Hand-Gesture based Human-Computer Interaction System

Dhruv Jethva
Department of Artificial Intelligence and Data Science
SIES Graduate School of Technology
Nerul, Navi Mumbai
India
dhruvjaids121@siesgst.ac.in

Ratul Raj
Department of Artificial Intelligence and Data Science
SIES Graduate School of Technology
Nerul, Navi Mumbai
India
ratulraids121@siesgst.ac.in

Devanshu Deshmukh
Department of Artificial Intelligence and Data Science
SIES Graduate School of Technology
Nerul, Navi Mumbai
India
Devanshudaids121@siesgst.ac.in

Shreya Kumar
Department of Artificial Intelligence and Data Science
SIES Graduate School of Technology
Nerul, Navi Mumbai
India
shreyakaids121@siesgst.ac.in

Abstract-

Welcome to the world of hand gesture-based Human-Computer Interaction (HCI), where the power of intuitive gestures meets cutting-edge technology to revolutionize the way we interact with computers. Hand gesture-based HCI systems enable users to control computers, devices, and applications using natural hand movements, eliminating the need for physical input devices such as keyboards or mice [6] . With the advancement of computer vision, machine learning, and sensor technologies, hand gesture-based HCI systems have gained increasing attention due to their potential for intuitive, efficient, and immersive interactions.

From gaming and virtual reality to healthcare and industrial automation, hand gesture-based HCI systems are transforming the way we interact with technology, making it more natural, engaging, and accessible to users of all ages and abilities. In this introduction, we will explore the exciting world of hand gesture-based HCI, its applications, challenges, and future prospects, unlocking the potential of human-computer interaction through the power of gestures[6].

Introduction-

Human-computer interaction (HCI) has become an increasingly important area of Research. Among the many modes of HCI, hand gesture recognition has garnered significant attention due to its natural and intuitive way of interacting with computers and other devices. With hand gesture recognition systems, users can control computers, smart devices, and virtual reality applications by simply moving their hands without the need for physical contact or complex input devices[4].

In recent years, there has been a growing interest in utilizing machine learning techniques for hand gesture recognition in HCI. One popular approach involves using classifiers along with Google's open-source library MediaPipe. MediaPipe provides a comprehensive framework for building real-time applications for various computer vision tasks such as hand tracking and pose estimation[5]. By leveraging the robust pipeline provided by MediaPipe to extract landmarks from video input data, it is possible to build accurate and real-time hand gesture recognition systems that can recognize a wide range of gestures. Hand landmarks are key points detected on the hand such as fingertips, knuckles, and wrists that can be used to represent different hand gestures. By utilizing these landmarks extracted from MediaPipe[4] along with machine learning classifiers, it is possible to build accurate and real-time hand gesture recognition systems that can recognize simple gestures like waving or thumbs-up as well as more complex gestures for controlling virtual objects or interacting with virtual environments. The objective of our research paper is to explore the potential of using classifiers with MediaPipe landmarks for hand gesture recognition in HCI[5]. We will review existing literature on various techniques used

for recognizing different types of gestures while focusing specifically on approaches that utilize MediaPipe landmarks. Additionally, we will propose our own methodology involving training a classifier using these landmarks extracted from MediaPipe followed by evaluating its performance on a dataset of different types of gestures. Our experiments aim at assessing the accuracy, speed, and robustness of our proposed system while comparing its performance with other state-of-the-art methods.

The results obtained from our research can contribute to the advancement of hand gesture recognition techniques for HCI and provide valuable insights for developing practical applications in various domains such as virtual reality, gaming, healthcare, and human-robot interaction[6]. This research paper contributes to developing a robust system utilizing both MediaPipe library tools for accurate landmark detection alongside our developed classifier model capable of accurately identifying various gestures with high precision rates across multiple applications such as virtual reality interactions or healthcare monitoring[4].

Objective -

The purpose of this study is to assess the effectiveness of a hand gesture recognition system that utilizes MediaPipe and landmarks for human-computer interaction. The research aims to delve into the accuracy, efficiency, and usability of the proposed system in identifying hand gestures and converting them into meaningful commands for interacting with digital devices, including computers. Additionally, this investigation will examine how different factors such as lighting conditions, hand poses, and shapes may influence the performance of the hand gesture recognition system. The results of this research will enhance our understanding of human-computer interaction and provide valuable insights for enhancing hand gesture recognition systems for various applications like accessibility technologies, gaming and virtual reality [2].

Literature Review -

A. Parth Shah, Raj Shah, Maulik Shah, Kiran Bhowmick “Comparative Study of Hand Gesture Recognition Techniques: A Review” [1] studied and compared various hand gesture recognition techniques, including support vector machine, k-nearest neighbours, and deep learning-based approaches where results showed that deep learning-based approaches achieved higher accuracy than traditional machine learning techniques.

B. Ishika Dhall, Shubham Vashisht, Garima Aggarwal “Automated Hand Gesture Recognition using a Deep Convolution Neural Network model” [2], shows the deep convolution neural network implementation where Max Pooling Convolution Neural Network is used to advance Human-robot interactions using segmentation, edge blurring with morphological digital image processing and then experimenting with mobile robots using ARM 11 533 MHz. This result is well shown in [5] where a training set of 50% database is used and successfully implemented.

C. Jitendra Kumar Jaiswal, Rita Samikannu “ Application of Random Forest Algorithm on Feature Subset Selection and Classification and Regression” [3] Shows the ‘Chronic Kidney Disease’ data from UCI machine learning repository and applied random forest algorithm with help of the ‘Boruta’ package from R programming language for the feature subset selection and achieved the reduced set that is subset of the variables. Also applied the ‘random Forest’ package from R programming language for the classification and regression, and observed the functioning of random forest method in different perspectives.

D. Zhang Bingzhen, Qiao Xiaoming, Yang Hemeng, Zhou Zhubo “A Random Forest Classification Model for Transmission Line Image Processing”[4] Method of random forest classification model selection based on confusion matrix is proposed. The confusion matrix is applied to the similarity measure of the decision tree, and the similarity between two trees is judged by using the distance and vector angle of the matrix, combined with the classification performance of decision tree. The idea of "delete inferior" is used to select the random forest model. The experimental results in [4] show that this method improves the accuracy of classification.

E. Veena N. Jokhakar, S. V. Pate [5] "A Random Forest Based Machine Learning Approach For Mild Steel Defect Diagnosis" explored machine learning algorithm, Random forest and to propose a model to defect cause diagnosis problem of steel industry explored machine learning algorithm, applied random forest machine learning algorithm, neural network, SVM, decision tree (rpart) and boosting algorithms and compared the results in terms of accuracy and other statistics. Found that random forest to outperform all the rest of the applied algorithm. proposed model in [5] outperforms other fault / defect diagnosis mechanisms by achieving highest accuracy of 95%.

F. Meenakshi Panwar, Pawan Singh Mehra "Hand Gesture Recognition for Human Computer Interaction" [6] proposed a simple shape-based approach for hand gesture recognition with several procedures, such as orientation detection, thumb detection, smudges elimination etc. The proposed approach use shape base features for recognizing different hand ; no complex feature calculation, and no any significant amount of training or post processing required, gives higher recognition rate with less computation time. Presented a system which is independent of user characteristics and can be used for person independent recognition of hand gesture against uniform plain backgrounds. The segmentation of image does not affect and gives better result even if the light conditions are different. The success rate has been improved from 91% to 92.3% and the number of test images have been increased from 200 to 390.

Methodology-

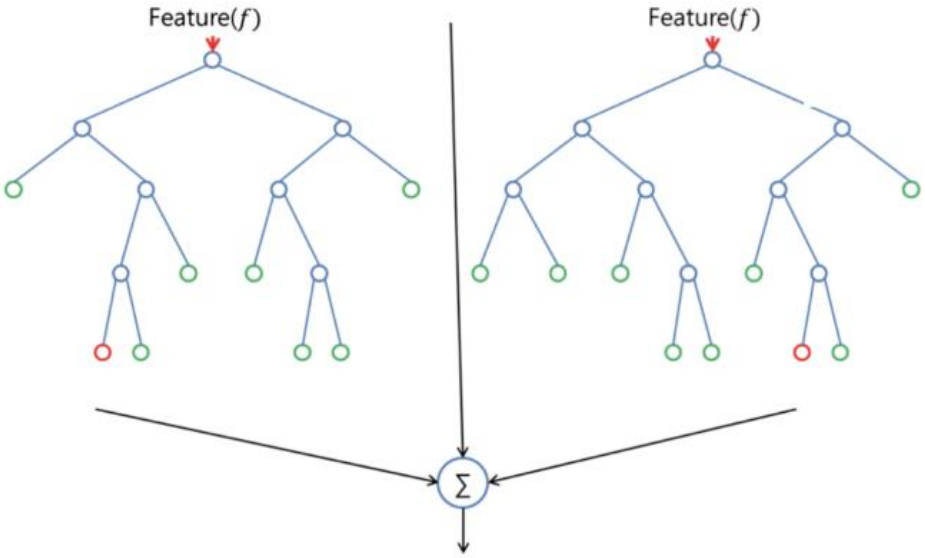
Random Forest Classifier

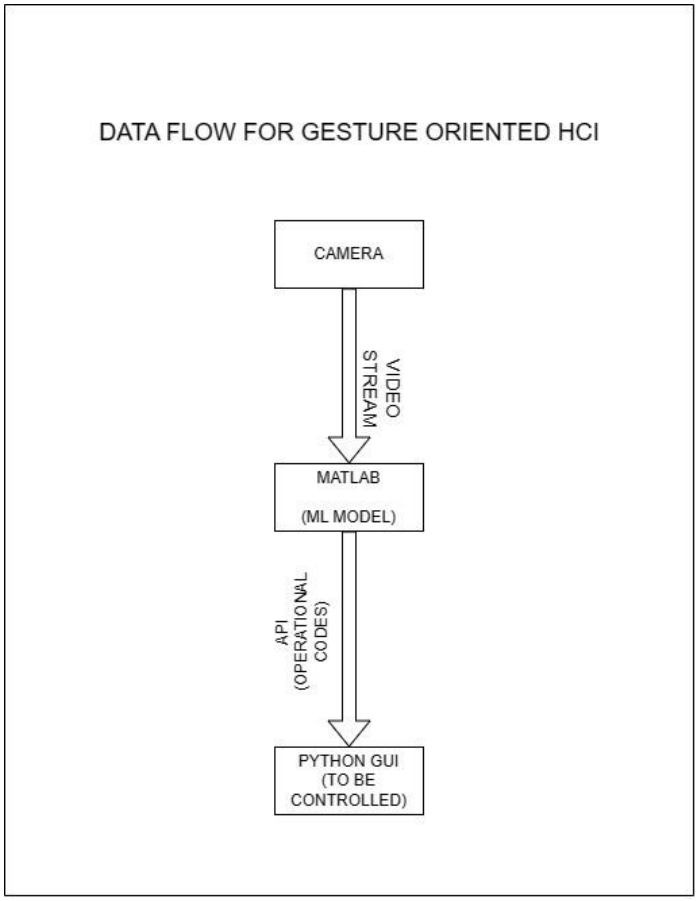
A random forest classifier is a type of ensemble learning method for classification, regression, and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. In a random forest classifier, each decision tree is trained on a different subset of the training data, and also randomly selects a subset of features to consider when making each decision. This helps to avoid overfitting and improves the accuracy and robustness of the classifier. At prediction time, the output of the random forest is the average prediction of all the individual decision trees. Random forest classifiers are widely used in machine learning applications because they can achieve high accuracy on a wide range of tasks and are relatively easy to use and interpret.

Proposed Implementation

Our aim is to create the proposed system using Python. As such, the following modules were imported.

- OS, for accessing file system
- Pickle, for serializing data
- Mediapipe for hand landmark detection
- Cv2 for image processing
- Matplotlib, pyplot, for visualisation
- NumPy, for performing numerical operations on arrays


Feature(f)
Feature(f)
Σ


DATA FLOW FOR GESTURE ORIENTED HCI
CAMERA
VIDEO STREAM
MATLAB
(ML MODEL)
API (OPERATIONAL CODES)
PYTHON GUI (TO BE CONTROLLED)

Proposed Implementation

The entire program can be divided into four sections:-

A) Hand mark detection

This code performs hand landmark detection on images stored in a directory, and saves the hand landmark data and corresponding labels in a pickle file.
Here is a step-by-step breakdown of the code: The necessary modules are imported: os,pickle, mediapipe , cv2 and matplotlib.pyplot .The mediapipe Hands object is initialized with the parameters static_image_mode set to True and min_detection_confidence set to 0.3. These parameters configure the hand detection pipeline to work on single images rather than video streams, and to only return detected hands with a confidence score above 0.3. The path to the data directory is set to './data'. Two empty lists, data and labels, are initialized. These lists will hold the hand landmark data and corresponding labels. A loop is started to iterate over the directories within the data directory. Each directory corresponds to a different label, contains images of hands with that label. Another loop is started to iterate over the image files within each label directory. An empty list, data_aux, is initialized. This list will hold the hand landmark data for a single image. Two empty lists, x_ and y_, are initialized. These lists will hold the x and y coordinates of the hand landmarks for normalization. The current image is read with cv2.imread and converted to RGB format with cv2.cvtColor. The hands object is used to process the image and detect hand landmarks. If hand landmarks are detected with sufficient confidence, the x and y coordinates of each landmark are extracted and added to the x_ and y_ lists. For each detected hand, the x and y coordinates of each landmark are normalized with respect to the minimum x and y coordinates. The normalized coordinates are then added to the data_aux list. The data_aux list is added to the data list, and the label for the current image directory is added to the labels list. After all images have been processed, the data and labels lists are saved in a pickle file named 'data.pickle'.

B) Collecting Data from Images.

The necessary modules are imported: os, and cv2.
The path to the data directory is set to './data'. If the directory does not exist, it is created with os.makedirs. The number of classes is set to 3, and the size of the dataset to 100. The cv2.VideoCapture object is initialized with index 1, which typically represents the second available camera. A loop is started to iterate over the number of classes. For each class, a directory is created within the data directory with os.makedirs if it does not already exist. The class label is the name of the directory, which is a string representation of the class index. A message is printed to the console to prompt the user to prepare to start recording images for the current class. A while loop is started to wait for the user to press the 'q' key to start recording images. After the user presses the 'q' key, another while loop is started to capture and save images for the current class. The cv2.imshow function is used to display the current frame, which is captured with cap.read. The cv2.waitKey function is used to wait for a short duration before capturing the next frame. The current frame is saved as an image file with cv2.imwrite, using the counter variable to name the file.

The counter variable is incremented by 1 after each image is saved, and the while loop continues until the desired number of images (dataset_size) have been collected. After all classes have been processed, the cv2.VideoCapture object is released with cap.release() and all windows opened by cv2.imshow are closed with cv2.destroyAllWindows().

C) Classification of hand gestures

This code performs real-time classification of hand gestures captured by a camera, using a pre-trained model. The necessary libraries imported at the beginning of the code are Pickle is used to load the pre-trained model from a file, cv2 , and mediapipe. The pre-trained model is loaded from a file using pickle.load(), and the model object is obtained. A video capture object is created using cv2.VideoCapture(2), which captures video from the camera with index 1. Then, an instance of the Hands class from mediapipe is created, which is used to detect and track hand landmarks. A dictionary is created to map the integer labels predicted by the model to the actual characters (letters) they represent. In an infinite loop, frames are captured from the video stream and processed to detect hand landmarks. If hand landmarks are detected, they are visualized on the frame using the draw_landmarks() function from mediapipe, and the relative positions of the hand landmarks are used to create a feature vector that is passed to the pre-trained model for classification. The predicted label is used to update the displayed text on the frame. Finally, the frame is displayed using cv2.imshow(). The loop continues until the user interrupts the program by pressing a key. The video capture object is then released using cap.release() and all windows created by OpenCV are closed using cv2.destroyAllWindows().

D)Training the Classifier model

This code is for training a Random Forest Classifier model on some input data, and then saving the model to a file using pickle. First, the code loads the input data from a file called 'data.pickle', which contains two arrays: 'data', which contains the input features, and 'labels', which contains the corresponding labels. It uses the pickle library to deserialize the data. The code then splits the data into training and testing sets using the train_test_split function from scikit-learn. The test set is set to be 20% of the total data, and stratification is used to ensure that the label distribution is the same in the training and test sets. Next, a new instance of a Random Forest Classifier is created, and the model is trained on the training data using the fit method. The trained model is then used to predict the labels of the test data using the predict method, and the accuracy of the predictions is calculated using the accuracy_score function. Finally, the trained model is saved to a file called 'model.p' using the pickle library, which serializes the model object and writes it to a file.

Conclusion :-

In the research paper, a hand gesture recognition system is presented using the Random Forest classifier with the Mediapipe library and landmark-based features. The study demonstrates the effectiveness of this approach in accurate real-time gesture recognition, making it a promising solution for human-computer interaction (HCI) applications The results of the study show that high accuracy in gesture recognition can be achieved by using random forest classification with features extracted by MediaPipe. This suggests that combining machine learning algorithms with computer vision techniques can provide robust and accurate gesture recognition for HCI applications. Furthermore, the study highlights the potential of using gestures as a simple and natural way of interacting with computer systems, especially in situations where traditional input methods e.g keyboards or touchscreens can be difficult or emphasize impractical. Functions can occur in different industries such as networking and networking However, the proposed system also has limitations, such as sensitivity to lighting conditions, hand compression, and changes in hand size and shape Further research could focus on overcoming these limitations and improve system performance under harsh real-world conditions. Overall, this study contributes to the HCI field by providing a manual recognition algorithm that combines marker-based features with random forest classification, providing a promising approach to acquisition natural human-computer interaction flexibility The results of this study can form the basis for further developments in the gesture recognition system.

REFRENCES

[1] Soeb Hussain, Rupal Saxena, Xie Han, Jameel Ahmed Khan, Hyunchal Shin "Hand Gesture Recognition using Deep Learning", *International SOC Design Conference (ISOCC) 2017,* in press

[2] Felix Zhan "Hand Gesture Recognition with Convolution Neural Networks", *IEE 20th international Conference on Information Reuse and Integration for Data Science (IRI) 2019,* in press

[3] Parth Shah, Raj Shah, Maulik Shah, Kiran Bhowmick "Comparative Study of Hand Gesture Recognition Techniques: A Review", *Advanced Computing Technologies and Applications,* pp. 471-478, 2020

[4] Ishika Dhall, Shubham Vashisht, Garima Aggarwal "Automated Hand Gesture Recognition using a Deep Convolution Neural Network model", *10th International Conference on Cloud Computing, Data Science & Engineering (Confluence) 2020*, in press

[5] Nagi, Jawad, et al. "Max-pooling convolution neural networks for vision-based hand gesture recognition", *IEEE International Conference on Signal and Image Processing Applications (ICSIPA),* pp. 342-347, 2011

[6] Jitendra Kumar Jaiswal, Rita Samikannu "Application of Random Forest Algorithm on Feature Subset selection and Classification and Regression*", World Congress on Computing and Communication Technologies (WCCCT)*

[7] Zhang Bingzhen, Qiao Xiaoming, Yang Hemeng, Zhou Zhubo "A Random Forest Classification Model for Transmission Line Image Processing", *The 15th International Conference on Computer Science & Education (ICCSE 2020) August 18-20, 2020. Online*

[8] Veena N. Jokhakar, S. V. Pate "A Random Forest Based Machine Learning Approach For Mild Steel Defect Diagnosis*" 2016 IEEE International Conference on Computational Intelligence and Computing Research, 978-1-5090-0612-0*

[9] Meenakshi Panwar, Pawan Singh Mehra "Hand Gesture Recognition for Human Computer Interaction" *2011 International Conference on Image Information Processing (ICIIP 2011)*