Hand Gesture Controlled Virtual Calculator

Chinmayee Pendyala (cxp65870@ucmo.edu)
Harshita Rudrararju (lxr63710@ucmo.edu)
Manasa Vathumilli (mxv54670@ucmo.edu)
Nakshathra Racha (nxr50720@ucmo.edu)

***Abstract*—**

Hand gestures are an important form of non-verbal communication that include movement of hands to convey messages. In day-to-day life, we see varied use of hand gestures: traffic personnel use gesture in a traffic control environment, car drivers use hand gestures for coordination, and manual features of American Sign Language include various hand signals etc. Over the last few years, hand gesture recognition has become one of the most significant research areas in computer vision as this is crucial for human computer interaction (HCI). Therefore, for the final project, we have worked to develop a hand gesture recognition system that can recognize nineteen different hand gestures, and we have further used the hand gesture recognition system to operate a three digit virtual calculator that can perform various arithmetic operations such as addition, subtraction, multiplication, division and modulus etc.

The data for the nineteen hand gestures has been collected using MediaPipe machine learning (ML) framework developed by Google. MediaPipe includes palm detection and hand landmark models that can detect hands and can identify 21 hand-knuckle (landmark) coordinates inside the detected hand regions. The coordinates of hand landmarks for different gestures constitute the data used for developing the hand gesture recognition system. Next, this data has been processed to extract features, which subsequently have been used to train a random forest classifier for hand gesture recognition. Finally, the hand gesture recognition model has been used to operate a three digit virtual calculator developed using functions from OpenCV library.

The primary component of the project has been data collection and data processing for feature extraction. The secondary component is model performance analysis and real-time demonstration of OpenCV calculator operation using hand gesture recognition system. The motivation behind selecting this topic was to explore and learn various concepts from computer vision, while working on the final project.

Index Terms — Hand gesture recognition, MediaPipe, Hand landmark points, OpenCV calculator, Random forest classifier

I. INTRODUCTION

With the development of computing technology, computers are becoming an essential part of our lives. The growing demand for computing devices has increased the need for human computer interaction. Besides, the hand gesture interaction is a well-known technique, which can be applied in multiple scenarios, like sign language translation [1], sports [2], human-robot interaction [3], and human-machine interaction [4] etc. Compared to a mouse or keyboard, any vision-based interface is more convenient, useful, and natural because gestures are intuitive. Therefore, hand gesture recognition is gradually becoming popular in the research community.

The methods to realize hand gesture recognition are typically categorized as follows: (1) using glove-based wearable devices [5], (2) using 3-dimensional locations of hand key points [6]-[7], and (3) using raw visual data [8]-[9]. The method of using glove-based wearable devices uses glove markers to extract the input data to realize hand gesture recognition [5]. The second method yields high hand gesture recognition accuracy by extracting and tracking hand-key points [7]. The third method, although only needs an image capturing sensor like a camera, usually relies on complex deep learning techniques for feature extraction and processing [9].

For this work, our primary goal is to implement a hand-gesture recognition system that can recognize nineteen different hand gestures. We intend to utilize Google's MediaPipe ML framework for hand gesture data collection (hand landmarks for different gestures) [7]. Next, we process the data for feature extraction and use the extracted features to train a random forest classifier for hand gesture recognition. Our end goal, in this project, is to use the hand gesture recognition system to operate a three digit virtual calculator that can perform various arithmetic operations such as addition, subtraction, multiplication, division and modulus etc. Through this work, we attempt to demonstrate a simple but effective ML pipeline developed for hand gesture recognition and its subsequent application.

II. MOTIVATION

The primary reason for developing a virtual calculator that responds to hand gestures is to give people who might find it challenging to use traditional physical calculators or keyboards an alternative input option. The purpose of this gadget is to provide accessibility and convenience, particularly for those who are physically limited or have disabilities or injuries that prevent them from using their hands normally. Furthermore, the creation of a virtual calculator that responds to hand gestures is a creative application of technology that could pave the way for future developments in gesture recognition and hands-free control of other devices.

Github source code link: https://github.com/Manasav17/Final_project

III. Main Contributions & Objectives

The major goal of developing a hand gesture-controlled virtual calculator is to offer an accessible, practical, and creative alternate input method for doing computations. For persons who would find it challenging to use conventional physical calculators or keyboards, such as those with disabilities or injuries, the device intends to enhance the user experience. Users can conduct calculations more quickly and effectively, particularly in circumstances where a hardware calculator may not be available or practical, by permitting hand gestures to control the virtual calculator. The overall goal of a hand gesture-controlled virtual calculator is to increase technology's usability, accessibility, and innovation while also enhancing the user experience for people who might find it challenging to use conventional input techniques.

IV. Related Work

Hand gesture recognition has emerged as one of the most important research areas in computer vision as it is critical for human-computer interaction. We see diverse applications of hand gestures in everyday life. Following are few works related to hand gesture recognition:

1) A Hierarchical Hand Gesture Recognition Framework for Sports Referee Training-Based EMG and Accelerometer Sensors.
2) A kinect-based gesture recognition approach for a natural human robot interface.
3) Glove-based hand gesture recognition sign language translator using capacitive touch sensor.
4) On-device Realtime Hand Tracking.
5) Hand Gesture Recognition using Infrared Imagery Provided by Leap Motion Controller.
6) Real-Time Hand Gesture Spotting and Recognition Using RGB-D Camera and 3D Convolutional Neural Network.

V. Hand Gesture Controlled Calculator Framework

The hand gesture controlled calculator framework is as shown in Fig. 1. This framework has been developed based on various works discussed in references [7], [10]-[13]. The framework mainly comprises of three steps with steps 1 and 3 having two sub-steps:

7) Data collection/feature extraction
   a) Quantify hand gesture using MediaPipe
   b) Feature engineering
8) Hand gesture recognition system
9) Data collection/real-time demonstration of calculator operation
   a) Quantify hand gesture using MediaPipe
   b) Hand gesture controlled calculator

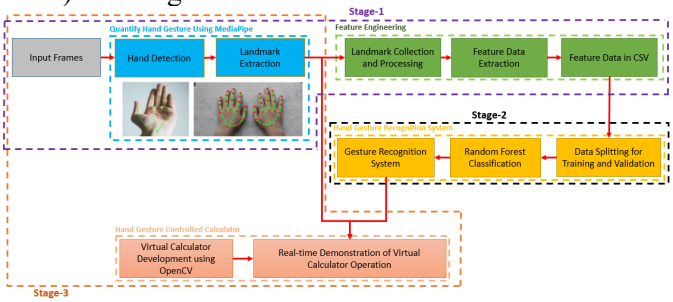

Fig. 1 Hand gesture controlled calculator framework

1) *Data Collection/Feature Extraction*

a) *Quantify Hand Gesture Using Mediapipe:*

In this sub-step, first, videos of hand gestures are captured using a computer webcam. The frames in the captured videos are processed using MediaPipe ML framework to detect hands and draw hand landmarks (hand-key points) on the detected hands. The coordinates of hand landmarks for different gestures are then processed for feature extraction.

b) *Feature Engineering:*

MediaPipe generates normalized x and y coordinates of hand landmarks (normalized with respect to frame width and height) [10]. These normalized coordinates cannot be used as is for training the gesture recognition model because the coordinates may differ for the same hand gesture at different hand positions in various captured frames (also based on how far the hand is from the camera – depth). Hence, in this sub-step, the normalized landmark coordinates from MediaPipe for different hand gestures are processed to extract features for model development and subsequently, the extracted features are saved to a .csv file.

2) *Hand Gesture Recognition System:*

In this step, the feature datasets along with the hand gesture label datasets are split into training and test data. The training dataset is then used to train a random forest classifier for hand gesture recognition and the classification model is subsequently tested by the test dataset. By the end of this step, a hand gesture recognition model with high recognition accuracy is obtained for further use.

3) *Data Collection/Real-Time Demonstration Of Calculator Operation*

a) *Quantify Hand Gesture Using Mediapipe:*

Similar to sub-step:1(a), MediaPipe calculates hand landmarks from the live feed video frames and passes it further for hand gesture recognition and real-time demonstration of calculator operation.

Github source code link: https://github.com/Manasav17/Final_project

b) Hand Gesture Controlled Calculator

In this sub-step, a virtual three digit calculator is created using various functions from OpenCV library [14]. Next, the hand gesture recognition model, developed in the previous step, processes input hand landmarks and generates gestures real-time to enter various entries (first number entry followed by arithmetic operator and then, second number entry) into the virtual calculator and subsequently, results from the calculator operations are obtained.

VI. Data Collection/Feature Extraction

A. Quantify Hand Gesture Using MediaPipe

MediaPipe is an open-source highly customizable ML framework, which typically is used for developing ML applications that use live and streaming media. MediaPipe offers ML solutions for various computer vision problems such as face detection, hand tracking, pose estimation, object detection, motion tracking, and iris estimation etc. [15]. For this work, we have utilized MediaPipe Hands, specifically its palm detection and hand landmark models [10]. The palm detection model, as its name suggests, can detect initial hand locations in a static image or in a video frame. This model has been developed using an approach called 'Single Shot MultiBox Detector' (SSD) that is based on a feed-forward convolutional network [10]. After the hand detection, the hand landmark model employs regression for key point localization of 21 3D hand-knuckle coordinates (hand landmarks) inside the detected hand regions [10]. Fig. 2 shows 21 hand-key points (landmarks) that can be identified by the hand landmark model.

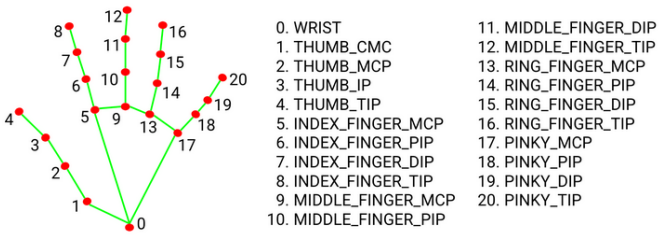

Fig. 2 21 hand landmarks

There are various configuration options that are to be set prior to using MediaPipe for hand landmark detection and tracking. The 'STATIC_IMAGE_MODE' option, in step-1, is set to 'True' so that MediaPipe treats the webcam feed as a batch of static images during data collection for feature extraction. For real-time hand tracking (in step-3 for real-time demonstration of calculator operation), this option is set to 'False'. The option 'MAX_NUM_HANDS' is always set to '1', so that MediaPipe at any time can detect maximum one hand (either left or right). 'MIN_DETECTION_CONFIDENCE' for the hand detection model is set at 0.8 and a default value of 0.5 is used for 'MIN_TRACKING_CONFIDENCE' for the landmark-tracking model [10].

In this sub-step, first, videos of hand gestures are captured using a computer webcam. The frames of the captured videos are input to MediaPipe Hands. MediaPipe treats the video frames as static images as discussed before and identifies hand positions in the frame/image using the palm detection model. Next, the hand landmark model identifies 21 hand-key points and renders the result on to the image/frame as shown in Fig. 3. The outputs from this sub-step are normalized x and y coordinates of hand landmarks for different hand gestures (normalized with respect to frame width and height), which are subsequently processed in the next sub-step, for feature extraction.

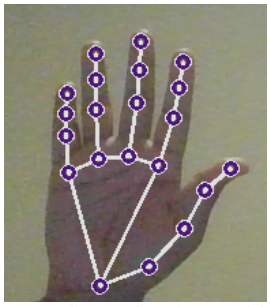

Fig. 3 MediaPipe results rendered on a video frame/image

B. Feature Engineering

For a hand gesture, MediaPipe generates normalized x and y coordinates of hand landmarks (normalized with respect to frame width and height) [10]. These normalized coordinates cannot be used for training the gesture recognition model because the coordinates may differ for the same hand gesture at different hand positions in various captured frames (also based on how far the hand is from the camera – depth). The following procedures are then followed to generate features specific to different gestures [11]-[12]:

For a hand gesture:

- Multiply normalized x-coordinates with frame width and y-coordinates with frame height to obtain actual x and y coordinates for all the hand landmarks.
- Obtain relative x and y coordinates of all hand landmarks with respect to wrist landmark coordinates.
- Arrange relative coordinates of all hand landmarks in a 1-D array.
- Normalize the elements of the 1D-array using the maximum absolute coordinate value in the 1D-array.

- Create a label for the hand gesture and append it to the beginning of the 1D-array with normalized coordinates.

- Repeat the above steps for the hand gesture at different hand positions in various captured frames until there are enough data samples collected for the hand gesture.

The above steps are then followed for nineteen different hand gestures for creating gesture-specific feature vectors. Fig. 4 shows the nineteen hand gestures and their corresponding labels used for developing the hand gesture recognition model. For this work, we have created around 80-100 feature vectors for each hand gesture (using both left and right hands at different hand positions and depth in various frames). Fig. 5 shows a flow-chart showing steps to be followed for feature extraction. Lastly, these feature vectors along with the hand gesture labels are saved in a .csv file for further processing in the subsequent steps.

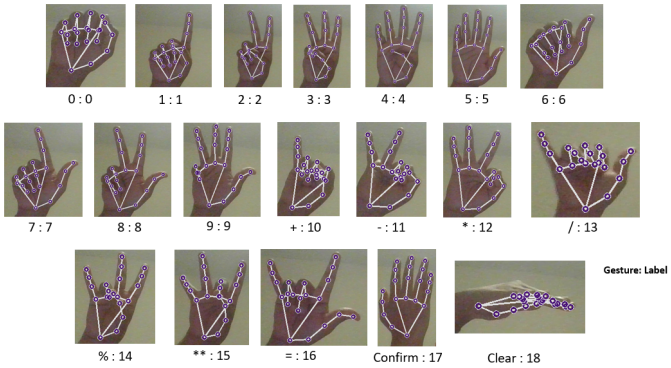

Fig. 4 Nineteen hand gestures with their corresponding labels

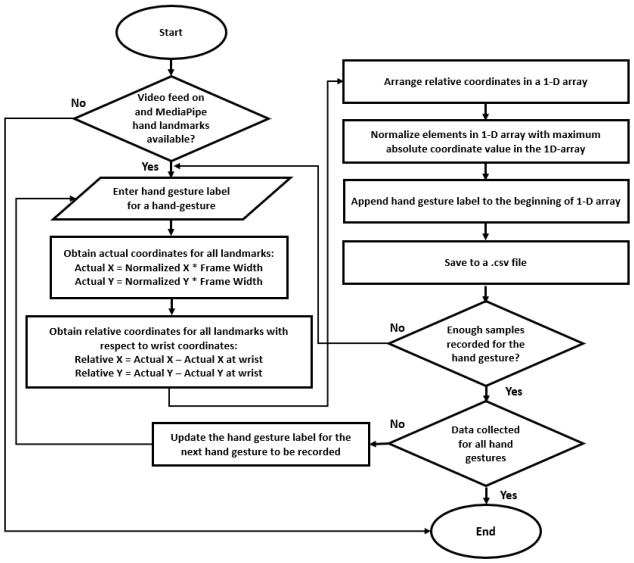

Fig. 5 Flow-chart showing steps to perform feature extraction from MediaPipe landmark coordinates for a hand gesture

VII. HAND GESTURE RECOGNITION SYSTEM

In this step, first, the feature and label datasets, saved in the .csv file during feature extraction, are imported and split into train and test data for hand gesture recognition model development. For this work, as mentioned before, there are between 80-100 data vectors collected for each of the hand gestures and care has been taken, while collecting samples, to keep the dataset balanced. Next, the data consisting of feature and label information is split into train and test data at 7:3 ratio.

During gesture recognition model development, initially various ML models including random forest and support vector classifiers were tested. References [11]-[12] suggested using feed-forward neural networks. However, high classification/recognition accuracy was already observed while using simpler random forest classifiers. Thus, for this work, random forest classifiers have been selected for hand gesture recognition model development.

The training data is then used to train a random forest classifier with 10-fold cross validation employed during training. Subsequently, the model is tested by the test data and saved for use during real-time gesture recognition and calculator operation. Results from training and testing including confusion matrix have been discussed in the subsequent sections.

VIII. DATA COLLECTION/REAL-TIME DEMONSTRATION OF CALCULATOR OPERATION

A. Quantify Hand Gesture Using MediaPipe

Similar to data collection before feature extraction in step-1(a), MediaPipe, in this sub-step, calculates hand landmarks from the live feed video frames and passes it to the hand gesture recognition model for real-time gesture recognition. As discussed before, the 'STATIC_IMAGE_MODE' option should be set to 'False' so that MediaPipe treats the webcam feed as a video stream for real-time gesture recognition.

B. Hand Gesture Controlled Calculator

In this sub-step, development and real-time demonstration of a three digit virtual calculator is discussed.

To show the effectiveness of the hand gesture recognition system, a three digit virtual calculator, operated by gestures identified real-time, has been developed using various OpenCV functions [13]-[14]. The number of frames and frames per second (fps) values in the live video feed, captured through the computer

Github source code link: https://github.com/Manasav17/Final_project

webcam, are tracked, and various calculator operations are timed accordingly and performed based on the gestures identified real-time. The user of the calculator can show various gestures to the webcam to enter various entries, first a number entry followed by an arithmetic operator and then, another number entry. The backend script waits for a few seconds between various entries and the user has options to confirm or clear (reset) the entries. Once, all the entries are recorded and confirmed by the user, backend calculation takes place and the results from the calculator operation appears on the live feed in just few seconds. Next, we present steps for virtual calculator operation in real-time:

- If the live video feed is on and real-time gesture recognition results are available, then 'Calculator ready' message appears for the first three seconds as shown in Fig. 6. If the user moves the hand (left or right) away from the webcam at any stage, then the calculator resets.

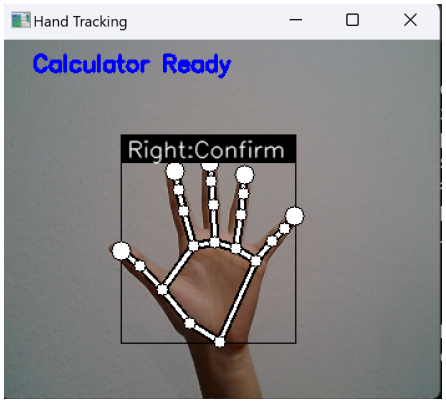

Fig. 6 Initialize virtual OpenCV calculator

- Next, 'Enter the first number' message appears for the next two seconds as shown in Fig. 7. The user then needs to show gestures to enter the first number entry into the calculator.

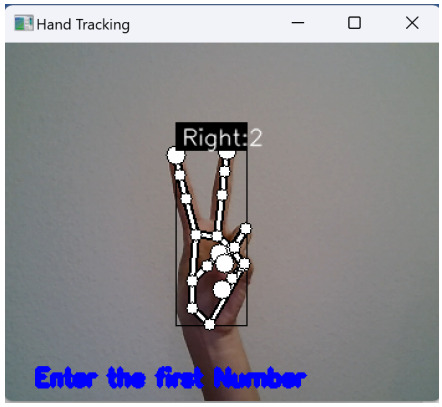

Fig. 7 Enter the first number entry

- The user has a total of nine seconds to enter the first number entry (three seconds for each digit and hence, maximum of three digits). For a digit entry, the user needs to show a gesture for three seconds for it to be recorded by the calculator. Once a digit entry is confirmed, it will appear on the screen as shown in Fig. 8. The user then shows a gesture for the next digit entry and so on, until all three single digit entries are recorded by the end of nine seconds. The user has an option to enter a single/two digit entry as the first number into the calculator and this can be done by showing 'Confirm' gesture after the single/two digit entry is recorded (before the end of nine seconds). The user can also clear an entry by showing 'Clear' gesture and then, can re-enter the digits. Once the first number entry is confirmed, it appears on the live feed as shown in Fig. 9.

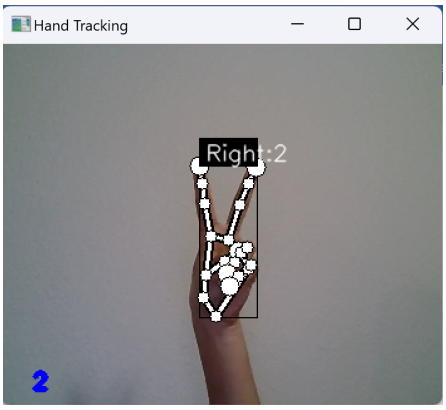

Fig. 8 '2' entered as the first digit entry

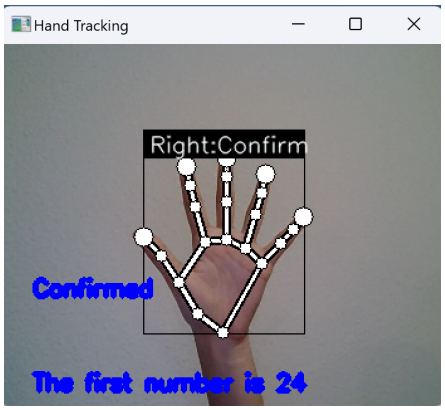

Fig. 9 '24' entered as the first number entry

- Next, the user needs to enter an operator for arithmetic operation after the message 'Enter the operator' appears. The user has three seconds total to show a gesture to enter the operator (needs to show gesture for two seconds at least) and Fig. 10 shows the confirmed operator entry. The user also has an option of clearing operator entry by showing 'Clear' gesture (within one second after the first two seconds to record the operator and hence, before the end of three seconds) and then, can re-enter the operator.

Github source code link: https://github.com/Manasav17/Final_project

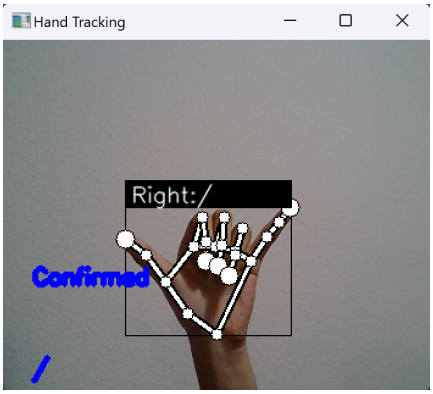

Fig. 10 '/' operator entered

- Likewise, the user can enter the second number entry after the message 'Enter the second number' appears. Similar to the first number entry, the user has a total of nine seconds to enter the second number entry (three seconds for each digit and hence, maximum of three digits). Fig. 11 shows the second number entered for the demonstration.

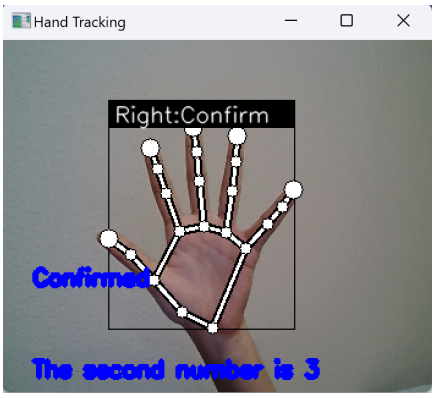

Fig. 11 '3' entered as the second number entry

- Within the next few seconds, the result appears on the screen as shown in Fig. 12. The result stays on the screen for the next nine seconds.

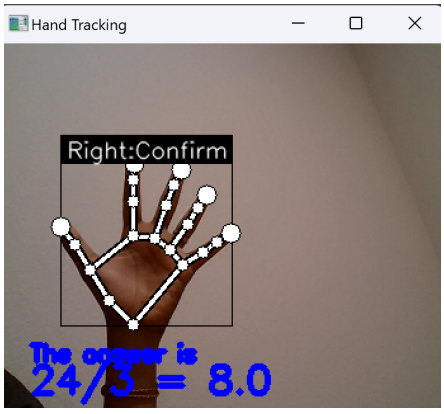

Fig. 12 Result from calculator operation

- Next, a message 'Reset calculator?' appears and the user can reset the calculator by showing 'Clear' gesture. Otherwise, the user may move the hand away from the webcam and show again to reset the calculator.

- Some error checking options have also been included in the virtual calculator. For e.g., if the user enters an operator '+' as a number entry, then the result appears as 'NA'. Similarly, if a number is entered in place of an operator, then the user sees the result as 'NA'.

Lastly, the calculator operations are timed by tracking the number of frames and fps values in the live video feed. The user can make simple modifications to the backend script to change the time allotted for different calculator operations.

IX. Results & Analysis

In this section, training and test results obtained during hand gesture recognition model development are presented and discussed.

As mentioned before, almost 80-100 data vectors were collected for each of the hand gestures during feature extraction and there were a total of 1751 data vectors with feature and gesture label information. The data was split into training and test data at 7:3 ratio for gesture recognition model development. Using the training data, a random forest classifier was trained with 10-fold cross validation and a high average cross validation score of 0.9967 was observed. The random forest classifier was then tested using the test data set and a test accuracy of 1.0 was observed. Fig. 13 and Fig. 14 show the confusion matrix and test statistics obtained. As can be deduced from the confusion matrix, all the gestures were predicted correctly.

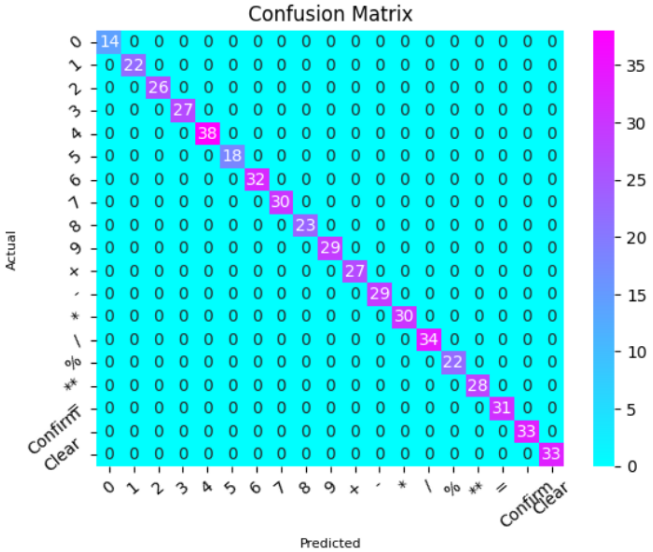

Fig. 13 Confusion matrix from the testing of hand gesture recognition model

Github source code link: https://github.com/Manasav17/Final_project

Test Statistics:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	14
1	1.00	1.00	1.00	22
2	1.00	1.00	1.00	26
3	1.00	1.00	1.00	27
4	1.00	1.00	1.00	38
5	1.00	1.00	1.00	18
6	1.00	1.00	1.00	32
7	1.00	1.00	1.00	30
8	1.00	1.00	1.00	23
9	1.00	1.00	1.00	29
10	1.00	1.00	1.00	27
11	1.00	1.00	1.00	29
12	1.00	1.00	1.00	30
13	1.00	1.00	1.00	34
14	1.00	1.00	1.00	22
15	1.00	1.00	1.00	28
16	1.00	1.00	1.00	31
17	1.00	1.00	1.00	33
18	1.00	1.00	1.00	33
accuracy			1.00	526
macro avg	1.00	1.00	1.00	526
weighted avg	1.00	1.00	1.00	526

Testing Accuracy: 1.0

Fig. 14 Test statistics from the testing of hand gesture recognition model

Before, when developing the model with less number of data samples for gestures, we would observe some instances of misclassifications between gestures '1' and '7', '3' and '9', '4' and '5', and '5' and 'Confirm', probably due to the resemblance of gestures. However, as we kept on collecting and using more gesture data (while accounting for different hand positions, camera angles, lighting conditions and distance/depth of hands from the webcam) for model development, the model accuracy improved significantly. Consequently, better results were obtained during real-time gesture recognition.

Again, as discussed in previous sections, our end goal was to demonstrate the capability of the hand gesture recognition system by using it to operate a three digit virtual calculator in real-time. We were able to operate the virtual calculator seamlessly in real-time using the robust hand gesture recognition model developed during the course of this project.

X. LIMITATIONS

As mentioned in the results section, the accuracy of the hand gesture recognition system was observed to be sensitive to the number of data samples used for the ML model development. With less number of data samples, the model would perform poorly. But, as we collected and used more data, the model performance improved significantly. That said, what we also have observed is, having a lot of data samples for similar hand positions/depth/camera angles/lighting conditions does not help with improving the accuracy. It is also important to include data collected for different camera angles, hand positions, lighting conditions and hand depth to improve the gesture recognition model performance. Essentially, the point we are trying to make is that the gesture recognition model (overall approach) has been observed to be very sensitive to the position of the camera that is used for capturing video frames, surrounding lighting conditions as well as different hand positions as captured by the camera.

Another limitation of the hand gesture recognition approach, discussed here, is that this approach at any time can only detect gestures shown by one hand. That sort of limits the number of gestures that can be tracked by the gesture recognition system presented here.

XI. FUTURE WORK

The hand gesture recognition approach presented in this project can only track gestures made by one hand at a time. This, in a way, limits the number of gestures that can be detected/recognized using this approach. Going forward, it would be interesting to see if multi-handedness could be included into the approach presented here. That could lead to the development of a really powerful hand gesture recognition approach that can recognize multiple hand gestures at any time and also a wide variety of hand gestures.

XII. CONCLUSION

This work discusses the development of a hand gesture recognition system that recognizes at least nineteen different hand gestures and presents its capability to operate a three digit virtual calculator seamlessly by identifying gestures real-time. The work started with collection of coordinates of hand-key points (also called hand landmarks) for different hand gestures using Google's MediaPipe ML framework, which were subsequently processed for feature extraction. Next, the gesture datasets consisting of gesture label and feature information were used to develop a hand gesture recognition system that showed high testing accuracy during model development. Lastly, the hand gesture recognition model was used for successful demonstration of real-time gesture recognition as well as for the real-time operation of a virtual calculator developed during the project.

REFERENCES

[1] Cheok, M. J., Omar, Z., and Jaward, M. H. (2019). A review of hand gesture and sign language recognition techniques. Int. J. Mach. Learn. Cybern. 10, 131–153. doi: 10.1007/s13042-017-0705-5

[2] T. -Y. Pan, W. -L. Tsai, C. -Y. Chang, C. -W. Yeh and M. -C. Hu, "A Hierarchical Hand Gesture Recognition Framework for Sports Referee Training-Based EMG and Accelerometer

Sensors," in *IEEE Transactions on Cybernetics*, vol. 52, no. 5, pp. 3172-3183, May 2022, doi: 10.1109/TCYB.2020.3007173.

[3] Cicirelli, G., Attolico, C., Guaragnella, C., and D'Orazio, T. (2015). A kinect-based gesture recognition approach for a natural human robot interface. Int. J. Adv. Robot. Syst. 12:22. doi: 10.5772/59974

[4] Haria, A., Subramanian, A., Asokkumar, N., Poddar, S., and Nayak, J. S. (2017). Hand gesture recognition for human computer interaction. Proc. Comput. Sci. 115, 367–374. doi: 10.1016/j.procs.2017.09.092

[5] K. S. Abhishek, L. C. F. Qubeley, and D. Ho. Glove-based hand gesture recognition sign language translator using capacitive touch sensor. In 2016 IEEE International Conference on Electron Devices and Solid-State Circuits (EDSSC), pages 334–337, Aug. 2016.

[6] R. Wen, L. Yang, C.-K. Chui, K.-B. Lim, and S. Chang. Intraoperative Visual Guidance and Control Interface for Augmented Reality Robotic Surgery. In 2010 8th IEEE International Conference on Control and Automation, ICCA 2010, pages 947–952, July 2010.

[7] F. Zhang, V. Bazarevsky, A. Vakunov, A. Tkachenka, G. Sung, C.L. Chang, M. Grundmann. MediaPipe Hands: On-device Real-time Hand Tracking. CVPR Workshop on Computer Vision for Augmented and Virtual Reality, Seattle, WA, USA, 2020. arXiv:2006.10214

[8] T. Mantecón, C.R. del Blanco, F. Jaureguizar, N. García, "Hand Gesture Recognition using Infrared Imagery Provided by Leap Motion Controller", Int. Conf. on Advanced Concepts for Intelligent Vision Systems, ACIVS 2016, Lecce, Italy, pp. 47–57, 24–27 Oct. 2016. (doi: 10.1007/978–3–319–48680–2_5)

[9] Tran, D.-S.; Ho, N.-H.; Yang, H.-J.; Baek, E.-T.; Kim, S.-H.; Lee, G. Real-Time Hand Gesture Spotting and Recognition Using RGB-D Camera and 3D Convolutional Neural Network. *Appl. Sci.* **2020**, *10*, 722. https://doi.org/10.3390/app10020722

[10] MediaPipe Hands [Online], Available from: https://google.github.io/mediapipe/solutions/hands.html [Accessed 02 Nov 2022].

[11] Hand gesture recognition using MediaPipe [Online], Available from: https://github.com/Kazuhito00/hand-gesture-recognition-using-mediapipe [Accessed 07 Nov 2022].

[12] Hand gesture recognition using MediaPipe [Online], Available from: https://github.com/kinivi/hand-gesture-recognition-mediapipe [Accessed 07 Nov 2022].

[13] Gesture controlled OpenCV calculator [Online], Available from: https://github.com/rishabh-arya/Gesture-controlled-opencv-calculator [Accessed 15 Nov 2022].

[14] OpenCV-Python Tutorials [Online], Available from: https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html [Accessed 15 Nov 2022]

[15] MediaPipe [Online], Available from: https://google.github.io/mediapipe/ [Accessed 02 Nov 2022].

Github source code link: https://github.com/Manasav17/Final_project