Platform of Communication for the Speech Impaired

CS 492 B.Tech Main Project Report 2020

Done By

ADHITHYA E - TCR16CS004

ANJANA G - TCR16CS017

ASHIN SHANLY K S - TCR16CS020

RIYA MARIUM THOMAS - TCR16CS046

Guided By

RAHAMATHULLA K

Assistant Professor

Dept of Computer Science and Engineering

Government Engineering College

Thrissur-680009

ABSTRACT

Human beings interact with each other to convey their ideas and thoughts to people around them via the magic of vocal chords. But unfortunately this is not the case for the speech/hearing impaired. With only about 250 certified sign language interpreters in India, translating for a deaf population of approximate size between 1.8 million and 7 million requires a strenuous effort.

The proposed project is a sign language recognition application that provides an intelligent, natural, and convenient way of human–computer interaction. Sign Language Recognition (SLR) and gesture-based control aims at interpreting sign languages by a computer and converting them to text format in order to aid in communication for the speech impaired. The proposed project also converts voice to text format for reverse communication. The application works offline and requires a laptop with a webcam. It also provides a learning interface for the users by encompassing a sign language manual. The friendly UI of the product also enables users to completely control the activity flow using gestures alone.

ACKNOWLEDGEMENT

It gives us great pleasure to present our main project report on ”Platform to Aid in Communication for the Speech Impaired”. No work, however big or small, has ever been done without the contribution of others. So these words of acknowledgement come as a small gesture of gratitude towards all those people, without whom the successful completion of this report would not have been possible.
We express our profound thanks to our project guide Rahamathulla K(Assistant Professor, Dept. of CSE), Government Engineering College Thrissur, who guided and helped us throughout the phases of designing this project.

We would like to express our gratitude towards project coordinators Associate Professor.Helen K J, Professor.George Mathew for their indirect support during project evaluation.
We are indebted to Vipin Kumar K S(Head of the Department, Dept. of CSE), Government Engineering College Thrissur, for giving us the opportunity to proceed with our project idea and for its successful presentation.
We would like to thank all the lab staff for providing us with the necessary facilities for the presentation, it would not have been possible to complete this report without their prompt cooperation.
Last but not the least we would like to thank all our friends, who supported me with their valuable criticism, advice and support.

Contents

List of Tables

List of Figures

ABSTRACT

Human beings interact with each other to convey their ideas and thoughts to the people around them. But this is not the case for deaf-mute people. With only about 250 certified sign language interpreters in India, translating for a deaf population of between 1.8 million and 7 million requires a strenuous effort.

The proposed project is a sign language recognition and gesture recognition application that provides an intelligent, natural, and convenient way of human–computer interaction. Sign language recognition (SLR) and gesture-based control aims to interpret sign languages automatically by a computer and convert to textual format in order to help the deaf communicate. The application works offline and requires a laptop with a camera. It also provides a learning interface for the users. The friendly interface of the product enables the users to completely control the interface using gestures alone.

Nomenclature

DNN	Deep Neural Network
LSTM	Long Short Term Memory
ISL	Indian Sign Lanuguage
CNN	Convolutional Neaural Network
CV	Computer Vision
ML	Machine Learning
UI	User Interface

Nomenclature

GUI	Graphical User Interface
HTTP	Hyper Text Markup Language

Chapter 1

Introduction

1.1 Problem Statement

The main objective is to translate sign language to text. The framework provides a helping-hand for speech-impaired to communicate with the rest of the world, mainly during online interviews, formal meetings using sign language. This leads to the elimination of the middle person who generally acts as a medium of translation. This would contain a user-friendly environment for the user by providing text output for a sign gesture input.

1.1.1 Current System

Speech impairments can make it hard to communicate. Sign language is a form of communication used by people with impaired hearing and speech. Language disorders can make it difficult for people to express their own thoughts and feelings through speech. It can be seen that there is no proper communication facility for people who have difficulty in communication. People use sign language gestures as means of non-verbal communication to express their thoughts and emotions. But non-signers find it extremely difficult to understand, hence trained sign language interpreters are needed during medical and legal appointments, educational and training sessions. Over the past five years, there has been an increasing demand for interpreting services.

1.1.2 Proposed System

The proposed system is to create a vision based system which offers sign language translation to text thus aiding communication between signers and non-signers.

1.2 Feasibility of the Project

1.2.1 Scope of the Project

Sign language translator have always been a great need in the society in order to break the barrier of communication between people who use sign language and the people who does not use to. Sign language translation has always been a difficult job for people to understand it.It often leads to improper delivery of messages to the person who does not understand sign language. This is where our sign language translator becomes useful. We can use it in medical appointments, meetings, interviews etc.

1.2.2 Technical Feasibility

We have analysed the technical feasibility of the project and it is feasible based on the following factors.

1.2.2.1 Hardware Feasibility

The minimum hardware requirements for developing this are:

- Camera with minimum resolution of 5MP
- A computer system

1.2.2.2 Software Feasibility

The project is technologically feasible,through the use of an application which uses Caffe DNN framework, Inception v1, and LSTMs.

1.2.3 Financial Feasibility

1.2.3.1 Development Cost

For developing an application no particular cost is required. The demonstration is done using Webcam, which comes integrated with every laptop computer today. Cost for human effort is the only expense.

1.2.3.2 Installation Cost

No particular installation cost is needed other than the cost of hardware devices.The application can be installed free of cost.

1.2.3.3 Operational Cost

Execution of the application requires minimum operational cost to keep the product up and running.

1.2.3.4 Maintenance Cost

No particular maintenance cost is needed for this software apart from the time expense of the developer.

1.3 Process Model

The project targets to develop a platform for recognizing sign language, thereby reducing the communication gap between the speech/hearing impaired and the rest.The application activity starts as the user whose gestures to be recognized align their hands with the camera. The project will use the captured input from the camera and match them with the predefined gestures.In order to define gestures, we will store the output of each gesture to a database. After this when a user performs a signing, the program will try to determine and match it with a predefined gesture. If successful , it will give the associated translation of the signing in text format.
The project is aimed at recognizing American Sign Language (ASL) and translating it to text format (English). Other languages will neither be considered as the input(the sign language) nor as the output language(the language that the output text is in). The program will have certain limitations like gestures based on finger movements and abstract words in the initial phase. Initially, the program will recognize only a few signs of the ASL. But, it is possible to define more gestures once it is well established that the program works well enough. The program is developed to work on a PC environment mainly.

1.3.1 Possible Process Models

We have analysed various process models like :

- Incremental Model
- Agile Model
- Waterfall Model
- Iterative Waterfall Model
- Prototype Model etc.

1.3.2 Selected Model

Agile model is selected for the project. We are planning to implement the system with basic facilities only. So many future enhancements are possible which are listed below. Agile model can satisfy this requirement efficiently. Since it follows the plan-do-check-act for the improvement, backtracking can done easily in Agile model.

1.3.3 Model Description

Agile SDLC model is a combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software product. Agile Methods break the product into small incremental builds. These builds are provided in iterations. Each iteration typically lasts from about one to three weeks. Every iteration involves cross functional teams working simultaneously on various areas like -

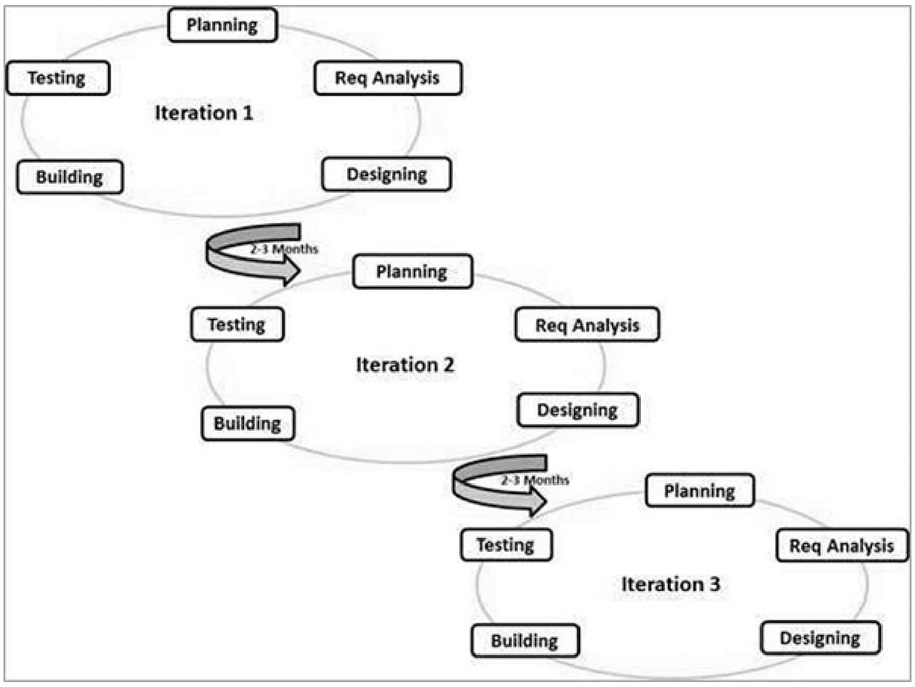


Figure 1.1: Agile Model

- Planning
- Requirements Analysis
- Design
- Coding
- Unit Testing and
- Acceptance Testing.

1.3.3.1 Why Agile Model?

- It is a very realistic approach to software development.
- Promotes teamwork and cross training.
- Functionality can be developed rapidly and demonstrated.
- Resource requirements are minimum.
- Suitable for fixed or changing requirements
- Good model for environments that change steadily.
- Easy to manage.

1.3.3.2 Agile Model solves issues like:

- Resource wastage
- Costly modifications
- Unclear requirements

1.3.3.3 Future Increments

- Making the textual predictions grammatically correct.
- Incorporating text to gesture conversion to the product.
- Implementing dynamic hand tracking.

Chapter 2

Requirement Analysis

Requirements analysis and validation is a process of refinement, modelling and specification of the already discovered user requirements. The systematic use of proven principles, techniques, languages, and tools for the cost effective analysis, documentation, and on-going evolution of user needs and the specification of the external behaviour of a system to satisfy those user needs form an integral part of this stage of software development. This chapter describes the method of requirement elicitation employed and the user requirements thus gathered. The requirements are also finalized after validation using a suitable method.

2.1 Method of requirement elicitation

Identifying the actual needs of users is necessary for the implementation of the project.To discover the requirements of the system from users, customers and other stake holders the following techniques are used :

- Interview
- Brainstorm
- Facilitated Application Specification Technique Function Deployment
- Online Survey

2.1.1 Interview

In order to collect the requirements, we visited Amritha Speech and Hearing Improvement School Ayyanthole, Thrissur. We spend time talking to the faculty and students. We prepared a questionnaire and asked questions to identify the requirements. The questions asked were:

- What is the sign language followed by the students of the school?
- Do students find it difficult following ASL?
- Do they find it difficult to communicate using the sign language with people who don't know about sign language?

- How do students communicate if they have to face an interview?
- How are they going to communicate in online interviews if they have to face one.
- What do you believe would address this problem?
- What are the limitations of sign language?
- How long does it take to learn sign language?
- Can you please give an overview of sign language teaching in your college?
- Can you please give an overview of sign language teaching in your college?
- Who would be benefited by such an application?
- What are the necessary features you would like to incorporate with a sign language translator app?

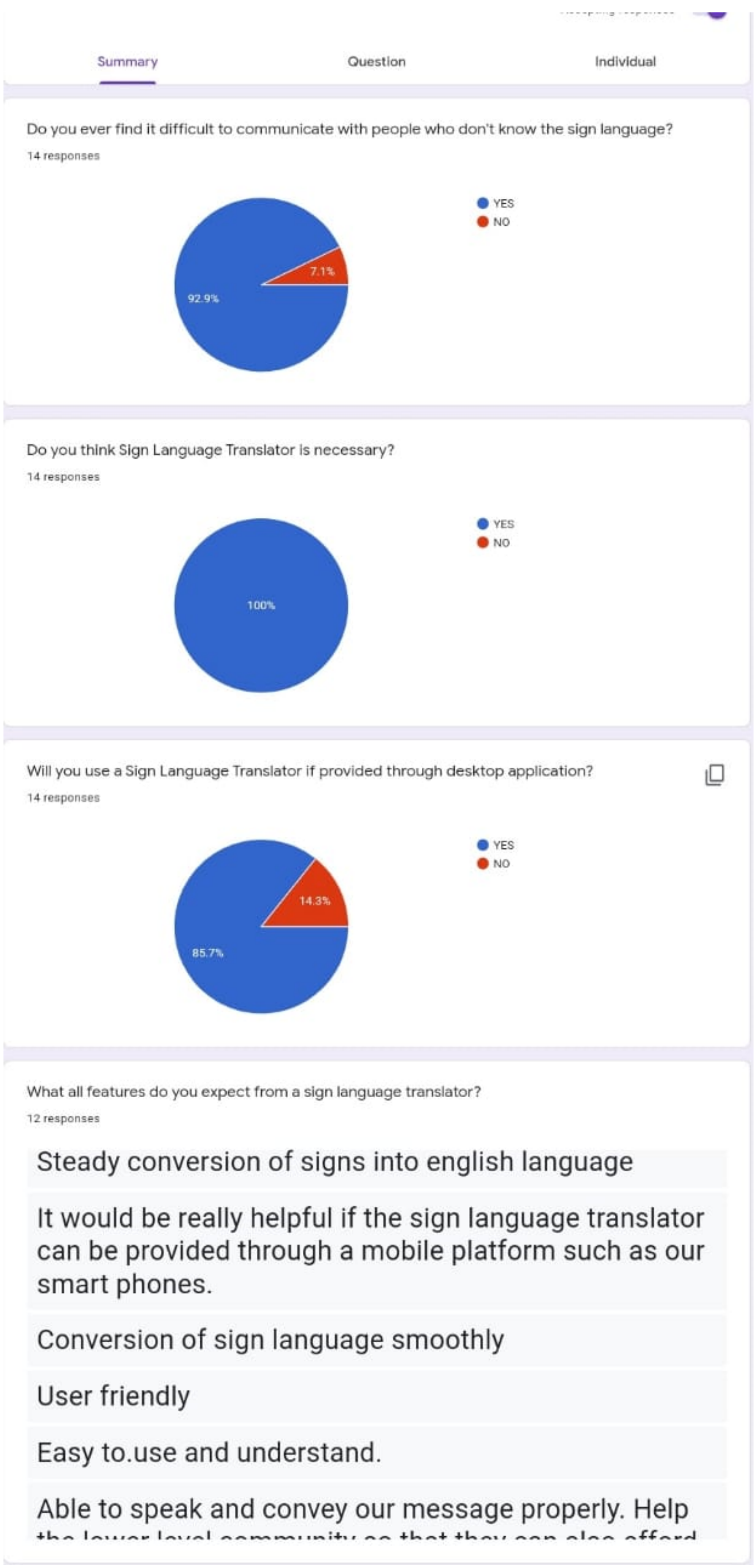
Their suggestions and opinions were noted down and in the light of that, we identified the users needs and requirements.There is a lack of trainers of sign language. Qualified trainers of sign language are very few in number. American Sign Language is a lot varying based on regions. People tend to develop their own signs in each region, hence there is no uniformity of sign representation. Communication with people who don't know sign language is difficult and in this context, they find relevance of this application.

2.2 Brainstorm

We conducted a brainstorming session amongst us to decide on the strategy to be put into action. After having a discussion, we decided to come up with something so that they too can participate in effortless interactions with the rest. So we decided to have a conversation with the students facing such problems in order to find the real requirements to accomplish this project.

2.3 Online Survey

We searched for the available methods of sign language translation and collected all the resources we could find, after which we studied it thoroughly to find out it's limitations and further scopes that can be included in our product. We aim to build a cross platform desktop application that could solve all the limitations of the existing systems such as accuracy and speed. In order to do so, we did a thorough survey online to collect all the information that could help us to understand what the requirements are. We circulated a google form asking suggestion from peers about what all features do they expect in an ideal sign language translator.



Do you ever find it difficult to communicate with people who don't know the sign language?

14 responses

Do you think Sign Language Translator is necessary?

14 responses

Will you use a Sign Language Translator if provided through desktop application?

14 responses

What all features do you expect from a sign language translator?

12 responses

Steady conversion of signs into english language

It would be really helpful if the sign language translator can be provided through a mobile platform such as our smart phones.

Conversion of sign language smoothly

User friendly

Easy to.use and understand.

Able to speak and convey our message properly. Help

Figure 2.1: Results of Online Survey

2.4 Literature Survey

As part of the project, the following journals and papers were referred :

- **Literature survey on hand gesture techniques for sign language recognition** [1]
  As compared to other sign languages, ISL interpretation has got less attention by the researcher. In this paper, some historical background, need, scope and concern of ISL are given. Various techniques available for hand tracking, segmentation, feature extraction and classification are listed. In this paper, Vision based approach have been discussed for interpreting the Indian sign language using hand modality. A Typical Hand Gesture Recognition system consists of mainly four modules: Gesture acquisition, Tracking and segmentation, Feature extraction and description, Classification and recognition. This paper focuses on a study of sign language interpretation system with reference to vision based hand gesture recognition. An attempt has also been made to explore about the need and motivation for interpreting ISL.

- **Appearance Based Recognition of American Sign Language Using Gesture Segmentation** [2]
  The work presented in this paper goals to develop a system for automatic translation of static gestures of alphabets in American Sign Language. In doing so three feature extraction methods and neural network is used to recognize signs. The system deals with images of bare hands, which allows the user to interact with the system in a natural way. An image is processed and converted to a feature vector that will be compared with the feature vectors of a training set of signs. The system is rotation, scaling of translation variant of the gesture within the image, which makes the system more flexible. The system is implemented and tested using data sets of number of samples of hand images for each signs. Three feature extraction methods are tested and best one is suggested with results obtained from ANN.

- **Real-time American Sign Language Recognition with Convolutional Neural Networks** [3]
  The work in this paper presents the development and implementation of an American Sign Language (ASL) translator based on a convolutional neural network. We utilize a pre-trained GoogLeNet architecture trained on the ILSVRC2012 dataset. The system features a pipeline that takes video of a user signing a word as input through a web application. We then extract individual frames of the video and generate letter probabilities for each using a CNN (letters a through y, excluding j and z since they require movement). With the use of a variety of heuristics, we group the frames based on the character index that each frame is suspected to correspond to. Finally, we use a language model in order to output a likely word to the user.

- **An Efficient Method for Sign Language Recognition from Image Using Convolutional Neural Network** [4]
  The work in this paper presents a new method which translates from American Sign Language (ASL) fingerspelling into a letter using Convolutional Neural Network and transfer learning. The method is using Google pre-trained model named MobileNet V1 which was trained on the ImageNet image database.

- **American Sign Language alphabet recognition using Convolutional Neural Networks with multiview augmentation and inference fusion** [5]
  This paper describes a method for ASL alphabet recognition using Convolutional Neural Networks (CNN) with multi view augmentation and inference fusion, from depth images captured by Microsoft Kinect. The approach augments the original data by generating more perspective views, which makes the training more effective and reduces the potential overfitting. During the inference step, the approach comprehends information from multiple views for the final prediction to address the confusing cases caused by orientational variations and partial occlusions. On two public benchmark datasets, our method outperforms the state-of-the-arts.

- **Video-based sign language recognition using Hidden Markov Models** [6]
  This paper is concerned with the video-based recognition of signs. Concentrating on the manual parameters of sign language, the system aims for the signer dependent recognition of 262 different signs taken from Sign Language of the Netherlands. For Hidden Markov Modelling a sign is considered a doubly stochastic process, represented by an unobservable state sequence. The observations emitted by the states are regarded as feature vectors, that are extracted from video frames. This work deals with three topics: Firstly the recognition of isolated signs, secondly the influence of variations of the feature vector on the recognition rate and thirdly an approach for the recognition of connected signs.

- **Real-Time Hand Gesture Detection and Recognition Using Bag-of-Features and Support Vector Machine Techniques** [7]
  This paper presents a novel and real-time system for interaction with an application or video game via hand gestures. Our system includes detecting and tracking bare hand in cluttered background using skin detection and hand posture contour comparison algorithm after face subtraction, recognizing hand gestures via bag-of-features and multiclass support vector machine (SVM) and building a grammar that generates gesture commands to control an application. In the training stage, after extracting the key points for every training image using the scale invariance feature transform (SIFT), a vector quantization technique will map key points from every training image into a unified dimensional histogram vector (bag-of-words) after K-means clustering.

- **Survey on gesture recognition for hand image postures** [8]
  This paper presents a review study on the hand postures and gesture recognition methods, which is considered to be a challenging problem in the human-computer interaction context and promising as well. The major tools for classification process include FSM, PCA, HMMs, and ANNs are discussed. Descriptions of recognition system framework also presented with a demonstration of the main three phases of the recognition system by detection the hand, extraction the features, and recognition the gersture. The major image preprocessing steps necessarily required to features extraction phase are segmentation, edge detection, noise removing, and normalization, these steps may not applied together depending on targeted application.

- **Real-Time Static Hand Gesture Recognition for American Sign Language (ASL) in Complex Background** [9]
  In this work, real-time hand gesture system is proposed. Experimental setup of the system uses fixed position low-cost web camera with 10 mega pixel resolution mounted on the top of monitor of computer which captures snapshot using Red Green Blue [RGB] color space from fixed distance. This work is divided into four stages such as image preprocessing, region extraction, feature extraction, feature matching.

- **Shape, texture and local movement hand gesture features for indian sign language recognition** [10]
  This paper proposes an automatic gesture recognition approach for Indian Sign Language (ISL). Indian sign language uses both hands to represent each alphabet. We propose an approach which addresses local-global ambiguity identification, inter-class variability enhancement for each hand gesture. Hand region is segmented and detected by YCbCr skin color model reference. The shape, texture and finger features of each hand are extracted using Principle Curvature Based Region (PCBR) detector, Wavelet Packet Decomposition (WPD-2) and complexity defects algorithms respectively for hand posture recognition process. To classify each hand posture, multi class non linear support vector machines (SVM) is used. Dynamic gestures are classified using Dynamic Time Warping (DTW).

- **Hand Gesture Recognition system for Real-Time Application** [11]
  The objective of this paper is to implement the vision based hand gesture recognition system to control the movement of robot. We can use of Scale invariant feature transform (SIFT) for extract the keypoint from the gesture image capture by single sensing device. Space incompatibility of SIFT keypoint causes bag of feature approach was introduced. Then use the vector quantization will map the keypoint extracted from SIFT into unified dimensional histogram vector after the K-mean clustering. The histogram vectors as an input to multiclass SVM classifier for recognize the gesture.

- **Survey on Various Gesture Recognition Technologies and Techniques** [12]
  In this paper a survey on various recent gesture recognition approaches is provided with particular emphasis on hand gestures. A review of static hand posture methods are explained with different tools and algorithms applied on gesture recognition system, including connectionist models, hidden Markov model, and fuzzy clustering. Challenges and future research directions are also highlighted.

- **Vision-based Hand Posture Detection and Recognition for Sign Language-A study** [13]
  Unlike general gestures, Sign Languages (SLs) are highly structured so that it provides an appealing test bed for understanding more general principles for hand shape, location and motion trajectory. Hand posture shape in other words static gestures detection and recognition is crucial in SLs and plays an important role within the duration of the motion trajectory. Vision-based hand shape recognition can be accomplished using three approaches 3D hand modelling, appearance-based methods and hand shape analysis. In this survey paper, we show that extracting features from hand shape is so essential during recognition stage for applications such as SL translators.

- **Recognizing hand gesture using motion trajectories** [14]
  An algorithm for extracting and classifying two-dimensional motion in an image sequence based on motion trajectories is proposed in this paper. First, a multiscale segmentation is performed to generate homogeneous regions in each frame. Regions between consecutive frames are then matched to obtain 2-view correspondences. Affine transformations are computed from each pair of corresponding regions to define pixel matches. Pixels matches over consecutive images pairs are concatenated to obtain pixel-level motion trajectories across the image sequence. Motion patterns are learned from the extracted trajectories using a time-delay neural network.

- **A Transform for Multiscale Image Segmentation by Integrated Edge and Region Detection** [15]
  This paper describes a new transform to extract image regions at all geometric and photometric scales. It is argued that linear approaches such as convolution and matching have the fundamental shortcoming that they require a priori models of region shape. The proposed transform avoids this limitation by letting the structure emerge, bottom-up, from interactions among pixels, in analogy with statistical mechanics and particle physics. The transform involves global computations on pairs of pixels followed by vector integration of the results, rather than scalar and local linear processing.

- **Visual Tracking of Bare Fingers for Interactive Surfaces** [16]
  Visual tracking of bare fingers allows more direct manipulation of digital

objects, multiple simultaneous user interacting with their two hands, and permits the interaction on large surfaces, possibly in “unsafe” locations such as public spaces. The only required specific equipment is an affordable, off-the-shelf video camera. Its design is based on our modeling of two classes of algorithms that are key to the tracker: Image Differencing Segmentation (IDS) and Fast Rejection Filter (FRF).

- **Posture Recognition using Combined Statistical and Geometrical Feature Vectors based on SVM** [17]
  Posture recognition is done for American Sign Language to recognize static alphabets and numbers. 3D information is exploited to obtain segmentation of hands and face using normal Gaussian distribution and depth information. Features for posture recognition are computed using statistical and geometrical properties which are translation, rotation and scale invariant. Hu-Moment as statistical features and; circularity and rectangularity as geometrical features are incorporated to build the feature vectors. These feature vectors are used to train SVM for classification that recognizes static alphabets and numbers.

- **Hand gesture recognition using Haar-like features and a stochastic context-free grammar** [18]
  This paper proposes a new approach to solve the problem of real-time vision-based hand gesture recognition with the combination of statistical and syntactic analyses. The fundamental idea is to divide the recognition problem into two levels according to the hierarchical property of hand gestures. The lower level of the approach implements the posture detection with a statistical method based on Haar-like features and the AdaBoost learning algorithm. With this method, a group of hand postures can be detected in real time with high recognition accuracy.

- **Static hand gesture recognition based on local orientation histogram feature distribution model** [19]
  This paper proposes a bottom up approach for static hand gesture recognition. The key steps are augmenting the local orientation histogram feature vector with its relative image coordinates, and clustering the augmented vector to find a compact yet descriptive representation of the hand shape. The recognition result is given by collective voting of all the local orientation histogram features extracted from the hand region of the testing image. The matching score is evaluated by retrieval in the image feature database of the training hand gestures.

- **Dynamic Gesture Recognition** [20]
  This paper introduce a method for enabling dynamic gesture recognition for hand gestures. The recognition is processed as part of three key stages, with a fourth in development. The first stage processes the visual information from the camera, and identifies the key regions and elements (such as the hand and fingers), this classified information is passed to a 2D to 3D module that transforms the 2D classified information into a full

3D space applying it to a calibrated hand model using inverse projection matrices and inverse kinematics. Simplifying this model into posture curvature information we apply this to a Hidden Markov Model (HMM). This model is used to identify and differentiate between different gestures, even ones using the same finger combinations.

2.5 User requirements

The application should facilitate better communication between the speech/hearing impaired and the rest. A major part of the problem is the lack of knowledge of sign languages among the fortunate. So the project must include functionalities to provide better and easier communication between them without undergoing actual training in signing the American Sign Language.

2.5.1 List of Problems Obtained After the Requirements Elicitation

- There is no proper sign language converter that could meet the requirements of such people.
- The students find it difficult to communicate with people who don't understand sign language.
- They cannot participate in any online/offline interviews because of the absence of a proper sign language translator.

2.6 The User Requirements Formulated from the Problem List

- The response time should be minimum such that the output should be provided to the user quickly in order to facilitate better communication between both parties of the conversation.
- Videos have to be sliced into frames and should be processed effectively to interpret the signs and to match with the predefined data set accurately.
- A clean and friendly user interface that interacts with user to take in the gesture input and output the textual translation.

2.7 Requirement Validation

During the requirement validation process,different types of checks are carried out and the result is displayed in the following section. The types of checks are:

- Validity checks

- Consistency checks
- Completeness checks
- Realism checks

2.7.1 Validity checks

This section ensures that the system performs the exact function as requested by the intended users. The project is to build a web application which works like a skype like platform system for conducting online interviews for specially abled people. This application is a service oriented system. So it aims to provide service like conducting the interview of people who rely upon sign language and translating the sign language for the interviewer in the text format. So we need to ensure that the system along with providing it's basic purpose of providing response for user, should also perform services with high accuracy and reliability. We formed a focus group including some of our classmates. This focus group is like a group of critics. We presented our whole idea and design to this group. They critically examined it and this helped us to conclude that our system will be able to perform all these functions.

2.7.2 Consistency Checks

As part of the consistency check, we verified that the documents are consistent. We also searched whether we have any redundant description of the same function. All the forms that the user may request are already stored in the server as documents. We analysed all these documents to ensure that there is no redundant form saved under the same name. There is a chance to save some frequently queried forms like translation request form to be saved redundantly. Our in depth analysis and verification helped us to free our documents and data in the data set from all these inconsistencies.

2.7.3 Completeness Checks

This level of verification is to ensure that the proposed system performs all these requirements and constraints. That is, this level ensures the completeness of the system. We have adopted the agile methodology for this project. Firstly, the system will provide only sign language translation to text, then services are added as we progress further. So, the completeness can be assured only after all these functionalities are incorporated to the system.

2.7.4 Realism Checks

This type of validation is to ensure that we can realize this project using existing technologies. We consulted lot of specialists like our professors, web application developers and some specialists in the field of machine learning to check the feasibility of the project. With their suggestion, we made the decided to optimize the existing system with Deep Neural Networks. They suggested us to include voice outputting also. We hope to include this voice as future modification of our system. By consulting this specialists and through online surveys, we were able to confirm that this project is practically possible and economically viable.

2.8 Conclusion of Requirement Analysis

This document pointed out the requirement analysis for our project and reasons behind the selection of this idea. During the process of requirement analysis, current scenario was studied and data was acquired through means of Questionnaire and Study of existing techniques and documents. Various people at the different strata of the society were surveyed and the data was collected. These acquired data were studied and thus the requirements were analysed to make specific modifications which will help us in planning the project.

Chapter 3

Software Requirements Specification

This section is a software requirement specification of a helper utility to aid the communication of speech impaired people through sign language recognition. Through this section, we are going to provide all specifications and functionalities of the project. This section will mention the functionality, that is what the resulting application is supposed to do, external interfaces which interacts with the users, performance and attributes, that is if the application is portable, or maintainable, and design constraints imposed on the implementation such as implementation language, input specifications and output expectations.

3.1 Document Purpose

The aim of this document is to specify the features, requirements of the product and the interface of the sign language recognition application. It will explain the scenario of the desired project and necessary steps in order to succeed in the task. To do this throughout the document, overall description of the project, the definition of the problem that this project presents a solution and definitions and abbreviations that are relevant to the project will be provided. The preparation of this SRS will help consider all of the requirements before design begins, and later during redesign, re-coding, and retesting. If there will be any change in the functional requirements or design constraints part, these changes will be stated by giving reference to this SRS in the following documents.

By reading this document a reader can learn about what the project is implemented for.

3.2 Product Scope

Communication is possible for the speech/hearing impaired without the means of acoustic sounds using sign language. However not everyone knows about this and hence there exists a gap in the communication. The aim behind this work is to overcome this gap and to develop a system for recognizing the sign language, which facilitates effortless communication between people with speech impairment and the rest, thereby reducing the communication gap between them. In the future, the functionalities may be extended to convert voice input to the corresponding textual translation.

The product will use the input and match them with the collected data sets. In order to define gestures, we will store the output of each signing to a database. After this when a user performs a gesture, the program will try to determine and match it with a predefined data set and then provide an appropriate translation.

The project is aimed at recognizing the American Sign Language and translating it to text in English only. Other languages will neither be considered as input language(the sign language) nor as output language(the language that the output text in). Obviously, our program will have some limitations like gestures based on finger movements so, the program will not cover abstract words. For the beginning the program will recognize only few gestures predefined. Yet, it will be possible to define more gestures once it is proved that the program works well enough. The program will work on PC environment mainly.

3.3 Intended Audience and Overview

The intended audience include end users who face difficulty in communicating with deaf-mute people. Also for, system developers and designers for further optimisation and extension.

3.4 Definitions, Acronyms and Abbreviations

3.4.1 Definitions

- Caffe - Caffe is a deep learning framework made with expression, speed, and modularity in mind.It is originally developed at University of California, Berkeley. It is open source, under a BSD license.
- Machine Learning - Machine learning is a sub-field of computer science that evolved from the study of pattern recognition and computational learning theory in artificial intelligence. Machine learning explores the study and construction of algorithms that can learn from and make predictions on data.

3.4.2 Abbreviations

- SRS
- SDK
- DNN

- CV
- UI

3.5 Document Conventions

This document follows IEEE standard format and the conventions followed for fonts, formatting and naming are the standards followed in Computer Science and Engineering Department of Government Engineering College, Thrissur.

3.6 Overall Description

3.6.1 Product Perspective

The application serves to sign dependent people as an interface to overcome the difficulty in communication by converting their gestures to the text for people who does not know sign language. It can also be used to learn the sign language.

The major functionalities are communication and education. The communication module translates the gesture obtained as input to corresponding text by comparing it on the predefined sets. The education module helps the users to find out what all signs the system recognizes and also to learn the signs with their meanings.

The overall intention of this system is to provide a clean and neat interface which can easily assist for the smooth communication of sign-to-text with the most achievable accuracy possible with no need for internet.

3.6.2 Product Functionality

The main function of our application is recognition and translation of real time hand signing. Our software will be able to detect and recognize gestures from the user using openCV. The data set for the recognition needs to be trained using caffe DNN. If the movement is not recognized, it will produce an error message. If the gesture is recognized, the translation will be written.

Another feature includes an educational manual where the users can learn the sign languages with meanings. Also the users could search for the specific word in the manual which responds by providing the corresponding gesture.

Another feature will be to update new signs which are not recognized by the software. It is currently a reserved feature for the developer.

3.6.3 Users and Characteristics

By user classes and characteristics we broadly define the users who need frequent use of the product and the various requirements of each particular user classes.

- The system is basically viewed as a translation medium in the eyes of the front-end users. The impaired user can input various gestures which the application will transform to corresponding words. The self explanatory UI enables the user to use it freely and comfortably. It can also be used to learn the sign language.
- From the viewpoint of a developer, there are a variety of functions to be performed. The duty includes to add new signs. Adding a new sign consists of having relevant number of images pertaining to the sign followed by training the system to understand that sign and classify any sign similar to it as so, if it is found. Finally the system should be trained to fit the new model. Developer should ensure there is no over fitting, but also that there is proper classification.

3.6.4 Operating Environment

The application is designed to be a desktop application. For the users, laptop/desktop with a webcam will suffice the requirement which is now a very common feature. Also adequate lighting in the environment should be maintained for proper gesture input. It is recommended that the user who is performing sign to be the only one under the radar of the camera. More specically, it is important that only a pair of hands are being analyzed by the camera at once.

For the developer, a laptop/desktop that has enough memory, CPU such as i7–7500U and storage to update and optimize the algorithms and train models would suffice the requirements.

3.6.5 Design and Implementation Constraints

- The current application works for the English sign language only, hence possess a language constraint.
- The accuracy of the model depends on how well its trained which eventually requires a huge data set.
- The application works only one way. In future the system shall be able to convert the written commands back to hand signals.
- Lighting and external noise affects the working accuracy of the model.
- Occlusion can also be a constraint.

3.6.6 Assumptions and Dependencies

Assumptions :

- All the fingers or hand for detecting the gesture are placed in correct position.
- The surrounding is properly lighted.
- The user is aware of different motions and gestures that can be recognized using the system.
- The model has been trained with the signs the user is trying to give as input.
- The user has a stable and fast internet connection, also a laptop.

Dependencies :

- Tensorflow

3.7 Specific Requirements

The specific requirements of this project will be considered in following three subsections, which are interface, functional and non-functional requirements.

3.7.1 External Interface Requirements

3.7.1.1 User Interfaces

The user is first directed to the home view. The top portion of the screen consists of the view that encases the real time video input and the bottom portion consists of a view that shows the real time translation of the sign language shown by the user. The video input is only captured once the user shows the reserved sign to start capture or presses the record button. After the user presses the stop button or the reserved gesture the video input is no longer captured and the recorded signs are classified by the model and the resulting translation is shown in the translation view below the current view. If the user presses the 'reset' button, the captured frames will be deleted off and the whole process is reset. The button labeled 'M' on the top left portion of the view when clicked will take the user to the sign language manual section of the application where the user can view the different hand signs for different phrases/words/letters associated with the SL.

Further along production, the interfaces may vary in their organizational structure. However, the functionalities will remain the same.

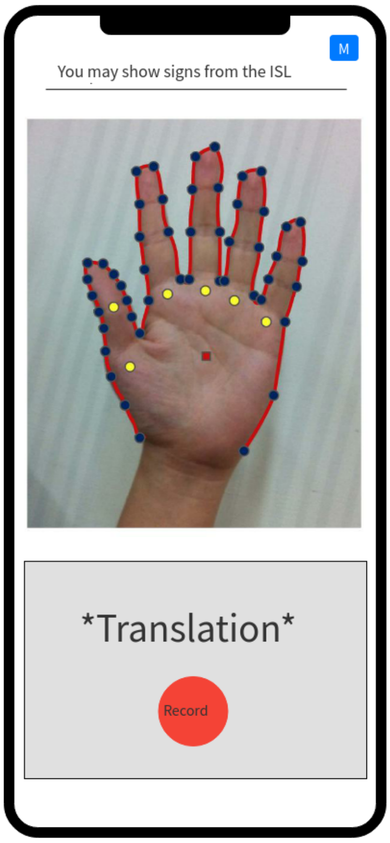


Figure 3.1: Home UI

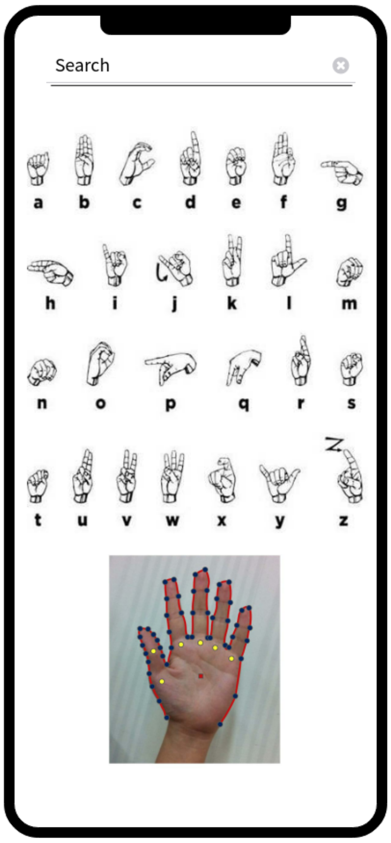


Figure 3.2: Sign Manual UI

3.7.1.2 Hardware Interfaces

A laptop/desktop with a web cam, which most of the above have as an inbuilt feature is required for the end users to use the product. Development of the application requires the availability of a laptop/desktop on the developers side that has enough memory, CPU such as i7–7500U and storage to update and optimize the algorithms and to train models.

3.7.1.3 Software Interfaces

The major layer of software that is visible to the user is the front end of the application. To make it, Lazarus is used. TensorFlow is an interface for expressing machine learning algorithms, and an implementation for executing such algorithms. The system is flexible and can be used to express a wide variety of algorithms, including training and inference algorithms for deep neural network models, and it has been used for conducting research and for deploying machine learning systems into production across more than a dozen areas of computer science and other fields. The sign-to-text conversion is done with the help of Caffe DNN framework. OpenCV is used for real-time computer vision applications. The use of various other packages in python are necessary, such as NumPy and Speech Recognition.

3.7.2 Functional Requirements

Functional requirements will analyze the services provided to the users by the system. The function of this system is to capture the sign language and convert it to corresponding text. It will have the following phases:

- Collecting data samples of American Sign Language
  Collect the data set corresponding to the American Sign Language and train the system for a minimum number of hours so as to achieve an efficient classifying model.

- Gray Scale conversion of images
  The frames captured by the camera are converted to gray scale images prior to forwarding them to the classifier to avoid any noises that can occur via colour variations in different frames of the sequence.

- Model training
  For learning four different deep learning techniques are used. Each technique results in respective model which are further used for comparative study.

In the following section, the functional output intended from the model is briefly described.

- Efficient Detection of Sign Language
  To complete the task of gesture detection, different signs of the sign language have to be identified properly. This is accomplished by training a system with data sets that include American Sign Language. The

model developed should capture the sign correctly and should be matched to the most approximate sign.

- Conversion of Sign Language
  The sign language captured needs to be converted to its gray scale image and then converted to the correct text.

3.8 Nonfunctional Requirements

3.8.1 Performance Requirements

- Detection of the sign must be as fast as possible. In the project, detection of a single sign should take up to a minute maximum.
- Steady performance in all conditions.
- The camera should offer minimum quality to be able to detect the curves and shadows of the ngers, along with their proper positioning.
- Single user in front of camera. If there are multiple users in front of the camera, it would require much more complex computation to gure out who is performing sign language or not.
- Good lighting in the environment for proper detection.
- High classification accuracy.

3.8.2 Security Requirements

- There are no requirements regarding privacy issues surrounding the use of the product by the users.
- Only developer should have access to update new signs.

3.8.3 Software Quality Attributes

3.8.3.1 Reliability

The proposed system shall be reliable in terms of accuracy. Accuracy is necessary for making sure that classier correctly identies all the correct gesture and output correctly. It depends on the availability of the sign, correct detection of the hand region and proper training and testing of the system.

3.8.3.2 Maintainability

The system should be easily maintainable and adding new data sets must be easy. Currently the system works for a particular language and for a limited data set. The ultimate goal should be to evolve as a system which translates any sign language and any gestures.

3.8.3.3 Adaptability

The system needs to be updated with new signs for better working as time passes. However, the addition should be adaptable and in no means should compromise with the performance and efficiency.

3.8.3.4 Usability

The platform provides a friendly UI and can easily be used by speech impaired people to communicate over the application. The corresponding meaning shall also be shown in the UI for the other end user.

Chapter 4

Design Implementation

The aim of the project is to create a platform that will potentially ease the communication of the speech impaired. The project belongs to image processing and deep learning category. The system is trained to recognize the different signings the user does. After recognition the sign is converted to textual form which the oppposite user/actor can view. Here we will see the system design, design of user interfaces and different UML Diagrams.

4.1 Overall Design

4.1.1 System Design

The hardware architecture of the system would comprise of a minimum requirement of webcam/camera on the laptop/desktop which the user is using. Whereas the software architecture consists of a user interface, the ability to read camera input, pre-process the video input, and use the result for classification using a trained model.
The system uses a CNN model that extracts temporal features from the frames which were used further to predict gestures based on sequence of frames. The CNN model used is Inception, which is a model developed by Google for image recognition and is widely regarded as the most used image recognition neural network which exists right now.

4.1.2 System Architecture Design

The architectural design is a high-level overview of how the responsibilities of the system were partitioned and then assigned to sub-systems. The working of the system can be explained by combining the functionalities of the sub modules. The collaboration of sub-modules are explained by the functions relating them. The sub-modules of the system are:

- **Gray scale conversion**: The frames captured by the camera are converted to gray scale images prior to forwarding them to the classifier to avoid any noises that can occur via colour variations in different frames of the sequence.

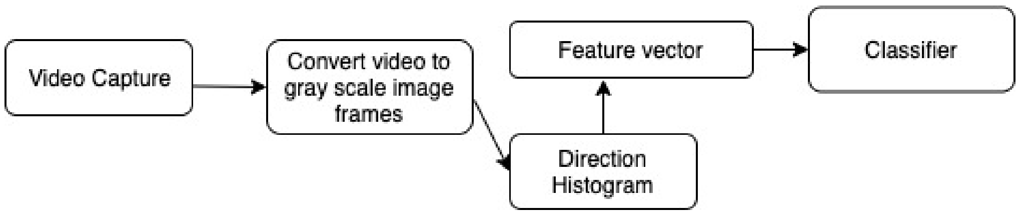


Figure 4.1: Workflow Diagram

- **Train the system for recognizing signs of the American Sign Language**: Collect the data set corresponding to the American Sign Language and train the system for a minimum number of hours so as to achieve an efficient classifying model.
- **Sign Language Manual**: The data set of the ASL is also made available to the user of the application via manual section in case he/she gets lost while interpreting or using a signing.

4.1.3 System Hardware Architecture

Desktop/Laptop with a webcam is the required piece of hardware for the developer and user of this application. The developer's system should have sufficient memory and computational power, because we train the system to recognize certain patterns using TensorFlow, and training requires a sufficient amount of memory and processing power. The product is a desktop application and therefore the user also needs a system, whether it be a desktop or laptop with a webcam of minimum 5MP for the usage of the application. The user system uses the webcam to capture images and sends it to the OpenCV program. The final translated text is shown on the hardware monitor to the user.

4.1.4 System Software Architecture

4.1.4.1 Sign Detection

After creating the data set of images for training the neural network to classify images, we can simply retrain the existing Inception model to work on our data set. By using transfer learning, we can take advantage of previous training and use a relatively small training set.
Using the Inception model place the onus of choosing what type of convolution to use (3x3, 5x5) onto the model itself. Inception performs all the convolutions in parallel and concatenates the resulting feature maps before going to the next layer. The Inception model repeats the operations several times to create a much deeper network. We take the outputs of Softmax Layer and the Max Pooling layer for our architecture.

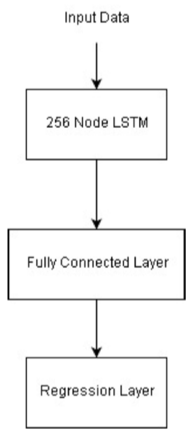


Figure 4.2: RNN Architecture

4.1.4.2 Sign Classification

We take the outputs of the Softmax Layer and the Max Pooling layer and feed it to the RNN architecture shown in Figure 4.2.
The gesture segments identified and processed by the CNN are classified by the LSTM into one of the gesture classes using sequence data. Since the input segments have to be a fixed size, we trimmed the length of all the frame sequences. We use an LSTM because of their efficiency with longer sequences of data. We train the LSTM on the outputs from the CNN Softmax layer and the Pool layer and compare the results. After researching about deeper and wider RNNs we decided that this network architecture would give the best results in accuracy. The LSTM layer has a dropout of 0.3 to prevent over fitting. It uses ADAM optimizer and a Softmax layer for predictions along with categorical cross-entropy loss.

4.1.5 User Interface Design

The user is first directed to the home view which is shown in Figure 4.3. The top portion of the screen consists of the view that encases the real time video input and the bottom portion consists of a view that shows the real time translation of the sign language shown by the user. The video input is only captured once the user presses the record button or shows the reserved gesture. After the user presses the stop button/reserved gesture the video input is no longer captured and the recorded signs are classified by the model and the resulting translation is shown in the translation view below the current view.

If the user presses the 'reset' button, the captured frames will be deleted off and the whole process is reset. The button labeled 'M' on the top right portion of the view when clicked will take the user to the sign language manual section of the application where the user can view the different hand signs for different phrases/words/letters associated with the ASL. This interface design is depicted in Figure 4.4

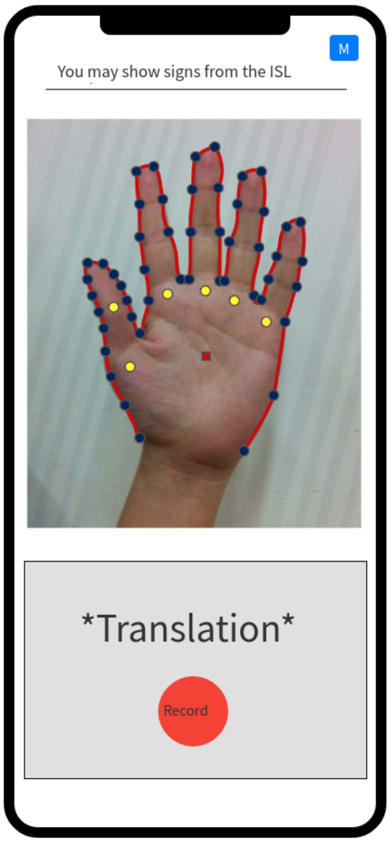


Figure 4.3: Home UI

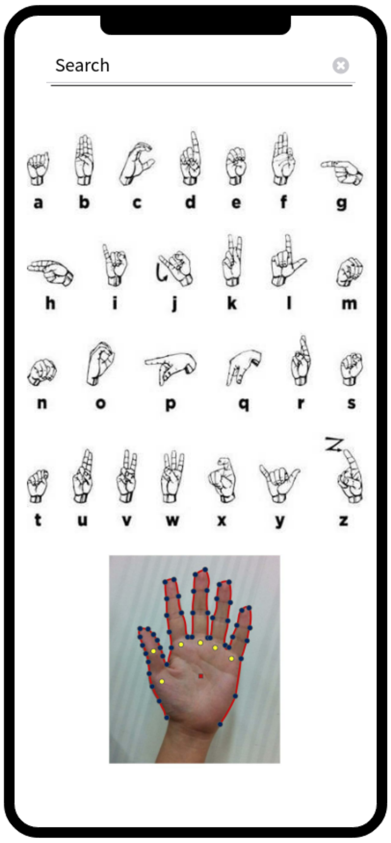


Figure 4.4: Sign Manual UI

4.1.6 Use Case Description

The Use Case diagram is shown in Figure 4.5.
User

- Navigate to the application : The user navigates to the application which would direct the user to a homepage where the camera input of the webcam and the record/stop button will be shown.
- Align hand with camera : Scan the hand of the user with the webcam of the system so that the model can locate 21 points on the hand and keep track of the motions of the same.
- Press Record Button : Once the interviewee presses the record button the video input begins to be captured.
- Reset Button : When the user feels like the signing made so far have to be reset or done again, he/she can simple press the reset button and it will reset the recorded signings so far.
- Stop Button : Pressing of the stop button will stop capture.
- Show Translation : Once the stop button is pressed the corresponding translation of the input is displayed on the associated view on the screen.

Developer

- Update data set : The developer updates the data set with videos used for training the model.
- Train model : The training and testing phase of the classifier model is done here.
- Push changes : The final changes made to the application has to be put into production here.

4.1.7 Sequence Diagram

The sequence diagram is depicted in Figure 4.6. The overall procedure that happens when a user starts using the application can be summarised as follows:

- The Once the user presses the record button the web cam starts capturing the video sequence.
- The captured sequence is then filtered by background subtraction and various other actions.
- This is then given to the Tensorflow Script that is pre-trained to recognize the signs, which then returns the highest probable phrase/word suitable for the recognized action.

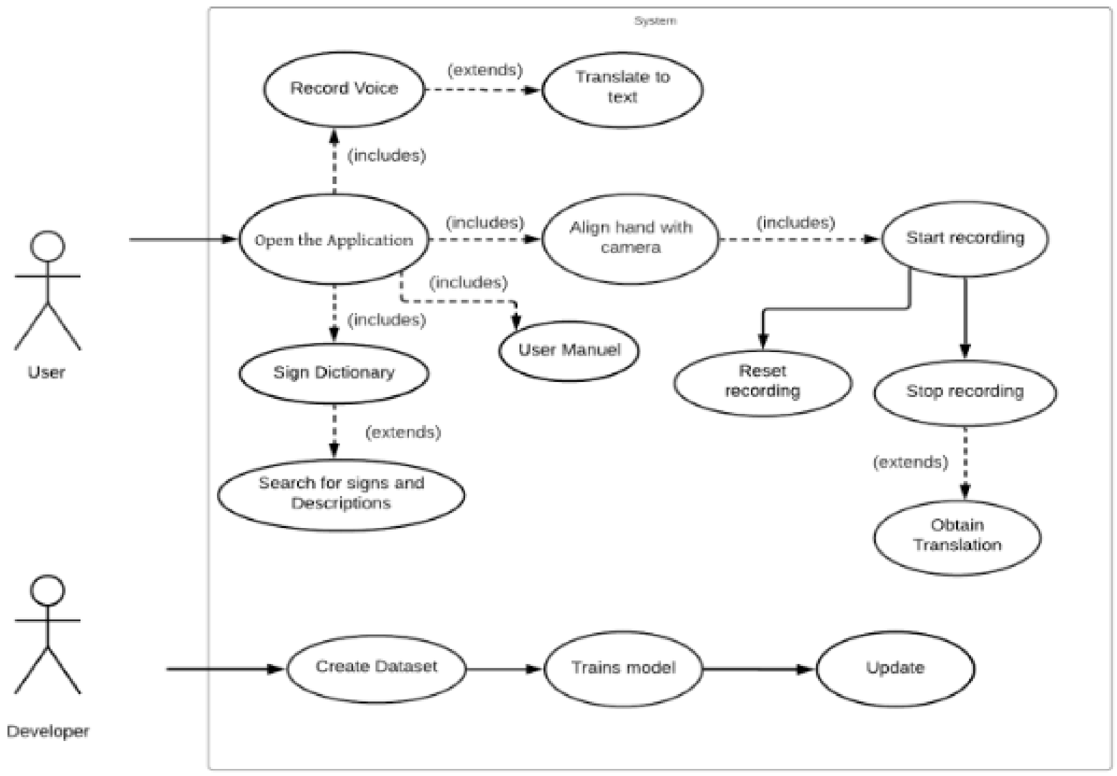


Figure 4.5: Use Case Diagram

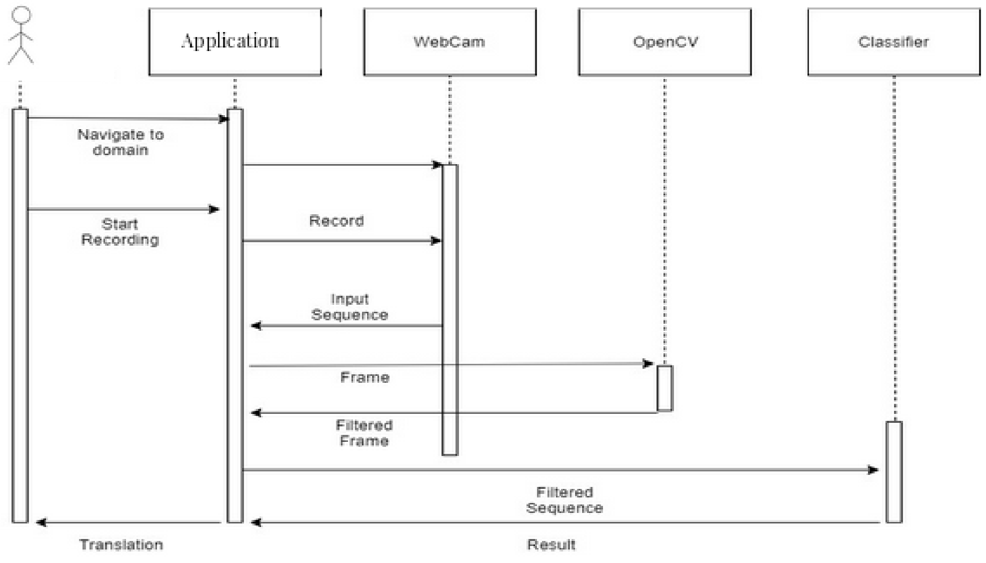


Figure 4.6: Sequence Diagram

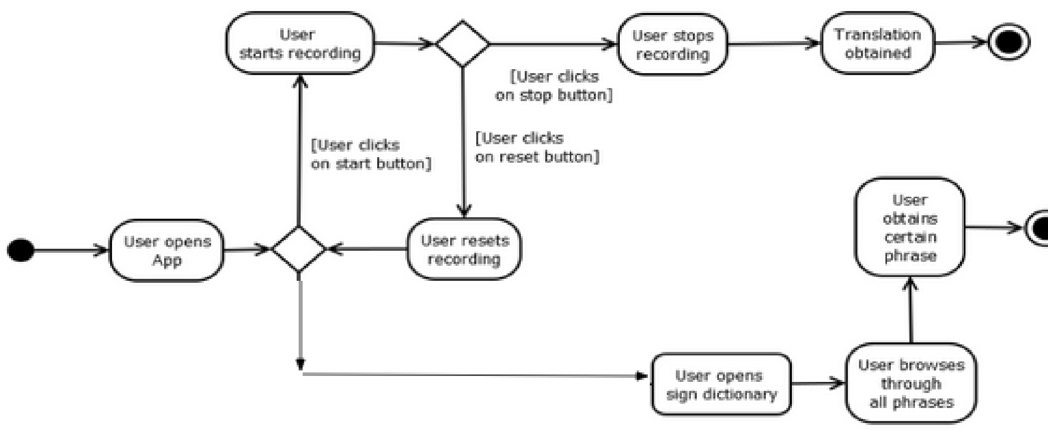

User starts recording
User stops recording
Translation obtained
[User clicks on stop button]
[User clicks on start button]
[User clicks on reset button]
User opens App
User resets recording
User obtains certain phrase
User opens sign dictionary
User browses through all phrases

Figure 4.7: Activity Diagram

4.1.8 Activity Diagram

Activity diagram is used to show all the available activities in a system. There are 2 major and 1 minor activity here. Each activity begins with the user with the open application. From there on there is a possibility of different activities based on the button the user chooses. If the use clicks on start, the user begins recording and if he stops recording he will obtain the translation of his input. This is the first major activity. Alternatively, once the user starts recording he can choose to reset the recording, this will bring the interface back to the position of either recording or going to the settings. The last 2 activities are activated once we access the settings. So the user first opens the app and then clicks on settings. From there he has two options either he can choose to view instructions which is the minor activity, or he can choose to access the sign dictionary. Once at the sign dictionary the user can choose to manually browse the list of phrases available or he can search for a certain phrase using a search bar, this is the last major activity. This is depicted in Figure 4.7.

4.2 Algorithm

Algorithms used: 1. Rule Based Classifier, 2. Background subtraction method by detecting the color of skin using HSV (Hue Saturation Value) model. The webcam starts capturing the video once the user clicks the record button. The video sequence is then shrunk to a dimension that supports the learning model after denoising it, which is later given as input to the OpenCV program. The final output will go through a filter that filters out all the interfering colours. This is then fed to the classifying model which then categorizes it and outputs the most fitting textual translation corresponding to the captured frame.

4.2.1 Training and Testing

During training, after loading the data set the initial parameters are set and weights are continuously updated using back propagation until they are stable. Whereas testing comprises of cross verifying the observed response of a relatively new video frame. The model has to be retrained with more data sets wherever the classification seems inaccurate.

4.3 Design Verification

Design verification is used to ensure that the product as designed is the same as the product intended. In order to meet the customer expectations and avoid costly design modifications, appropriate selection of design verification method is essential. We have analysed the following project activities for accomplishing this goal:

- Concept through detailed design
- Specification development
- Detailed design through to pre-production

4.4 Case Tools

Computer Aided Software Engineering (CASE) is the application of a set of tools and methods to a software system with the desired end result of high-quality, defect free, and maintainable software products. In this section, we describe tools that have been used so far, i.e. up to the design phase - Draw.io ,Overleaf and Mockflow wireframe.

4.4.1 Overleaf

Overleaf is a free service that lets you create, edit and share your scientific ideas easily online using LaTeX, a comprehensive and powerful tool for scientific writing. This was used in the making of this document to synchronize inputs from all members of the group.

4.4.2 Draw.io

Draw.io is a free online diagram software for making flowcharts, process diagrams, org charts, UML, ER and network diagrams. The same was used to create the different UML diagrams of this model.

4.4.3 Mockflow wireframe

It is a cloud-based wireframe software allowing designers to collaborate in real time on user interface prototypes for websites and software. This was used for creating the user interface of this application.

4.5 Test Case Design

To generate a test case, initially the criterion to evaluate a set of test cases is put forth and then the set of test cases satisfying that condition is generated.

4.5.1 Designing test cases

Test cases are designed based on different scenarios as shown in Table 4.1.

Table 4.1: Test Cases

Input	**Expected Output**
User grants webcam access	Application loads home screen with camera enabled
User starts recording	Webcam starts capturing video input
User presses reset	Captured frames are deleted
User clicks manual	All signs available for translation appear on screen
User clicks a phrase	User obtains information about the phrase
User searches for a phrase	User obtains the phrase if it exists
User stops recording	Camera stops recording and translation is displayed

Chapter 5

Coding

5.1 Training the Model

Keras is a user-friendly neural network library written in Python. Below given code is to import essential modules and train the classification model to recognize the user inputted gesture with atmost accuracy. Models in Keras are defined as a sequence of layers. Fully connected layers were defined using the Dense class. We specified the number of neurons or nodes in the layer as the first argument, and specified the activation function using the activation argument. We created a Sequential model and added layers one at a time until the desired model was achieved. We used 'relu' as the activation function.

After defining model we compiled it. we used cross entropy as the loss argument to evaluate different weights and specified the optimizer which is used to search through different weights for the network and any optional metrics we would like to collect and report during training.

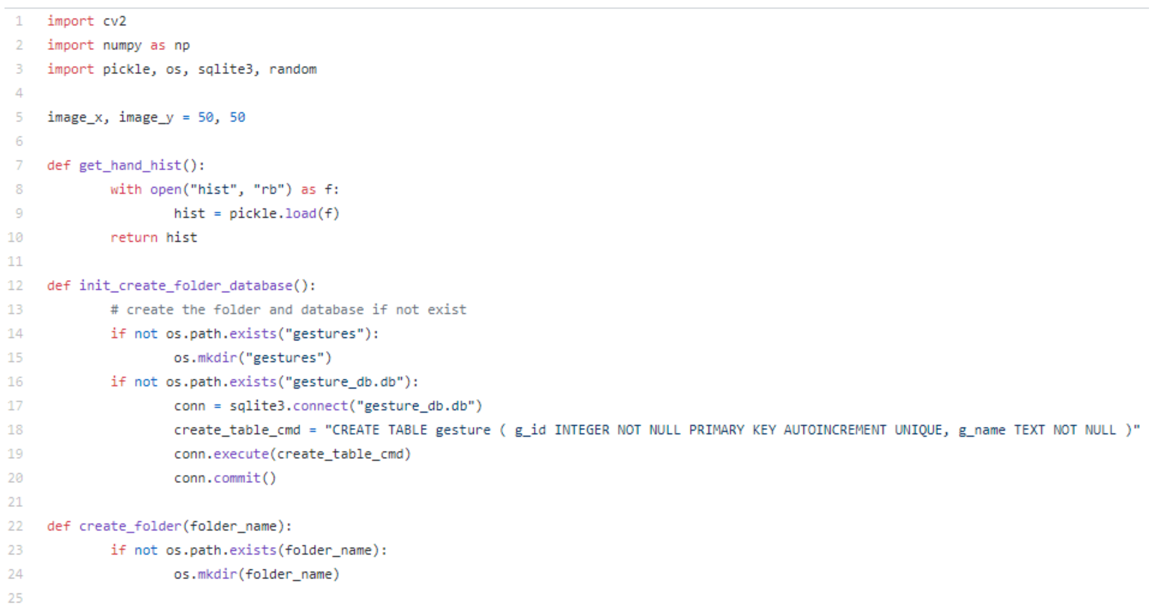
```
import numpy as np
import pickle
import cv2, os
from glob import glob
from keras import optimizers
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import Flatten
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.utils import np_utils
from keras.callbacks import ModelCheckpoint
from keras import backend as K
K.set_image_dim_ordering('tf')

    os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
```

```
def get_image_size():
  img = cv2.imread('gestures/1/100.jpg', 0)
  return img.shape

def get_num_of_classes():
  return len(glob('gestures/*'))

   image_x, image_y = get_image_size()

def cnn_model():

  num_of_classes = get_num_of_classes()
  model = Sequential()
  model.add(Conv2D(16, (2,2), input_shape=(image_x, image_y, 1), acti-
vation='relu'))
  model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2), padding='same'))
  model.add(Conv2D(32, (3,3), activation='relu'))
  model.add(MaxPooling2D(pool_size=(3, 3), strides=(3, 3), padding='same'))
  model.add(Conv2D(64, (5,5), activation='relu'))
  model.add(MaxPooling2D(pool_size=(5, 5), strides=(5, 5), padding='same'))
  model.add(Flatten())
  model.add(Dense(128, activation='relu'))
  model.add(Dropout(0.2))
  model.add(Dense(num_of_classes, activation='softmax'))
  sgd = optimizers.SGD(lr=1e-2)
  model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])
  filepath="cnn_model_keras2.h5"
  checkpoint1 = ModelCheckpoint(filepath, monitor='val_acc', verbose=1,
save_best_only=True, mode='max')
  callbacks_list = [checkpoint1]

  from keras.utils import plot_model
  plot_model(model, to_file='model.png', show_shapes=True)
  return model, callbacks_list

def train():

  with open("train_images", "rb") as f:
    train_images = np.array(pickle.load(f))
  with open("train_labels", "rb") as f:
    train_labels = np.array(pickle.load(f), dtype=np.int32)

   with open("val_images", "rb") as f:
    val_images = np.array(pickle.load(f))
  with open("val_labels", "rb") as f:
    val_labels = np.array(pickle.load(f), dtype=np.int32)
```

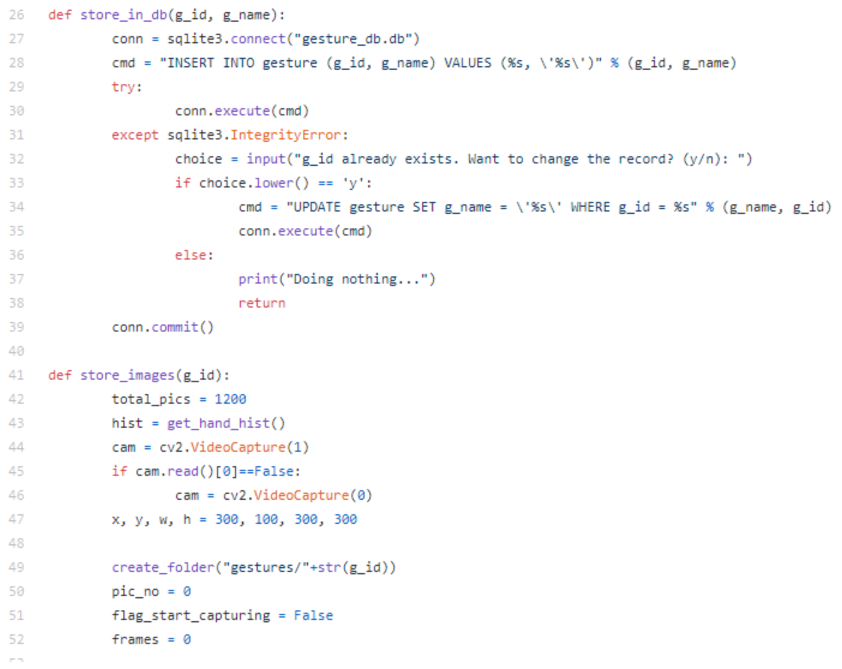
```
        train_images = np.reshape(train_images, (train_images.shape[0], image_x,
image_y, 1))
       val_images = np.reshape(val_images, (val_images.shape[0], image_x, im-
age_y, 1))
       train_labels = np_utils.to_categorical(train_labels) val_labels = np_utils.to_categorical(val_la

        print(val_labels.shape)

        model, callbacks_list = cnn_model()
       model.summary()
       model.fit(train_images, train_labels, validation_data=(val_images, val_labels),
epochs=20, batch_size=500, callbacks=callbacks_list)
       scores = model.evaluate(val_images, val_labels, verbose=0)
       print("CNN Error:         model.save('cnn_model_keras2.h5')

    train()
   K.clear_session();
```

5.2 Gesture Creation

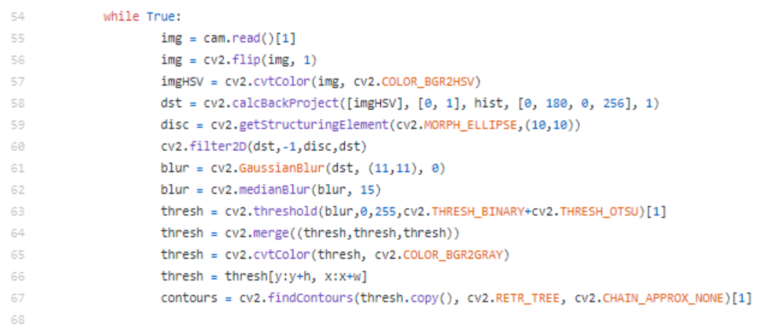
This consists of the code for enabling users to create gestures of their own. We can add gestures when ever we wish to. So it allows improvements in our model. For adding gestures, after taking the hand histogram, the user signs using his/her hand in the contour and the images are captured at 60 frames per second until 1200 images are captured. These images are then stored on the database with the index phrase the user gives.



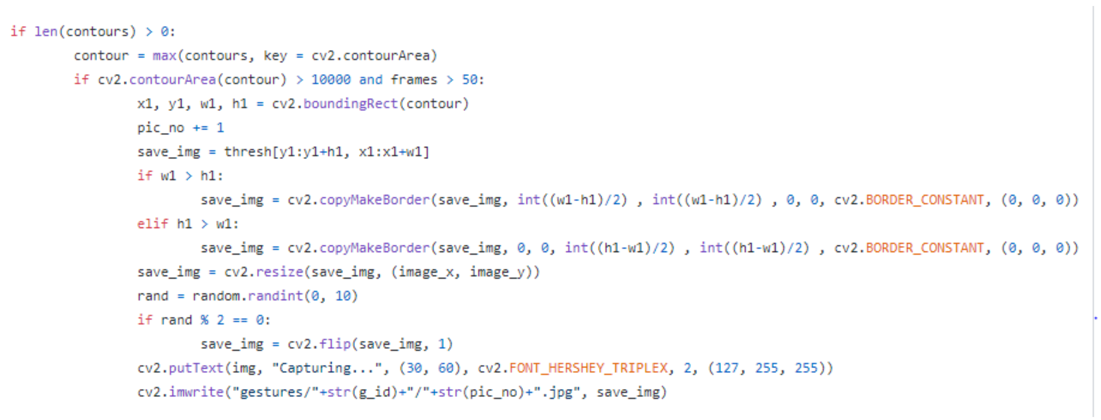
```python
import cv2
import numpy as np
import pickle, os, sqlite3, random

image_x, image_y = 50, 50

def get_hand_hist():
        with open("hist", "rb") as f:
                hist = pickle.load(f)
        return hist

def init_create_folder_database():
        # create the folder and database if not exist
        if not os.path.exists("gestures"):
                os.mkdir("gestures")
        if not os.path.exists("gesture_db.db"):
                conn = sqlite3.connect("gesture_db.db")
                create_table_cmd = "CREATE TABLE gesture ( g_id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT UNIQUE, g_name TEXT NOT NULL )"
                conn.execute(create_table_cmd)
                conn.commit()

def create_folder(folder_name):
        if not os.path.exists(folder_name):
                os.mkdir(folder_name)

```



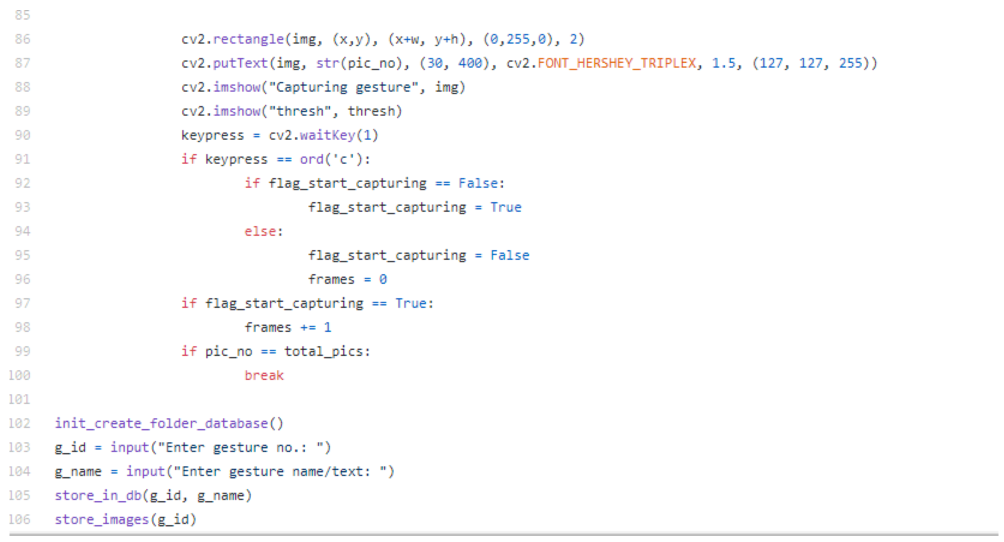
```python
def store_in_db(g_id, g_name):
        conn = sqlite3.connect("gesture_db.db")
        cmd = "INSERT INTO gesture (g_id, g_name) VALUES (%s, \'%s\')" % (g_id, g_name)
        try:
                conn.execute(cmd)
        except sqlite3.IntegrityError:
                choice = input("g_id already exists. Want to change the record? (y/n): ")
                if choice.lower() == 'y':
                        cmd = "UPDATE gesture SET g_name = \'%s\' WHERE g_id = %s" % (g_name, g_id)
                        conn.execute(cmd)
                else:
                        print("Doing nothing...")
                        return
        conn.commit()

def store_images(g_id):
        total_pics = 1200
        hist = get_hand_hist()
        cam = cv2.VideoCapture(1)
        if cam.read()[0]==False:
                cam = cv2.VideoCapture(0)
        x, y, w, h = 300, 100, 300, 300

        create_folder("gestures/"+str(g_id))
        pic_no = 0
        flag_start_capturing = False
        frames = 0
```



```
    while True:
        img = cam.read()[1]
        img = cv2.flip(img, 1)
        imgHSV = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
        dst = cv2.calcBackProject([imgHSV], [0, 1], hist, [0, 180, 0, 256], 1)
        disc = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(10,10))
        cv2.filter2D(dst,-1,disc,dst)
        blur = cv2.GaussianBlur(dst, (11,11), 0)
        blur = cv2.medianBlur(blur, 15)
        thresh = cv2.threshold(blur,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)[1]
        thresh = cv2.merge((thresh,thresh,thresh))
        thresh = cv2.cvtColor(thresh, cv2.COLOR_BGR2GRAY)
        thresh = thresh[y:y+h, x:x+w]
        contours = cv2.findContours(thresh.copy(), cv2.RETR_TREE, cv2.CHAIN_APPROX_NONE)[1]

```



```
if len(contours) > 0:
    contour = max(contours, key = cv2.contourArea)
    if cv2.contourArea(contour) > 10000 and frames > 50:
        x1, y1, w1, h1 = cv2.boundingRect(contour)
        pic_no += 1
        save_img = thresh[y1:y1+h1, x1:x1+w1]
        if w1 > h1:
            save_img = cv2.copyMakeBorder(save_img, int((w1-h1)/2) , int((w1-h1)/2) , 0, 0, cv2.BORDER_CONSTANT, (0, 0, 0))
        elif h1 > w1:
            save_img = cv2.copyMakeBorder(save_img, 0, 0, int((h1-w1)/2) , int((h1-w1)/2) , cv2.BORDER_CONSTANT, (0, 0, 0))
        save_img = cv2.resize(save_img, (image_x, image_y))
        rand = random.randint(0, 10)
        if rand % 2 == 0:
            save_img = cv2.flip(save_img, 1)
        cv2.putText(img, "Capturing...", (30, 60), cv2.FONT_HERSHEY_TRIPLEX, 2, (127, 255, 255))
        cv2.imwrite("gestures/"+str(g_id)+"/"+str(pic_no)+".jpg", save_img)
```



```

        cv2.rectangle(img, (x,y), (x+w, y+h), (0,255,0), 2)
        cv2.putText(img, str(pic_no), (30, 400), cv2.FONT_HERSHEY_TRIPLEX, 1.5, (127, 127, 255))
        cv2.imshow("Capturing gesture", img)
        cv2.imshow("thresh", thresh)
        keypress = cv2.waitKey(1)
        if keypress == ord('c'):
            if flag_start_capturing == False:
                flag_start_capturing = True
            else:
                flag_start_capturing = False
                frames = 0
        if flag_start_capturing == True:
            frames += 1
        if pic_no == total_pics:
            break

init_create_folder_database()
g_id = input("Enter gesture no.: ")
g_name = input("Enter gesture name/text: ")
store_in_db(g_id, g_name)
store_images(g_id)
```

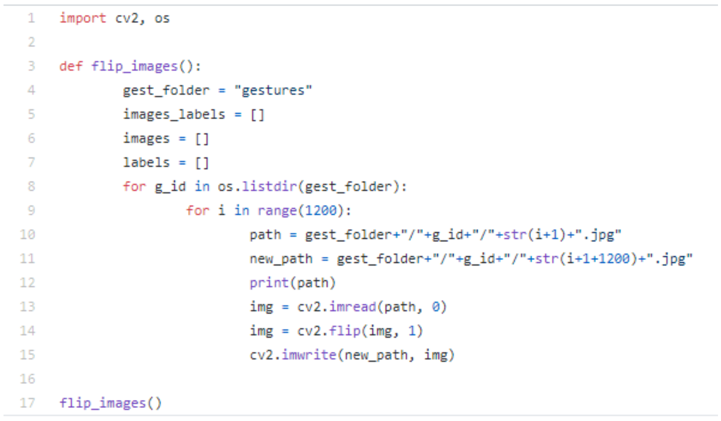
5.3 Displaying Gestures

Code for displaying the images of gestures stored in the database.

```
import cv2, os, random
import numpy as np

def get_image_size():
        img = cv2.imread('gestures/0/100.jpg', 0)
        return img.shape

gestures = os.listdir('gestures/')
gestures.sort(key = int)
begin_index = 0
end_index = 5
image_x, image_y = get_image_size()

if len(gestures)%5 != 0:
        rows = int(len(gestures)/5)+1
else:
        rows = int(len(gestures)/5)

full_img = None
for i in range(rows):
        col_img = None
        for j in range(begin_index, end_index):
                img_path = "gestures/%s/%d.jpg" % (j, random.randint(1, 1200))
                img = cv2.imread(img_path, 0)
                if np.any(img == None):
                        img = np.zeros((image_y, image_x), dtype = np.uint8)
                if np.any(col_img == None):
                        col_img = img
                else:
                        col_img = np.hstack((col_img, img))

        begin_index += 5
        end_index += 5
        if np.any(full_img == None):
                full_img = col_img
        else:
                full_img = np.vstack((full_img, col_img))


cv2.imshow("gestures", full_img)
cv2.imwrite('full_img.jpg', full_img)
cv2.waitKey(0)
```

5.4 Flip Images

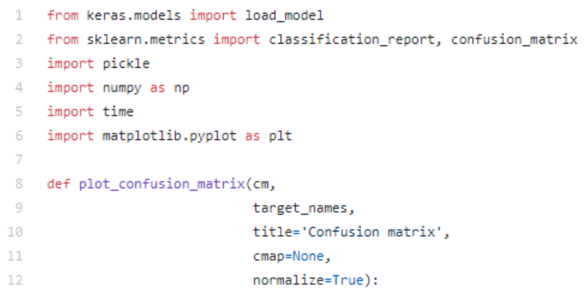
Flipping the image is a vital step for accurate gesture identification. While signing, there are chances that the background lighting conditions can interrupt with the classification process. So converting the images to their binary form eliminates this interference.



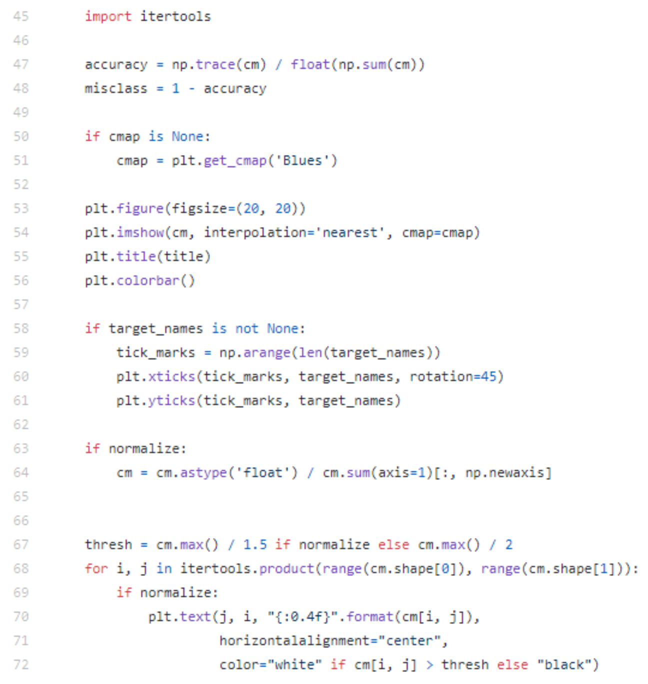
```python
import cv2, os

def flip_images():
        gest_folder = "gestures"
        images_labels = []
        images = []
        labels = []
        for g_id in os.listdir(gest_folder):
                for i in range(1200):
                        path = gest_folder+"/"+g_id+"/"+str(i+1)+".jpg"
                        new_path = gest_folder+"/"+g_id+"/"+str(i+1+1200)+".jpg"
                        print(path)
                        img = cv2.imread(path, 0)
                        img = cv2.flip(img, 1)
                        cv2.imwrite(new_path, img)

flip_images()
```

5.5 Get Model Report

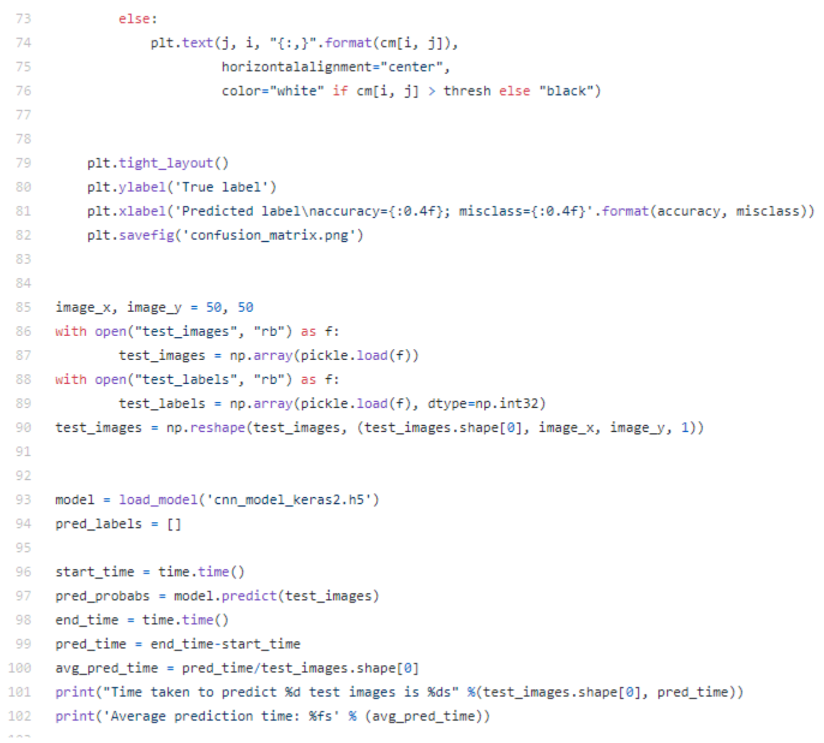
Code given below outputs the overall report of our model. It gives basic information regarding the accuracy, prediction time, loss etc of the model.



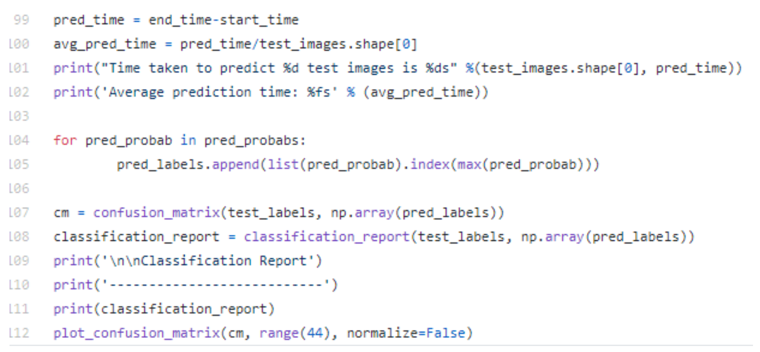
```python
from keras.models import load_model
from sklearn.metrics import classification_report, confusion_matrix
import pickle
import numpy as np
import time
import matplotlib.pyplot as plt

def plot_confusion_matrix(cm,
                          target_names,
                          title='Confusion matrix',
                          cmap=None,
                          normalize=True):
```



```
    import itertools

    accuracy = np.trace(cm) / float(np.sum(cm))
    misclass = 1 - accuracy

    if cmap is None:
        cmap = plt.get_cmap('Blues')

    plt.figure(figsize=(20, 20))
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()

    if target_names is not None:
        tick_marks = np.arange(len(target_names))
        plt.xticks(tick_marks, target_names, rotation=45)
        plt.yticks(tick_marks, target_names)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]


    thresh = cm.max() / 1.5 if normalize else cm.max() / 2
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        if normalize:
            plt.text(j, i, "{:0.4f}".format(cm[i, j]),
                     horizontalalignment="center",
                     color="white" if cm[i, j] > thresh else "black")
```



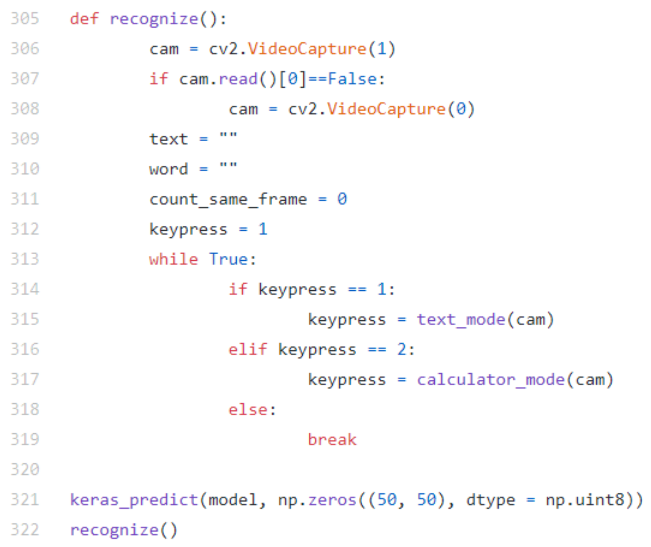
```
        else:
            plt.text(j, i, "{:,}".format(cm[i, j]),
                     horizontalalignment="center",
                     color="white" if cm[i, j] > thresh else "black")


    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label\naccuracy={:0.4f}; misclass={:0.4f}'.format(accuracy, misclass))
    plt.savefig('confusion_matrix.png')


image_x, image_y = 50, 50
with open("test_images", "rb") as f:
        test_images = np.array(pickle.load(f))
with open("test_labels", "rb") as f:
        test_labels = np.array(pickle.load(f), dtype=np.int32)
test_images = np.reshape(test_images, (test_images.shape[0], image_x, image_y, 1))


model = load_model('cnn_model_keras2.h5')
pred_labels = []

start_time = time.time()
pred_probabs = model.predict(test_images)
end_time = time.time()
pred_time = end_time-start_time
avg_pred_time = pred_time/test_images.shape[0]
print("Time taken to predict %d test images is %ds" %(test_images.shape[0], pred_time))
print('Average prediction time: %fs' % (avg_pred_time))
```



```
pred_time = end_time-start_time
avg_pred_time = pred_time/test_images.shape[0]
print("Time taken to predict %d test images is %ds" %(test_images.shape[0], pred_time))
print('Average prediction time: %fs' % (avg_pred_time))

for pred_probab in pred_probabs:
        pred_labels.append(list(pred_probab).index(max(pred_probab)))

cm = confusion_matrix(test_labels, np.array(pred_labels))
classification_report = classification_report(test_labels, np.array(pred_labels))
print('\n\nClassification Report')
print('---------------------------')
print(classification_report)
plot_confusion_matrix(cm, range(44), normalize=False)
```

5.6 Sign Recognition

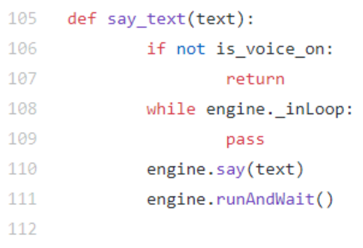
This code fragment classifies the signs shown by the user and outputs the respective translations on the screen.



```
def recognize():
        cam = cv2.VideoCapture(1)
        if cam.read()[0]==False:
                cam = cv2.VideoCapture(0)
        text = ""
        word = ""
        count_same_frame = 0
        keypress = 1
        while True:
                if keypress == 1:
                        keypress = text_mode(cam)
                elif keypress == 2:
                        keypress = calculator_mode(cam)
                else:
                        break

keras_predict(model, np.zeros((50, 50), dtype = np.uint8))
recognize()
```

It also includes the feature of voice outputting. So that apart from displaying the phrase it also speaks out the translation.

It combines different consecutive gestures to create phrases which is a combination of several gestures. If we need to tell a number say '231', then we need to show the gestures for 2, 3, and 1 consecutively. So that the gestures are appended and the application will speak out "two-hundred-thirty-one".



```
def say_text(text):
        if not is_voice_on:
                return
        while engine._inLoop:
                pass
        engine.say(text)
        engine.runAndWait()
```

5.7 Hand Histogram

We used the orientation histogram as a feature vector for gesture classification and interpolation. We seek a simple and fast algorithm, which works in real-time on a workstation. We want the recognition to be relatively robust to changes in lighting. All the computation occurs on a workstation. For dynamic gestures, the histogram of the spatio-temporal gradients of image intensity form the analogous feature vector. For implementing the histogram, OpenCV and Python3 are used.
The implementation of histogram is given below.

```
import cv2
import numpy as np
import pickle
def build_squares(img):
    x, y, w, h = 420, 140, 10, 10
     d = 10
     imgCrop = None
     crop = None
     for i in range(10):
         for j in range(5):
             if np.any(imgCrop == None):
                 imgCrop = img[y:y+h, x:x+w]
             else:
                 imgCrop = np.hstack((imgCrop, img[y:y+h, x:x+w]))
             print(imgCrop.shape)
             cv2.rectangle(img, (x,y), (x+w, y+h), (0,255,0), 1)
             x+=w+d
         if np.any(crop == None):
             crop = imgCrop
         else:
             crop = np.vstack((crop, imgCrop))
         imgCrop = None
         x = 420
         y+=h+d
    return crop

     def get_hand_hist():
     cam = cv2.VideoCapture(1)
     if cam.read()[0]==False:
         cam = cv2.VideoCapture(0)
     x, y, w, h = 300, 100, 300, 300
     flagPressedC, flagPressedS = False, False
     imgCrop = None
     while True:
         img = cam.read()[1]
         img = cv2.flip(img, 1)
         img = cv2.resize(img, (640, 480))
         hsv = cv2.cvtColor(img, cv2.COLORBGR2HSV)
        keypress = cv2.waitKey(1)
        ifkeypress == ord('c') :
            hsvCrop = cv2.cvtColor(imgCrop, cv2.COLOR_BGR2HSV)
            flagPressedC = True
            hist = cv2.calcHist([hsvCrop], [0, 1], None, [180, 256], [0, 180, 0, 256])
            cv2.normalize(hist, hist, 0, 255, cv2.NORM_MINMAX)
        elifkeypress == ord('s') :
            flagPressedS = True
            break
        ifflagPressedC :
            dst = cv2.calcBackProject([hsv], [0, 1], hist, [0, 180, 0, 256], 1)
            dst1 = dst.copy()
            disc = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (10, 10))
            cv2.filter2D(dst, −1, disc, dst)
            blur = cv2.GaussianBlur(dst, (11, 11), 0)
            blur = cv2.medianBlur(blur, 15)
            ret, thresh = cv2.threshold(blur, 0, 255, cv2.THRESH_BINARY+
cv2.THRESH_OTSU)
            thresh = cv2.merge((thresh, thresh, thresh))
            cv2.imshow(”Thresh”, thresh)
        ifnotflagPressedS :
            imgCrop = build_squares(img)
        cv2.imshow(”Sethandhistogram”, img)
    cam.release()
    cv2.destroyAllWindows()
    withopen(”hist”, ”wb”)asf :
        pickle.dump(hist, f)

    get_hand_hist()
```

Chapter 6

Testing

Testing is evaluation of the software against requirements gathered from users and system specification. Testing is necessary to ensure that actual results are the same as the expected results. In the sign language translation process, testing is very important to indicate how well the training procedure of the classification model has been preformed.

Various methods of testing include black box testing, white box testing, integration testing and system testing. It covers all possible test cases that can be encountered and their predictive outputs as well. The test cases designed earlier has been expanded in detail over here.

6.1 Black Box Testing

Black-box testing is a method of software testing that examines the functionality of an application without peering into its internal structures or workings. These tests can be functional or non-functional, though usually functional.

6.1.1 Equivalence Class Partitioning

This approach is use to reduce huge set of possible inputs to small but equally effective inputs. This is done by dividing inputs into the classes and gets one value from each class. Such method is used when exhaustive testing is most wanted to avoid the redundancy of inputs.
Regards to sign language translation, we have 10 main classes depicting 10 different phrases. We take an image from each class and test it with our classifier. Testing is done with each of theses 10 images whose classes are known. Testing remains successful if the actual input is the same as the expected output.

6.2 White Box Testing

White box testing is a testing technique, that examines the program structure and derives test data from the program logic/code. Testing is said to be successful if the examined output deems to be the same as the expected output.

6.2.1 Model Testing

The efficiency of a model is determined by its accuracy. During each step of training the dataset is split into training, testing and validation folders. Where,

- Training dataset is the sample of data used to fit the model.
- Testing dataset is the sample of data used to provide an unbiased evaluation of the final model fit on the training dataset.
- Validation dataset is the sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyperparameters.

While training is done with a set of images, validation using validation images both unique to one another. Once training is complete, we test the training images on the trained model and using this we can find loss. The loss function used here is Cross Entropy. Depending on the value of cross entropy weight adjustments are made. During the next epoch/step the same will happen again. During the same time of giving training data as input to the model, we give validation data as well to ensure that overfitting doesnt take place. Overfitting is when the model only recognizes the training data and nothing more. This continues for however many epoch or steps it is defined for, here we continued the process for 4000 steps. With larger datasets it is advised to increase the number of steps to improve accuracy.
The training accuracy is found at each step, its loss the determined by cross entropy and the validation accuracy. Training accuracy is the percentage of images correctly classified with respect to total number of images. Same goes for validation accuracy, but it is applied to the validation set while training accuracy is applied to the training dataset.

$$TrainingAccuracy = \frac{Total\ number\ of\ images\ in\ training\ set\ classified\ correctly}{Total\ number\ of\ images\ in\ training\ set}$$

$$ValidationAccuracy = \frac{Total\ number\ of\ images\ in\ validation\ set\ classified\ correctly}{Total\ number\ of\ images\ in\ validation\ set}$$

In case of the testing set, it is not labelled. We take an image from it and test it on the model after it is trained.

Chapter 7

Quality Assurance

7.1 Introduction

Software quality assurance (SQA) is a process that ensures that developed software meets and complies with defined or standardized quality specifications. SQA is an ongoing process within the software development life cycle (SDLC) that routinely checks the developed software to ensure it meets desired quality measures. SQA consists of a means of monitoring the software engineering processes and methods used to ensure quality. SQA encompasses the entire software development process, which includes processes such as requirements definition, software design,coding, source code control,code reviews, change management, configuration management, testing, release management, and product integration. SQA is organized into goals, commitments, abilities, activities, measurements, and verification. Rather than checking for quality after completion, SQA processes test for quality in each phase of development until the software is complete. With SQA, the software development process moves into the next phase only once the current/previous phase complies with the required quality standards.

7.2 Purpose

Through our project we implemented a Sign Language Translation System. In the current scenario, those who don't know any sign language are unable to communicate. So there is great need for a system that that overcomes the limitations of the present systems and at the same time is economically feasible. The proposed system solves this problem. The application produces dynamic text output for the recognized gestures.

7.3 Scope

Every member in our team is responsible for the actions planned in this document such as documenting the results throughout the development of the project, reviewing the project progress, and testing the project quality,controlled by this plan. The following are the portions of the software life cycle that are covered by the SQAP: User requirements, Software requirements,Design, Implementation and Testing. The list of software items to be covered in each of the above mentioned life cycle phases are in table. The Software Quality Assurance Plan covers the software items. In addition, the SQAP is also covered by this quality assurance plan. The Software Quality Assurance Plan covers the software items.

Table 7.1: SQAP Plan

Software Lifecycle Phase	**Software Item**
User Requirements	User requirement document
Software Requirements	Software Requirement Specification Document
Design	Software Design Document
Implementation	Document including template of functions
Testing	Document showing testing done in project with summary of results

Chapter 8

Conclusion

This platform is a machine learning based sign language translation application that was developed to recognize signs from Indian Sign Language that includes common words/phrases like Hello, Bye, Best of luck, I love you etc apart from 10 numbers(0-9) and 26 alphabets(a-z). It was trained using a dataset of 4000 images(per gesture). The translation is obtained on the screen almost instantaneously along with a voice output. Furthermore, if hand signs are consecutively shown the sign translations are appended and then converted to voice output that speaks out the combined sentence/numerical number.

The prior step of taking the hand histogram ensures that all the background noises are eliminated before converting the images to binary and finally storing it to the database. This project is the culmination of work done by a group of four passionate computer science students and we hope this simple CNN based application would potentially help millions of speech impaired people provided it is made available effectively to the public.

8.0.1 Limitations

- The application performs at it's worst when the lighting conditions are poor. This would lead to improper detection of the position of the user's hand.
- The contour is static and thus does not move according to the user's hand movements. This restricts the application from recognizing dynamic signs.
- It doesn't include face recognition. Sometimes the same sign can have different meanings according to the expression on the signer's face.
- This application cannot classify two individuals in a screen doing a sign. This is a problem because our classifier is trained to detect one object at once. Including a multi object detection network can change things later on.

8.0.2 Future Plans

- Expanding the dataset is something that would expand its application range in different scenarios.
- Adding face recognition could improve the semantics of sign language as it is a major part of it. As stated early expressions on the face can decide the context of the sign and change its meaning.
- Grammatically correcting the translated text automatically would be a good feature.
- We could also add a sign language dictionary for the ones not literate on ASL.
- Dynamic hand sign recognition would also prove to be a great addition which can be brought about by making the contour non-static and being able to track hand movements.

.

Bibliography

[1] Archana S. Ghotkar and Dr. Gajanan K. Kharate. Study of vision based hand gesture recognition using indian sign language. *INTERNATIONAL JOURNAL ON SMART SENSING AND INTELLIGENT SYSTEMS*, 7(1):96–115, 2014.

[2] Vaishali.S.Kulkarni et al. Appearance based recognition of american sign language using gesture segmentation. *(IJCSE) International Journal on Computer Science and Engineering*, 2(3):560–565, 2010.

[3] Brandon Garcia and Sigberto Alarcon Viesca. *Real-time American Sign Language Recognition with Convolutional Neural Networks.*

[4] Sebastian Kotarski and Bernadetta Maleszka. An efficient method for sign language recognition from image using convolutional neural network. *International Conference on Multimedia and Network Information System*, 833:99–108, 2018.

[5] Ming C. Leu Wenjin Tao and Zhaozheng Yin. American sign language alphabet recognition using convolutional neural networks with multiview augmentation and inference fusion. Technical report, 2018.

[6] Marcell Assan and Kirsti Grobel. Video-based sign language recognition using hidden markov models. Technical report, 2006.

[7] Nasser H. Dardas and Nicolas D. Georganas. Real-time hand gesture detection and recognition using bag-of-features and support vector machine techniques. Technical report, 2011.

[8] R.Z. Khan and N.A. Ibraheem. Survey on gesture recognition for hand image postures. page 110, 2012.

[9] S.H. Gawande J.R. Pansare and M. Ingle. Real-time static hand gesture recognition for american sign language (asl) in complex background. Technical report, 2012.

[10] J. Bhattacharya J. Rekha and S. Majumder. Shape, texture and local movement hand gesture features for indian sign language recognition. Technical report, 2011.

[11] M. Murugeswari ; S. Veluchamy. Hand gesture recognition system for real-time application. Technical report, 2015.

[12] RafiqulZaman Khan Noor Adnan Ibraheem. Survey on various gesture recognition technologies and techniques. Technical report, 2012.

[13] Momoh Jimoh El Salami Amir A. Shafie Sara Bilal1, Rini Akmeliawati. Vision-based hand posture detection and recognition for signlanguage-a study. Technical report, 2011.

[14] Narendra Ahuja Ming-Hsuan Yang. Recognizing hand gesture using motion trajectories. Technical report, 2011.

[15] Narendra Ahuja. A transform for multiscale image segmentation by integrated edge and region detection. Technical report, 1998.

[16] François Bérard Julien Letessier. Visual tracking of bare fingers for interactive surfaces. Technical report, 2011.

[17] Bernd Michaelis Ayoub Al-Hamadi, Omer Rashid. Posture recognition using combined statistical and geometrical feature vectors based on svm. Technical report, 2009.

[18] Nicolas D. Georganas Qing Chen and Emil M. Petriu. Hand gesture recognition using haar-like features and a stochastic context-free grammar. Technical report.

[19] Dennis J. Lin Hanning Zhou and Thomas S. Huang. Static hand gesture recognition based on local orientation histogram feature distribution mode. Technical report, 2005.

[20] Qing Chen Nicolas Georganas Chris Joslin, Ayman El-Sawah. Dynamic gesture recognition. Technical report, 2008.

[toc,page]appendix

Appendix A

User Documentation

A.1 User Manual

The application translates the hand signs of 10 integers(0-9), 26 English alphabets and 8 common English phrases currently, which can be expanded as and when desired using the create gestures module in the project. The corresponding English translation is given as output on the screen and the voice output is spoken out.

A.1.1 Calibration

Initially the user has to place their hand in the green contour on the screen and create a hand histogram. This is where the application detects the colour gradient of the hand with respect to the background. Be sure to place the hand covering all the 50 green squares shown on the screen. The user may stop the process as and when they are satisfied with the histogram. The process is shown in the figures A.1 and A.2.

A.1.2 Translation

During the translation process the user should place their hand in the green contour on the screen and start signing. The application will display the translation and give a voice output as well, which is shown in figures A.3 and A.4.

A.1.3 New Gestures

New gestures can be added to the database only by the developer which can be established using the create_gestures() function in the application. Upon starting, capturing the gesture will begin after a few seconds. Move your hand a little bit here and there. You can pause capturing by pressing 'c' and resume it by pressing 'c'. After the counter reaches 1200 the window will close automatically.

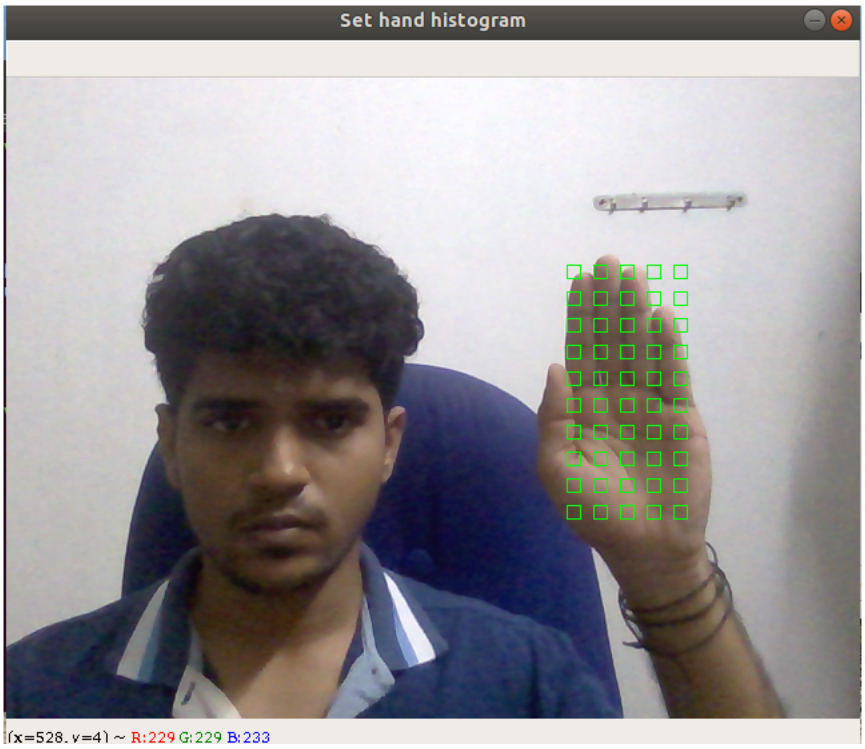


Figure A.1: Hand histogram recording frame

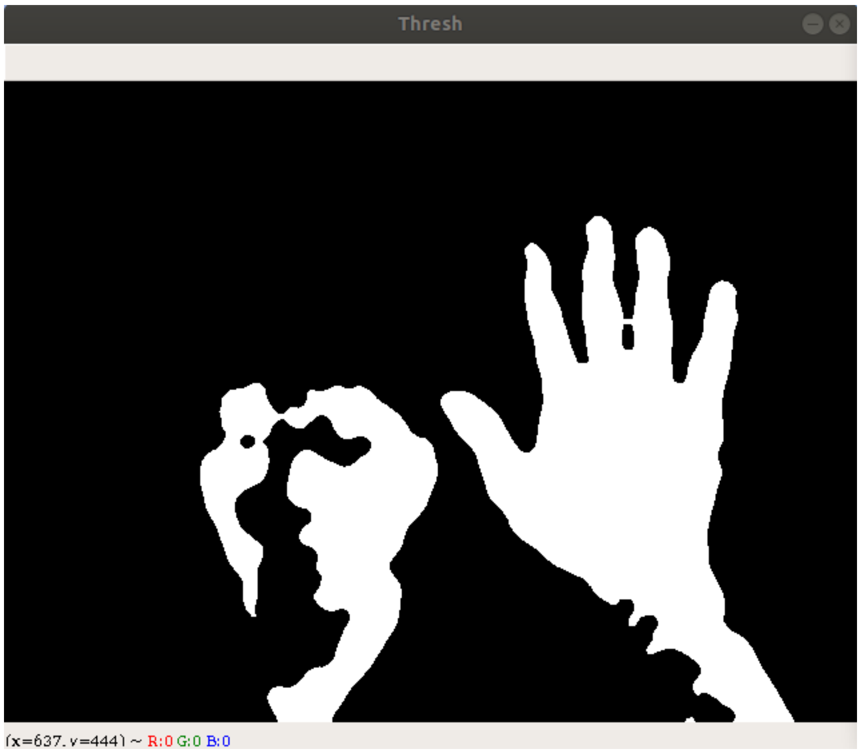


Figure A.2: Hand histogram flipped frame

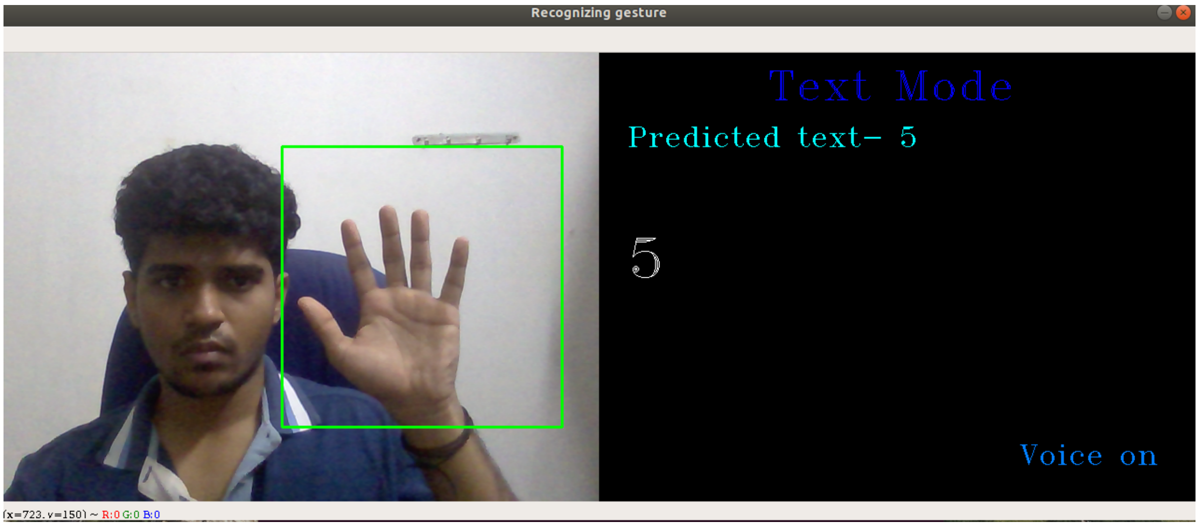


Figure A.3: Prediction Frame (Number)

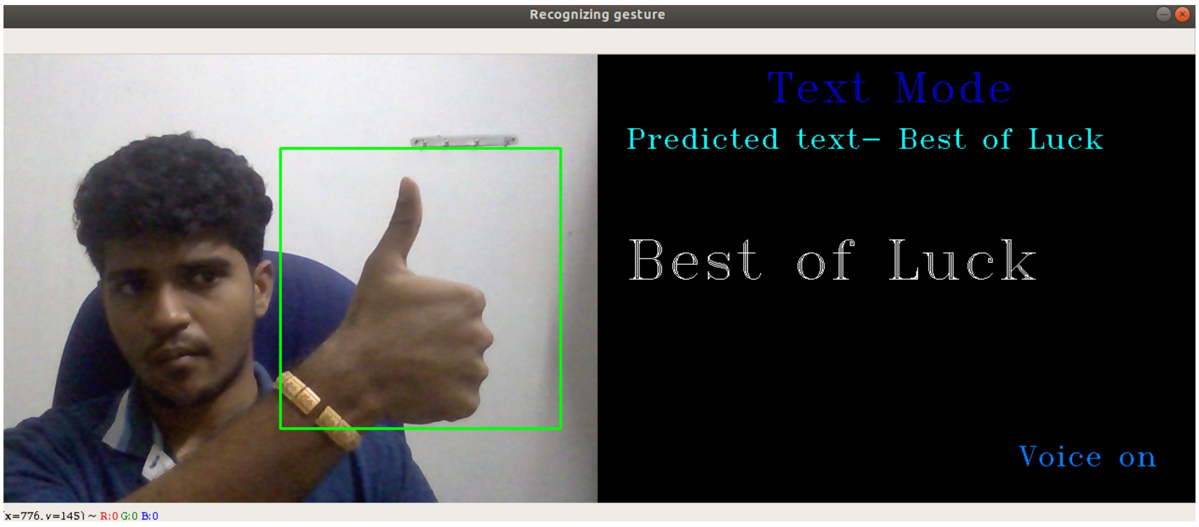


Figure A.4: Prediction Frame (Phrase)