A

Course End Project Report on

Real-Time Hand and Face Detection with Finger Gesture Recognition using Mediapipe and Haar Cascades

*Submitted in the Partial Fulfillment of the Requirements*

*for the Award of the Degree of*

## BACHELOR OF TECHNOLOGY

in

**Computer Science and Engineering(DS)**

Submitted by

**K.Mahammad Basha 232P1A3216**

Under the esteemed guidance of

**Dr.P.Pavankumar**



**Department of Computer Science and Engineering (DS)**

**CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY**

**(Approved by AICTE, New Delhi & Affiliated to JNTUA, Ananthapuramu)**

**(Accredited by NAAC with “A” Grade and Accredited by NBA (CE, EEE, ECE, CSE))**

**(Recognized by UGC under section 2(f) and 12(b) of UGC Act, 1956)**

**VIDYA NAGAR, PALLAVOLU (V), PRODDATUR-516360, Y.S.R. (Dt.), A.P**

**2025-26**

 **CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY**

**(Approved by AICTE, New Delhi & Affiliated to JNTUA, Ananthapuramu)**

**(Accredited by NAAC with “A” Grade and Accredited by NBA (CE, EEE, ECE, CSE))**

**(Recognized by UGC under section 2(f) and 12(b) of UGC Act, 1956)**

**VIDYA NAGAR, PALLAVOLU (V), PRODDATUR-516360, Y.S.R. (Dt.), A.P**

**Department of Computer Science and Engineering (DS)**

# CERTIFICATE

This is to certify that the project titled **Real-Time Hand and Face Detection with Finger Gesture Recognition using Mediapipe and Haar Cascades** is carried out by

**K.Mahammad Basha 232P1A3216**

in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering (DS)** during the year 2024-25.

**Signature of the Supervisor Signature of the HOD**

**Dr. P Pavankumar Mr.G.Sreenivasa Reddy**

**Professor HOD, CSE (DS)**

# 

# ACKNOWLEDGMENT

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

I wish to express my deep sense of gratitude to **Dr. P Pavan Kumar**, **Professor** and Project Supervisor, Department of Computer Science and Engineering, Chaitanya Bharathi Institute of Technology, for his able guidance and useful suggestions, which helped me in completing the project in time.

I am particularly thankful to **Mr.G.sreenivasa Reddy**, Head of the Department, Department of Computer Science and Engineering (AI), her guidance, intense support and encouragement, which helped us to mould my project into a successful one.

I show gratitude to my honorable Principal **Dr. S. Sruthi** and Director Admin **Dr. G. Sreenivasula Reddy** for providing all facilities and support.

I avail this opportunity to express my deep sense of gratitude and heart-full thanks to **Sree V. Jaya Chandra Reddy**, Chairman and **Sree V. Lohit Reddy**, CEOof CBIT, for providing a congenial atmosphere to complete this project successfully.

I also thank all the staff members of Computer Science and Engineering (AI)department for their valuable support and generous advice. Finally thanks to all my friends and family members for their continuous support and enthusiastic help.

**K.Mahammad Basha**

# ABSTRACT

Real-time hand and face detection with finger gesture recognition is a crucial aspect of human-computer

interaction, enabling intuitive and touch-free control of devices. This system combines Mediapipe for accurate hand landmark detection and gesture recognition with Haar cascades for efficient face detection. The integration allows simultaneous tracking of multiple hands and faces in live video streams, making it suitable for applications like virtual interfaces, gaming, and assistive technologies. Finger gestures are recognized by analyzing hand landmarks, enabling predefined actions or commands to be executed in real time. The system also incorporates a data storage mechanism to save detected faces and hand gestures for further processing or training purposes. By leveraging the efficiency of Haar cascades and the precision of Mediapipe, this approach provides a robust and low-latency solution for gesture-based control. This methodology not only improves interactive experiences but also demonstrates the practical implementation of computer vision and machine learning techniques in real-time scenarios.

**Keywords:** Real-Time Detection, Mediapipe, Haar Cascades, Finger Gesture Recognition, Computer Vision

# TABLE OF CONTENTS

**Title Page No.**

[ACKNOWLEDGEMENTS](#_30j0zll) 1

[ABSTRACT](#_3znysh7) 3

TABLE OF CONTENTS………………………………………………………………………….………………4

CHAPTER 1: INTRODUCTION …………………………………………………………….……………… 5-7

1.1 Background ………………………………………………………………………

1.2 Objective of the Project ……………………………………….……………

1.3 Significance of the Project ……………………………..…………………

CHAPTER 2: LITERATURE SURVEY …………………………………………………..……..… 8-14

2.1 Software REQUIREMENTS .…….………………………..

2.2 Hardware REQUIREMENTS …………………………...………………

CHAPTER 3: IMPLEMENTATION …………………………………………………………………...…. 15-16

3.1 Source Code …………………………………………………………………..…17-21

CHAPTER 4: RESULTS …………………………………….…………………………………….….……..… 22-23

4.1 Outputs .…………….......…………………………………………….…………

CHAPTER 5: CONCLUSION AND FUTURE SCOPE …………………………...….………… 24-25

5.1 Conclusion …………………………………………………………..….…….…

5.2 Future Enhancements..……………………………….……………….……

REFERENCES ……………………………………………………..…………..…… 26

# 

**CHAPTER 1**

**INTRODUCTION**

**1.1Background**

Real-time hand and face detection with finger gesture recognition is an important area of computer vision and human-computer interaction. It involves detecting human faces and tracking hand movements from live video streams, and recognizing specific finger gestures to perform predefined actions. This technology leverages **Mediapipe**, a robust framework for precise hand landmark detection and gesture recognition, alongside **Haar cascades**, which provide fast and efficient face detection.

The integration of these techniques allows the system to detect multiple hands and faces simultaneously, making it highly suitable for interactive applications such as virtual reality, gaming, sign language interpretation, and assistive technologies for people with disabilities. By analyzing hand landmarks, finger gestures can be recognized accurately, enabling users to control digital devices intuitively and without physical contact.

The combination of real-time detection, machine learning algorithms, and gesture recognition ensures both high accuracy and low latency, which are crucial for responsive interactions. Such systems demonstrate the practical implementation of AI and computer vision in everyday technology, improving accessibility, user experience, and the overall efficiency of human-computer interfaces.

**1.2** **Objective of the Project**

The main objective of this project is to develop a real-time system for detecting faces and hands while recognizing finger gestures using computer vision techniques. The system aims to enable intuitive, touch-free interaction with digital devices by accurately identifying hand movements and facial positions in live video streams. By integrating **Mediapipe** for hand landmark detection and gesture recognition with **Haar cascades** for efficient face detection, the project seeks to provide a fast, accurate, and responsive solution for applications such as virtual interfaces, gaming, sign language interpretation, and assistive technologies. The ultimate goal is to enhance human-computer interaction, improve accessibility, and demonstrate the practical implementation of real-time computer vision in everyday technology.

**1.3 Significance of the Project**

Real-time hand and face detection with finger gesture recognition holds significant importance in the field of human-computer interaction and computer vision. This system enables users to interact with digital devices intuitively and without physical contact, providing a more natural and efficient interface. By integrating **Mediapipe** for accurate hand landmark detection and gesture recognition with **Haar cascades** for fast face detection, the project allows for precise tracking of multiple hands and faces in live video streams. Such a system is valuable in applications like virtual reality, gaming, sign language interpretation, and assistive technologies for individuals with disabilities. The ability to recognize gestures in real time enhances accessibility, improves user experience, and demonstrates the practical implementation of AI and computer vision techniques in everyday technology.

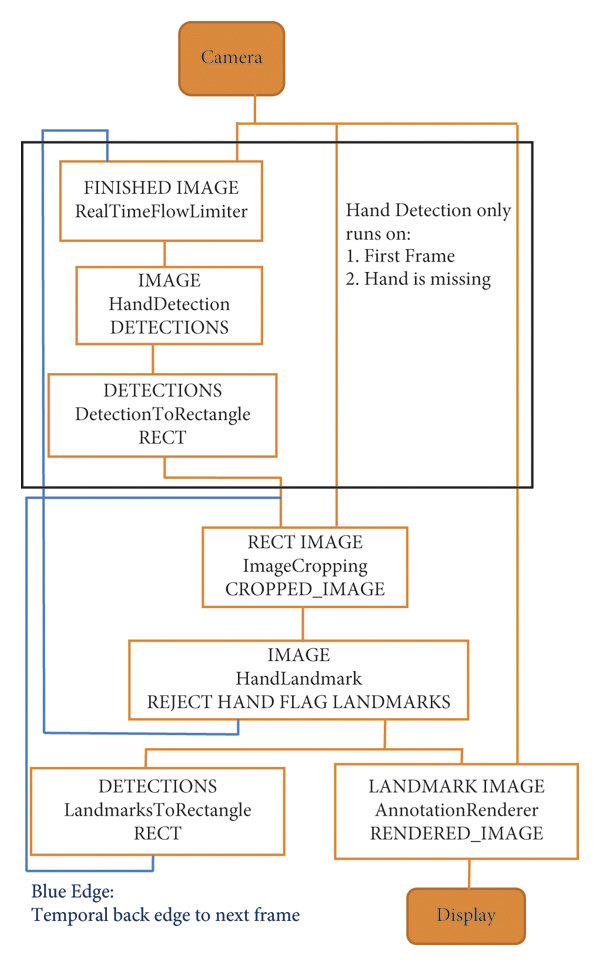


FIGURE :1.1 PROCESS OF FACE DETECTION

**CHAPTER 2**

**LITERATURE SURVEY**

The recognition and tracking of hand gestures are essential elements in human-computer interaction systems, providing intuitive control and facilitating interaction with a wide range of devices and applications. We have developed a hand gesture detection system using Python and JavaScript. Our system uses MediaPipe, an open-source framework for machine learning-based perception tasks, to accurately identify and classify hand gestures. MediaPipe handles hand tracking, interpreting the hand curl and movement of hand gestures with high precision. We've built the frontend interface using HTML, CSS, and JavaScript. By combining these technologies, we've created a user-friendly interface where you can interact with the system through hand gestures captured via a webcam. Additionally, we've added our custom machine-learning methods to detect and recognize various hand gestures.Our system has numerous potential applications, including hand-sign translation and gesture-based control for smart devices. By offering an efficient and accessible solution, our project advances human-computer interaction paradigms, enabling more natural and intuitive interactions between users and machines**.** [1]

Gesture recognition technology has emerged as a transformative solution for natural and intuitive human–computer interaction (HCI), offering touch-free operation across diverse fields such as healthcare, gaming, and smart home systems. In mobile contexts, where hygiene, convenience, and the ability to operate under resource constraints are critical, hand gesture recognition provides a compelling alternative to traditional touch-based interfaces. However, implementing effective gesture recognition in real-world mobile settings involves challenges such as limited computational power, varying environmental conditions, and the requirement for robust offline–online data management. In this study, we introduce *ThumbsUp*, which is a gesture-driven system, and employ a partially systematic literature review approach (inspired by core PRISMA guidelines) to identify the key research gaps in mobile gesture recognition. By incorporating insights from deep learning–based methods (e.g., CNNs and Transformers) while focusing on low resource consumption, we leverage Google’s MediaPipe in our framework for real-time detection of 21 hand landmarks and adaptive lighting pre-processing, enabling accurate recognition of a “thumbs-up” gesture. The system features a secure queue-based offline–cloud synchronization model, which ensures that the captured images and metadata (encrypted with AES-GCM) remain consistent and accessible even with intermittent connectivity. Experimental results under dynamic lighting, distance variations, and partially cluttered environments confirm the system’s superior low-light performance and decreased resource consumption compared to baseline camera applications. Additionally, we highlight the feasibility of extending ThumbsUp to incorporate AI-driven enhancements for abrupt lighting changes and, in the future, electromyographic (EMG) signals for users with motor impairments. Our comprehensive evaluation demonstrates that *ThumbsUp* maintains robust performance on typical mobile hardware, showing resilience to unstable network conditions and minimal reliance on high-end GPUs. These findings offer new perspectives for deploying gesture-based interfaces in the broader IoT ecosystem, thus paving the way toward secure, efficient, and inclusive mobile HCI solutions. [2]

Hand sign recognition is an important technology in human-computer interaction because it allows people to communicate with machines in a natural and simple manner. This paper offers a revolutionary real-time hand sign identification method using MediaPipe landmark methods and OpenCV. This system uses a deep learning and computer vision combination to accurately understand and classify a large range of hand gestures and signs. This technique can be applied in many different contexts, particularly in the domains of immersive technologies (Virtual and Augmented Reality), and sign language interpretation. This project is based on MediaPipe, a library that enables for the real-time monitoring of hand landmarks in video streams. These landmarks stand in for important hand locations including the palm, knuckles, and fingertips. From the landmark data, significant features are extracted and used as input to a machine learning model to perform the classification task. The outcomes validate the system’s precision and show that it can operate in real-time, which qualifies it for interactive applications where hand sign recognition is crucial. There are plenty of applications using hand gesture recognition like photo snapping, video games and application control systems. This method makes an important contribution to the fields of computer vision and human-computer interaction by providing a realistic and efficient solution for hand sign recognition. Ten gestures have been considered and we were able to achieve promising results for almost all of the gestures[3].

Computers and people interact in a number of ways, and these interactions are made possible by the interface between the two. To make interactions between people and computers as simple and efficient several methods has been proposed. The vision-based systems enabling facial and hand gesture identification are the most promising ones for contactless human-computer interaction (HCI). This project proposed a human-computer interaction (HCI) system by combining a face and gesture recognition system. Face recognition based authentication system is more popular than any other biometric features like fingerprint and eye iris recognition. The face recognition system performs identifying and verifying a person in front of the camera. If the image matches with the photo in the database, the face is labelled with the person's name and a successful login is done within system. If it does not match, the image displays unknown and an unauthorized access is detected. Here face detection and recognition is implemented using HAAR-cascade classifiers and LBPH recognizers. The hand gesture recognition system recognises the gesture and links it with a variety of actions, including launching programmes like windows media player, MS word, PowerPoint, Notepad, screenshot capturing, opening Google and some of the mouse control actions. Here hand tracking performs with the help of Media Pipe library provided by Google which is open-source and it is a well-trained model to achieve high performance. The Media Pipe library can be used to analyse hand gestures using a variety of technologies. Media Pipe hands library will employ two models. 1) A palm detector model that generates a bounding box of hand and 2) A hand landmark model that predicts the hand skeleton. Gestures can be used by users to effortlessly communicate with computers that have RGB cameras. Key Word : Human-Computer Interaction, Face and hand gesture recognition, HAAR-cascade classifiers, LBPH recognizers, Media Pipe library.[4]

In human-computer interaction, gender classification and hand gesture recognition are essential technologies. This paper presents a real-time system that integrates gender prediction and face and hand gesture detection, utilising the Mini Xception model and MediaPipe framework. We address the requirements for efficient real-time operation by utilizing frame-skipping mechanisms in our work. The system comprises video capture with OpenCV, model integration using TensorFlow, and efficient data handling with NumPy. Extensive testing has proven its generalization to diverse settings and excellent performance, with 84% accuracy for both tasks, capable of real-time processing[5].

This major project centers around the critical domain of Action Detection for Sign Language Gestures, addressing the unique communication needs of individuals with hearing impairments. The project involves creating a custom dataset that has been carefully selected to capture the minute details of various hand motions, facial expressions, and body language used in sign language. The system consists of four steps, including segmentation, feature extraction, classification, and verification. The hand region is separated from the background using the background subtraction method, and the palm and fingers are identified to recognize hand gestures. The deep learning model is trained using information such as hand motion size, shape, and color to recognize four different gestures, including the fist, palm, two-finger, and three-finger movements. To evaluate the effectiveness of the system, we compare its performance to other state-of-the-art gesture detection algorithms, and its accuracy and speed are measured. The results show that our proposed system outperforms other algorithms, achieving high recognition accuracy and fast processing speed. The LSTM NN algorithm is a good fit for this project, as it can learn and extract features from data. It has been effectively used for a variety of applications in machine learning, including speech recognition and language modeling. Therefore, our proposed solution can contribute significantly to the fields of artificial intelligence and communication. The proposed smart system that utilizes the LSTM algorithm for hand gesture recognition can facilitate communication with individuals who use sign language as their primary form of speech. By achieving high recognition accuracy and fast processing speed, this system can be applied in various fields and help reduce the communication gap between sign language users and non-users[6].

This work harnesses the capabilities of OpenCV and MediaPipe libraries providing a comprehensive suite of hand gesture interactions using GUIs. The "Volume Control" feature dynamically adjusts system volume in response to hand gestures, utilizing landmarks detected by the MediaPipe hand tracking model. Meanwhile, the "Virtual Mouse" functionality transforms hand movements into cursor control actions, offering left-click capabilities for intuitive navigation. Another noteworthy feature is "Brightness Control", where screen brightness adapts based on the proximity of the hand to the camera that leverages MediaPipe's hand tracking capabilities to gauge distance and adjust screen brightness accordingly.[7]

The technology of identifying movements in real-time video is called “motion recognition”. These actions are classified according to the properties they represent. Creating awareness of movements is a difficult task because it overcomes two major challenges. The first challenge was to enable control of movement, allowing users to effectively interact with computers or other devices using only one hand. This technology has many applications, especially in human-computer interaction and linguistic tasks. Simple techniques such as hand classification and measurement using Haar cascade classifiers in Python and OpenCV can be used to generate gesture recognition. This article focuses on gesture recognition as a qualitative analysis. This setup includes a camera that captures the user's movements, which are then processed by the system. The main purpose of gesture recognition is to develop awareness and use human movements to control devices and communicate. Live gesture recognition allows users to work with the front camera on their computers, providing greater interaction and understanding with the technology. We will create an orientation with the help of the OpenCV module which can control the system with gestures without using a keyboard or mouse[8].

The hand gesture recognition project aims to develop a real-time system capable of detecting and recognizing hand gestures from a video stream captured by a webcam. The system utilizes the MediaPipe and OpenCV libraries for hand tracking and gesture recognition. The project involves capturing video frames from the webcam, preprocessing the frames, detecting hand landmarks using MediaPipe, and recognizing specific gestures based on the detected landmarks. Once a gesture is recognized, the system displays the corresponding text overlay on the video frame using OpenCV. The project is designed to provide a user-friendly interface for interpreting hand gestures, enabling applications such as gesture-based control systems, sign language translation, and interactive user interfaces. OpenCV is a widely used open-source computer vision and machine learning software library. It provides a wide range of functionalities for image and video processing, MediaPipe is an open-source framework for building multimodal (e.g., video, audio) applied ML pipelines. It provides ready-to-use ML solutions for various tasks.[9]

The computer mouse is one of the incredible inventions of Human-Computer Interaction. Wireless or Bluetooth mice we use currently are not free devices as they require batteries and dongles to plug into the Computer. Since computer vision is at its pinnacle and is used in many different aspects of day-to-day life, such as Face Recognition, Automatic car, and Color detection, we here are using it, to create an AI mouse by using hand tip detection and hand gestures. We also add face recognition using the Eigen face algorithm to revamp its security. The algorithm will first confirm the user’s authenticity by scanning their face once confirmed then one can access his computer through hand gestures, one can perform click and scroll the mouse without using the hardware mouse. The algorithm uses Eigenface and deep learning for the detection of hands.[10]

**2.1 Software Requirements**

To implement a real-time hand and face detection system with finger gesture recognition, the necessary software requirements and specifications typically include: a Python programming environment with libraries for computer vision and machine learning, such as **OpenCV** for image processing, **Mediapipe** for hand landmark detection and gesture recognition, and Haar cascade classifiers for face detection. Additional tools may include **NumPy** for numerical operations, **Matplotlib** for visualization, and IDEs like **PyCharm** or **VS Code** for development. The system requires access to live video input through a webcam or camera module, and robust data handling capabilities to process image frames in real time. Key functionalities include detecting and tracking multiple faces and hands, recognizing predefined finger gestures, and executing corresponding actions efficiently with minimal latency.

* 1. **Hardware Requirements**

1. **Computer/Processor:** A system with at least a quad-core processor (Intel i5/i7 or AMD equivalent) to handle real-time video processing.
2. **RAM:** Minimum 8 GB RAM for smooth performance; 16 GB or more is recommended for handling multiple video streams.
3. **Graphics Processing Unit (GPU):** A dedicated GPU (NVIDIA or AMD) is recommended to accelerate hand and face detection algorithms, especially when using Mediapipe for gesture recognition.
4. **Camera/Webcam:** A high-definition webcam (720p or 1080p) for capturing live video streams.
5. **Storage:** At least 50 GB of free storage to store libraries, project files, and any captured images or video data.
6. **Operating System:** Windows 10/11, Linux, or macOS compatible with Python and relevant libraries.

**6.Programming Languages:**

* **OpenCV:** For image and video processing, face detection using Haar cascades, and handling real-time video streams.
* **Mediapipe:** For accurate hand landmark detection and finger gesture recognition.
* **NumPy:** For efficient numerical operations and matrix handling in image processing.
* **Matplotlib / Seaborn:** For visualization of hand and face detection results.
* **TensorFlow / PyTorch (optional):** For advanced machine learning or gesture classification models if required.

.

**7.GIS Software (Optional):**

While the main implementation relies on Python and its libraries, additional tools can be used to

eanhance system performance and visualization

* OpenCV: For real-time image and video processing, face detection, and frame manipulation.
* Mediapipe: For accurate hand landmark detection and finger gesture recognition.
* Matplotlib / Seaborn: For visualizing detection results and gesture analysis.
* IDE Tools (PyCharm, VS Code): For development and debugging**.**

**8.Machine Learning Algorithms:**

Various algorithms can be employed depending on the complexity of gesture recognition:

* **Rule-based classifiers:** For simple predefined gestures based on hand landmarks.
* **Support Vector Machines (SVM) or Random Forests:** For classifying complex hand gestures.
* **Neural Networks / Deep Learning Models:** For advanced gesture recognition and multi-hand tracking.

**9.Weather Data:**

The system requires real-time video input data from a webcam or camera module, capturing:

* **Face Data:** For detecting and tracking faces in live video streams.
* **Hand and Finger Landmarks:** For gesture recognition and tracking multiple hands simultaneously.
* **Frame Sequence Data:** To analyze movement patterns and recognize dynamic gestures accurately.

**CHAPTER 3**

**IMPLEMENTATION**

**3.1 GUI Layout for Real-Time Hand and Face Detection:**

A graphical user interface (GUI) for real-time hand and face detection with finger gesture recognition provides a visual platform to interact with the system and display detection results dynamically. The GUI is designed to show live video streams from the webcam, highlight detected faces and hand landmarks, and indicate recognized finger gestures in real time.

**Key aspects of the GUI layout for this system:**

* **Live Video Display:** Streaming the camera feed while overlaying detected faces with bounding boxes and hand landmarks with skeleton points.
* **Gesture Indicators:** Displaying recognized gestures as text or icons on the interface, providing immediate feedback to the user.
* **Multiple Detection Panels:** Allowing tracking of multiple hands and faces simultaneously with separate sections for each detected entity.
* **Interactive Controls:** Buttons or sliders to start/stop detection, adjust detection sensitivity, or switch between gesture recognition modes.
* **User Feedback and Logs:** Displaying detected gestures, face coordinates, and tracking status to help users understand the system’s actions in real time.

This GUI layout ensures an intuitive, interactive, and responsive experience for users, enabling seamless monitoring and control of the gesture recognition system.

* **Camera Input:** Option to select the active webcam or camera module for live video streaming.
* **Detection Mode:** Dropdown menu to choose between face detection, hand detection, or combined face and hand detection.
* **Gesture Recognition Settings:** Input fields or sliders to adjust sensitivity, confidence threshold, or select specific gestures to recognize.
* **Display Options:** Checkboxes to toggle visualization of hand landmarks, face bounding boxes, or gesture labels on the video feed.
* **Recording & Logging:** Buttons to start/stop recording of live video, save detected frames, and log gesture data for analysis.
* **User Feedback Panel:** Real-time display of recognized gestures, number of hands/faces detected, and tracking status.
* **Calibration Settings:** Input fields or sliders for adjusting frame resolution, detection speed, or landmark accuracy to optimize performanc

**Source Code:**

import cv2

import mediapipe as mp

import os

# ===== Create Save Directories =====

save\_faces = "saved\_faces"

save\_left\_hands = "saved\_left\_hands"

save\_right\_hands = "saved\_right\_hands"

save\_background = "saved\_background"

for folder in [save\_faces, save\_left\_hands, save\_right\_hands, save\_background]:

if not os.path.exists(folder):

os.makedirs(folder)

# ===== Initialize =====

mp\_hands = mp.solutions.hands

mp\_drawing = mp.solutions.drawing\_utils

hands = mp\_hands.Hands(max\_num\_hands=2, min\_detection\_confidence=0.7)

face\_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade\_frontalface\_default.xml")

cap = cv2.VideoCapture(0)

# ===== Counters =====

face\_count = 0

left\_hand\_count = 0

right\_hand\_count = 0

background\_count = 0

print("✅ Live detection started. Press 'q' to quit.")

finger\_names = ["Thumb", "Index", "Middle", "Ring", "Pinky"]

finger\_tips = [4, 8, 12, 16, 20] # Mediapipe landmark indices for fingertips

# ===== Helper: Check if finger is up =====

def fingers\_up(hand\_landmarks):

fingers\_status = []

# Thumb

fingers\_status.append(

hand\_landmarks.landmark[4].x < hand\_landmarks.landmark[3].x

if hand\_landmarks.landmark[17].x < hand\_landmarks.landmark[5].x

else hand\_landmarks.landmark[4].x > hand\_landmarks.landmark[3].x

)

# Other fingers

for tip\_id in [8, 12, 16, 20]:

fingers\_status.append(

hand\_landmarks.landmark[tip\_id].y < hand\_landmarks.landmark[tip\_id - 2].y

)

return fingers\_status

while cap.isOpened():

ret, frame = cap.read()

if not ret:

break

rgb\_frame = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)

# ===== Detect Hands =====

results = hands.process(rgb\_frame)

detected\_hands = []

if results.multi\_hand\_landmarks:

for hand\_landmarks, hand\_handedness in zip(

results.multi\_hand\_landmarks, results.multi\_handedness

):

hand\_label = hand\_handedness.classification[0].label # Left or Right

detected\_hands.append(hand\_label)

# Draw landmarks on live frame only (for display)

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

# Get fingers up

fingers\_status = fingers\_up(hand\_landmarks)

# Show Finger Names only when raised

h, w, \_ = frame.shape

for tip\_index, name, is\_up in zip(finger\_tips, finger\_names, fingers\_status):

if is\_up:

cx = int(hand\_landmarks.landmark[tip\_index].x \* w)

cy = int(hand\_landmarks.landmark[tip\_index].y \* h)

cv2.putText(

frame,

f"{name}",

(cx + 5, cy - 5),

cv2.FONT\_HERSHEY\_SIMPLEX,

0.5,

(0, 255, 255),

1,

)

# Determine gesture

if all(fingers\_status):

gesture = "Open Hand"

elif not any(fingers\_status):

gesture = "Closed Hand"

else:

gesture = "Partial"

cv2.putText(

frame,

f"{hand\_label}: {gesture}",

(10, 30 + 20 \* detected\_hands.index(hand\_label)),

cv2.FONT\_HERSHEY\_SIMPLEX,

0.6,

(0, 255, 0),

2,

)

# === Crop hand for saving WITHOUT landmarks ===

x\_min, y\_min, x\_max, y\_max = w, h, 0, 0

for lm in hand\_landmarks.landmark:

cx, cy = int(lm.x \* w), int(lm.y \* h)

x\_min, y\_min = min(x\_min, cx), min(y\_min, cy)

x\_max, y\_max = max(x\_max, cx), max(y\_max, cy)

clean\_frame = cv2.cvtColor(rgb\_frame, cv2.COLOR\_RGB2BGR)

hand\_crop = clean\_frame[y\_min:y\_max, x\_min:x\_max].copy()

if hand\_crop.size != 0:

if hand\_label == "Left":

left\_hand\_count += 1

cv2.imwrite(

os.path.join(

save\_left\_hands, f"left\_hand\_{left\_hand\_count}.jpg"

),

hand\_crop,

)

else:

right\_hand\_count += 1

cv2.imwrite(

os.path.join(

save\_right\_hands, f"right\_hand\_{right\_hand\_count}.jpg"

),

hand\_crop,

)

# ===== Detect Faces =====

gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

faces = face\_cascade.detectMultiScale(gray, 1.3, 5)

if len(faces) > 0:

for (x, y, w\_f, h\_f) in faces:

# Draw rectangle on live frame

cv2.rectangle(frame, (x, y), (x + w\_f, y + h\_f), (0, 255, 0), 2)

# Crop face for saving WITHOUT rectangle

clean\_frame = cv2.cvtColor(rgb\_frame, cv2.COLOR\_RGB2BGR)

face\_crop = clean\_frame[y:y + h\_f, x:x + w\_f].copy()

if face\_crop.size != 0:

face\_count += 1

cv2.imwrite(

os.path.join(save\_faces, f"face\_{face\_count}.jpg"), face\_crop

)

# ===== Save full background frame =====

if len(faces) == 0 and len(detected\_hands) == 0:

background\_count += 1

cv2.imwrite(

os.path.join(save\_background, f"background\_{background\_count}.jpg"),

frame.copy(),

)

# ===== STATUS LOGIC =====

num\_hands = len(detected\_hands)

face\_detected = len(faces) > 0

if num\_hands == 0 and not face\_detected:

status = "Background"

elif num\_hands == 1 and not face\_detected:

status = f"1 Hand ({detected\_hands[0]})"

elif num\_hands == 2 and not face\_detected:

status = "2 Hands"

elif num\_hands == 0 and face\_detected:

status = "Face"

elif num\_hands == 1 and face\_detected:

status = f"1 Hand ({detected\_hands[0]}) + Face"

elif num\_hands == 2 and face\_detected:

status = "2 Hands + Face"

else:

status = "Unknown"

# ===== Show Status + Counters on live frame =====

cv2.putText(

frame,

f"Status: {status}",

(20, 180),

cv2.FONT\_HERSHEY\_SIMPLEX,

0.8,

(0, 0, 255),

2,

)

cv2.putText(

frame,

f"Faces Saved: {face\_count}",

(20, 210),

cv2.FONT\_HERSHEY\_SIMPLEX,

0.7,

(0, 255, 0),

2,

)

cv2.putText(

frame,

f"Left Hands Saved: {left\_hand\_count}",

(20, 240),

cv2.FONT\_HERSHEY\_SIMPLEX,

0.7,

(255, 0, 0),

2,

)

cv2.putText(

frame,

f"Right Hands Saved: {right\_hand\_count}",

(20, 270),

cv2.FONT\_HERSHEY\_SIMPLEX,

0.7,

(255, 0, 255),

2,

)

cv2.putText(

frame,

f"Backgrounds Saved: {background\_count}",

(20, 300),

cv2.FONT\_HERSHEY\_SIMPLEX,

0.7,

(0, 255, 255),

2,

)

cv2.imshow("Live Detection", frame)

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

break

cap.release()

cv2.destroyAllWindows()

print("✅ Detection ended.")

**CHAPTER 4**

**RESULTS**

**[Live Video Feed]**

**┌─────────────────────────┐**

**│ │**

**│ [Face Box] │**

**│ ┌─────┐ │**

**│ │ :) │ │ <- Face detected**

**│ │**

**│ Hand Landmarks │ <- Lines and circles on hand joints**

**│ Gesture: Thumbs Up │ <- Recognized gesture label**

**│ │**

**└─────────────────────────┘**

**[Status Panel]**

**Faces Detected: 1**

**Hands Detected: 2**

**Gesture: Peace Sign**

FIGURE 4.1:process of how it works code

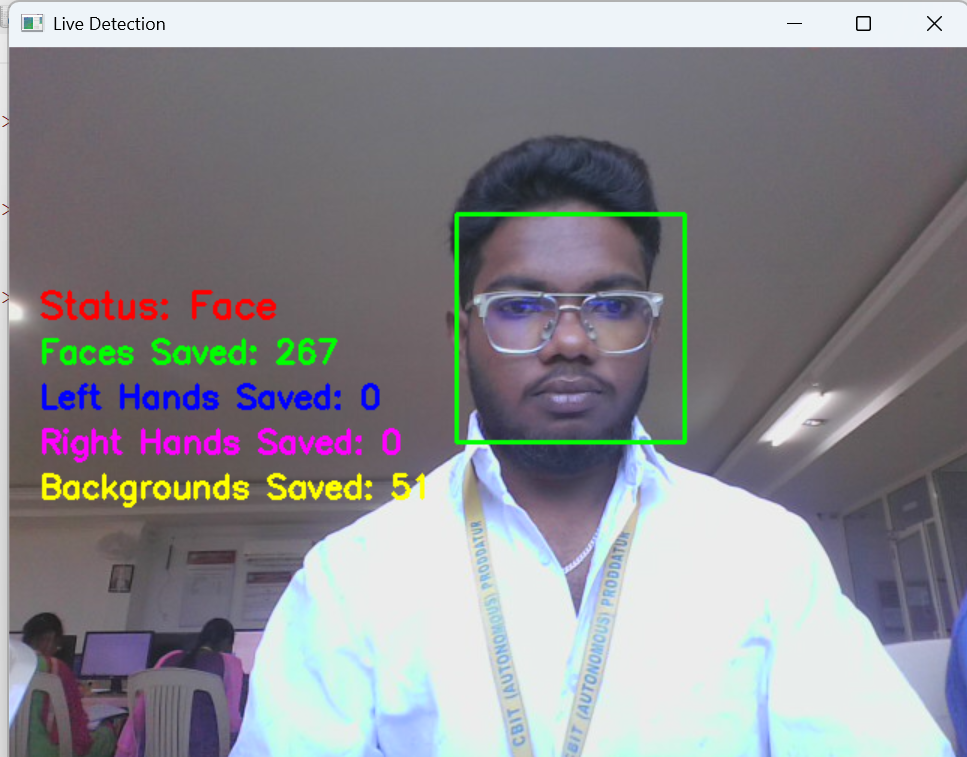


FIGURE:4.2 Status face

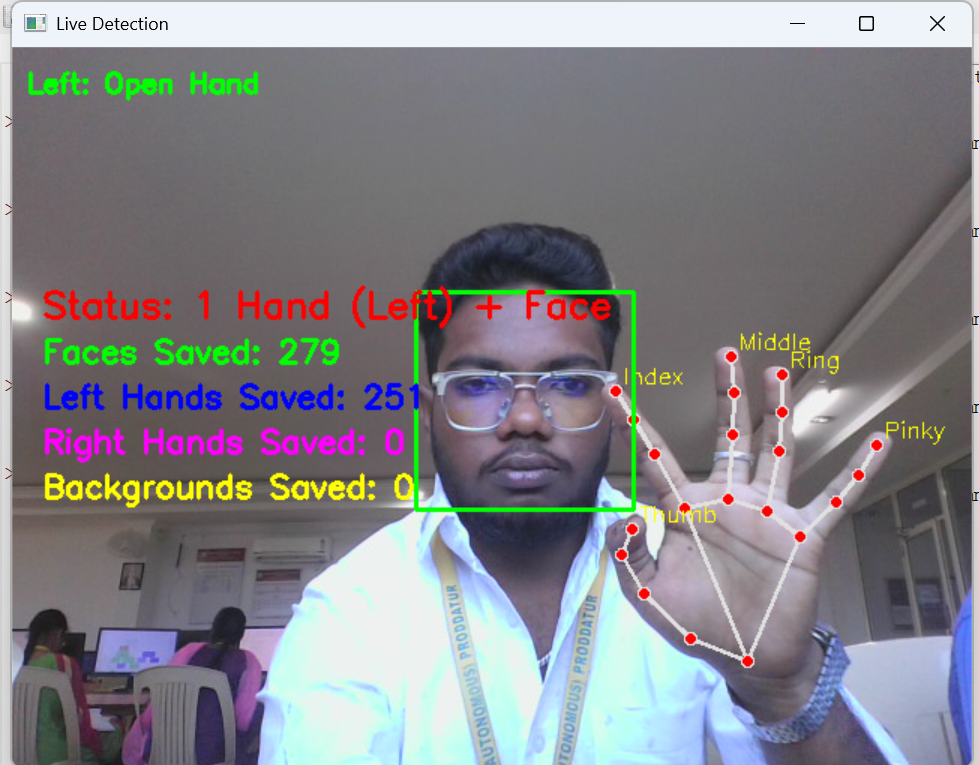


FIGURE 4.3: status face +hand

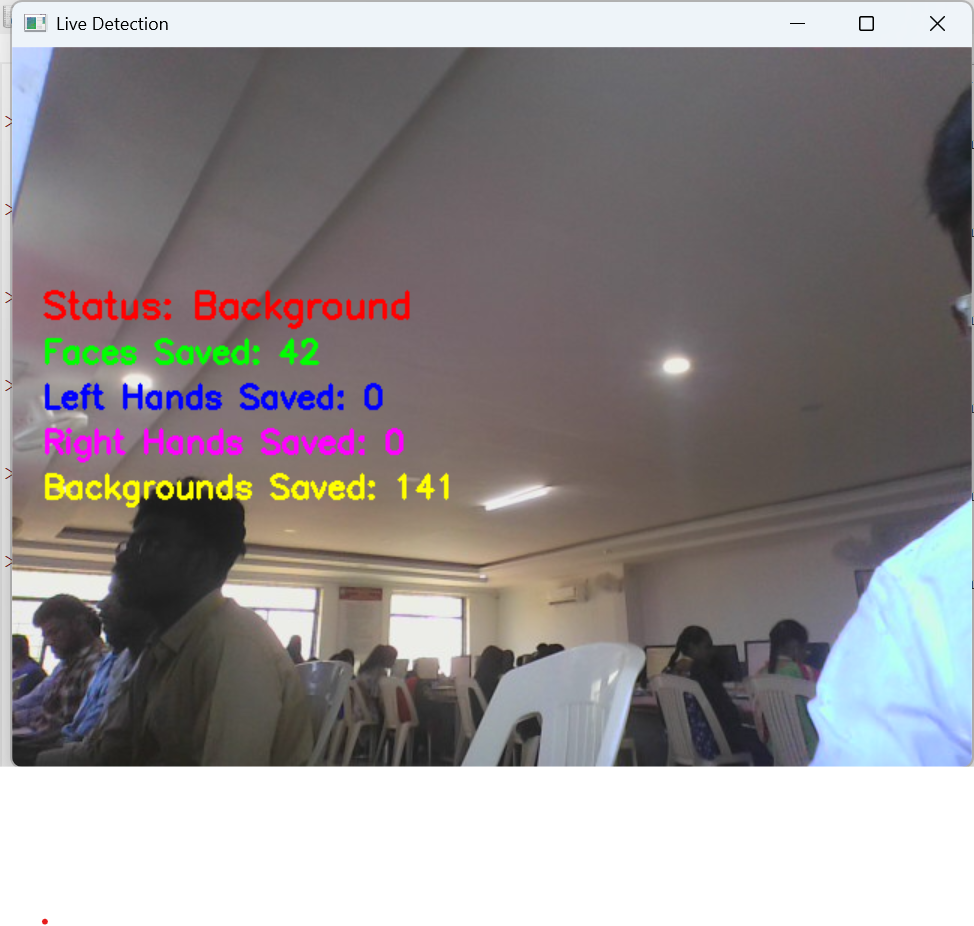
****

FIGURE 4.4:status background

**CHAPTER 5**

**CONCLUSION AND FUTURE SCOPE**

**5.1 Conclusion**

In conclusion, real-time hand and face detection with finger gesture recognition provides a powerful tool for enhancing human-computer interaction by enabling intuitive, touch-free control of digital devices. By combining **Mediapipe** for precise hand landmark detection with **Haar cascades** for efficient face detection, the system can track multiple hands and faces simultaneously and recognize predefined gestures accurately in live video streams. This technology improves accessibility, supports applications in gaming, virtual interfaces, and assistive technologies, and demonstrates the practical implementation of computer vision and machine learning techniques in real-time scenarios. Continuous advancements in detection algorithms, gesture recognition models, and real-time processing will further improve accuracy, responsiveness, and usability, empowering users with seamless and interactive experiences.

**Key Points:**

* Accurate real-time detection of faces and hands.
* Reliable recognition of multiple finger gestures.
* Enhanced human-computer interaction without physical contact.
* Applications in gaming, virtual interfaces, and assistive technologies.
* Potential for further improvement with advanced models and optimization.

**Improved decision making:**

By accurately detecting faces, hands, and gestures, the system can enable more intuitive and reliable human-computer interaction, allowing applications to respond precisely to user commands.

**Resource optimization:**

Optimizing computational resources and leveraging GPU acceleration can enhance real-time performance, ensuring low-latency detection even with multiple hands or faces in the frame.

**Climate change adaptation:**

Incorporating deep learning models such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs) can enable recognition of more complex or dynamic gestures beyond simple predefined ones.

**Data-driven approach:**

Utilizing large datasets of hand gestures, facial expressions, and video sequences can improve model accuracy, robustness, and adaptability to different lighting conditions and backgrounds.

**Future directions:**

* **Enhanced Multi-Person Tracking:** Support for multiple users simultaneously with accurate gesture recognition for each.
* **Integration with IoT and AR/VR:** Using gesture detection in virtual/augmented reality environments or smart home/IoT applications.
* **Region-Specific Models:** Tailoring gesture recognition models to different cultural or regional gesture variations.
* **Explainable AI:** Implementing methods to visualize and understand why a gesture was recognized, improving system transparency.

**Key Areas for Improvement:**

* **Advanced Data Integration:** Using large video datasets for training to handle diverse hand shapes, skin tones, and lighting conditions.
* **Real-Time Sensor Fusion:** Combining depth sensors or additional cameras to improve detection accuracy and gesture understanding in 3D space.

**5.2 Future Enhancements**

Future enhancements in real-time hand and face detection with finger gesture recognition are likely to focus on improving accuracy, responsiveness, and usability by integrating advanced algorithms, additional sensors, and optimized hardware. Key areas for improvement include:

Advanced data integration:

**High-resolution satellite imagery:**

Utilizing high-resolution camera feeds to capture detailed hand and face movements, ensuring precise detection of landmarks and gestures even in complex environment.

**Real-time sensor networks:**

Incorporating additional sensors such as depth cameras or motion trackers to enhance gesture recognition accuracy, track hand movements in 3D space, and improve responsiveness in dynamic scenarios**.**

**REFERENCES**

[1].Patel, Meenu, et al. "Real-time Hand Gesture Recognition Using Python and Web Application." *2024 1st International Conference on Advances in Computing, Communication and Networking (ICAC2N)*. IEEE, 2024.

[2].Marques, Pedro, et al. "Real-Time Gesture-Based Hand Landmark Detection for Optimized Mobile Photo Capture and Synchronization." *Electronics* 14.4 (2025): 704.

[3].Kavitha, M. N., et al. "A Real-Time Hand-Gesture Recognition Using Deep Learning Techniques." *International Conference on Artificial Intelligence and Smart Energy*. Cham: Springer Nature Switzerland, 2024.

[4].Niya, K. S., and Anu Augustin. "Face and Gesture Based Human-Computer Interaction System." (2023).

[5].Kulkarni, Manas Girish, et al. "Real-Time Gender Classification Using MiniXception and Hand Gesture Detection Using MediaPipe Framework." *2025 International Conference on Computational, Communication and Information Technology (ICCCIT)*. IEEE, 2025.

[6].Sharma, Amit, Vedanta Koul, and Aman Sharma. "Action Detection for Sign Language Gestures." (2024).

[7].Dhamodaran, Sasikala, et al. "Implementation of Hand Gesture Recognition using OpenCV."

[8].Tamilkodi, R., et al. "Hand Gesture Recognition and Volume Control." *International Conference on Computational Innovations and Emerging Trends (ICCIET-2024)*. Atlantis Press, 2024.

[9].Shaikh, Ms Tamanna, et al. "IMPLEMENTATION OF: REAL-TIME HAND GESTURE RECOGNITION SYSTEM FOR DEAF MUTE FRIENDLY BANKING USING MEDIAPIPE AND OPENCV."

[10].Kumar, Akshay, et al. "AI based mouse using Face Recognition and Hand Gesture Recognition." *2023 International Conference on Artificial Intelligence and Applications (ICAIA) Alliance Technology Conference (ATCON-1)*. IEEE, 2023.