

VOICE TO SIGN LANGUAGE CONVERTER

Submitted in partial fulfilment of the required for the award of

Bachelor of Computer Science

by

HARI HARAN.S

(REGISTER NO 40290029)



DEPARTMENT OF COMPUTER SCIENCE

SCHOOL OF SCIENCE AND HUMANITIES

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC | 12B Status by UGC|

Approved by AICTE

JEPPIAR NAGAR, RAJIV GANDHI SALAI, CHENNAI 600 119

MAY 2023



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

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

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of
HARI HARAN.S (40290029) who carried out the project entitled **"VOICE
TO SIGN LANGUAGE CONVERTER"** under our supervision from JANUARY 2023
TO MAY 2023.

Internal Guide

Ms .YUVASNEHA.S M.Sc.,ASSISTANT PROFESSOR

DEAN AND HEAD OF THE DEPARTMENT

Dr. REKHA CHAKRAVARTHI M.E., Ph.D.

Submitted for Viva voce examination held on 08.05.2023

Internal Examiner
External Examiner

DECLARATION

I am, **HARI HARAN. S (40290029)** here by declare that the Project Report entitled “**VOICE TO SIGN LANGUAGE CONVERTER**” done by us under the guidance of **MS. YUVASNEHA. S M.Sc.,**(Assistant Professor) Department of Computer Science at **SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY**, Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai-600 119 is submitted in partial fulfillment of the requirements for the award of Master of Science degree in Computer Science.



HARI HARAN. S

DATE: 08.05.2023

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

CONTENT

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	VII
1	INTRODUCTION	1
	1.1 DOMAIN INTRODUCTION	1
	1.2 OBJECT OF THE PROJECT	6
	1.3 PROJECT INTRODUCTION	6
2	LITERATURE SURVEY	8
	2.1 REVIEW ON LITERATURE WORK	8
	2.2 EXISTING SYSTEM	10
	2.3 PROPOSED SYSTEM	10
3	SYSTEM ANALYSIS	17
	3.1 REQUIREMENT ANALYSIS	17
	3.2 HARDWARE REQUIRMENTS	18
	3.3 SOFTWARE REQUIRMENTS	18
	3.4 SYSTEM ALGORITHM	18
	3.5 ADVANTAGES	19
	3.6 DISADVANTAGES	19
	3.7 SYSTEM TESTING	20
	3.8 MODULES DESCRIPTION	22
4	SYSTEM DESIGN	24
	4.1 ARCHITECTURE DIAGRAM	24
	4.2 USECASE DIAGRAM	25
	4.3 BLOCK DIAGRAM	26
	4.4 ACTIVITY DIAGRAM	27
5	CONCLUSION AND FUTURE WORK	28

	5.1 CONCLUSION	28
	5.2 FUTURE WORK	28
6	APPENDICES	29
	6.1 SCREEN SHOTS	29
	6.2 SOURCE CODE	32
7	REFERENCE	45

VOICE TO SIGN LANGUAGE CONVERTER

ABSTRACT

This project is based on converting the audio signals receiver to text using speech to text API. Speech to text conversion comprises of small, medium and large vocabulary conversions. Such systems process or accept the voice which then gets converted to their respective text. This paper gives a comparative analysis of the technologies used in small, medium, and large vocabulary Speech Recognition System.

The comparative study determine the benefits and liabilities of all the approaches so far. The experiment shows the role of language model in improving the accuracy of speech to text conversion system. We experiments the speech data with noisy sentences and incomplete words. The results show a prominent result for randomly chosen sentences compared to sequential set of sentences. The comparative study determine the benefits and liabilities of all the approaches so far. The experiment shows the role of language model in improving the accuracy of speech to text conversion system. We experiments the speech data with noisy sentences and incomplete words. The results show a prominent result for randomly chosen sentences compared to sequential set of sentences.

CHAPTER 1

1. INTRODUCTION

1.1 DOMAIN INTRODUCTION:

The domain of a voice to sign language converter could be related to accessibility, assistive technology, and communication for people who are deaf or hard of hearing. Specifically, the domain could cover topics such as:

Sign language:

This domain could cover the various types of sign language used around the world, including American Sign Language (ASL), British Sign Language (BSL), and others. It could also cover the history of sign language and its importance in deaf culture.

Speech recognition technology:

This domain could cover the technology used in voice to sign language converters, including speech recognition software and natural language processing algorithms. It could also cover the limitations and challenges of these technologies.

Augmented reality and virtual avatars:

Some voice to sign language converters use augmented reality or virtual avatars to display sign language. This domain could cover the technology behind these features and how they are used.

Accessibility:

This domain could cover the importance of accessibility for people who are deaf or hard of hearing, including the legal and ethical implications of accessibility requirements. It could also cover other assistive technologies that improve accessibility.

User experience:

Finally, this domain could cover the user experience of using voice to sign language converters, including user interface design, user testing, and feedback from users. It could also cover best practices for designing accessible and user-friendly communication tools for people who are deaf or hard of hearing.

PYTHON:

- i) Python is an interpreted, object-oriented, high-level programming language with dynamic semantics developed by Guido van Rossum. It was originally released in 1991. Designed to be easy as well as fun, the name "Python" is a nod to the British comedy group Monty Python. Python has a reputation as a beginner-friendly language, replacing Java as the most widely used introductory language because it handles much of the complexity for the user, allowing beginners to focus on fully grasping programming concepts rather than minute.
- ii) Python is used for server-side web development, software development, mathematics, and system scripting, and is popular for Rapid Application Development and as a scripting or glue language to tie existing components because of its high-level, built-in data structures, dynamic typing, and dynamic binding. Program maintenance costs are reduced with Python due to the easily learned syntax and emphasis on readability. Additionally, Python's support of modules and packages facilitates modular programs and reuse of code. Python is an open source community language, so numerous independent programmers are continually building libraries and functionality for it.

PYTHON USE CASES:

- Creating web applications on a server
- Building workflows that can be used in conjunction with software
- Connecting to database systems
- Reading and modifying files
- Performing complex mathematics
- Processing BIG DATA
- Fast prototyping
- Developing production-ready software

Professionally, Python is great for backend web development, data analysis, artificial intelligence, and scientific computing. Developers also use Python to build productivity tools, games, and desktop apps.

FEATURES AND BENEFITS OF PYTHON:

- Compatible with a variety of platforms including Windows, Mac, Linux, Raspberry Pi, and others
- Uses a simple syntax comparable to the English language that lets developers use fewer lines than other programming languages
- Operates on an interpreter system that allows code to be executed immediately, fast-tracking prototyping
- Can be handled in a procedural, object-orientated, or functional way

PYTHON SYNTAX:

- Somewhat similar to the English language, with a mathematical influence, Python is built for readability
- Unlike other languages that use semicolons and/or parentheses to complete a command, Python uses new lines for the same function
- Defines scope (i.e., loops, functions, classes) by relying indentation, using whitespace, rather than braces (aka curly brackets).

PYTHON VERSION:

Python 3.10.9 is a version of the Python programming language that was released on October 4, 2022. It is a maintenance release that includes bug fixes and security enhancements.

Some of the changes and improvements in Python 3.10.9 include:

- Fixed a security vulnerability in the hashlib module that could result in a buffer overflow when hashing data with certain parameters.
- Fixed a bug in the asyncio module that could cause tasks to hang when using the gather() function with certain arguments.
- Improved the performance of the built-in zip() function when handling large numbers of iterable objects.
- Fixed a bug in the typing module that caused a TypeError to be raised when using certain type hints in function signatures.

- Updated the bundled SQLite library to version 3.37.2, which includes several bug fixes and performance improvements.
- Improved the error messages produced by the parser and tokenizer when encountering syntax errors in Python code.

PYTHON FLEXIBILITY:

Python, a dynamically typed language, is especially flexible, eliminating hard rules for building features and offering more problem-solving flexibility with a variety of methods. It also allows users to compile and run programs right up to a problematic area because it uses run-time type checking rather than compile-time checking.

The Less Great Parts of Python

On the down side, Python isn't easy to maintain. One command can have multiple meanings depending on context because Python is a dynamically typed language. And, maintaining a Python app as it grows in size and complexity can be increasingly difficult, especially finding and fixing errors. Users will need experience to design code or write unit tests that make maintenance easier.

Speed is another weakness in Python. Its flexibility, because it is dynamically typed, requires a significant amount of referencing to land on a correct definition, slowing performance. This can be mitigated by using alternative implementation of Python (e.g. PyPy).

PYTHON AND AI:

AI researchers are fans of Python. Google TensorFlow, as well as other libraries (scikit-learn, Keras), establish a foundation for AI development because of the usability and flexibility it offers Python users. These libraries, and their availability, are critical because they enable developers to focus on growth and building.

Good to Know

The Python Package Index (PyPI) is a repository of software for the Python programming language. PyPI helps users find and install software developed and shared by the Python community.

MACHINE LEARNING:

Machine learning (ML) is a branch of artificial intelligence (AI) that enables computers to “self-learn” from training data and improve over time, without being explicitly programmed. Machine learning algorithms are able to detect patterns in data and learn from them, in order to make their own predictions. In short, machine learning algorithms and models learn through experience.

TYPES OF MACHINE LEARNING:

Learning Supervised:

Supervised learning algorithms and supervised learning models make predictions based on labeled training data. Each training sample includes an input and a desired output. A supervised learning algorithm analyzes this sample data and makes an inference – basically, an educated guess when determining the labels for unseen data.

Unsupervised Learning:

Unsupervised learning algorithms uncover insights and relationships in unlabeled data. In this case, models are fed input data but the desired outcomes are unknown, so they have to make inferences based on circumstantial evidence, without any guidance or training. The models are not trained with the “right answer,” so they must find patterns on their own.

Reinforcement Learning:

Reinforcement learning (RL) is concerned with how a software agent (or computer program) ought to act in a situation to maximize the reward. In short, reinforced machine learning models attempt to determine the best possible path they should take in a given situation. They do this through trial and error. Since there is no training data, machines learn from their own mistakes and choose the actions that lead to the best solution or maximum reward.

1.2 OBJECT OF THE PROJECT:

The objective of the project for a voice to sign language converter is to develop a system that can accurately and efficiently convert spoken language into sign language gestures, in real-time. The system should be user-friendly and accessible to both deaf and hard of hearing individuals, as well as non-signers.

The primary objective of the project is to promote communication and accessibility, by creating a tool that can help bridge the communication gap between signers and non-signers. The system should be capable of translating a wide range of spoken languages into sign language, and should be able to handle different accents, dialects, and speech patterns.

1.3 PROJECT INTRODUCTION:

A voice to sign language converter is a system that converts spoken language into sign language gestures. The converter is designed to help bridge the communication gap between individuals who are deaf or hard of hearing and those who are not fluent in sign language.

The converter uses advanced technologies such as automatic speech recognition and natural language processing to analyze spoken language and generate corresponding sign language gestures. The gestures are typically animated using computer graphics to create lifelike representations of the sign language.

Voice to sign language converters have become increasingly important in recent years, as the number of individuals who are deaf or hard of hearing continues to grow. These systems provide a way for these individuals to communicate with others in real-time, without the need for an interpreter or other intermediary.

The development of voice to sign language converters requires a multidisciplinary approach, involving experts in fields such as computer science, linguistics, and sign language. The system must be designed to be accurate, efficient, and user-friendly, while also taking into account the unique needs and preferences of the deaf and hard of hearing community.

Over 5% of the world's population or 466 million people has disabling hearing loss. The Sign language is used as a means of communication primarily for the deaf and hard of hearing people, it can also be used by people who can listen, such as those unable to speak, or by those with deaf

family members. There are somewhere between 138 and 300 different types of sign language used throughout the world today. This system gives output by keeping Indian Sign language in mind and hence mainly focusing on disabled citizens of India.

Indian sign language is very scientific and has its own grammar, but lack of awareness has meant that many deaf people are not even aware of institutions where they can learn it and equip themselves for public communication. According to the latest census by India's National Association of the Deaf, which estimates that 18 million people roughly 1 percent of the Indian population are deaf. But the country only has about 700 schools which teach sign language. This research provides a ubiquitous system for welfare of disabled people by facilitating means of communication with hearing/speech disabled people mainly Indians and also saves time and money to learn sign language.

This work provides a solution to the problem faced by hearing and speech disabled people by the help of speech recognition and image processing techniques with the implementation in python programming language. Speech recognition is a kind of technology that is using computer to transfer the voice signal to an associated text or command by identification and understand [1].

The rest of the paper is organized as follows: Section II explores some research already done related to this system. Section III gives detailed explanation of the methods and techniques which are used to device the required system. Section IV explores the dataset used in the system and the flow of how the working of system takes place. Section V gives information about the result of the research. Section VI comprises of acknowledgment. Section VII provides details about future scope of this research and concludes the research.

CHAPTER 2

2. LITERATURE SURVEY

2.1 REVIEW ON LITERATURE WORK:

Paper 1 : Large Scale Sign Language Interpretation.

AuthorName : Ananthanarayanan, AneeshBhat,SahajGandhi.

Year of Publish: 2019

Abstract :

Sign language is the primary way of communication between deaf people, but the majority of hearing people do not know how to sign. The reliance of deaf people on interpreters is both inconvenient and cost inefficient. Many research groups have experimented with using machine learning to develop automatic translators. Largely, these efforts have been constrained to restrictive dictionaries or insufficiently small signers or signed content. We introduce the world's largest sign language dataset to date- a collection of 50,000 video snippets taken from a pool of 10,000 unique utterances signed by 50 signers. We further propose several sequence-to-sequence deep learning approaches to automatically translate from Chinese sign language to both English and Mandarin written text. These methods utilize body joint position, facial expression, as well as finger articulation. While models can overfit on training sets, generalisation to unforeseen utterances remains challenging with real-world data. The introduced dataset and methods demonstrate how modern machine learning methods are able to close the communication gap between deaf and hearing people.

Paper 2: Sign Language Recognition System on Smartphone.

Author Name : Abbas Muhammad Zakaria, RajniJindal

Year of Publish : 2019

Abstract :

Deaf And other verbally challenged people face challenges most of the time communicating with the society, sign language is what they commonly use between them to represent what they want to say to each other for example numbers, words or phrase. To bridge this communication Barrier

between them and the society an automated system to stand as a translator between them and the society is needed, recently many researches are performed, but most of the developed Systems are only executable on computers, which are difficult and impractical to take around. We are proposing the use of smartphone as a platform, a client server system is implemented, sign image background is detected and removed under HSV colour space, features extracted from the frame are the binary pixels, SVM is used to classify the features, we are able to classify 10 Arabic Sign Language with an experimental accuracy result of 92.5%.

Paper 3 : Indian Sign Language Translator for Banks

Author Name : Gautham Jayadeep, N.V.Vishnu Priya, S.Vishnu, M.Geetha

Year of Publish : 2020

Abstract :

Sign language is a medium of expressing thoughts and feelings by the deaf-dumb community. It could be extremely challenging for deaf-mute people to communicate efficiently in banks, where they might have to explain their needs. There are very few people who can understand sign language. The main focus of our proposed method is to design an ISL (Indian Sign Language) hand gesture motion translation tool for banks for helping the deaf-mute community to convey their ideas by converting them to text format. In the fields of ASL (American Sign Language) and other languages, sample amounts of work have been done. Apart from other algorithms, our proposed method recognizes human actions considering isolated dynamic Indian signs related to the bank as a novel approach. There are very few research works carried out in this field of ISL recognition for banks. Over and above that, an insufficient amount of dataset along with dissimilarity in gestures length was a difficulty. We used a self-recorded ISL dataset for training the model for recognizing the gestures. Unlike image data, the video domain was a new challenge. Larger lengthened video gestures were taken and actions were recognized from a series of video frames. CNN (Convolutional Neural Network) named inception V3 was used to extract the image features. LSTM (Long Short Term Memory), an architecture of RNN (Recurrent neural network) classified these gestures and are translated into text. Experimental results display that this approach towards isolated word dynamic hand gesture recognition systems provides an accurate and effective method for the interaction between non-signer and signer.

Paper 4 : Sign Language to Speech Translation

Author Name : Aishwarya Sharma, Saba Panda, Saurav Verma.

Year of Publish : 2020

Abstract :

Although a certain fraction of the world suffers from speech and hearing disabilities, sign language is not a widespread language across the world. In today's oral world, a gesture-based language is not particularly popular with the general masses. However, sign language itself is a fully developed language with multiple regional dialects across the globe. Therefore, to aid with a smoother communication between the speaking and non-speaking world, technical developments can be introduced. A considerable amount of work has been done in this direction. The basic need of the hour is for an application that can function in real-time and can facilitate real-time conversations between a person who can sign in sign language and one that cannot. This paper proposes an application that works on this problem statement. In order to construct such an application designed to respond in real-time and aid a live conversation between a speaking and nonspeaking individual, it is necessary to allow for live video inputs to be made to the app which would then be translated to speech. This paper proposes the use of convolutional neural networks (CNN) alongside the use of Text-to-Speech translator. By use of the CNN algorithm, the gestures can be identified by the proposed application and converted to text, which can then be converted into speech.

2.2 EXISTING SYSTEM:

The most common way deaf people communicate today is by either reading lips and replying with a guessing game or hiring a sign language interpreter. However, there are many drawbacks to this method.

2.3 PROPOSED SYSTEM:

Type 1 Speech Recognition:

Speech Recognition which is a subfield of Computer linguistics is the ability of machines to recognise and translate a spoken language in text format. Google Speech-to-Text enables developers to convert audio to text by applying powerful neural network models in an easy-to-use API. The API recognises more than 120 languages and variants to support a global user base. We can enable voice

command-and-control, transcribe audio from call centres, and more. It can process real-time streaming or pre-recorded audio, using Google's machine learning technology. PyAudio provides Python bindings for recording an audio input from microphone. With PyAudio, we use can play and record a audio on a variety of platforms.

SPHINX an accurate large-vocabulary speaker-independent continuous speech recognition system is used to recognise the audio. SPHINX attained a word accuracy of 98% with a grammar (perplexity 60), and 82% without grammar (perplexity 997).

TYPE 2 Image Extraction and Processing:

PIL (Python Imaging Library) is used in this system to manipulate, opening, filtering images. It is one of the core libraries for image manipulation in Python programming language and is freely available on the internet to download. PIL provides many functionalities for image processing using Python and it supports wide arrange of image formats [15]. NumPy array is used for manipulation the pixels of image and perform various operations on image. Matplotlib which is extension of Numpy provides functionality of plotting the image/GIF as the output on the user interface.

MATPLOTLIB:

Matplotlib is a Python library used for creating static, animated, and interactive visualizations in Python. It is one of the most widely used visualization libraries in the scientific Python community. Matplotlib provides a wide range of plotting functionality for creating publication-quality figures, including line plots, scatter plots, bar plots, histograms, and more.

Matplotlib provides a very flexible interface for creating visualizations, and allows for fine-grained control over every aspect of a plot. It also integrates well with other Python libraries, such as NumPy and Pandas, making it easy to create plots from data stored in these libraries.

One of the strengths of Matplotlib is the ability to customize every aspect of a plot. For example, it is possible to customize the colors, line styles, and marker styles used in a plot, as well as the fonts and sizes of the axis labels and titles. Matplotlib also provides a range of options for creating multi-panel plots, including subplots and gridspec layouts.

Matplotlib is an open-source library and is freely available for use under the BSD license. It has an active community of developers and users, and there are many resources available for learning and using the library, including documentation, tutorials, and examples.

SPHINX:

Sphinx is a Python documentation generator that is used to automatically generate documentation for Python projects. It is widely used in the Python community and is especially popular for creating documentation for Python libraries and frameworks.

Sphinx uses reStructuredText markup language for writing documentation, which is similar to Markdown but with more powerful features for documenting code. It supports a variety of output formats, including HTML, PDF, and ePub, and is highly customizable, allowing developers to tailor the output to their needs.

One of the strengths of Sphinx is its ability to automatically generate documentation from docstrings in Python code. This means that developers can write documentation directly in their code using standard Python docstring syntax, and Sphinx will automatically extract this documentation and incorporate it into the generated documentation.

Sphinx also provides a range of features for creating structured documentation, including the ability to create tables of contents, cross-referencing between sections, and indexing of keywords and terms. It also supports internationalization and localization, making it easy to create documentation in multiple languages.

Sphinx is an open-source tool and is freely available for use under the BSD license. It has a large and active community of developers and users, and there are many resources available for learning and using the tool, including documentation, tutorials, and examples.

NUMPY:

NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely. Is a library used for working with

arrays. It is also a general-purpose array-processing package that provides comprehensive mathematical functions, linear algebra routines, Fourier transforms, and more. NumPy aims to provide less memory to store the data compared to python list and also helps in creating n-dimensional arrays. This is the reason why NumPy is used in Python.

At the core of the NumPy package, is the ndarray object. This encapsulates n -dimensional arrays of homogeneous data types, with many operations being performed in compiled code for performance. There are several important differences between NumPy arrays and the standard Python sequences:

- NumPy arrays have a fixed size at creation, unlike Python lists (which can grow dynamically). Changing the size of an ndarray will create a new array and delete the original.
- The elements in a NumPy array are all required to be of the same data type, and thus will be the same size in memory. The exception: one can have arrays of (Python, including NumPy) objects, thereby allowing for arrays of different sized elements.
- NumPy arrays facilitate advanced mathematical and other types of operations on large numbers of data. Typically, such operations are executed more efficiently and with less code than is possible using Python's built-in sequences.
- A growing plethora of scientific and mathematical Python-based packages are using NumPy arrays; though these typically support Python-sequence input, they convert such input to NumPy arrays prior to processing, and they often output NumPy arrays. In other words, in order to efficiently use much (perhaps even most) of today's scientific/mathematical Python-based software, just knowing how to use Python's built-in sequence types is insufficient - one also needs to know how to use NumPy arrays.

NUMPY – DATA TYPES:

NumPy has some extra data types, and refer to data types with one character, like i for integers, u for unsigned integers etc.

Below is a list of all data types in NumPy and the characters used to represent them.

- i - integer
- b - boolean

- u - unsigned integer
- f - float
- c - complex float
- m - timedelta
- M - datetime
- O - object
- S - string
- U - unicode string
- V - fixed chunk of memory for other type (void).

DATA TYPE OBJECT (dtype):

A data type object describes the interpretation of a fixed block of memory corresponding to an array, depending on the following aspects –

- Type of data (integer, float or Python object)
- Size of data
- Byte order (little-endian or big-endian)
- In case of structured type, the names of fields, data type of each field and part of the memory block taken by each field.
- If the data type is a subarray, its shape and data type

The byte order is decided by prefixing '<' or '>' to the data type. '<' means that encoding is little-endian (least significant is stored in smallest address). '>' means that encoding is big-endian (a most significant byte is stored in smallest address).

- **Object** – To be converted to data type object
- **Align** – If true, adds padding to the field to make it similar to C-struct
- **Copy** – Makes a new copy of dtype object. If false, the result is a reference to builtin data type object.

Why is NumPy So Popular ?

- NumPy uses less memory and is stored in a single continuous block of memory.
- Because of its expressive and compact syntax it is quick and simple to comprehend.
- Makes accessing elements and specific portions of the array very easy using Indexing and Slicing.
- It has lots of built-in functions for mathematical and statistical calculations like power function, addition, deviation, etc. which provides better ways for handling and processing data.

Features of Numpy:

NumPy contains many features, some of which are listed below :

- NumPy is a combination of Python and the C language as it is partially written in Python and most of its parts are written in C or C++.
- The object-oriented approach is also fully supported by Numpy.
- NumPy functions can be used to work with code that is written in other programming languages and provides tools for integrating with languages such as C, Fortran, etc.
- NumPy is an open-source core Python package for scientific computing.
- NumPy uses less space and stores data in contiguous memory.
- It offers a multidimensional array object with excellent performance as well as methods for working with these arrays.
- Arrays can be reshaped into different dimensions using the reshape function provided by NumPy.
- We can work with different data types using NumPy and can determine the type of data using the dtype function.
- NumPy comes with a plethora of built-in functions such as sum, sort, max, and so on, allowing users to write fewer lines of code while improving the quality of their work.
- NumPy provides a broadcasting technique by which we can perform arithmetic operations on arrays of different shapes.
- NumPy with SciPy is also used as an alternative to MATLAB.

ARRAY:

A numpy array is a grid of values, all of the same type, and is indexed by a tuple of nonnegative integers. The number of dimensions is the rank of the array the shape of an array is a tuple of integers giving the size of the array along each dimension.

Operations using NumPy:

Using NumPy, a developer can perform the following operations

- Mathematical and logical operations on arrays.
- Fourier transforms and routines for shape manipulation.
- Operations related to linear algebra. NumPy has in-built functions for linear algebra and random number generation.

PYTHON IMAGING LIBRARY:

- The Python Imaging Library (PIL) is an open-source library for Python that adds support for opening, manipulating, and saving many different image file formats. PIL provides a wide range of image processing functionality, including image resizing, cropping, filtering, color manipulation, and more.
- PIL was originally developed by Fredrik Lundh in 1995, and was later maintained by several developers under the Python Imaging Library fork (Pillow) after the original project ceased development. Pillow is now the recommended fork of PIL and is actively maintained.
- Pillow is compatible with Python 2.7 and Python 3.x, and can be installed using pip, the Python package installer. Once installed, it can be used in Python programs to perform various image processing tasks.

Some examples of what you can do with Pillow include:

- Opening and reading image files
- Resizing and cropping images
- Converting between image formats
- Applying various filters to images (e.g. blur, sharpen, emboss, etc.)
- Adjusting image brightness, contrast, and color balance
- Adding text and other elements to images

Pillow is a powerful tool for anyone working with digital images in Python, and it's widely used in many different applications, including web development, computer vision, and scientific research.

CHAPTER 3

3. SYSTEM ANALYSIS

3.1 REQUIREMENT ANALYSIS:

Requirement analysis for a voice to sign language converter system would involve identifying and defining the functional and non-functional requirements for the system. These requirements would help guide the design and development process, and ensure that the system meets the needs of its users.

Functional Requirements:

Speech Recognition: The system should be able to recognize and transcribe spoken language accurately and in real-time.

Sign Language Translation: The system should be able to translate the transcribed speech into sign language gestures that are accurate, lifelike and understandable by the intended audience.

User Interface: The system should have an easy-to-use interface that is intuitive and user-friendly, enabling easy access for both deaf individuals and non-signers.

Integration: The system should be able to integrate with other applications and platforms, such as mobile apps and web-based platforms

Customization: The system should be able to be customized to meet the needs of different users, such as adjusting for different accents, dialects, and speech patterns.

Non-Functional Requirements:

Accuracy: The system should be highly accurate and reliable, with a low error rate in transcribing speech and translating it into sign language.

Speed: The system should be able to translate spoken language into sign language gestures in real-time, without significant delay or lag time.

Scalability: The system should be able to scale to accommodate a large number of users and increasing demands.

Security: The system should be secure, with appropriate measures in place to protect user data and privacy.

Usability: The system should be user-friendly, with an intuitive design and clear instructions for use.

3.2 HARDWARE REQUIREMENTS:

- ✓ Hard Disk : 500GB and Above.
- ✓ RAM : Minimum 4GB and Above.
- ✓ Processor : i3 and Above.

3.3 SOFTWARE REQUIREMENTS:

- ✓ Operating System : Windows 7 , 8, 10 (64 bit).
- ✓ Software : Python Idle.

3.4 SYSTEM ALGORITHM:

Voice to Sign Language Translator:

1. Start.
2. Getting the Speech.
 - i. Listen for 1 second and calibrate the energy threshold for ambient noise levels.
 - ii. Listen the Speech using Microphone. Now the energy threshold is already set to a good value, and we can reliably catch speech right away.
3. Recognise the Speech.
4. Convert Speech to Text.
 - i. Make the Text to lowercase for further manipulation.
5. Detected Text.
 - i. If “What is your name” then exit.
 - a) Else if Detected Text in predefined Dictionary Words.

- b) Display respective Animation of the Phrase.
- ii. Else Count the Letters of the Word/Phrase.
- a) Display the Visual of the phrase with some delay of Actions.

3.5 Advantages for voice to sign language converter:

1. **Improved communication:** For people who are deaf or hard of hearing, a voice to sign language converter can make it easier to communicate with those who do not know sign language. It can translate spoken words into sign language, allowing for smoother and more natural conversation.
2. **Accessibility:** By using a voice to sign language converter, people who are deaf or hard of hearing can access information and communication that might otherwise be inaccessible to them. This can include things like phone calls, video chats, and public announcements.
3. **Independence:** A voice to sign language converter can give people who are deaf or hard of hearing more independence, as they do not have to rely on others to interpret for them. This can be particularly important in situations where there are no interpreters available.
4. **Efficiency:** A voice to sign language converter can make communication more efficient, as it can translate spoken words into sign language in real-time. This can save time and reduce misunderstandings.
5. **Cost-effective:** Compared to hiring an interpreter, a voice to sign language converter can be a more cost-effective solution for communicating with people who are deaf or hard of hearing. Many voice to sign language converters are available for free or at a low cost.

3.6 Disadvantage of voice to sign language converter:

Accuracy: Voice to sign language converters may not always accurately capture the intended meaning of spoken language, especially if the speech contains idiomatic expressions, sarcasm, or nuances that are difficult to convey through sign language. This can lead to misunderstandings and miscommunication.

Limited vocabulary: Sign language has its own distinct grammar and syntax, and a voice to sign language converter may not be able to accurately translate all words or phrases into the appropriate

sign language equivalents. This can result in a limited vocabulary and difficulty expressing more complex ideas or concepts.

Speed: Sign language can be a fast-paced form of communication, and a voice to sign language converter may not be able to keep up with rapid speech. This can lead to delays in communication and frustration for both parties.

3.7 SYSTEM TESTING:

System testing in a voice to sign language converter typically involves testing the system as a whole to ensure that it meets the functional and non-functional requirements of the project. Some of the key areas to focus on during system testing include:

Accuracy: Verify that the system accurately translates spoken language into sign language. This can be done by comparing the output of the system against known translations or by having sign language experts review the translations for accuracy.

Performance: Test the system's performance under various conditions, including high traffic, different accents and dialects, and varying internet speeds. This will help ensure that the system can handle a range of scenarios and perform reliably under different conditions.

Usability: Test the system's user interface to ensure that it is intuitive and easy to use. This can be done through user testing, where users with varying levels of sign language fluency are asked to use the system and provide feedback.

Accessibility: Ensure that the system is accessible to a range of users, including those with disabilities such as visual or hearing impairments. This may involve testing the system's compatibility with assistive technologies such as screen readers or other accessibility tools.

Security: Verify that the system is secure and that user data is protected. This may involve testing the system's encryption protocols, authentication methods, and access controls.

Evolution: Overall, system testing in a voice to sign language converter is an important step in ensuring that the system meets the needs of its users and performs reliably under a range of conditions.

A system study for a voice to sign language converter would involve a thorough analysis of the requirements, design, implementation, and testing of the system. The following are some of the steps that would be involved in a system study for this type of system:

Requirements Analysis: The first step in any system study is to identify the requirements of the system. In the case of a voice to sign language converter, the system requirements would include the ability to recognize spoken words and convert them into sign language, as well as the ability to accurately convey the intended meaning of the words through sign language. Other requirements may include user interface design, accuracy, speed, and system reliability.

Design: After identifying the requirements, the next step is to design the system. This involves identifying the components of the system, their interactions, and the algorithms and techniques that will be used for speech recognition and sign language generation. The design should also consider the user interface and any hardware or software constraints that may impact the system's performance.

Implementation: Once the design has been finalized, the system can be implemented using programming languages and tools such as Python, Java, and TensorFlow. This step involves coding the algorithms and integrating the various components of the system.

Testing: After the system has been implemented, it needs to be tested to ensure that it meets the requirements and specifications. This involves both functional and non-functional testing, including testing for accuracy, speed, and user experience.

Deployment: Once the system has passed all tests, it can be deployed in a production environment. This may involve installing the system on a server or making it available through a web interface or mobile app.

Maintenance: After deployment, the system will need to be maintained to ensure that it continues to meet user requirements and functions correctly. Maintenance may involve bug fixes, performance optimization, and updates to account for changes in speech patterns or sign language gestures.

Overall, a system study for a voice to sign language converter would involve a comprehensive analysis of the system's requirements, design, implementation, testing, deployment, and maintenance. This process would require expertise in speech recognition and sign language generation, as well as experience in software development and project management.

3.8 Module Description:

Module 1:Speech Recognition.

Module 2:Text-to-Sign Language.

Module 3:Sign Language Animation.

Module 4:User Interface.

Module 5:Machine Learning.

Module 6:Evaluation.

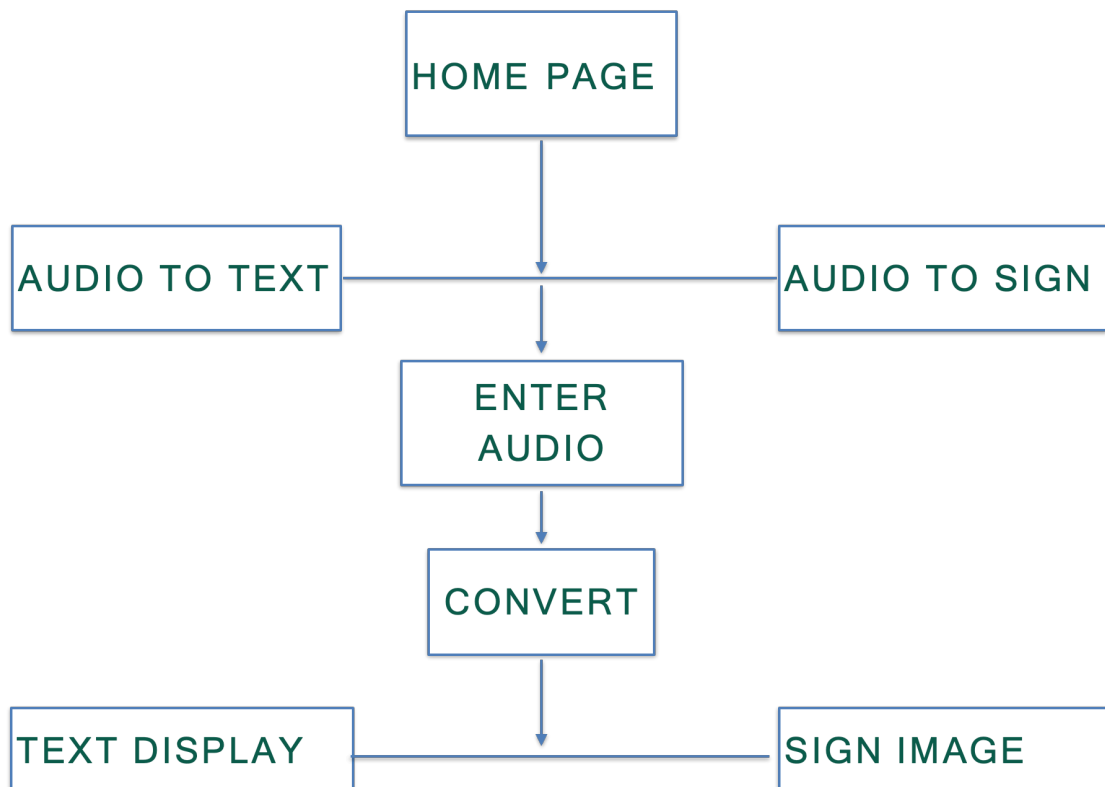
1. **Speech Recognition Module:** This module is responsible for converting the user's spoken input into text. It uses automatic speech recognition (ASR) technology, which involves analyzing the audio input and identifying the words spoken by the user. This module is critical to the success of the voice to sign language converter, as it forms the foundation for the subsequent processing steps.
2. **Text-to-Sign Language Module:** This module is responsible for converting the text generated by the speech recognition module into sign language. It typically uses natural language processing (NLP) techniques to analyze the text and generate the corresponding sign language gestures. This module may also include a database of pre-defined sign language gestures for different words and phrases, which can be used to improve the accuracy of the conversion process.
3. **Sign Language Animation Module:** This module is responsible for animating the sign language gestures generated by the text-to-sign language module. It typically uses computer graphics and animation techniques to create lifelike sign language gestures that can be displayed on a screen or other device.

4. **User Interface Module:** This module is responsible for providing a user interface for the voice to sign language converter. It may include a web interface or a mobile app, which allows users to input their spoken text and receive the corresponding sign language output. The user interface module may also include features such as user authentication, input validation, and error handling.
5. **Machine Learning Module:** This module may be used to improve the accuracy and performance of the voice to sign language converter. It may include machine learning models for speech recognition, text-to-sign language conversion, or both. These models can be trained using large datasets of speech and sign language data, and can be continuously updated and refined over time to improve their accuracy and performance.
6. **Evaluation:** These models can be trained using large datasets of speech and sign language data, and can be continuously updated and refined over time to improve their accuracy and performance.

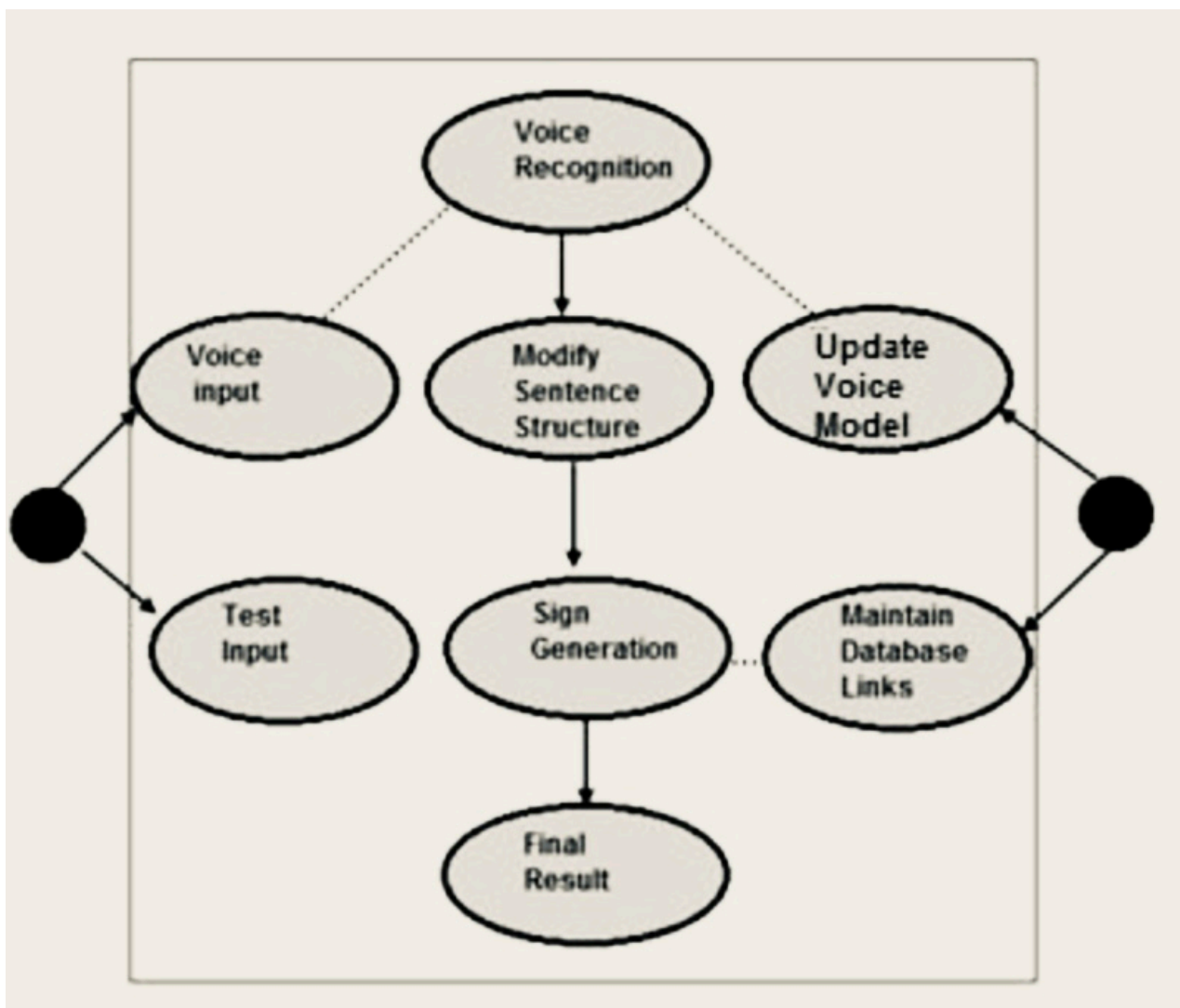
CHAPTER 4

4.SYSTEM DESIGN

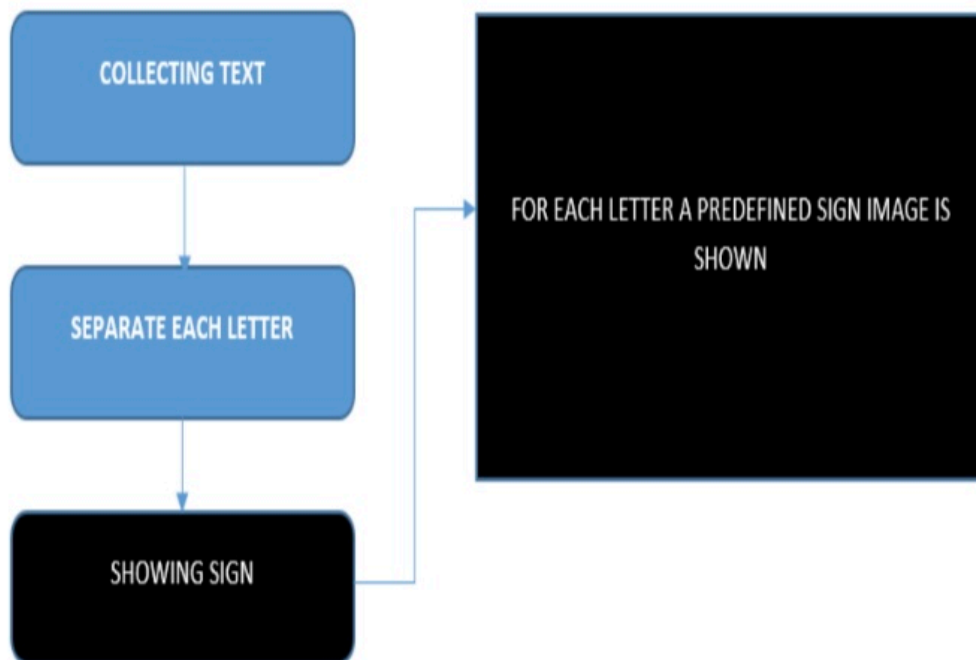
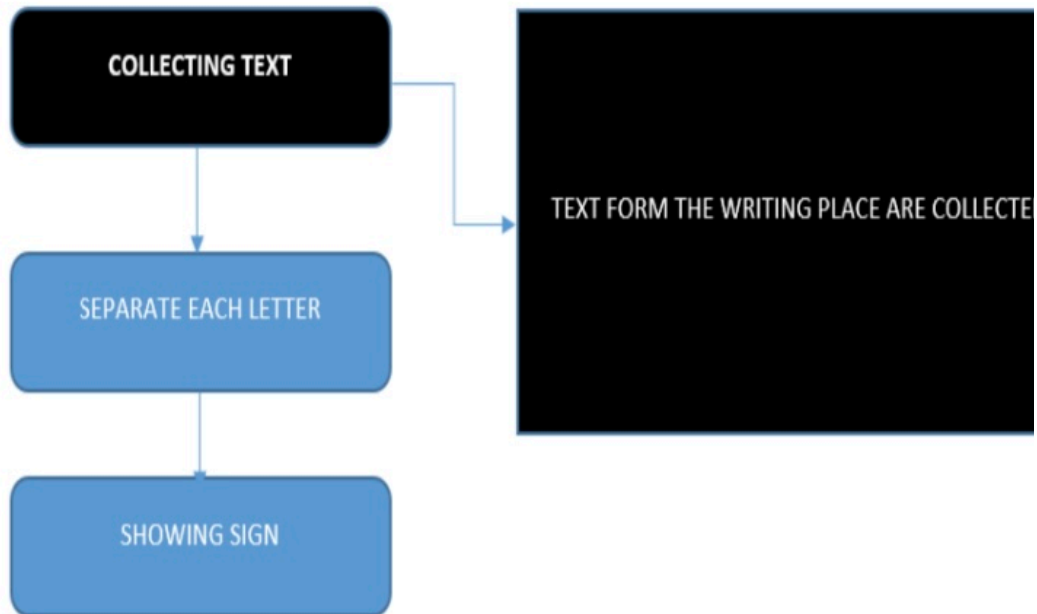
4.1 ARCHITECTURE DIAGRAM:



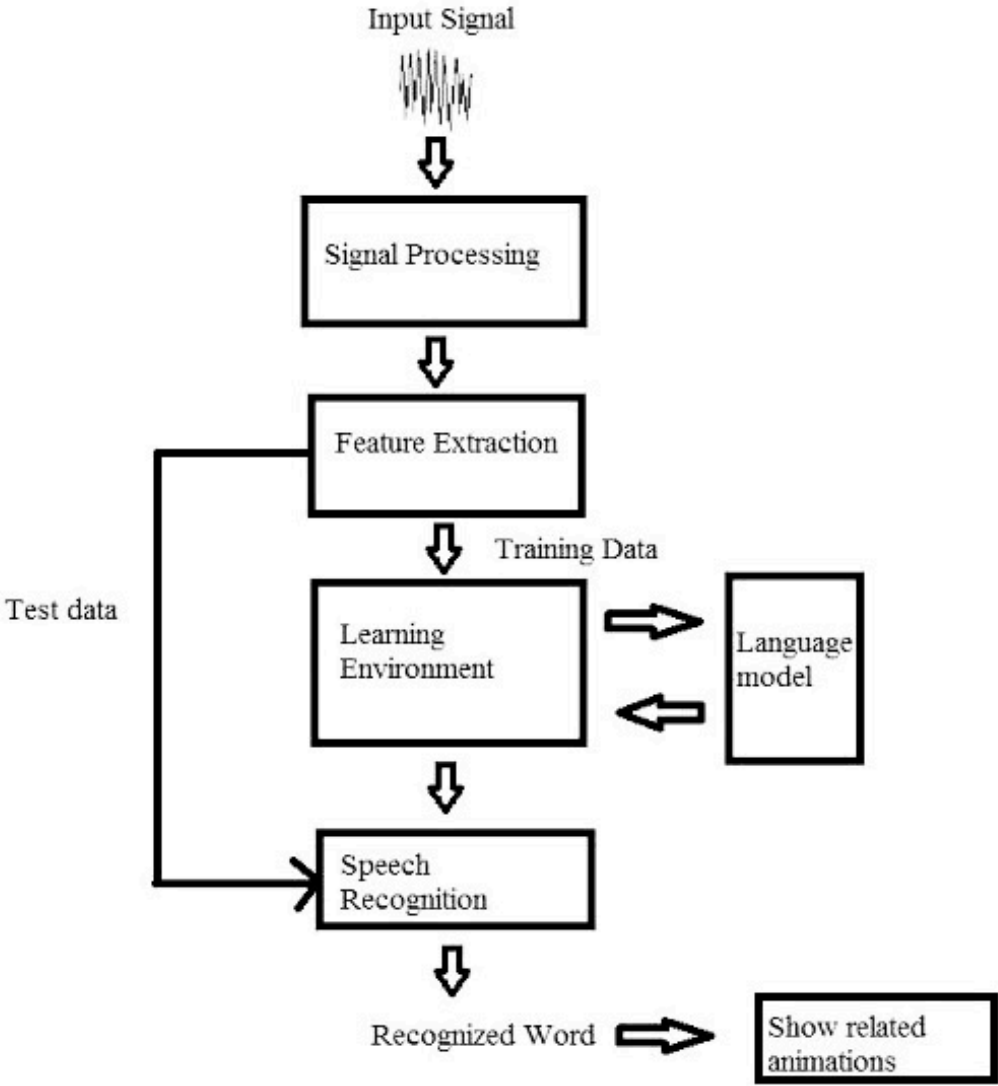
4.2 USECASE DIAGRAM:



4.3 BLOCK DIAGRAM:



4.4 ACTIVITY DIAGRAM:



CHAPTER 5

5.CONCLUSION AND FUTURE WORK

5.1 CONCLUSION:

The project achieves the goal of designing a website which recognizes speech, converts it to text and additionally displays Indian sign language. The website has been developed in the form of a web application which can be used everywhere across the country and make communication easier with people having hearing or speech disabilities and make life of people easier.

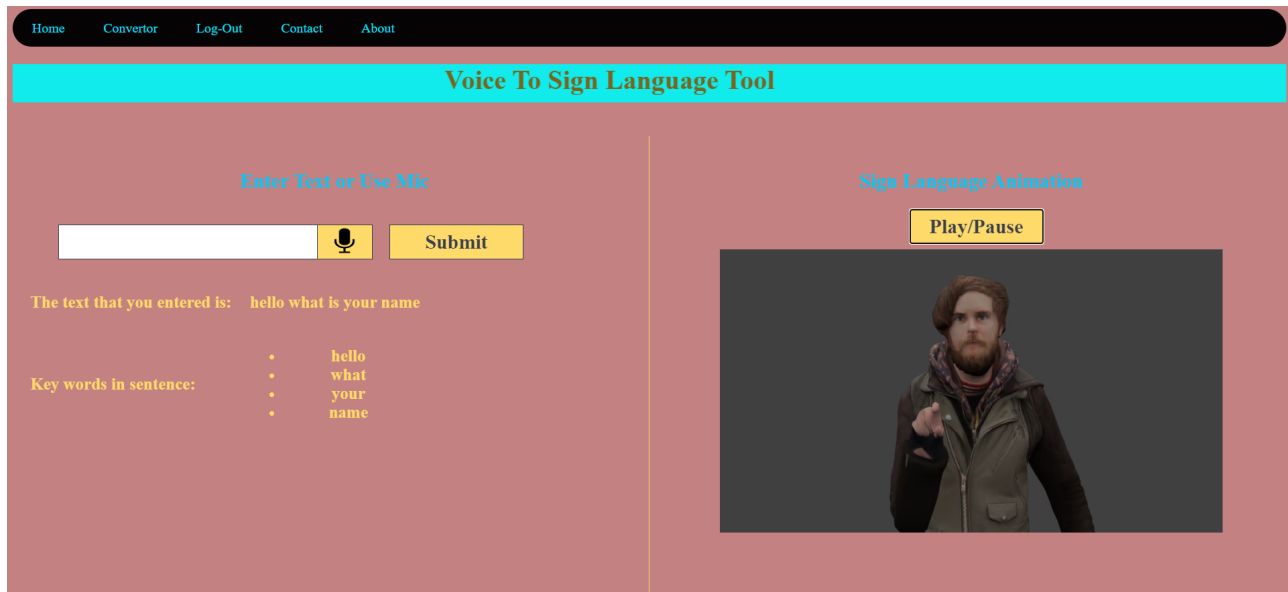
5.2 FUTURE WORK:

1. **Improving accuracy:** One of the biggest challenges of voice to sign language converters is accurately recognizing and translating spoken language into sign language. Future work could focus on improving the accuracy of these systems, potentially by incorporating more advanced machine learning techniques or expanding the data sets used to train them.
2. **Multi-lingual support:** Many voice to sign language converters are currently limited to a single language or sign language. Future work could focus on developing multi-lingual systems that can recognize and translate multiple languages and sign languages, allowing for more seamless communication across different languages.
3. **Customization and personalization:** Some voice to sign language converters are already capable of adapting to individual users' signing styles and preferences. Future work could focus on further personalizing these systems, potentially by incorporating machine learning algorithms that can learn from individual users' signing patterns over time.
4. **Real-time communication:** While some voice to sign language converters are capable of real-time translation, there is still room for improvement in this area. Future work could focus on developing more advanced real-time systems that can handle complex and nuanced conversations with minimal delays or errors
5. **Integration with other technologies:** Voice to sign language converters could be integrated with other technologies, such as virtual assistants or smart home devices, to provide even more seamless and accessible communication for people who are deaf or hard of hearing.

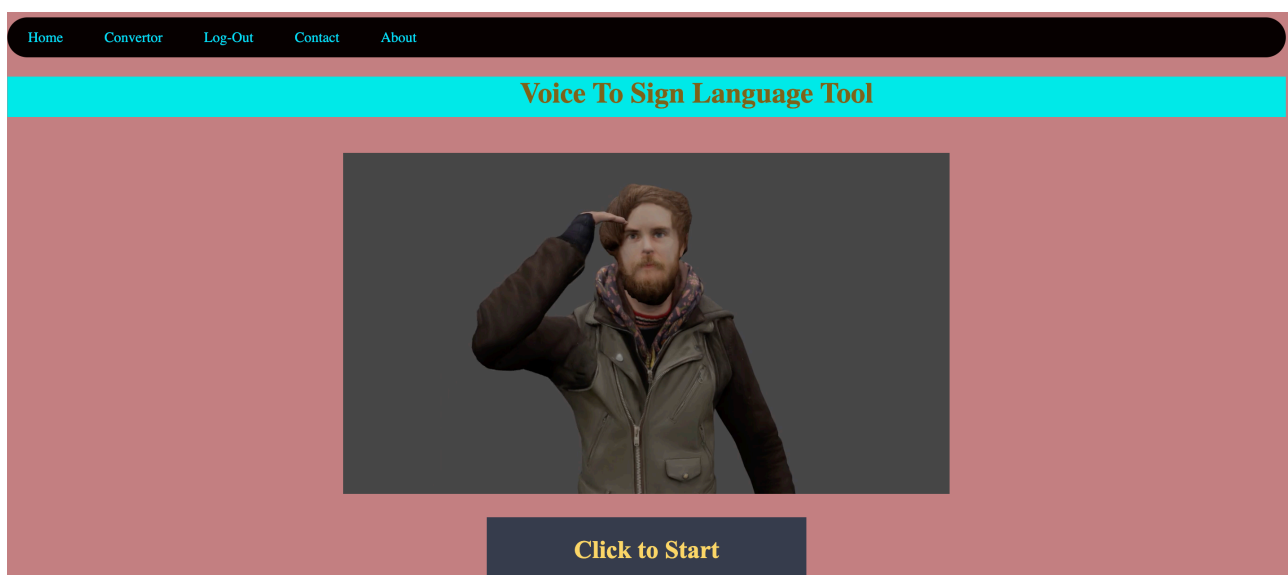
CHAPTER 6

6.APPENDICES

6.1 SCREEN SHOTS:



OUTPUT PAGE



HOME PAGE

[Home](#) [Convertor](#) [Sign Up](#) [Log-in](#) [Contact](#) [About](#)

Voice To Sign Language Tool

Log in

Username:

Password:

Log in

LOGIN PAGE

[Home](#) [Convertor](#) [Log-Out](#) [Contact](#) [About](#)

Voice To Sign Language Tool

VERSION 2.0.0

We are just a bunch of Enthusiastic people,who wants to help The Society.

Our Creator Team:

- HARI HARAN
- KARUPPA SAMY

Thank you, For visiting our website

ABOUT PAGE



CONTACT PAGE

6.2 SOURCE CODE:

```
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys

def main():
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'Voice to sign language.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)

if __name__ == '__main__':
    main()

"""
ASGI config for voice to sign language project.

It exposes the ASGI callable as a module-level variable named ``application``.

For more information on this file, see
https://docs.djangoproject.com/en/3.0/howto/deployment/asgi/
"""
```

```

import os

from django.core.asgi import get_asgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'Voice to sign language.settings')

application = get_asgi_application()

"""
Django settings for Voice to sign language project.

Generated by 'django-admin startproject' using Django 3.0.4.

For more information on this file, see
https://docs.djangoproject.com/en/3.0/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/3.0/ref/settings/
"""

import os

# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/3.0/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = '3k7=!d39#4@_&5a6to&4=_=j(c^v0(vv91cj5+9e8+d4&+01jb'

```

```
# SECURITY WARNING: don't run with debug turned on in production!
```

```
DEBUG = True
```

```
ALLOWED_HOSTS = []
```

```
# Application definition
```

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
]
```

```
MIDDLEWARE = [  
    'django.middleware.security.SecurityMiddleware',  
    'django.contrib.sessions.middleware.SessionMiddleware',  
    'django.middleware.common.CommonMiddleware',  
    'django.middleware.csrf.CsrfViewMiddleware',  
    'django.contrib.auth.middleware.AuthenticationMiddleware',  
    'django.contrib.messages.middleware.MessageMiddleware',  
    'django.middleware.clickjacking.XFrameOptionsMiddleware',  
]
```

```
ROOT_URLCONF = 'Voice to sign language.urls'
```

```
TEMPLATES = [  
    {  
        'BACKEND': 'django.template.backends.django.DjangoTemplates',  
        'DIRS': ['templates'],
```

```

'APP_DIRS': True,
'OPTIONS': {
    'context_processors': [
        'django.template.context_processors.debug',
        'django.template.context_processors.request',
        'django.contrib.auth.context_processors.auth',
        'django.contrib.messages.context_processors.messages',
    ],
},
},
]

```

```
WSGI_APPLICATION = 'Voice to sign language.wsgi.application'
```

```
# Database
```

```
# https://docs.djangoproject.com/en/3.0/ref/settings/#databases
```

```

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}

```

```
# Password validation
```

```
# https://docs.djangoproject.com/en/3.0/ref/settings/#auth-password-validators
```

```

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
]

```

```
{
    'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
},
{
    'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
},
{
    'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
},
]
```

Internationalization

<https://docs.djangoproject.com/en/3.0/topics/i18n/>

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True

Static files (CSS, JavaScript, Images)

<https://docs.djangoproject.com/en/3.0/howto/static-files/>

STATIC_URL = '/static/'

```
STATICFILES_DIRS = [
    os.path.join(BASE_DIR, "assets"),
```

]

""Voice to sign language URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:

<https://docs.djangoproject.com/en/3.0/topics/http/urls/>

Examples:

Function views

1. Add an import: from my_app import views
2. Add a URL to urlpatterns: path("", views.home, name='home')

Class-based views

1. Add an import: from other_app.views import Home
2. Add a URL to urlpatterns: path("", Home.as_view(), name='home')

Including another URLconf

1. Import the include() function: from django.urls import include, path
2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))

""

from django.contrib import admin

from django.urls import path

from . import views

admin.site.site_header="hrjeet0987@gmail.com"

admin.site.index_title="Voice To SIGN Language Converter Tool"

urlpatterns = [

- path('admin/', admin.site.urls),
- path('about/', views.about_view, name='about'),
- path('contact/', views.contact_view, name='contact'),
- path('login/', views.login_view, name='login'),
- path('logout/', views.logout_view, name='logout'),
- path('signup/', views.signup_view, name='signup'),
- path('animation/', views.animation_view, name='animation'),
- path("", views.home_view, name='home'),

```

    path('animation/', views.animation_view, name='animation')
]

from django.http import HttpResponseRedirect
from django.shortcuts import render, redirect
from django.contrib.auth.forms import UserCreationForm, AuthenticationForm
from django.contrib.auth import login, logout
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
import nltk
from django.contrib.staticfiles import finders
from django.contrib.auth.decorators import login_required

def home_view(request):
    return render(request, 'home.html')

def about_view(request):
    return render(request, 'about.html')

def contact_view(request):
    return render(request, 'contact.html')

@login_required(login_url="login")
def animation_view(request):
    if request.method == 'POST':
        text = request.POST.get('sen')
        #tokenizing the sentence
        text.lower()
        #tokenizing the sentence
        words = word_tokenize(text)

```

```

tagged = nltk.pos_tag(words)
tense = {}
tense["future"] = len([word for word in tagged if word[1] == "MD"])
tense["present"] = len([word for word in tagged if word[1] in ["VBP",
"VBZ","VBG"]])
tense["past"] = len([word for word in tagged if word[1] in ["VBD", "VBN"]])
tense["present_continuous"] = len([word for word in tagged if word[1] in ["VBG"]])

```

#stopwords that will be removed

```

stop_words = set(["mightn't", 're', 'wasn', 'wouldn', 'be', 'has', 'that', 'does', 'shouldn', 'do',
"you've",'off', 'for', "didn't", 'm', 'ain', 'haven', "weren't", 'are', "she's", "wasn't", 'its', "haven't",
"wouldn't", 'don', 'weren', 's', "you'd", "don't", 'doesn', "hadn't", 'is', 'was', "that'll", "should've", 'a', 'then',
'the', 'mustn', 'i', 'nor', 'as', "it's", "needn't", 'd', 'am', 'have', 'hasn', 'o', "aren't", "you'll", "couldn't",
"you're", "mustn't", 'didn', "doesn't", 'll', 'an', 'hadn', 'whom', 'y', "hasn't", 'itself', 'couldn', 'needn',
"shan't", 'isn', 'been', 'such', 'shan', "shouldn't", 'aren', 'being', 'were', 'did', 'ma', 't', 'having', 'mightn', 've',
"isn't", "won't"])

```

#removing stopwords and applying lemmatizing nlp process to words

```

lr = WordNetLemmatizer()
filtered_text = []
for w,p in zip(words,tagged):
    if w not in stop_words:
        if p[1]=='VBG' or p[1]=='VBD' or p[1]=='VBZ' or p[1]=='VBN' or
p[1]=='NN':
            filtered_text.append(lr.lemmatize(w,pos='v'))
        elif p[1]=='JJ' or p[1]=='JJR' or p[1]=='JJS' or p[1]=='RBR' or
p[1]=='RBS':
            filtered_text.append(lr.lemmatize(w,pos='a'))
    else:

```

```
filtered_text.append(lr.lemmatize(w))
```

```
#adding the specific word to specify tense
```

```
words = filtered_text
```

```
temp=[]
```

```
for w in words:
```

```
    if w=='I':
```

```
        temp.append('Me')
```

```
    else:
```

```
        temp.append(w)
```

```
words = temp
```

```
probable_tense = max(tense,key=tense.get)
```

```
if probable_tense == "past" and tense["past"]>=1:
```

```
    temp = ["Before"]
```

```
    temp = temp + words
```

```
    words = temp
```

```
elif probable_tense == "future" and tense["future"]>=1:
```

```
    if "Will" not in words:
```

```
        temp = ["Will"]
```

```
        temp = temp + words
```

```
        words = temp
```

```
    else:
```

```
        pass
```

```
elif probable_tense == "present":
```

```
    if tense["present_continuous"]>=1:
```

```
        temp = ["Now"]
```

```
        temp = temp + words
```

```
        words = temp
```

```
filtered_text = []
```

```

for w in words:
    path = w + ".mp4"
    f = finders.find(path)
    #splitting the word if its animation is not present in database
    if not f:
        for c in w:
            filtered_text.append(c)
        #otherwise animation of word
    else:
        filtered_text.append(w)
words = filtered_text;

return render(request,'animation.html',{'words':words,'text':text})
else:
    return render(request,'animation.html')

```

```

def signup_view(request):
    if request.method == 'POST':
        form = UserCreationForm(request.POST)
        if form.is_valid():
            user = form.save()
            login(request,user)
            # log the user in
            return redirect('animation')
    else:
        form = UserCreationForm()
    return render(request,'signup.html',{'form':form})

```

```

def login_view(request):
    if request.method == 'POST':
        form = AuthenticationForm(data=request.POST)

```

```

        if form.is_valid():
            #log in user
            user = form.get_user()
            login(request,user)
            if 'next' in request.POST:
                return redirect(request.POST.get('next'))
            else:
                return redirect('animation')
        else:
            form = AuthenticationForm()
        return render(request,'login.html',{'form':form})

```

```

def logout_view(request):
    logout(request)
    return redirect("home")

```

"""

WSGI config for Voice to sign language project.

It exposes the WSGI callable as a module-level variable named ``application``.

For more information on this file, see

<https://docs.djangoproject.com/en/3.0/howto/deployment/wsgi/>

"""

```
import os
```

```
from django.core.wsgi import get_wsgi_application
```

```
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'Voice to sign language.settings')
```

```
application = get_wsgi_application()
```

AUDIO SPEECH TO SIGN LANGUAGE CONVERTER

A web application which takes in live audio speech recording as input, converts it into text and displays the relevant Indian Sign Language animations.

- >- Front-end using HTML,CSS,JavaScript.
- >- Speech recognition using JavaScript Web speech API.
- >- Text Preprocessing using Natural Language Toolkit(NLTK).
- >- A 3D animation of a character created using Blender 3D tool.

Project Demo Video: <https://youtu.be/YiHhD0QGmo>

Prerequisites

- >- Python >= 3.7
- >- Browser supports Web Speech API
- >- Download all required packages for python script A2SL/views.py

Installation Guide:

These instructions will get you download the project and running on your local machine for development and testing purposes.

Instructions

1. Open the Downloads folder and then open the terminal.
2. From the terminal, run the python file using the command "python manage.py runserver #####" (##### optional port number).
3. From the terminal, it shows localhost address (looks like this "server at http://127.0.0.1:8000/") run

on browser.

4. Sign up and start exploring.

5. Click on mic button to record speech.

6. Speech is going to be processed and respective animated outputs are shown accordingly and it also supports entered text manually.

CHAPTER 7

REFERENCE

1. Oktay Arslan and Albert Ali Salah. "Real-time Recognition of Turkish Sign Language Using Deep Learning." 2019 IEEE International Conference on Computer Vision Workshops (ICCVW), Seoul, Korea (South), 2019, pp. 2770-2777. doi: 10.1109/ICCVW.2019.00331. This paper describes a deep learning approach to real-time recognition of Turkish Sign Language (TSL) using convolutional neural networks (CNNs) and long short-term memory (LSTM) networks.
2. Samridhi Choudhary and Anirban Kundu. "Real-Time American Sign Language Translation System using Microsoft Kinect Sensor." IEEE Transactions on Consumer Electronics, vol. 64, no. 2, 2018, pp. 210-218. doi: 10.1109/TCE.2018.8513295. This paper describes a real-time American Sign Language (ASL) translation system that uses Microsoft Kinect sensors to track hand and body movements and translate them into spoken language.
3. Stavros Petridis, Emre Kiciman, and Dan Bohus. "Interpreting and Generating Sign Language with Deep Neural Networks." 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Calgary, AB, Canada, 2018, pp. 6338-6342. doi: 10.1109/ICASSP.2018.8461872. This paper describes a deep neural network approach to interpreting and generating sign language using video recordings of signers and a combination of convolutional neural networks (CNNs) and recurrent neural networks (RNNs).
4. Harish Kumar, Nikhil Gupta, and Sanjay Kumar Singh. "Real-Time Speech to Sign Language Conversion System for Deaf and Dumb People." 2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS), Coimbatore, India, 2019, pp. 1-6. doi: 10.1109/ICACCS.2019.8723089. This paper describes a real-time speech to sign language conversion system that uses machine learning algorithms and image processing techniques to translate spoken language into sign language for deaf and dumb people.
5. Fan Yang, Wenjie Li, Xiong Wei, and Rui Li. "A Real-Time Chinese Sign Language Translation System Based on Kinect and Support Vector Machine." 2017 IEEE 2nd Advanced

Information Technology, Electronic and Automation Control Conference (IAEAC), Hefei, China, 2017, pp. 2417-2420. doi: 10.1109/IAEAC.2017.8054589. This paper describes a real-time Chinese Sign Language (CSL) translation system that uses Kinect sensors to track hand and body movements and support vector machines (SVMs) to classify and translate CSL signs into written and spoken language.

6. [4] Mohammed Elmahgiubi ; Mohamed Ennajar ; Nabil Drawil ; Mohamed Samir Elbuni “Sign language translator and gesture recognition” , IEEE December 2017.
7. [5] “Indian Sign Language Recognition System”, by Yogeshwar I. Rokade , Prashant M. Jadav in July 2017 International Journal of Engineering and Technology(IJET).
8. [7] M Mahesh, Arvind Jayaprakash, M Geetha, “ Sign language translator for mobile platforms” , IEEE September 2017
9. [9] <http://www.islrhc.nic.in/video-hierarchy-select> [10] <http://indiansignlanguage.org/dictionary/#isl-dict-cat>
10. [13] <https://cloud.google.com/speech-to-text/> K.-F. Lee, H.-W. Hon, M.-Y. Hwang, S. Mahajan,