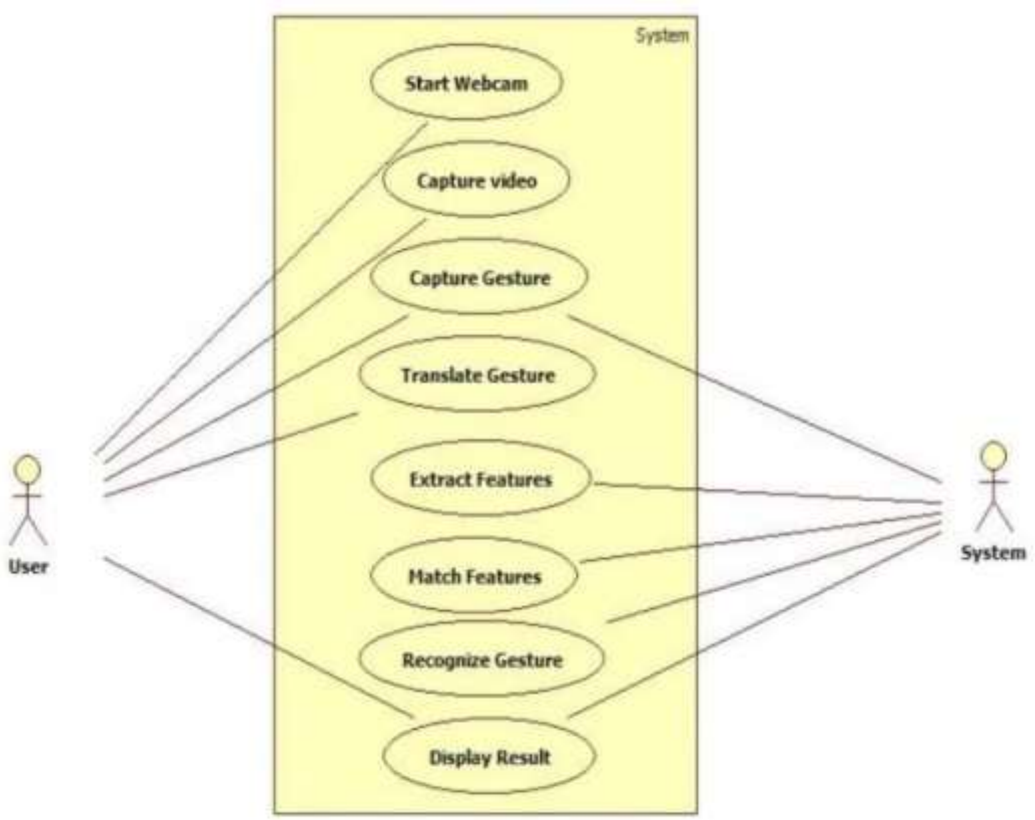
Produce Action

# ALGORITHM

```
Create a path for the data collection
Load dependencies
Background capture model
If a background has been captured
    Capture frame
    Flip frame
Applying smoothing filter that keeps edges sharp
Remove background
Selecting region of interest
Converting image to gray
Blurring the image
Threshold the image
Predicting and storing predictions
Draw new frame with graphs
Draw new data frame with mask
Show the frame
If q is pressed,
    quit the app
If r is pressed,
    reset the background
If d is pressed,
    go into to data collection mode
```

```
If p is pressed,
        predict
If g is pressed
                go into music player/gesture mode
Release the cap and close all windows if loop is broken
```

## UML DIAGRAM



# IMPLEMENTATION

```
print('Loading dependencies') #

            Importing Modules
      import os
      import time
      import numpy as np
      import cv2
      import time
      from tensorflow.keras.models import Sequential
      from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
      import tensorflow as tf
      import pygame
            #GLOBAL VARIABLES
      # Layout/FrontEnd of Frame
      IMAGEHEIGHT = 480
      IMAGEWIDTH = 640
      ROIWIDTH = 256
      LEFT = int(IMAGEWIDTH / 2 - ROIWIDTH / 2)
      RIGHT = LEFT + ROIWIDTH
      TOP = int(IMAGEHEIGHT / 2 - ROIWIDTH / 2)
      BOTTOM = TOP + ROIWIDTH
      SCOREBOXWIDTH = 320
      BARCHARTLENGTH = SCOREBOXWIDTH - 50
      BARCHARTTHICKNESS = 15
      BARCHARTGAP = 20
      BARCHARTOFFSET = 8
      FONT = cv2.FONT_HERSHEY_SIMPLEX
```

```
            # Model variables
NUMBEROFGESTURES = 8
WEIGHTS_URL = './my_model_weights.h5'
GESTURE_ENCODING = {0: 'fist', 1: 'five', 2: 'none', 3: 'okay', 4: 'peace', 5: 'rad', 6: 'straight', 7: 'thumbs'}
            # OpenCV image processing variables
BGSUBTHRESHOLD = 50
THRESHOLD = 50
            # Gesture Mode variables
GESTUREMODE = False # Don't ever edit this!
GESTURES_RECORDED = [10, 10, 10, 10, 10, 10, 10, 10, 10, 10]
SONG = 'The Beatles - I Saw Her Standing There'
ACTIONS_GESTURE_ENCODING = {'fist': 'Play/Unpause', 'five': 'Pause', 'none': 'Do Nothing', 'okay': 'Increase Volume', 'peace': 'Decrease Volume', 'rad': "Load Song", 'straight': "Stop", "thumbs": "NA"}
            # Data Collection Mode variables
DATAMODE = False # Don't ever edit this!
WHERE = "train"
GESTURE = "okay"
NUMBERTOCAPTURE = 100
            # Testing Predictions of Model Mode variables
PREDICT = False # Don't ever edit this!
HISTORIC_PREDICTIONS = [np.ones((1, 8)), np.ones((1, 8)), np.ones((1, 8)), np.ones((1, 8)), np.ones((1, 8))]
IMAGEAVERAGING = 5
            #MUSIC PLAYER CLASS
            # Music Player Class. Contains functions to load, play, pause, adjust volume, and stop music
class MusicMan(object):
    def __init__(self, file):
```

```
    print("Loading music player...")
    pygame.mixer.init()
    self.player = pygame.mixer.music
    self.song = file
    self.file = f"./music/{file}.mp3"
    self.state = None
  def load(self):
    if self.state is None:
      self.player.load(self.file)
      self.state = "loaded"
  def play(self):
    if self.state == "pause":
      self.player.unpause()
      self.state = "play"
    elif self.state in ["loaded", "stop"]:
      self.player.play()
      self.state = "play"
  def pause(self):
    if self.state == "play":
      self.player.pause()
      self.state  =  "pause"
 def increase_volume(self):
    if self.state is not None:
      self.player.set_volume(self.player.get_volume() - 0.02)
  def decrease_volume(self):
        if self.state is not None:
            self.player.set_volume(self.player.get_volume() + 0.02)
    def stop(self):
        if self.state == "play":
            self.player.stop()
            self.state = "stop"
            #USEFUL FUNCTIONS
            # Creating a path for storing data
def           create_path(WHERE,           GESTURE):
    print("Creating path to store data for collection...")
    DIR_NAME = f"./data/{WHERE}/{GESTURE}"
    if not os.path.exists(DIR_NAME):
        os.makedirs(DIR_NAME)
    if len(os.listdir(DIR_NAME)) == 0:
        img_label = int(1)
    else:
        img_label = int(sorted(os.listdir(DIR_NAME), key=len)[-1][:-4])
    return img_label
            # Creating our deep learning model to recognize the hand image
def
    create_model(outputSize):
    model = Sequential()
    model.add(Conv2D(filters=32, kernel_size=(3, 3), activation='relu', input_shape=(256, 256, 1)))
    model.add(Conv2D(filters=32, kernel_size=(3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=2))
    model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
    model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=2))

    model.add(Conv2D(filters=128, kernel_size=(3, 3), activation='relu'))

    model.add(Conv2D(filters=128, kernel_size=(3, 3), activation='relu'))

    model.add(MaxPooling2D(pool_size=2))

    model.add(Flatten())

    model.add(Dropout(rate=0.5))

    model.add(Dense(512, activation='relu'))

    model.add(Dense(units=outputSize, activation='softmax'))

    model.compile(optimizer=tf.keras.optimizers.Adam(lr=1e-4),

    loss='categorical_crossentropy',
metrics=['accuracy'])

    return model

            # Function to load the model

def load_model(outputSize, weight_url):

             # Loading the model

    modelName = 'Hand Gesture Recognition'

    print(f'Loading model {modelName}')

    model = create_model(NUMBEROFGESTURES)

    model.load_weights(weight_url)

    return model, modelName

# Function that counts the number of times the last element is repeated (starting at the end of the
array)

# without any gap. Returns the count and the percentage (count/length of array)

# For example [1, 1, 1, 2] would return 1, 0.25 while [1,1,2,2] would return 2, 0.5

# [1,1,1,1] would return 4, 1

def find_last_rep(array):

    last_element = array[-1]

    count = 0
    for ele in reversed(array):
        if last_element == ele:
            count += 1
        else:
            return count, count / len(array)
    return count, count / len(array)
# Draw frame to side of video capture. Populate this frame with front
# end for gesture and prediction modes
def drawSideFrame(historic_predictions, frame, modelName, label):
    global GESTURES_RECORDED
            # Streaming
    dataText = "Streaming..."
            # Creating the score frame
    score_frame = 200 * np.ones((IMAGEHEIGHT, SCOREBOXWIDTH, 3), np.uint8)
            # GESTURE MODE front end stuff
    if GESTUREMODE:
        GESTURES_RECORDED.append(label)
        GESTURES_RECORDED = GESTURES_RECORDED[-10:]
        count, percent_finished = find_last_rep(GESTURES_RECORDED)
            # If the array recording the timeline of gestures only contains one gesture
        if len(set(GESTURES_RECORDED)) == 1:
            # See the command
            command = GESTURE_ENCODING[GESTURES_RECORDED[-1]]
            # Use the Music Class to play the command accordingly
            if command == "fist":
                music.play()
            elif command == "five":
                music.pause()
            elif command == "none":
                pass
            elif command == "okay":
                music.decrease_volume()
            elif command == "peace":
                music.increase_volume()
            elif command == "rad":
                music.load()
            elif command == "straight":
                music.stop()
            elif command == "thumbs":
                pass
        # Colors of the bar graph showing progress of the gesture recognition
        if percent_finished == 1:
            color = (0, 204, 102)
        else:
            color = (20, 20, 220)
        # Drawing the bar chart
        start_pixels = np.array([20, 175])
        text = '{} ({}%)'.format(GESTURE_ENCODING[GESTURES_RECORDED[-1]],
percent_finished * 100)
        cv2.putText(score_frame, text, tuple(start_pixels), FONT, 0.5, (0, 0, 0), 2, cv2.LINE_AA)
        chart_start = start_pixels + np.array([0, BARCHARTOFFSET])
        length = int(percent_finished * BARCHARTLENGTH)
        chart_end = chart_start + np.array(
            [length, BARCHARTTHICKNESS])
        cv2.rectangle(score_frame, tuple(chart_start), tuple(chart_end), color, cv2.FILLED)
        # Adding text
        cv2.putText(score_frame, 'Press G to turn of gesture mode', (20, 25), FONT, 0.55, (0, 0, 0), 2, cv2.LINE_AA)

        cv2.putText(score_frame, f'Model : {modelName}', (20, 50), FONT, 0.55, (0, 0, 0), 2, cv2.LINE_AA)

        cv2.putText(score_frame, f'Data source : {dataText}', (20, 75), FONT, 0.55, (0, 0, 0), 2, cv2.LINE_AA)

        cv2.putText(score_frame, f'Song : {music.song}', (20, 100), FONT, 0.55, (0, 0, 0), 2, cv2.LINE_AA)

        cv2.putText(score_frame, f'Label : {GESTURE_ENCODING[label]}', (20, 125), FONT, 0.55, (0, 0, 0), 2, cv2.LINE_AA)

        cv2.putText(score_frame, f'Action : {ACTIONS_GESTURE_ENCODING[GESTURE_ENCODING[label]]}', (20, 150), FONT, 0.55,

                    (0, 0, 0), 2, cv2.LINE_AA)

    # PREDICT MODE front end stuff

    elif PREDICT:

        cv2.putText(score_frame, 'Press P to stop testing predictions', (20, 25), FONT, 0.55, (0, 0, 0), 2, cv2.LINE_AA)

        cv2.putText(score_frame, f'Model : {modelName}', (20, 50), FONT, 0.55, (0, 0, 0), 2, cv2.LINE_AA)

        cv2.putText(score_frame, f'Data source : {dataText}', (20, 75), FONT, 0.55, (0, 0, 0), 2, cv2.LINE_AA)

     # Converting predictions into an array

        predictions = np.array(historic_predictions)

        # Taking a mean of historic predictions

        average_predictions = predictions.mean(axis=0)[0]

        sorted_args = list(np.argsort(average_predictions))

        # Drawing the prediction probabilities in a bar chart

        start_pixels = np.array([20, 150])

        for count, arg in enumerate(list(reversed(sorted_args))):

            probability = round(average_predictions[arg])

            predictedLabel = GESTURE_ENCODING[arg]

            if arg == label:
                color = (0, 204, 102)

            else:

                color = (20, 20, 220)

            text = '{}. {} ({}%)'.format(count + 1, predictedLabel, probability * 100)

            cv2.putText(score_frame, text, tuple(start_pixels), FONT, 0.5, (0, 0, 0), 2, cv2.LINE_AA)

            chart_start = start_pixels + np.array([0, BARCHARTOFFSET])

            length = int(probability * BARCHARTLENGTH)

            chart_end = chart_start + np.array(

                [length, BARCHARTTHICKNESS])
            cv2.rectangle(score_frame, tuple(chart_start), tuple(chart_end), color, cv2.FILLED)
            start_pixels = start_pixels + np.array([0, BARCHARTGAP + BARCHARTTHICKNESS + BARCHARTOFFSET])

    # No mode active front end stuff

    else:

        cv2.putText(score_frame, 'Press P to test model', (20, 25), FONT, 0.55, (0, 0, 0), 2, cv2.LINE_AA)

        cv2.putText(score_frame, 'Press G for Gesture Mode', (20, 50), FONT, 0.55, (0, 0, 0), 2, cv2.LINE_AA)

        cv2.putText(score_frame, 'Press R to reset background', (20, 75), FONT, 0.55, (0, 0, 0), 2, cv2.LINE_AA)

        cv2.putText(score_frame, f'Model : {modelName}', (20, 100), FONT, 0.55, (0, 0, 0), 2, cv2.LINE_AA)

        cv2.putText(score_frame, f'Data source : {dataText}', (20, 125), FONT, 0.55, (0, 0, 0), 2, cv2.LINE_AA)

        music.stop()

    return np.hstack((score_frame, frame))

# The controller/frontend that subtracts the background

def capture_background():

    cap = cv2.VideoCapture(0)

    while True:

        ret, frame = cap.read()
        if not ret:
            break
        frame = cv2.flip(frame, 1)
        cv2.putText(frame, "Press B to capture background.", (5, 50), FONT, 0.7, (255, 255, 255), 2, cv2.LINE_AA)
        cv2.putText(frame, "Press Q to quit.", (5, 80), FONT, 0.7, (255, 255, 255), 2, cv2.LINE_AA)
        cv2.rectangle(frame, (LEFT, TOP), (RIGHT, BOTTOM), (0, 0, 0), 1)
        cv2.imshow('Capture Background', frame)
        k = cv2.waitKey(5)
        # If key b is pressed
        if k == ord('b'):
            bgModel = cv2.createBackgroundSubtractorMOG2(0, BGSUBTHRESHOLD)
            # cap.release()
            cv2.destroyAllWindows()
            break
        # If key q is pressed
        elif k == ord('q'):
            bgModel = None
            cap.release()
            cv2.destroyAllWindows()
            break
    return bgModel
# Remove the background from a new frame
def remove_background(bgModel, frame):
    fgmask = bgModel.apply(frame, learningRate=0)
    kernel = np.ones((3, 3), np.uint8)
    eroded = cv2.erode(fgmask, kernel, iterations=1)
    res = cv2.bitwise_and(frame, frame, mask=eroded)
    return res
# Show the processed, thresholded image of hand in side frame on right
def drawMask(frame, mask):
    mask = cv2.cvtColor(mask, cv2.COLOR_GRAY2BGR)
    mask_frame = 200 * np.ones((IMAGEHEIGHT, ROIWIDTH + 20, 3), np.uint8)
    mask_frame[10:266, 10:266] = mask
    cv2.putText(mask_frame, "Mask",
                (100, 290), FONT, 0.7, (0, 0, 0), 2, cv2.LINE_AA)
    return np.hstack((frame, mask_frame))
if _name____== '_main_':
    # Create a path for the data collection
    img_label = create_path(WHERE, GESTURE)
    # Load dependencies
    model, modelName = load_model(NUMBEROFGESTURES, WEIGHTS_URL)
    music = MusicMan(SONG)
    print("Starting live video stream...")
    # Background capture model
    bgModel = capture_background()
    # If a background has been captured
    if bgModel:
        cap = cv2.VideoCapture(0)
       while True:
            # Capture frame
            label, frame = cap.read()
        # Flip frame
           frame = cv2.flip(frame, 1)
            # Applying smoothing filter that keeps edges sharp
            frame = cv2.bilateralFilter(frame, 5, 50, 100)
            cv2.rectangle(frame, (LEFT, TOP), (RIGHT, BOTTOM), (0, 0, 0), 1)
            # Remove background
           no_background = remove_background(bgModel, frame)
            # Selecting region of interest
           roi = no_background[TOP:BOTTOM, LEFT:RIGHT]
            # Converting image to gray
            gray = cv2.cvtColor(roi, cv2.COLOR_BGR2GRAY)
            # Blurring the image
            blur = cv2.GaussianBlur(gray, (41, 41), 0)
            # Thresholding the image
            ret, thresh = cv2.threshold(blur, THRESHOLD, 255, cv2.THRESH_BINARY)
            # Predicting and storing predictions
            prediction = model.predict(thresh.reshape(1, 256, 256, 1) / 255)
            prediction_final = np.argmax(prediction)
            HISTORIC_PREDICTIONS.append(prediction)
            HISTORIC_PREDICTIONS = HISTORIC_PREDICTIONS[-IMAGEAVERAGING:]
           # Draw new frame with graphs
            new_frame = drawSideFrame(HISTORIC_PREDICTIONS, frame, 'Gesture Model',
prediction_final)
           # Draw new dataframe with mask
            new_frame = drawMask(new_frame, thresh)
            # If Datamode
            if DATAMODE:
                time.sleep(0.03)
                cv2.imwrite(f"./data/{WHERE}/{GESTURE}" + f"/{img_label}.png", thresh)
                cv2.putText(new_frame, "Photos Captured:", (980, 400), FONT, 0.7, (0, 0, 0), 2,

cv2.LINE_AA)

                cv2.putText(new_frame, f"{i}/{NUMBERTOCAPTURE}", (1010, 430), FONT, 0.7, (0, 0,
0), 2, cv2.LINE_AA)
```

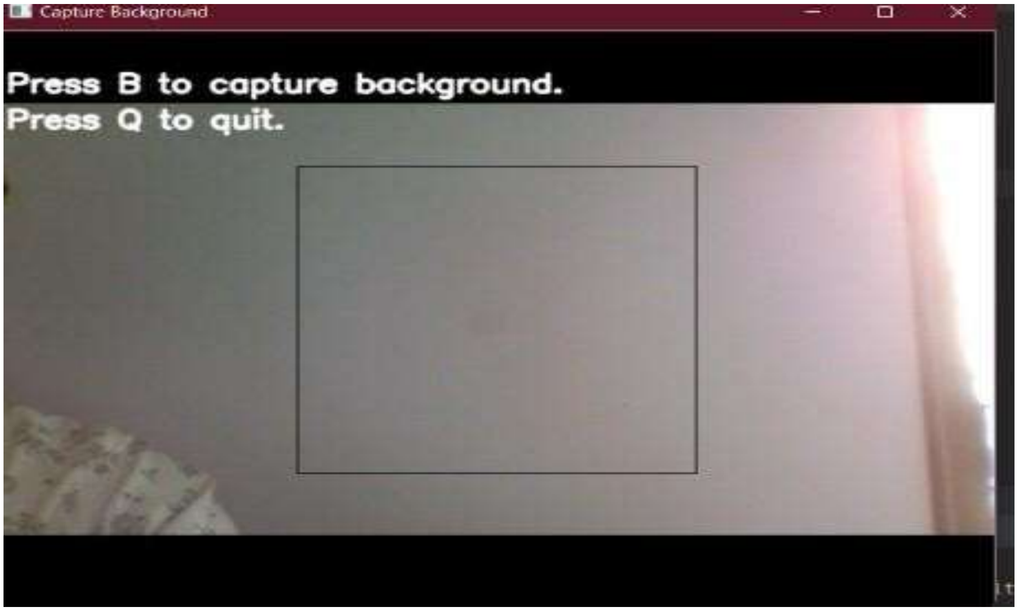
```
    img_label += 1

    i += 1

    if i > NUMBERTOCAPTURE:

        cv2.putText(new_frame, "Done!", (980, 400), FONT, 0.7, (0, 0, 0), 2, cv2.LINE_AA)

        DATAMODE = False

        i = None

else:

    cv2.putText(new_frame, "Press D to collect", (980, 375), FONT, 0.6, (0, 0, 0), 2, cv2.LINE_AA)

    cv2.putText(new_frame, f"{NUMBERTOCAPTURE} {WHERE} images", (980, 400), FONT, 0.6, (0, 0, 0), 2,cv2.LINE_AA)

    cv2.putText(new_frame, f"for gesture {GESTURE}", (980, 425), FONT, 0.6, (0, 0, 0), 2, cv2.LINE_AA)

# Show the frame

cv2.imshow('Gesture Jester', new_frame)

key = cv2.waitKey(5)

# If q is pressed, quit the app

if key == ord('q'):

    break

# If r is pressed, reset the background

if key == ord('r'):

    PREDICT = False

    DATAMODE = False

    cap.release()

    cv2.destroyAllWindows()

    bgModel = capture_background()

    cap = cv2.VideoCapture(0)

# If d is pressed, go into to data collection mode

if key == ord('d'):
```

```
            PREDICT = False
            GESTUREMODE = False
            DATAMODE = True
            i = 1
        # If p is pressed, predict
        if key == ord('p'):
            GESTUREMODE = False
            DATAMODE = False
            PREDICT = not PREDICT
       # If g is pressed go into music player/gesture mode
        if key == ord('g'):
            DATAMODE  = False
            PREDICT = False
            GESTUREMODE = not GESTUREMODE
    # Release the cap and close all windows if loop is broken
    cap.release()
cv2.destroyAllWindows()
```

# TESTING

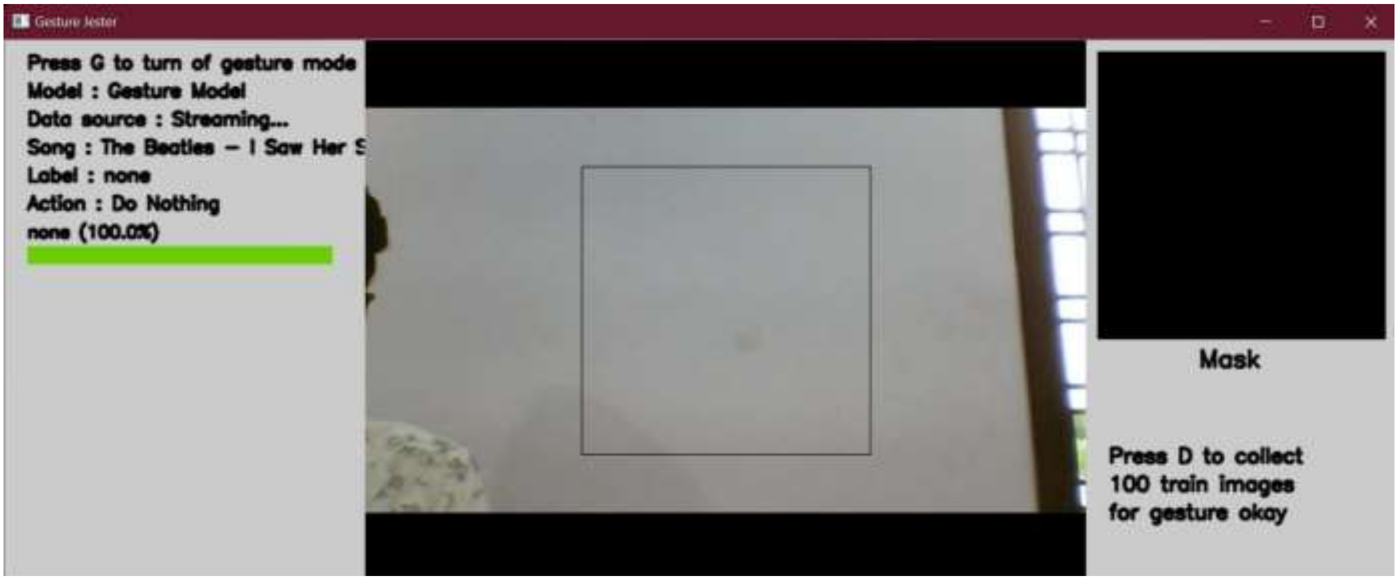
| S.no | Description | Gestures | Status |
|---|---|---|---|
| 1) | Capturing, masking Background | Background is captured and masked accordingly.(Fig-1) | Success |
| 2) | Load song | Song is loaded by **rad gesture** (fig-3) | Success |
| 3) | Play song | Song is played by **fist gesture** (fig-4) | Success |
| 4) | Increase volume | Volume is increased by **okay gesture**(fig-6) | Success |
| 5) | Decrease volume | Volume is decreased by **peace gesture**(fig-5) | Success |
| 6) | Pause song | Song is paused by **five gesture**(fig-7) | Success |
| 7) | Stop song | Song is stopped by **straight gesture**(fig-8) | Success |
| 8) | Doing Nothing | **None**(fig-2) | Success |
| 9) | Action not applicable | **Thumb gesture**(fig-9) | Success |

# SCREENSHOTS



```
Press B to capture background.
Press Q to quit.
```

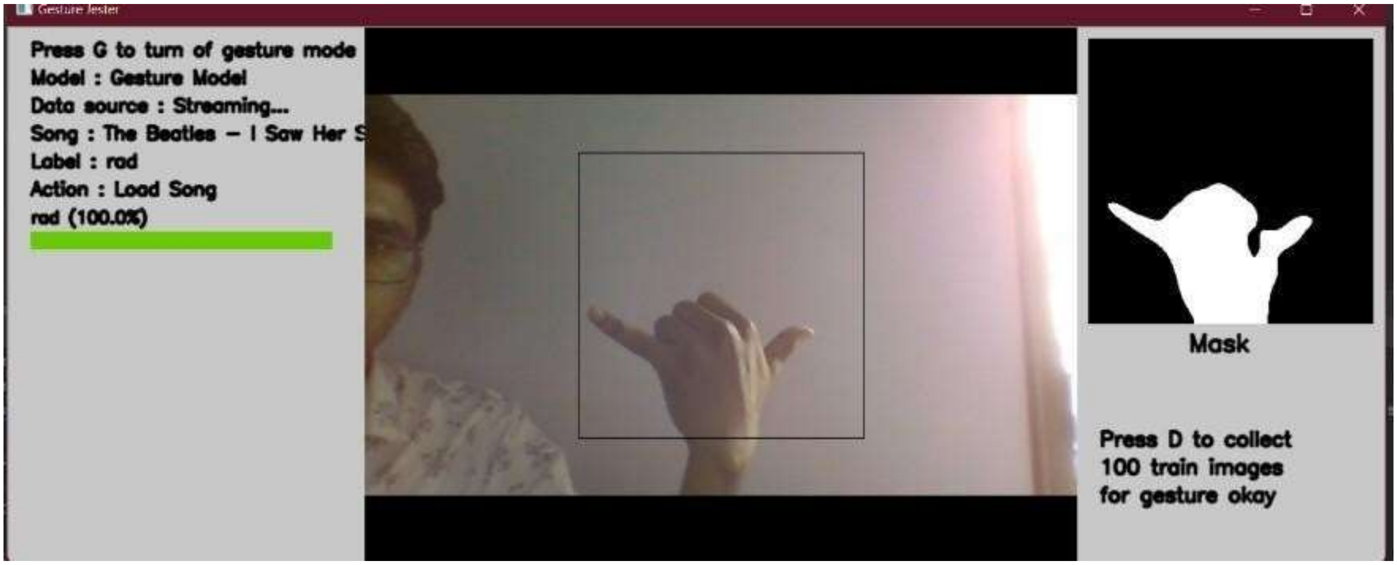
FIG-1



```
Press G to turn of gesture mode
Model : Gesture Model
Data source : Streaming...
Song : The Beatles – I Saw Her S
Label : none
Action : Do Nothing
none (100.0%)
```

```
Press D to collect
100 train images
for gesture okay
```

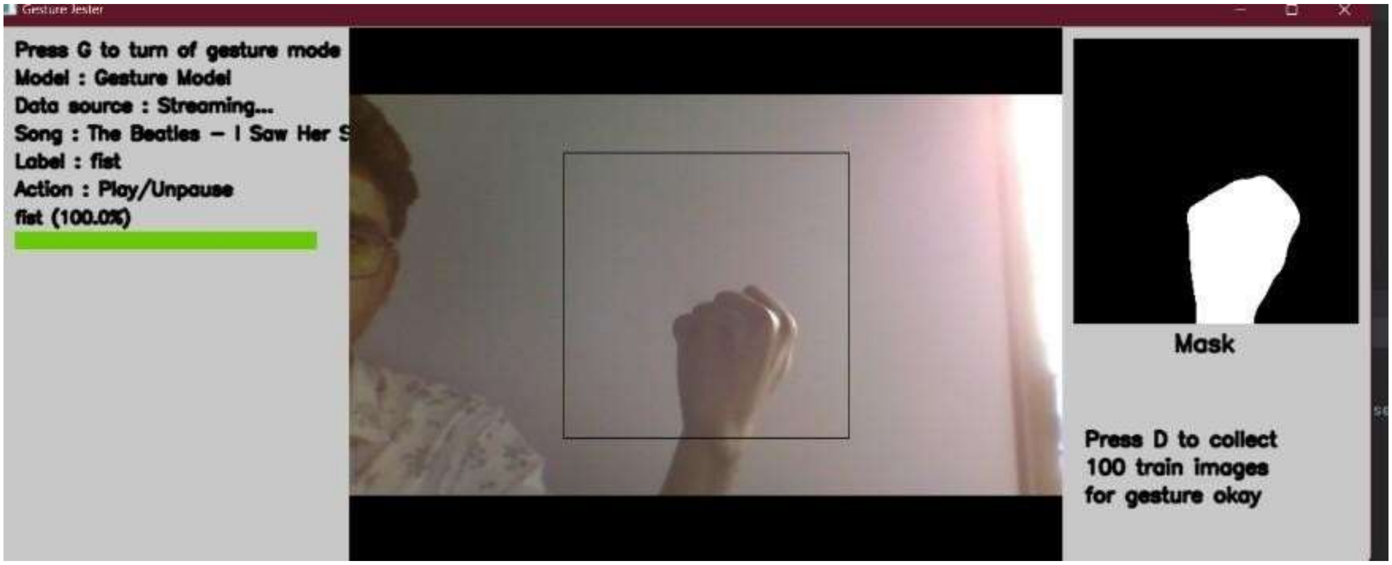
FIG-2



Press G to turn of gesture mode
Model : Gesture Model
Data source : Streaming...
Song : The Beatles – I Saw Her S
Label : rad
Action : Load Song
rad (100.0%)

Press D to collect
100 train images
for gesture okay

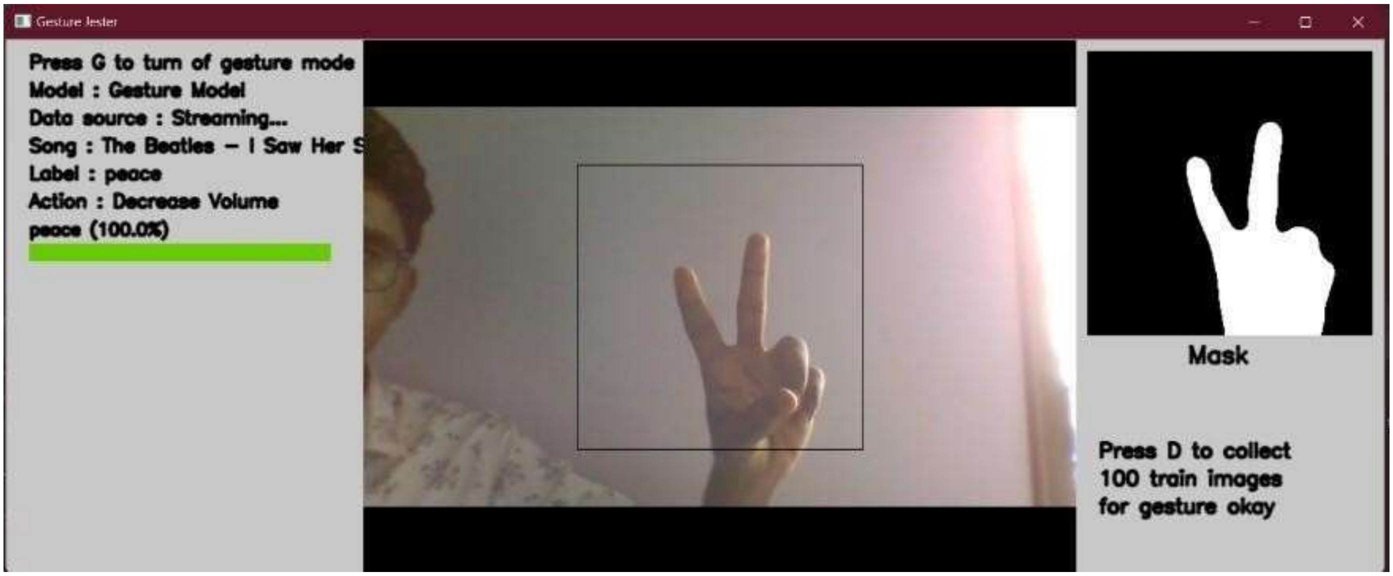
FIG-3



Press G to turn of gesture mode
Model : Gesture Model
Data source : Streaming...
Song : The Beatles – I Saw Her S
Label : fist
Action : Play/Unpause
fist (100.0%)

Press D to collect
100 train images
for gesture okay

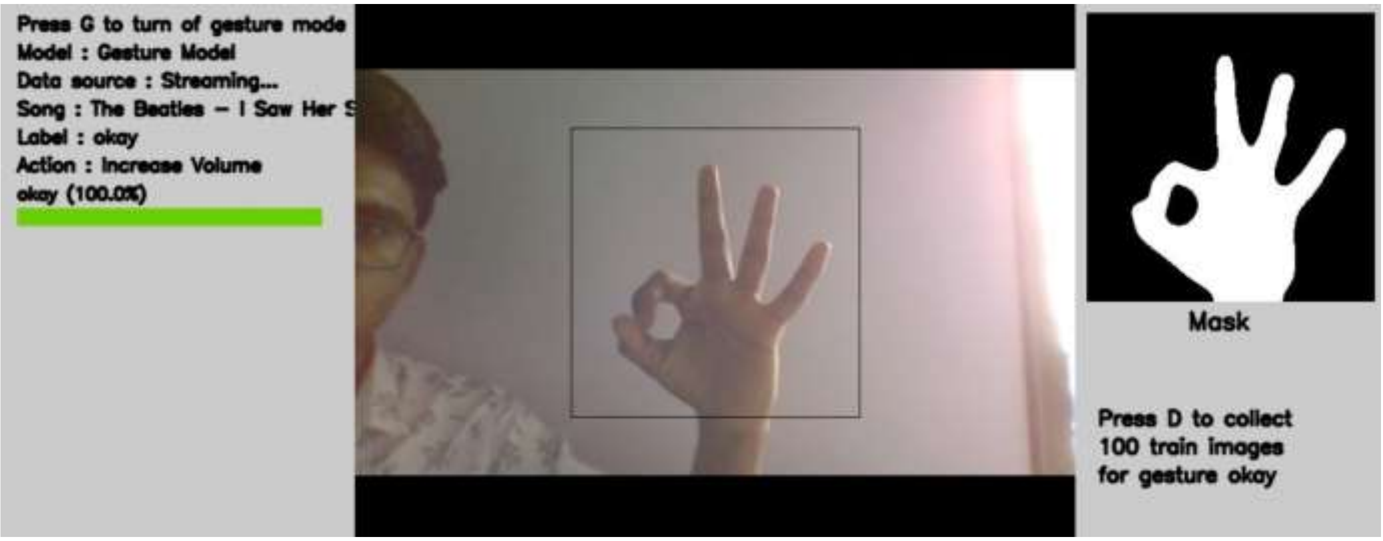
FIG-4



Press G to turn of gesture mode
Model : Gesture Model
Data source : Streaming...
Song : The Beatles – I Saw Her S
Label : peace
Action : Decrease Volume
peace (100.0%)

Press D to collect
100 train images
for gesture okay

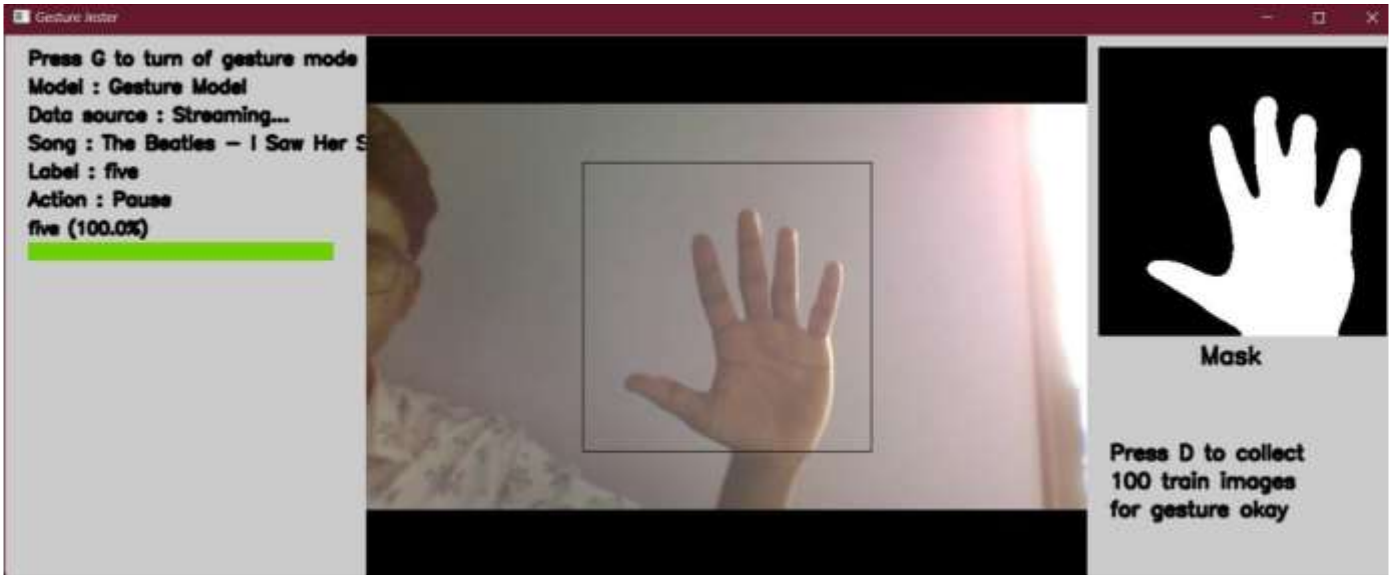
FIG-5



Press G to turn of gesture mode
Model : Gesture Model
Data source : Streaming...
Song : The Beatles – I Saw Her S
Label : okay
Action : Increase Volume
okay (100.0%)

Press D to collect
100 train images
for gesture okay

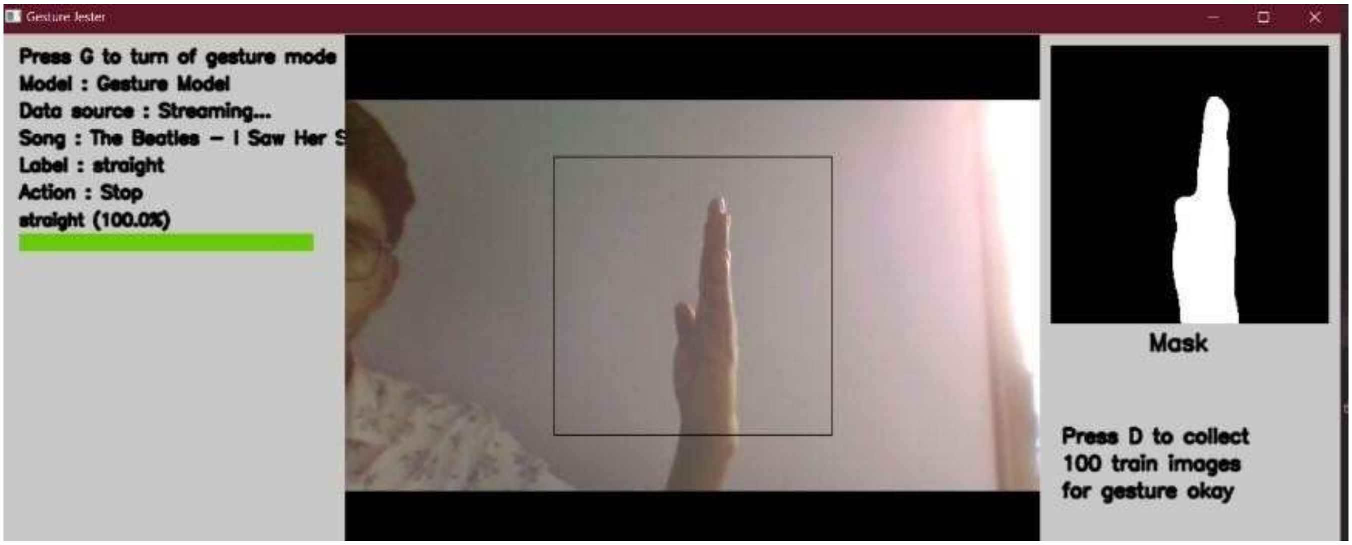
FIG-6



Press G to turn of gesture mode
Model : Gesture Model
Data source : Streaming...
Song : The Beatles – I Saw Her S
Label : five
Action : Pause
five (100.0%)

Press D to collect
100 train images
for gesture okay

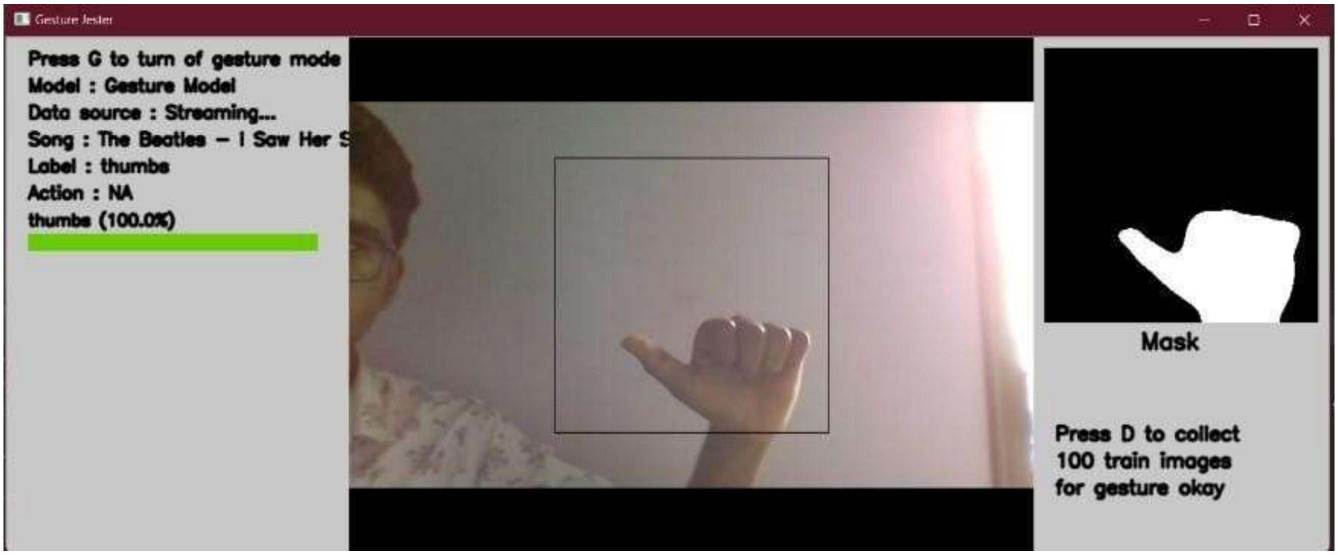
FIG-7



Press G to turn of gesture mode
Model : Gesture Model
Data source : Streaming...
Song : The Beatles – I Saw Her S
Label : straight
Action : Stop
straight (100.0%)

Press D to collect
100 train images
for gesture okay

FIG-8



Press G to turn of gesture mode
Model : Gesture Model
Data source : Streaming...
Song : The Beatles – I Saw Her S
Label : thumbs
Action : NA
thumbs (100.0%)

Press D to collect
100 train images
for gesture okay

FIG-9

# 10. CONCLUSION AND FUTURE ENHANCEMENT

## CONCLUSION

Through this System a user can interact with the computer easily without any traditional devices like Keyboard, Mouse and etc. future development can be made to recognize face gestures so even disabled people can easily interact with computer

## FUTURE SCOPE

As a future prospect of this research we are also going to investigate the large number of gestures with different persons and motion type hand gestures are developed. We are also going to generalize our system so that it can be useful for other different media players available in the market.

# REFERENCES

Google Medium

Kaggle Dataset

OpenCv and autogui

CNN Documentation

N.Krishna Chaitanya and R.Janardhan Rao "Controlling of windows media player using hand recognition system", The International Journal Of Engineering And Science (IJES), Vol. 3, Issue 12, Pages 01-04, 2014.

For demo video click here

https://youtu.be/LHZYJFbuRwY

# APPENDICES

## Abbreviations:

**RAM**: Random Access Memory.

**GB**: Gigabyte.

## List of Figures:

## List of tables: