

Aim

To develop a real-time hand gesture recognition system that translates hand movements into cursor control and mouse clicks.

Objective

1. Recognize hand gestures using a webcam.
2. Translate hand movements into cursor position on the screen.
3. Implement a click action based on hand gesture detection.

Summary

The project leverages Mediapipe for hand gesture recognition and PyAutoGUI for controlling the cursor and executing mouse clicks. It captures video from the webcam, processes it to detect hand landmarks, maps these landmarks to screen coordinates, and performs actions based on hand gestures.

Tools and Libraries Used

- **OpenCV:** For video capture and image processing.
- **Mediapipe:** For hand gesture recognition and landmark detection.
- **PyAutoGUI:** For controlling the cursor and executing mouse clicks.

Procedure

1. **Initialization:**
 - Set up the camera and retrieve the screen size using PyAutoGUI.
 - Initialize Mediapipe's hand tracking module and drawing utilities.
2. **Capture Video Frame:**
 - Read frames from the webcam.
 - Flip the image horizontally to create a mirror effect.
3. **Pre-process Frame:**
 - Convert the image to RGB format as Mediapipe requires RGB images.
4. **Hand Landmark Detection:**
 - Process the RGB image with Mediapipe to detect hands and extract landmarks.
 - Draw landmarks on the image for visualization.
5. **Gesture Recognition:**
 - Identify specific landmarks (e.g., tip of the index finger) to track the cursor position.

- Calculate distances between landmarks to detect gestures (e.g., a pinch gesture for clicking).

6. Cursor Control:

- Map hand positions to screen coordinates.
- Move the cursor using PyAutoGUI based on the index finger position.
- Detect gestures such as pinch to perform a mouse click.

7. Display and Termination:

- Display the processed video feed with hand landmarks.
- Break the loop and release resources if the escape key is pressed.

CODE:

```

import cv2
import mediapipe
import pyautogui
capture_hands = mediapipe.solutions.hands.Hands()
drawing_option = mediapipe.solutions.drawing_utils
screen_width, screen_height = pyautogui.size()
camera = cv2.VideoCapture(0)
x1 = y1 = x2 = y2 = 0
while True:
    _, image = camera.read()
    image_height, image_width, _ = image.shape
    image = cv2.flip(image, 1)
    rgb_image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    output_hands = capture_hands.process(rgb_image)
    all_hands = output_hands.multi_hand_landmarks
    if all_hands:
        for hand in all_hands:
            drawing_option.draw_landmarks(image, hand)
            one_hand_landmarks = hand.landmark

```

```
for id, lm in enumerate(one_hand_landmarks):
    x = int(lm.x*image_width)
    y = int(lm.y*image_height)
    print(x,y)

    if id == 8:
        mouse_x = int(screen_width /image_width *x)
        mouse_y = int(screen_height /image_height *y)
        cv2.circle(image, (x,y), 10, (0,255,255))
        pyautogui.moveTo(mouse_x,mouse_y)

        x1=x
        y1=y

    if id == 4:
        x2=x
        y2=y
        cv2.circle(image, (x,y), 10, (0,255,255))

        dist =y2-y1
        print(dist)

    if(dist<20):
        pyautogui.click()

cv2.imshow("Hand movement video capture",image)
key = cv2.waitKey(100)
if key == 27:
    break
camera.release()
cv2.destroyAllWindows()

while True:
    _, image = camera.read()
    image_height, image_width, _ = image.shape
    image = cv2.flip(image, 1)
```

```
rgb_image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
```

```
mp_hands = mp.solutions.hands
```

```
hands = mp_hands.Hands()
```

```
drawing_options = mp.solutions.drawing_utils
```

```
output_hands = hands.process(rgb_image)
```

```
all_hands = output_hands.multi_hand_landmarks
```

```
if all_hands:
```

```
    for hand in all_hands:
```

```
        drawing_options.draw_landmarks(image, hand)
```

```
        one_hand_landmarks = hand.landmark
```

```
x1, y1, x2, y2 = 0, 0, 0, 0
```

```
for id, lm in enumerate(one_hand_landmarks):
```

```
    x = int(lm.x * image_width)
```

```
    y = int(lm.y * image_height)
```

```
    print(x, y)
```

```
if id == 8:
```

```
    mouse_x = int(screen_width / image_width * x)
```

```
    mouse_y = int(screen_height / image_height * y)
```

```
    cv2.circle(image, (x, y), 10, (0, 255, 255))
```

```
    pyautogui.moveTo(mouse_x, mouse_y)
```

```
x1, y1 = x, y
```

```
if id == 4:
```

```
x2, y2 = x, y
cv2.circle(image, (x, y), 10, (0, 255, 255))

dist = y2 - y1
print(dist)

if dist < 20:
    pyautogui.click()

cv2.imshow("Hand movement video capture", image)
key = cv2.waitKey(100)
if key == 27:
    break

camera.release()
cv2.destroyAllWindows()

import cv2
import pyautogui
import mediapipe as mp

camera = cv2.VideoCapture(0)
screen_width, screen_height = pyautogui.size()

while True:
    _, image = camera.read()
    image_height, image_width, _ = image.shape
    image = cv2.flip(image, 1)
    rgb_image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
```

```
mp_hands = mp.solutions.hands
hands = mp_hands.Hands()
drawing_options = mp.solutions.drawing_utils

output_hands = hands.process(rgb_image)
all_hands = output_hands.multi_hand_landmarks

if all_hands:
    for hand in all_hands:
        drawing_options.draw_landmarks(image, hand)
        one_hand_landmarks = hand.landmark

        x1, y1, x2, y2 = 0, 0, 0, 0

        for id, lm in enumerate(one_hand_landmarks):
            x = int(lm.x * image_width)
            y = int(lm.y * image_height)
            print(x, y)

            if id == 8:
                mouse_x = int(screen_width / image_width * x)
                mouse_y = int(screen_height / image_height * y)
                cv2.circle(image, (x, y), 10, (0, 255, 255))
                pyautogui.moveTo(mouse_x, mouse_y)
                x1, y1 = x, y

            if id == 4:
                x2, y2 = x, y
                cv2.circle(image, (x, y), 10, (0, 255, 255))
```

```

dist = y2 - y1
print(dist)

if dist < 20:
    pyautogui.click()

cv2.imshow("Hand movement video capture", image)
key = cv2.waitKey(100)
if key == 27:
    break

camera.release()
cv2.destroyAllWindows()

```

Code Highlights

- **Mediapipe Integration:**

CODE:

```

import mediapipe as mp
mp_hands = mp.solutions.hands
hands = mp_hands.Hands()
drawing_options = mp.solutions.drawing_utils

```

- **Cursor Movement and Click Detection:**

CODE:

```

mouse_x = int(screen_width / image_width * x)
mouse_y = int(screen_height / image_height * y)
pyautogui.moveTo(mouse_x, mouse_y)

if dist < 20:
    pyautogui.click()

```

- **Landmark Drawing:**

CODE:

```
drawing_options.draw_landmarks(image, hand)
```

Conclusion

This project successfully implements a hand gesture-based cursor control system. It demonstrates how real-time hand tracking can be used to interact with a computer interface, providing a hands-free method for cursor control and clicking. The system effectively maps hand movements to cursor actions, with real-time feedback and gesture-based click detection.