Low Cost Home Automation Using Offline Speech Recognition

Prasanna G. and Ramadass N.
College of Engineering Guindy, Anna University/Department of Electronics and Communication Engineering, Chennai, India
Email: mrgprasanna@gmail.com, ramadassn@annauniv.edu

***Abstract*—Sound intelligence is added to a home automation based on acoustics for sophistication of physically challenged people as a broad perspective of the thesis. Home automation already in practice is by switching on or off a device via wired networks. But this is inefficient for people with impairment in mobility or spinal cord disability due to ageing factors. The easiest way of controlling a device is through human voice. Existing products are expensive and also speech recognition is available with usage of internet (online). This paper is presented with the concept of speech recognition being implemented using Hidden Markov Model Toolkit (HTK) in an offline manner (without internet). Human voice is converted to text using HTK and it is wirelessly Trans received using GSM modems. According to the received texts appliances can be controlled. The module contains a secured speech recognizer for automatic door opening/closing and a general voice recognizer to control appliances like television, music player, fan, light, etc. All the above are implemented in a low cost Raspberry Pi board. Thus a goal of producing an automation device has been designed at low cost using offline speech recognition [1].**

***Index Terms*—HTK, SPHINX, RASPBERRY PI term**

I. Introduction

With the advancements in Information Technology, the next generation of user interface is desired to be more user-friendly and powerful. As the choice for natural and expressive means of communication, speech is more desirable for the human-computer interaction. Furthermore, considering desktop PCs to mobile phones, toys and other embedded devices, the user interface becomes smaller in size which limits its operation. Speech has the potential to provide a direct and flexible interaction for the embedded system operations [2].

Generally speaker independent systems are more widely used, since the user voice training is not required. Speech recognition is classified as connected word recognition and isolated word recognition. For embedded devices, implementation of isolated word recognition is sufficient. Generally, speech recognition is a kind of pattern recognition based on training and recognition [3]. Speech signal can be modeled as a time sequence and is characterized by Hidden Markovian Model (HMM). The input of the system characterized by human voice signal, detected by a USB microphone connected to the system. Then their Mel Frequency Cepstral Coefficients (MFCC) vectors are obtained by a powerful HTK toolkit. After passing through sequence of steps word gets recognized. The recognized word is converted to text and we will get the output in terminal window. Using gpio pins of Raspberry Pi these texts are transmitted wirelessly using a GSM modem. One more GSM modem at the receiver end will receive those words and according to that automation can be done [4].

This paper deals with the basic steps in hidden markovian model. Then it deals with Sphinx. Sphinx is an open source platform that incorporates state-of-the art methodologies and also addresses the needs of emerging research areas, provided our technical aims and targets as well as our diversity. Sphinx implemented in the Java programming language, which is made available to a large variety of development platforms.

First and foremost, Sphinx is a modular and pluggable framework that incorporates design patterns from pre-existing systems, with sufficient flexibility enough to support emerging areas of research and development interest. The framework is modular in which it comprises components which are separable dedicated to specific tasks, and it is pluggable in those modules which can be easily replaced during runtime. To exercise the framework, to provide researchers with a working system, "Sphinx" also consists of a variety of modules that implement state-of-the-art speech recognition techniques [5].

Finally deals with the proposed hardware model, their features, and implementation steps involved in it. Cost analysis also been provided at the end of this paper.

II. Fundamentals of HTK

HTK is a toolkit for building Hidden Markov Models (HMMs). HMMs can be used to model any time series and the core of HTK is similarly general-purpose. However, HTK is primarily and mainly designed for building HMM-based speech processing tools, in particular recognizers. Thus, much of the infrastructure support in HTK is dedicated to this task. As shown in the Fig. 1, there are two major processing stages involved [6]. Firstly, the HTK training tools are used to estimate the parameters of a set of HMMs using training utterances

Manuscript received June 20, 2013; revised June 10, 2014.

doi: 10.12720/ijsps.2.2.96-101

and their associated transcriptions. Secondly, the HTK recognition tools transcripts the unknown utterances.

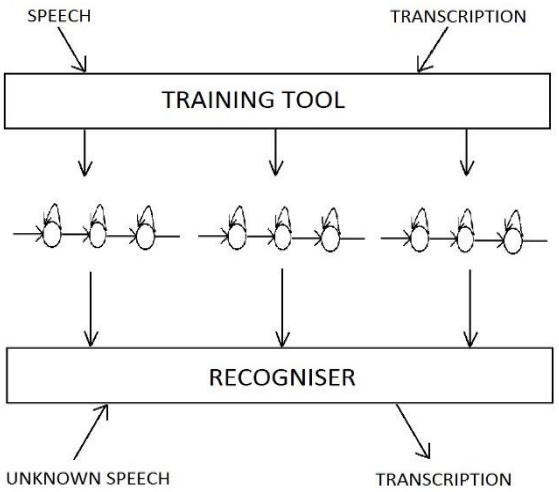

Figure 1. Layout of HTK

A. General Principles of HMMs

Speech recognition systems generally assume that the speech signal is a realization of some message encoded as a sequence of one or more symbols. To effect the reverse operation of recognizing the underlying symbol sequence provided a spoken utterance, the continuous speech waveform is initially converted to a sequence of equally spaced discrete parameter vectors. This parameter sequence vectors is assumed to form an exact representation of the speech waveform on the basis that for the duration covered by a single vector (typically 10ms or etc.), the speech waveform could be regarded as being stationary. Typical parametric representations in common use are smoothed spectra or linear prediction coefficients plus various other representations derived from these [7].

The role of the recognizer is to effect a mapping between sequences of speech vectors and the wanted underlying symbol sequences. This is made very difficult by two problems. Initially, from symbols to speech mapping is not one-to-one since different underlying symbols can give rise to similar speech soundings. Moreover, there is large range of variations in the realized speech waveform due to speaker variability, atmosphere, etc. Secondly, the boundaries between symbols cannot be identified explicitly from the speech waveform [7]. Therefore, it is impossible for the speech waveform to be treated as a sequence of concatenated static pattern.

B. HTK Speech Recognition

1) Recording and labeling

Words are recorded using arecord command followed by syntax:

arecord –D plughw:0,0 – r 16000 –f s16_LE –t wav –d 11 data $1.wav

Open the recorded voice using wave surfer. Then label the utterances, silences and save it.

2) MFC file names generation

Using mapfilegeneration code creates. mfc extension file names for every samples of our speech. This contains the Mel frequency cepstral coefficients of our speech.

3) Feature extraction

Using HCopy command, it needs configuration files as input. HCopy converts data to a parameterized form. In speech recognition it generates mfc files. This program will copy one or more data files to a designated output file, optionally converting data to a parameterized form. While the source files can be in any supported format, but the output format always in htk. By default the whole of the source file is copied to the target but options exist to copy only a specified segment.

4) Protos-Generation

Each pronunciation can be divided into N-states and mixtures. This step will define those states and mixtures by calculating mean and variances values.

5) Generation of word list states

Defining word list states i.e., assign syllables (states) for each word. For example assign a word say zero as 5. Number indicates the states where zero can be splitted.

6) Re-Estimation

Re-estimation of mean and variances is done using HInit and HRest commands. HInit calculates initial parameters of the model using Viterbi algorithm. Here accuracy is less so one more steps has to be processed. HRest refines the parameters using Baum-Welch Re-estimation Both commands follows an iterative procedure until max iteration has reached or estimation converges. Also both look for max likelihood occurrences. Finally re-estimated HMM files created.

7) Lexicon creation

It contains allwordlist, allwordHMMs, allwordnetwork and allwordsyn. Allwordsyn defines number of words and words which are going to be recognized. Allwordnet forms a finite state network using HParse command. HParse command generates network based on EBNF (Extended Backus Naur Form).

8) Decoding

This is the final recognition step using HVite command. HVite has internal HRec command which performs recognition. It follows a token passing algorithm to find the best match. SLF files which are generated are standard lattice format files. Hvite is a general purpose Viterbi word recognizer. It will match a speech file against a network of HMM's and output a transcription for each. When performing N-best recognition, it can provide a word level lattice containing multiple hypotheses.

III. SPHINX-4 SPEECH RECOGNITION

A. SPHINX-4 Framework

The Sphinx-4 framework has been designed with a high degree of flexibility and modularity. Fig. 2 shows the overall architecture of the system. Sphinx-4 framework consists of three primary modules: the FrontEnd, the Decoder and the Linguist. One or more input signals are taken by the FrontEnd and parameterizes them into a sequence of Features. Any type of standard

language model is translated by the Linguist, along with the information of pronunciation from the Dictionary and the structural information from one or more sets of AcousticModels into a SearchGraph [8].

The SearchManager present in the Decoder uses the Features from the FrontEnd and the SearchGraph from the Linguist to perform the actual decoding, generating Results. Results are nothing but the actual recognized word spoken by the user. Sphinx uses an effective Viterbi search, which is most accurate comprising a large number of users to take part.

The Sphinx-4 system is like most speech recognition systems in that it has huge number parameters which are configurable, such as search beam size used for tuning the system performance. Such parameters are configured by using the Sphinx-4 ConfigurationManager.

Unlike other systems, the ConfigurationManager also gives Sphinx-4 the ability to dynamically load and configure modules at run time, which yields a flexible and pluggable system. For example: Sphinx-4 is configured typically with a FrontEnd that produces Mel-Frequency Cepstral Coefficients (MFCCs) [9]. Using the ConfigurationManager, 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 to recompile the system.

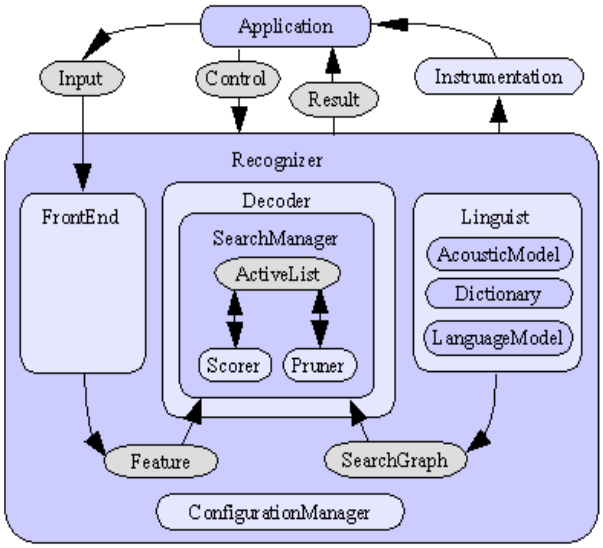

Figure 2. SPHINX-4 decoder framework

B. Dictionary

The Dictionary provides pronunciations for words found in the LanguageModel. The words are broken by pronunciations into sequences of sub-word units found in the AcousticModel. The Dictionary interface also provides supports for the classification of words and allows for a single word to be in multiple classes. Sphinx-4 currently provides implementations of the Dictionary interface to support the CMU Pronouncing Dictionary. The various implementations optimize for usage patterns based on the size of the vocabulary which are active. For example, each implementation will load the entire vocabulary at the time during system initialization, but another implementation will obtain only pronunciations on demand.

SearchGraph

Though Linguists are implemented in different ways and the topologies of the search spaces generated by these Linguists may vary to a great extent, the search spaces are represented as a SearchGraph. Illustrated by example in Fig. 3, the SearchGraph is the primary data structure used during the decoding process.

The graph is a directed graph consisting of each node, called a SearchState represents an emitting or a non-emitting state and in which Emitting states can be scored against incoming acoustic features while non-emitting states represent higher-level linguistic constructs such as words and phonemes that are not directly scored against the features which are incoming. The arcs between the states are representing the possible state transitions, every single one of which has a probability which represents the likelihood of transitioning along the arc.

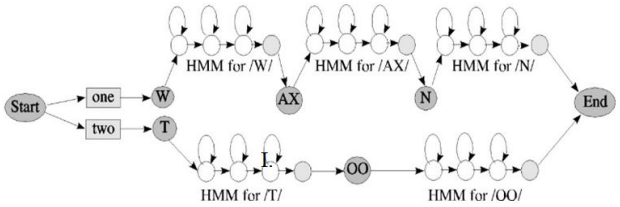

Figure 3. Example SearchGraph

The SearchGraph is a directed graph composed of optionally emitting SearchStates and SearchStateArcs with transition probabilities. Each and every state present in the graph will be able to represent components from the LanguageModel (words in rectangles), Dictionary (sub-word units in dark circles) or AcousticModel (HMMs).

C. Decoder

The primary role of the Sphinx-4 Decoder block is to use Features from the FrontEnd in conjunction with the SearchGraph from the Linguist so that it generates Result hypotheses. The Decoder block comprises SearchManager which is pluggable and other supporting code that simplifies the decoding process for an application. Hence, the most interesting component of the Decoder block is the SearchManager. The Decoder tells the SearchManager only to recognize a set of Feature frames. At every step of the process, the SearchManager creates a Result object that contains all the paths that have reached a final state which is non-emitting. To process the result, Sphinx-4 does also provide utilities capable of producing a lattice and confidence scores from the Result.

Unlike other systems, however, applications can modify the search space and the Result object in between steps, allowing the application so that it becomes a partner in the recognition process. Sphinx-4 provides a sub-framework to support SearchManager composed of an ActiveList, Pruner, and Scorer. The SearchManager sub-framework generates ActiveLists from currently active tokens in the search trellis by pruning using a Pruner which is pluggable. Applications can be used to

configure the Sphinx-4 implementations of the Pruner to perform both relative and absolute beam pruning.

The implementation of the Pruner is greatly simplified by the garbage collector of the Java platform. By the garbage collection, the Pruner can prune a complete path by just removing the terminal token of the path from the ActiveList. This act of removing the terminal token would identify the token and any unshared tokens for that path would be as unused, allowing the garbage collector to re-claim the associated memory.

IV. Implementation Steps in SPHINX-4

A. Explanation of JAVA Code

The Recognizer is the main class any application should be interacting with. The Result is being returned by the Recognizer to the application after recognition completes. The Configuration Manager creates the entire Sphinx-4 system according to the configuration specified by the user. Main method creates the URL of the XML-based configuration file.

The Configuration Manager then internally reads in the file. As the configuration file specifies the components which are "recognizer" and "microphone", a lookup is performed in the Configuration Manager to obtain these components. The allocate method of the Recognizer is then called to allocate the resources needed for the recognizer. The Microphone class is then used for capturing live audio from the system audio device.

Both the Recognizer and the Microphone are configured as specified in the configuration file. The program first turns on the Microphone. After the microphone is turned on, the program enters a loop that is repeating the following: It tries to recognize what the user is saying, using the Recognize method.

Recognition stops when the user stops speaking, which is detected by the end pointer built into the front end by configuration. Once an utterance is recognized, the text which is recognized is returned by the method Result. getBestResult NoFilter, is printed out. If the Recognizer recognized nothing (i.e., result is null), then it will print out a message. If the program cannot turn on the microphone in the first location, the Recognizer will be deallocated and then the program exits. It is usually a good practice to call the method deallocate after the work is done to release all the resources. Recognition Results are shown in Table I below.

TABLE I. Recognition Results

Percentage of words correct	93.55%
Word accuracy percentage	88.76%
Percentage of sentences	42.56%

V. Proposed Hardware Model

A. Raspberry Pi Features

Raspberry Pi has 256 MB of RAM and 700MHz ARM-11 processor. The chip's main purpose is to power a cheap computer with a basic level of functionality.

The Model B has two USB ports, HDMI out and a 10/100 Ethernet port as shown in the Fig. 4. It has 3.5mm audio jack and the HDMI output, which even supports audio transmission. The Raspberry Pi's GPU will be boasting 1 Gpixel/s, 1.5 Gtexel/s or 24 GFLOPs of general purpose compute power and is OpenGL 2.0. Overall hardware model is shown in Fig. 5.

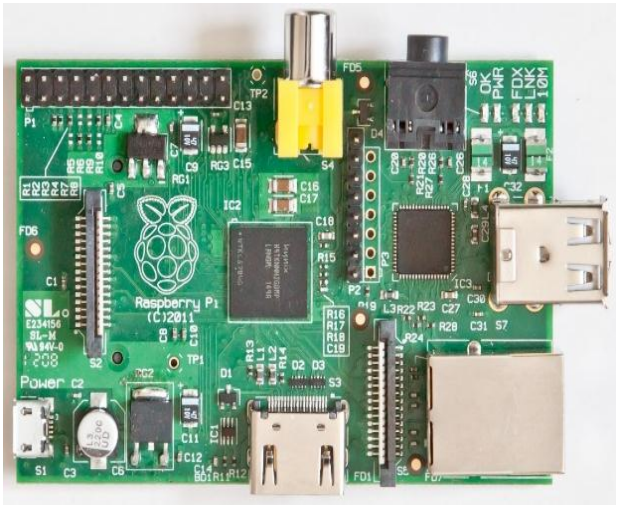

Figure 4. Raspberry Pi board

Figure 5. Proposed hardware model

B. GSM Modem

Modem used here is SIM300. SIM300 is a dual band GSM/GPRS engine. It works on frequencies EGSM 900MHz and DCS 1800MHz. Has plug and play feature. Supports baud rate varies from 1200 to 115200 bps. Converted text is send as SMS from GSM transmitter and a GSM receiver will receive that text accordingly automation is done.

C. Installing the Raspberry Pi Fedora Remix (Linux OS)

Installing Fedora Remix on Raspberry Pi consists of 4 steps:

- Copy the ready-made Fedora Remix image onto an 4GB (or larger, use an 8GB card) SD memory card
- Hook up a monitor (preferably through HDMI), a USB keyboard and a mouse (preferably that share a single USB port) and optionally an internet connection (Ethernet cable, wireless USB card or USB connecting to a smartphone)
- Power on and walk through the Fedora "first boot" wizard
- Reboot, login and geek out

The SD card serves as the Raspberry Pi's primary drive and by default the Pi will look on the SD card for the operating system to load (though it's possible to boot from other devices also). Hence, the first step is to grab the image and put it on the card using dd write command. Then install HTK tools on Raspberry Pi. Copy the script file and paste it to Pi. Now the script can be run from terminal as shown in Fig. 6.

```
[root@raspi htkfinal]# ./scripts/trial.scr
+arecord –D plughw:1,0 –r 16000 –f S16_LE –t wav –d 2 temp/test.wav
RECORDING WAVE ' temp/test.wav' | Signed 16bit Little Endian, Rate 16000 Hz, Mono
+HCopy –C config_files/config_speech temp/test.wav temp/test.mfo
+echo temp/test.mfc
+HVite –T 1 –C config_files/config_feature –p -40 –S temp/test_file –H –lexicon/allwordHmms
+out – d = -f1
+grep == temp/output
+awk "(print $2 "\t" $4)" temp/res
motor on
++awk "(print $2)" temp/res
+VAR1 = motor
++awk "(print $4)" temp/res
+VAR2 = on
+gpio –g node 16 out
+gpio –g node 17 out
+gpio –g node 4 out
+gpio –g node 22 out
+gpio –g write 18 1
+gpio –g write 17 1
+gpio –g write 4 1
+gpio –g write 22 1
+case "$VAR1" in
+case "$VAR1" in
+case "$VAR2" in
+gpio –g write 4 0
[root@raspi htkfinal]#
```

Figure 6. Sample output

D. Wireless Transmission of Text

Raspberry Pi has Tx and Rx pins using that serial communication can be done. GSM modem at the transmitter side sends the recognized word as SMS as shown in Fig. 7(a). Another GSM modem at the receiver end receives the Text and appliances can be controlled using a microcontroller at the receiver end as shown in Fig. 7(b). To access raspberry pi GPIO pins, we have variety of languages can be used like Python, Perl, and BASH etc. Here we are using BASH shell scripts to access GPIO pins.

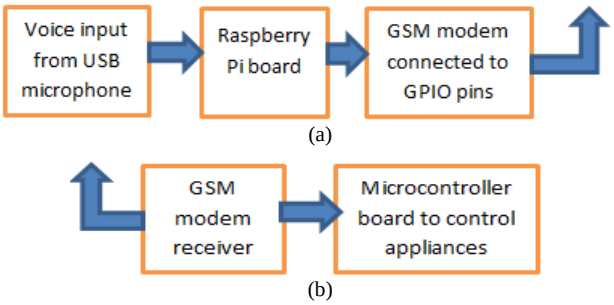

Figure 7. Block diagram (a) transmitter and (b) receiver section

TABLE II. COST ANALYSIS OF TOTAL HARDWARE MODEL

S No	Component	Cost
1	Raspberry Pi Board	$25
2	GSM Modem - SIM 300	$12
3	Microcontroller board	$4
Total Cost		**$41**

VI. CONCLUSION

In this paper, a speech based home automation is done at a low cost as shown in Table II. The system contributes towards empowering the aged and impaired to meet their needs. Accuracy can be improved by incorporating various filters. Thus an offline low cost speech recognition hardware model has been implemented using Raspberry Pi board.

REFERENCES

[1] Istvan I. Papp, "Hands free voice communication with TV," *IEEE Trans. on Consumer Electronics*, vol. 57, no. 2, pp. 606-614, May 2011.
[2] M. R. Alam, "A review of smart homes – Past, present and future," *IEEE Trans. on Systems, Man and Cybernetics*, vol. 42, no. 2, pp. 1190-1203, Nov. 2012.
[3] Q. Y. Hong, C. H. Zhang, X. Y. Chen, and Y. Chen, "Embedded speech recognition system for intelligent robot," in *Proc. IEEE Conf. on Mechatronics and Machine Vision in Practice*, Dec. 2007, pp. 35-38.
[4] Junichi Yamagishi, "Thousands of voices for HMM based speech synthesis-analysis and application of TTS systems built on various ASR corpora," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 5, pp. 984-1004, Jul. 2010.
[5] K. Kumar and R. K. Aggarwal, "Hindi speech recognition system using HTK," *International Journal of Computing and Business Research*, vol. 2, no. 2, May 2011.
[6] V. Tiwari, "MFCC and its applications in speaker recognition," *International Journal on Emerging Technologies*, vol. 1, no. 1, pp. 19-22, 2010.
[7] T. Hain, P. C. Woodland, T. R. Nielser, and E. W. D. Whittaker, "The 1998 HTK System for Transcription of Conversational Telephone speech" in *Proc. IEEE Conf. on Acoustics, Speech and Signal Processing*, Mar 1999, pp. 57-60.
[8] J. G. Wilpon, "Automatic recognition of keywords in unconstrained speech using hidden markov model," *IEEE Transactions on Acoustic, Speech and Signal Processing*, vol. 38, no. 11, pp. 1870-1878, Nov. 1990.
[9] A. G. Veeravalli, W. D. Pan, R. Adhami, and P. G. Cox, "A tutorial on using hidden markov models for phoneme recognition," in *Proc. IEEE SSST'05*, Mar. 2005, pp. 154-157.

Prasanna Gobinathan born in Madurai, a city in Tamil Nadu, India on Dec 29 1989 graduated with M.E. Applied Electronics from the College of Enginnering Guindy, Chennai, Tamil Nadu, India earned the degree in the year 2013 and completed B.E. Electronics and Communication, from the college Sri Sivasubramania Nadar College of Engineering, Chennai, Tamil Nadu, earned the degree in the year 2011.
He is working currently in Wheels India Ltd Chennai, in the designation of SENIOR ENGINEER (R&D Electronics) since Aug 2013 till date. Pursued an internship in the field of Radar Image Processing for a period of 4 months in ISRO (Indian Space Research Organization), Bangalore. Involved currently in a research in the field of embedded systems.

Ramadass Narayanadass was born in Chennai, Tamilnadu, India in 1975. He received the B.E. degree in Electrical and Electronics Engineering from Madras University in 1997, M.E. degree in Applied Electronics and Ph.D. from Anna University, Chennai, India in 2001 and 2008, respectively.

He has been associated with the Faculty of Information and Communication Engineering, Anna University, Chennai, since 2001. He is currently an Associate Professor in Department of Electronics and Communication Engineering at Anna University, Chennai.

His research interests include Embedded Systems, VLSI design and Reconfigurable Computing.