International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 3 Issue 3, March - 2014

An Adaptable Speech to Sign Language Translation System

Shekainah Paulson
II M.Tech. Embedded Systems, Dept. of Electronics and Instrumentation
Karunya University
Coimbatore, India

Mrs. B. Thilagavathi
Assistant Professor, Dept. of Electronics and Instrumentation
Karunya University
Coimbatore, India

***Abstract*— This project is a new version of a speech to sign language translation system with new tools and characteristics for increasing adaptability to a new task or a new semantic domain. It consists of a speech recognizer that converts spoken sentences into utterances and silences, and recognizes it as text- a sequence of words, and a video displaying the sign language interpretation of the spoken sentence. It is an adaptable system capable of reducing significantly the effort and the parallel corpus needed for adapting a speech to sign language translation system to a new domain.**

Keywords- Speech recognition, Sphinx 4.0, American Sign Language, Java, Eclipse IDE.

I. INTRODUCTION

The deaf are a growing population of every nation. Communication with this group of people is a problem with just one solution, that of sign language. The deaf learn to communicate with the hearing through this dialect, if it may be called so, with a grammar and style of usage of its own. However, the ratio of deaf people to those who can interpret sign language is one of concern, with an average of 93:1 in the US. The goal of this project is to benefit the deaf, who sign language is a prerequisite to communicate with, in the absence of interpreters who can translate to and from sign language and spoken languages. Hence, it greatly benefits the deaf who wish to lead a normal day-to-day life, performing activities like face-to-face interaction with a government employee, etc.

II. SYSTEM OVERVIEW

The system consists of a speech recognizer, Sphinx 4.0, written in Java and executed using the Eclipse IDE. The input is a sentence spoken into the microphone of the system, and the output is recognized speech in the form of text (sequence of words) and a video displaying the spoken sentence in sign language. This project is a prototype of the main system, hence five basic sentences (Good Morning, Hello, Sit down and I am Sorry) have been chosen, to be recognized by the program. Stored videos are then displayed for each sentence. Figure 1 is the block diagram of the whole system, depicting the overall conversion process of speech to text and sign language.

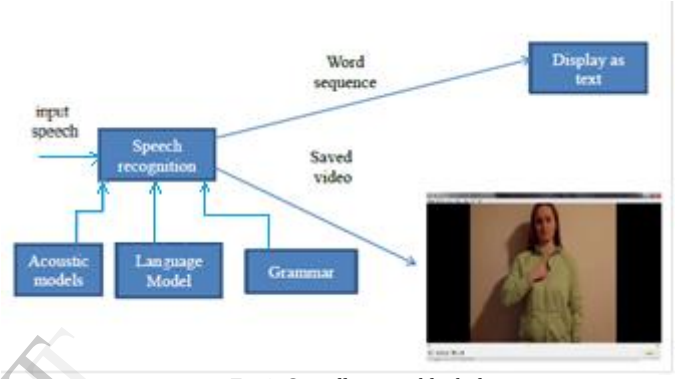

Fig.1. Overall system block diagram

III. SPEECH RECOGNITION

Speech recognition (SR), in the field of computer science, is the translation of speech (spoken words) into text. This is also known as "automatic speech recognition", "ASR", "STT", "speech to text", or just "computer speech recognition". Speech processing and recognition has, since early days, always been an active area of research in computer science.

The speech recognition system used in this project is Sphinx-4, a speech recognition algorithm written entirely in the Java programming language. Sphinx-4 is a flexible, modular and pluggable framework to help foster new innovations in the core research of Hidden Markov model (HMM) recognition systems. Sphinx-4 has been designed based on patterns emerged from the design of past systems as well as new requirements based on areas that researchers currently want to explore. Sphinx-4 also includes several implementations of both simple and state-of-the-art techniques, to exercise this framework, and to provide researchers with a “research-ready” system. The framework and the implementations are all freely available via open source, that is, they are open for downloading and editing.

Sphinx-4’s framework has been designed with flexibility and modularity, as much as possible, for the convenience of users. Figure 2 shows the framework of the speech recognition system. Each labeled element in the figure represents a module that can be easily replaced, leaving room for experimentation. There are three primary modules in the Sphinx-4 framework: the *FrontEnd*, the *Decoder*, and the *Linguist*. The FrontEnd takes one or a set of input signals and

parameterizes them into a sequence of *Features*. The Linguist converts any type of standard language model, along with pronunciation information from the *Dictionary* and structural information from the *AcousticModels*, into a *SearchGraph*. The *SearchManager* in the Decoder uses Features from the FrontEnd and the SearchGraph from the Linguist to perform the actual decoding, generating *Results*. At any time prior to or during the recognition process, the application can issue *Controls* to each of the modules, effectively becoming a partner in the recognition process.

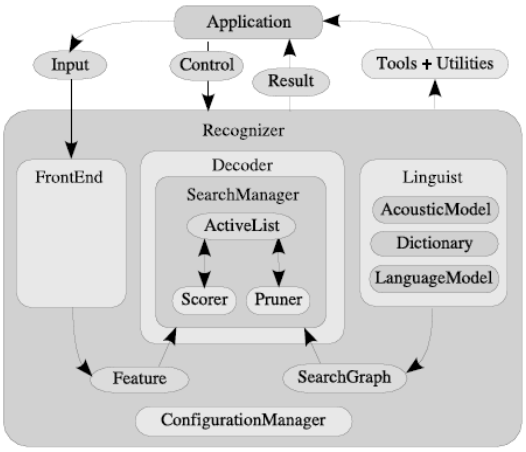

Fig. 2. Sphinx-4 decoder framework

Like most speech recognition systems, the Sphinx-4 system has a large number of configurable parameters, like search beam size, for tuning the system performance. The *ConfigurationManager* is used to configure such parameters. However, what makes Sphinx-4 so unique is that the ConfigurationManager also enables it to dynamically load and configure modules at run time, yielding a flexible and pluggable system. For example, Sphinx-4 is typically configured with a FrontEnd that produces MFCCs or Mel-Frequency Cepstral Coefficients. Using the ConfigurationManager, however, it is possible to reconfigure Sphinx-4 to construct a different FrontEnd that produces Perceptual Linear Prediction coefficients (PLP) without needing to modify any source code or recompile the system. To give applications and developers the ability to track decoder statistics such as word error rate, runtime speed, and memory usage, Sphinx-4 also provides a number of *Tools*. As with the rest of the system, the Tools are highly configurable, allowing users to perform a wide range of system analysis. Furthermore, the Tools also provide an interactive runtime environment that allows users to modify the parameters of the system while the system is running, allowing for rapid experimentation with various parameters settings.

Sphinx-4 also provides support for *Utilities* that support application-level processing of recognition results. For example, these utilities include support for obtaining result lattices, confidence scores, and natural language understanding.

IV. ECLIPSE IDE

Eclipse is a vast extendable set of tools for software development. This project involved the use of Eclipse's Integrated Development Environment (IDE) component for writing Java software. Eclipse is an open source project of Eclipse Foundation. Eclipse is available free of charge under the Eclipse Public License.

Eclipse runs on multiple platforms including Windows, Linux, and Mac OS. There may be minor differences between Eclipse versions for different platforms and operating systems, but the core features work the same way. This project made use of the 2013 version of Eclipse, known as Kepler Release for Java Developers. There are many ways to learn how to learn to language in Java, as are advantages to learning Java using the Eclipse integrated development environment (IDE). Some of these are that Eclipse provides a number of aids that make writing Java code much quicker and easier than using a text editor. This means that more time can be spent learning Java, and less time typing and looking up documentation. The Eclipse debugger and scrapbook allows one to look inside the execution of the Java code. This allows the programmer to "see" objects and to understand how Java is working behind the scenes. Eclipse provides full support for agile software development practices such as test-driven development and refactoring.

V. PROGRAM

The flowchart of the program used is shown in Figure 3.

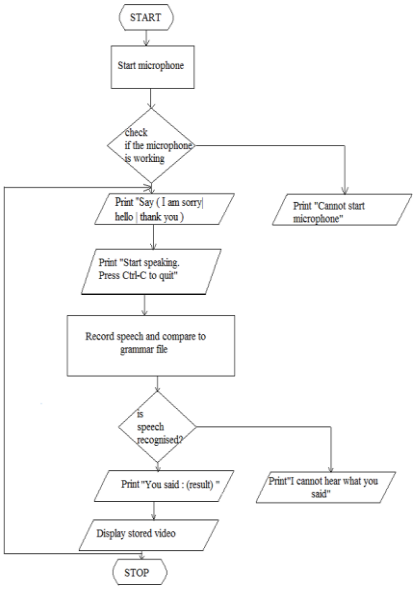

Fig. 3 Program flowchart

The input variable was spoken speech, through a microphone. Output variables were written text (the sequence of words spoken) in the Console window of Eclipse, and a video played from a file location, displaying the sentence spoken in sign language. The program's performance is measured by the effectiveness and accuracy of translation.

VI. SIMULATION OUTPUT

The output obtained by running the Java program using Sphinx-4 in Eclipse, gives the output shown in Figure 4. The figure shows the sentence "I am sorry" in the console window, as an outcome of translation from speech to text. Figure 5 shows the output video as well, which is of a person saying the same sentence in sign language.

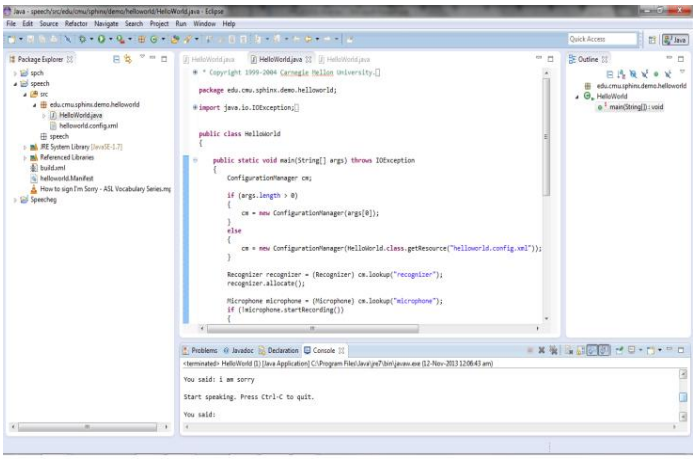

Fig 4. Simulated output, showing speech translated to text.

VII. CONCLUSIONS AND FUTURE SCOPE

This system is a prototype of the main system, which can translate any spoken language into sign language. For this, an animation module will have to be developed, and language models must be trained, which would be projects on their own. This system can be used in schools, hospitals and public meetings, in the place of an interpreter, to convey message to the deaf. It can also be developed for different regional languages, in India, to solve the problem of communication with the deaf even for small regional language groups.

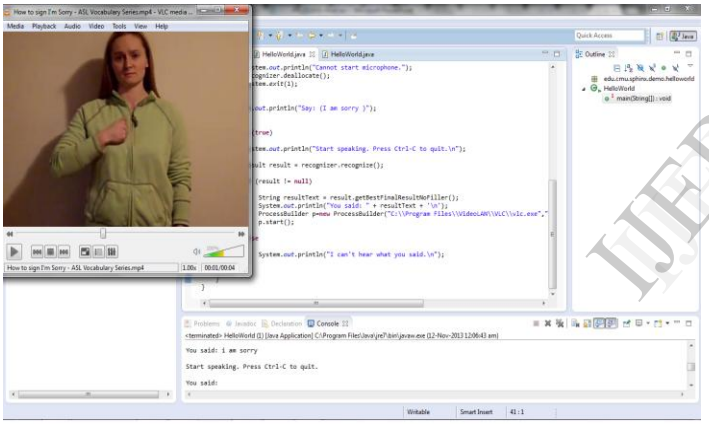

Fig 5. Simulated output, showing speech translated to text and sign language

REFERENCES

[1] Conroy, P. (2006). Signing in and Signing out: "The education and employment experiences of Deaf adults in Ireland. Dublin: Irish Deaf Society. Research Report."

[2] Wheatley, M., & Pabsch, A. (2010). Sign Language in Europe: "Corpora and sign language technologies." . In 4th workshop on the representation and processing of sign languages.

[3] San-Segundo, R., Pardo, J. M., Ferreiros, F., Sama, V., Barra-Chicote, R., Lucas, J. M., et al. (2010). "Spoken Spanish generation from sign language". Interacting with Computers, 22(2), 123–139.

[4] S. J. Young, N. H. Russell, and J. H. S. Russell (1989), "Token passing: A simple conceptual model for connected speech recognition systems," Cambridge University Engineering Dept, UK, Tech. Rep. CUED/F-INFENG/TR38.

[5] X. X. Li, Y. Zhao, X. Pi, L. H. Liang, and A. V. Nefian (Sept. 2002), "Audio-visual continuous speech recognition using a coupled hidden Markov model," in Proceedings of the 7th International Conference on Spoken Language Processing, Denver, CO, pp. 213–216.