Hand Gestures

***Project report***

J Himasree(R21EJ107)

Aashrita Kolisetty(R21EJ061)

Sakshi Upadhye (R21EJ073)

Shaik Heera Eram (R21EJ070)

Name of School or Institution-*C&IT*

Course Name-*Machine Learning*

Instructor *- Dr Muthireddy P*

Date-*20/11/23*

Abstract 3

Introduction 4

Objectives 5 Problem definition 6

System Requirements 7

Implementation 9

Results 13

Conclusion 14

References 16

**Abstract**

In the realm of human-computer interaction, this project introduces a hand gesture recognition system designed to revolutionize user control over media playback and cursor manipulation. Departing from traditional input devices, the system leverages computer vision techniques to interpret dynamic hand gestures, offering users an intuitive and expressive means of interfacing with computing systems.

The algorithm, built upon OpenCV, Mediapipe, and PyAutoGUI libraries, facilitates real-time gesture recognition, seamlessly translating hand movements into actionable commands. Beyond media control, the project explores the potential applications of gestural interfaces across various technological domains, envisioning a future where the language of hand gestures seamlessly integrates with the language of technology.

# 

# Introduction

As technology continues to evolve, the quest for more natural and intuitive methods of human-computer interaction becomes increasingly prominent. Traditional input devices, while effective, often fall short in capturing the fluidity and expressiveness inherent in human gestures. This project addresses this gap by introducing a hand gesture recognition system, aiming to redefine the way users interact with computing devices.

The fundamental premise of the project is to enable users to control media playback and cursor movement through dynamic hand gestures captured by a standard camera. Unlike conventional input methods, such as keyboards and mice, which can be restrictive and lack immediacy, hand gestures provide an immersive and instinctive channel for communication.

The core of the system lies in a sophisticated hand-tracking algorithm, implemented using computer vision libraries such as OpenCV and Mediapipe. This algorithm not only identifies and tracks the landmarks of the hand in real-time but also interprets specific gestures associated with media control and cursor manipulation. The integration with the PyAutoGUI library facilitates the translation of these gestures into tangible actions, creating a seamless bridge between user intent and system response.

**Objectives**

1. **Gesture-Based Interface Development:**
   * Design and implement a system capable of capturing and interpreting hand gestures in real-time.
2. **Hand-Tracking Algorithm Implementation:**
   * Develop a robust hand-tracking algorithm using computer vision techniques, leveraging OpenCV and Mediapipe libraries.
3. **Media Control Gestures:**
   * Define and implement specific hand gestures for controlling media playback, including play, pause, and resume functionalities.
4. **Integration with PyAutoGUI:**
   * Integrate the recognized gestures with appropriate functions from the PyAutoGUI library for executing actions in the computing environment.
5. **Real-Time Interaction:**
   * Ensure real-time responsiveness of the system, allowing users to control media playback and cursor movement fluidly and intuitively.
6. **User-Friendly Interaction:**
   * Prioritize a user-friendly interface, minimizing the learning curve for users to adapt to the gesture-based control system.
7. **Gesture Recognition Accuracy:**
   * Optimize the accuracy of gesture recognition to enhance the reliability and precision of the system in interpreting user gestures.

.

# Problem definition

The way we interact with computers using keyboards and mice can sometimes be a bit complicated. This project tackles the challenge of making computer interactions more natural and easy by using hand gestures. The goal is to create a system that recognizes and responds to hand movements, allowing users to control things like volume, media playback, and cursor movement just by using their hands. The problem is to figure out the best way to make this happen and to ensure that it works smoothly for users, providing a simpler and more intuitive way to interact with computers.

·

# 

# 

# 

# 

# 

# 

# 

**System requirements**

To deploy and use the hand gesture recognition system for media control and cursor manipulation, the following system requirements must be met:

1. **Hardware Requirements:**
   1. Camera: A webcam or any suitable camera capable of capturing video feed. The camera should be positioned to capture the user's hand gestures effectively.
   2. Computer: A computer with sufficient processing power to handle real-time video processing. The system should meet or exceed the minimum requirements for running computer vision libraries such as OpenCV and Mediapipe.
2. **Software Requirements:**
   1. Operating System: The system should run on a compatible operating system. The code provided assumes compatibility with Windows, but adjustments may be necessary for other operating systems.
   2. Python: The latest version of Python should be installed on the system. The code utilizes Python for implementing the hand gesture recognition system.
   3. Libraries:
      1. OpenCV: The Open Source Computer Vision Library is essential for image and video processing.
      2. Mediapipe: A library for building real-time multi-modal perceptual systems, used for hand tracking.
      3. Numpy: A library for numerical operations, used for mathematical calculations in the code.
      4. PyAutoGUI: A library for programmatically controlling the mouse and keyboard, utilized for executing actions based on recognized gestures.
3. **Dependencies Installation:**
   1. The required libraries (OpenCV, Mediapipe, Numpy, PyAutoGUI) should be installed using package managers. The provided code includes a comment with the necessary command (pip install opencv-python numpy mediapipe pyautogui).
4. **Screen Resolution:**
   1. The system assumes a standard screen resolution. Adjustments may be required in the code if the screen resolution differs significantly.
5. **Internet Connection:**
   1. An internet connection is required to download and install the necessary libraries if not already available.
6. **Environment Setup:**
   1. The system should provide a suitable environment for capturing hand gestures without significant interference. Adequate lighting is essential for accurate hand tracking.

# Implementation

pip install opencv-python numpy mediapipe pyautogui

import cv2

import numpy as np

import math

import mediapipe as mp

import pyautogui

import time

# Initialize volume control parameters

volume = 0

max\_volume = 100

min\_volume = 0

vol\_range = max\_volume - min\_volume

# Initialize hand tracking

mp\_drawing = mp.solutions.drawing\_utils

mp\_hands = mp.solutions.hands

hands = mp\_hands.Hands()

# Initialize webcam

cap = cv2.VideoCapture(0)

# Set screen width and height

screen\_width, screen\_height = pyautogui.size()

# Define hand landmarks for volume control

thumb\_tip\_id = 4

index\_finger\_tip\_id = 8

# Define gesture thresholds

gesture\_distance\_threshold = 50

# Initialize gesture variables

previous\_gesture = None

gesture\_start\_time = time.time()

while True:

# Read frame from webcam

ret, frame = cap.read()

if not ret:

break

# Flip the frame horizontally for a mirrored view

frame = cv2.flip(frame, 1)

# Convert the BGR image to RGB

image\_rgb = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)

# Process the image and detect hands

results = hands.process(image\_rgb)

# Draw hand landmarks on the frame

if results.multi\_hand\_landmarks:

for hand\_landmarks in results.multi\_hand\_landmarks:

mp\_drawing.draw\_landmarks(frame, hand\_landmarks, mp\_hands.HAND\_CONNECTIONS)

# Get thumb and index finger coordinates

thumb\_tip = hand\_landmarks.landmark[thumb\_tip\_id]

index\_finger\_tip = hand\_landmarks.landmark[index\_finger\_tip\_id]

# Convert thumb and index finger coordinates to screen coordinates

thumb\_x, thumb\_y = int(thumb\_tip.x \* screen\_width), int(thumb\_tip.y \* screen\_height)

index\_x, index\_y = int(index\_finger\_tip.x \* screen\_width), int(index\_finger\_tip.y \* screen\_height)

# Calculate distance between thumb and index finger

distance = math.sqrt((index\_x - thumb\_x) \*\* 2 + (index\_y - thumb\_y) \*\* 2)

# Map the distance to the volume range

volume = np.interp(distance, [0, screen\_width], [min\_volume, max\_volume])

volume = int(volume)

# Determine the gesture based on the distance

if distance < gesture\_distance\_threshold:

current\_gesture = 'CLOSED'

else:

current\_gesture = 'OPEN'

# Perform action based on the gesture

if current\_gesture != previous\_gesture:

if current\_gesture == 'CLOSED':

gesture\_start\_time = time.time()

elif current\_gesture == 'OPEN':

gesture\_duration = time.time() - gesture\_start\_time

if gesture\_duration < 1.0: # If the gesture duration is less than 1 second

pyautogui.press('playpause')

else: # If the gesture duration is more than 1 second

pyautogui.press('space')

previous\_gesture = current\_gesture

# Show the frame

cv2.imshow('Hand Gesture media Control', frame)

# Exit if 'q' is pressed

if cv2.waitKey(1) & 0xFF == ord('q'):

break

# Release the webcam and destroy windows

cap.release()

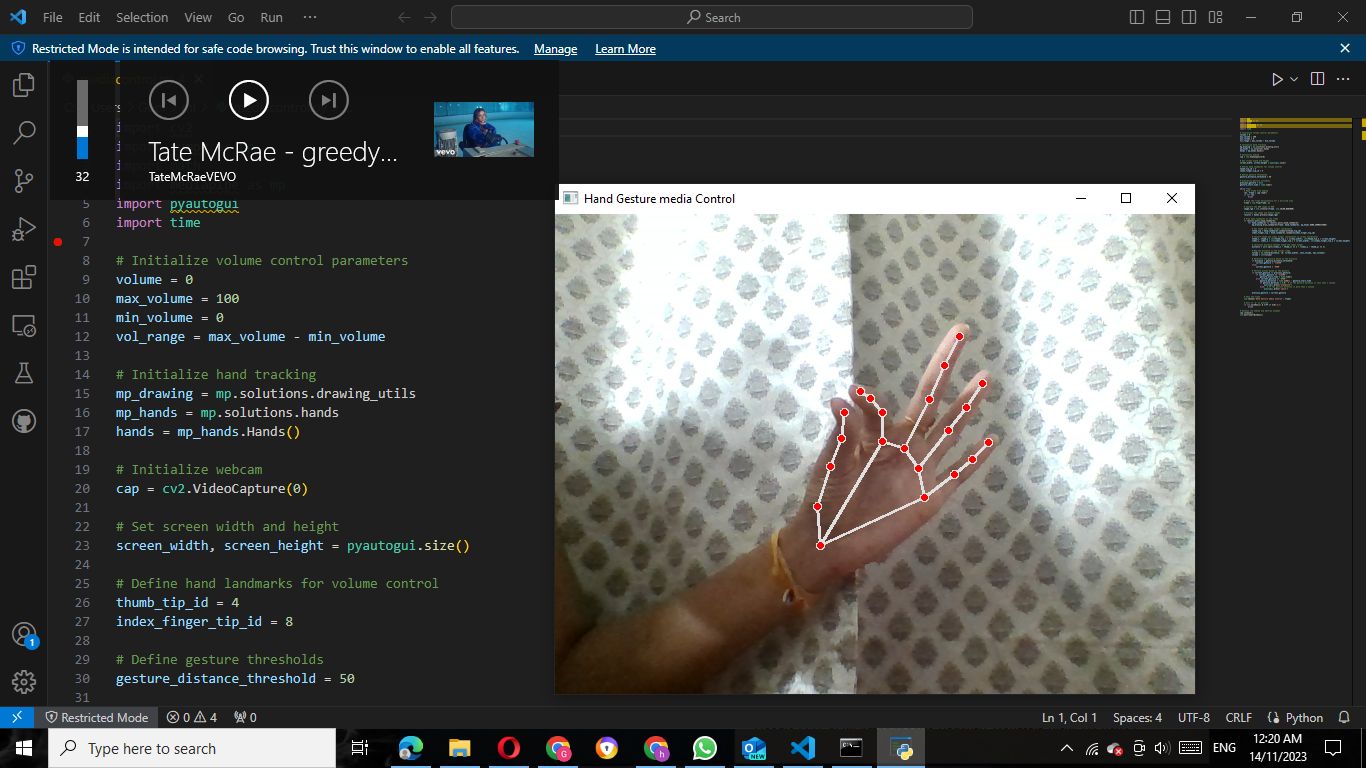
cv2.destroyAllWindows()

# 

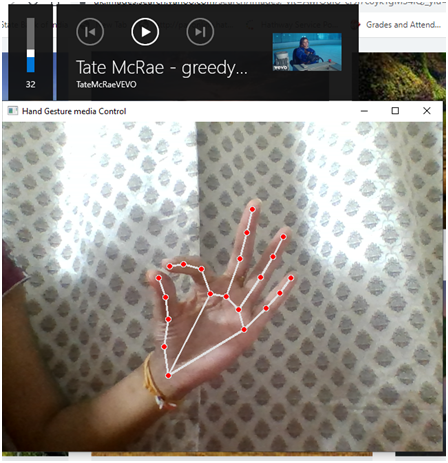
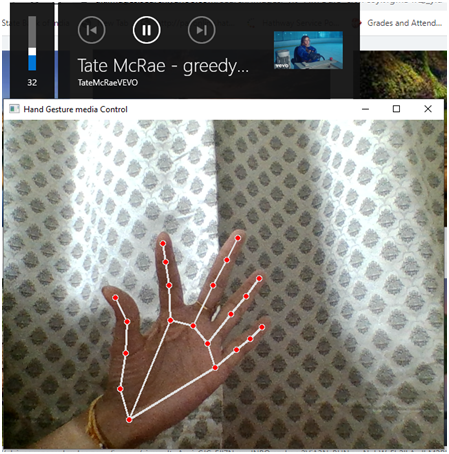
# 

# Results

*Overview of this project*



*When a media is paused When a media is played*



**Conclusion**

In the realm of human-computer interaction, this project has successfully introduced a hand gesture recognition system tailored for media control. Leveraging computer vision techniques through OpenCV, Mediapipe, and PyAutoGUI libraries, the system provides users with an intuitive and expressive means of interfacing with computing environments.

The implemented algorithm facilitates real-time and accurate tracking of hand landmarks, allowing users to control media playback seamlessly through defined hand gestures. The integration of PyAutoGUI translates these gestures into tangible actions, enhancing the user's ability to interact with digital media effortlessly.

The project not only achieves its immediate goal of creating a responsive media control system but also opens doors to broader applications of gestural interfaces. By exploring the potential of hand gestures in domains such as virtual reality, augmented reality, and robotics, the project contributes to the ongoing evolution of natural and accessible human-computer interaction.

## 

## 

# References

1. D. M. S. Reddy, S. Kukkamudi, R. Kunda and T. Mamatha, "Virtual Mouse Using Hand Gesture," 2022 International Conference on Knowledge Engineering and Communication Systems (ICKES), Chickballapur, India, 2022, pp. 1-5, doi: 10.1109/ICKECS56523.2022.10060367.

2. G. Bhole, D. Bhingare, R. Bhise, S. Bhegade, S. Bhokare and A. Bhosle, "System Control using Hand Gesture," 2023 International Conference for Advancement in Technology (ICONAT), Goa, India, 2023, pp. 1-4, doi: 10.1109/ICONAT57137.2023.10080493.

3. A. Baheti, D. Patwa and S. Gajjar, "Gesture Based Computer Control System," 2023 9th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India, 2023, pp. 220-225, doi: 10.1109/ICACCS57279.2023.10113075.

4. H. Dong, A. Danesh, N. Figueroa and A. E. Saddik, "An Elicitation Study on Gesture Preferences and Memorability Toward a Practical Hand-Gesture Vocabulary for Smart Televisions," in IEEE Access, vol. 3, pp. 543-555, 2015, doi: 10.1109/ACCESS.2015.2432679.