Controlling a Virtual Mouse using the live feed coming from the Webcam

Yash Mohan Jadhav
Department of Artificial Intelligence
Indian Institute of Technology Kharagpur
West Bengal, India

Krishna Kumar Sharma
Department of Artificial Intelligence
Indian Institute of Technology Kharagpur
West Bengal, India

***Abstract*—This project introduces a computer-vision-based hand gesture recognition system designed to control mouse movements and perform clicks without physical contact. Using a standard webcam and Python, the system captures and processes hand gestures in real time to emulate mouse actions. The program applies image pre-processing techniques, contour detection, and convexity defect analysis to identify specific gestures for moving, left-clicking, and right-clicking. This gesture-based control system aims to provide an alternative input method for individuals with disabilities or users in sterile environments. The experimental results indicate the feasibility of the approach and highlight areas for improvement in gesture recognition accuracy and responsiveness.**

***Index Terms*—human-computer interaction , Hand Gestures , Video Capture , Contour Detection , Convexity Defects , Virtual mouse**

I. Introduction

With the rapid advancement of computer vision technology, gesture-based control has become an innovative area for human-computer interaction (HCI). Traditional input devices, such as the mouse and keyboard, may not be practical or accessible in all environments or for all users. Gesture recognition offers a contactless and intuitive alternative, particularly beneficial for accessibility and sterile environments. This project demonstrates a webcam-based system that uses hand gestures to control mouse actions like movement, left-click, and right-click. By leveraging Python libraries like OpenCV and PyAutoGUI, the system can detect specific gestures based on hand contour features in real time, offering a new method for intuitive computer interaction while keeping the implementation cheap and easy compared to other methods.making it ideal in rapid and mass deployment in emergency cases.

II. Related Works

Gesture recognition has been a significant research area in HCI due to it's helpfulness

A. Using Wearable Glove with Sensors

Early approaches relied on sensor-based systems,like with a wearable gloves with sensors to capture finger movements which then will be used for hand gesture detection, which provided accurate results but were costly and less accessible due to its complex manufacturing as well as its custom nature as it has to be calibrated or custom made to each hand type. Making it less viable for mass deployment.

B. Using Machine Learning

Recent work has leveraged machine learning and computer vision techniques to develop hand gesture recognition systems without physical devices, using cameras to detect gestures through skin color segmentation and contour analysis [2]. Studies using deep learning methods, such as convolutional neural networks (CNNs), have demonstrated improved accuracy in gesture recognition. Making it really good for mass deployment. however, they often require large datasets and extensive computational resources making it not ideal for cheap and quick deployment in new fields.

III. Methodology

The proposed system uses Python, OpenCV, and PyAutoGUI libraries to interpret hand gestures from a webcam feed and emulate mouse actions. The methodology consists of several stages:

A. Video Capture

This project is utilizing a captured live video taken from the webcam of the device as a input for the pre-processing before the detection of the hand is to be performed.We are using the opencv-python library for cv2.VideoCapture as it has the necessary function for capturing of the live video.

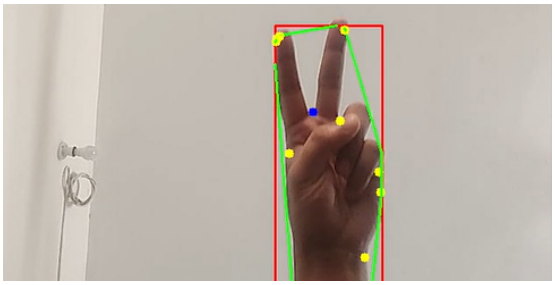

Fig. 1. the live feed

B. Pre - processing

In this project we are flips the image horizontally to create a mirror effect.Then we are applying a gray scale to the frame using cv2.cvtColor for adding consistency in processing the frame. Then applying Gaussian blur using cv2.GaussianBlur to smoothen the frame and to reduce noise.then a binary

thresholding with Otsu's method using cv2.threshold is applied for converting it to a binary image that highlights the hand's shape.

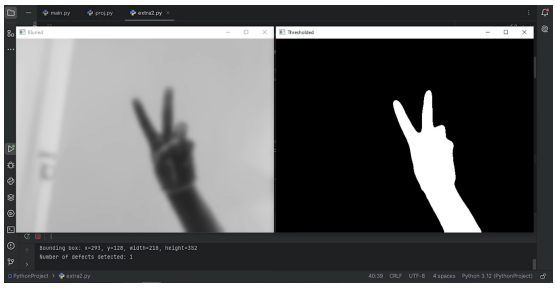

Fig. 2. GaussianBlur and Otsu's binary thresholding applied to the live feed

C. Contour Detection

In computer vision, a contour is like a digital representation of that outline. It can be described as the series of connected points that define the boundary of an object, separating and/or highlighting it from the background. These points tend to share similar color or intensity values, making them distinct from their surroundings.
In this project we are using Contour Detection to help us find our hand and separate it from the background noise.
Take the largest contours based on the contour Area of the contours that will most likely to be the hand and this process will remove all small disjointed contour that may produce from the background noise.

D. convexity hall

The convex hull is the smallest convex set that encloses all the points, forming a convex polygon
In this project we are using the newly taken Convex Hull to find the convexity defects.

E. convexity defects

Any deviation of the contour from its convex hull is known as the convexity defect
we are going to take the deepest vallies in the dataset as or final defects
Based on all the convexity defect find the angles and distances between the convexity defects to identify if there is a finger or not

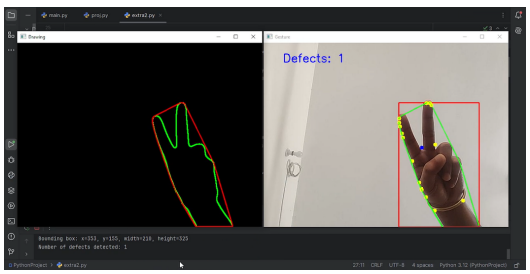

Fig. 3. convexity hall and convexity defects applied to the live feed

Algorithm 1 Gesture Recognition with Mouse Control

Initialize:
Disable PyAutoGUI failsafe.
Get screen resolution $(SCREEN\_X, SCREEN\_Y)$.
Initialize variables: $CLICK \leftarrow$ None, $MOVEMENT\_START \leftarrow$ None.
Start Video Capture:
Capture video from the default webcam.
while video capture is open **do**
  Capture a frame from the webcam.
  if frame is invalid **then**
    Break the loop.
  end if
  Flip the frame horizontally.
  Convert the frame to grayscale.
  Apply Gaussian blur to reduce noise.
  Apply Otsu's thresholding to create a binary image.
  Find Contours:
  Detect all contours in the binary image.
  Identify the largest contour based on area.
  if largest contour exists **then**
    Draw a bounding rectangle around the contour.
    Compute the convex hull of the contour.
    Identify convexity defects in the hull.
    Process Defects:
    Initialize $count\_defects \leftarrow 0$ and $used\_defect \leftarrow$ None.
    for each defect in the contour **do**
      Calculate triangle sides and angle at the far point.
      if angle $\leq 90°$ **then**
        Increment $count\_defects$.
        Mark the far point as a valid defect.
        if $count\_defects == 2$ **then**
          Track the defect as $used\_defect$.
        end if
      end if
    end for
    Perform Actions Based on Defects:
    if $count\_defects == 2$ **then**
      Move mouse based on the tracked defect.
    else if $count\_defects == 3$ and $CLICK ==$ None **then**
      Perform a left click.
      Set $CLICK \leftarrow$ current_time.
    else if $count\_defects == 4$ and $CLICK ==$ None **then**
      Perform a right click.
      Set $CLICK \leftarrow$ current_time.
    end if
  end if
  Display the frame with annotations.
  if ESC key is pressed **then**
    Break the loop.
  end if
end while
Release video capture and close all windows.

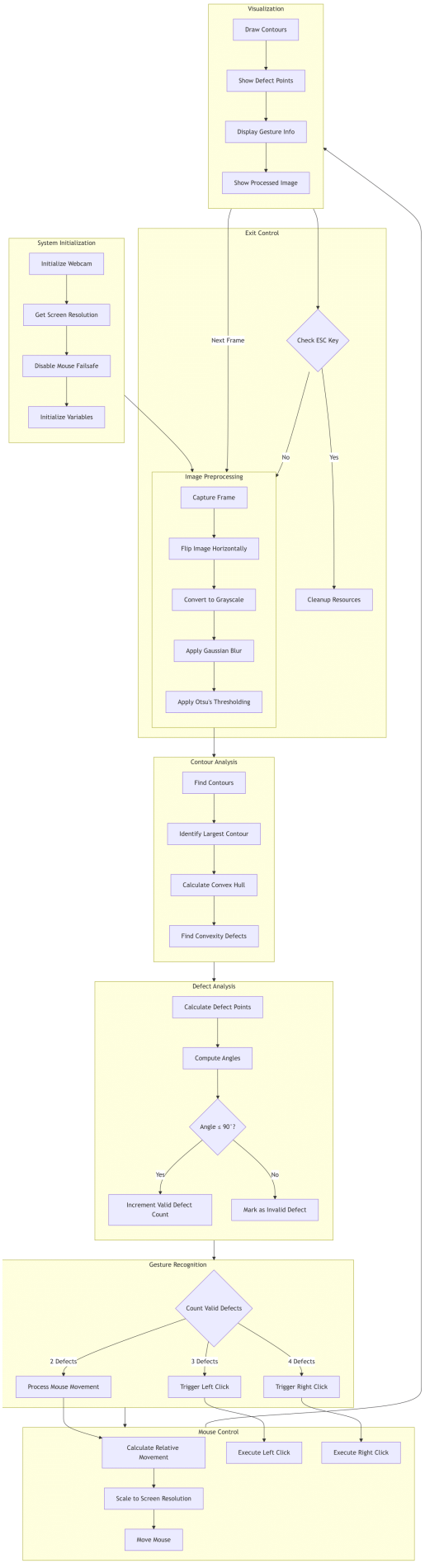

Fig. 4. the block diagram of the project

F. Identify specific gestures

Using the previously mentioned number of defects we can use them to control the mouse
As we can select certain options based on the specific number of defects we can perform the following:
1. mouse movement if 2 number of defects are counted
2. left click action if 3 number of defects are counted
3. right click action if 4 number of defects are counted

IV. Experimental Result

as the project is based on live feed and active processioning the detection of the hand gestures are as follows

A. Identify mouse movement gestures

when number of defects are 2 then mouse movement is performed

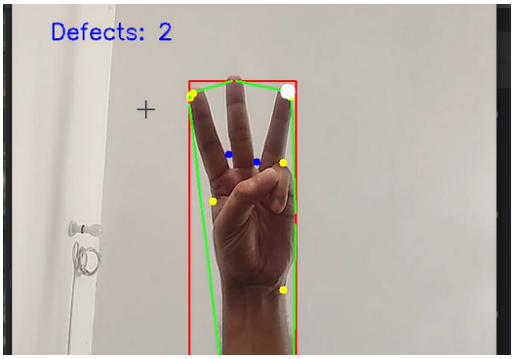

Fig. 5. mouse movement gestures

B. Identify Left Click gestures

when number of defects are 3 then Left Click action is performed

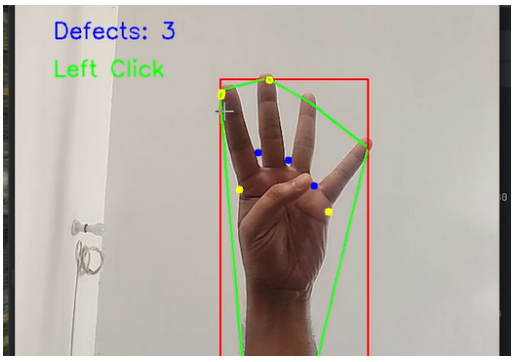

Fig. 6. Left Click gestures

C. Identify Right Click gestures

when number of defects are 4 then Right Click action is performed

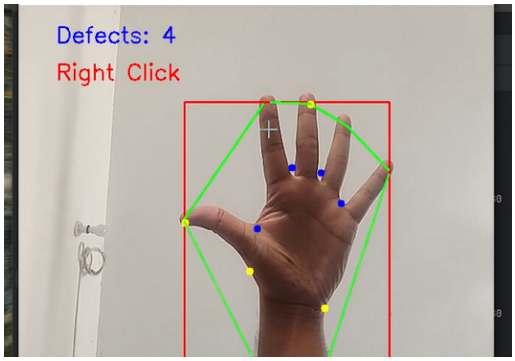

Fig. 7. Right Click gestures

V. Demonstration

You can access the Demonstration file on Google Drive by clicking the following link:
Click here to access the Live Feed on Google Drive file

You can access the Github by clicking the following link:
Click here to access the Github

VI. Conclusion

- **Gesture Detection Accuracy :** Accurate gesture recognition was observed in controlled lighting conditions as well as keeping a clear background . Performance degrades in low-light or low contrast situations due to variations in contour detection.
- **Responsiveness :** Real-time performance was achieved with minimal delay, but rapid hand movements led to occasional missed gestures due to the motion blur occurred in the camera.
- **Shortcomings :** fine movement control was quite challenging, indicating that may be a need of more calibration adjustments might be necessary .

VII. Future Work

Future work includes refining the gesture detection model, possibly by integrating machine learning models to improve accuracy across varying lighting conditions. Additionally, extending the gesture vocabulary and enhancing tracking stability will make the system more robust for practical applications.

References

- Mistry, P., Maes, P., Chang, L. (2009). WUW - Wear Ur World: A Wearable Gestural Interface. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.
- Yang, Y., Tian, Y., Gao, W. (2012). Skin-Color Segmentation-Based Hand Gesture Recognition. Journal of Information and Computational Science.
- Zhang, J., Chang, X. (2017). Real-Time Hand Gesture Recognition Based on Deep Learning. IEEE Transactions on Multimedia.

Appendix: Contribution

In this project, two students worked collaboratively on different components. Their contributions are as follows:

Student 1: Yash Mohan Jadhav

- Conceptualization: in determining on what can be used for hand detection.
- Data Curation: in setting up the live camera feed.
- Methodology: implemented the pre-processing and choosing of the contour with the highest area for background removal and hand isolation.
- Writing – Original Draft: lead the project presentation and report making process

Student 2: Krishna Kumar Sharma

- Conceptualization: in determining that convexity defect can be used for hand gesture detection process.
- Data Curation: test the working of the code.
- Methodology: implemented the contour detection and convex hull code.
- Writing – Original Draft: lead the code making process and live implementation.