**A Project Report**

**On**

**SIGN LANGUAGE INTERPRETER**

**Submitted to**

**Department of Computer Science**

**CHAITANYA (DEEMED TO BE UNIVERSITY)**

In the partial fulfillment of the requirement for the III year I semester

**B.sc (Computer Science with Cognitive Systems)**

By

**SAI VINDHYA TENNETI (120106008)**

**E. RITHVIK (120160028)**

**K.ARJUN REDDY (120106048)**

Under the guidance of

**Dr.V. Ramu**

Assistant Professor



DEPARTMENT OF COMPUTER SCIENCE

**CHAITANYA DEEMED TO BE UNIVERSITY**

Kishanpura, Hanamkonda, Warangal (T.S)-506002

2022-2023

**CHAITANYA (DEEMED TO BE UNIVERSITY)**

Kishanpura, Hanamkonda, Warangal (T.S)-506002

**DEPARTMENT OF COMPUTER SCIENCE**

****

**CERTIFICATE**

This is to certify that **SAI VINDHYA TENNETI** bearing HTNO: **120106008**, **E. RITHVIK** bearing HTNO: **120106028**, **K. ARJUN REDDY** bearing HTNO: **120106048** at Chaitanya (Deemed to be University) have Satisfactorily completed the project entitled “SIGN LANGUAGE INTERPRETER” in the Partial fulfillment of the requirements for the award of Degree B.Sc. (CS) with Cognitive Systems during the academic year 2022-2023

GUIDE HEAD DEAN ADMINSTRATION

Dr.V.Ramu Dr.A.Ramesh Babu Dr.S.Kavitha **Assistant Professor Professor Professor**

**Internal Examiner External Examiner**

**DECLARATION**

We hereby declare that project report titled **“SIGN LANGUAGE INTERPRETER”** is an original work done at **Chaitanya (Deemed to be University)**, Hanamkonda, Warangal, submitted in partial fulfillment for the award of BSC(CS) With Cognitive Systems III year I semester, to the department of Computer Science. We assure you that this project has not been submitted by any degree anywhere in this college or university.

**SAI VINDHYA TENNETI (120106008)**

**E. RITHVIK (120160028)**

**K.ARJUN REDDY (120106048)**

**ACKNOWLEDGEMENT**

We express our sincere gratitude to **Dr. Ch. V. Purushotham Reddy,** **Founder&Chancellor**, **Prof. B.S. Kavitha,** **Dean of Administration** **CHAITANYA (Deemed to be University)** for their unstained interest in providing me facilities to complete the project successfully.

We thank **Prof. A. Ramesh Babu,** Professor & Head of the Computer Science, for encouraging us to do such project work and providing all the facilities to carry out this Project work.

We extend our special thanks to our guide **Dr.V.Ramu**, Assistant Professor, who helped immensely with his guidance and valuable suggestions during the completion of the project work and who allowed us to do the Project work and gave us lot of support and constant encouragement throughout this project. We experience from this kind of work is great and will be useful to us in future.

We thank all the faculty members of the Department of Computer Science for sharing their valuable knowledge with us. We extend our thanks to the Technical Staff of the department for their valuable suggestions to technical problems. We also thank our family, friends for their hand in bringing this project to this present shape.

**SAI VINDHYA TENNETI (120106008)**

**E. RITHVIK (120160028)**

**K.ARJUN REDDY (120106048)**

**CONTENTS**

1. Abstract
2. Introduction
   1. Existing system
   2. Proposed system
   3. Vision Based Approach
3. System Requirements
   1. Software Requirements
   2. Hardware Requirements
   3. Functional Requirements
   4. Non-Functional Requirements
4. System Design
5. Class Diagram
6. Use case Diagram
7. Architecture of system
8. Sample screens and sample code
9. Future Enhancements
10. Conclusion

**ABSTRACT**

Communication. This is particularly significant in social settings. We use communication for a number of purposes, including asking and telling. The capacity to openly communicate may not come naturally to those who are deaf or mute. They should still be able to communicate with others despite this. As a result, sign languages were explained to us. The use of sign languages improves communication. However, not everyone may be fluent with sign language. The sign language interpreter, enters the scene. This aids in deciphering a sign language user's intended message. It uses the webcam to take a picture and attempts to understand what the user is trying to convey through text. The study offers a novel method for the deaf and dumb society to recognize hand signs.

**INTRODUCTION**

For those who cannot speak, sign language serves as their primary form of communication. It conveys their message through gestures. Sign language is a method of communicating using hand shapes, hand orientation, hand, arm, or body movement, facial expressions, and lip patterns. Some of the most common sign languages in the world are American Sign Language (ASL), British Sign Language (BSL), Indian Sign Language (ISL), and others. Communication is a means of exchanging thoughts and feelings that should not be limited by the use of words or speech. Sign language interpreters (SLI) must translate sign language into spoken language.

Sign interpreters have been developed in the last decade and have gained significant importance in the field of research and development. Several hand-movement trackers have been developed over the years, each with its own set of performance parameters such as accuracy, jitter, drift, and latency.

This application is another step towards this mission. One of its benefits is the ease with which it can be used. A person's sign can be interpreted by us with the fewest possible steps, making it straightforward for everyone to grasp what the former is attempting to communicate.

Since American Sign Language is a natural language, it is used by Deaf communities worldwide as a common language even when other cultures have their own SLs. ASL is a comprehensive and well-structured visual language that uses both manual and non-manual characteristics to communicate. Therefore in our project, we included some of such words.

**EXISTING SYSTEM**

The use of sensors like optical fibres and flex sensors to track the movement of the palm and fingers for mapping different characters is one of the standard automated sign interpretation approaches.

For Example, A microcontroller based on the Arduino Uno serves as the main controlling element of a general-purpose sign interpreter. Wearable gloves are the suggested sign language interpreter device. To record hand gestures and hand movements, interpreter gloves feature a microcontroller and several other sensors. Through the Bluetooth module, the two gloves connect with one another to create a variety of indications that are then relayed through the speakers.

This type of system requires a significant amount of hardware and software, which is impossible to transport everywhere. This makes achieving our initial objective difficult.

**PROPOSED SYSTEM**

Sign language recognition is the process of translating the user's motions and signs into text. It fills the communication gap between the general public and those who cannot speak. Raw photos or videos are transformed into relevant text that can be read and understood using image processing algorithms and neural networks to map the gesture to the proper text in the training data.

A dependable hand and finger tracking gadget is Mediapipe Hands. With just one frame, it can recognise 21 local hand marks in three dimensions using machine learning (ML). Although most contemporary approaches rely heavily on the robust desktop computers as the discovery locations, our method advantages from real-time performance on mobile phones and even scalability to several hands.

The interpreter uses a camera to detect the signs we are using mediapipe module. The window that opens can be directed to the interpreter. It uses the mediapipe module to mark the coordinates on hand and starts recognizing the signs.

**VISION BASED APPROACH**

This method records images on the camera as touch data. The vision-based approach focuses heavily on touch-captured images and highlights the most prominent and distinguishing feature.

At the start of the vision-based approach, colour belts were used. The standard colour to be applied to the fingers was the main disadvantage of this method. Then, instead of using coloured ribbons, use your bare hands. This presents a difficult problem because these systems require background, continuous lighting, personal frames, and a camera to achieve real-time performance. Furthermore, such systems must be designed to meet the requirements, which include accuracy and robustness.

Although theoretical analysis is based on how people perceive information about their surroundings, it is likely the most difficult to apply effectively. So far, several methods have been tested. The first step is to construct a three-dimensional human hand model. The model is compared to hand images captured by one or more cameras, and the parameters corresponding to the palm shape and combined angles are estimated. The touch phase is then created using these parameters. The second method is to take a picture with the camera and extract specific features, which are then used as input in the partition algorithm to separate.

**SYSTEM REQUIREMENTS**

All computer software requires specific hardware parts or other software resources to function properly on a computer. These prerequisites, also referred to as (computer) system requirements, are frequently employed as a general norm rather than an unbreakable law. The majority of software specifies minimal and recommended sets of system requirements. System requirements tend to rise over time due to the rising demand for more processing power and resources in newer versions of software. According to industry observers, this trend more than technological improvements is what is pushing changes to current computer systems.

A generalization of the first concept, known as "system requirements," refers to the specifications that must be met while designing a system or subsystem.

**SOFTWARE REQUIREMENTS**

* **Python**

A high-level, all-purpose programming language is Python. Code readability is prioritized in its design philosophy, which makes heavy use of indentation. Python uses garbage collection and has dynamic typing. It supports a variety of paradigms for programming, including functional, object-oriented, and structured programming.

* Python can be used on a server to create web applications.
* Python can be used alongside software to create workflows.
* Python can connect to database systems. It can also read and modify files.
* Python can be used to handle big data and perform complex mathematics.
* Python can be used for rapid prototyping, or for production-ready software development.
* **Machine Learning**

Machine learning (ML) is a topic of study focused on comprehending and developing "learning" methods, or methods that use data to enhance performance on a certain set of tasks. It is considered to be a component of artificial intelligence. Without being expressly taught to do so, machine learning algorithms create a model using sample data, also referred to as training data, in order to make predictions or judgments. Speech recognition, email filtering, computer vision, and other fields where it is impossible or impractical to create traditional algorithms for the required tasks, among many others, use machine learning methods.

Machine learning is significant because it aids in the development of new goods and provides businesses with a picture of trends in consumer behavior and operational business patterns. A significant portion of the operations of many of today's top businesses, like Facebook, Google, and Uber, revolve around machine learning. For many businesses, machine learning has emerged as a key competitive differentiation.

* **MediaPipe**

MediaPipe is a cross-platform pipeline framework that allows you to create unique machine learning solutions for live and streaming video. Google has open-sourced the framework, which is presently in alpha.

These components in computer vision pipelines include model inference, media processing algorithms, data transformations, and so on. Sensory data, such as video streams, enter the graph, while perceived descriptions, such as object-localization or face-keypoint streams, exit the graph.

The MediaPipe framework is primarily intended for quick development of perception pipelines that include reusable parts and AI models for inferencing. Additionally, it makes it easier to integrate computer vision software into applications and demos running on many hardware platforms. Teams may gradually enhance computer vision pipelines thanks to the configuration language and evaluation tools.

* **Open CV**

Computer vision is a method that enables us to comprehend how images and videos are stored, how to change them, and how to extract data from them. The foundation or primary tool utilised in artificial intelligence is computer vision. Self-driving cars, robotics, and photo-editing apps all heavily rely on computer vision.

A set of programming tools called OpenCV is primarily focused on real-time computer vision. It was initially created by Intel and then backed by Willow Garage and Itseez.

OpenCV is a significant open-source library for image processing, machine learning, and computer vision. It now plays a significant part in real-time operation, which is crucial in modern systems. Using it, one may analyse pictures and movies to find faces, objects, and even human handwriting. Python is able to handle the OpenCV array structure for analysis when it is integrated with different libraries, such as NumPy. We use vector space and apply mathematical operations to these features to identify visual patterns and their various features.

OpenCV's initial release was 1.0. OpenCV is free for both academic and commercial use because it is distributed under a BSD licence. It supports Windows, Linux, and Python and offers C++, C, and Java interfaces.

* **Neural Networks**

Deep learning techniques are based on neural networks, sometimes referred to as artificial neural networks (ANNs) or simulated neural networks (SNNs), which are a subset of machine learning. Their structure and nomenclature are modelled after the human brain, mirroring the communication between organic neurons.

A node layer of an artificial neural network (ANN) consists of an input layer, one or more hidden layers, and an output layer. Each node, or artificial neuron, is connected to others and has a weight and threshold that go along with it. Any node whose output exceeds the defined threshold value is activated and begins providing data to the network's uppermost layer. If not, no information is transferred to the next tier of the network.

**HARDWARE REQUIREMENTS**

* Windows computer or Linux, Python installed and Libraries.
* CMOS sensor (Webcam)
* Hand Touch for Visibility Computer Software We Used to Recognize Project Signature Recognition
* CPU - Intel core i5 9th Gen.
* GPU - Nvidia GTX 1050 Ti.
* Pycharm IDE/Visual Studio code
* Anaconda navigator

**FUNCTIONAL REQUIREMENTS**

A Functional Requirements Specification describes what is required to meet the users' business needs. Functional requirements specify which actions must be provided by the design in order for the system's users to benefit. The current system's functional requirements are determined by an analysis of its needs, users, and tasks.

* A good quality working camera.
* Any one of the users must have knowledge on the Sign Language.
* It describes the function of a system or its components as a specification of behavior between inputs and outputs.

**NON FUNCTIONAL REQUIREMENTS**

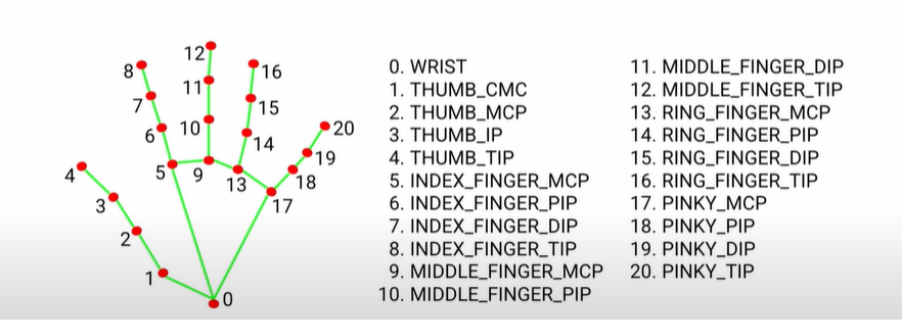
Non-functional requirements are those that have no direct bearing on the particular services that the system provides. Performance requirements and software quality attributes are among the non-functional criteria necessary for this sign language interpreter.

* Interpretation speed
* Real Time Training
* Immediate Results

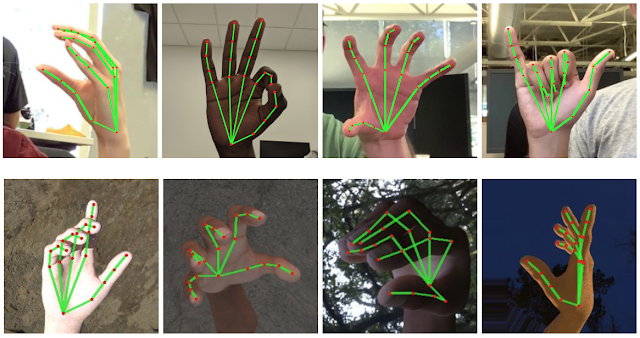
**SYSTEM DESIGN**

This project's System Design steps are as follows:

A machine learning pipeline is used by Mediapipe Hands to combine various coworking models: A fixed hand-held binding box is produced by a palm-type acquisition model that operates in entire images. A handwriting model that restores 3D accurate key points from a clipped image position determined by a palm detector.



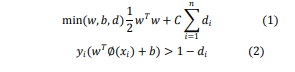
Following palm detection across the entire image, the subsequent hand landmark model uses regression to perform precise keypoint localization of 21 3D hand-knuckle coordinates within the detected hand regions, which is direct coordinate prediction. The model learns a consistent internal hand pose representation and is resistant to partially visible hands as well as self-occlusions.



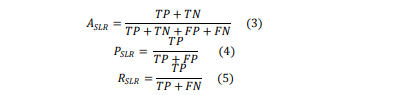
**Using a machine learning algorithm for prediction**

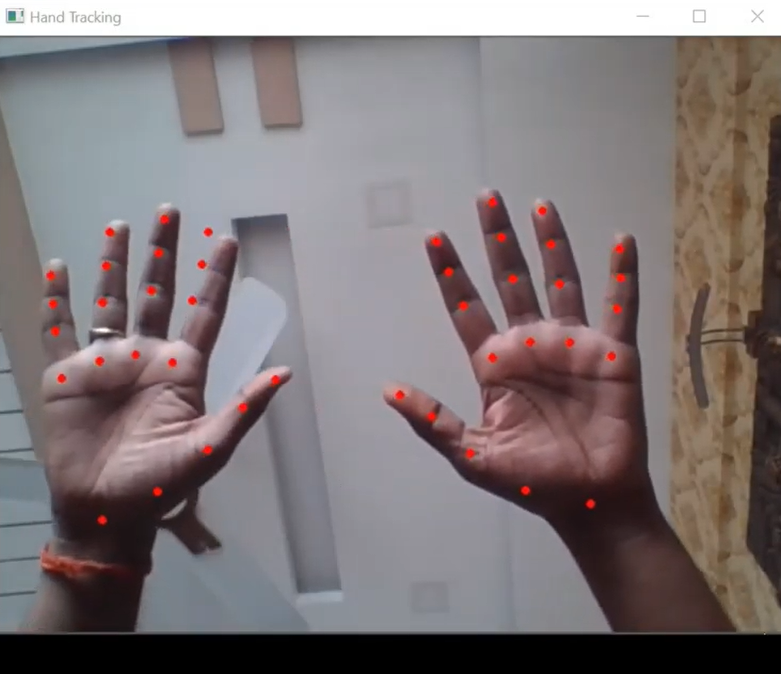
Support Vector Machine (SVM) performed better than other algorithms in a predictive analysis of various sign languages using machine learning algorithms. SVM works well in environments with several dimensions. SVM functions well when the number of samples exceeds the number of dimensions. A group of supervised learning techniques known as SVM are capable of classifying data, regressing it, and identifying outliers.

The optimization issue that SVMs try to solve is represented by the following formula:

****

*di* represents the distances to the correct margin in equations (1) and (2) with di>= 0, I = 1,..., n, C representing a regularisation parameter, wTw representing the normal vector, xi representing the transformed input space vector, b representing a bias parameter, and yi representing the i-th target value. By maximising the margin from the Support Vectors to the hyperplane and decreasing the term wTw, the goal is to accurately categorise as many data points as feasible. RBF (radical basis function) is the chosen kernel function, which transforms the input space into a higher-dimensional space without explicitly mapping each data point. When there is a large gap between classes, SVM performs comparatively well. Consequently, we classified various classes of sign language alphabets using SVM.



****

**CLASS DIAGRAM**

Static diagrams include class diagrams. It represents the application's static view. Class diagrams are used to create executable code for software applications as well as for visualizing, explaining, and documenting various elements of systems.

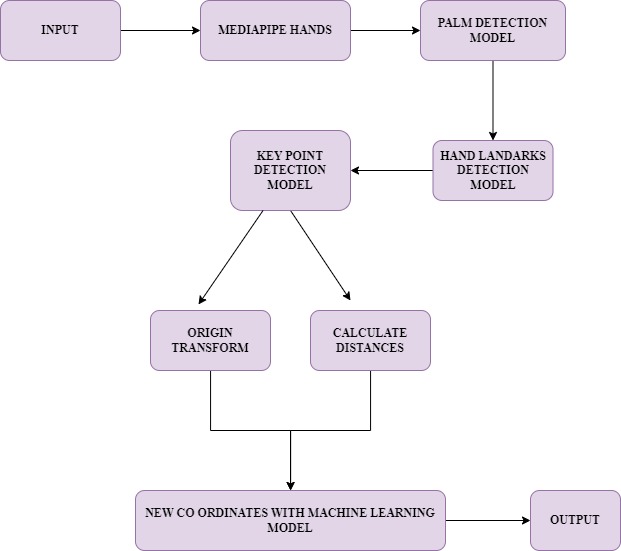
The characteristics and functions of a class are described in a class diagram, along with the restrictions placed on the system. Because they are the only UML diagrams that can be directly mapped with object-oriented languages, class diagrams are frequently utilized in the modeling of object-oriented systems.

A collection of classes, interfaces, affiliations, collaborations, and constraints are displayed in a class diagram. Another name for it is a structural diagram.

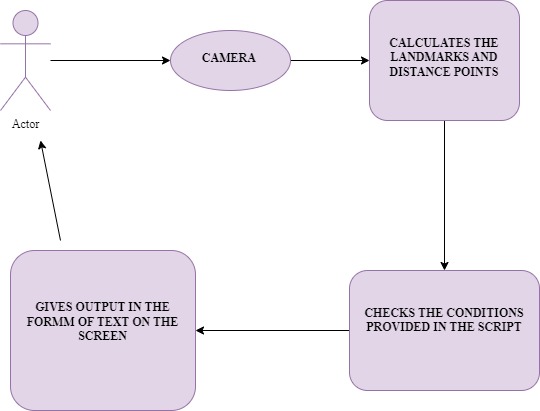
The class diagram's goal can be summed up as

Analysis and design of an application's static view.

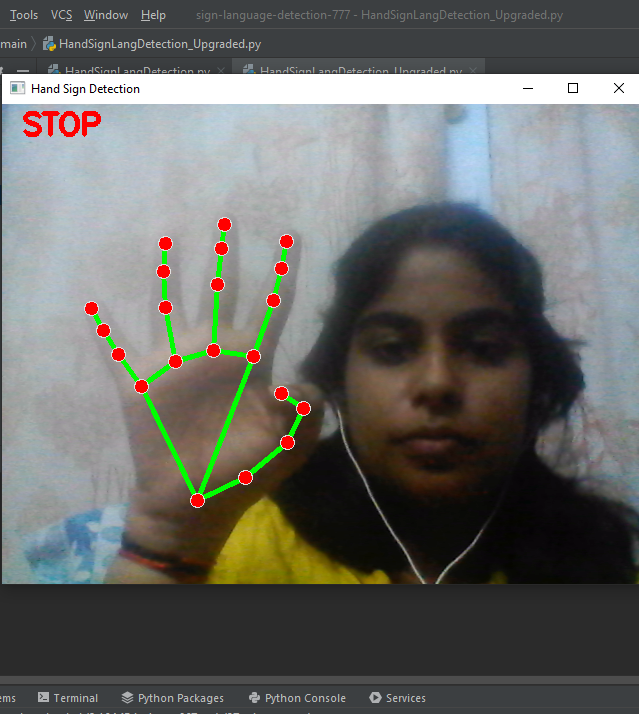
* Describe a system's obligations.
* Component and deployment diagrams' starting point.
* Both forward and backward engineering.

****

**USE CASE DIAGRAM**



**SAMPLE SCREEN**

****

****

**SAMPLE CODE**

import cv2  
import mediapipe as mp  
  
mp\_hands = mp.solutions.hands  
hands = mp\_hands.Hands()  
mp\_draw = mp.solutions.drawing\_utils  
cap = cv2.VideoCapture(0)  
  
finger\_tips = [8, 12, 16, 20]  
thumb\_tip = 4  
  
# like\_img = cv2.imread("images/like.png")  
# like\_img = cv2.resize(like\_img, (200, 180))  
#  
# dislike\_img = cv2.imread("images/dislike.png")  
# dislike\_img = cv2.resize(dislike\_img, (200, 180))  
  
while True:  
 ret, img = cap.read()  
 img = cv2.flip(img, 1)  
 h, w, c = img.shape  
 results = hands.process(img)  
  
 if results.multi\_hand\_landmarks:  
 for hand\_landmark in results.multi\_hand\_landmarks:  
 lm\_list = []  
 for id, lm in enumerate(hand\_landmark.landmark):  
 lm\_list.append(lm)  
 finger\_fold\_status = []  
 for tip in finger\_tips:  
 x, y = int(lm\_list[tip].x \* w), int(lm\_list[tip].y \* h)  
 # print(id, ":", x, y)  
 #cv2.circle(img, (x, y), 15, (255, 0, 0), cv2.FILLED)  
  
 if lm\_list[tip].x < lm\_list[tip - 2].x:  
 #cv2.circle(img, (x, y), 15, (0, 255, 0), cv2.FILLED)  
 finger\_fold\_status.append(True)  
 else:  
 finger\_fold\_status.append(False)  
  
 print(finger\_fold\_status)  
  
 x, y = int(lm\_list[8].x \* w), int(lm\_list[8].y \* h)  
 print(x, y)  
  
 # stop  
 if lm\_list[4].y < lm\_list[2].y and lm\_list[8].y < lm\_list[6].y and lm\_list[12].y < lm\_list[10].y and \  
 lm\_list[16].y < lm\_list[14].y and lm\_list[20].y < lm\_list[18].y and lm\_list[17].x < lm\_list[0].x < \  
 lm\_list[5].x:  
 cv2.putText(img, "STOP", (20, 30), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (0, 0, 255), 3)  
 print("STOP")  
  
 # Forward  
 if lm\_list[3].x > lm\_list[4].x and lm\_list[8].y < lm\_list[6].y and lm\_list[12].y > lm\_list[10].y and \  
 lm\_list[16].y > lm\_list[14].y and lm\_list[20].y > lm\_list[18].y:  
 cv2.putText(img, "FORWARD", (20, 30), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (0, 0, 255), 3)  
 print("FORWARD")  
  
 # Backward  
 if lm\_list[3].x > lm\_list[4].x and lm\_list[3].y < lm\_list[4].y and lm\_list[8].y > lm\_list[6].y and lm\_list[12].y < lm\_list[10].y and \  
 lm\_list[16].y < lm\_list[14].y and lm\_list[20].y < lm\_list[18].y:  
 cv2.putText(img, "BACKWARD", (20, 30), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (0, 0, 255), 3)  
 print("BACKWARD")  
  
 # Left  
 if lm\_list[4].y < lm\_list[2].y and lm\_list[8].x < lm\_list[6].x and lm\_list[12].x > lm\_list[10].x and \  
 lm\_list[16].x > lm\_list[14].x and lm\_list[20].x > lm\_list[18].x and lm\_list[5].x < lm\_list[0].x:  
 cv2.putText(img, "LEFT", (20, 30), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (0, 0, 255), 3)  
 print("LEFT")  
  
 # Right  
 if lm\_list[4].y < lm\_list[2].y and lm\_list[8].x > lm\_list[6].x and lm\_list[12].x < lm\_list[10].x and \  
 lm\_list[16].x < lm\_list[14].x and lm\_list[20].x < lm\_list[18].x:  
 cv2.putText(img, "RIGHT", (20, 30), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (0, 0, 255), 3)  
 print("RIGHT")  
  
 if all(finger\_fold\_status):  
 # like  
 if lm\_list[thumb\_tip].y < lm\_list[thumb\_tip - 1].y < lm\_list[thumb\_tip - 2].y and lm\_list[0].x < lm\_list[3].y:  
 print("LIKE")  
 cv2.putText(img, "LIKE", (20, 30), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (0, 255, 0), 3)  
 # h, w, c = like\_img.shape  
 # img[35:h + 35, 30:w + 30] = like\_img  
 # Dislike  
 if lm\_list[thumb\_tip].y > lm\_list[thumb\_tip - 1].y > lm\_list[thumb\_tip - 2].y and lm\_list[0].x < \  
 lm\_list[3].y:  
 cv2.putText(img, "DISLIKE", (20, 30), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (0, 0, 255), 3)  
 print("DISLIKE")  
 # h, w, c = dislike\_img.shape  
 # img[35:h + 35, 30:w + 30] = dislike\_img  
  
  
 mp\_draw.draw\_landmarks(img, hand\_landmark,  
 mp\_hands.HAND\_CONNECTIONS,  
 mp\_draw.DrawingSpec((0, 0, 255), 6, 3),  
 mp\_draw.DrawingSpec((0, 255, 0), 4, 2)  
 )  
  
 cv2.imshow("Hand Sign Detection", img)  
 cv2.waitKey(1)

**CONCLUSION**

Applications today require a variety of images as sources of data for clarification and analysis. To carry out a variety of applications, a number of features must be extracted. Degradation happens when a picture is changed from one form to another, such as when digitizing, scanning, sharing, storing, etc. As a result, the final image must go through a process known as image enhancement, which consists of a collection of techniques meant to increase an image's visual prominence. Fundamentally, image enhancement improves the readability or awareness of information in images for human listeners while also giving other autonomous image processing systems better input. The image is then subjected to feature extraction utilizing a variety of techniques to improve the image's computer readability.

A powerful tool to prepare expert knowledge, discover edges, and combine erroneous information from several sources is a sign language recognition system.

**FUTURE ENHANCEMENTS**

The suggested sign language recognition system can be expanded to recognize gestures and facial expressions in addition to sign language letters. Sentences will be displayed as a more proper translation of language rather than letter labels, which is more appropriate. This improves readability as well. The range of several sign languages can be expanded. The letter can be detected with greater accuracy by adding more training data. The idea can be expanded further to speak the signs. For even easier identification, we can include more gesture and speech generation modules. We may also evolve this into a voice-to-sign language to facilitate communication with those who are especially abled. This facilitates easier communication

**REFERENCES**

**Base Paper:**

1. <http://cs231n.stanford.edu/reports/2016/pdfs/214_Report.pdf>
2. <http://www.iosrjen.org/Papers/vol3_issue2%20(part-2)/H03224551.pdf>

**Other References:**

1. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.734.8389&rep=rep1&typ e=pdf
2. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.734.8389&rep=rep1&typ e=pdf
3. <https://ieeexplore.ieee.org/document/7507939>
4. <https://ieeexplore.ieee.org/document/7916786>
5. <https://www.sciencedirect.com/science/article/pii/S1877050917320720>
6. https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neuralnetwork-cnn-deep-learning-99760835f148
7. <https://adeshpande3.github.io/adeshpande3.github.io/>A-Beginner's-Guide-To Understanding-Convolutional-Neural-Networks