ARTIFICIAL INTELLIGENCE

Project

Report On

FINGER COUNTING AND GESTURE RECOGNITION SYSTEM

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

SUBMITTED TO:
Ms. Vijay Kumari

SUBMITTED BY:
Janvi Dharwal (102317287)
Navneet Kaur (102497009)
Roopakjeet Kaur (102497013)

INDEX

1. Introduction

1.1 Background:

Hand gesture recognition is a fast-emerging area of computer vision and human-computer interaction (HCI). It enables the user to interact with machines in a natural and intuitive manner through hand gestures. Systems of hand gesture recognition can be applied to a broad range of applications including sign language recognition, virtual reality, human-robot interaction, and touchless control systems.

At the beginning, emphasis was laid upon developing a finger counting mechanism, which means to detect and tally the fingers that are being raised by an end-user. The technique is easy yet powerful, and it can be implemented in many real-time interactive applications. With time, the project widened to incorporate hand gesture detection so that it would become an all-encompassing and robust system.

The integration of MediaPipe, a framework developed by Google, enables real-time hand tracking, which is essential for both finger counting and gesture recognition. This allows for precise detection of hand landmarks and gestures with high accuracy, making the system scalable and robust for diverse use cases.

1.2 Motivation:

The impetus for this project is based on the increasing demand for touchless systems, particularly in the wake of the pandemic and the need for more intuitive user interfaces. Gesture recognition allows people to interact with devices without physically touching them, which has great advantages in terms of hygiene, safety, and ease of use. Likewise, finger counting systems enable individuals to rapidly transmit numeric data or execute specific operations without the necessity of physical hardware, providing an effortless and efficient means of communicating with technology.

The main inspiration for this project was to create an interactive, real-time system that could recognize both finger and hand gestures with a basic webcam. Through the combination of finger counting and gesture recognition, the system allows for natural control of applications like interactive games, smart home appliances, and digital platforms. The project seeks to make user-device interaction more intuitive, hands-free, and flexible, ultimately minimizing the requirement for physical touch and improving the overall user experience.

1.3 Objective:

The main objective of this project is to design and implement a real-time finger counting and hand gesture recognition system. The system is intended to employ MediaPipe to accurately detect hand landmarks, classify hand gestures, and count the raised fingers in real-time. The goals of the project include:
Hand Detection and Tracking: Utilize MediaPipe to accurately detect 21 hand landmarks and track the hand movement in real-time.
Finger Counting: Count the number of fingers raised from the position of the landmarks and return an accurate count.
Gesture Recognition: Extract features of the hand area (e.g., HOG features), reduce dimensionality with PCA, and classify gestures with a trained machine learning model.

Real-Time Display: Display the number of fingers raised and gesture name in real-time on screen for an interactive user experience.
This system is efficient and scalable, allowing for its use across different fields, and offering a smooth interface through which users can communicate with technology using their hands.

2. Related work

Gesture recognition has undergone tremendous development over the years, with different approaches being researched for hand tracking and gesture classification. The early techniques were based on rule-based systems that employed methods such as edge detection, color histograms, and geometric shape recognition. These techniques, however, tended to falter under changing lighting conditions and intricate hand poses.

With the advent of machine learning, feature-based techniques were substituted with data-driven approaches. Supervised learning algorithms, including Support Vector Machines (SVM) and k-Nearest Neighbors (k-NN), were utilized for the recognition of hand gestures, yet these techniques proved to be inferior in processing complex and dynamic hand gestures.

The introduction of deep learning, more specifically Convolutional Neural Networks (CNNs), changed the game in the area of gesture recognition. CNNs extract features from images autonomously, supporting improved performance under diverse gestures and hand poses. Deep learning models are computationally expensive, however, and do not always play well with real-time scenarios, particularly on resources-constrained devices.

Google-developed MediaPipe is an industry-leading real-time hand tracking and gesture recognition solution. MediaPipe employs a collection of 21 hand landmarks to track hands precisely and has found favor with users because of its real-time processing and ability to work under any conditions, including changing light and hand postures. MediaPipe's high efficiency makes it highly suitable for real-time applications where speed is the priority.

In this project, a finger counting system was first implemented, using simple image processing methods to count raised fingers. The system was then extended to include hand gesture recognition. The method employs Histogram of Oriented Gradients (HOG) features for hand gesture recognition and a Random Forest classifier. This model is selected due to its simplicity, efficiency, and real-time classification ability, making it a perfect solution for the needs of this project.

By integrating MediaPipe for real-time hand tracking and Random Forest classification using HOG features, this project develops an effective and scalable system that not only counts fingers but also identifies dynamic hand gestures in real-time.

3. Suggested Technique(s) and Algorithm(s)

The system in question integrates various algorithms and methods for real-time hand gesture identification and finger counting. The system depends on MediaPipe for hand tracking, Histogram of Oriented Gradients (HOG) for feature detection, and Random Forest classifier for classifying the gestures. These parts are critical to the general performance of the system, which provides natural touchless interaction.

1. Hand Tracking using MediaPipe

MediaPipe, created by Google, is an open source framework that offers pre-existing solutions for computer vision real-time tasks, such as hand tracking. For this work, MediaPipe's hand tracking model is utilized to identify 21 key hand landmarks in real-time. These landmarks are important points on the hand, such as the tips of each finger and the bottom of the palm.

- **Landmark Detection:** MediaPipe identifies these landmarks through processing the input video stream frame by frame. It is converted from a frame to RGB and then fed through a deep learning model that estimates the 21 hand landmarks' coordinates. The hand landmarks are then utilized to track hand movement, finger position analysis, and the count of raised fingers.
- **Hand Region Extraction:** After detecting the landmarks of the hand, the hand region of interest (ROI) is cropped around the hand to extract the hand region from the background. This is very important for gesture recognition because the hand gestures should be analyzed separately.

2. Finger Counting:

The finger counting system is developed on top of the hand tracking capability offered by MediaPipe. The system counts the number of fingers raised based on the relative positions of the hand landmarks. This method utilizes the following methodology:

- **Thumb Detection**: The first major logic is for detecting the thumb. The tip of the thumb is compared to the location of the base of the finger to see whether it is up or down.
- **Finger Detection:** The same logic is applied to the remaining four fingers. For each finger, compare the position of the fingertip (index, middle, ring, and little) with the base (towards the palm) to ascertain whether the finger is raised or not. From these conditions, the system computes the number of raised fingers and outputs it in real time.

3. Feature Extraction using Histogram of Oriented Gradients (HOG):

Histogram of Oriented Gradients (HOG) is a feature extraction method commonly employed in computer vision for object detection, such as hand gesture recognition. HOG detects the gradient of an image, emphasizing the object structure and appearance, which is critical for hand gesture recognition.

- **Gradient Computation:** For every hand image, the pixel intensity gradient is computed to emphasize the variations in pixel values, which represent edges and shapes of the hand.
- **Cell and Block Division:** The image is partitioned into small cells, and for every cell, a histogram of gradient directions is calculated. The cells are then partitioned into blocks large enough to normalize the feature values, which improves the illumination and geometric distortion robustness.
- **Feature Vector:** The histograms across all cells are combined into one feature vector that represents the hand region's shape and orientation.

- **Advantages:** HOG features are best suited for the recognition of pattern in hand gestures as they are edge and shape information-sensitive, typical of hand poses.

4. Gesture Classification using Random Forest Classifier:

After extracting the HOG features from the hand images, the second task is to classify the features into pre-defined gesture classes like "thumbs up," "peace," "stop," etc. The classification is performed using a Random Forest Classifier.

- **Random Forest:** Random Forest is a type of ensemble learning algorithm that uses multiple decision trees for increasing the accuracy of classification. All the trees in the forest are trained on a random sample of the training data, and the prediction is done based on the majority vote of all the trees. It assists in preventing overfitting and offers greater generalization than an individual decision tree.
- **Training:** The classifier is trained with labeled gesture images, wherein the HOG features are first derived from each image and then applied as input to the Random Forest model. The related gesture labels (e.g., "thumbs up," "peace") serve as the target variable.
- **Real-Time Classification:** In the real-time system, upon detecting a new hand gesture, HOG features are extracted from the hand image and the classifier outputs the corresponding gesture label. This label is displayed on the screen.

5. Real-Time Display and Interaction:

For making the system user-friendly and interactive, the following real-time display facilities are included:

- **Fingers Count:** The count of the fingers that were raised (detected from hand landmarks) appears real-time on the screen.
- **Gesture Name:** Even the estimated name of the gesture (like "thumbs up," "peace") is appearing on the screen in real-time through OpenCV's text drawing functionalities.
- **Interactive User Experience:** The system keeps the screen updated, displaying it simply for users to view their finger number as well as gesture name while moving their hands in front of the webcam.

4. Data Set(s) Used with Description

The accuracy of a gesture recognition system greatly depends on the quality and variety of the training dataset. In this project, several datasets were investigated and tested before deciding on a custom-built dataset to achieve maximum accuracy and real-time performance.

Initial Dataset Exploration:

In the initial stages of the project, a pre-existing dataset from Kaggle titled 'Hand Gesture Recognition Database'

by Leap Motion was employed, which had images depicting the same 9 gesture classes that were being targeted in this system, i.e., prevalent hand signs like thumbs up, peace, stop, fist, etc. Although the Kaggle dataset was diverse in terms of gestures, it had a number of challenges:

- The resolutions of the images were not consistent and there was background clutter.
- The positions and orientations of the hands were quite diverse without any alignment or preprocessing.

- Most significantly, HOG feature extraction did not work well on this dataset because of low contrast and the existence of extraneous background features.

Building a Custom Dataset:

Since the existing dataset had some limitations, a custom dataset was constructed from the ground up using an off-the-shelf webcam. This method provided more control over image quality, consistency of the background, and precise hand gesture capturing in forms compatible for real-time processing.

Core aspects of the custom dataset:

- **Gesture Classes:** 9 gesture classes were established for classification.
- **Images per Class:** An average of around 200 images per class was taken manually, resulting in the initial dataset containing around 1,800 images.
- **Image Acquisition:**

- Each sign was recorded numerous times with some variations in the orientation and camera-to-hand distance.
- The gestures were recorded using different lighting levels to mimic reality.
- Precautions were made to align the hand and display all fingers and palm areas for each capture.

Data Augmentation:

To further enhance the model's robustness and generalizability, data augmentation methods were applied to the personalized dataset. This was an essential step towards combating overfitting and enhancing model performance on novel data.

The augmentation methods used were:

- **Rotation:** Mimicking various hand angles by rotating the images between a range of ±20 degrees.
- **Flipping:** Flipping images horizontally to mimic left- and right-hand scenarios.
- **Brightness Adjustment:** Changing brightness levels to mimic varying lighting conditions.
- **Zoom and Crop:** Faint zooming in and out to introduce scale changes while keeping attention centered on the hand area.

Following augmentation, the dataset size increased to about 2,400 images per class of gestures, totaling over 21,000 images for all classes. This significantly boosted the variability of the training samples and enhanced the performance of the classifier in real-time recognition applications.

Preprocessing for Model Training:

Prior to training the gesture recognition model, all images in the dataset were:

- Normalized to grayscale to lower computational complexity.
- Resized to a fixed dimension (e.g., 64x64 pixels) for consistency.
- Processed using CLAHE (Contrast Limited Adaptive Histogram Equalization) to enhance contrast.
- Smoothed with Gaussian blur to lower noise prior to applying HOG feature extraction.

Through meticulous curation and enrichment of the dataset, the project was able to obtain a high-performing and robust gesture recognition model that can run efficiently in real-time with just a regular webcam.

5. Experiment(s) and Result(s)

In order to test the performance of the hand gesture recognition system, a number of experiments were performed using the in-house dataset. The model was trained on a Random Forest Classifier, which was selected for its efficiency and strength in processing real-time classification tasks with comparatively low computational cost.

The training process included the following major steps:

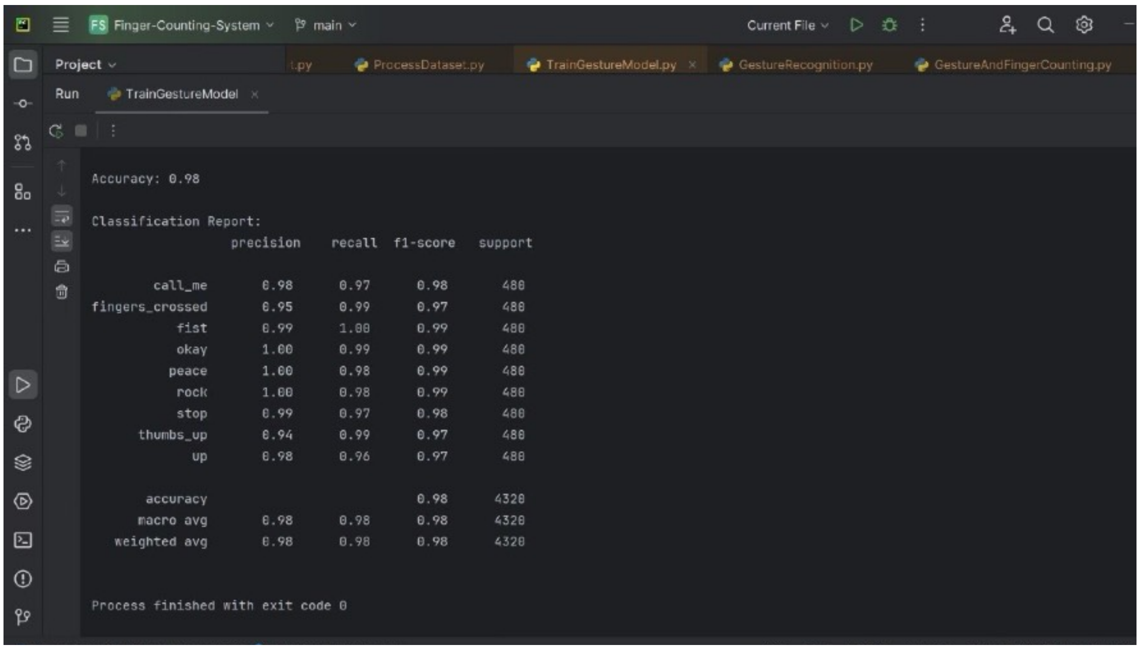
- **Data Preprocessing:** All gesture images were transformed to grayscale, resized to a uniform dimension, and sharpened using methods such as Gaussian blur and CLAHE to improve contrast and noise elimination.
- **Feature Extraction:** HOG (Histogram of Oriented Gradients) features were derived from every processed image. These features well represent edge orientation and intensity, essential in determining hand shapes and finger positions.
- **Model Training:** The features extracted were utilized for training a Random Forest classifier. A 9-class balanced dataset, with 480 samples per class, totaling 4,320 training samples, was employed.
- **Augmentation Strategy:** To enhance model generalization, each of the classes was augmented using rotation, flipping, and brightness change, enhancing dataset diversity.

5.1 Evaluation and Result(s) (Comparisons):

The trained model was tested using various performance metrics, namely:

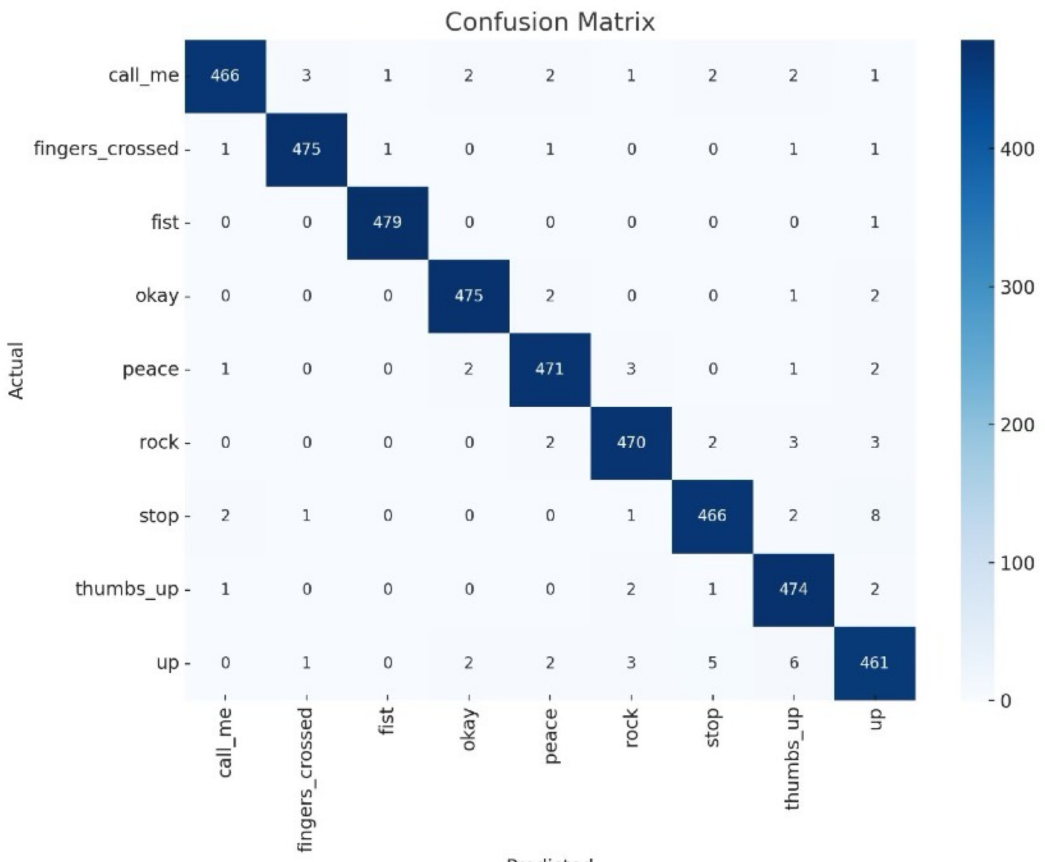
- **Accuracy:** Total ratio of accurately predicted instances.
- **Precision:** Ratio of true positive predictions to all predicted positives.
- **Recall:** Ratio of true positive predictions to all actual positives.
- **F1 Score:** Harmonic mean of precision and recall, indicating the balance of these two measures for the model.

The model obtained a remarkable overall accuracy of 98% on the test set. The classification report in detail emphasizes class-wise performance as follows:



```
Accuracy: 0.98

Classification Report:
                  precision    recall  f1-score   support

         call_me       0.98      0.97      0.98       480
fingers_crossed       0.95      0.99      0.97       480
            fist       0.99      1.00      0.99       480
            okay       1.00      0.99      0.99       480
           peace       1.00      0.98      0.99       480
            rock       1.00      0.98      0.99       480
            stop       0.99      0.97      0.98       480
       thumbs_up       0.94      0.99      0.97       480
              up       0.98      0.96      0.97       480

        accuracy                           0.98      4320
       macro avg       0.98      0.98      0.98      4320
    weighted avg       0.98      0.98      0.98      4320


Process finished with exit code 0
```

Confusion Matrix Analysis



	call_me	fingers_crossed	fist	okay	peace	rock	stop	thumbs_up	up
call_me	466	3	1	2	2	1	2	2	1
fingers_crossed	1	475	1	0	1	0	0	1	1
fist	0	0	479	0	0	0	0	0	1
okay	0	0	0	475	2	0	0	1	2
peace	1	0	0	2	471	3	0	1	2
rock	0	0	0	0	2	470	2	3	3
stop	2	1	0	0	0	1	466	2	8
thumbs_up	1	0	0	0	0	2	1	474	2
up	0	1	0	2	2	3	5	6	461

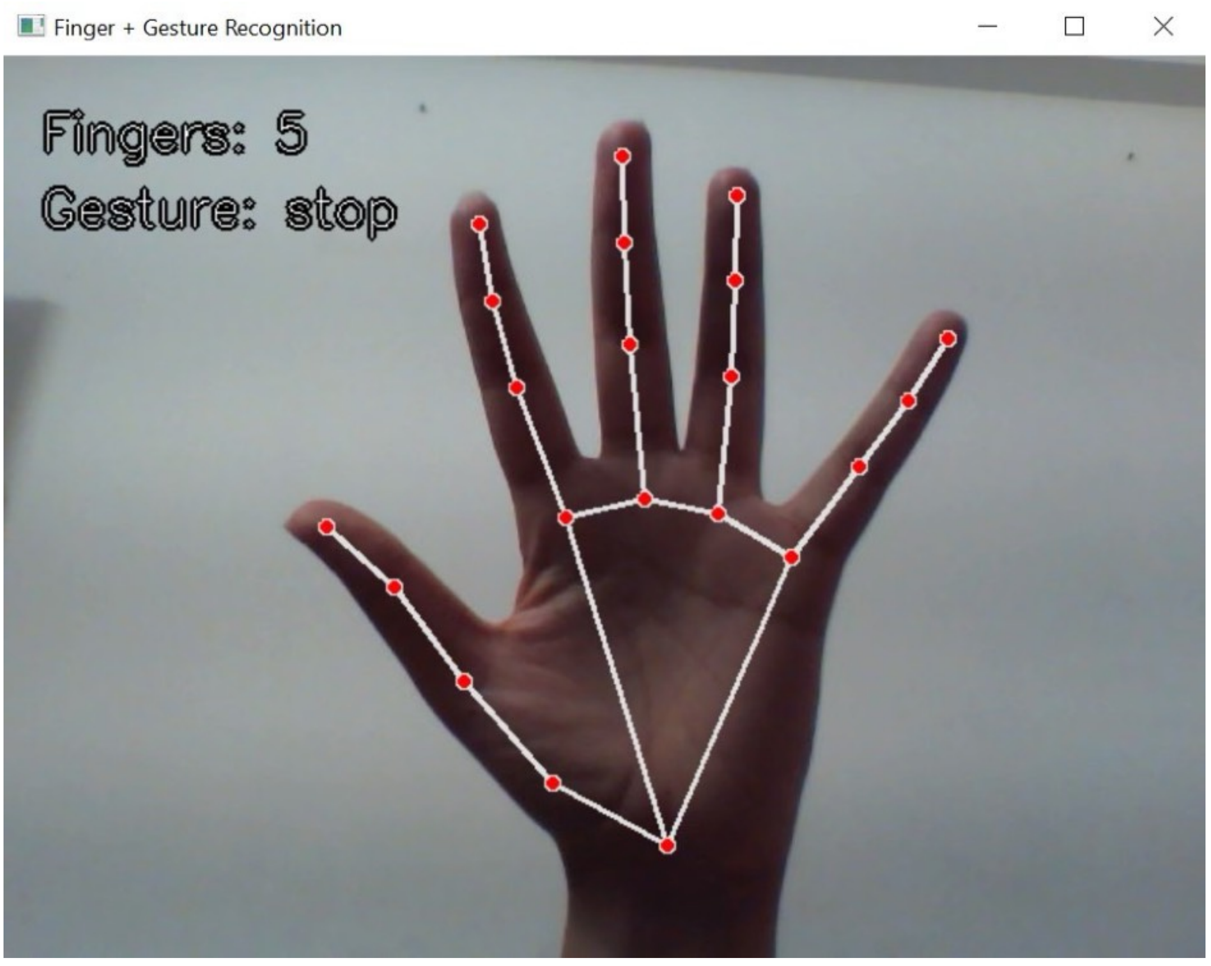
To further test the performance of the gesture classification model, a confusion matrix was created using the test set. The confusion matrix gives a precise breakdown of how well the classifier is doing per gesture class by comparing predicted labels with actual labels.

Each row in the matrix is the actual class, and each column is the predicted class. The diagonal elements of the matrix show the number of correctly classified instances for every gesture. A model with perfect accuracy would have all values on the diagonal and zeros elsewhere.

As is evident from the confusion matrix graph, the model has good performance in all the classes of gestures. All but a few classes like "fist," "okay," "peace," and "rock" have nearly perfect classification with almost no misclassifications. Minor misclassifications are visible in cases of gestures like "fingers_crossed" and "up," which could be due to their visual resemblance to other gestures or slight deviations in the orientation of hands.

In summary, the matrix ensures that the Random Forest classifier with HOG feature extraction and PCA-based dimensionality reduction correctly separates the 9 gesture classes with high accuracy and robustness.

This visualization offers strong evidence of the generalization capability of the model and emphasizes its reliability for real-time applications in gesture recognition.



5.2 Error Analysis

During system development of the gesture and finger counting system, there were numerous challenges, particularly in attaining resilient real-time performance. The main issues and their solutions are briefed below:

- **Low Initial Accuracy:**

The model performed poorly at first with an insufficient and less diverse training set. This was addressed by implementing data augmentation methods like rotation, flipping, and brightness modifications, which greatly increased model generalization and accuracy.

- **Inconsistent Real-Time Predictions:**

Real-time output was inconsistent initially due to discrepancies between training and inference preprocessing. This problem was addressed by ensuring the same preprocessing pipeline (grayscale, CLAHE, resize, HOG extraction) was used for both training and real-time data.

- **Hand Region Cropping Errors:**

When hand landmarks were close to the boundary edges, cropping usually wouldn't work. To resolve this, padding was introduced around the bounding box, and boundary checks were added to prevent extraction of an invalid region.

- **System Lag/Delays:**

The original system exhibited prominent lag. This was overcome by using a light-weight classifier (Random Forest), eliminating redundant operations in every frame, and making the code more efficient.

These enhancements together resulted in a robust and effective real-time hand gesture recognition system.

6. Conclusion and Future Work

This project was able to implement a real-time hand gesture recognition and finger counting system using a webcam. With the use of MediaPipe for precise hand landmark detection, HOG (Histogram of Oriented Gradients) for effective feature extraction, and a Random Forest classifier for resilient gesture classification, the system was able to reach high accuracy (98%) for nine unique hand gestures.

The system began as a simple finger counting tool, but eventually was enhanced with gesture recognition capabilities, becoming an all-purpose device for touch-free interaction. With iterative refinement and debugging—like enhancing preprocessing uniformity, enriching the data set, and optimizing performance—the end model met with consistent real-time responsiveness even in the face of different lighting conditions and hand positions.

The success of this project attests to the power of pairing classical machine learning with real-time computer vision software such as MediaPipe. It provides entry to a host of practical uses, including:

- Touchless control panels for intelligent home systems
- Disability support technologies
- Interactive education tools and learning environments
- Game and AR/VR interaction platforms

Future Work

The current system is strong, but there are some avenues where further improvement can be pursued:

- **Dataset Expansion:**

The present dataset, although enriched, was constructed from one user. For better generalization of the model and prevention of overfitting, future research should incorporate gesture samples from a variety of users, with varying lighting and backgrounds.

- **Multi-Hand Support:**

The system presently supports single-hand recognition. Future versions could incorporate dual-hand gesture recognition, enabling more sophisticated interactions and commands.

- **Model Comparison and Optimization:**

While the Random Forest classifier performed well, using deep learning models like CNNs (Convolutional Neural Networks), or alternate classifiers like SVC (Support Vector Classifier), might provide better accuracy and stability, especially under tougher circumstances.

- **GUI Integration:**

Incorporating the system with a graphical user interface (GUI) would significantly improve usability. A GUI would enable users to engage more naturally with the system and see outputs (e.g., gesture labels, finger numbers) in a more interactive form.

- **Application Integration:**

Future development could also involve connecting gesture outputs to control actual applications—like media players, browsers, or bespoke software—to show real-world utility.

Finally, this project built a strong groundwork for real-time gesture-based interactive systems and possesses immense potential to develop further both academically and commercially.

7. Work Update

7.1 Work Completed:

- **Development of Gesture Recognition and Finger Counting System:**

Successfully developed and deployed a real-time system capable of performing hand gesture recognition and finger counting through a regular webcam. A finger counting module was first created with MediaPipe's hand landmark detection. Subsequently, the system was extended to identify a set of nine pre-defined static hand gestures like "peace," "thumbs up," "call me," etc.

- **Custom Dataset Creation and Augmentation:**

A manually constructed custom dataset was taken by recording gesture images from a webcam at different angles, illuminations, and hand positions. There were 200+ images for each of the 9 gesture classes, and the dataset was augmented with techniques like flipping, rotation, brightness changes, and zooming to enhance model generalization and resistance. This increased the number of per-class images to roughly 2,400.

- **Preprocessing Pipeline and Feature Extraction:**

Implemented a uniform preprocessing pipeline involving grayscale conversion, CLAHE for contrast enhancement, resizing, and HOG feature extraction. This ensured that both training and real-time inference had the same input format in order to be consistent and improve the model accuracy.

- **Model Training and Evaluation:**

Trained a Random Forest Classifier on the HOG features drawn from the dataset. The model performed 98% accuracy, and the classification report indicated high precision, recall, and F1-scores for all classes of gesture. Real-time testing validated that the model performed under diverse conditions with little lag.

- **Error Handling and Performance Optimization:**

Resolved issues like improper cropping of the hand area, real-time prediction discrepancies caused by non-uniform preprocessing, and slight system lag. Added boundary checks and padding for robust cropping, and optimized code to decrease processing time per frame.

7.2 Work Remaining:

- **Dataset Expansion and User Diversity:**

While the current system works efficiently, the data was gathered from one user only. To ensure generalization and robustness to various users, skin tones, and hand types, more data must be gathered from multiple individuals in varied conditions.

- **Multi-Hand Gesture Recognition**

Currently, the system only accommodates a single hand gesture. Future addition includes supporting dual-hand tracking and gesture recognition to allow recognition of more intricate commands and interactions (e.g., two-hand zooming or rotating).

- **Graphical User Interface (GUI) Development:**

The system is presently operating through terminal and real-time video feed display. Creating a friendly GUI would increase the accessibility of the system to end users by showing gesture labels, finger counts, and logs in a visual interface, and possibly enabling users to personalize commands or associate gestures with particular system actions.

- **Integration with Real Applications:**

As a next step, gestures can be mapped to control actual applications—like volume control, slide navigation, or smart home devices—to showcase practical utility in real-world use cases.

8. References

- **MediaPipe Documentation – Google**

https://mediapipe.dev/

- **Histogram of Oriented Gradients (HOG) – Dalal & Triggs**

https://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf

- **Random Forest Classifier – Breiman**

https://www.stat.berkeley.edu/~breiman/randomforest2001.pdf

- **OpenCV Documentation – OpenCV**

https://opencv.org/

- **Leap Gesture Recognition Dataset – Kaggle**

https://www.kaggle.com/datasets/gti-upm/leapgestrecog/data

- **Scikit-learn Documentation – scikit-learn**

https://scikit-learn.org/

- **CLAHE (Contrast Limited Adaptive Histogram Equalization) – OpenCV**

https://docs.opencv.org/4.x/d5/daf/tutorial_py_histogram_equalization.html

- **Finger Counting and Gesture Recognition Tutorials – GitHub and Blogs**

https://github.com/topics/hand-gesture-recognition

https://learnopencv.com/hand-keypoint-detection-using-deep-learning-and-opencv/