Sensors and Materials, Vol. 30, No. 4 (2018) 803–820
MYU Tokyo

S & M 1540

Speech Processing Based on Hidden Markov Model and Vector Quantization Techniques Applied to Internet of Vehicles

Neng-Sheng Pai,* Jun-Yu Chen, Pi-Yun Chen, and Jiun-Hao Hong

Department of Electrical Engineering, National Chin-Yi University of Technology,
No. 57, Sec. 2 Zhongshan Rd., Taiping Dist., Taichung 41170, Taiwan (ROC)

(Received October 20, 2017; accepted December 20, 2017)

Keywords: Internet of Vehicles (IoV), speech recognition, speaker recognition, mel-frequency cepstral coefficients (MFCC), hidden Markov model (HMM), Viterbi algorithm, vector quantization (VQ)

In this study, we develop an intelligence device to apply speech processing function in an Internet of Vehicles (IoV). The voice-based interactions will improve drive safety and in-time awareness of the vehicle status. This interaction can be achieved through speech recognition and response generation between the driver and the smart vehicle. Thus, the driver can focus on the driving. The proposed speech processing can be divided into three portions: (1) voice signal preprocessing, (2) speech recognition, and (3) speaker recognition. Firstly, speech signal pre-processing consists of five steps: sampling, pre-emphasis, frame, window function, and mel-frequency cepstral coefficients (MFCC), so as to be able to extract the characteristic parameters in the speech signal. Secondly, the speech model is built via the hidden Markov model (HMM), and the Viterbi algorithm is used to search the best sequence in the model to achieve the function of speech recognition. Finally, we use the Linde–Buzo–Gray (LBG) algorithm in vector quantization (VQ) to train for the speaker model, and then use cosine similarity to achieve the function of speaker recognition. The proposed speech processing function has been validated experimentally, and the experimental results demonstrate its feasibility for drivers to easily control the IoV system via voice-based command. In addition, the system distinguishes different speakers and provides the corresponding usage privileges, which will improve drive safety and in-time awareness of the general vehicle status.

1. Introduction

Internet of Vehicles (IoV) is an integrated network with wireless communication and information exchange to support traffic management, dynamic information service, and vehicle control. Intelligent transportation system is a typical application of IoV.

Most modern transportation vehicles are equipped with communication equipment [such as mobile devices, global positioning system (GPS) equipment, and embedded computers] and many sensors, which allow them to monitor, communicate, and process data. The mobile system that facilitates the communications between a vehicle and a vehicle, a vehicle and the

*Corresponding author: e-mail: pai@ncut.edu.tw
http://dx.doi.org/10.18494/SAM.2018.1780

ISSN 0914-4935

road, a vehicle and a human, and a vehicle and sensors is realized by the use of communications protocols such as HTTP, TCP/IP, SMTP, and WAP. Both driving safety and convenience are improved by such a system, which is the concept of the IoV.[1,2] Riding the trend of the Internet of Things (IoT), Tesla, the US electric automaker, has been adding IoT systems into its automobiles for several years and has also developed self-driving cars with the help of machine vision.[3] Samsung Electronics acquired Harman International Industries, another US automobile maker, in 2016 and announced that it will invest 100 billion dollars in the IoV by 2025. It is estimated that 75% of cars worldwide will be connected to the internet by 2020, creating huge IoV business opportunities.

A smart vehicle will be integrated with the audio and speech system for spoken-command recognition, voice-based driver identification, and text-to-speech synthesis applications. Voice-based interaction provides hand-free and eye-free to command the GPS system, radio, and smart phone and allows a driver to focus on driving. These can be achieved by speed command recognition, speech understanding, and response generations. In addition, voice-based driver identification can offer smart vehicle to response to the driver's critical commands and ignore any commands. Therefore, the main goal of this paper was the design of an IoV system capable of voice recognition and speaker identification. The first one is dynamic time warping (DTW),[4–6] which can adjust to voice durations of different lengths, and this solves the problem of alignment in recognition. In the second method, hidden Markov model (HMM)[7,8] statistics were used to describe voice signals and derive a probabilistic model, making the method suitable for continuous voice recognition. There are two main speaker identification methods: the Gaussian mixture model (GMM)[9–11] and data clustering.[12–15] In the first method, the feature parameters are modeled by GMM and long-term statistics are then taken from the model during recognition. This method has a high recognition rate but is computationally intensive. In the second method, a model is generated from voice feature parameters by clustering. This method has a low recognition rate and its computation time varies with codebook size.

This paper has five sections. In Sect. 1, research motivation and background are introduced. With a voice processing literature review, which is followed by research objectives and ideas. The voice processing system architecture and related hardware used in this paper are discussed in Sect. 2. The pre-processing of voice signals, feature parameter extraction, model training and voice recognition methods are presented in Sect. 3. Speaker model training methods and speaker identification are described in Sect. 4. Recognition results of the voice processing system and its application to IoV are discussed and explored in Sect. 5.

2. System Architecture and Hardware

In this study, a voice-based processing system is designed and its applications to IoV are developed. The hardware system architecture adopted in this study is shown in Fig. 1 and includes client, server, and IoV. The user opens a web page on the voice server from a smartphone, tablet, or in-vehicle device. The user gives commands by recording his voice with the built-in microphone in the client device and then transmitting via WebSocket protocol to the server for decision and execution. The server function includes a user interface, voice

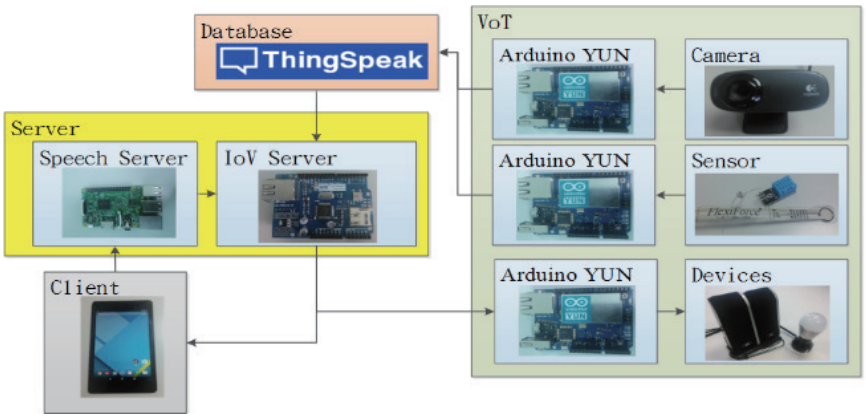


Fig. 1. (Color online) System architecture.

processing, and IoT control and was coded using the HTML5 and JavaScript web development languages. Voice processing turns the signals (received under WebSocket protocol) into commands. The commands are then passed to the IoV server and used to control IoV devices. In the IoV network, interface and data processing are implemented by Arduino Yun sensors, and other electronic devices allow the data to be uploaded to a cloud server or they can be used to turn vehicle devices on or off.

3. Voice Signal Processing and Recognition

3.1 Extraction of voice features

Voice signal processing involves converting the continuous analog signals of the human voice into discrete digital signals. Further processing by computers facilitates the communication between humans and computers. The process is divided into voice signal preprocessing and voice feature extraction. The four steps involved in voice signal preprocessing,[16–18] digital sampling, pre-emphasis, framing, and windowing, are shown in Fig. 2.

The human hearing range lies between the frequencies of 20 and 20000 Hz. In fact, humans are more sensitive to low-frequency sounds and thus easier to discern the distinctions in low frequency situations. To accommodate this auditory property, the MFCC method[18–20] was employed to extract the voice signal features, as seen in Fig. 3.

3.2 HMM based voice recognition

The goal of voice recognition is to convert the human voice into corresponding words through computers. Two main methods of voice recognition are DTW[4,5] and HMM.[7,8] Since the recognition with DTW is achieved by adjusting the time axes of two voices, a shared voice model is difficult to build. On the other hand, voices modeled through a long-term statistical process as in the HMM method deliver better performance in continuous voice recognition.

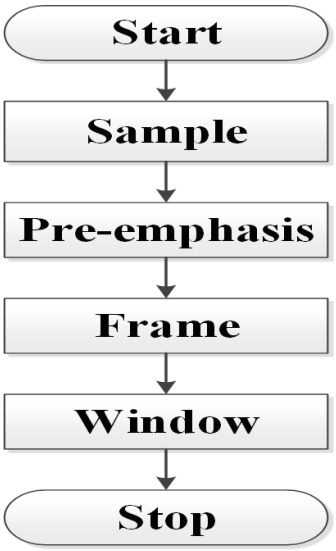


Fig. 2. Pre-processing steps for voice signals.

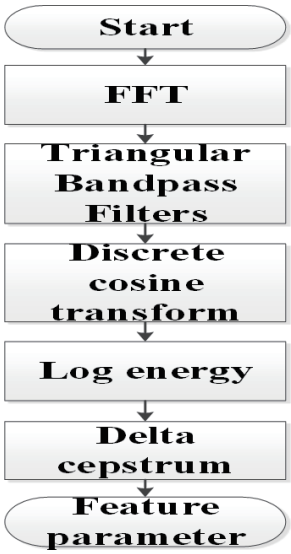


Fig. 3. MFCC computation flow chart.

When building an HMM acoustic model, phrases, words, phones, or phonemes can be used as units. Several Gaussian functions are also used to synthesize the probability density function. To approach a nearly arbitrary density distribution smoothly, a GMM is used in voice recognition to build an HMM model with the phone as a basic unit.

Since the distribution of voice signal features cannot be represented by simple probability distribution, GMM is used instead to describe the output probability in this paper, as shown by

$$b_j(o_t) = \frac{1}{\sqrt{(2\pi)^n \left|\Sigma_j\right|}} e^{-\frac{1}{2}(o_t-\mu_j)' \Sigma_j^{-1}(o_t-\mu_j)}, \tag{1}$$

where o stands for feature parameters, μ is the mean vector of the Gaussian distribution, Σ is the weight of the Gaussian distribution, and n is the total number of feature parameters.

The forward-backward algorithm is used to train the HMM model to start with and, for the known model λ, forward variables $\alpha_t(i)$ are defined as

$$\alpha_t(i) = P(o_1, o_2, ..., o_t, q_t = s_i \mid \lambda). \tag{2}$$

$\alpha_t(i)$ is the probability of observation sequence $\{o_1, o_2, ..., o_t\}$ at time t and state s_i. Forward probability can be computed recursively with these variables. Figure 4 shows the concept of forward probability.

1. Initialization

$$\alpha_1(i) = \pi_i b_i(o_1), 1 \le i \le N \tag{3}$$

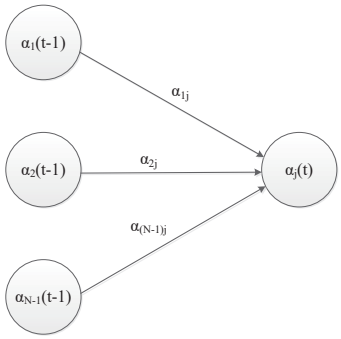


Fig. 4. Concept of forward probability.

2. Recursion

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i)\alpha_{ij}\right] b_j(o_{t+1}), 1 \le t \le T-1, 1 \le j \le N \tag{4}$$

3. Termination

$$P(O \mid \lambda) = \alpha_{t+1}(j) = \sum_{i=1}^{N} \alpha_T(i) \tag{5}$$

Next, backward variables are defined as

$$\beta_t(i) = P(o_{t+1}, o_{t+2}, ..., o_T \mid q_t = s_i, \lambda). \tag{6}$$

$\beta_t(i)$ is the probability of the observation sequence $\{o_{t+1}, o_{t+2}, ..., o_t\}$ after time t and state s_j. $\beta_t(i)$ can also be computed recursively using the following formula. Figure 5 shows the concept of backward probability.

1. Initialization

$$\beta_t(i) = 1, 1 \le i \le N \tag{7}$$

2. Recursion

$$\begin{aligned} &\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(o_{t+1})\beta_{t+1}(j), \\ &t = T-1, T-2, ..., 1, 1 \le i \le N \end{aligned} \tag{8}$$

3. Termination

$$P(O \mid \lambda) = \beta_t(i) = \sum_{i=1}^{N} \beta_1(i) \tag{9}$$

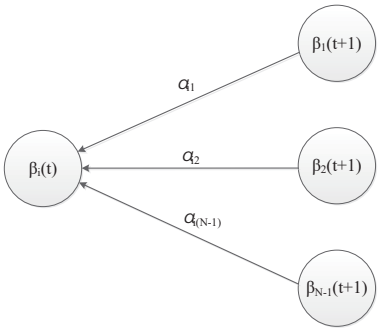


Fig. 5. Concept of backward probability.

Once forward variables $\alpha_t(i)$ and backward variables $\beta_t(i)$ are obtained, the evaluation problem can be solved as

$$\mathrm{P}(O \mid \lambda) = \sum_{i=1}^{N} \alpha_t(i)\beta_t(i), 1 \le t \le T. \tag{10}$$

The Viterbi dynamic programming algorithm[17] was proposed by Andrew Viterbi in 1967. Figure 6 shows the concept, where the *X*- and *Y*-axes indicate the time and HMM states respectively. The circle represents the observed output probability of the state at the time. The line between circles shows the transition probability. The Viterbi algorithm finds the best path from lower left to upper right.

The variables $\delta_t(i)$ are defined first as

$$\delta_t(i) = \max_{q_1,q_2,\ldots,q_{t-1}} P(q_1, \ldots, q_{t-1}, q_t = s_i, o_1, o_2, o_t \mid \lambda) . \tag{11}$$

$\delta_t(i)$ is the observation sequence with the highest probability before time t. Compute $\delta_t(i)$ with iteration and arrange the form to obtain

$$\delta_{t+1}(j) = \left[\max_i \delta_t(i)\, a_{ij}\right] b_j(o_{t+1}). \tag{12}$$

To obtain the best state sequence, $\psi_t(j)$ is used to record the best path at time t and state j. Computation details are as follows.

1. Initial value

$$\delta_1(i) = \pi_i b_i(o_i), 1 \le i \le N, \psi_1(i) = 0 \tag{13}$$

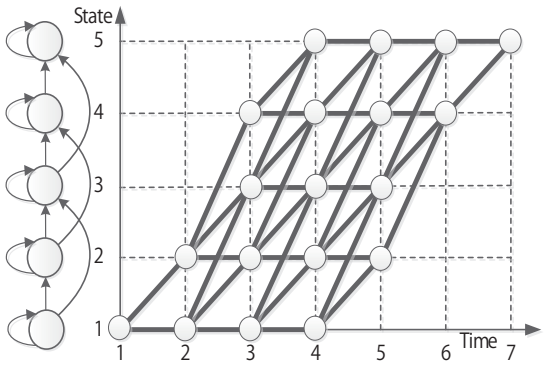


Fig. 6. Concept of the Viterbi algorithm.

2. Recursive calculation

$$\begin{aligned} \delta_t(j) &= \max_{1\le i\le N}\left[\delta_{t-1}(i)a_{ij}\right] b_j(o_t),\ 2\le t\le T,\ 1\le j\le N, \\ \psi_t(j) &= \arg\max_{1\le i\le N}\left[\delta_{t-1}(i)a_{ij}\right],\ 2\le t\le T,\ 1\le j\le N \end{aligned} \tag{14}$$

3. Termination

$$\begin{aligned} P^* &= \max_{1\le i\le N}[\delta_T(i)] \\ q_t^* &= \arg\max_{1\le i\le N}[\delta_T(i)] \end{aligned} \tag{15}$$

4. Backtrack path

$$q_t^* = \psi_{t+1}(q_{t+1}^*), t = T-1, T-2, ..., 1 \tag{16}$$

The Baum–Welch re-estimation algorithm is then used to adjust HMM model parameters λ, so that the observation sequence will have the highest probability under the conditions of this model. When refining the model, a statistical method is sufficient for the re-estimation of parameters if both the observation and state sequences are available. However, in the HMM model, the observation sequence is known, but the state sequence is not. Therefore, in this study, we used the Baum–Welch re-estimation algorithm,[17] an instantiation of the expectation maximization (EM) algorithm, to solve this problem.

Figure 7 shows the concept of the Baum–Welch re-estimation algorithm. Parameter re-estimation can be represented with forward variables $\alpha_t(i)$ and backward variables $\beta_t(i)$. For representation convenience, two variables are defined to simplify the formula. First, the variable $\xi_t(i, j)$ is defined as

$$\xi_t(i, j) = P(q_t = S_i, q_{t+1} = S_j \,|\, O, \lambda). \tag{17}$$

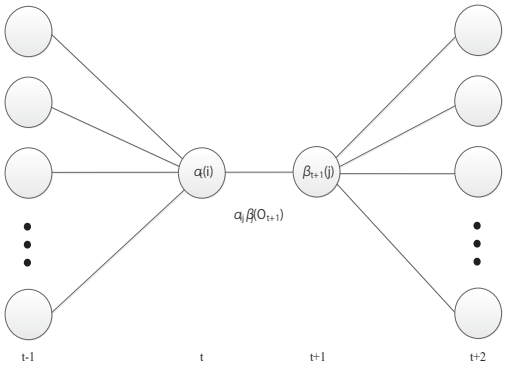


Fig. 7. Concept of Baum–Welch re-estimation algorithm.

Substitute the forward–backward algorithm into Eq. (17) and rearrange the form to obtain

$$\xi_t(i,j) = \frac{\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)}{\sum_{j=1}^{N}\alpha_t(j)\beta_t(j)}. \tag{18}$$

Then another variable $\gamma_t(i)$ is defined as

$$\gamma_t(i) = \sum\nolimits_{j=1}^{N}\xi_t(i,j) = \frac{\alpha_t(i)\beta_t(j)}{\sum_{i=1}^{N}\alpha_t(i)\beta_t(j)}. \tag{19}$$

The parameters of the HMM model can be estimated using the data in Fig. 7 and Eqs. (18) and (19) The computation equations are as follows.

Re-estimation equation of π:

$$\bar{\pi}_i = \gamma_1(i). \tag{20}$$

Re-estimation equation of a_{ij}:

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1}\xi_t(i,j)}{\sum_{t=1}^{T-1}\gamma_t(i)}. \tag{21}$$

Re-estimation equation of b_j:

$$\bar{b}_j(v_k) = \frac{\sum_{t=1,\, o_t=v_k}^{T}\gamma_t(j)}{\sum_{t=1}^{T}\gamma_t(j)}. \tag{22}$$

Lastly, the original HMM model is replaced with the re-estimated model $\overline{\lambda} = \{\overline{A}, \overline{B}, \overline{\pi}\}$. Re-estimation is repeated until model λ reaches optimization.

After a series of model training, a voice model with the phone as the basic unit was built. To recognize a voice sentence, the trained models should be connected as shown in Fig. 8.

4. Vector Quantization (VQ)-based Speaker Identification

A human voice signal carries a great deal of unique information because each person's tone quality is different. The main causes of these differences, which can be used for speaker identification, include vocal organs, the content of speech, and the way the speech is delivered. Although expressions, features, and habits of a language can be learned, the characteristics of a vocal organ cannot be changed or imitated; thus acoustic features are used in most speaker identification methods.

4.1 Linde–Buzo–Gray (LBG) algorithm

The LBG algorithm[13] uses clustering to derive a data codebook and train feature parameters of a voice segment to reduce the amount of information in the original voice. The advantages of using this method include less distortion and a lower bit rate. The LBG algorithm is basically a k-means clustering algorithm,[21] which divides all voice feature parameters or training vectors into k clusters. The cluster centers become the codevectors that represent their clusters. Figure 9 shows the algorithm flow chart.

1. Initialization

   Give the training vector X_m and distortion threshold ε, which is a very small positive number.
2. Compute initial center and distortion rate

   Set the number of codevectors N to be 1 and compute the mean center of the training samples c_1^* and total distortion rate D_{ave}^*, as shown in Eqs. (23) and (24).

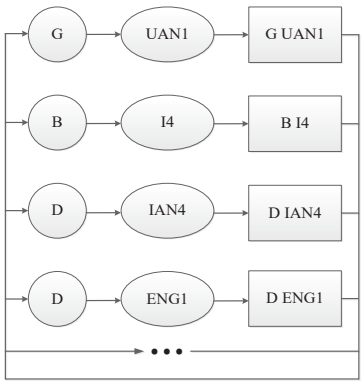


Fig. 8. Continuous voice recognition.

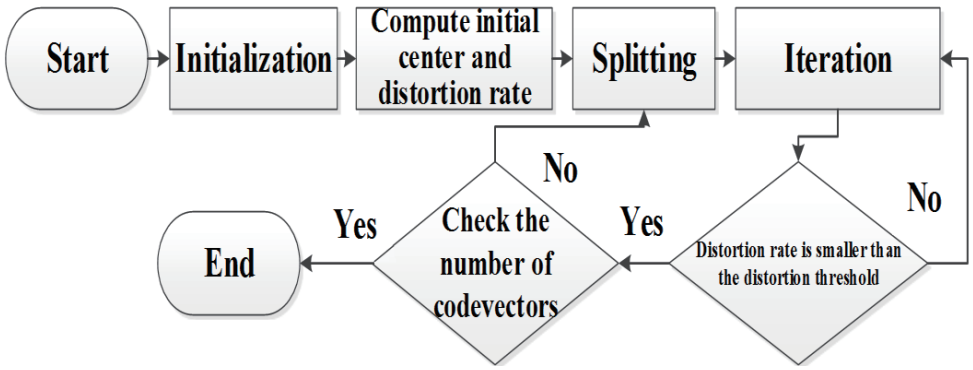


Fig. 9. LBG algorithm flow chart.

$$c_1^* = \frac{1}{M}\sum_{m=1}^{M} X_m \tag{23}$$

$$D_{ave}^* = \frac{1}{Mk}\sum_{m=1}^{M} \left\| X_m - c_1^* \right\|^2 \tag{24}$$

3. Splitting

Multiply each codevector by the coefficient of disturbance, as shown in Eq. (25), where i is the number of codevectors (i = 1, 2, …, n). Let $N = 2N$ and each codevector be split into the following two codevectors:

$$\begin{aligned} c_i^{(0)} &= (1+\varepsilon)c_i^*, \\ c_{n+i}^{(0)} &= (1-\varepsilon)c_i^*. \end{aligned} \tag{25}$$

4. Iteration

Let the initial distortion rate $D_{ave}^{(0)} = D_{ave}^*$. Reset the iteration index to 0.

(1) Compute the distance between each training set and its center, as shown in Eq. (26). Set the sample with the shortest distance to the center to be $Q(x_m)$, as shown in Eq. (27).

$$\left\| x_m - c_n^{(i)} \right\|^2 \tag{26}$$

$$Q(x_m) = c_n^{(i)} \tag{27}$$

(2) Update all codevectors as shown by

$$c_n^{(i+1)} = \frac{\sum_{Q(x_m)=c_n^{(i)}} x_m}{\sum_{Q(x_m)=c_n^{(i)}} 1}. \tag{28}$$

(3) Increase the iteration index by 1 as shown by

$$D_{ave}^{(i)} = \frac{1}{Mk} \sum_{m=1}^{M} \left\| x_m - Q_{(x_m)} \right\|^2. \tag{29}$$

(4) Compute and determine if the distortion rate is lower than the distortion threshold, as shown in Eq. (30). If not, go back to Eq. (26).

$$\frac{(D_{ave}^{(i-1)} - D_{ave}^{(i)})}{D_{ave}^{(i-1)}} > \varepsilon \tag{30}$$

5. Termination condition

Check if the number of codevectors has been reached. If not, go back to step 3.

Here M is the number of feature parameters, X_m the M-th feature parameter, D_{ave} is the distortion rate, and $C_i^{(0)}$ the center point of the i-th codevector in the 0-th iteration.

4.2 Cosine similarity and Euclidean distance

Cosine similarity, also called cosine distance, is a measure of similarity between two vectors in vector space by measuring the cosine of the angle between them, which will give an indication as to whether they are pointing in the same direction or not. The cosine similarity is 1 when the two vectors are pointing in the same direction. Two vectors at 90° have a cosine similarity of 0 and two diametrically opposed vectors have a cosine similarity of −1. Cosine similarity is dependent on vector direction and independent of vector length.

The law of cosines relates the lengths of the sides of a triangle to the cosine of one of its angles. Given the lengths of the three sides of a triangle, the angles can be computed using the law of cosines. Let the three sides of a triangle be r, s, and t, and their opposite angles be R, S, and T, respectively. The cosine of angle R is

$$\cos R = \frac{s^2 + t^2 - r^2}{2st}. \tag{31}$$

If the two sides s and t of the triangle are vectors, the above equation can be rewritten as

$$\cos R = \frac{\sum_{i=1}^{n} S_i \times T_i}{\sqrt{\sum_{i=1}^{n} (S_i)^2} \times \sqrt{\sum_{i=1}^{n} (T_i)^2}}, \tag{32}$$

where T_i and S_i are the components of the respective vectors T and S, respectively.

The Euclidean distance is commonly used to measure distance. It is defined as the ordinary distance between two points in space. For speaker identification, the Euclidean distance

between the test sentence and each N speaker model $d(o, v)$ is computed, as shown in Eq. (33), and the model sample with the smallest distance difference is chosen as the identification result.

$$d(o, v) = \frac{1}{T} \sum_{t=1}^{T} [\min_{1 \le i \le c} d(o_t, v_i)] \tag{33}$$

Here, v_i is the codeword, c the codebook size, o the test voice, and T the test voice sequence. Add o_t together and compute the shortest distance between it and the codewords in the codebook. The codeword with the shortest distance is the identification result.

In addition to the shortest distance computation, the lowest threshold can be added to identify whether the voice belongs to any of the clusters.

4.3 Receiver operating characteristic (ROC) curve

In the signal detection theory, the ROC curve[22] is a graphical binary classifier system used to select the best model or set the best threshold in the same model. When the signals or measurement results are continuous, a discrimination threshold must be used to define the boundary between classes. The *Y*-axis of the curve represents true positive rate (TPR), also known as sensitivity, which measures the proportion of correctly identified positives. Its formula is shown as

$$TPR = \frac{TP}{TP + FN}. \tag{34}$$

The *X*-axis represents the false positive rate (FPR), which is calculated as (1-specificity). Specificity measures the proportion of correctly identified negatives. The formula for FPR is shown as

$$FPR = \frac{FP}{FP + TN}, \tag{35}$$

where *TP* stands for true positive; *FP* stands for false positive, a Type I error; *TN* stands for true negative; and *FN* stands for false negative, a Type II error.

5. Experimental Results

Figure 10 shows the flow chart of the proposed voice processing system in this paper, the theories presented in Sect. 5 are tested and the resulting system is used in the IoV. The results of the experimental procedure and analysis are as follows.

(1) Recognition results of speaker verification (SV)
(2) Recognition results of speaker identification (SI)
(3) Results of system application to IoV.

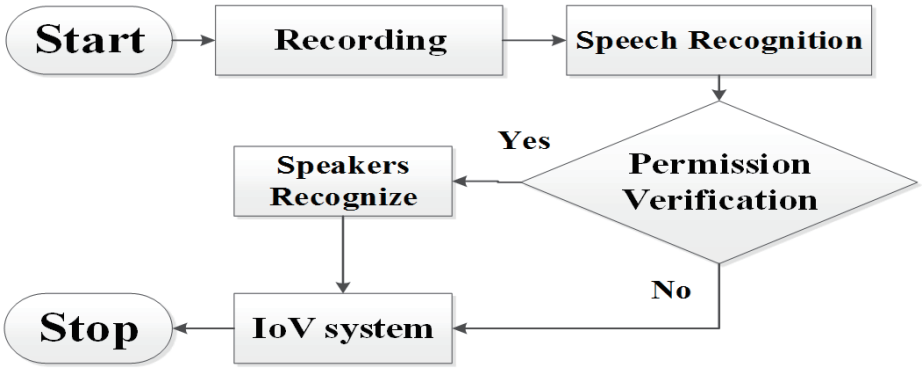


Fig. 10. The flow chart of the proposed voice processing system.

5.1 Results of speaker verification experiment

The speaker verification experiment was carried out in a known speaker mode and was aimed at selecting better parameters to be used in the speaker identification system. Twenty model samples and 50 test samples (25 positive and 25 negative) were used.

From Fig. 11, it can be seen that no matter what the codebook size, ROC curves are close to diagonal lines. This shows that the use of Euclidean distance and 13-dimensional codebooks does not generate discriminating classification. Figure 12 shows the ROC curves produced using cosine distance and 13-dimensional codebooks. They are more discriminating than those in Fig. 11. However, the differences between them are not significant. The trained features are too close to each other and this might be because the codebook dimensions are not large enough.

Figure 13 shows the ROC curves produced using Euclidean distance and 39-dimensional codebooks. Although better than those in Fig. 11, the curves of different codebooks are very similar and close to diagonal lines. There is still not enough discrimination. The classification results are fairly good when the codebook size reaches 256. However, the file was larger than that of feature parameters and the parameters were not used. Figure 14 shows the ROC produced curves using cosine distance and 39-dimensional codebooks. The discrimination improves steadily as the codebook size increases.

Tables 1 and 2 show the equal error rates (EERs) formulated from the above recognition results. Considering both discrimination and the amount of computation needed, we adopted a 39-dimensional codebook with a size of 64 and cosine distance for recognition, which were incorporated into the speaker identification system.

5.2 Results of speaker identification experiment

The speaker identification experiment was carried out using the parameters in the previous section. Models for three speakers were built first and each speaker model was trained with 20

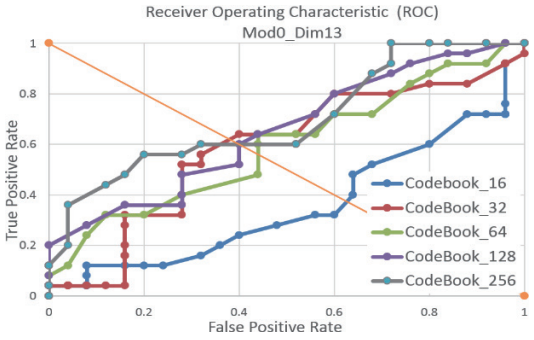


Fig. 11. (Color online) ROC curves produced using 13-dimensional codebooks and Euclidean distance.

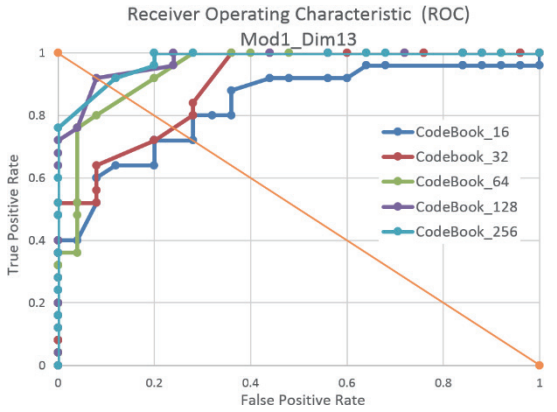


Fig. 12. (Color online) ROC curves produced using 13-dimensional codebooks and cosine distance.

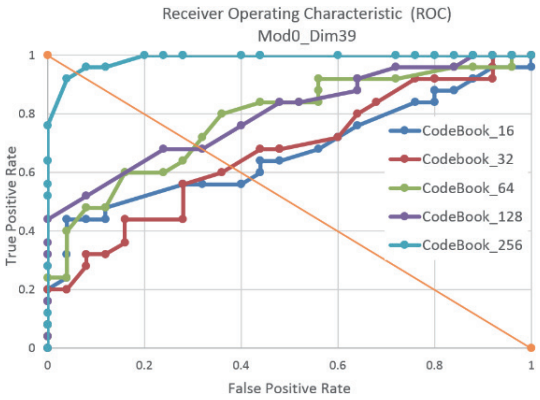


Fig. 13. (Color online) ROC curves produced using 39-dimensional codebooks and Euclidean distance.

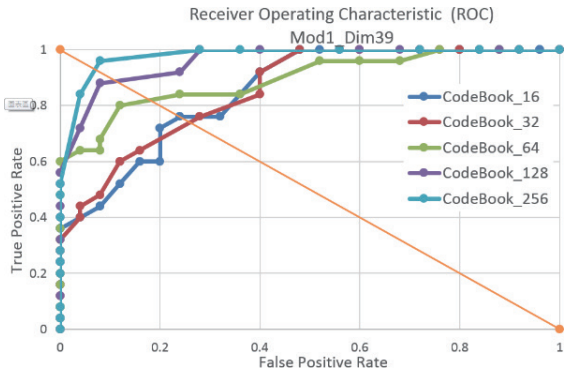


Fig. 14. (Color online) ROC curves produced using 39-dimensional codebooks and cosine distance.

Table 1
Recognition results obtained using codebooks of various sizes and dimensions and Euclidean distance.

EER		Code book dimension			
		13		39	
		Error rate	Threshold	Error rate	Threshold
Codebook size	16	62	0.120	42	0.390
	32	38	0.250	37	0.405
	64	40	0.360	30	0.440
	128	40	0.400	32	0.490
	256	36	0.460	5	0.475

Table 2
Recognition results obtained using codebooks of various sizes and dimensions and cosine distance.

EER		Code book dimension			
		13		39	
		Error rate	Threshold	Error rate	Threshold
Codebook size	16	28	0.560	24	0.400
	32	23	0.475	26	0.380
	64	13	0.435	16	0.360
	128	8	0.410	10	0.350
	256	10	0.400	6	0.320

sentences and 25 tests were performed for each speaker. The experimental results are shown in Table 3. The experimental results indicated a greater than 90% hit rate. This confirmed that the proposed device could provide promising results for response generations to the drivers' critical commands.

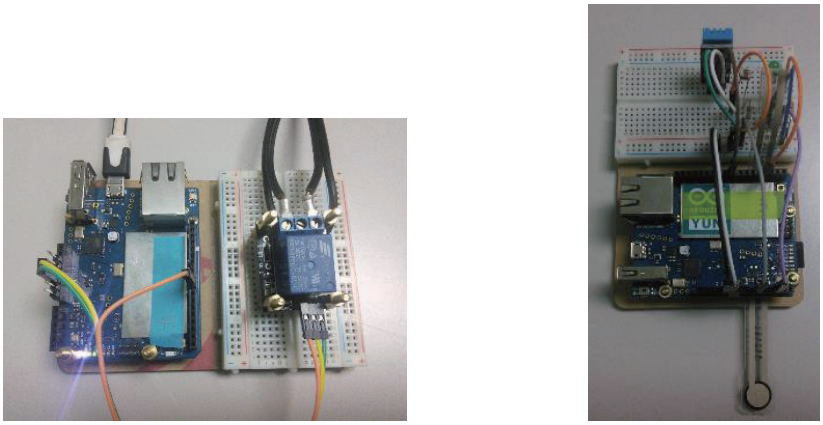
5.3 Results of system application for IoV

After it was completed, the voice processing system was tested with the IoV. Figure 15(a) shows the relay module used to simulate car interior lighting, and Fig. 15(b) shows the sensor module. In this module, the optical sensor measured the ambient light level outside the car and was used by the IoV to decide to turn the headlights on or off. A temperature/humidity sensor monitored the engine temperature to determine if it was within the normal range. Pressure sensors particularly flexible pressure sensor, were used to measure tire pressure.

The voice server in the operating state is shown in Fig. 16. The commands were given to the voice server through the user interface, as shown in Fig. 17, to access information on the vehicle (see Fig. 18) and control IoV-related devices (see Fig. 19). Figure 20 shows the results of speaker identification. The experimental results showed the IoV could respond to the drivers' requests and indicate warnings to the drivers, such as for headlight control, engine temperature monitor, unfastened seat-belt, and so on.

Table 3
Results of speaker identification experiments.

Speaker	Correct identification	Identification rate (%)
JHH	24	96
WLC	24	96
YCL	23	92



(a) (b)

Fig. 15. (Color online) Arduino module. (a) Relay module and (b) sensor module.

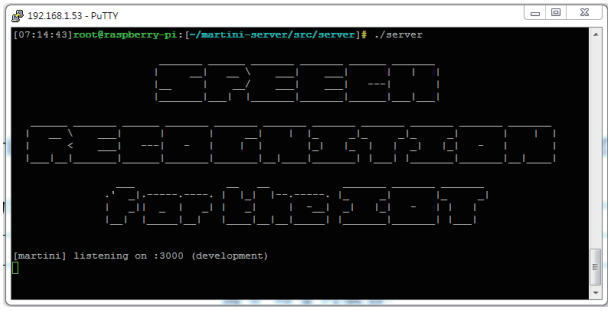


Fig. 16. (Color online) Voice server.

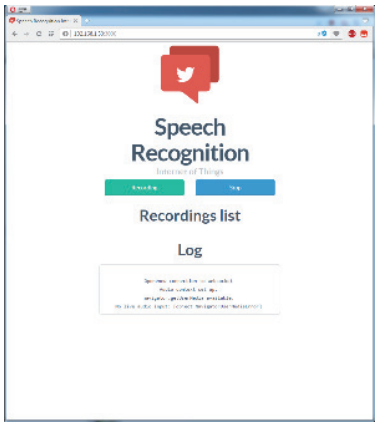
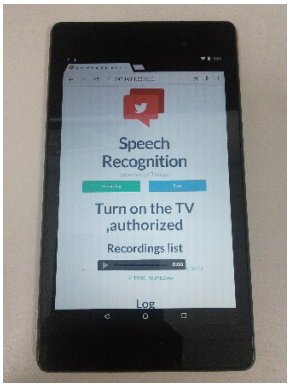


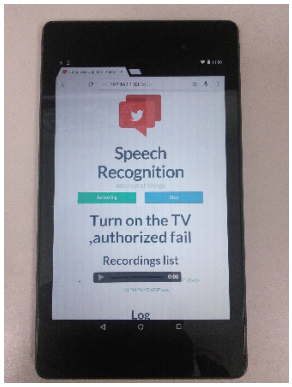
Fig. 17. (Color online) User interface.

Fig. 18. (Color online) IoV system information.

Fig. 19. (Color online) IoV system equipment.



(a)



(b)

Fig. 20. (Color online) Results of speaker identification. (a) Identified user and (b) unidentified user.

6. Conclusion

A voice processing application for an IoV system is presented in this paper and implemented using a Raspberry Pi and Arduino. The system comprised two main parts, namely voice recognition, which was built using the MFCC method to extract voice features, and HMM for long-term statistics. The model was then optimized with forward-backward, Viterbi, and Baum–Welch algorithms. Voice recognition was realized on the basis of this optimized voice model. The second part was speaker identification. Speaker models were built by voice encoding and the LBG algorithm that employs the concept of data clustering. The best thresholds were found through ROC curves and cosine similarity was then used to implement the functions of speaker identification. The results of voice processing were transmitted to the IoV system. From the results of recognition, the system determines whether the voice commands should be executed or not. The IoV system can monitor the vehicle situation via sensors and activate the related services automatically on the basis of the IoT concept.

Acknowledgments

The financial support of this research by the National Science Council of Taiwan, under Grant no. MOST 105-2221-E-167-026 is greatly appreciated.

References

1 S. H. Chen, J. Y. Chen, and K. Y. Lu: IMIS **5** (2011) 567.
2 W. J. Fleming: IEEE Sens. J. **1** (2001) 296.
3 The Verge: Tesla shows what its self-driving cars see while on the road, http://www.theverge.com/2016/11/20/13693120/tesla-self-driving-car-elon-musk (accessed March 2017).
4 E. Keogh and C. A. Ratanamahatana: Knowl. Inf. Syst. **7** (2005) 358.
5 G. Al-Naymat, S. Chawla, and J. Taheri: AusDM '09 **101** (2009) 117.
6 S. Salvador and P. Chan: Intell. Data Anal. **11** (2007) 561.
7 L. R. Rabiner: Proc. IEEE. **77** (1989) 257.
8 S. Z. Yu: J. Artif. **174** (2010) 215.
9 R. Saeidi, H. R. S. Mohammadi, and T. Ganchev: IEEE T. Audio. Speech **17** (2009) 344.
10 K. Park, J. S. Park, and Y. H. Oh: IEEE Trans. Consum. Electron. **56** (2010) 1123.
11 B. Xiang and T. Berger: IEEE Trans. Speech Audi. P. **11** (2003) 447.
12 S. Lloyd: IEEE Trans. Inform. Theory **28** (1982) 129.
13 Y. Linde, A. Buzo, and R. M. Gray: IEEE Trans. Commun. **28** (1980) 84.
14 D. K. Burton: IEEE Trans. Acoust. Speech Signal Proc. **35** (1987) 133.
15 T. Kinnunen, E. Karpov, and P. Franti: IEEE Trans. Audio Speech **14** (2006) 277.
16 J. S. Jang: Audio Signal Processing and Recognition, http://mirlab.org/jang/books/audioSignalProcessing/ (accessed April 2017).
17 N. S. Pai, S. X Chen, P. Y. Chen, C. L. Kuo, and H. Y. You: ICIC-EL **7** (2016) 1901.
18 F. Zheng, G. Zhang, and Z. Song: J. Comput. Sci. Technol. **6** (2001) 582.
19 T. Ganchev, N. Fakotakis, and G. Kokkinakis: Proc. SPECOM-2005 **1** (2005) 191.
20 V. Tiwari: INT. J. Emerg. Tech. **1** (2010) 19.
21 T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu: IEEE Trans. Pattern Anal. **24** (2002) 881.
22 H. Altay Guvenir and M. Kurtcephe: IEEE Trans. Knowl. Data Eng. **25** (2012) 2356.

About the Authors

Neng-Sheng Pai received his B.S. and M.S. degrees from the Department of Automatic Control Engineering of Feng Chia University, Taichung, Taiwan, ROC, in 1983 and 1986 respectively. In December 2002, he was awarded a Ph.D. by the Department of Electrical Engineering, National Cheng Kung University, Tainan, Taiwan, ROC. He is currently a professor in the Department of Electrical Engineering, National Chin-Yi University of Technology, Taichung, Taiwan, ROC. He was also the chairman of the department from 2004 to 2007. His current research interests include fuzzy, advanced control, and microprocessor systems.

Jun-Yu Chen was born in Jan 1993. He is a master's student in the Department of Electrical Engineering in National Chin-Yi University of Technology, Taiwan. His main research focuses on deep learning of image recognition.

Pi-Yun Chen was awarded a Ph.D. by the Graduate School of Engineering Science and Technology from National Yunlin University of Science & Technology in 2011. She is now an assistant professor in the Department of Electrical Engineering, National Chin-Yi University of Technology, Taiwan. Her current research interests include fuzzy and advanced control systems.

Jiun-Hao Hong was born in May 1992. He received his master's degree in electrical engineering from the National Chin-Yi University of Technology, Taiwan, in January 2017. His main research interests include heuristic algorithms.