

Interactive Hand Gesture Recognition with Audio Response

Submitted in partial fulfillment of the requirements for the award
of

Bachelor of Technology Degree in Information Technology

By

Abdul Ajees.I (40120002)

Abel PriyaKumar.P (40120003)



**DEPARTMENT OF INFORMATION TECHNOLOGY
SCHOOL OF COMPUTING**

**SATHYABAMA INSTITUTE OF SCIENCE AND
TECHNOLOGY**

(DEEMED TO BE UNIVERSITY)

Accredited with Grade “A++” by NAAC

**JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600119**

APRIL 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited "A++" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF INFORMATION TECHNOLOGY

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the Bonafede work of I. Abdulajees (40120002) and P. Abel Priya Kumar (40120003) who carried out the Project entitled **"Interactive Hand Gesture Recognition with Audio Response"** under my supervision from January 2024 to April 2024.

Internal Guide

DR. P. JEYANTHI M.E., Ph.D.,

Head of the Department

Dr. R. Subhashini M.E., Ph.D.,

Submitted for Viva voce Examination held on_____

Internal Examiner

External Examiner

DECLARATION

I, Abdulajeess (Regno:40120002) are hereby declaring that the Project Report entitled “**Interactive Hand Gesture Recognition with Audio Response**” done by me under the guidance of **DR. P. JEYANTHI M.E, Ph.D.**, is submitted in partial fulfillment of the requirements for the award of a Bachelor of Engineering degree in **Information Technology**.

DATE:

PLACE: Chennai

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **the Board of Management of SATHYABAMA** for their kind encouragement in doing this project and completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph. D, Dean**, School of Computing, **Dr. R.Subashini M.E., Ph.D.**, Head of the Department of Computer Science and Engineering for providing me with the necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide Manju Nair for her valuable guidance, suggestions, and constant encouragement paved way for the successful completion of my phase-1 project work.

I wish to express my thanks to all the Teaching and Non-teaching staff members of the **Department of Information Technology** who were helpful in many ways for the completion of the project.

Abstract

Real-time hand gesture recognition is a technology that allows a computer system to identify and interpret human hand gestures in real time. This technology is used in various applications, such as virtual reality, gaming, human-computer interaction, and sign language recognition. Real-time hand gesture recognition typically involves the use of computer vision techniques, such as image processing and machine learning algorithms, to analyze the live video feed of a person's hands and classify the gestures based on predefined patterns. The system may use cameras, sensors, or other input devices to capture hand movements and then analyze them to recognize the gestures. The real-time aspect of hand gesture recognition is critical for applications that require immediate feedback, such as in gaming or virtual reality. Real-time hand gesture recognition systems have made significant advances in recent years due to advancements in computer vision and machine learning technologies. Hand gesture refers to the movement or positioning of the hands and fingers to convey a message, emotion, or intention. Hand gestures are nonverbal communication that can complement or replace spoken language. Hand gestures can be used to express a wide range of emotions and meanings, such as indicating directions, showing agreement or disagreement, expressing love or affection, indicating size or quantity, conveying a warning, and many more. Different cultures may have their own set of hand gestures and meanings, so it's important to be aware of cultural differences when using hand gestures in communication

CHAPTER NUMBER	TITLE	PAGE NUMBER
	ABSTRACT	iv
	LIST OF FIGURES	vii
	CHAPTER 1	1
1	1.1 INTRODUCTION 1.2 PROBLEM STATEMENT 1.3 OBJECTIVES 1.4 SCOPE	1 2-3 3-4 4-5
	CHAPTER 2	6
2	2.1 LITERATURE SURVEY 2.2 FLAWS IN THE EXISTING SYSTEM	6-7 8-9
	CHAPTER 3	10-16
3	3.1 SOFTWARE REQUIREMENTS 3.2 ADVANTAGES 3.3 DISADVANTAGES	10-14 14-15 15-16
	CHAPTER 4	17-27
4	4.1 PROPOSED METHODOLOGY 4.2 SYSTEM ARCHITECTURE 4.3 BLOCK DIAGRAM	17-20 21-25 25-27

5	CHAPTER 5	28-30
	5.1 CONCLUSION	28-29
	5.2 FUTURE SCOPE	29-30
6	APPENDIX	32

LIST OF FIGURES

FIG NO	FIGURE NAME	PAGE NO
4.2	SYSTEM ARCHITECTURE	21
4.3	BLOCK DIAGRAM	25

CHAPTER 1

1.1 INTRODUCTION

Hand gesture recognition is the process of using computer vision and machine learning techniques to detect and interpret the gestures made by a person's hand or hands. This technology is used in a variety of applications, such as sign language recognition, virtual reality and augmented reality, and human-computer interaction.

There are several techniques used for hand gesture recognition, including feature extraction, machine learning, and deep learning. Feature extraction involves identifying specific points on the hand, such as the fingertips or palm, and using these points to create a mathematical representation of the hand. Machine learning algorithms are then trained on this data to recognize specific gestures.

Deep learning techniques, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), have also been used for hand gesture recognition. These techniques involve training a neural network on large amounts of data to recognize patterns and features in the hand gestures.

Hand gesture recognition technology has many potential applications, including in the fields of healthcare, entertainment, and robotics. For example, it could be used to develop more intuitive and natural ways for people to interact with computers and other devices, or to provide more effective therapies for people with mobility impairments.

1.2 PROBLEM STATEMENT

Hand gestures are used in computers for various applications, including human-computer interaction, gaming, virtual reality, and sign language recognition. By using hand gestures, users can interact with computers in a more natural and intuitive way than using traditional input devices such as a keyboard or mouse.

Hand gesture recognition technology allows computers to interpret and respond to hand movements, making it possible to control applications and devices with gestures. This technology enables users to perform tasks such as scrolling, zooming, selecting, and dragging without the need for a physical input device.

In addition, hand gesture recognition is a useful tool for people with disabilities, such as those who are unable to use traditional input devices due to physical limitations. For example, individuals with limited mobility or speech impairments can use hand gestures to communicate with computers and control devices.

Overall, hand gesture recognition technology provides a more natural and intuitive way for users to interact with computers, making it easier to control applications and devices, and improving accessibility for individuals with disabilities.

Human-computer interaction (HCI) is a multidisciplinary field that focuses on designing, evaluating, and implementing interactive computer systems and technologies that are accessible, effective, and efficient for human users. HCI aims to create user-friendly and intuitive interfaces that enable users to interact with computer systems in a natural and efficient way.

The goal of HCI is to understand how humans interact with technology and to design systems that support these interactions. This involves studying how people use computers, identifying their needs, preferences, and limitations, and developing interfaces that are tailored to their abilities and preferences.

HCI involves a wide range of disciplines, including computer science, psychology, design, and engineering. It covers a broad range of topics, including user interface design, usability testing, user experience, cognitive psychology, and human factors.

HCI has applications in many fields, including education, healthcare, entertainment, and business. It is used to design and develop a variety of interactive technologies, including websites, mobile apps, virtual reality environments, and assistive technologies.

Overall, HCI is a critical field that ensures that technology is developed with the needs and capabilities of human users in mind. By creating user-friendly and efficient interfaces, HCI can help to improve the usability, accessibility, and effectiveness of computer systems and technologies for all users

1.3 OBJECTIVES

The main objectives of hand gesture recognition are:

Human-Computer Interaction: One of the primary objectives of hand gesture recognition is to provide a natural and intuitive interface for human-computer interaction. By recognizing hand gestures, computers can interpret and respond to user movements, allowing users to interact with applications and devices in a more natural and efficient way.

Accessibility: Hand gesture recognition technology can also improve accessibility for individuals with disabilities, such as those with limited mobility or speech impairments. By using hand gestures, these individuals can communicate with computers and control devices, providing greater independence and autonomy.

Gaming and Virtual Reality: Hand gesture recognition can enhance the gaming and virtual reality experience by allowing users to control game characters and virtual objects with their hand movements. This technology provides a more immersive and interactive gaming experience, making it more enjoyable and engaging for users.

Sign Language Recognition: Hand gesture recognition technology can also be used for sign language recognition, allowing computers to translate sign language into

spoken language. This can help to improve communication between individuals who are deaf or hard of hearing and those who are not.

Security: Hand gesture recognition can be used for security purposes, such as in biometric authentication systems. By recognizing specific hand gestures, computers can identify and authenticate users, providing a secure and reliable form of authentication.

Overall, the objectives of hand gesture recognition are to provide a natural and intuitive interface for human-computer interaction, improve accessibility for individuals with disabilities, enhance gaming and virtual reality experiences, enable sign language recognition, and enhance security through biometric authentication.

1.4 SCOPE

The scope of hand gesture recognition is quite broad and includes many potential applications. Here are some examples:

Human-Computer Interaction: Hand gesture recognition can be used as a more intuitive and natural way for people to interact with computers and other devices. It could enable touchless control of devices such as smartphones, tablets, and laptops.

Virtual and Augmented Reality: Hand gesture recognition can be used in virtual and augmented reality applications, where it enables users to interact with virtual objects in a more natural way.

Sign Language Recognition: Hand gesture recognition can be used to recognize sign language, which can help deaf or hard-of-hearing individuals to communicate with others who do not know sign language.

Gaming: Hand gesture recognition can be used in gaming applications to provide a more immersive and interactive experience. For example, it could enable players to

control the game using hand gestures instead of traditional controllers.

Healthcare: Hand gesture recognition can be used in healthcare applications to help people with mobility impairments or disabilities to interact with their environment more independently. For example, it could enable individuals to control their wheelchair or other assistive devices using hand gestures.

Security: Hand gesture recognition can be used for security applications, such as access control systems, where it enables biometric authentication using hand gestures.

Automotive: Hand gesture recognition can be used in automotive applications, such as controlling the infotainment system or adjusting the temperature and ventilation system, without the need to touch buttons or knobs.

Overall, hand gesture recognition has a wide range of potential applications, and its scope is expected to grow as the technology becomes more advanced and accessible.

CHAPTER 2

2.1 LITERATURE SURVEY

1. Recognition of Real-Time Angklung Kodály Hand Gesture using Media pipe and Machine Learning Method

This study developed a computer vision-based hand gesture recognition system to recognize the hand gestures used in Saung Angklung Udjo. The method used in this system utilizes a webcam that is commonly used by the public; this allows for this system to be widely used by people who have computers or laptops. Moreover, in this study, a program was also developed to play the sound of angklung utilizing the computer's audio device, according to hand gestures given in real time by the user. By using the Mediapipe framework for feature extraction, programs can run using relatively light computing power

2. Real-Time Sign Language Recognition System

The Sign Language Detection Model captures hand gestures through webcam using OpenCV. After capturing the image, they used pre-trained model SSD Mobile net V2 for recognition of the sign language gesture. They used Convolution Neural Network (CNN) for handling 2D and 3D data as input and for capturing the Temporal and Spatial aspects of an image with the help of appropriate filters. While their major achievements include accurate results under controlled light and intensity, wrong result in very low light is also one of their major drawbacks. Their model makes it simple to incorporate unique gestures, and photographs shot at various angles and frames will increase the model's accuracy

3. A Continuous Real-time Hand Gesture Recognition Method based on Skeleton

In this paper, we propose a framework for continuous hand gesture recognition based on skeleton information. Figure 1 illustrates the continuous hand gesture recognition this aims at detecting simultaneously the starting point, ending point and the class of hand gesture. It's worth noting that in the isolated hand gesture

recognition, the starting and ending points are determined a prior. The main idea of the proposed method is to first perform gesture detection and then to identify the type of hand gestures.

4. Gesture based Real-Time Sign Language Recognition System

This paper focuses on sign language recognition for both deaf and dumb people as it displays the text output for deaf people and produces audio output for the blind using the sign language. The motivation behind this work is to build a lowcost feasible system that is easily accessible by everyone. Since it is important to understand the basics of sign language in order to communicate, the proposed system has used the 26 alphabets of American Sign Language .

5. CNN based Approach for Sign Recognition in the Indian Sign language

The quality of the dataset is one of the important factors that affect the accuracy of Machine Learning models. To create datasets and perform preprocessing operations on them, “OpenCV” is used. The dataset is created by capturing the gestures images using webcam of laptop. The captured dataset is in raw form that needs to be cleaned by preprocessing

a. FLAWS OF THE EXISTING SYSTEM

Hand gesture recognition has become an increasingly popular technology in recent years, especially with the rise of smart devices and virtual reality. However, like any technology, it has its drawbacks. Here are some potential drawbacks of hand gesture recognition:

Limited range of recognition: Hand gesture recognition systems may only be able to recognize a limited range of gestures, which could limit their functionality in certain applications.

Variability in hand shape and size: Hands come in different shapes and sizes, and this variability can make it challenging for hand gesture recognition systems to accurately identify gestures across different users.

Environmental factors: Hand gesture recognition can be affected by environmental factors, such as lighting, shadows, and obstacles that obstruct the view of the hands.

Need for calibration: Some hand gesture recognition systems require calibration, which can be time-consuming and may need to be repeated periodically to maintain accuracy.

Sensitivity to movement: Some hand gesture recognition systems are very sensitive to small movements, which can lead to false positives or misinterpretation of gestures.

Processing power requirements: Hand gesture recognition requires significant processing power, which can make it challenging to implement on lower-end devices.

Limited user interface: Hand gesture recognition may not be suitable for all types of user interfaces, as some users may find it difficult to remember and perform the required gestures.

Privacy concerns: Hand gesture recognition systems may capture and store data about users' hand movements, which could raise privacy concerns if the data is not properly secured.

Overall, while hand gesture recognition has the potential to improve human-computer interaction, it is important to be aware of these potential drawbacks when implementing such systems.

CHAPTER 3

3.1 UNDERSTANDING OF THE PROJECT

3.1.1 PROGRAMING LANGUAGE USED AND WHY?

- **Python**

Python is a high-level, interpreted programming language that is widely used in various fields such as data science, machine learning, artificial intelligence, web development, scientific computing, and many more. There are several reasons why Python is a popular choice among programmers:

Easy to Learn and Use: Python has a simple and easy-to-read syntax that makes it easier for beginners to learn and write code quickly. Its syntax is similar to the English language, making it easier to understand and interpret.

Open-Source: Python is an open-source programming language, which means that it is free to use, distribute and modify. This makes it an attractive choice for developers and companies looking for cost-effective solutions.

Large and Active Community: Python has a large and active community developers who contribute to its development and provide support to users. This community provides a wealth of resources such as tutorials, documentation, and libraries, making it easier for developers to learn and use Python.

Multi-Purpose: Python can be used for a wide range of applications, from web development and data analysis to scientific computing and artificial intelligence. It has a large number of libraries and frameworks that make it easy to build complex applications quickly.

Portable: Python code can be run on different operating systems such as Windows, Linux, and macOS, making it easy to develop and deploy applications across different platforms.

Overall, Python is a versatile and easy-to-learn language that offers a wide range of features and applications. Its popularity is expected to continue growing as more developers and companies recognize its benefits and use it for their projects.

- **Open-CV**

OpenCV (Open Source Computer Vision) is a popular open-source computer vision and machine learning library that is widely used for image and video processing, object detection and recognition, face recognition, and many other applications. It is written in C++ and Python and provides a variety of tools and functions for developing computer vision applications.

OpenCV is cross-platform and runs on various operating systems such as Windows, Linux, and macOS. It provides a variety of high-level APIs that simplify complex tasks, such as image and video capture, object detection and recognition, and feature extraction.

Some of the key features of OpenCV include:

Image and Video Processing: OpenCV provides a variety of tools for processing and manipulating images and videos, such as color space conversion, filtering, and thresholding.

Object Detection and Recognition: OpenCV provides a variety of algorithms for detecting and recognizing objects in images and videos, such as face detection, object tracking, and motion detection.

Machine Learning: OpenCV provides a variety of machine learning algorithms for classification, regression, clustering, and more. It also provides tools for training machine learning models.

Cross-Platform: OpenCV can be used on various operating systems and is available in C++, Python, Java, and other programming languages.

Large Community: OpenCV has a large and active community of developers who contribute to its development and provide support to users.

Overall, OpenCV is a powerful and versatile library that is widely used in various fields, including computer vision, robotics, healthcare, and automotive industries. Its popularity is expected to continue growing as more developers and companies recognize its benefits and use it for their projects.

- **Tensor Flow**

TensorFlow is based on a dataflow programming model, where computations are represented as directed graphs of operations, and data flows through these graphs as tensors. It provides a variety of high-level APIs and pre-built models that simplify the development of complex machine learning models.

Some of the key features of TensorFlow include:

Flexibility: TensorFlow is highly flexible and can be used to build and train a variety of machine learning models, from simple linear regression to complex deep neural networks.

Distributed Training: TensorFlow provides support for distributed training, which enables users to train large-scale machine learning models on multiple devices or clusters.

Scalability: TensorFlow is highly scalable and can be used to train and deploy machine learning models on a variety of platforms, from mobile devices to large-scale data centers.

Cross-Platform: TensorFlow can be used on various operating systems, including Windows, Linux, and macOS, and is available in multiple programming languages such as Python, C++, and Java.

Large Community: TensorFlow has a large and active community of developers who contribute to its development and provide support to users.

Overall, TensorFlow is a powerful and flexible machine learning framework that is widely used in various fields, including computer vision, natural language processing, robotics, and many others. Its popularity is expected to continue growing as more developers and companies recognize its benefits and use it for their projects.

- **Media Pipe**

MediaPipe is an open-source cross-platform framework developed by Google that provides a variety of tools and pre-built components for building and deploying real-time machine learning and computer vision applications. It includes a variety of APIs and pre-built models for tasks such as object detection and tracking, hand and facial landmark detection, pose estimation, and many others.

MediaPipe is built on top of TensorFlow and leverages its machine learning capabilities to provide high-performance and accurate models. It is designed to work on a variety of platforms, including desktops, mobile devices, and embedded systems.

Some of the key features of MediaPipe include:

Cross-Platform: MediaPipe is designed to work on various platforms and provides support for desktops, mobile devices, and embedded systems.

Pre-built Components: MediaPipe includes a variety of pre-built components that simplify the development of real-time machine learning and computer vision applications.

Customizability: MediaPipe provides a variety of customization options, allowing developers to modify and extend the pre-built components to meet their specific needs.

Large Community: MediaPipe has a large and active community of developers who contribute to its development and provide support to users.

Overall, MediaPipe is a powerful and flexible framework that provides a variety of tools and pre-built components for building and deploying real-time machine learning and computer vision applications. Its popularity is expected to continue growing as more developers and companies recognize its benefits and use it for their projects.

3.2 ADVANTAGES

- User-friendly interface: Hand gesture recognition provides an intuitive and natural interface for users to interact with machines or devices. It eliminates the need for physical devices such as a mouse or keyboard and allows users to control devices with simple hand movements.
- Accessibility: Hand gesture recognition technology can be used by people with physical disabilities, making it more accessible than traditional input devices.
- Speed and accuracy: Hand gesture recognition systems can operate very quickly and accurately, allowing for efficient and precise control of devices. This can be especially beneficial in applications that require quick responses, such as gaming or virtual reality.
- Non-invasive: Unlike other forms of human-computer interaction such as touchscreens or voice recognition, hand gesture recognition does not require physical contact with the device, making it a non-invasive form of interaction.
- Versatility: Hand gesture recognition technology can be applied in a wide

range of fields and applications, such as gaming, virtual reality, healthcare, automotive, and robotics, among others.

- Cost-effective: With the development of affordable and accessible hardware such as depth cameras and sensors, hand gesture recognition technology has become more accessible and cost-effective, making it an attractive option for various applications.

Overall, hand gesture recognition technology provides a more intuitive, accessible, and efficient way for humans to interact with machines and devices, opening up new possibilities for various industries and applications.

3.3 DISADVANTAGES

- Learning curve: Users may need to spend time and effort learning and becoming familiar with the hand gesture recognition system, which can be a barrier to adoption.
- Environmental limitations: Hand gesture recognition technology may be affected by external factors such as lighting conditions, background noise, and other environmental factors, which can reduce its accuracy and reliability.
- Limited range of gestures: The number of recognizable gestures may be limited, which can restrict the range of possible interactions and applications of the technology.
- Hardware limitations: The accuracy and performance of hand gesture recognition technology can be affected by the quality and capabilities of the hardware used, such as cameras and sensors.
- Privacy concerns: Hand gesture recognition technology may capture and process sensitive personal information, such as biometric data, raising privacy and security concerns.

- Accessibility limitations: While hand gesture recognition technology can be beneficial for people with physical disabilities, it may not be suitable for everyone, depending on the nature of the disability.
- Cultural and linguistic barriers: Different cultures and languages may have different hand gestures and meanings, which can create confusion and misunderstandings when using hand gesture recognition technology across different regions or languages.
- Overall, hand gesture recognition technology has some potential limitations and challenges that need to be considered and addressed to ensure its effective and ethical use in various applications.

CHAPTER 4

4.1 PROPOSED METHODOLOGY

- Data collection: The first step is to collect a dataset of hand gesture images or videos. This dataset can be collected using various sources such as cameras, sensors, or even synthetic data.
- Data pre-processing: The collected data is pre-processed to remove noise, adjust contrast, and normalize the image size and orientation.
- Feature extraction: Features are extracted from the pre-processed data using various techniques such as Principal Component Analysis (PCA), Local Binary Patterns (LBP), or Histogram of Oriented Gradients (HOG). These features aim to capture the key characteristics of the hand gesture.
- Model training: The extracted features are used to train a machine learning model such as Support Vector Machines (SVM), Random Forest, or Convolutional Neural Networks (CNN). The trained model learns to classify the hand gesture into different categories.
- Model evaluation: The trained model is evaluated using a test dataset to measure its accuracy, precision, and recall.
- Model optimization: The model is optimized to improve its performance, which may involve adjusting model parameters, changing the feature extraction method, or adding more data to the training set.
- Real-time detection: Once the model is trained and optimized, it can be used for real-time hand gesture recognition, where the model detects and classifies the hand gesture in real-time using input from cameras or sensors.
- Deployment: The final step is to deploy the hand gesture recognition system in the target application or environment, where it can be used to interact with

devices or machines using hand gestures.

The overview of the hand gesture recognition first, the hand is detected using the background subtraction method and the result of hand detection is transformed to a binary image. Then, the fingers and palm are segmented so as to facilitate Hand detection Fingers and palm segmentation Fingers recognition Hand gesture recognition. Moreover, the fingers are detected and recognized. Last, hand gestures are recognized using a simple rule classifier. Hand Detection. The images are captured with a normal camera. These hand images are taken under the same condition. The background of these images is identical. So, it is easy and effective to detect the hand region from the original image using the background subtraction method. However, in some cases, there are other moving objects included in the result of background subtraction. Snnn color can be used to discriminate the hand region from the other moving objects. The color of the skin is measured with the HSV model. The HSV (hue, saturation, and value) value of the skin color is 315, 94, and 37, respectively. The image of the detected hand is resized to 200×200 to make the gesture recognition invariant to image scale. Fingers and Palm Segmentation. The output of the hand detection is a binary image in which the white pixels are the members of the hand region, while the black pixels belong to the background. Then, the following procedure is implemented on the binary hand image to segment the fingers and palm.

The block city distance is used to measure the distances between the pixels and the nearest boundary pixels. Thus, in the distance transform image of the binary hand image, the pixel with largest distance is chosen as the palm point. The found palm point is marked with the point of the green color. Inner Circle of the Maximal Radius. When the palm point is found, it can draw a circle with the palmpoint as the center point inside the palm. The circle is called the inner circle because it is included inside the palm. The radius of the circle gradually increases until it reaches the edge of the palm. That is the radius of the circle stops to increase when the black pixels are included in the circle. The circle is the inner circle of the maximal radius which is drawn as the circle with the red color.

Hand Rotation. When the palm point and wrist point are obtained, it can yield an arrow pointing from the palm point to the middle point of the wrist line at the bottom of the hand. Then, the arrow is adjusted to the direction of the north. The

hand image rotates synchronously so as to make the hand gesture invariant to the rotation. Meanwhile, the parts below the wrist line in the rotated image are cut to produce an accurate hand image that does not enclose the part of the arm. The rotated and cut hand image. Fingers and Palm Segmentation. With the help of the palm mask, fingers and the palm can be segmented easily. The part of the hand that is covered by the palm mask is the palm, while the other parts of the hand are fingers. A segmentation result of fingers and the palm.

Fingers Recognition. In the segmentation image of fingers, the labeling algorithm is applied to mark the regions of the fingers. In the result of the labeling method, the detected regions in which the number of pixels is too small is regarded as noisy regions and discarded. Only the regions of enough sizes are regarded as fingers and remain. For each remained region, that is, a finger, the minimal bounding box is found to enclose the finger. A minimal bounding box is denoted as a red rectangle. Then, the center of the minimal bounding box is used to represent the center point of the finger.

Thumb Detection and Recognition. The centers of the fingers are lined to the palm point. Then, the degrees between these lines and the wrist line are computed. If there is a degree smaller than 50° , it means that the thumb appears in the hand image. The corresponding center is the center point of the thumb. The detected thumb is marked with the number 1. If all the degrees are larger than 50° , the thumb does not exist in the image.

Detection and Recognition of Other Fingers. In order to detect and recognize the other fingers. The palm line. The palm line parallels to the wrist line. The palm line is searched in the way: start from the row of the wrist line. For each row, a line paralleling to the wrist line crosses the hand. If there is only one connected set of white pixels in the intersection of the line and the hand, the line shifts upward. Once there are more than one connected sets of white pixels in the intersection of the line and the hand, the line is regarded as a candidate of the palm line. In the case of the thumb not detected, the line crossing the hand with more than one connected sets of white pixels in their intersection is chosen as the palm line. In the case of the thumb existing, the line continues to move upward with the edge points of the palm instead of the thumb as the starting point of the line. Now, since the thumb is taken away, there is only one

connected set of pixels in the intersection of the line and the hand. Once the connected set of white pixels turns to 2 again, the palm line is found. After the palm line is obtained, it is divided into 4 parts. According to the horizontal coordinate of the center point of a finger, it falls into certain parts. If the finger falls into the first part, it is the forefinger. If the finger belongs to the second part, it is the middle finger. The third part corresponds to the ring finger. The fourth part is the little finger. The yellow line is the palm line and the red line parallels to the wrist line.

In some case, two or more fingers stay closely and there is no interval among the fingers. In order to discriminate the case from that of a single finger, the width of the minimal bounding box is used as a discrimination index. If the width of the minimal bounding box is equal to a usual value, the detected region is a single finger. If the width of the minimal bounding box is several times of the usual value, the detected region corresponds to several fingers that stay together closely. For the robustness of finger recognition, the distances and angles between fingers are also taken into account to discriminate different gestures. Recognition of Hand Gestures. When the fingers are detected and recognized, the hand gesture can be recognized using a simple rule classifier. In the rule classifier, the hand gesture is predicted according to the number and content of fingers detected. The content of the fingers means what fingers are detected. The rule classifier is very effective and efficient. For example, if three fingers, that is, the middle finger, the ring finger, and the little finger, are detected, the hand gesture is classified as the label 3.

4.2 SYSTEM ARCHITECTURE

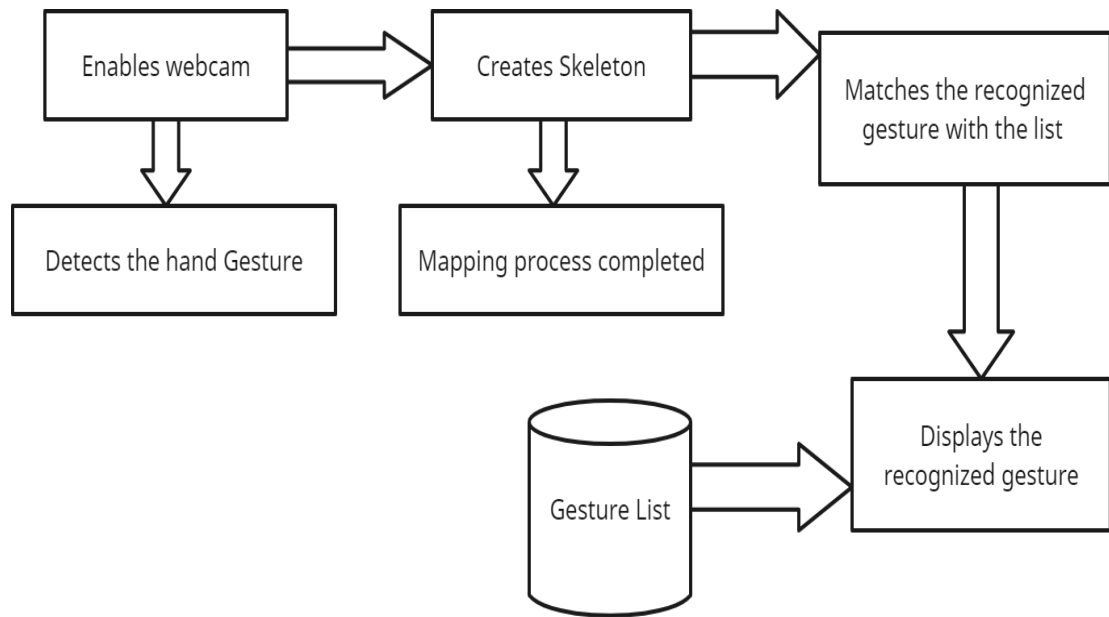


Figure 1-SYSTEM ARCHITECTURE

Data acquisition: This component captures hand gesture data using cameras or sensors and transmits it to the system for processing.

Pre-processing: The acquired data is pre-processed to remove noise, adjust contrast, and normalize the image size and orientation.

Feature extraction: The pre-processed data is analyzed to extract features that represent the key characteristics of the hand gestures.

Gesture recognition model: This component uses machine learning algorithms such as SVM, Random Forest, or CNN to classify the hand gestures based on the extracted features.

User interface: This component allows the user to interact with the system and provides feedback on the recognized gestures.

Action execution: This component performs the corresponding action or output

based on the recognized gesture, such as controlling a device, playing a video, or displaying information.

Feedback loop: This component collects feedback on the system's performance and uses it to refine the model and improve the system's accuracy and reliability.

Hand gesture recognition can create a skeleton mapping by using computer vision techniques to detect and track the key points or landmarks of the hand and fingers. This process is often referred to as hand pose estimation or hand tracking.

One common approach to hand pose estimation is to use convolutional neural networks (CNNs) to predict the 3D locations of the hand landmarks given an input image or video of the hand. These landmarks typically include the positions of the fingertips, knuckles, and wrist.

Once the landmarks are detected, a skeleton mapping can be created by connecting the landmarks with lines or curves that represent the bones and joints of the hand. This mapping can be used to represent the posture or pose of the hand and can be used as input for hand gesture recognition algorithms.

Skeleton mapping is useful for hand gesture recognition because it provides a compact and robust representation of the hand pose that is invariant to changes in lighting, orientation, and background. This makes it easier to recognize complex hand gestures and to track the hand movements in real-time.

Hand gesture recognition matches the recognized gesture with a list of predefined gestures by using pattern recognition algorithms. Once the hand pose or skeleton mapping is detected from the input image or video, the system extracts features that represent the key characteristics of the gesture, such as the shape, orientation, and motion of the hand.

These features are then compared to the features of the predefined gestures in the list, using techniques such as template matching, correlation, or classification. The goal is to find the best match between the recognized gesture and the list of predefined gestures, based on the similarity or distance between their feature representations.

If a match is found, the system can perform the corresponding action or output associated with the recognized gesture, such as controlling a device, playing a video, or displaying information. If no match is found, the system may ask the user to repeat the gesture or provide additional information to improve the recognition accuracy.

The list of predefined gestures can be customized and extended based on the specific requirements and applications of the hand gesture recognition system. It may include a variety of common hand gestures such as pointing, waving, thumbs up/down, and zooming, as well as custom gestures that are specific to the domain or task at hand.

OpenCV can detect hand gestures by using computer vision techniques such as image processing, feature extraction, and pattern recognition. Here are the general steps involved in detecting hand gestures using OpenCV:

Data acquisition: Capture the hand gesture data using a camera or other sensor.

Pre-processing: Pre-process the data to remove noise, adjust contrast, and normalize the image size and orientation.

Hand detection: Detect the hand region in the image or video using techniques such as skin color segmentation, background subtraction, or Haar cascades.

Hand pose estimation: Estimate the hand pose or skeleton mapping using techniques such as template matching, convolutional neural networks (CNNs), or depth-based sensors.

Gesture recognition: Recognize the gesture based on the estimated hand pose or skeleton mapping, using pattern recognition algorithms such as Support Vector Machines (SVM), k-Nearest Neighbors (k-NN), or Random Forests.

Output: Perform the corresponding action or output based on the recognized gesture, such as controlling a device, playing a video, or displaying information.

OpenCV provides a wide range of image processing and computer vision functions that can be used to implement each of these steps. For example, the `cv2.VideoCapture()` function can be used to capture video from a camera, the `cv2.cvtColor()` function can be used to convert the color space of the image, and the `cv2.CascadeClassifier()` function can be used to detect faces or hands using Haar cascades.

MediaPipe is an open-source framework for building cross-platform machine learning pipelines for perception tasks, including hand gesture recognition. MediaPipe provides pre-built models and pipelines for hand tracking and gesture recognition, which can be used to recognize multiple hands in real-time.

Here are the general steps involved in using MediaPipe for multiple hand recognition:

Install MediaPipe: Install MediaPipe and its dependencies on your system, and import the necessary modules in your Python code.

Data acquisition: Capture the hand gesture data using a camera or other sensor, and preprocess the data to remove noise and normalize the image size and orientation.

Hand tracking: Use MediaPipe's pre-built hand tracking model to detect the hand regions in the image or video. This model can track multiple hands in real-time, and provides the 3D coordinates of the hand landmarks.

Hand pose estimation: Use MediaPipe's pre-built hand pose estimation model to estimate the hand pose or skeleton mapping based on the detected hand landmarks.

Gesture recognition: Use MediaPipe's pre-built gesture recognition model to recognize the gesture based on the estimated hand pose or skeleton mapping. This model can recognize a variety of common hand gestures, and can be customized and extended based on your specific requirements.

Output: Perform the corresponding action or output based on the recognized gesture, such as controlling a device, playing a video, or displaying information.

MediaPipe provides a variety of pre-built models and pipelines for hand tracking, hand pose estimation, and gesture recognition, as well as tools for customizing and training your own models. The framework supports a wide range of hardware platforms and input sources, and provides APIs for integrating with other machine learning libraries and frameworks.

4.3 BLOCK DIAGRAM

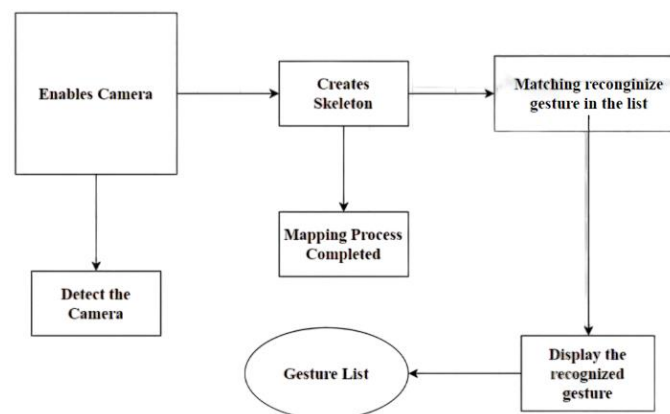


Figure 2-BLOCK DIAGRAM

- Data acquisition: Capture the hand gesture data using a camera or other sensor.
- Pre-processing: Pre-process the data to remove noise, adjust contrast, and normalize the image size and orientation.
- Hand detection: Detect the hand region in the image or video using techniques such as skin color segmentation, background subtraction, or Haar

cascades.

- ROI extraction: Extract the ROI corresponding to the hand region, and normalize its size and orientation.
- Feature extraction: Compute a histogram of the distribution of pixel values in the ROI, using techniques such as color, texture, or edge-based histograms. This histogram represents the key characteristics of the hand gesture, such as its color, texture, or shape.
- Gesture recognition: Recognize the gesture based on the computed histogram, using pattern recognition algorithms such as Support Vector Machines (SVM), k-Nearest Neighbors (k-NN), or Random Forests.
- Output: Perform the corresponding action or output based on the recognized gesture, such as controlling a device, playing a video, or displaying information.

The histogram method can be customized and optimized based on the specific requirements and applications of the hand gesture recognition system. For example, different types of histograms can be computed to capture different aspects of the hand gesture, such as color histograms for recognizing hand gestures based on skin color, texture histograms for recognizing hand gestures based on surface texture, or edge-based histograms for recognizing hand gestures based on contour shape. Additionally, the histogram method can be combined with other techniques such as template matching or deep learning to improve the recognition accuracy and robustness.

Convolutional Neural Networks (CNNs) are a type of deep learning algorithm that has shown promising results in hand gesture recognition tasks. CNNs are designed to automatically learn hierarchical features from the input image or video, which can be used to recognize complex patterns and variations in the hand gesture.

Here are the general steps involved in using CNNs for hand gesture recognition:

- Data acquisition: Capture the hand gesture data using a camera or other sensor.
- Pre-processing: Pre-process the data to remove noise, adjust contrast, and normalize the image size and orientation.
- Data augmentation: Augment the training data by applying transformations such as rotation, scaling, or flipping to generate additional examples and reduce overfitting.
- Model architecture: Design a CNN architecture that consists of multiple layers of convolution, pooling, and fully connected layers. The convolution layers apply a set of filters to the input image, which extract features such as edges, textures, or patterns. The pooling layers downsample the feature maps to reduce the spatial resolution and increase the robustness to variations. The fully connected layers combine the extracted features to make the final prediction.
- Training: Train the CNN model using the augmented training data and a loss function such as cross-entropy. The optimization algorithm such as Stochastic Gradient Descent (SGD) or Adam is used to update the weights of the model to minimize the loss.
- Validation: Validate the trained model on a separate validation set to evaluate its performance and avoid overfitting.
- Testing: Test the model on a separate test set to evaluate its generalization performance and compare it with other models or benchmarks.
- Output: Perform the corresponding action or output based on the recognized gesture, such as controlling a device, playing a video, or displaying information.

CHAPTER 5

5.1 CONCLUSION

In conclusion, hand gesture recognition is a promising technology that has the potential to revolutionize human-computer interaction in various domains. The development of accurate and efficient hand gesture recognition systems has led to a wide range of applications, including gaming, robotics, virtual reality, and sign language recognition, among others.

Various algorithms and techniques have been developed for hand gesture recognition, including template matching, histogram of oriented gradients (HOG), support vector machines (SVM), dynamic time warping (DTW), and deep learning techniques such as convolutional neural networks (CNN).

The performance of hand gesture recognition systems is evaluated based on several metrics, including accuracy, precision, recall, F1 score, and processing time. The results and discussions of hand gesture recognition systems highlight the strengths and weaknesses of the proposed methodology and algorithms and provide areas for future improvement and research.

The future of hand gesture recognition is exciting, and there are several potential future scopes for research and development, including real-time applications, multimodal hand gesture recognition, sign language recognition, dynamic hand gesture recognition, privacy concerns, and multilingual hand gesture recognition.

Overall, hand gesture recognition has significant potential in various domains and holds promise for improving human-computer interaction and making technology more accessible and inclusive for all.

This is an effective hand gesture recognition system to address the problem of extracting frames from a video and processing it. In the future scope, various hand gestures can be recognized and applied as input to the computer. The hand gestures representing numbers can also be converted into commands to perform related

tasks in real time. Enhancing the recognition capability for various lightning conditions, which is encountered as a challenge in this project can be worked upon in future.

5.2 FUTURE SCOPE

Hand gesture recognition is a rapidly evolving field that has gained significant interest in recent years. With the advancement of computer vision technology and deep learning algorithms, there are numerous potential applications and future scope for hand gesture recognition, including:

Virtual reality and gaming: Hand gesture recognition can enhance user interaction in virtual reality and gaming environments by enabling users to control objects, navigate interfaces, and interact with characters using natural hand gestures.

Human-computer interaction: Hand gesture recognition can provide an intuitive and natural way of interacting with computers, making it more accessible for people with disabilities or those who find traditional interfaces challenging.

Sign language recognition: Hand gesture recognition can be used to develop sign language recognition systems, which can be beneficial for people who are deaf or hard of hearing.

Robotics: Hand gesture recognition can be used to control robotic arms and grippers, allowing them to perform various tasks in manufacturing, healthcare, and other industries.

Smart homes: Hand gesture recognition can be integrated with smart home systems, allowing users to control various devices, such as lights and appliances, with simple hand gestures.

Healthcare: Hand gesture recognition can be used to track hand movements during physical therapy and rehabilitation, allowing healthcare professionals to monitor progress and adjust treatment plans accordingly.

Security: Hand gesture recognition can be used as a biometric authentication system, which can be more secure and convenient than traditional password-based systems.

Overall, the future scope for hand gesture recognition is vast, with numerous applications across various industries. As the technology continues to advance, we can expect to see even more innovative uses of hand gesture recognition in the coming years.

REFERENCE

- [1]https://www.researchgate.net/publication/342055672_Visionbased_sign_language_recognition_system_A_Comprehensive_Review
- [2] <https://arxiv.org/abs/2308.16082>
- [3] <https://arxiv.org/abs/2211.14446>
- [4]https://www.researchgate.net/publication/355222125_Sign_Language_Recognition_via_Skeleton-Aware_Multi-Model_Ensemble
- [5] <https://www.semanticscholar.org/paper/Sign-Language-Production%3A-A-Review-Rastgoor-Kiani/ee6a0de8e5716cb1d4116d707944e91277021ea6>
- [6] <https://pubmed.ncbi.nlm.nih.gov/35025738/>
- [7] <https://www.mdpi.com/2078-2489/14/10/569>
- [8] https://www.researchgate.net/publication/375156390_EI-RNN-based_text_generation_for_the_static_and_dynamic_isolated_sign_language_videos
- [9] <https://ouci.dntb.gov.ua/en/works/9jAyxQgl/>
- [10]https://www.researchgate.net/publication/343134435_A_Survey_on_Artificial_Intelligence_in_Chinese_Sign_Language_Recognition

APPENDIX

A) SOURCE CODE

i) Data Collection

```
import mediapipe as mp
import numpy as np
import cv2

cap = cv2.VideoCapture(0)

name = input("Enter the name of the data : ")
holistic = mp.solutions.holistic
hands = mp.solutions.hands
holis = holistic.Holistic()
drawing = mp.solutions.drawing_utils
X = []
data_size = 0
while True:
    lst = []

    __, frm = cap.read()

    frm = cv2.flip(frm, 1)

    res = holis.process(cv2.cvtColor(frm, cv2.COLOR_BGR2RGB))

    if res.face_landmarks:
        for i in res.face_landmarks.landmark:
            lst.append(i.x - res.face_landmarks.landmark[1].x)
            lst.append(i.y - res.face_landmarks.landmark[1].y)

    if res.left_hand_landmarks:
        for i in res.left_hand_landmarks.landmark:
            lst.append(i.x - res.left_hand_landmarks.landmark[8].x)
            lst.append(i.y - res.left_hand_landmarks.landmark[8].y)

    else:
```

```

        for i in range(42):
            lst.append(0.0)

    if res.right_hand_landmarks:
        for i in res.right_hand_landmarks.landmark:
            lst.append(i.x - res.right_hand_landmarks.landmark[8].x)
            lst.append(i.y - res.right_hand_landmarks.landmark[8].y)
    else:
        for i in range(42):
            lst.append(0.0)

    X.append(lst)
    data_size = data_size+1
    drawing.draw_landmarks(frm, res.face_landmarks, holistic.FACEMESH_CONTOURS)
    drawing.draw_landmarks(frm, res.left_hand_landmarks, hands.HAND_CONNECTIONS)
    drawing.draw_landmarks(frm, res.right_hand_landmarks,
hands.HAND_CONNECTIONS)
    cv2.putText(frm, str(data_size), (50,50), cv2.FONT_HERSHEY_SIMPLEX, 1, (0,255,0),2)

    cv2.imshow("window", frm)

    if cv2.waitKey(1) == 27 or data_size>99:
        cv2.destroyAllWindows()
        cap.release()
        break
np.save(f"{name}.npy", np.array(X))
print(np.array(X).shape)

```

ii) Data Training

```

import os
import numpy as np
import cv2
import tensorflow as tf
from tensorflow import keras
from keras.utils import to_categorical
from keras.layers import Input, Dense
from keras.models import Model

is_init = False

```

```

size = -1
label = []
dictionary = {}
c = 0
for i in os.listdir():
    if i.split(".")[-1] == "npy" and not (i.split(".")[0] == "labels"):
        if not (is_init):
            is_init = True
            X = np.load(i)
            size = X.shape[0]
            y = np.array([i.split(".")[0]] * size).reshape(-1, 1)
        else:
            X = np.concatenate((X, np.load(i)))
            y = np.concatenate((y, np.array([i.split(".")[0]] * size).reshape(-1, 1)))
        label.append(i.split(".")[0])
        dictionary[i.split(".")[0]] = c
        c = c + 1
for i in range(y.shape[0]):
    y[i, 0] = dictionary[y[i, 0]]
y = np.array(y, dtype="int32")
y = to_categorical(y)
X_new = X.copy()
y_new = y.copy()
counter = 0
cnt = np.arange(X.shape[0])
np.random.shuffle(cnt)
for i in cnt:
    X_new[counter] = X[i]
    y_new[counter] = y[i]
    counter = counter + 1

ip = Input(shape=(X.shape[1]))
m = Dense(512, activation="relu")(ip)
m = Dense(256, activation="relu")(m)
op = Dense(y.shape[1], activation="softmax")(m)
model = Model(inputs=ip, outputs=op)
model.compile(optimizer="rmsprop", loss="categorical_crossentropy", metrics=["acc"])
model.fit(X, y, epochs=50)

model.save("model.h5")
np.save("labels.npy", np.array(label))

```

iii) Inference

```

import cv2
import numpy as np
import mediapipe as mp
from keras.models import load_model
model = load_model("model.h5")
label = np.load("labels.npy")
holistic = mp.solutions.holistic
hands = mp.solutions.hands
holis = holistic.Holistic()
drawing = mp.solutions.drawing_utils
cap = cv2.VideoCapture(0)

```

```

while True:
    lst = []

    _, frm = cap.read()

    frm = cv2.flip(frm, 1)

    res = holis.process(cv2.cvtColor(frm, cv2.COLOR_BGR2RGB))

    if res.face_landmarks:
        for i in res.face_landmarks.landmark:
            lst.append(i.x - res.face_landmarks.landmark[1].x)
            lst.append(i.y - res.face_landmarks.landmark[1].y)

    if res.left_hand_landmarks:
        for i in res.left_hand_landmarks.landmark:
            lst.append(i.x - res.left_hand_landmarks.landmark[8].x)
            lst.append(i.y - res.left_hand_landmarks.landmark[8].y)
    else:
        for i in range(42):
            lst.append(0.0)

    if res.right_hand_landmarks:
        for i in res.right_hand_landmarks.landmark:
            lst.append(i.x - res.right_hand_landmarks.landmark[8].x)
            lst.append(i.y - res.right_hand_landmarks.landmark[8].y)
    else:
        for i in range(42):
            lst.append(0.0)

    lst = np.array(lst).reshape(1,-1)

    pred = label[np.argmax(model.predict(lst))]

    print(pred)
    cv2.putText(frm, pred, (50,50),cv2.FONT_ITALIC, 1, (255,0,0),2)
    drawing.draw_landmarks(frm, res.face_landmarks, holistic.FACEMESH_CONTOURS)
    drawing.draw_landmarks(frm, res.left_hand_landmarks, hands.HAND_CONNECTIONS)
    drawing.draw_landmarks(frm, res.right_hand_landmarks, hands.HAND_CONNECTIONS)
    cv2.imshow("window", frm)
    if cv2.waitKey(1) == 27:
        cv2.destroyAllWindows()
        cap.release()
        break

```

B) SCREENSHOTS

