Speech recognition for Indonesian language and its application to home automation

Zulkarnaen Hatala
Electrical Engineering Department
Politeknik Negeri Ambon
Ambon, Indonesia
dzulqarnaenhatala@gmail.com

Abstract— **The practical aspects of developing an Automatic Speech Recognition System with Hidden Markov Toolkit are reviewed. Steps are explained concerning hardware, software, libraries, applications and computer programs used. The common procedure to rapidly apply the speech recognition system is summarized. The procedure is illustrated, to implement a speech-based electrical switch in home automation for the Indonesian language. The main key of the procedure is to match the environment for training and testing using the training data recorded from the testing program, HVite. Often the silence detector of HVite is wrongly triggered by noises because the microphone is too sensitive. This problem is mitigated by simply scaling down the volume. In this sub-word phone-based speech recognition, noise is included in the training database and labeled particularly. Illustration of the procedure is applied to a home automation application. Electrical switches are controlled by Indonesian speech recognizer. The results show a 100% command completion rate.**

Keywords—Indonesian speech recognizer; hidden Markov toolkit; home automation

I. Introduction

An automatic voice recognition system or automatic speech recognition system (ASR) aims to translate the digital signals of the human voice into text or written forms that are meaningful according to certain grammar. The human voice signal is captured by the microphone and processed by the computer to get a regular text. Thus ASR can be used for further purposes such as giving a command to a computer to do a group of tasks by just talking. For example, someone at his home instructs the computer to turn off and turn on the bathroom lights, garden lights without having to press the electrical switch, enough to give a voice command. Another case of applying ASR is when someone is driving a vehicle and wants to instruct his smartphone to notify the current position, or destination route information or even information about the on-off condition of all the electrical switches at his home.

Based on the number of words supported, ASR is categorized into small size vocabulary ASR (SVASR) and large vocabulary ASR (LVASR). SVASR only supports very few words compared to LVASR. An example of SVASR is turning on-off electrical switches at home. Another example would be digit recognition applications or instruction of a phonebook contact to dial automatic calls. In contrast, LVASR is used as a keyboard replacement for typing by naturally speaking naturally on search engines for topics and unlimited words. Even LVASR can be used as an automatic translator machine between two different languages such as between English to Indonesian and vice versa. Here the speaker with an LVASR machine (desktop or smartphone) does not need to understand the language of the interlocutor because it will be translated by the device [1].

But LVASR requires relatively high-quality hardware with a powerful processor and large memory. This makes LVASR difficult to apply directly to limited hardware environments such as embedded devices, smartphones, and home appliance peripherals. This is where the SVASR has the opportunity to be developed. A small number of words certainly do not require high hardware prerequisites. SVASR can also be developed quickly with high accuracy. Some of these things make the SVASR get a significant place to translate simple voice commands that are useful for everyday life. LVASR often is not practical when applied to some daily life cases.

Many reports on ASR research are based on a speech corpus that recorded on the soundproof noise-free environment that results in crystal clear speech databases[2][3]. The acoustic modeled achieved from these speech databases significantly suffer low performance when applied directly to the noisy environment[4]. Unfortunately, most applications are deployed in environments that are contaminated by noise and disturbance. To the best of our knowledge, no noisy ASR is applied for Indonesian especially for small vocabulary let alone implementation of ASR for home automation.

Hidden Markov Toolkit (HTK) [1] is a group of programs and libraries used to develop ASR with the Hidden Markov model. Actually, HTK itself is complete to build the whole ASR starting from speech database, feature extraction, construction and training of acoustic models and conducting tests offline and real-time online. To the best of our knowledge, no Indonesian ASR for home automation has been specifically implemented using HTK.

In the next section II summarizes the proposed steps for implementing noisy speech recognition for Indonesian. Section III adds a detail explanation with a real case illustration. Section IV discusses and summarizes the important result and the drawback of this research. The last section concludes the most important results of this writing.

II. Proposed procedures

A procedure is proposed to build a small vocabulary automatic speech recognition system (SVASR) using HTK. In general, these steps are to determine the grammar of the language to be supported, creating a speech training database, feature extraction, creating and training sub-word acoustic, i.e. model training, and testing and deployment.

A. Grammar

In designing the SVASR the first thing is to determine the grammar received by the ASR. Grammar for SVASR differs from grammar from LVASR in terms of the number of words for SVASR is less. Grammar SVASR is simpler with fewer production rules. After the grammar has been determined, the following steps are followed:

a) Determining word lists: Based on grammar, all word lists can be found and sorted.

b) Make a spelling dictionary (subword pronunciation dictionary): Every word that is listed on the word list is mapped to its subword (phoneme). For a list of phonemes to be used are adapted from the list of the International Phonetic Association (IPA) [5], or TI-MIT [2][6].

B. Creating a noisy database

To create a voice database, the procedures that are carried out are:

a) Voice recording: Sound recorded using a microphone that matches the desired conditions and matches the test conditions. In the sense that it avoids differences between training conditions and test conditions. For example, when recording was performed in a quiet laboratory condition, but then testing is carried out in high disturbance conditions, the accuracy of testing will certainly decrease. Also recorded voice carefully designed to refers to the word list in step A (grammar) before. By not recording words that are outside the word list, the recording process is certainly faster compared to recording additional unused words. The sampling frequency used is 16KHz [6], and the results are saved as a 16-bit PCM file. In recording, microphone sensitivity is also tuned to produce a good sound database. If the microphone is set too sensitive, the sound that occurs can contain too much noise, which can reduce ASR performance.

b) Labeling of the voice database: each recorded file is then marked by subword or phonemes. Softwares used to label voice are Speech Filing System, SFS [7] and Praat [8]. SFS has advantages in terms of compatibility with HTK. Sound files are displayed in the spectral domain, then each phoneme is marked in sequence based on time. Naming phonemes using the International Phonetic Association (IPA) guidance [5] or ARPABET even TIMITBET. While examples of labeling each particular phoneme against its spectral model follow the example of CMU Arctic Speech Databases [9] or TIMIT Database [2]. Recording is also performed upon noise patterns in addition to the words in grammar.

C. Feature Extraction

Recorded sound files are then converted to feature format. The format used is the Mel Frequency Cepstral Coefficient (MFCC) [10]. After that, the 39 coefficients could be filtered using the PCA method [11]. This will increase the recognition accuracy in case the input sound has been distorted by noise [12]. Besides that PCA also has the advantage of reducing computer computation which gains performance on low computing systems and embedded devices.

D. Acoustic Modeling

After obtaining feature files in the MFCC format, an acoustic model was constructed in the Hidden Markov Model format with an emission distribution in the form of a Gaussian Mixture Model (HMM-GMM) [13]. This phase is often also called the training phase. The proposed acoustic model is sub-word based. That is one HMM model for a particular phone (sub-word), for example, one HMM for **/s/**, one HMM for **/ah/** and so on.

E. Testing

After the acoustic model is obtained, it is time to test the model. In this phase, the level of accuracy or success of an SVASR is measured.

F. Refinement

Sometimes after in the field testing online or in real-time, there are errors. These errors can be corrected if the test data has been recorded. Data from this field can be used to correct the errors. In this case, the system is retrained by including the newly acquired error data.

G. Deployment

After the desired results of some performance criterions are achieved, the ASR is planted into the target environment. The deployment environment can be different but possibly the same as the offline and online testing environment. If the testing environment is different, certain adaptation techniques must be performed. Adaptation processes are not needed if the planting environment is exactly the same between the testing environment or the same as the sound database recording environment.

Finally, all the steps above are summarized in Figure-1.

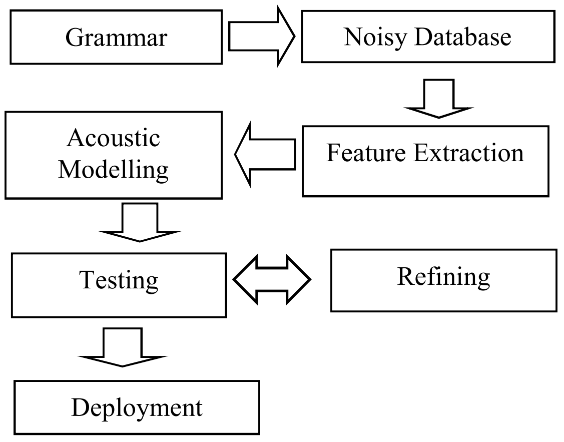


Figure. 1. *Research method for SVASR*

III. ILLUSTRATION

The following example illustrates the development of SVASR with the proposed procedure. The application that is exemplified is building a recognition system for the Indonesian voice-based electrical switcher *(*IDSwitcher*)* and planting it into home automation electrical circuit switches.

A. Prerequisites software

Besides preparing computer and Bluetooth microphones, supporting software, libraries and applications must be installed as seen in TABLE I. HTK is downloaded and then compiled using Cygwin-gcc or Visual Studio. A slight modification of HTK source prior compilation is required to support raw database recording, volume scaling and Principle Component Analysis, PCA[11].

TABLE I. SOFTWARES AND DEVELOPMENT LIBRARIES

No	Software, Libraries		
	Software	***Version***	***License***
1	Speech Filing System	Release 4.9, SFSWin 1.9	UCL
2	HTK	3.4.5	Cambridge University
3	Visual Studio	2008	Microsoft
4	Praat	6.0.43	Paul Boersma
5	Windows OS	Pro 10.0.1439	Microsoft
6	Cygwin-gcc	2.11.2(0.329/5/3) - i686	GPL
7	Java VM and SDK	1.8.0_40	Oracle

B. Grammar and words list

This version of IDSwitcher is designed to recognize Indonesian words in TABLE II.

TABLE II. WORDS LIST DESIGNED FOR IDSWITCHER

Words Type	***Words (Indonesian)***	***(Words) English***
Number	Satu, Dua, Tiga, Empat	One, two, three, four
Color	Merah, Hijau, Biru Kuning	Red, green, Blue, Yellow
Electrical Devices	Tivi,teve, pompa air, kipas angin, ase	television, water pump, fan, air conditioner
Switch Status	On, Off, Hidup, Mati	On, Off, Live, Death

The language grammar for IDSwitcher is as shown in Figure 2. As depicted by the grammar, this is a continuous word speech recognition. In this figure, the first 4 natural numbers in the Indonesian language represent the number of electrical switches for home installation. The *bar '|'* means logical 'OR'.

```
$SIL_NOISE = {SILSP | NOISE} ;
$tv = TV;
$pump = pompa | pompa $SIL_NOISE air;
$fan  = kipas | kipas $SIL_NOISE angin ;
$AC = AC;
$SW_1 = NUM_1 | biru | $tv ;
$SW_2=  NUM_2 | kuning | $pump;
$SW_3=  NUM_3 | hijau | $fan;
$SW_4 = NUM_4 | merah | $AC;
$SWITCH_DGT =  $SW_1| $SW_2 | $SW_3 | $SW_4  ;
$SILENCE = SILSP ;
$ON = on | hidup;
$OFF = mati | off;
$ON_OFF = $ON| $OFF;
$CONT =    $SWITCH_DGT $SIL_NOISE $ON_OFF |
SILSP ;
$SENTENCE = { <$CONT $SIL_NOISE>  } ;
```

Figure. 2. *Grammar* for IDSwitcher

And for the dictionary of the spelling of the sub-word pronunciation dictionary following the phoneme list in [6] is quoted in part as in Fig.3:

```
NOISE     t sp
NOISE     d  sp
NUM_0     k oh s oh ng
NUM_1     s ah t uh
NUM_2     d uw  ah
NUM_3     t ih g ah
NUM_4     ah m p ah t
```

Figure. 3. Pronunciation dictionary for IDSwitcher

So on the dictionary in Fig.3 above words are placed in the first column and the pronunciation is placed in the next columns. Sub-word **sp** at the end of some entries in the dictionary above stands for *short pause* which is if silence occurs but in a short time. The word NOISE represents noise that might occur during testing. For example, in Fig.3 noise that appears is defined as a pattern of certain phoneme sequences such as the sequence of two consonants without vowels between.

In Indonesian some words appear sequentially as contraction which is the suffix of the word is the prefix of the next word. An example is if the sentence NUM_3 NUM_4 is pronounced then /ah/ is the phoneme suffix of the word NUM_3 at the same time become the prefix to the next word NUM_4. In this case, **HVite** often only recognizes one word, which means that word deletion error occurs. The solution used is to create new words in the dictionary. An example is for the case of NUM_3 NUM_4 the solution is like in Fig. 4

```
NUM_3_4   t ih g ah m p ah t
```

Figure. 4. *Contraction modeling in the dictionary*

Contraction modeling with *dictionary entry* is effective enough for SVASR, but could be a problem when applied to LVASR because of huge entries in order to accommodate all possibilities of contractions.

C. Speech database

a) Voice recording: sound is recorded using HSLab and HVite software with Bluetooth microphone input devices. Other recording software such as Audacity and CoolEdit is possible, but these software are not used, because the desired database is really the one that equal with the test conditions. Bluetooth microphones used are common branded ones and are easily found on the market. This device can be obtained at a low price but without the noise cancellation function. This microphone also contains only one single sensor as according to various literature is less good than those containing an array of sensors. An example of a microphone used can be seen in Fig. 5 The advantage of using a wireless microphone is that the speaker is free to move and not tied to an ASR processor, computer or smartphone.

Figure. 5. *Bluetooth Microphone*

The recording processes take place on a housing situation, office and laboratory conditions with a moderate level of disturbance. Setting sensitivity of the microphone also affects accuracy. For the Bluetooth microphone, the amplitude or volume is scaled by *c*. Suppose *x(t)* is the audio signal produced by the microphone then *x^(t)* is a signal that is processed by HTK after being scaled according to the equation (1).

$$\hat{x}(t) = cx(t), \quad 0 < c < 1 \tag{1}$$

This scaling aims to reduce environmental noises encounter such as human steps, animal sounds and possibly the sound of the engine of a vehicle. Many times, these noises trigger the silence detector of **HVite**, showing that these noises are detected as words from a word list. This reduces the accuracy of SVASR.

As an illustration of the noise parameters, the condition of the room with a common standing fan within 30 cm from the microphone will show an *N* indicator of 35 dB without voice and total (*S*+*N*) 70 dB with commands voiced. The average power measurements of various recording environments are listed in Table III.

The sampling frequency used is 16000 Hz or the period between samples is 625 ns. The recorded file is stored in the raw format of HTK waveform, 16 bits signed integer.

TABLE III. VARIOUS NOISE CONDITIONS

No	Noise parameter under variety conditions		
	Condition	***Measurer***	***S+N, N (dB)***
1	Living room no fan	HVite	60, 20
2	Living room 30 cm fan from mic.	HVite	70, 35
3	Living room midnight silence	HVite	70, 15
4	Raining (0.3 scaled)	HVite	43, 15

b) Transcribing (annotating): Sound is transcribed using Praat [8], then converted into SFS [7] format and converted again into HTK format. This is because Praat cannot convert directly to the HTK format. To label training data, the subword (phoneme) sequences follow the grammar generated in section B. In Figure 6 a sample example of subword with Praat is shown for the order of Indonesian words NUM_2 NUM_1.

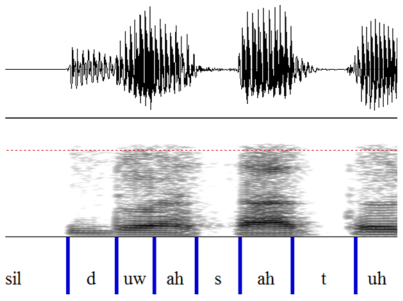


Figure. 6. *subword marking*

c) Noise data: in this phase *noise* data are included in the training database. Noise present is a must in online testing. An example of inclusive *noise* training is shown in SFS display as shown in Figure 7.

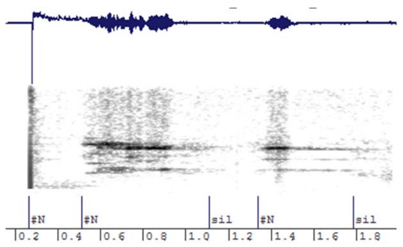


Figure. 7. *Inclusive Noise marking*

D. Feature Extraction

Feature extraction is used to convert waveform to MFCC speech feature format. In this case, the raw waveform sound is recorded from the *replay buffer* of HVite or using HSLab program recording. Feature extraction is performed by an HTK program called **HCopy**, which is to convert raw PCM sound files with extension **.htk** to speech feature files with extension **.mfc**. The specific MFCC format used is with the HTK code MFCC_0_D_A. This format is the MFCC format with spectral coefficients to zero, (zeroth coefficient), delta coefficient and acceleration coefficient [8]. Previously pre-processing was done, namely emphasizing with coefficients of 0.97 with windowing of 250 ms (400 samples) consisting of framing, striding of 100ms (160 samples) and overlapping of 150 ms (240 samples). After obtaining MFCC_0_D_A, then proceed with PCA [11], where based on the results of PCA, the number of the required coefficients can be reduced.

E. Acoustic model creation

In HTK, acoustic models are made using **HInit** and **HRest** and **HERest** programs. HInit initializes parameters using the Viterbi Extraction algorithm. HRest estimates HMM-GMM parameters using the Baum-Welch algorithm for a small vocabulary version [1]. HRest is outperformed HERest here because of better performance in a noisy environment and limited training data. This is because HERest is mainly intended to train acoustic speech where the exactly sub-word time boundary transcription is not completely present[1] and requires large data corpus. So for complete labeled data and small training examples, only HInit and HRest are used as shown in figure 8.

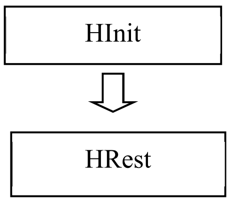


Figure. 8. *Creating the acoustic model from timed labeled corpus*

F. Testing

Tests were carried out with the **HVite** program. Hvite can perform testing offline using a recorded database or online or live test directly from the microphone. In this case, a Bluetooth microphone is used.

G. Refining

Repair is done by looking at the results of testings. If in online testing an error occurs in the sentence that has been spoken, then the sentence is relabeled and added to the database. After that, it is examined whether the previous sentence can be recognized correctly now. The results showed that this method succeeded in correcting the errors of online testing.

Sometimes even though error data have been entered in the database but still cannot eliminate errors, repairs can be done by looking at the phoneme sequence that is wrongly detected then adding the entry to the dictionary as a word explicitly. In Figure 9, we show a multi-version entry for spelling ***hijau*** (eng: green), whenever there is an entry that is not explicitly added mainly will cause an error.

hijau	h ih j au
hijau	ih j oh

Figure. 9. *Multiple word pronunciation accommodation in dictionary*

H. Deployment

The recognizer is planted on computers with Windows operating systems within the same noise environments between testing and training stages. The software that is planted is a modified **HVite** from HTK. The speaker will pronounce the digits repeatedly and randomly via the Bluetooth microphone input. **HVite** will record and recognize the speech based on the trained HMM-GMM acoustic model. The records kept can be used to refine the system whenever the error occurs.

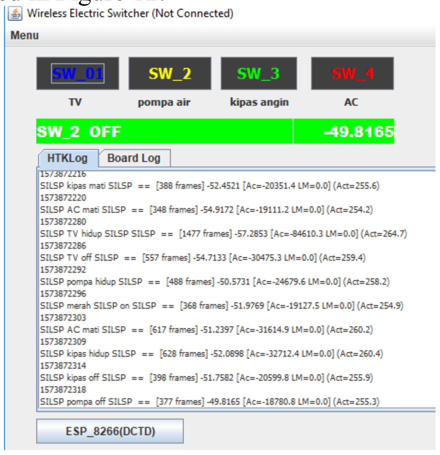
When applying to real-life applications like controlling electrical switch for home-based electrical appliances, a threshold is applied to revalidate recognition results. This mechanism is provided by HTK with a parameter *of frame probability*. An upper-layer application will execute a command if only the frame probability is higher than a threshold. If the value is below the threshold, the command is discarded.

An example of deploying the SVASR application is presented below. This configuration will enable the toggle of electrical switches using voice commands. The main components are:

1) HTK Speech Recognition server, is a personal computer or smartphone in which a modified HTK is installed and the Bluetooth microphone is connected. This software can be run in embedded devices or personal computers. The front end of the server running under a personal computer as depicted in Figure 10. The GUI form shown 4 electrical switches SW_1,…, SW_4. This application is developed using java programming language[14]. The user interface logs command received from the recognizer of HTK. Right below the buttons of switches we see the most recent command received. Next to this command, outlined also score of recognition or *total frame probability*. It this score fall below some threshold than the command from HVite is ignored, the circuit will not be switched. For example, in this experiment, we are using the threshold values as in (2). This value should be tuned differently, dependent on each working condition. The higher or stricter values chosen for example if we use the second epsilon in (2), will result in more command rejection. In this case, the speaker has to repeat the commands more frequent until it actually executed successfully. But the positive gain achieved is the reduction of the number of mistaken commands interpreted. On the other hand, if we choose the first threshold in (2), it will result in more electrical switches wrongly toggled.

$$\varepsilon_1 = -73.0 \quad \varepsilon_2 = -52.0 \tag{2}$$

2) Arduino Uno-Wifi R3 Robodyn, contains Espressif ESP 8266 connected with Atmel Atmega 328p through serial pins. Atmega328p[15] is used to control electrical switch while the ESP 8266[16] is used to communicate with the speech recognition server. The appearance of this board as in Figure 11. The implementation diagram of the whole system is presented in Figure 12.



```
1573872216
SILSP kipas mati SILSP  ==  [388 frames] -52.4521 [Ac=-20351.4 LM=0.0] (Act=255.6)
1573872220
SILSP AC mati SILSP  ==  [348 frames] -54.9172 [Ac=-19111.2 LM=0.0] (Act=254.2)
1573872280
SILSP TV hidup SILSP SILSP  ==  [1477 frames] -57.2853 [Ac=-84610.3 LM=0.0] (Act=264.7)
1573872286
SILSP TV off SILSP  ==  [557 frames] -54.7133 [Ac=-30475.3 LM=0.0] (Act=259.4)
1573872292
SILSP pompa hidup SILSP  ==  [488 frames] -50.5731 [Ac=-24679.6 LM=0.0] (Act=258.2)
1573872296
SILSP merah SILSP on SILSP  ==  [368 frames] -51.9769 [Ac=-19127.5 LM=0.0] (Act=254.9)
1573872303
SILSP AC mati SILSP  ==  [617 frames] -51.2397 [Ac=-31614.9 LM=0.0] (Act=260.2)
1573872309
SILSP kipas hidup SILSP  ==  [628 frames] -52.0898 [Ac=-32712.4 LM=0.0] (Act=260.4)
1573872314
SILSP kipas off SILSP  ==  [398 frames] -51.7582 [Ac=-20599.8 LM=0.0] (Act=255.9)
1573872318
SILSP pompa off SILSP  ==  [377 frames] -49.8165 [Ac=-18780.8 LM=0.0] (Act=255.3)
```

Figure. 10. *Graphical Front End Application of HTK Server*

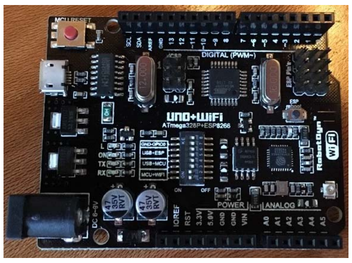


Figure. 11. *Embedded board equipped with Atmega 328p and ESP 8266*

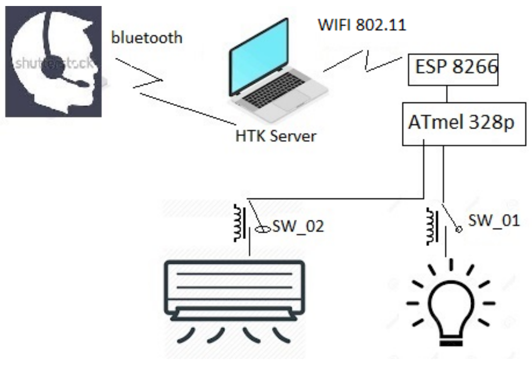


Figure. 12. *IDSwitcher implementation diagram*

IV. Results and Discussion

Using the proposed procedure we have built successfully a working prototype of SVASR for home automation in the Indonesian language. These are achieved mainly using libraries and programs of HTK. It is also introduced a modified HTK server on its *replaybuffer* to refine the system. We use frame recognition probability as a threshold to reject low score command. We applied volume scaler when operating too sensitive microphone to overcome miss behave HVite silence detector.

This system is limited to the dependency on the same environmental conditions between testing and training. It also requires the similarity of speakers and microphone input devices.

The later development should support a large number of vocabulary words. This is because the number of switches and the variety of electrical appliances connected to the system will certainly grow.

It is also expected to minimize or even eliminate the need for pre-recording procedures by utilizing the existing global database. The speech database creation processes, recording and annotating is considered and advanced tasks for the ordinary user of home automation application. So this phase should be skipped in the future release of the system.

V. Conclusion

The procedure using HTK is able to build noise-robust automatic speech recognition systems in a moderated noisy level environment, especially for a small vocabulary system. That is HTK is practically a solution to develop SVASR fast and accurate. When applying for voice-based electrical switcher applications for home automation in the Indonesian language, HTK achieved a 100% command completion rate. In this case, the system designed can be uttered to toggle all four switches correctly, by speaking the switch number, color and also by using the switch meaningful name.

References

[1] S. Young, E. Gunnar, G. Mark, T. Hain, and D. Kershaw, ‘The HTK Book version 3.5 alpha’, Cambridge University, 2015.

[2] C. Lopes and F. Perdigão, ‘Phone Recognition on the TIMIT Database’, 2009.

[3] D. P. Lestari, K. Iwano, and S. Furui, ‘A large vocabulary continuous speech recognition system for Indonesian language’, *15th Indones. Sci. Conf. Japan Proc.*, pp. 17–22, 2006.

[4] H. Kokubo, A. Amano, and N. Hataoka, ‘Robust Speech Recognition for Car Environment Noise’, *Electron. Commun. Japan, Part 3*, vol. 85, no. 11, pp. 2190–2197, 2002.

[5] C. D. Soderberg and K. S. Olson, ‘Illustration of the IPA: Indonesian’, *J. Int. Phon. Assoc.*, vol. 38, no. 2, pp. 209–213, 2008.

[6] K. Lee and H.-W. Hon, ‘Speaker-Independent Phone Recognition Using Hidden Markov Models’, *IEEE Trans. Acoust.*, vol. 37, no. 11, pp. 1641–1648, 1989.

[7] M. A. Huckvale, D. M. Brookes, L. T. Dworkin, M. E. Johnson, D. J. Pearce, and L. Whitaker, ‘The SPAR Speech Filing System’, *Eur. Conf. Speech Technol.*, pp. 305–308, 1987.

[8] P. Boersma and V. van Heuven, ‘Speak and unSpeak with Praat’, *Glot Int.*, vol. 5, no. 9–10, pp. 341–347, 2001.

[9] K. John and A. W. Black, ‘The CMU ARCTIC Speech Databases’, in *5th ICSA Speech Synthesis Workshop - Pittsburg*, 2004, pp. 223–224.

[10] S. B. Davis and P. Mermelstein, ‘Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences’, *IEEE Trans. Acoust.*, vol. 28, no. 4, pp. 357–366, 1980.

[11] L. I. Smith, ‘A tutorial on Principal Components Analysis’, 2002.

[12] T. Takiguchi and Y. Ariki, ‘PCA-Based Speech Enhancement for Distorted Speech Recognition’, *J. Multimed.*, vol. 2, no. 5, pp. 13–18, 2007.

[13] L. R. Rabiner, ‘A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition’, in *Proceedings of the IEEE*, 1989, vol. 77, no. 2, pp. 257–286.

[14] K. Arnold and J. Gosling, *The Java Programming Language, Fourth Edition*. 2005.

[15] Microchip, ‘ATmega328P 8-bit AVR Microcontroller with 32K Bytes In-System Programmable Flash DATASHEET’.

[16] Espressif Systems, ‘ESP8266EX Version 6.0 Datasheets’, 2018.