SRH University Heidelberg

Professional Technical Report

Hand Gesture Recognition based Human-Computer Interaction

Author:	Abhijith Pavithran
Matriculation Number:	11018118
Email Address:	11018118@stud.hochschule-heidelberg.de
Supervisor:	Prof. Dr. Milan Gnjatovic
Project Begin Date:	16-02-2023
Project Submission Date:	15-03-2023

Table of Contents

ABSTRACT

This project report explores the implementation of hand gesture recognition based human-computer interaction using OpenCV, Mediapipe, and Pyautogui in Python language through Jupyter Notebook. The project involves capturing video input from a webcam or a video file using OpenCV and utilizing Mediapipe's Hand Detection module to detect the hand's position and landmarks in each frame of the video. The index finger (landmark 8) is separated to use it as a mouse pointer, which can be moved using Pyautogui's moveTo() function. The project also includes click operations using Pyautogui's click() function. By combining these features, the project creates a simple hand gesture recognition based human-computer interface that enables the user to move the mouse pointer and perform click operations using their index finger.

CHAPTER 1

INTRODUCTION

The ability to interact with computers using hand gestures has become an increasingly popular research topic in recent years. This technology has the potential to revolutionize human-computer interaction by providing a more natural and intuitive way to interact with digital devices. In this project report, we explore the use of OpenCV, Mediapipe, and Pyautogui to enable hand gesture recognition based human-computer interaction.

OpenCV is a powerful computer vision library that provides tools for image and video processing, while Mediapipe is a framework for building multimodal machine learning pipelines. Pyautogui, on the other hand, is a cross-platform GUI automation tool that enables us to control the mouse and keyboard using Python code.

In this project, we capture video input from a webcam or a video file using OpenCV, and use Mediapipe's Hand Detection module to detect the position and landmarks of the hand in each frame of the video. We then separate the index finger (landmark 8) so that we can use that as a mouse pointer and move the mouse pointer using Pyautogui's moveTo() function. We also perform click operations using Pyautogui's click() function. By combining these features, we create a simple hand gesture recognition based human-computer interface where the user can move the mouse pointer and perform click operations using their index finger.

This project has many potential applications in the field of human-computer interaction, especially in situations where conventional input devices may not be available or practical. For example, this technology could be used to develop touchless interfaces for public computers or kiosks, where users could interact with the interface using hand gestures.

However, there are limitations to this technology, such as its dependence on lighting conditions and the quality of the camera being used. The position of the index finger may also be affected by occlusion or other objects in the scene.

Therefore, this technology may not be suitable for applications that require high accuracy or reliability.

In summary, this project provides a good introduction to hand gesture recognition based human-computer interaction and serves as a starting point for further research and development in this area.

CHAPTER 2

BACKGROUND

Hand gesture recognition based human-computer interaction is an emerging field that has the potential to revolutionize the way we interact with digital devices. This technology has many potential applications in various fields, including gaming, healthcare, robotics, and education.

OpenCV is a widely used computer vision library that provides tools for image and video processing. It has many built-in functions for feature detection, object tracking, and machine learning, which make it an ideal choice for developing hand gesture recognition systems.

Mediapipe is a machine learning framework developed by Google that enables the development of multimodal machine learning pipelines. It provides pre-built models for common tasks such as face detection, hand detection, and pose estimation, which makes it easy to develop complex systems without having to develop everything from scratch.

Pyautogui is a cross-platform GUI automation tool that enables us to control the mouse and keyboard using Python code. It provides functions for mouse movement, click operations, keyboard input, and other GUI automation tasks.

The combination of OpenCV, Mediapipe, and Pyautogui provides a powerful toolset for developing hand gesture recognition based human-computer interaction systems. By using the Hand Detection module provided by Mediapipe, we can detect the position and landmarks of the hand in each frame of the video. By separating the index finger landmark, we can use it as a mouse pointer and move it using Pyautogui's moveTo() function. We can also perform click operations using Pyautogui's click() function.

The development of hand gesture recognition based human-computer interaction systems has the potential to revolutionize the way we interact with digital devices. It can provide a more natural and intuitive way to interact with computers, which can be especially useful for people with disabilities or in situations where conventional input devices may not be available or practical.

In summary, the combination of OpenCV, Mediapipe, and Pyautogui provides a powerful toolset for developing hand gesture recognition based human-computer interaction systems. This technology has many potential applications and can provide a more natural and intuitive way to interact with digital devices.

CHAPTER 3

FLOW CHART

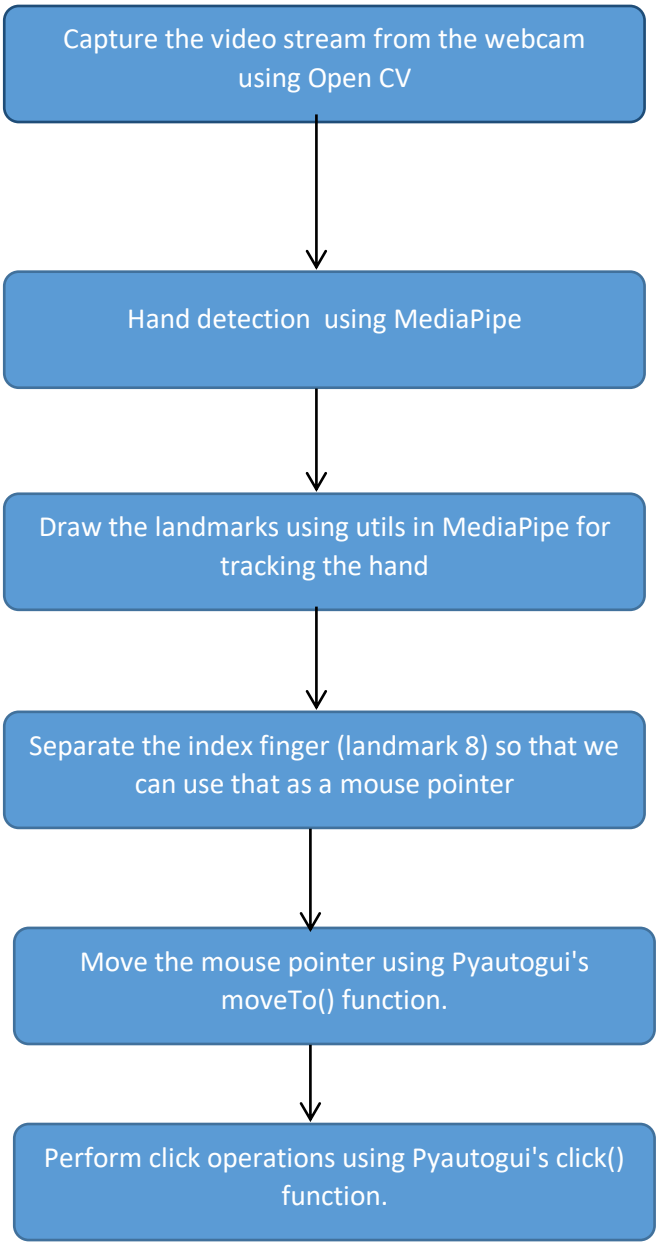


Figure 3.1: Flow chart representing the process involved in the implementation of Hand Gesture Recognition based Human Computer Interaction

CHAPTER 4

Hand Gesture Recognition based Human Computer Interaction

Python programming language is used for developing the Hand Gesture Recognition based Human- Computer Interaction system, and OpenCV which is the library for computer vision is used in the Hand Gesture Recognition based Human- Computer Interaction system. In the proposed Hand Gesture Recognition based Human- Computer Interaction system, the model makes use of the MediaPipe package for the shadowing of the hands and for shadowing of the tip of the hand, and PyAutoGUI packages were used for moving around the window screen of the computer for performing functions similar as click. The algorithm makes use of the machine learning generalities to track and fete the hand gestures and hand tip.

4.1 OpenCV

OpenCV is a computer vision library which contains image-processing algorithms for object detection. OpenCV is a library of python programming language, and real-time computer vision applications can be developed by using the computer vision library. The OpenCV library is used in image and video processing and analysis such as face detection and object detection.

In a hand gesture recognition-based human-computer interaction project, the first step in capturing the video stream is to initialize the webcam. This is done using OpenCV's VideoCapture function.

The VideoCapture function is a built-in function in OpenCV that is used to capture video frames from a webcam or video file. When called, the function creates a connection to the default camera of the system and returns a VideoCapture object.

The VideoCapture object can then be used to access the video frames as a sequence of images. Each frame is represented as an image matrix in the form of an array of pixels. The dimensions of the image matrix depend on the resolution of the camera and the size of the captured video frame.

To initialize the webcam using the VideoCapture function, the first step is to import the OpenCV library. This can be done using the following code:

```
import cv2
```

Once the OpenCV library is imported, the VideoCapture function can be used to initialize the webcam. This can be done using the following code:

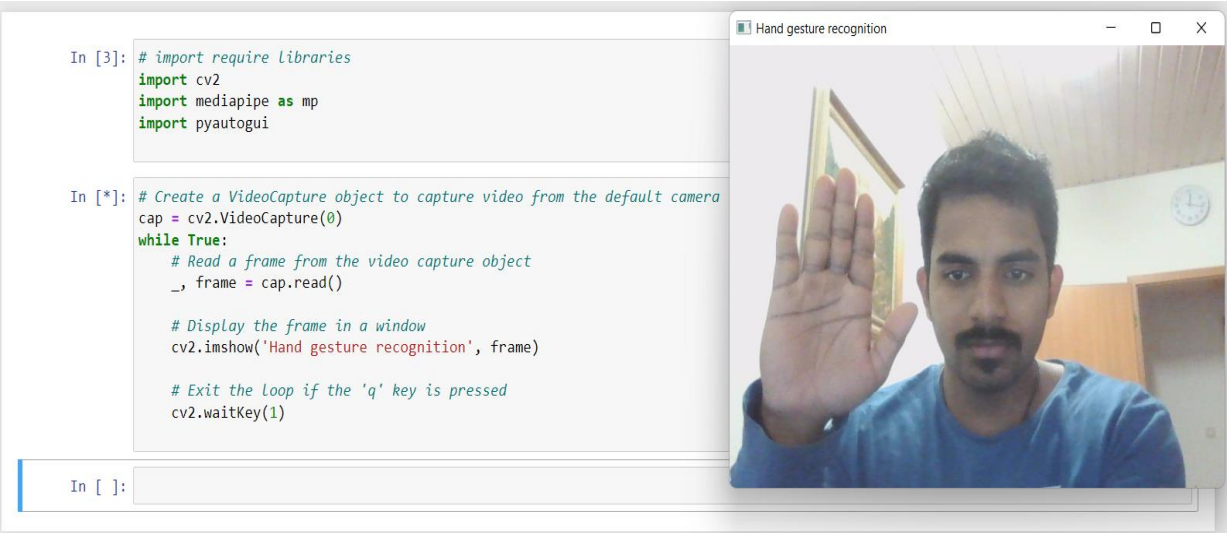


Figure 4.1:Capture video input from a webcam using OpenCV.

The above code initializes the default camera of the system with index 0. If multiple cameras are connected to the system, the index can be changed accordingly to select a different camera.

After initializing the webcam, the next step is to continuously capture video frames from the webcam and process them for hand gesture recognition. This is typically done using a loop that reads the current frame from the VideoCapture object and passes it to the hand detection and gesture recognition functions.

In summary, initializing the webcam using OpenCV's VideoCapture function is a crucial step in capturing the video stream for a hand gesture recognition-based human-computer interaction project. The VideoCapture function creates a

connection to the default camera of the system and returns a VideoCapture object, which can be used to access the video frames as a sequence of images.

4.2 MediaPipe

In this project, Mediapipe is used to perform hand detection and landmark estimation. The Mediapipe library provides pre-trained machine learning models and a set of easy-to-use Python APIs for processing and analyzing various types of media data, including video and images.

In particular, we use the Hands module of Mediapipe, which is a real-time hand tracking solution that can detect and track multiple hands in a video stream. The Hands module takes as input the image frames captured by OpenCV and outputs the position of each detected hand in the image as well as a set of 21 landmarks for each hand. These landmarks correspond to key points on the hand, such as the tips of the fingers and the base of the palm.

By utilizing the Hand module of Mediapipe, we can accurately track the position and movement of the hand in real-time, which enables us to perform hand gesture recognition and use it for human-computer interaction. Overall, Mediapipe provides a powerful and convenient tool for processing and analyzing media data, and its Hands module is especially useful for applications that involve hand tracking and gesture recognition.

4.3 Initialize Hand detection model

In a hand gesture recognition-based human-computer interaction project, the hand detection model plays a crucial role in detecting the position and orientation of the hand and fingers in real-time. One popular library for hand detection is the MediaPipe library, which provides pre-trained hand detection models.

To initialize the hand detection model using MediaPipe, the first step is to import the library into the project. This can be done using the following code:

```
import mediapipe as mp
```

Once the MediaPipe library is imported, the hand detection model can be initialized using the Hands class. This class provides a pre-trained hand detection model that can detect the position and orientation of the hand and fingers in real-time.

hand_detector = mp.solutions.hands.Hands()

In the given line of code, we are using Mediapipe's hands module to create a hands detector object. The hands detector object is created by calling the Hands() function provided by the mp.solutions.hands module.

This function creates an instance of the Hands class, which can be used to detect hands in an input image or video stream. The Hands class is based on a deep learning model that has been trained to detect hand landmarks and gestures.

By creating an instance of the Hands class using the mp.solutions.hands.Hands() function, we can access various methods and attributes that allow us to detect the hand landmarks and gestures in real-time video input using the Mediapipe library.

After initializing the hands object, the next step is to process the video frames from the webcam for hand detection. Before passing the video frames to the hand detector, the frame must be converted from the default OpenCV BGR color format to the RGB format required by MediaPipe.

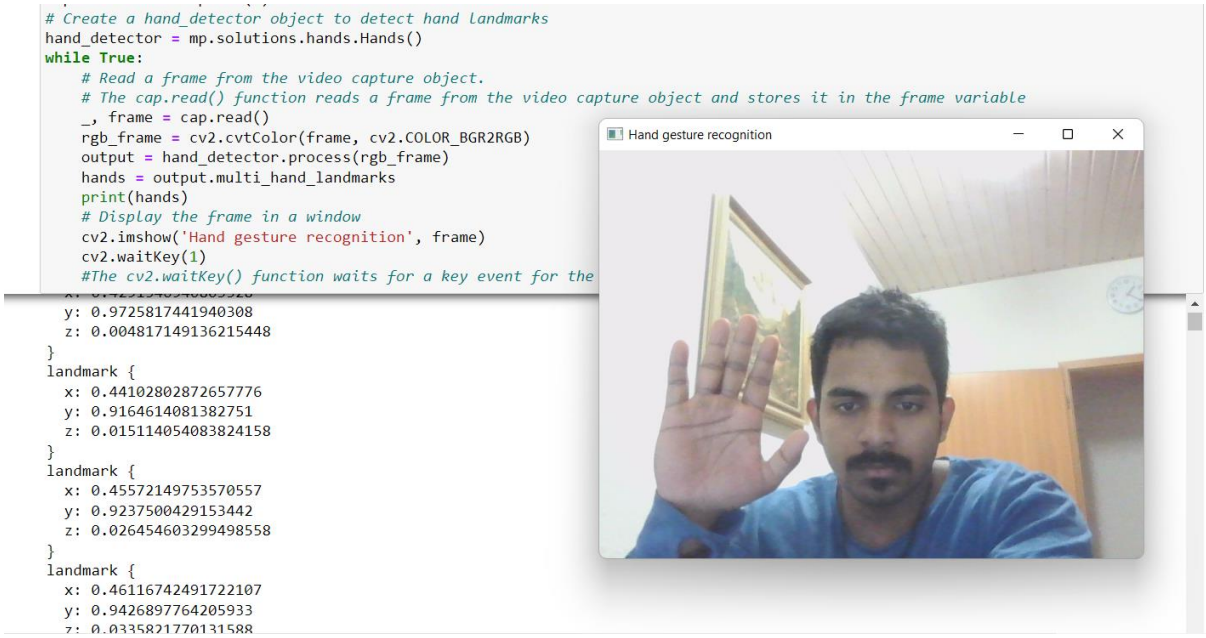


Figure 4.2: Convert the color of the frame from BGR to RGB

```
while True:
  _, frame = cap.read()
  rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
  output = hand_detector.process(rgb_frame)
  hands = output.multi_hand_landmarks
```

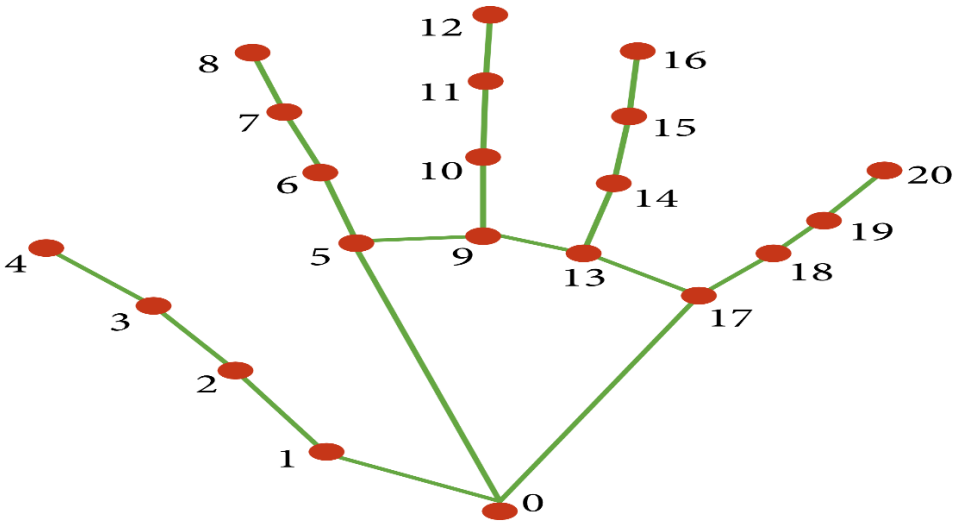
The above code reads the current frame from the VideoCapture object and converts it from BGR to RGB format using OpenCV's cvtColor function. The RGB frame is then passed to the hand detector using the process function of the hands object.

The process function returns a dictionary of hand detection results, including the multi-hand landmarks. These landmarks are the position and orientation of the hand and fingers in the form of (x, y, z) coordinates. The landmarks can be used to identify the hand gestures and map them to specific computer actions using PyAutoGUI.

In summary, initializing the hand detection model using the MediaPipe Hands class is an essential step in detecting the position and orientation of the hand and fingers in real-time. The hand detector is initialized by passing RGB frames to the hands object, and the multi-hand landmarks are obtained as part of the hand detection results. These landmarks can be used to identify the hand gestures and map them to specific computer actions.

4.4 Multi Hand Landmark

Single-shot detector model is used for detecting and recognizing a hand or palm in real time. The single-shot detector model is used by the MediaPipe. First, in the hand detection module, it is first trained for a palm detection model because it is easier to train palms. Furthermore, the non-maximum suppression works significantly better on small objects such as palms or fists. A model of hand landmark consists of locating 21 joint or knuckle co-ordinates in the hand region, as shown in Figure 2.



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Figure 4.3: A model of hand landmark consists of locating 21 joint or knuckle co-ordinates in the hand region.

In a hand gesture recognition-based human-computer interaction project, once the hand detection model has been initialized using MediaPipe and the multi-hand landmarks have been obtained, the next step is to visualize the landmarks on the video frames. This can be done using OpenCV's drawing functions and the MediaPipe utility functions.

In this project, drawing_utils is used to draw the landmarks of the hand detected by the Hand Detection module on each frame of the video stream, which makes it easier for the user to see the position and movement of their hand on the screen.

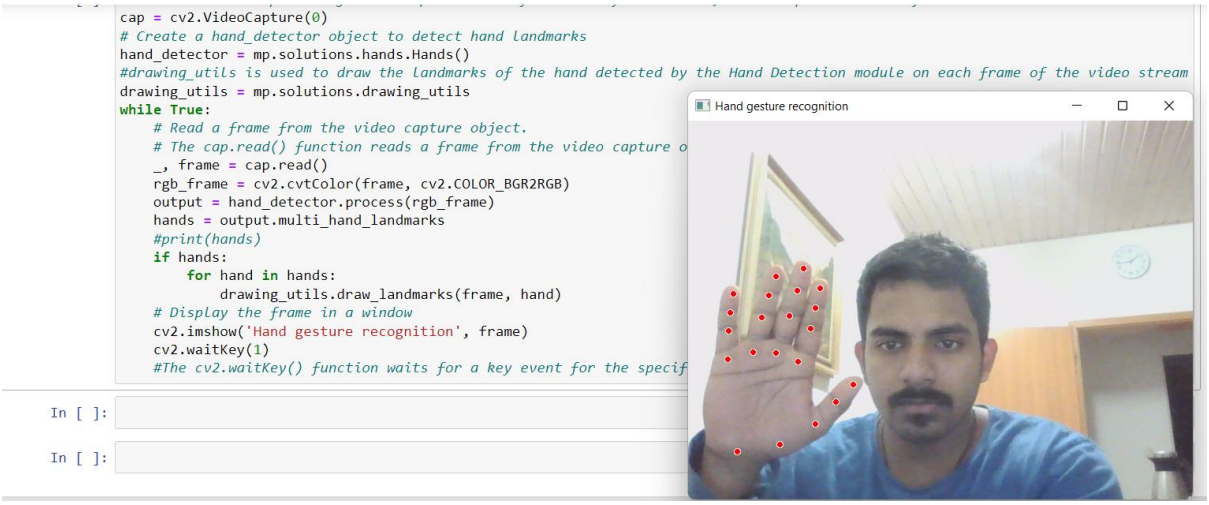


Figure 4.3: Representation of multi hand landmarks

The mp.solutions.drawing_utils function returns a module containing various utility functions that can be used to draw annotations on the input images or frames. These functions include drawing landmarks, connections, bounding boxes, and text labels. The landmarks are represented as small circles. The drawing_utils module simplifies the process of visualizing the output of the Hand Detection module, making it easier to understand the position and orientation of the hand in each frame of the video.

Each landmark in the multi-hand landmark list corresponds to a specific point on the hand, such as the tip of the index finger or the base of the thumb. To recognize hand gestures, it is necessary to identify which landmarks correspond to which fingers and how they move relative to each other.

One way to do this is to extract the coordinates of each landmark and use them to calculate the angles and distances between the fingers. To extract the coordinates, the enumerate() function can be used to iterate over the landmarks and obtain their ID, x-coordinate, and y-coordinate:

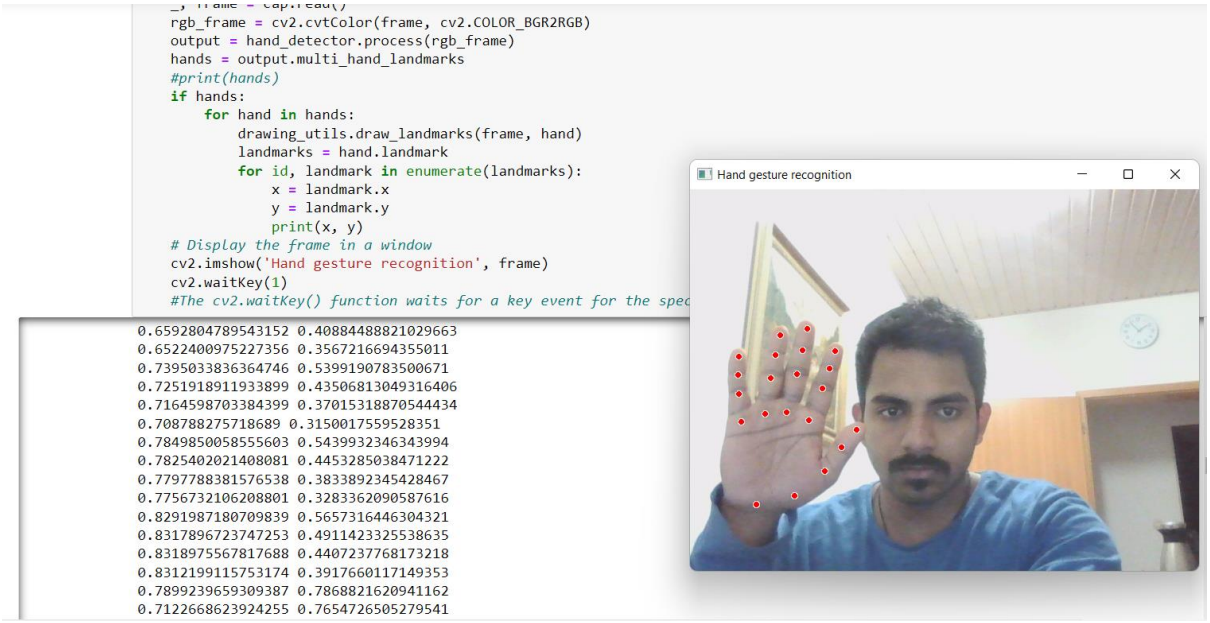


Figure 4.4: For each landmark detected, get the coordinates of the landmark

The frame_height, frame_width, _ = frame.shape operation is used to get the height and width of the input frame. The shape attribute of the frame object returns a tuple containing the height, width, and number of channels of the image. In this project, we are only interested in the height and width of the frame.

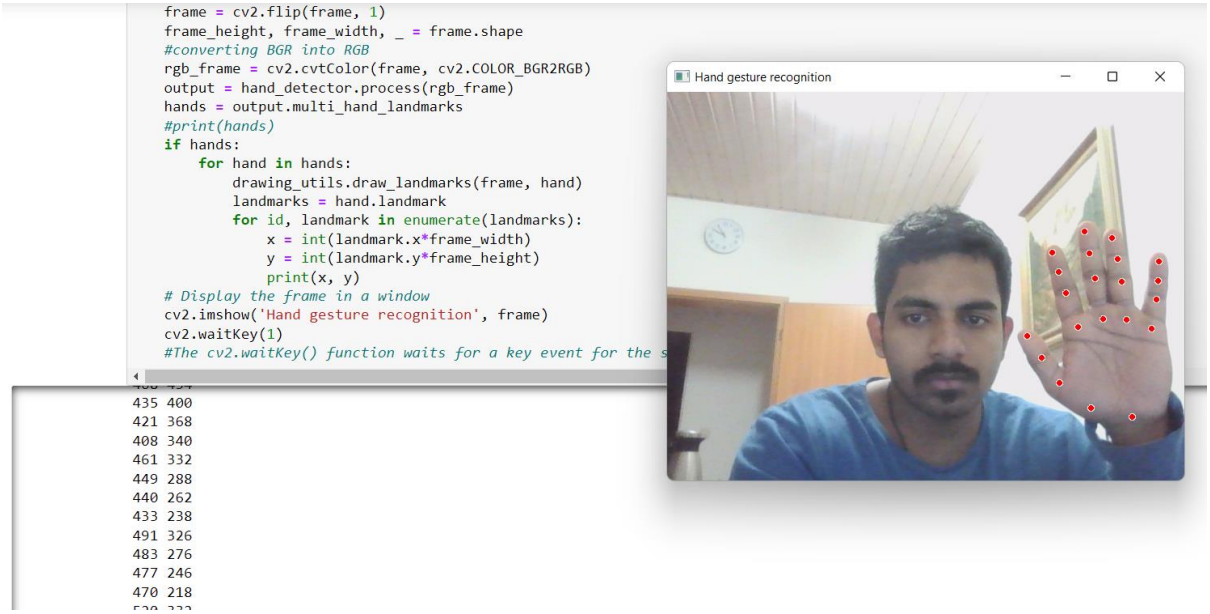


Figure 4.4: Get the dimensions of the frame

Later in the code, these values are used to calculate the pixel coordinates of the landmarks detected by the hand detection model. Specifically, the x and y coordinates of each landmark are normalized values between 0 and 1, representing the relative position of the landmark within the frame. To convert these normalized values into pixel coordinates, we need to multiply them by the height and width of the frame, respectively.

Thus, we use the frame_height and frame_width variables to scale the normalized landmark coordinates and obtain the actual pixel coordinates in the input frame.

4.5 Separating the index finger

In the project of hand gesture recognition based human-computer interaction using Mediapipe and PyAutoGUI, the separating of index finger (landmark 8) is used as a gesture to communicate with the computer. When the index finger is separated from the rest of the hand, the program recognizes this gesture and performs a specific action, such as clicking.

To detect the separating of index finger, the program uses the landmark coordinates provided by Mediapipe. Each landmark has a unique ID, and the index finger landmark has ID 8. Therefore, the program can check if the index finger landmark is separated by checking the distance between the index finger landmark and other landmarks in the hand.

```
if id == 8:

    cv2.circle(img=frame, center=(x,y), radius=10, color=(0, 255, 255))
```

In the code snippet provided, the program checks if the landmark ID is 8 using the if statement if id == 8. If the landmark ID is 8, the program draws a circle around the landmark using the cv2.circle function. This is done to provide visual feedback to the user and indicate that the program has recognized the gesture.

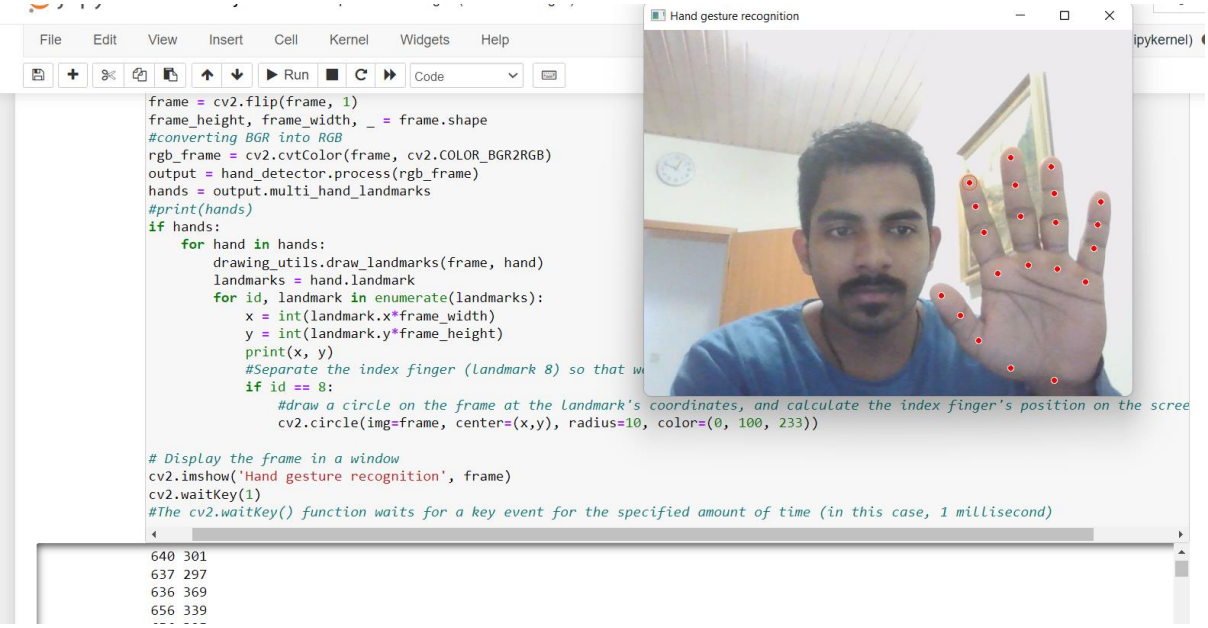


Figure 4.5: Draw a circle on the fra:me at the landmark's coordinates, and calculate the index finger's position on the screen in pixels

4.6 Full screen cursor movement

In the project of hand gesture recognition based human-computer interaction using PyAutoGUI, full screen cursor movement is achieved by using the index_x and index_y variables to map the position of the index finger landmark to the full screen coordinates.

index_x = screen_width/frame_width*x

index_y = screen_height/frame_height*y

pyautogui.moveTo(index_x, index_y)

The index_x and index_y variables are calculated based on the ratio of the position of the index finger landmark in the captured video frame to the size of the screen. Specifically, the index_x variable is calculated by dividing the width of the screen by the width of the captured video frame and multiplying it by the x coordinate of the index finger landmark. The index_y variable is calculated in the same way, but using the height of the screen and the height of the captured video frame.

Once the index_x and index_y variables are calculated, the program can use them to control the movement of the mouse cursor on the full screen. The

pyautogui.moveTo() function is used again, but this time with the index_x and index_y variables as arguments to move the cursor to the corresponding position on the full screen.

The benefit of using index_x and index_y variables is that it allows the program to work on screens with different resolutions and aspect ratios. By mapping the position of the index finger landmark to the full screen coordinates, the program can provide full-screen cursor movement regardless of the screen size and aspect ratio.

Overall, the full-screen cursor movement is an important feature in the project of hand gesture recognition-based human-computer interaction using PyAutoGUI. By allowing users to control the position of the mouse cursor on the full screen using hand gestures, the program provides a more natural and intuitive way to interact with the computer, especially for tasks that require precise cursor movement on the full screen.

4.7 Click operation

In the project of hand gesture recognition based human-computer interaction using PyAutoGUI, the click operation is implemented using the index and thumb finger landmarks. Specifically, the program detects the distance between the index and thumb finger landmarks and uses it to determine whether the user is making a "click" gesture.

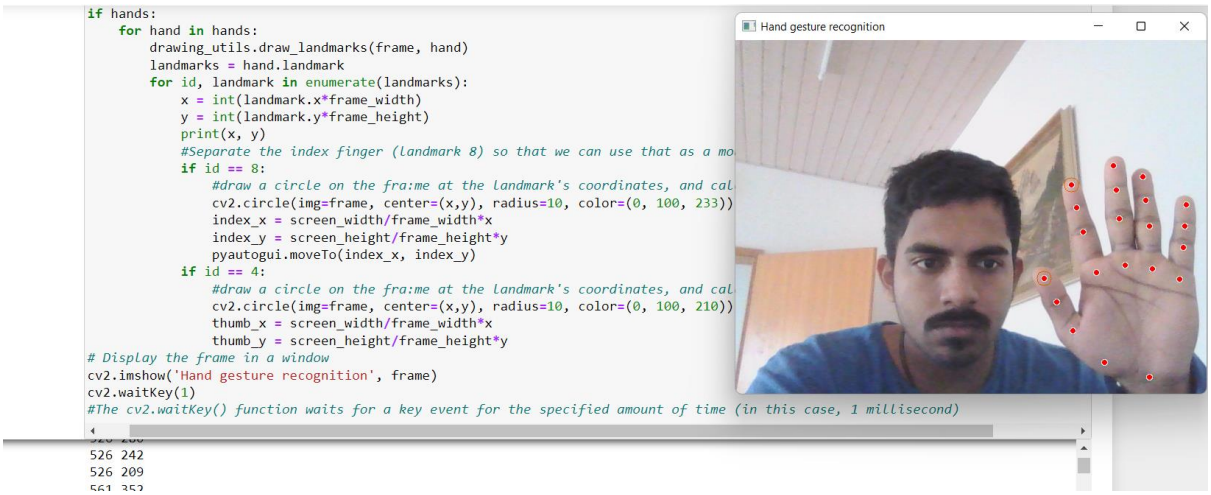


Figure 4.6: simulate a left mouse button click

To perform a click operation, the user must bring the tips of their index and thumb fingers together, creating a pinch gesture. The program detects this gesture by calculating the Euclidean distance between the index and thumb

finger landmarks. If the distance is less than a certain threshold (20 pixels in this case), the program considers it a "click" gesture.

If the distance between the index and thumb fingers is greater than the threshold but less than 100 pixels, the program moves the mouse cursor to the x and y coordinates of the index finger using the pyautogui.moveTo() function. This allows the user to move the mouse cursor using the index finger.

Overall, this implementation allows the user to perform left-click operations by bringing the tips of the index and thumb fingers together, while also providing the ability to move the mouse cursor using the index finger.

CHAPTER 5

Advantages and Disadvantages

5.1 Advantages

There are several advantages to using hand gesture recognition based human computer interaction such as:

5.1.1 Natural interaction

This method allows users to interact with the computer in a natural and intuitive way, using hand gestures that are similar to how they would interact with objects in the physical world. This can make the interaction more comfortable and efficient, especially for tasks that require precise control or manipulation.

5.1.2 Accessibility

Hand gesture recognition based interaction can be useful for people with disabilities that prevent them from using traditional input devices, such as a mouse or keyboard. This method can also be used in scenarios where using traditional input devices is not possible, such as in virtual or augmented reality environments.

5.1.3 Cost effective

Hand gesture recognition based interaction does not require any additional hardware beyond a standard webcam or camera, making it a cost-effective solution.

5.1.4 Customizability

The use of open-source software such as OpenCV, Mediapipe, and Pyautogui allows for customization of the system to fit specific user needs or preferences. This can include adjusting the gestures recognized, the actions performed, or the sensitivity of the system.

5.1.5 Scalability

Hand gesture recognition based interaction can be used in a variety of scenarios, ranging from personal computing to industrial or commercial applications. It can also be easily integrated into existing software systems or applications.

5.2 Disadvantages

Following are some of the disadvantages of Hand gesture recognition based human computer interaction.

5.2.1 Limited recognition

The hand gesture recognition system may not be able to recognize all hand gestures or variations in hand movements, leading to inaccuracies or errors in interaction.

5.2.2 Environmental constraints

The system's performance can be affected by environmental factors such as lighting conditions, camera placement, or obstacles in the user's workspace, leading to inconsistent performance.

5.2.3 Physical strain

Extended use of hand gesture recognition based interaction may result in physical strain or discomfort, especially if the user is required to hold their hand in a certain position for a prolonged period.

CHAPTER 6

6.1 Applications

Hand gesture recognition based human computer interaction has a wide range of potential applications in various industries, including:

6.1.1 Gaming

Hand gesture recognition can enhance the gaming experience by allowing players to interact with games using hand gestures instead of traditional input devices such as controllers or keyboards.

6.1.2 Robotics

Hand gesture recognition can be used in robotics to enable robots to recognize and respond to human gestures, allowing for more natural and intuitive communication between humans and robots.

6.1.3 Healthcare

In healthcare, hand gesture recognition can be used to control medical devices without touching them, reducing the risk of contamination and infection. It can also be used in rehabilitation to monitor and track the progress of patients.

6.1.4 Education

In education, hand gesture recognition can be used to create more engaging and interactive learning experiences, allowing students to interact with educational materials using hand gestures.

6.1.5 Security

In security applications, hand gesture recognition can be used to enable access control systems that recognize hand gestures, increasing security and convenience.

6.1.6 Accessibility

Hand gesture recognition can be used to enable people with disabilities to interact with computers and other devices using hand gestures, increasing accessibility and independence.

6.1.7 Industrial control systems

Hand gesture recognition can be used in industrial control systems to enable workers to control machinery and equipment using hand gestures, potentially increasing safety and efficiency.

CHAPTER 7

Conclusion

In conclusion, we have explored the use of OpenCV, Mediapipe, and Pyautogui to enable hand gesture recognition based human-computer interaction. We started by capturing video input from a webcam or a video file using OpenCV, which allowed us to process each frame of the video and detect the position of the hand using Mediapipe's Hand Detection module. We then separated the index finger (landmark 8) so that we could use that as a mouse pointer.

Using the position of the index finger, we were able to move the mouse pointer using Pyautogui's moveTo() function. We also performed click operations using Pyautogui's click() function. By combining these features, we were able to create a simple hand gesture recognition based human-computer interface where the user could move the mouse pointer and perform click operations using their index finger.

This project has many potential applications in the field of human-computer interaction, especially in situations where conventional input devices such as mice and keyboards may not be available or practical. For example, this technology could be used to develop touchless interfaces for public computers or kiosks, where users could interact with the interface using hand gestures.

However, there are some limitations to this technology. The accuracy of hand gesture recognition is heavily dependent on the lighting conditions and the quality of the camera being used. The position of the index finger may also be affected by occlusion or other objects in the scene. Therefore, this technology may not be suitable for applications that require high accuracy or reliability.

Overall, this project provides a good introduction to hand gesture recognition based human-computer interaction and serves as a starting point for further research and development in this area.

Reference

1.OpenCV documentation: https://docs.opencv.org/master/

2.Mediapipe documentation: https://google.github.io/mediapipe/

3.Pyautogui documentation: https://pyautogui.readthedocs.io/en/latest/

4.https://developers.google.com/mediapipe/solutions/vision/gesture_recognizer/python

5. https://www.ijsrp.org/research-paper-1214/ijsrp-p3693.pdf

6. https://www.computervision.zone/lessons/code-files-17/

7. https://www.youtube.com/watch?v=8gPONnGIPgw&t=366s

8. https://iopscience.iop.org/article/10.1088/1757-899X/1045/1/012043/meta

9. http://gnjatovic.info/imageprocessing/