MINI PROJECT REPORT ON

HAND GESTURE RECOGNITION

Submitted in partial fulfillment of the requirement for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE & ENGINEERING

Submitted by:

Devansh Rautela

2018314

Under the Mentorship of

Mr. Samir Rana

Department of Computer Science and Engineering
Graphic Era Hill University
Dehradun, Uttarakhand

January 2023

Graphic Era
HILL UNIVERSITY
Established by an Act of the State Legislature of Uttarakhand (Adhiniyam Sankhya 12 of 2011)

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the project report entitled **"Hand Gesture Recognition"** in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science and Engineering of the Graphic Era Hill University, Dehradun shall be carried out by the under the mentorship of **Mr. Samir Rana**, Department of Computer Science and Engineering, Graphic Era Hill University, Dehradun.

Devansh Rautela
2018314

Table of Contents

Chapter 1

Introduction

Hand gesture recognition is a rapidly growing field that uses computer vision and machine learning techniques to interpret hand movements and gestures in real-time. One of the most promising frameworks for building hand gesture recognition systems is MediaPipe, an open-source platform developed by Google. In this project, we will be using MediaPipe to create a hand gesture recognition system that is able to detect and classify different hand gestures in real-time video streams.

The first step in building our system will be to collect a dataset of hand gestures. This dataset will be used to train a machine learning model that can accurately recognize different hand gestures. We will be using a combination of convolutional neural networks (CNNs) and recurrent neural networks (RNNs) to build our model. CNNs will be used to extract features from the images of hand gestures, while RNNs will be used to process the sequence of hand gestures over time.

Once the model has been trained, it will be integrated into the MediaPipe framework. MediaPipe provides a number of pre-built pipelines for computer vision tasks such as object detection, facial landmark detection, and hand tracking. We will use these pipelines to detect and track the hand in the video stream, and then use our trained model to classify the hand gestures.

One of the key advantages of using MediaPipe is its ability to perform real-time processing of video streams. This means that our hand gesture recognition system will be able to provide immediate feedback to the user, making it suitable for a wide range of applications. Some examples include gaming, virtual reality, human-computer interaction, and sign language translation.

In conclusion, the use of MediaPipe for hand gesture recognition system provides a robust and efficient solution for detecting and classifying hand gestures in real-time video streams. With its ability to perform real-time processing and pre-built pipelines, MediaPipe makes it easy to build a hand gesture recognition system that can be applied to a wide range of applications.

Chapter 2

Methodology

This project aims to create a virtual mouse using MediaPipe and hand gestures as inputs. The goal is to use the MediaPipe framework to track and detect hand movements in real-time video streams, and then use those movements to control the cursor on a computer screen. The virtual mouse will be implemented using Python, and will be based on the MediaPipe hand tracking and detection model.

The first step in building the virtual mouse will be to collect a dataset of hand gestures that will be used to train the MediaPipe model. This dataset will include a variety of hand gestures such as pointing, clicking, and scrolling. Once the dataset is collected, the MediaPipe model will be trained using a combination of convolutional neural networks (CNNs) and recurrent neural networks (RNNs).

Once the model is trained, it will be integrated into the MediaPipe framework. MediaPipe provides a number of pre-built pipelines for computer vision tasks such as object detection, facial landmark detection, and hand tracking. We will use these pipelines to detect and track the hand in the video stream, and then use our trained model to classify the hand gestures.

The next step will be to map the hand gestures to mouse movements and actions. For example, pointing the hand in a certain direction may move the cursor in that direction, while a clicking gesture may simulate a left-click. This mapping will be done using Python, and will utilize the data provided by the MediaPipe model.

Finally, the virtual mouse will be tested and evaluated using a variety of different hand gestures and movements. This will include both standard

mouse actions (clicking, scrolling, etc.) as well as more complex gestures such as zooming or rotating.

Overall, this project will demonstrate the capabilities of MediaPipe for hand gesture recognition and how it can be used to create a virtual mouse using hand gestures as inputs. The virtual mouse will be implemented using Python and will be based on the MediaPipe hand tracking and detection model. The use of MediaPipe for hand gesture recognition provides a robust and efficient solution for detecting and classifying hand gestures in real-time video streams, making it suitable for a wide range of applications.

CHAPTER 3

CODE EXPLANATION

Let's import packages for this project cv2 is opencv and mediapipe is well mediapipe, mp is the short notation that we are using here for mediapipe so anywhere in this code you see mp it is referring to mediapipe ,landmark_pb2 is special class imported from mediapipe it will help us find landmark coordinates for hand features like thumb tip,index finger bottom etc , math class is used to derive the formula to find the distance between two coordinates, win32api is used for cursor movement and click automation

Then we are calling the hands class from mediapipe.solutions. This is the class that will help us recognize all 20 geometric points on the hand. After calling this class let's move on to the OpenCV part, the first step is to take the input from the video capturing device. Of course, there are many devices that you can use for this but for simplicity. I am using the default webcam of the laptop.

Following steps are being performed here

- in the with condition we are calling mediapipe.hands.Hands class with 2 parameters , first is minimum detection confidence, in simple words this is the sensitivity of hand feature detection, values can vary depending on lighting , camera quality, busynes of background etc and same goes for second parameter which is for minimum tracking sensitivity ,value is stored in hands variable but its a bunch or data and we need specific data points that we will extract afterwards.
- Then we are running a while loop that runs infinitely until a specific condition is met (pressing q button) after this the camera

will close and program will be completed, incoming data from webcam is stored in frame variable using .read() class of opencv

- then we have to change the format of image before we process it because opencv work on BGR format (Blue Green Red) but mediapipe works on RGB (Red Green Blue) format so after taking input using opencv we need to change format to be used by mediapipe,
- then we flip the video extract the height and width of frame
- the image is processed and output is stored in results variable
- after this we can convert RGB back to BGR as we have the results variable for all the tasks now
- Now in results variable we will check if there is any multi_hand_landmarks available basically we are checking for frames with hands in it
- then we will iterate on results.multi_hand_landmarks value (landmark type and coordinate ratio)
- then by using .DrawingSpec() class from mediapipe.solutions.drawing_utils we will draw the points on hand landmark and connect them ,parameters of this class are :
    - image on which points to be drawn
    - mediapipe.solutions.hands.HAND_CONNECTIONS() method which will connect the points with line
    - mediapipe.solutions.drawing_utils.DrawingSpec() class to basically draw points with will again have these paramters
        - color or the circle
        - thickness of the circle
        - radius of the circle

After drawing the points on hand landmarks we need to focus on the functionality of the program i.e moving the cursor and clicking with finger action, for this we need coordinates of a specific landmark like for movement of cursor we need coordinates of the index finger, for

clicking we need coordinates of index finger and thumb,soo lets see what's happening in the code

- with first if condition we are checking if there is a hand in the frame for not
- first for loop will iterate in results.multi_hand_landmark but it has multiplae points and we are only interested in coordinates
- second for loop will iterate on points or each coordinate (there are 20 coordinates on hand)
- normalizedLandmark will have the actual data for each point but its in ratio form and for opencv to recognize points we need x,y coordinate , so we are using _normalized_to_pixel_coordinates() class to convert ratio into coordinates parameters inside are normalizedx,normalizedy,framewidth,frameheight
- but these values are special data type and we need to convert it into string so that we can work on it

we want only the coordinates of the index fingertip and thumb tip so we will create if condition for each one and store their value separately in indexfingertip_x,indexfingertip_y,thumbfingertip_x, and thumbfingertip_y, also we are using win32api to move our cursor with index finger, we are not using pyautogui for this task as it was a bit slow and was causing our program to appear sluggish for automating the click we have assigned the action of pressing the index finger and thumb together just like a snap, for this, we need to calculate the distance between coordinates of the index fingertip and thumb fingertip, distance_x and Distance_y is the variable in which difference value is stored we will then compare this value to integer 5 (or any small number) to check if they are close together and if they are we will use pyautogui.click() to click.
After this, the only thing left is to display the output stream to the user we will do this using imshow() method of opencv then we will create an

exit condition for the while loop, the condition here is that the user presses key ‘q’.

CHAPTER 4

CONCLUSION AND FUTURE WORK

Overall, the virtual mouse project utilizing MediaPipe and hand gestures as inputs has the potential to be a useful and innovative technology. By using the MediaPipe framework to track and detect hand movements in real-time video streams, and then using those movements to control the cursor on a computer screen, it is possible to create a virtual mouse that can be used as an alternative to traditional input devices.

However, there are several changes that could be made to make the virtual mouse even more useful. One potential change would be to improve the accuracy and reliability of the hand gesture recognition model. This could be done by collecting a larger and more diverse dataset of hand gestures, as well as fine-tuning the model using transfer learning.

Another change that could be made is to add more advanced features to the virtual mouse. For example, the ability to recognize multiple hand gestures simultaneously could be implemented. This would allow the virtual mouse to recognize different hand gestures for different actions such as left and right click.

Another change that could be made would be to add support for more than one user. This could be done by training the model to recognize multiple hands and mapping each hand to a different cursor.

Finally, it would be a great idea to make the system more robust to different lighting conditions, background and hand sizes so that it can be used in real world scenarios.

In conclusion, the virtual mouse project using MediaPipe and hand gestures as inputs has the potential to be a useful and innovative technology. By making changes such as improving the accuracy and

reliability of the hand gesture recognition model and adding more advanced features, the virtual mouse can become even more useful and versatile.

REFERENCES

STUDY MATERIAL :

https://www.udemy.com/course/machine-learning-using-python-starttech/learn/lecture/33828664?src=sac&kw=machine+learning#overview

https://www.youtube.com/watch?v=jAFfkR0U5Bo

https://google.github.io/mediapipe/solutions/hands