Open Ended System for Speech Data Handler: Hexapod

H K Anasuya Devi
Professor & Faculty, CCE, IISc
Bangalore

Prathap Ramesh
CCE Student, IISc
Bangalore

Aravind Krishnan
CCE Student, IISc
Bangalore

ABSTRACT
In the era of information processing and use of information technology, speech recognition and processing plays a very important role for many human computing tasks. Speech pattern and Speech recognition is fundamental for speech analysis in speech data handling system. No doubt it allows all users to communicate with computers and along with electronic devices using any natural language. Today the potential power of the PCs enables the implementation of the novel method proposed in this paper. The prime concern is that none of the software is freely available, user friendly and cross platform. Therefore it necessitates using open-source programming language to an Open Ended System for Speech Data Handler using self created Hexapod: a six legged robot. This system facilitates the way any electronic device with an UART port or USB port can be controlled using speech commands.

Keywords
speech pattern, speech data processing, speech recognizer, speech processing, hexapod robot, sphinx-4, microcontrollers

1. INTRODUCTION
"Speech is the most natural way of human communication". Thus Speech data handling becomes necessary. According to B Yagnarayana in his inaugural address [1] at ICCVSP- 2013 proclaims "The basis in speech recognition lies in the speech production system itself. There is magic about speech. What is speech? Speech constitutes the signals and systems. It is in the relationship and the characteristics of the system to identify the signal. Signal processing helps machines to do the task. How do we understand? It is in ASCII, bits, bytes, pixels, sequences, acoustics/audio/video. The speech production mechanism lies in the twisting of the tongue, rotation etc. It is the medium vs message." This paper considers some of these aspects highlighted in the context of speech data handler for open ended systems using the Hexapod:

- Speech is faster:
  - 3-4 times than typewriting
  - 8-10 times than handwriting
- The specially-abled people can interact with computers and electronic appliances with ease [ex: Blind, Paralysis]
- Simple Interface: without any special training, user can interact with the system.
- Interaction and speech data handling is possible with this application.
- The existing method of using the electronic devices today can be relooked by using this application.

The aims and objectives of this paper highlight the following:

- Open ended Speech Data handling application
- Sphinx-4 speech engine to analyze audio input speech commands
- Software application layer to interface with serial port / usb port based robots/ devices
- Provide open source, freely available application to control and interact with electronic devices

2. RELATED WORK
Here we discuss some of the available software for speech recognition in computers.
Tazti Speech Recognition Software [2] is commercially-available software with options like key binding for PC games, voice search of many search engines, custom speech commands for easy internet navigation and much more. But, it is not free, Open Source, or available on multiple platforms (currently it works only in Windows).
The Microsoft Speech SDK 5.1 [3] is a mature library that is freely available. It is only available on Windows and is not an Open Source.
The Carnegie Mellon University [CMU] Sphinx project [4] has developed several free, Open Source libraries like Sphinx-2, Sphinx-3 and Sphinx-4 which can handle speech recognition. Sphinx-4 is the most advanced version, which is mainly a research platform. As it is written totally in Java, it is portable to different platforms.
Currently most of the available software in connection with the related work of this paper has limitations such as a few are not Open Source, some run only on particular operating systems and some have no option for customization. Accordingly, this paper addresses all the problems mentioned above by providing a new application.
The following list has the requirements for this new application and the tools used to develop it:

- Development Environment –Processing [5] is a free, Open Source programming language and development environment.
- Speech recognition – CMU Sphinx-4 was used to handle recognition.
- Cross-platform – Processing runs on Windows [6], Mac OS [7] & Linux [8]. Sphinx-4 is written in Java, making them easily portable to platforms with Java virtual machine implementations. The program itself is mostly written in Java with bindings for C++.
- Minimal and Simple User Interface [UI] – Only essential functions which are simple and easy to customize are exposed to the end users.
- Multiple programming language support – Java and C++ are supported.

3. SPEECH DATA HANDLING SYSTEM FOR HEXAPOD

Speech recognition basically means talking to a computer, having it recognize what we are saying, and lastly, doing this in real time. This process fundamentally functions as a pipeline that converts PCM (Pulse Code Modulation) digital audio from a sound card into recognized speech. The block diagram as shown in Figure-1 is self explanatory.

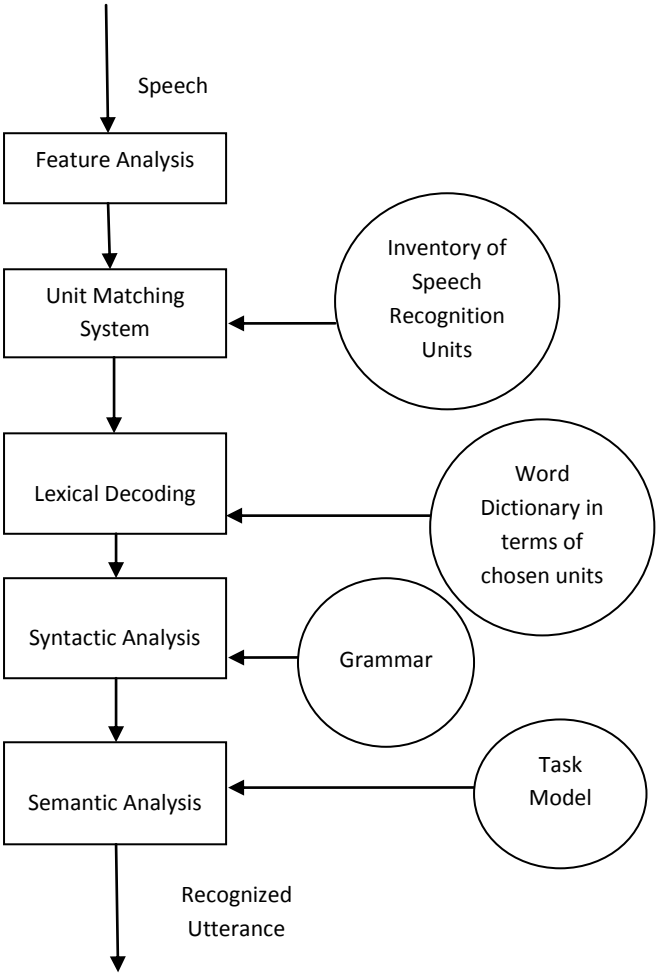

Fig. 1: Block Diagram of a Speech Handling System

There are many speech recognition software available. Various dictation software have been developed by Dragon [9], IBM and Philips [10]. Genie [11] is an interactive speech recognition software developed by Microsoft. Various voice navigation applications, one developed by AT&T, allow users to control their computer by voice, like browsing the Internet by voice. Many more applications of this kind are appearing every day.

The SPHINX speech recognizer of CMU [12] provides the acoustic as well as the language models used for recognition. It is based on the Hidden Markov Models (HMM). The SONIC recognizer [13] is also one of them, developed by the University of Colorado. There are other recognizers such as XVoice [14] for Linux that take input from IBM's ViaVoice which, now, exists just for Windows. Background noise is the worst part of a speech recognition process. It confuses the recognizer and makes it unable to hear what it is supposed to. One such recognizer has been devised for robots that, despite of the inevitable motor noises, makes it communicate with the people efficiently. This is made possible by using a noise-type-dependent acoustic model corresponding to a performing motion of robot. Optimizations for speech recognition on a HP SmartBadge IV embedded system [15] has been proposed to reduce the energy consumption while still maintaining the quality of the application. Another such scalable system has been proposed in [16] for DSR (Distributed Speech recognition) by combining it with scalable compression and hence reducing the computational load as well as the bandwidth requirement on the server.

Various capabilities of current speech recognizers in the field of telecommunications are described in [17][18][19] like Voice Banking and Directory Assistance that are also primarily based on the Hidden Markov Models [HMM]. However, the following sections provide solutions to some of the problems mentioned above.

3.1 Implementation

The Speech Data Handler [SDH] for Hexapod architecture is described as shown in Figure 1.

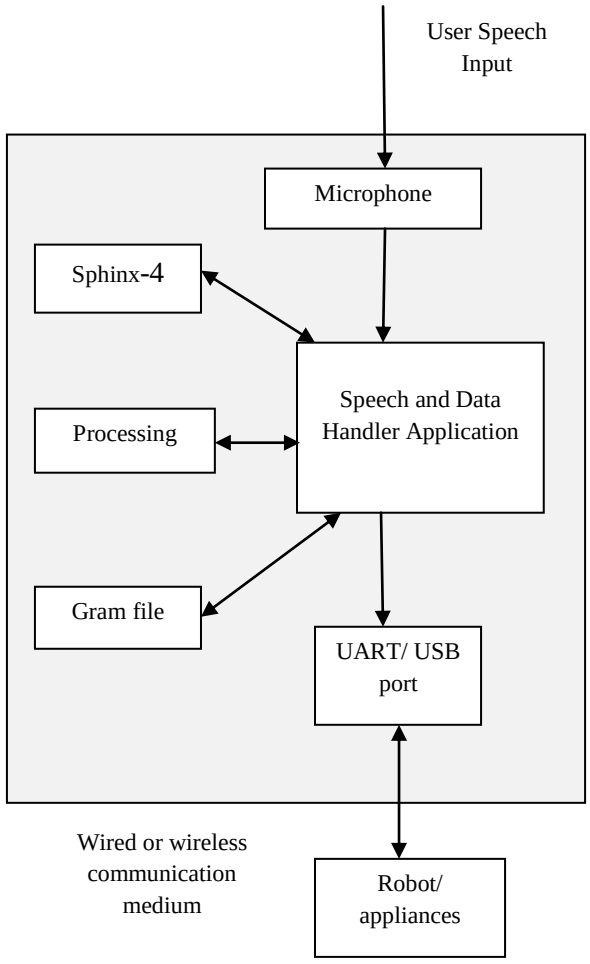

Fig. 2: SDH App Architecture

For speech recognition, SDH application allocates a Sphinx-4 microphone which continuously listens to incoming audio data from the user's audio hardware. This audio input is sampled internally in the Sphinx-4 engine itself at a frequency either at 8KHz (narrow band) or 16KHz (wide band) depending on the quality of the output required. It is in the in-built Sphinx-4 engine the features are embedded for entire

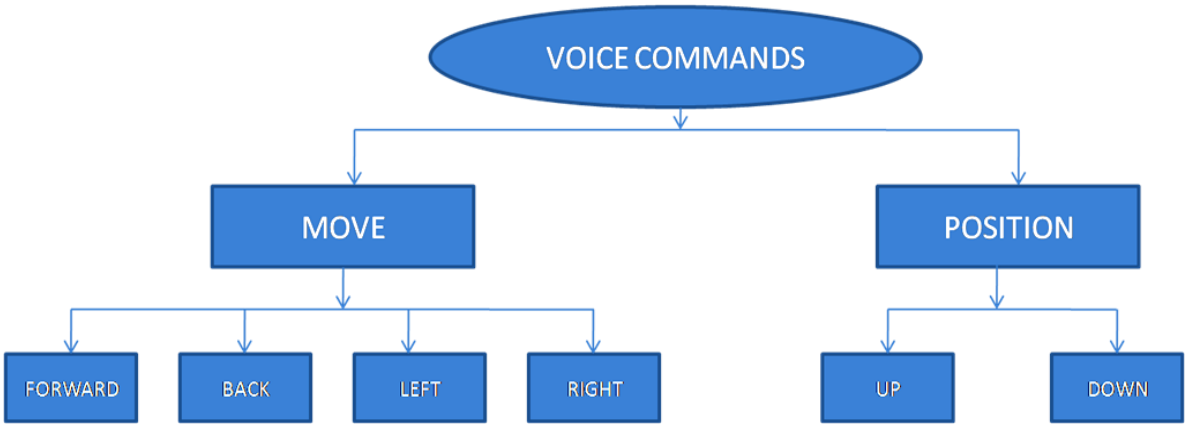

Fig. 3: Categorization of Voice Commands

speech handling system. Some of the features considered by Sphinx-4 are classification of phonemes into vowels, consonants, voiced, unvoiced, oral and nasal etc. The Sphinx-4 recognizer processes the data continuously, and adds the recognized strings and words to an internal queue which can be accessed by the application. Once the user stops speaking, the SDH application checks for an assigned reaction in its database. If the speech sample is valid, the SDH application sends a data-byte to an electronic device which is connected to the UART or USB port of computer to perform the assigned task. Most of the complex recognition technology underneath is hidden from application users which allow them to treat this interaction like a simple input/output device. The Sphinx-4 also includes low level audio support through the Java Sound API, which also provides a cross- platform way to deal with audio hardware.

3.2 Grammar Files

Grammar files define what words should be recognized during speech recognition. Although the list of possible words is huge, most applications will only use a small set of words defined by the grammar files. This file can be edited by the user according to the requirement. They must conform to the Java Speech Grammar File (JSGF) format. The following is an extremely simple JSGF grammar:

grammar animal;

public <animalTypes> = (lion | tiger | panther);

This file is named "animal." The name can be specified when initializing SDH App to inform it which file to use. The grammar rule "animalTypes" is satisfied if any one of the items in the list is recognized. When this rule is satisfied, the spoken words that satisfied the rule are put into a single string and added to the SDH App's internal queue. The "or" ("|") operator here specifies that only one of the members of the list is required to satisfy the rule.

This example involves more complex grammar:
grammar hexapod;

public <command> = (move | halt | <rotate>);

<rotate> = rotate <side>;

<side> = (left | right);

Here, a set of simple commands are being used to control a Hexapod robot. The "command" grammar rule is satisfied if the words "move" or "halt" are spoken or if the "rotate" rule is satisfied. The "rotate" rule is satisfied if the word "rotate" followed by "left" or "right" is spoken.

3.3 Post-processing Recognized Text

Once the audio sample has been recognized and verified in the grammar file, the SDH can perform the following tasks:

- Play a music file stored in the hard drive
- Copy or move files to an USB flash drive
- Access the UART or USB port to which a robot or electronic device is connected and send a data byte to it.

Advanced users can actually do additional analysis in the program, such as natural language processing, on the recognized strings which is not present in this version of the SDH App. The following algorithm demonstrates common ways to process the recognized strings.

A simple method is to check whether the recognized strings contain certain keywords. For example, to control a hexapod robot the SDH App could check if the voice command given by the user exists in the grammar file. Here the list has commands like "left", "right", "forward" and "back", "up" and "down", it will need to search the recognized strings for all of these words.

1. Is there any recognized string available?
2. If yes, then store the recognized string in a buffer.
3. Initialize a serial variable for serial communication over UART port
4. Check if the recognized input is one of the following:
   forward, right, left, back (basic commands)
5. If the recognized input is "forward" then write a character 'f' to
   the serial variable to send it over the UART.
6. The Hexapod's microcontroller receives the character
   written to the serial variable and after processing it, performs the appropriate action.
7. Similarly, if the recognized input is "right" then the character written to the serial port is 'r', for "left" it is 'l', for "back" it is 'b'.
8. If the input is not recognized properly and doesn't match any of the pre-set words then it goes to Step 1 again for a fresh input.

This method works alright for a few trial-sets of commands, however, more set of examples are being tested which needs more processing time. Further groups of commands are also being considered for the purpose. In the above example if we categorize the words as shown in Fig. 3, the processing time can be reduced.

The program implementation is as follows:

```
while (app::RecognizedString() > 0)
{
s1::string comm = app::popRecognizedString();

     // Check if the string contains 'move'.
if (s1::string::npos != comm.rfind("move"))
{
      //Check for all the commands in this category
     if (s1::string::npos != comm.rfind("forward"))
     {
        //Send a byte to serial port
 myserialport.write('f');
     }
else if (s1::string::npos != comm.rfind("left"))
     {
        //Send a byte to serial port
 myserialport.write('l');
     }
else if (s1::string::npos != comm.rfind("right"))
     {
        //Send a byte to serial port
 myserialport.write('r');
     }

else if (s1::string::npos != comm.rfind("back"))
     {
        //Send a byte to serial port
 myserialport.write('b');
     }
}

     // Check if the string contains 'position'.
if (s1::string::npos != comm.rfind("position"))

{
 if (s1::string::npos != comm.rfind("up"))
     {
        //Send a byte to serial port
 myserialport.write('u');
     }

else if (s1::string::npos != comm.rfind("down"))
     {
        //Send a byte to serial port
 myserialport.write('d');

     }

 }
}
```

Users of this example activate the robot by saying; for example, "move forward," instead of simply telling "forward." This categorical organization can result in more natural commands if implemented carefully. Also in this method, the SDH parses speech commands using fewer string comparisons. Additionally, a number of synonyms for the existing speech commands can also be programmed to be recognized. For instance: Words like "lift up", "bend down", "sit", "stand", "rise", "turn", "turn around", "bye", "go", "dance" and many more can be programmed to be recognized and appropriate action to be performed by the Hexapod.
Any additional commands to work should result in improving the present design of the Hexapod itself.

3.4 Problems Encountered

Two significant problems arose during the development:

1) Speech recognition takes a significant amount of time to produce its first output when running the SDH Application, and
2) The recognition is not highly noise resistant. It recognizes few words for random loud noise.

The first problem is dependent on the processor speed and RAM size of the computer. Increasing the computer specifications reduced the delay time for output. Using an external microphone in the computer drastically reduces the false detections due to noise in the testing premises. No other serious problems are encountered. The Sphinx-4 and Processing libraries are solid, well-tested pieces of software that works as described.

4 TEST APPLICATIONS

This section describes the Hexapod robot, developed in house to test SDH App, giving subjective results on how well the SDH App works in real time.

Hexapod robot[20] is a mechanical vehicle that walks on six legs. The robot can move using various gaits on different terrains. The Hexapod robot is controlled by an ATMega32 microcontroller [Figure 3]. The ATMega32 microcontroller has an UART port which is connected to the target computers' USB port. The SDH App is configured with the grammar file which contains commands such as "move", "forward", "down" and "up". When the particular command is recognized by the SDH App, it sends a unique data byte to the UART port. The Hexapod is programmed to move when it receives a particular data byte from the UART port. So according to the voice command, the Hexapod robot moves

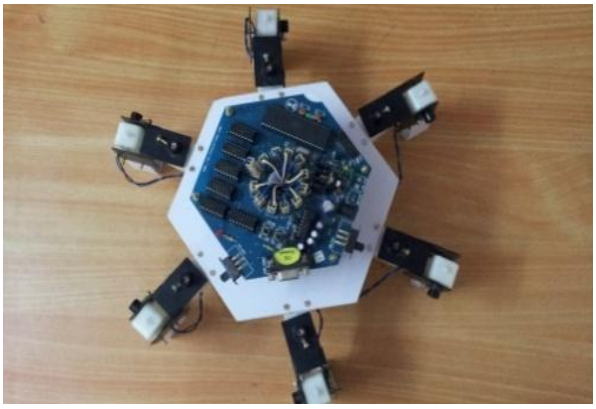

Fig. 4: Hexapod Robot

using a basic pentapod gait. The technical specifications are in Appendix A.

5 CONCLUSION

The SDH application is an Open Source, cross- platform program for speech recognition for robotics to perform some specific actions in a context dependent environment with a simple user interface and program that can be used in Java and C++. This is the simplest of its kind designed for the purpose and this work enables to improve the system for the larger context to include wider sets in the performance using Natural Language. Furthermore this SDH system can be applied to Satellite Data analysis and Medical Diagnosis using robots.

The context in which the open source programming has been used necessitates the use of elementary program, because the Hexapod has to understand these basic commands at the beginning. Otherwise it will not perform action or commands given by the user.

The grammatical rules have to be framed for the purpose of creating a knowledge base in the system for the Hexapod to understand and recognize the voice commands. In this context these rules are being designed for the purpose of recognizing various commands which will be taken up for the future work. Only after implementing the present program the new ideas can be introduced and tackled for the purpose of voice recognition.

Despite its simple user interface, the technology underneath is very powerful, thanks to the hard work by the Processing and Sphinx-4 development team.

Appendix A

Hexapod Robot Specifications:

- AVR ATmega32 microcontroller by ATMEL
- L298 Motor drivers
- DC motors

6 FUTURE WORK

One of the main goals is to keep the SDH application simple and easily accessible. The following is the list of features that can be added to the SDH application in the future versions.

- Natural language processing – to increase the performance and stability for longer sentences.
- Multiple programming language support.
- Additional gesture based commands to supplement speech commands.

REFERENCES

[1] International Conference on Communication, VLSI and Signal Processing ICCVSP – 2013. (In his inaugural address)

[2] Tatzi Speech Recognition Software

[3] Microsoft Speech SDK 5.1

[4] CMU Sphinx

[5] Processing

[6] Windows OS

[7] Mac OS

[8] Linux OS

[9] Dragonsys

[10] Speech Recognition System

[11] Microsoft Development Agent

[12] Kai-Fu Lee, Hsiao-Wuen Hon, and Raj Reddy, an Overview of the SPHINX Speech Recognition System. IEEE Transactions on Acoustics, Speech and Signal Processing,

[13] Pellom, B., Sonic: The University of Colorado Continuous Speech Recognition System.

[14] http://www.zachary.com/s/xvoice

[15] Yoshitaka Nishimura, Mikio Nakano, Kazuhiro Nakadai, Speech Recognition for a Robot under its Motor Noises by Selective Application of Missing Feature Theory and MLLR.

[16] Brian Delaney, Tajana Simunic, Nikil Jayant, Energy Aware Distributed Speech Recognition for Wireless Mobile Devices.

[17] Naveen Srinivasamurthy, Antonio Ortega, Shrikanth Narayanan, Efficient Scalable Speech Compression for Scalable Speech Recognition.

[18] Lawrence R. Rabiner, Applications of Speech Recognition in the Area of Telecommunications.

[19] Lawrence R. Rabiner, A Tutorial on Hidden Markov Models and Selected Applicationsin Speech Recognition.

[20] Hexapod robot

[21] Li Deng, Xiao Li, May 2003, Machine Learning Paradigms for Speech Recognition: An Overview

[22] Dileep A.D, Sekhar C.C, Dec 2013, HMM Based Intermediate Matching Kernel for Classification of Sequential Patterns of Speech Using Support Vector Machines

[23] Jose Enrique Garcia, Alfonso Ortega, Antonio Miguel, Eduardo Lleida, Low bit rate compression methods of feature vectors for distributed recognition.

[24] Naoya Wada, Noboru Hayasaka, Shingo Yoshizawa, Yoshikazu Miyanaga, "Direct Control on Modulation Spectrum for Noise-Robust Speech Recognition and Spectral Subtraction," IEEE International Symposium on Circuits and Systems (ISCAS), pp. 2533-2536, May 2006.

[25] Jirabhorn Chaiwongsai, Werapon Chiracharit, Kosin Chamnongthai, Yoshikazu Miyanaga, "An Architecture of HMM-Based Isolated-Word Speech Recognition with Tone Detection Function", Proceedings of 2008 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS), December 2008.

[26] HTK book

[27] Java Speech Grammar Format(JSGF)