THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

Submitted by :
Karanvir Singh (102303435)
Sahil Popli (102303438)
Deepak (102303459)
Bibek Thagunna (102367009)
Siddartha Nepal (102367010)

# REAL TIME COMPUTER VISION ELC ACTIVITY

## Title: Real-time object detection and drawing effects

### Hand Gesture Recognition Project Report

### 1. Introduction

In this project, we are going to determine the gesture of the hand in real-time using a webcam. First, the hand is marked with different landmarks, and a hand gesture is determined by the positions of those landmarks. With different hand gestures, we can perform different actions.

Imagine effortlessly pausing your favorite movie with a simple hand raise, adjusting volume through finger movements, or navigating lecture slides without reaching for your mouse during those long online classes. This technology eliminates the constant need to fumble for controls, creating a more immersive and comfortable viewing experience whether you're binge-watching series, attending virtual meetings, or following educational content from your couch.

The system works by capturing hand movements, isolating them from the background, analyzing finger positions, and translating these natural gestures into meaningful actions—making digital interaction as intuitive as pointing or waving in real life.

## 2. Technologies Used

We used the following tech stacks for our CV project:

- **Python:** Primary programming language.
- **OpenCV:** To access the camera feed for capturing video and processing images in real-time.
- **MediaPipe:** To detect and track hand landmarks in real-time.
- **PyAutoGUI:** To enable programmatic mouse and keyboard control, to press keys for media controls like adjusting volume and seeking.
- **win32api:** To access the Windows API, used to send system-level media key events like play and pause.

## 3. Objectives

The objectives for this project are listed below:

- **Real-time Hand Detection:** Create a reliable system to locate and track hands from a live webcam feed in real-time.
- **Gesture Recognition:** Develop a method for the system to recognize hand gestures accurately based on the detected hand landmarks.
- **Media Command Mapping:** Link recognized gestures to media control actions.
- **System Integration:** Utilize Windows APIs and automation libraries to send media control commands to the system.
- **User Interface:** Improve the user experience by providing visual feedback for hand tracking and active commands.

## 4. Methodology

The project's main function involves video input, image processing, hand analysis, gesture classification, and command execution.

**It works in the following way:**

1. **Camera Input:** The program captures a live video feed from the webcam.
2. **Hand Tracking:** In each frame, MediaPipe tracks the 21 landmarks of the user's hand.
3. **Gesture Recognition:** The program analyzes the landmark positions to see which fingers are extended and identify the corresponding gesture.

4. **Action Execution:** The system triggers a media control action based on the identified gesture.

### 4.1. Video Capture and Preprocessing

Initialize webcam with OpenCV and process every video frame in the following way:

- Flip the frame horizontally to create a mirror image for better interaction.
- Convert the color space from BGR to RGB (the required format for MediaPipe).

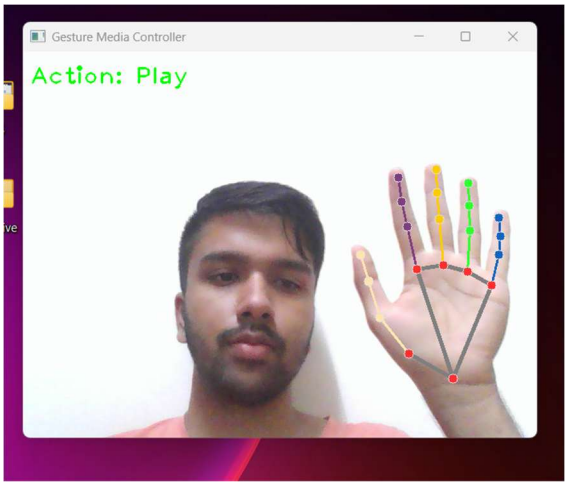
### 4.2. Hand Landmark Detection with MediaPipe

The MediaPipe Hands solution (mp.solutions.hands) assists in hand detection and estimating landmarks. The model processes the RGB frames to detect hands and returns 21 accurate 3D coordinates (landmarks) for each hand which are key landmarks for estimating the hand's pose. The landmarks and their connections are rendered onto the live video feed for user feedback.

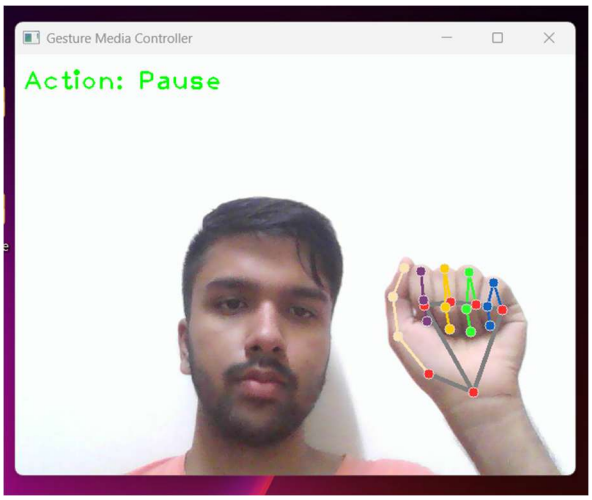
### 4.3. Gesture Recognition Logic

A custom detect_gesture function uses the list of landmark coordinates to determine the hand pose.

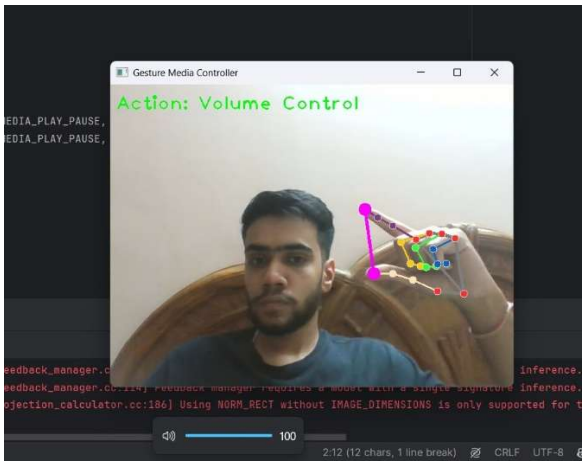
- **Finger Extension Detection:** For each finger, its state (extended or bent) is determined by comparing the Y-coordinate of its tip to a lower joint. Due to its different structure, the thumb's state is determined using the X-coordinate.
- **Recognized Gestures and Actions:**
    - **Open Palm:** Plays media.



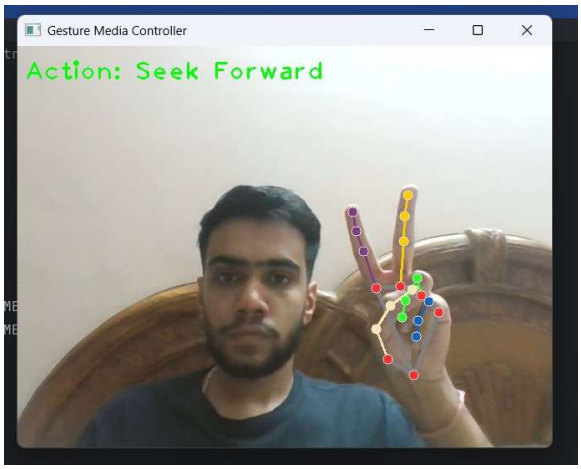
- **Closed Fist:** Pauses media.



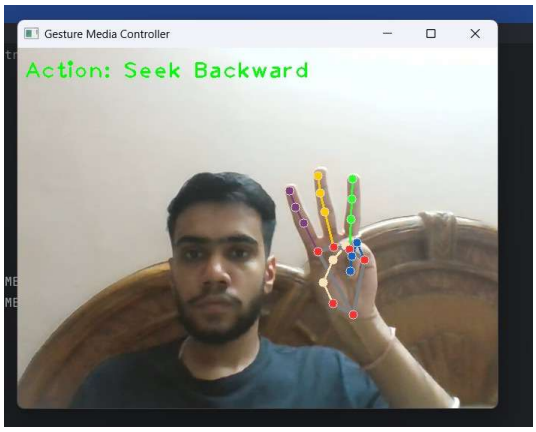

- **Pinch (Index Finger and Thumb):** Adjusts the volume.



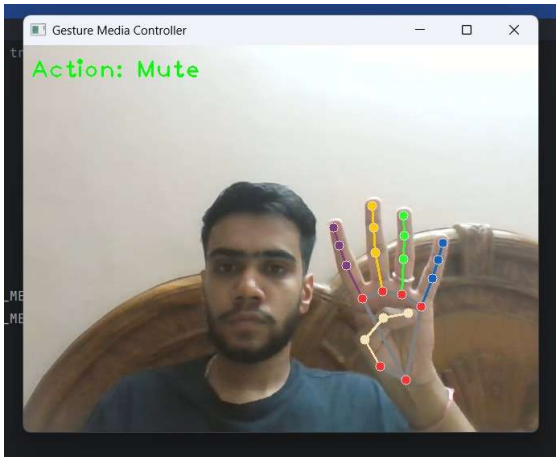

- **Peace Sign (Index and Middle Finger):** Seeks forward in the media.



- **Three Fingers (Index, Middle, Ring):** Seeks backward in the media.




- **Four Fingers (Index, Middle, Ring, Pinky):** Mutes or unmutes the volume.




- **Cooldown Mechanism:** A 1-second cooldown prevents the same gesture from being executed multiple times which provides smoother control.

#### 4.4. Media Command Execution

Once we recognize a gesture, the associated command is activated:

- **Play/Pause:** win32api.keybd_event mimics a VK_MEDIA_PLAY_PAUSE key press.
- **Seek Forward/Backward:** pyautogui.press simulates the 'right' and 'left' arrow key presses.
- **Volume Control:** The Euclidean distance between the index fingertip and thumb tip is measured. Depending on a threshold,

pyautogui.press("volumeup") or pyautogui.press("volumedown") is performed.
- **Mute:** pyautogui.press("volumemute") toggles the system mute on or off.

#### 4.5. User Feedback and Control

The on-screen display shows the camera feed with hand landmarks and indicates the last performed action (e.g. "Play," "Pause," "Volume Control"). Press ‘q’ to exit the live webcam.

## 5. Results

The Gesture Media Controller meets its primary functions that are:

- **Successful Hand Tracking:** Hand is successfully tracked using MediaPipe.
- **Robust Gesture Interpretation:** Hand gestures are translated into various media control commands.
- **Responsive Control:** We experienced minimal latency while executing media control commands.
- **Practical Utility:** The project offers a functional and enjoyable alternative to traditional input devices eliminating the need to fumble for control.

## 6. Conclusion

This project successfully develops a gesture-based media control system, showcasing the significant possibilities of computer vision in creating user-friendly interfaces. The application enhances accessibility and convenience by providing an interactive experience by enabling users to control media with basic hand gestures. It eliminates the need to press the keys on keyboard repeatedly for media controls.