Dynamic hand gesture detector using python and open CV

Lakshya Gaur[1], Kapil Tomar[2], Dr. Naveen Tyagi[3]
[1]M.Tech Scholar, Department of Computer Science and Engineering, MIT, Bulandshahr
[2]Assistant Professor, Department of Computer Science and Engineering, MIT, Bulandshahr
[3]Professor, Department of Computer Science and Engineering, MIT, Bulandshahr

Abstract- Hand Gesture Recognition and Image Overlay Using OpenCV and MediaPipe
This research paper presents a method for real-time hand gesture recognition and image overlay using OpenCV and MediaPipe. The system captures live video feed from a webcam, detects hand gestures, and overlays corresponding images based on the detected gestures. The implementation leverages the capabilities of MediaPipe for hand tracking and OpenCV for image processing and display. The proposed method is efficient and runs in real-time, providing immediate feedback on detected gestures. This paper discusses the system model, the underlying algorithm, and the results obtained from the implementation.

1. INTRODUCTION

Hand gesture recognition is a crucial technology in the field of human-computer interaction. It allows users to interact with digital devices using natural hand movements, enhancing the user experience. This project focuses on creating a hand gesture recognition system that identifies specific gestures and overlays images accordingly. The system uses Python, OpenCV, and MediaPipe, leveraging their powerful image processing and machine learning capabilities.
The journey towards gesture-based interaction represents a departure from the conventional notion of computer interfaces as passive tools controlled solely through manual manipulation. Instead, it seeks to imbue technology with a deeper understanding of human intention and expression, enabling seamless interaction that mirrors the fluidity of human communication. At its core, gesture-based interaction draws inspiration from the rich tapestry of human movement, encompassing gestures, postures, facial expressions, and even subtle nuances of body language. By decoding and interpreting these non-verbal cues, computers can discern user intent and respond in a manner that feels more natural and intuitive.
The evolution of gesture-based interfaces has been propelled by advancements in a myriad of technologies, including computer vision, machine learning, sensor technology, and augmented reality.

The main objectives of this project are:
- To develop a real-time hand gesture recognition system.
- To overlay images based on detected gestures.
- To provide a seamless and responsive user experience.

2. SYSTEM MODEL

2.1 The system comprises the following components:
- Webcam: Captures live video feed.
- OpenCV: Handles video capture, image processing, and display.
- MediaPipe: Detects and tracks hand landmarks.
- Gesture Recognition: Identifies specific hand gestures.
- Image Overlay: Overlays corresponding images based on detected gestures.

2.2 The flow of the system is as follows:
1. Capture live video feed using OpenCV.
2. Process each frame to detect hand landmarks using MediaPipe.
3. Identify gestures based on the positions of the landmarks.
4. Overlay images on the video feed based on detected gestures.
5. Display the processed video feed in real-time.

3. ALGORITHM

3.1 Initialization

- Import necessary libraries.
- Initialize video capture using OpenCV.
- Load images for overlay.
- Initialize MediaPipe for hand detection.

3.2 Frame Processing

For each frame captured from the webcam:

1. Flip the frame horizontally for a mirror effect.
2. Convert the frame to RGB format.
3. Detect hand landmarks using MediaPipe.
4. Identify gestures by analyzing the positions of landmarks.
5. Overlay images based on detected gestures.
6. Display the processed frame.

3.3 Gesture Detection

The algorithm identifies gestures by analyzing the positions of finger tips and thumb:

- LIKE: Thumb up gesture.
- DISLIKE: Thumb down gesture.
- NEUTRAL: No specific gesture.
- DYBALA CELEBRATION: Specific finger arrangement.

4. CODE IMPLEMENTATION

4.1 Initialization

python

Copy code

```
import cv2 import numpy as np import mediapipe as
mp cap = cv2.VideoCapture(0) mpHands =
mp.solutions.hands hands = mpHands.Hands() draw =
mp.solutions.drawing_utils # Load images for overlay
like_img = cv2.imread('like.jpg') dislike_img =
cv2.imread('dislike.jpg') neutral_img =
cv2.imread('neutral.jpg') dybala_img =
cv2.imread('dybala.jpg') Finger_tips = [8, 12, 16, 20]
thumb_tip = 4
```

4.2 Frame Processing

python

Copy code

```
while True: success, img = cap.read() img =
cv2.flip(img, 1) h, w, c = img.shape rgb =
cv2.cvtColor(img, cv2.COLOR_BGR2RGB) results =
hands.process(rgb) if results.multi_hand_landmarks:
for hand_landmarks in results.multi_hand_landmarks:
lm_list = [] for lm in hand_landmarks.landmark:
lm_list.append(lm) finger_fold_status = [] for tip in
Finger_tips: x, y = int(lm_list[tip].x * w),
int(lm_list[tip].y * h) cv2.circle(img, (x, y), 13, (0,
255, 0), cv2.FILLED) if lm_list[tip].x < lm_list[tip -
3].x: cv2.circle(img, (x, y), 13, (0, 0, 255),
cv2.FILLED) finger_fold_status.append(True) else:
finger_fold_status.append(False) if
all(finger_fold_status): if lm_list[thumb_tip].y <
lm_list[thumb_tip - 1].y < lm_list[thumb_tip - 2].y: h,
w, c = like_img.shape img[0:h, 0:w] = like_img else:
h, w, c = dislike_img.shape img[0:h, 0:w] =
dislike_img if lm_list[Finger_tips[0]].x >
lm_list[Finger_tips[0] - 1].x: h, w, c =
neutral_img.shape img[0:h, 0:w] = neutral_img if
lm_list[Finger_tips[0]].x < lm_list[Finger_tips[0] -
3].x: if lm_list[thumb_tip].x < lm_list[thumb_tip -
2].x: h, w, c = dybala_img.shape img[0:h, 0:w] =
dybala_img draw.draw_landmarks(img,
hand_landmarks,
mpHands.HAND_CONNECTIONS)
cv2.imshow('Hand Gesture Recognition', img) if
cv2.waitKey(1) == ord('q'): break cap.release()
cv2.destroyAllWindows()
```

5. CONCLUSION

This project demonstrates an efficient and real-time hand gesture recognition system using OpenCV and MediaPipe. By leveraging these libraries, we achieved accurate hand tracking and gesture recognition, providing immediate visual feedback through image overlay. The system can be further enhanced with additional gestures and applications, making it a versatile tool for human-computer interaction.

REFERENCE

1. OpenCV: Open Source Computer Vision Library. Available at: https://opencv.org/
2. MediaPipe: Cross-platform Framework for Building Multimodal Applied ML Pipelines. Available at: https://mediapipe.dev/
3. Python: The Python Programming Language. Available at: https://www.python.org/

Images to Include

1. Development Environment:

- Screenshot of the code editor with the project open.
- Photo of the setup with the webcam.

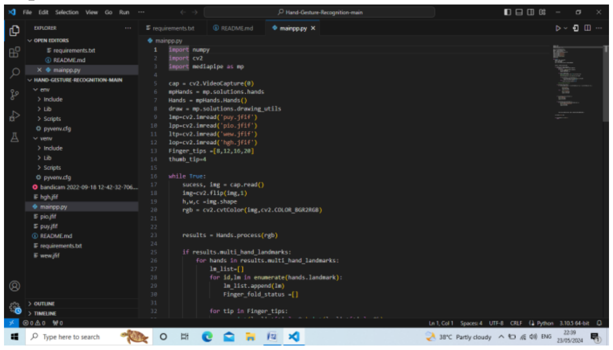
Hand Detection Process:

- Raw webcam feed before processing.
- Hand landmarks detection with landmarks highlighted.

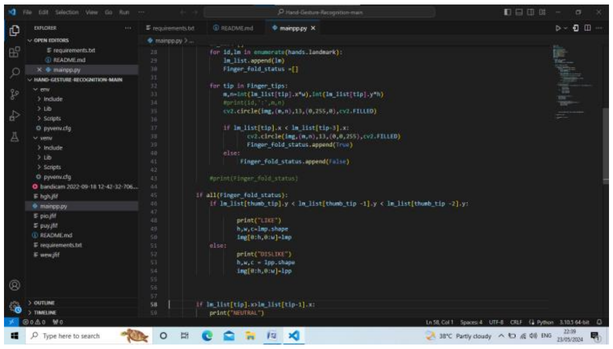
Gesture Recognition Output:

- Examples of each gesture ("LIKE", "DISLIKE", "NEUTRAL", "DYBALA CELEBRATION") with the overlay image.

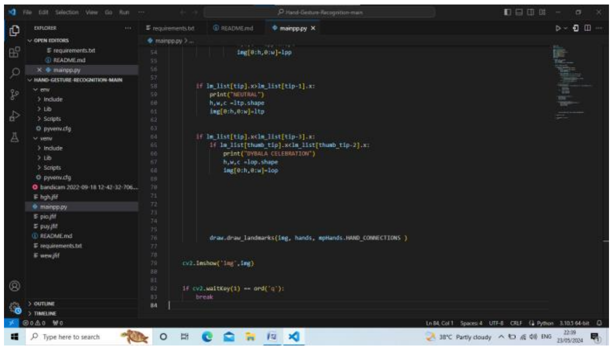
By incorporating these images and code snippets, this thesis will effectively illustrate the project's implementation and results.



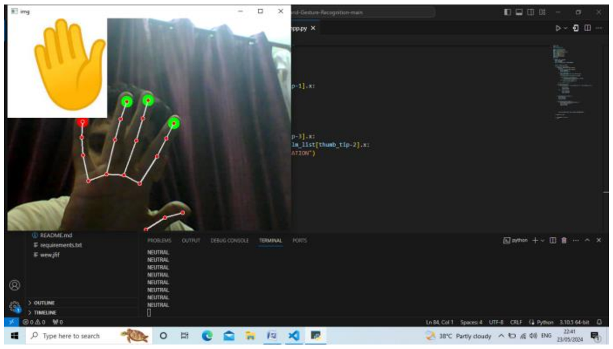
```python
import numpy
import cv2
import mediapipe as mp

cap = cv2.VideoCapture(0)
mpHands = mp.solutions.hands
Hands = mpHands.Hands()
draw = mp.solutions.drawing_utils
lmp=cv2.imread('puy.jfif')
lpp=cv2.imread('pio.jfif')
ltp=cv2.imread('wew.jfif')
lop=cv2.imread('hgh.jfif')
Finger_tips =[8,12,16,20]
thumb_tip=4

while True:
    sucess, img = cap.read()
    img=cv2.flip(img,1)
    h,w,c =img.shape
    rgb = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)


    results = Hands.process(rgb)

    if results.multi_hand_landmarks:
        for hands in results.multi_hand_landmarks:
            lm_list=[]
            for id,lm in enumerate(hands.landmark):
                lm_list.append(lm)
                Finger_fold_status =[]

            for tip in Finger_tips:
```



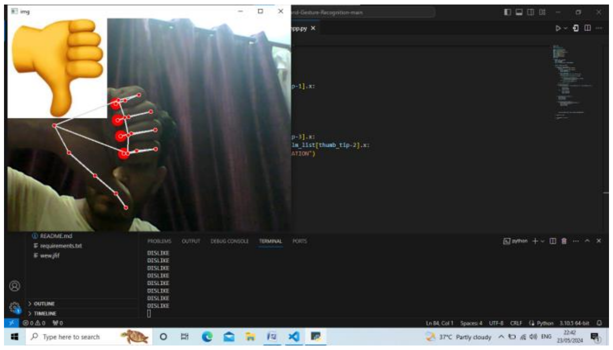
```python
            for id,lm in enumerate(hands.landmark):
                lm_list.append(lm)
                Finger_fold_status =[]

            for tip in Finger_tips:
                m,n=int(lm_list[tip].x*w),int(lm_list[tip].y*h)
                #print(id,':',m,n)
                cv2.circle(img,(m,n),13,(0,255,0),cv2.FILLED)

                if lm_list[tip].x < lm_list[tip-3].x:
                    cv2.circle(img,(m,n),13,(0,0,255),cv2.FILLED)
                    Finger_fold_status.append(True)
                else:
                  Finger_fold_status.append(False)

            #print(Finger_fold_status)

        if all(Finger_fold_status):
            if lm_list[thumb_tip].y < lm_list[thumb_tip -1].y < lm_list[thumb_tip -2].y:

                    print("LIKE")
                    h,w,c=lmp.shape
                    img[0:h,0:w]=lmp
            else:
                    print("DISLIKE")
                    h,w,c = lpp.shape
                    img[0:h,0:w]=lpp



        if lm_list[tip].x>lm_list[tip-1].x:
            print("NEUTRAL")
```



```python
                    img[0:h,0:w]=lpp



        if lm_list[tip].x>lm_list[tip-1].x:
            print("NEUTRAL")
            h,w,c =ltp.shape
            img[0:h,0:w]=ltp


        if lm_list[tip].x<lm_list[tip-3].x:
            if lm_list[thumb_tip].x<lm_list[thumb_tip-2].x:
                print("DYBALA CELEBRATION")
                h,w,c =lop.shape
                img[0:h,0:w]=lop







            draw.draw_landmarks(img, hands, mpHands.HAND_CONNECTIONS )


    cv2.imshow('img',img)


    if cv2.waitKey(1) == ord('q'):
        break
```



```
NEUTRAL
NEUTRAL
NEUTRAL
NEUTRAL
NEUTRAL
NEUTRAL
NEUTRAL
NEUTRAL
```



```
DISLIKE
DISLIKE
DISLIKE
DISLIKE
DISLIKE
DISLIKE
DISLIKE
DISLIKE
```