PBR VISVODAYA INSTITUTE OF TECHNOLOGY & SCIENCE

A
MINI PROJECT
ON
AI VIRTUAL MOUSE

AIM: To develop a mini project on AI VIRTUL MOUSE using Hand gesture.

DESCRIPTION:

INTRODUCTION:
AI Virtual mouse is a software that allows users to control their computer mouse without using a physical mouse by using hand gestures and computer vision techniques.

PROJECT BREAKDOWN:
Here's a breakdown of the main functionalities:

1. **Video Capture:**
    - The program starts by capturing video frames from a specified video source, typically a webcam. It utilizes OpenCV's **VideoCapture** function to access the video feed.
    - The captured frames are processed to detect hand gestures and control the mouse pointer.

2. **Hand Detection:**
    - Hand detection is performed using the MediaPipe library, which provides pre-trained models for detecting hand landmarks.
    - The **hand_detector** object from MediaPipe's **Hands** module is used to process the RGB frames captured from the video source.
    - The output of the hand detection process provides information about the detected hands, including landmarks such as fingertips, joints, and palm keypoints.

3. **Fingertip Extraction:**
    - After detecting hands in the video frames, the program isolates the index fingertip's position from the detected hand landmarks.
    - Each hand landmark contains coordinates relative to the frame's dimensions, which are then converted to screen coordinates to facilitate mouse pointer control.

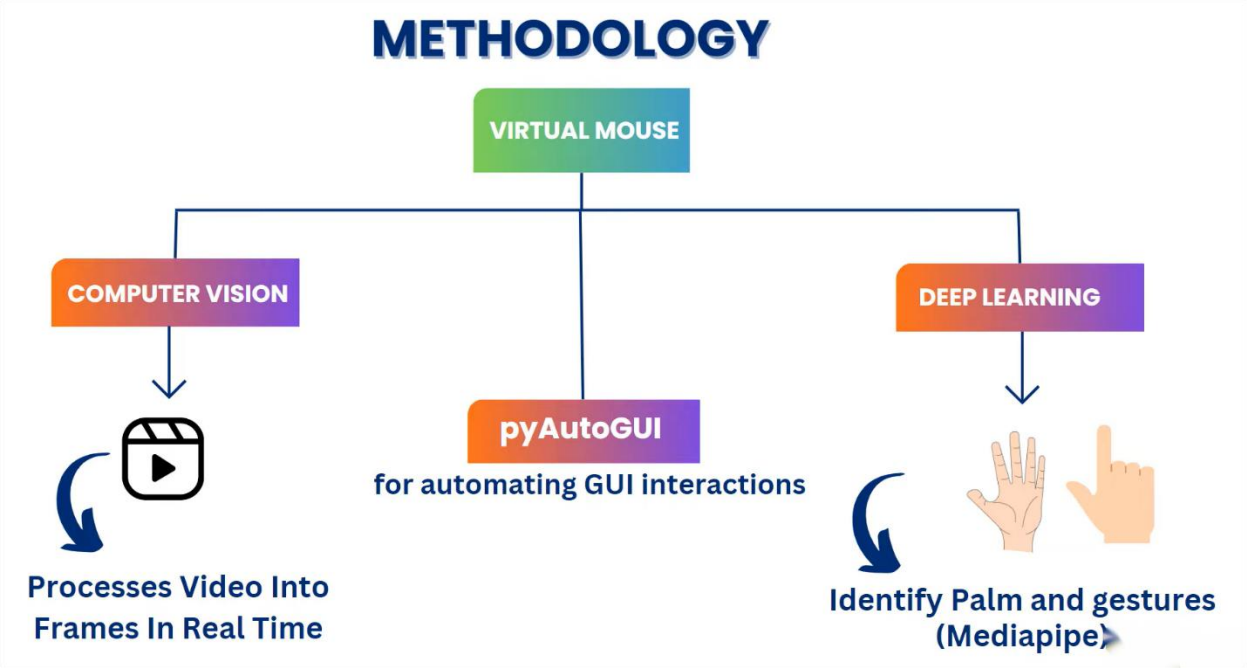
4. **Mouse Pointer Control:**
    - The position of the index fingertip is used to control the movement of the mouse pointer on the screen.
    - The program maps the fingertip position from the video frame's coordinate system to the screen's coordinate system, allowing for accurate mouse movement.
    - The **pyautogui.moveTo()** function is used to move the mouse pointer.

5. **Click Operation:**
    - A click operation is simulated when a specific gesture is detected, such as bringing the thumb close to the index finger.
    - The program monitors the distance between the thumb and index fingertip positions.
    - If the distance falls below a certain threshold, indicating a predefined gesture, the **pyautogui.click()** function is called to simulate a mouse click.
    - This enables users to interact with the computer using hand gestures, with the thumb and index finger gesture serving as a click action.

6. **Smooth Cursor Movement:**
    - To enhance user experience, the program applies an Exponential Moving Average (EMA) algorithm to smoothen cursor movements.
    - EMA calculates the weighted average of the current and previous fingertip positions, reducing sudden jumps in cursor movement.
    - Smoother cursor movement provides a more natural and responsive interaction experience for users.

7. **Real-time Feedback:**
    - The program continuously displays the video feed with overlaid hand landmarks and cursor movement.
    - This real-time feedback allows users to visualize their hand gestures and observe how they control the mouse pointer on the screen.
    - The **cv2.imshow()** function is used to display the video feed with the detected hand landmarks and cursor movement overlaid on the frames.

METHODOLOGY:



Here's a detailed methodology for implementing the provided code:

1. Setup and Dependencies:
    - Install the necessary dependencies: OpenCV (cv2), MediaPipe (mediapipe), and PyAutoGUI (pyautogui).
    - Ensure that the webcam or video source is connected and accessible by the system.

2. Video Capture:
    - Initialize the video capture using cv2.VideoCapture(0) to access the default webcam (or adjust the index for other video sources).
    - Continuously read frames from the video source using cap.read() within a loop.

3. Hand Detection with MediaPipe:
    - Import the MediaPipe library and instantiate a Hands object for hand detection.

- Convert the captured frames from BGR to RGB format using cv2.cvtColor().
- Process each frame with the hand detector using hand_detector.process().
- Extract hand landmarks from the processed output.

4. Fingertip Extraction and Mouse Control:
    - Iterate through each detected hand and its landmarks.
    - Identify the index fingertip and thumb positions from the landmarks.
    - Convert the fingertip positions from frame coordinates to screen coordinates using the screen resolution and frame dimensions.
    - Use pyautogui.moveTo() to move the mouse pointer to the index fingertip position.
    - Optionally, implement smoothing algorithms like Exponential Moving Average (EMA) to enhance cursor movement.

5. Gesture Recognition and Click Operation:
    - Monitor the distance between the thumb and index fingertip positions.
    - Define a threshold distance to recognize a specific gesture (e.g., thumb close to the index finger).
    - If the distance falls below the threshold, simulate a mouse click using pyautogui.click().

6. Real-time Feedback and Display:
    - Overlay hand landmarks and cursor movement on the video frames for real-time feedback.
    - Use drawing_utils.draw_landmarks() from MediaPipe to visualize hand landmarks on each frame.
    - Display the modified frames with overlaid information using cv2.imshow().

7. Loop Execution and User Interaction:
    - Continuously execute the loop to process each frame from the video source.
    - Allow users to interact with the virtual mouse by moving their hands and performing gestures.
    - Terminate the program gracefully when the user exits or closes the application window.

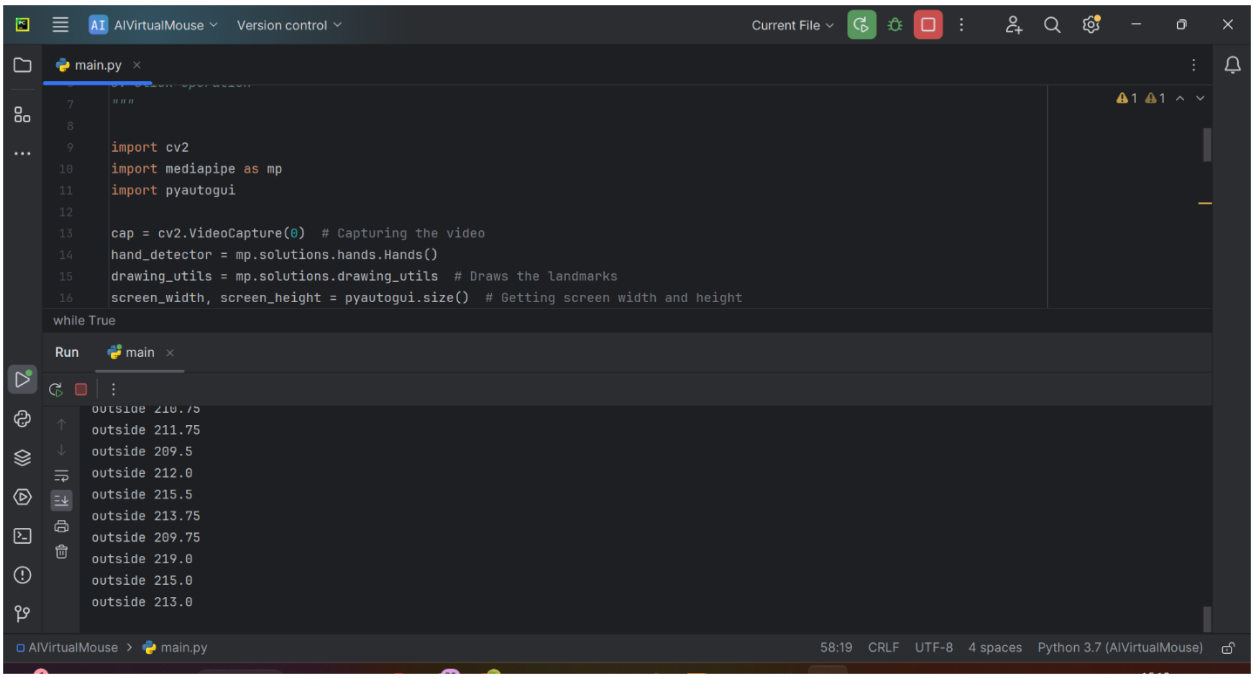
8. Adjustments and Optimization:
    - Fine-tune parameters such as thresholds, smoothing factors, and gesture recognition criteria based on user testing and feedback.
    - Optimize performance by adjusting frame processing techniques, such as frame resizing or skipping frames if necessary.

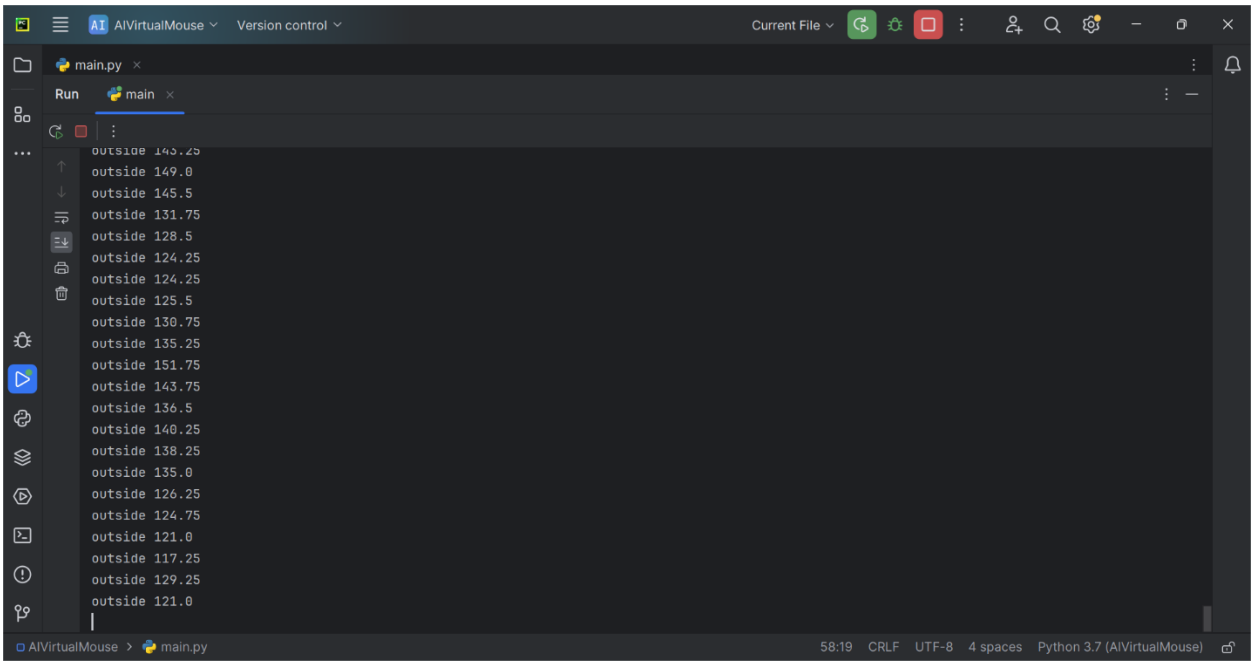
SOURCE CODE:

```
"""
1. Capture the video from the video source
2. Detecting the Hand by importing mediapipe
3. Separating the index fingertip
4. Moving the mouse pointer using finger
5. Click operation
"""

import cv2
import mediapipe as mp
import pyautogui

cap = cv2.VideoCapture(0)  # Capturing the video
hand_detector = mp.solutions.hands.Hands()
drawing_utils = mp.solutions.drawing_utils  # Draws the landmarks
screen_width, screen_height = pyautogui.size()  # Getting screen width and height
index_x, index_y = 0, 0
threshold = 60  # Adjust threshold as needed
# Exponential Moving Average parameters
alpha = 0.5
prev_x, prev_y = index_x, index_y
pyautogui.PAUSE = 0
pyautogui.FAILSAFE = 0

while True:
    _, frame = cap.read()
    frame = cv2.flip(frame, 1)
    frame_height, frame_width, _ = frame.shape
    rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)  # RGB mode is
used for Detecting anything from Frame or Video
    output = hand_detector.process(rgb_frame)  # Processing the RGB Frame
    hands = output.multi_hand_landmarks  # landmarks are the points in our Hand
```

```
 if hands:
        for hand in hands:
            drawing_utils.draw_landmarks(frame, hand)  # drawing_utils draws the
landmarks and displaying on the frame
            landmarks = hand.landmark
            for id, landmark in enumerate(landmarks):
                x = int(landmark.x * frame_width)  # since x-axis is horizontal
                y = int(landmark.y * frame_height)  # since y-axis is vertical
                if id == 8:  # For index finger
                    cv2.circle(img=frame, center=(x, y), radius=10, color=(0, 255, 255))
# Yellow circle formation
                    index_x = screen_width / frame_width * x
                    index_y = screen_height / frame_height * y
                    pyautogui.moveTo(index_x, index_y)  # moving the cursor
                if id == 4:  # For thumb finger
                    cv2.circle(img=frame, center=(x, y), radius=10, color=(0, 255, 255))
                    thumb_x = screen_width / frame_width * x
                    thumb_y = screen_height / frame_height * y
                    print('outside', abs(index_y - thumb_y))
                    if abs(index_y - thumb_y) < threshold:  # absolute difference
                        pyautogui.click()  # clicking
                        pyautogui.sleep(1)
    # Apply Exponential Moving Average to smooth cursor movement
    index_x = int(alpha * index_x + (1 - alpha) * prev_x)
    index_y = int(alpha * index_y + (1 - alpha) * prev_y)
    prev_x, prev_y = index_x, index_y
    pyautogui.moveTo(index_x, index_y, duration=0.1)  # Smoothly move the
mouse
    cv2.imshow('Virtual Mouse', frame)  # Displaying the image
    cv2.waitKey(1)
```

OUTPUT:





RESULT:

We have successfully completed a project on AI VIRTUAL MOUSE using hand gestures.