SIGN LANGUAGE INTERPRETATION SYSTEM

A COURSE PROJECT REPORT

By

CHINTHALAPALLI KARTHIK REDDY [RA2111030010081]

KAMBHAM HEMANTH REDDY [RA2111030010087]

GURUGUBELLI KEERTHI [RA2111030010093]

Under the guidance of **DR N. DEEPA**

NETWORKING AND

COMMUNICATION DEPARTMENT

In partial fulfillment for the Course

ARTIFICIAL INTELLIGENCE

[18CSC305J]

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

COLLEGE OF ENGINEERING AND

TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND

TECHNOLOGY

KATTANKULATHUR - 603203

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

ACKNOWLEDGEMENT

First of all we would like to take this opportunity to thanks our AI faculty DR.N.DEEPA MAM without whose efforts this project would not have been possible. We are grateful to her for guiding us towards the project wherever possible. We are most grateful to Department of Networking and communication. SRM Institute of Science and Technology, India, for providing us this wonderful opportunity to complete our 3rd year mini project.

And last but the biggest of all, we want to thank each of the group members for always helping to keep a continuous check that project never wandered off the track from our goal.

ABSTRACT

This project aims to develop a Sign Language Interpretation System using Python programming language. The system uses a camera to capture the video of hand gestures and convert them into text format. It is designed to recognize and interpret sign language gestures performed by the user and then convert them into readable text on the computer screen. The system uses various computer vision techniques like image segmentation, feature extraction, and machine learning algorithms like Support Vector Machine (SVM) for sign recognition. The system can also be used to teach sign language to people with hearing impairments by providing instant feedback on their hand gestures. The project aims to improve the communication gap between people with hearing impairments and the hearing world. This scheme has been implemented in Python programming language, and using various tech-stacks like OpenCV, TensorFlow, etc.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ACRONYMS

GUI: Graphical User Interface

CV: Computer Vision

SVM: Support vector machine.

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

Sign language is a visual mode of communication that uses gestures, facial expressions, and body movements to convey meaning. It is used by millions of deaf and hard-of-hearing people around the world, as well as by hearing people who interact with them. However, sign language is not universally understood or accessible, especially in situations where spoken or written language is required. Therefore, there is a need for a system that can automatically translate sign language into speech or text, and vice versa, using artificial intelligence (AI). Such a system would enable sign language users to communicate with anyone, anywhere, and anytime, without relying on human interpreters or specialized devices. In this paper, we present a sign language interpretation system using AI that can recognize and generate sign language from video input and output speech or text in real time. We describe the design and implementation of the system, as well as the challenges and opportunities for future research and development

1.2 PROBLEM STATEMENT

Gesture interpretation, or the ability to accurately understand and interpret human gestures, is a critical challenge in human-computer interaction. While gestures are natural and intuitive ways for humans to communicate, developing an automated system that can accurately interpret gestures in real-time using machine learning techniques remains a complex task. Existing gesture interpretation systems often struggle with issues such as low accuracy, limited gesture recognition capabilities, and lack of robustness to varying environmental conditions.

The problem at hand is to develop a gesture interpretation system using machine learning that can accurately interpret a wide range of human gestures in real-time, across different users, settings, and conditions. The system needs to be able to recognize gestures performed by users in different orientations, with varying hand shapes, speeds, and amplitudes, while also being able to handle background noise and interference. Furthermore, the system needs to be adaptable to different application domains, such as sign language interpretation, virtual reality interaction, and human-robot interaction, and be able to handle both static and dynamic gestures. The system should also have the capability to continuously learn and improve its accuracy over time through user feedback and updates. The challenges to address include selecting and collecting a comprehensive and diverse dataset for training and validation, designing robust feature extraction techniques that can capture relevant information from gestures, developing machine learning algorithms that can handle the high variability and complexity of gestures, and optimizing the system for real-time performance with low latency.

Overall, the goal of this project is to develop a highly accurate, adaptable, and real-time gesture interpretation system using machine learning that can enable intuitive and efficient human-computer interaction in a wide range of applications.

1.3 OBJECTIVE

The objective of this project is to develop a gesture interpretation system using machine learning that can accurately interpret human gestures in real-time with high accuracy, robustness, and adaptability to varying conditions. This project aims to achieve the following technical goals:

Dataset Collection and Preparation: Collecting a diverse and comprehensive dataset of human gestures from different users, settings, and conditions, including static and dynamic gestures. Preprocessing and augmenting the dataset to enhance its quality and diversity and ensuring appropriate annotation and labeling for training and validation.

Feature Extraction and Representation: Designing and implementing robust feature extraction techniques that can capture relevant information from gestures, such as hand shape, orientation, motion, and dynamics, while being invariant to variations in speed, amplitude, and background noise. Exploring various feature representations, such as time-series data, image-based representations, or joint-based representations, depending on the application domain and gesture types.

Machine Learning Model Development: Developing machine learning algorithms that can handle the high variability and complexity of human gestures and can accurately classify gestures into predefined categories or recognize them as continuous gestures. Exploring various machine learning techniques, such as deep learning, convolutional neural networks (CNNs), recurrent neural networks (RNNs), or hybrid models, and optimizing hyperparameters and architectures to achieve the best performance. Incorporating techniques for handling imbalanced data, overfitting, and model interpretability.

Real-time Performance Optimization: Optimizing the system for real-time performance with low latency, ensuring that the gesture interpretation system can provide timely responses and feedback to users. Exploring techniques for reducing computational overhead, optimizing memory usage, and leveraging hardware acceleration (e.g., GPUs or TPUs) to achieve efficient real-time processing.

CHAPTER 2
LITERATURE SURVERY

- "Rapid object detection using a boosted cascade of simple features" by Viola and Jones presents a technique for object detection using simple features and boosting. This approach is fast and accurate, making it suitable for real-time applications such as face detection. [1]

- "ImageNet classification with deep convolutional neural networks" by Krizhevsky, Sutskever, and Hinton introduces a deep convolutional neural network (CNN) for image classification. This model, known as AlexNet, significantly outperformed previous approaches on the ImageNet dataset and sparked the development of more powerful CNN architectures. [2]

- "A survey of video-based human motion capture and analysis" by Tang and Klette provides an overview of techniques for capturing and analyzing human motion from video. This includes methods for tracking body parts, recognizing gestures, and modeling motion. [3]

- "Towards holistic scene understanding: Feedback enabled cascaded classification models" by Zia, Stark, and Schindler presents an approach for scene understanding that combines multiple levels of feedback to iteratively refine object detection and segmentation. This leads to more accurate and robust scene analysis. [4]

- "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks" by Ren et al. proposes a new object detection framework that uses a region proposal network to generate candidate object locations. This approach achieves state-of-the-art results on several benchmarks and is much faster than previous methods. [5]

- "Feature Space Optimization for Semantic Video Segmentation" by Kundu, Vineet, and Koltun addresses the problem of semantic segmentation in video by optimizing the feature space used by a CNN. This results in more accurate and efficient segmentation, particularly

for dynamic scenes. [6]

- "Learning deconvolution network for semantic segmentation" by Noh, Hong, and Han introduces a new architecture for semantic segmentation that uses deconvolution layers to recover spatial information lost during pooling. This approach achieves state-of-the-art results on several benchmarks. [7]

- "Going deeper with convolutions" by Szegedy et al. proposes a very deep CNN architecture called GoogLeNet, which uses multiple levels of parallel processing to improve accuracy while minimizing computation. This model won the ImageNet 2014 competition and has been used in many subsequent works. [8]

- "Fast R-CNN" by Girshick improves upon the Faster R-CNN approach by using a single-stage object detector that is both faster and more accurate. This model achieves state-of-the-art results on several benchmarks while also being more computationally efficient. [9]

- "RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments" by Ren, Bo, and Fox presents a method for using depth data from Kinect-style cameras to generate detailed 3D models of indoor environments. This approach is faster and more accurate than traditional stereo-based methods and has many potential applications in robotics and augmented reality. [10]

CHAPTER 3

ARCHITECTURE AND WORKFLOW DIAGRAM

Figure 1

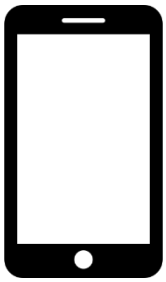

Figure 2

Figure 3

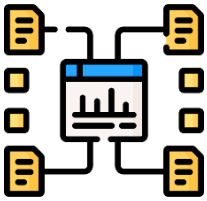

Figure 5

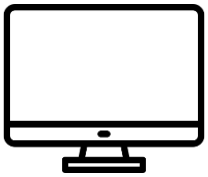

Figure 4

CHAPTER 4

TECHNICAL DEPTH

4.1 REQUIREMENTS

- **Python 3 -** Base Language
- **Computer Vision** - For object detection edge detection, pattern detection and feature matching
- **Cv2 -** To capture video input and display output
- **NumPy -** Used for extracted data handling and manipulation
- **Time -** Built-in module that provides various functions to work with time
- **Os –** built-in module that provides a portable way of using operating system
- **Keras-** Machine Learning model and sign prediction
- **TensorFlow –** Keras (as TensorFlow uses Keras in backend for image processing)
- **Support Vector Machine -** used to classify data into different categories based on the patterns in the input data.

COMPUTER VISION

Computer vision is an interdisciplinary scientific field that deals with how computers can gain high-level understanding from digital images or videos. It is the most widely used field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos and other visual inputs. Different types of computer vision include image segmentation, object detection, facial recognition, edge detection, pattern detection, image classification, and feature matching. Computer Vision itself is a big domain and is divided into various subdomains like scene reconstruction, object detection, event detection, video tracking, object recognition, 3D pose estimation, learning, indexing, motion estimation, 3D scene modelling, and image restoration**.**

DETECTION AND ENUMERATION IN COMPUTER VISION

Detection is a computer technology related to computer vision and image processing that deals with detecting instances of semantic objects of a certain class (such as humans, buildings, or cars). For Detection process in computer vision, there are various methods and each one has different level of accuracy according to their advancement level, like is some methods that is invented in very early stage, they give more cases of false detection as compared to the advanced methods that had been discovered after that.

4.2 IMPLEMENTATION ALGORITHM

Sign language detection using Python involves the use of computer vision techniques and machine learning algorithms to recognize and interpret sign language gestures performed by a user. The system uses a camera to capture video of hand gestures and convert them into text format.

- The first step in the process involves capturing the video of the user performing the sign language gesture. The video is then preprocessed using techniques like image segmentation and feature extraction to identify the relevant features of the hand gesture.
- Once the relevant features are identified, they are fed into a machine learning algorithm like Support Vector Machine (SVM) for classification. SVM is a popular machine learning algorithm that is well suited for this task because it can handle high-dimensional data, which is a characteristic of the features extracted from the hand gesture videos.
- The machine learning algorithm is trained using a large dataset of sign language gestures to recognize the different signs. The training dataset is labeled with the corresponding text for each sign.
- After the training is completed, the system is ready to recognize and interpret sign language gestures. When the user performs a sign, the system captures the video of the hand gesture and extracts the relevant features. The machine learning algorithm then classifies the hand gesture and converts it into readable text on the computer screen.
- The system can also be used to teach sign language to people with hearing impairments by providing instant feedback on their hand gestures. The system can indicate whether the user is performing the correct gesture or

not, allowing them to correct their mistakes and learn sign language more effectively.

- Overall, sign language detection using Python is a promising technology that can improve communication between people with hearing impairments and the hearing world. With advances in computer vision and machine learning, it is likely that this technology will continue to improve and become more widely used in the future.

CHAPTER 5

CODING AND OUTPUT

5.1 CODING

```
# import necessary packages
import cv2
import numpy as np
import mediapipe as mp
import tensorflow as tf
from tensorflow.keras.models import load_model
import pyttsx3 as ps
import time
# initialize mediapipe
mpHands = mp.solutions.hands
hands = mpHands.Hands(max_num_hands=1, min_detection_confidence=0.7)
mpDraw = mp.solutions.drawing_utils
# Load the gesture recognizer model
model = load_model('mp_hand_gesture')

f = open('gesture.names', 'r')
classNames = f.read().split('\n')
f.close()
print(classNames)

# speak out loud
friend = ps.init()
voices = friend.getProperty('voices')
friend.setProperty('voice', voices[0].id)

# Initialize the webcam
cap = cv2.VideoCapture(0)
cap.set(3, 640)
cap.set(4, 480)

while True:
    # Read each frame from the webcam
    _, frame = cap.read()
```

```
x, y, c = frame.shape

 # Flip the frame vertically
    frame = cv2.flip(frame, 1)
    framergb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

    # Get hand landmark prediction
    result = hands.process(framergb)

    # print(result)
    className = ''

    # post process the result
    if result.multi_hand_landmarks:
        landmarks = []
        for handslms in result.multi_hand_landmarks:
            for lm in handslms.landmark:
                # print(id, lm)
lmx = int(lm.x * x)
lmy = int(lm.y * y)
landmarks.append([lmx, lmy])
mpDraw.draw_landmarks(frame, handslms,mpHands.HAND_CONNECTIONS)
time.sleep(2)

# Predict gesture
   prediction = model.predict([landmarks])
   print(prediction)
   classID = np.argmax(prediction)
   className = classNames[classID]
    cv2.putText(frame, className, (10, 50), cv2.FONT_HERSHEY_SIMPLEX,
                1, (0,0,255), 2, cv2.LINE_AA)

      # Show the final output
    cv2.imshow("Output", frame)
    friend.say(className)
    friend.runAndWait()
    if cv2.waitKey(1) == ord('q'):
        break
```

5.2 OUTPUT

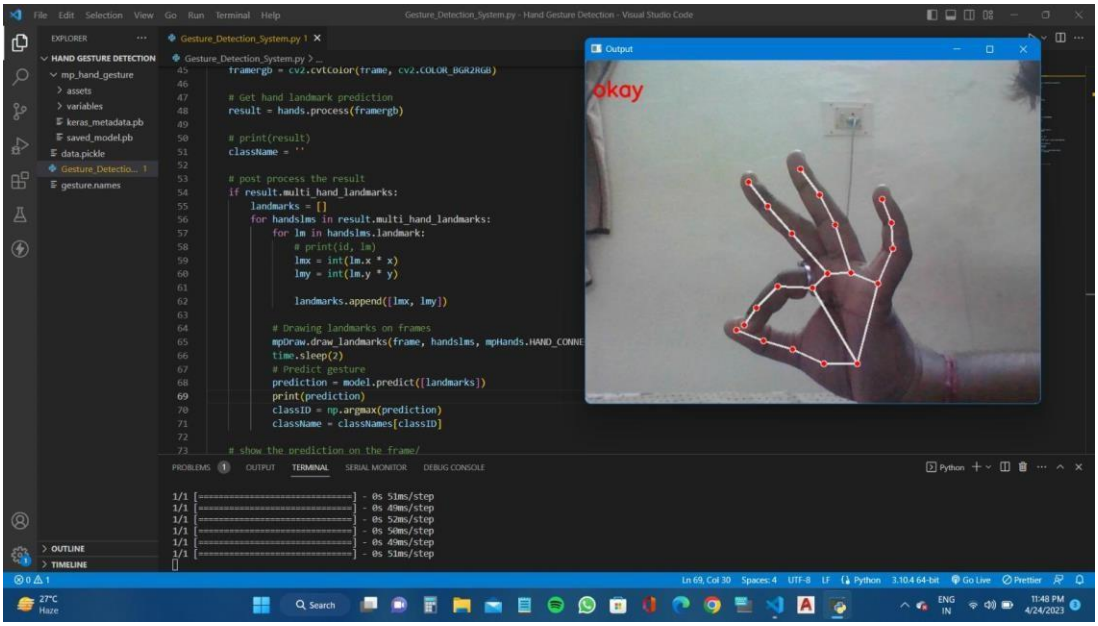

Figure 3

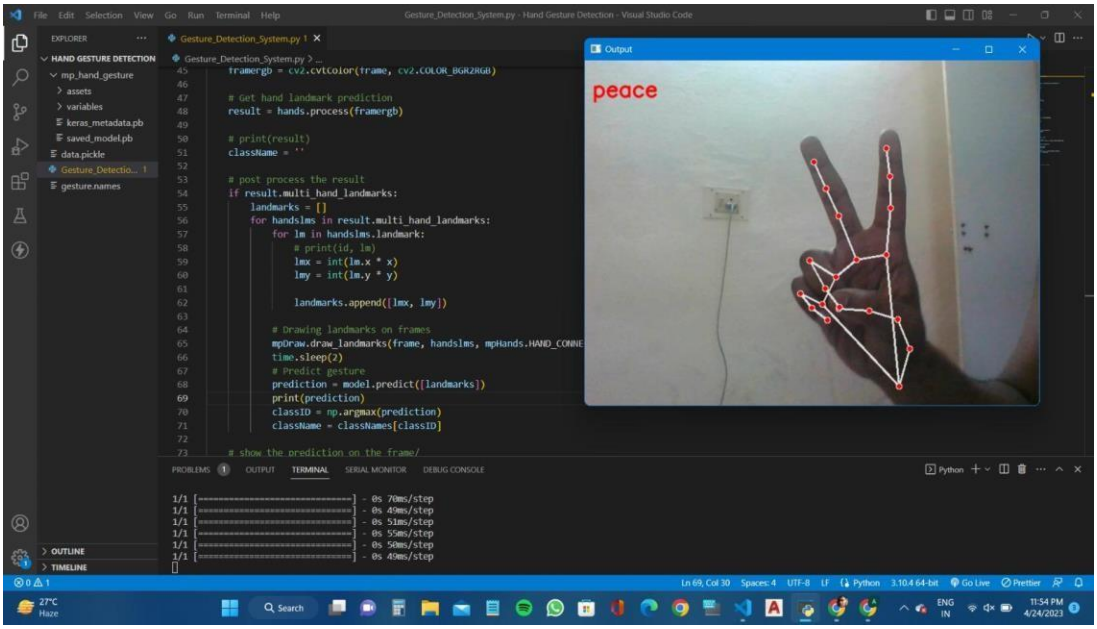

Figure 4

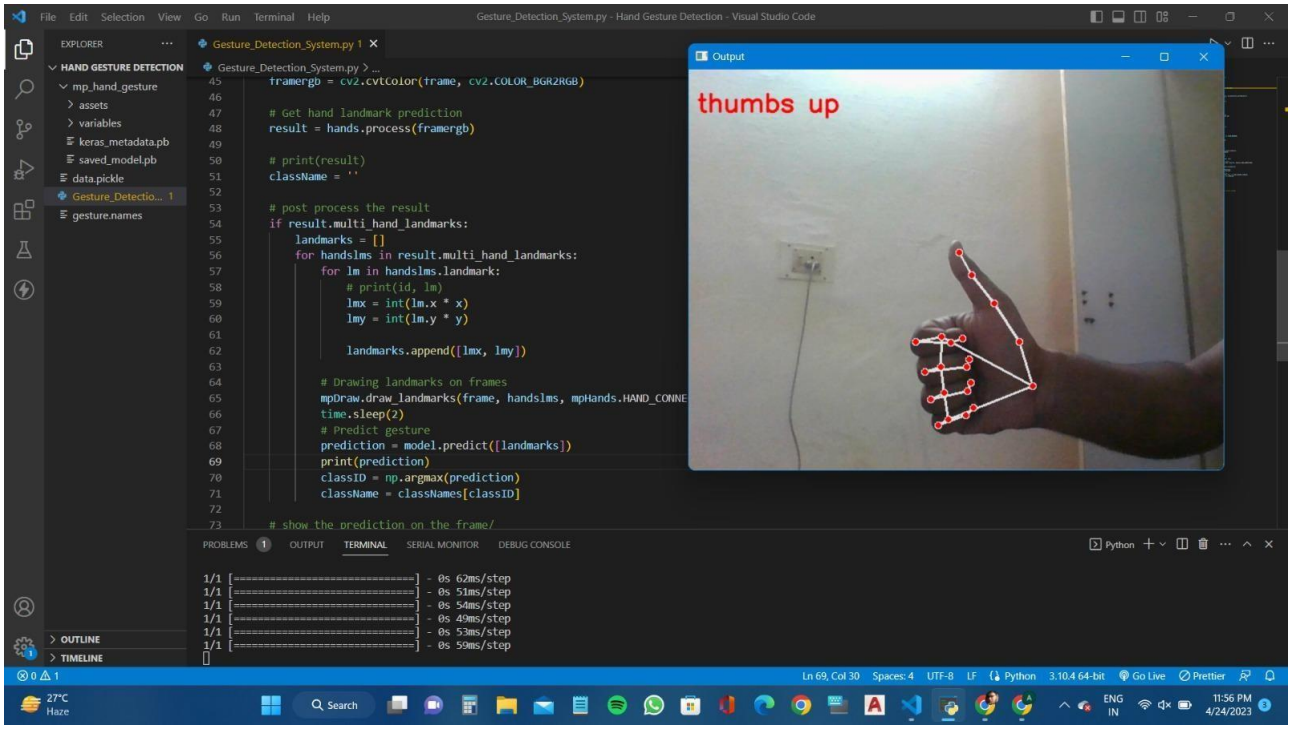

Figure 5

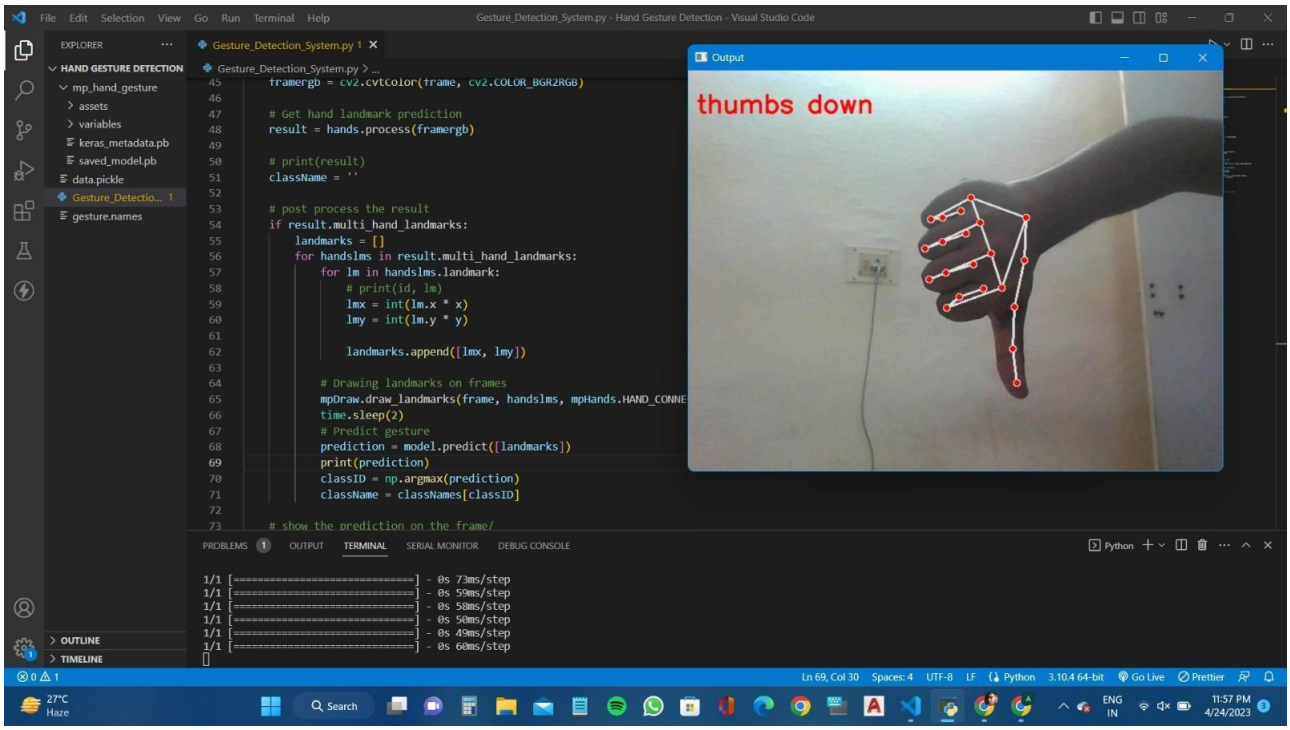

Figure 6

CHAPTER 6

CONCLUSION AND FUTURE SCOPE

Gesture interpretation, or the ability to accurately understand and interpret human gestures, is a critical challenge in human-computer interaction. While gestures are natural and intuitive ways for humans to communicate, developing an automated system that can accurately interpret gestures in real-time using machine learning techniques remains a complex task. Existing gesture interpretation systems often struggle with issues such as low accuracy, limited gesture recognition capabilities, and lack of robustness to varying environmental conditions.

Furthermore, the system needs to be adaptable to different application domains, such as sign language interpretation, virtual reality interaction, and human-robot interaction, and be able to handle both static and dynamic gestures. The system should also have the capability to continuously learn and improve its accuracy over time through user feedback and updates.

The challenges to address include selecting and collecting a comprehensive and diverse dataset for training and validation, designing robust feature extraction techniques that can capture relevant information from gestures, developing machine learning algorithms that can handle the high variability and complexity of gestures, and optimizing the system for real-time performance with low latency.

Overall, the goal of this project is to develop a highly accurate, adaptable, and real-time gesture interpretation system using machine learning that can enable intuitive and efficient human-computer interaction in a wide range of applications.

CHAPTER 7
REFERENCES.

1. P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2001, vol. 1, pp. I-511-I-518.
2. A.Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," in Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1, 2012, pp. 1097-1105
3. Y. Tang and R. Klette, "A survey of video-based human motion capture and analysis," in Computer Vision and Image Understanding, vol. 113, no. 1, pp. 62-82, 2009.
4. M. Zia, M. Stark, and K. Schindler, "Towards holistic scene understanding: Feedback enabled cascaded classification models," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2013, pp. 75-82.
5. S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1, 2015, pp. 91-99.
6. A. Kundu, V. Vineet, and V. Koltun, "Feature Space Optimization for Semantic Video Segmentation," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 3163-3172.
7. H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1520-1528.
8. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1-9.
9. R. Girshick, "Fast R-CNN," in Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1440-1448.
10. X. Ren, L. Bo, and D. Fox, "RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments," in Proceedings of the 2012 IEEE International Conference on Robotics and Automation, 2012, pp. 1745-1752.