Study on speech fingerprints based on spikegram using sparse coding technique

By Dung Kim TRAN

A thesis submitted to
School of Information Science,
Japan Advanced Institute of Science and Technology,
in partial fulfillment of the requirements
for the degree of
Master of Information Science
Graduate Program in Information Science

Written under the direction of
Professor Masashi UNOKI

September, 2017

Study on speech fingerprints based on spikegram using sparse coding technique

By Dung Kim TRAN (1510217)

A thesis submitted to
School of Information Science,
Japan Advanced Institute of Science and Technology,
in partial fulfillment of the requirements
for the degree of
Master of Information Science
Graduate Program in Information Science

Written under the direction of
Professor Masashi UNOKI

and approved by
Professor Masato AKAGI
Professor Jianwu Dang

August, 2017 (Submitted)

Abstract

Keywords: Speech fingerprint, spikegram, matching pursuit algorithm, Gammatone filterbank, non-negative matrix factorization

With the advances of information technologies, the Internet has become a convenient place where we can store and share our media content and a large portion of the content is in the form of audio signals. Audio fingerprints have been adopted to a wide variety of useful applications to organize and retrieve audio content effectively. A typical application, which uses audio fingerprints to identify audio content, includes two basic steps. At first, a database of fingerprints is needed to store information about all available audio content. Then, given an unknown excerpt of an audio signal, an audio fingerprint technique derives its fingerprint, and then a search algorithm searches the database to identify the audio signal. The key to the success of this particular application lies in the technique that is used to create the audio fingerprints.

A lot of effort has long been invested in studying audio fingerprints in response to the rising demands of practical applications in different industries, many techniques have evolved to increase the accuracy of audio fingerprints and to expand its capabilities. Many techniques were developed under the inspiration of biological mechanisms such as human hearing system, others using statistics and probability principles. Regardless of the complexity of the fingerprinting systems, their designs appear to have three steps. The first step is transforming audio waveforms into their representation models to reveal their features. The second step is extracting unique features of the audio signals. The third step is combining the extracted features to create audio fingerprints using hashing techniques.

In case of music, block-based audio fingerprint systems can achieve industrial standard; however, its performance reduces dramatically when applied to speech. The main cause of the problem is that the representation model of the speech signals is unable to provide suitable features for the process of creating speech fingerprints.

Block-based coding technique divides a speech signal into blocks and processes these blocks separately using discrete Fourier transform (DFT) or discrete cosine transform (DCT). This technique is substantially sensitive to signal shifting and phoneme scaling; randomly blocking speech signals does not take into account the alignment of acoustic cues. Given a speech content, it is unlikely for a speaker to produce signals having the same length and phonemes having the same duration. Perceptually, these speech signals may

sound the same but techniquely, the alignments and durations of their acoustic features are considerably different. Therefore, block-based coding creates different spectrograms for speech signals that have the same content and same speaker.

One way to overcome the drawback of block-based coding is using filterbank based shift invariant coding but this technique greatly increase the dimension of speech signals because of its convolutional calculations. Data redundancy on the representation models makes it difficult to recognize the underlying structures of speech signals.

The sources of problems do not confine only in the signal representation process, they also appear in the features extraction methods. To gather features for the production of speech fingerprints, several methods have been proposed; one used the signs of energies, another used the peaks of spectrogram. Although their methods achieved great experimental results, none of them take into account patterns of the features on the representation model.

This study aims at creating speech fingerprints using sparse coding technique. Recent studies have revealed that over-complete representation model, which can be produced by using sparse coding techniques, outperforms spectrogram in speech signal representation, which is the current technique used for audio signals like music. Indeed, sparse coding is a data-driven technique that is able to capture the underlying structures and adapt to the variations of speech.

In step one of the speech fingerprint extraction process, traditional block-based coding is replaced with sparse coding and spikegram will be used as the representation model for speech signals instead of spectrogram. In step two, features on spikegrams that represent linguistic content and speaker individuality are analyzed and extracted as they are necessary to improve the quality of speech fingerprints. In step three, a suitable hashing technique is selected to combine the extracted features to create final fingerprints for speech signals.

In fact,this research exploits the abilities of a matching pursuit algorithm in representing speech signals as equivalent spikegrams. It identifies the best sparse representation of a signal in an over-complete set of dictionary by performing a nonlinear approximation in the dictionary.

The matching pursuit algorithm we use is mathematically based on non-negative matrix factorization that approximates one matrix with two other non-negative matrices, one is a matrix of bases and the other is a matrix of coefficients. Therefore, the quality of the approximation depends on the selection of appropriate bases and the method of finding the coefficients. Two kinds of base spectra – Gabor dictionary and Gammatone dictionary are thus used. Results of using each kind of dictionaries will be shown and compared.

While analyzing features on spikegrams, there is one obstacle that we need to deal with is the unalignment of the features. As the result, their features on spikegrams are difficult to compare. Further more, the features that represent linguistic content and speaker individuality are encoded in the spike distributions. Therefore, it is necessary to convert spikegrams into a nomalized dimension to compare their features. Considering the robustness of of local binary pattern (LBP) in representing facial features that can be used for face recognition, this research has tried to utilize the advantages of LBP to

analyze the features of spikegrams. One characteristics of LBP that makes it become a candidate for analyzing features of spikegrams is its robustness in texture analysis. Our assumption is that spike distributions of speech signals that have the same content and same speaker should be similar and changes in speech content and or speakers should result in different spike distributions. Therefore, a variation of LBP was applied to this research. Moreover, histogram of the LBP matrix is calculated to evaluate the texture of the spikegram.

An experiment conducted to evaluate and compare the abilities of representing speech signals of block-based coding and sparse coding using a dataset consisting of 75 speech signals. A Gammatone filterbank, the representative for block-based coding, was used to create spectrograms for speech signals. To create spikegrams, the matching pursuit algorithm and two kinds of dictionaries, namely Gabor dictionary and Gammatone dictionary, was used for the sparse coding technique.

We found that the matching pursuit algorithm used with Gabor or Gammatone dictionary outperformed the Gammatone filterbank in decomposing speech signals by evaluating the original speech signals and the resynthesized signals with SNR and PESQ, and sparse coding provided a better representation model for speech signals. More importantly, spikegrams could be used to replace spectrograms as a better alternative in the process of creating speech fingerprints. Based on the experiments, we concluded that the spike distributions of spikegrams conveyed the linguistic content and voice identities of the utterers of speech signals. These features play a significant role in the accuracy and reliability of speech fingerprints.

Contents

This dissertation was prepared according to the curriculum for the Collaborative Education Program organized by Japan Advanced Institute of Science and Technology and Ho Chi Minh City University of Science.

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Research background

Audio fingerprints have been adopted to a wide variety of useful applications. Media agencies can use audio fingerprints to monitor where and when their products are played, then they record usage statistics for other purposes such as advertisement. Audio fingerprints can also be used as a security measure to detect illegal use of digitalized contents. A typical application using audio fingerprints to identify content of a song including two basic steps. At first, a database of fingerprints of all available songs is built. Then, given an unknown excerpt of a song, an audio fingerprint technique derives its fingerprint, and then a search algorithm searches the database to identify the song. The key to the success of this particular application lies in the technique that is used to create the audio fingerprints.

In computer science, audio fingerprint has been an intriguing yet challenging topic to many researchers over the past few decades. In response to the rising demands of practical applications in different industries, many techniques have evolved to increase the accuracy and reliability of audio fingerprints. For a number of reasons, some techniques were developed under the inspiration of biological mechanisms such as human hearing system, others using statistics and probability principles. Regardless of the intricacy of the fingerprinting systems, their designs appear to have three steps consisting of transforming audio waveforms into their representation models to reveal their features, extracting unique features of the audio signals, and combining the extracted features to create audio fingerprints using hashing techniques.

Three steps of an audio fingerprint technique do not seem to be many but each is an interesting research field and they have to support each other to form a thorough solution. Signal representation can be generalized into three categories that are block-based coding, filterbank based shift invariant coding, and sparse coding. Each type of coding has its unique characteristics and provides specific features on its representation model and the choices are left for the designers to decide depending on their ideas. Extracting relevent features from the representation models is a matter of pattern recognition problems. Again, how feature extraction methods are designed heavily relied on one's intentions

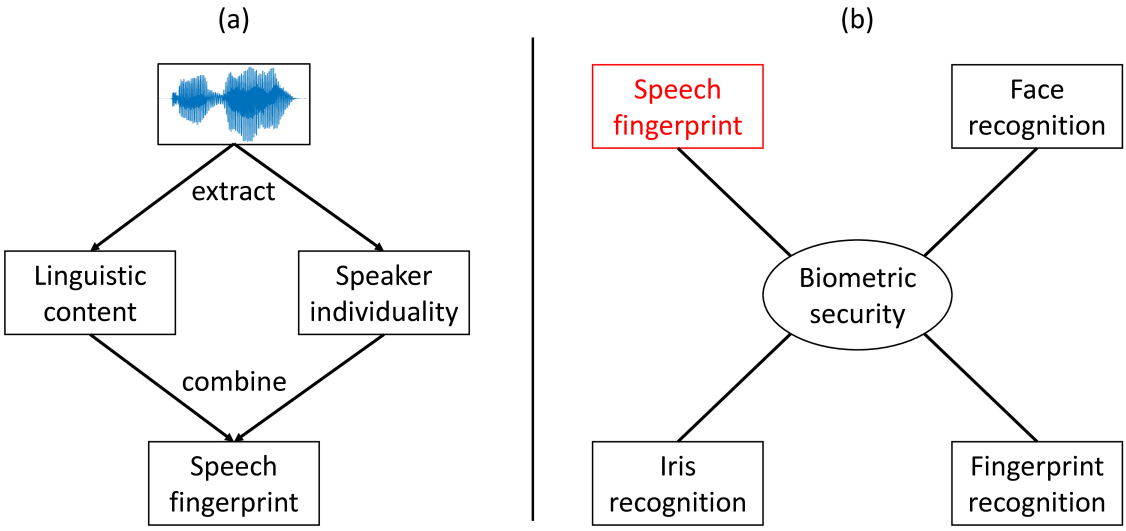

Figure 1.1: Speech fingerprints and biometric security. Panel (a) shows the important features of a speech signal – linguistic content and speaker individuality – in the process of creating speech fingerprints. Panel (b) shows some unique traits of humans including speech fingerprint that can be used in biometric security.

and the ultimate goal is to obtain distinctive information that can help identify audio signals. The hashing techniques that are used to create audio fingerprints can be as simple as applying a hash function to generate hash chunks or complicated as converting the set of features from step two into another set of features. No matter how sophisticated they are, the final fingerprints are bounded to some requirements such as distance metrics, storage size, and indexing speed.

Speech fingerprinting is a special case of audio fingerprinting that deals with unique characteristics of speech. The main purposes of speech fingerprints is similar to that of audio fingerprints as they are used for monitoring and identifying speech signals. A speech fingerprinting system also consists of two processes that are creating a database of speech fingerprints for all available speech signals and identifying unknown speech signals. The most important element of a speech fingerprint application is also the fingerprint extraction module. The overall design of this module also includes three steps that are creating representation models for speech signals, extracting inherent features of the speech signals based on their representation models, and hashing these features to generate final speech fingerprints. Working with speech signals, one may have to pay close attention to their instability. Unlike music clips, which are controled by machines and copied from same sources, acoustic events of speech signals vary significantly even if they have the same content and are produced by the same speaker.

As shown in Fig. 1.1(b), humans possess many unique traits such as fingerprints, irises, facial features, and we believe that speech can also become a measurement in biometric security. Regardless of the wide variations of speech, it is a matter of fact that we can recognize the spoken words and distinguish the voice of a speaker from others. Therefore,

we believe that speech signals must have contained information about linguistic content and speaker individuality as depicted in Fig. 1.1(a). These features are essential to the process of creating speech fingerprints; by extracting and combining them, the accuracy and reliability of speech fingerprint can be greatly improved.

1.2 Problem statement

In case of music, the audio fingerprint system described in the section above can achieve industrial standard; however, its performance reduces dramatically when applied to speech. The main cause of the problem is that the representation model of the speech signals is unable to provide suitable features for the process of creating speech fingerprints. Spectrogram is one of the most commonly used representation models for speech signals; it can be create by applying block-based coding or auditory filterbank based coding. However, none of the coding techniques is able to represent the underlying structure of speech signals. Due to the fact that a speaker produces different speech signals for the same content, conventional coding methods produce different spectrograms for the speech signals. As the result, different speech fingerprints are produced for the speech signals that have the same content and the same speaker.

1.3 Research purpose

This study aims to create speech fingerprints using sparse coding technique. Recent studies have revealed that over-complete representation model, which can be produced by using sparse coding techniques, outperforms spectrogram in speech signal representation [4][5] because sparse coding is a data-driven technique that is able to capture the underlying structures and adapt to the variations of speech. In step one of the speech fingerprint extraction process, traditional block-based coding will be replaced with sparse coding and spikegram will be used as the representation model for speech signals instead of spectrogram. In step two, features on spikegrams that represent linguistic content and speaker individuality will be analyzed and extracted as they are necessary to improve the quality of speech fingerprints. In step three, a suitable hashing technique will be selected to combine the extracted features to create final fingerprints for speech signals.

1.4 Organization of thesis

The rest of this thesis is organized into four chapters as follow:

Chapter two begins with a review of audio fingerprint and current techniques that are used to create audio fingerprints. Then the problems of applying audio fingerprinting technique to speech will be discussed.

Chapter three elaborates the method that is studied in this research. More specifically, the technique that is used to create representation models for speech signals is illustrated

first. In the next section of this chapter, the method that is used to analyzed features of the representation models is described.

Chapter four reports and compares the results obtain by applying the method described in chapter three. Finally, the remaining issues and future works is described in chapter five.

Chapter 2

Literature review

2.1 Audio fingerprints

Audio fingerprinting is the process of analyzing audio signals to obtain their unique features, then combining these unique features to create compact representatives for the audio signals [1]. Unlike audio watermarking, which is the process of embeding metadata into audio signals, audio fingerprinting processes the waveforms of audio signals directly to capture their inherent characteristics so as to produce distinguishable information.

In the digital era, when music, speech, and videos are digitalized, uploaded, and transferred in the Internet, audio fingerprints play a significant role in the content identification tasks as media can be quickly identified. According to a report [2], illegal usage of digitalized media have dealt serious damages to our economy; content creators can use audio fingerprints to prevent unauthorized broadcasting of their assets. Applications can be developed to help people identify music quickly using audio fingerprinting techniques; this efficiency can boost productivities and convenience in our lives.

Audio fingerprinting is a very complicated technique. To achieve the effectiveness and efficiency that fit for the demands of content-identification applications, the technique must be accurate and reliable enough to distinguish one audio signal from others. It has to be fast in response to timing requirements. In reality, we do not usually compare two original audio signals together; therefore, the fingerprints must be durable against attacks such as distortions and signals shifting. In addition, audio fingerprints have to be small in size storage requirements and searching performance [3].

Figure 2.1 illustrates a simple application of audio fingerprinting technique that can be used for identifying unknown audio signals. The application can be divided into two processes. The main objective of the first process is creating a database of audio fingerprints. In this process, a fingerprints extraction unit analyzes audio signals to create their audio fingerprints and these fingerprints are kept in a database for future references. The main purpose of the second process is identifying unknown audio signals. In this process, the fingerprints extraction unit analyzes the unknown audio signals to determine their fingerprints, then these unknown fingerprints are looked up in the database of audio fingerprints by a search algorithm. Once their matches are found, the unknown audio

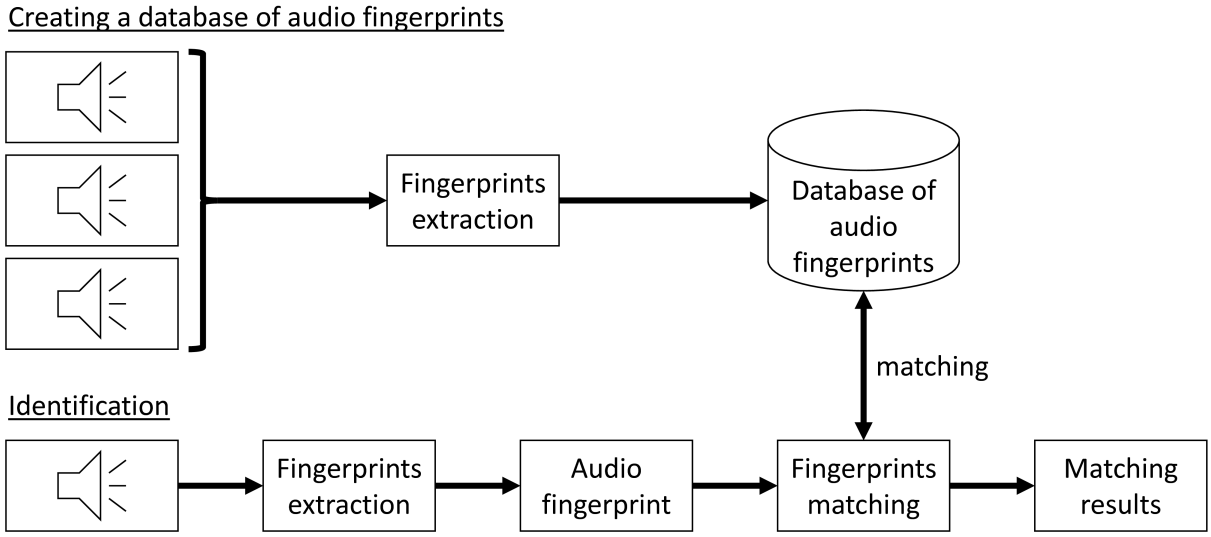

Figure 2.1: A simple application of audio fingerprints used for content identification.

signals can be identified and their additional information can be retrieved. Among the various stages, the fingerprints extraction unit is the heart of this application and will be discussed carefully in the following sections.

2.2 Typical method used for creating audio fingerprints

In the attempts to derive the fingerprints of audio signals, many researches and different approaches have been conducted and proposed by researchers [4-7]. In spite of their ingenious ideas, their methods appear to consist of three stages as depicted in Fig. 2.2. In the first stage, an encoding technique is applied to transform raw audio waveforms into their representation models. In the second stage, a feature extraction module is used to analyze and extract unique features of the audio waveforms from their representation models. In the last stage, a hashing technique is used to pack the extracted features to create fingerprints for the input audio signals.

The main idea of stage one is converting a raw input waveform into a set of features that represents acoustic events and block-based coding is commonly used for this purpose. This method assumes that for a period of a few milliseconds, the audio signal is unchanged. For this reason, a windowing technique is usually applied to divide the signal into frames. Then linear transformation techniques, namely fast Fourier transform (FFT), discrete cosine transform (DCT), and wavelet transformation, converts the frames into a set of acoustic features. The output of this stage is a representation model of the input signal, commonly known as a spectrogram. One important concern when using windowing technique is that there is a trade of between loss of information and computational complexity.

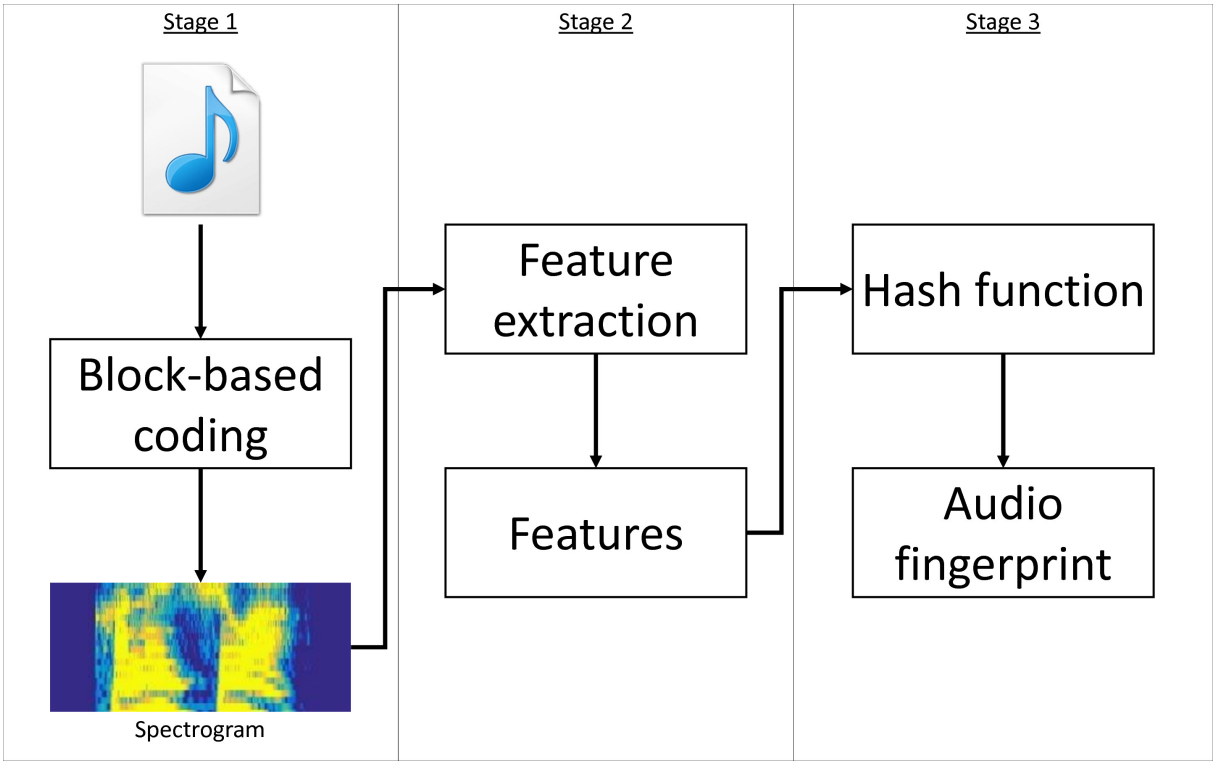

Figure 2.2: The process of creating audio fingerprints. The main task of stage one is creating a representation model for the input signal. The main task of stage two is extracting features of the input signal from its representation model. In stage three, a hash function is used to combine the extracted features from stage two to create a fingerprint for the input signal.

The main purpose of stage two is extracting features of the representation model that convey inherent information of the input audio signal. The extracted features of one signal should be distinguishable with features of other signals and resilient to attacks such as distortions and signal shiftings. For this purpose, a handful of techniques have been proposed for different rationales. Some researches utilized linear prediction coefficients (LPCs), linear prediction cepstral coefficients (LPCCs), or Mel-frequency cepstral coefficients (MFCCs) to represent the spectral envelope. Some were inspired by the vibrations of the basilar membrane and used signs of energy to simulate its fluctuations. Other researchers used only maximum energies or peaks of the spectrograms with the justification that lower energies are unlikely to survive noise.

Creating fingerprints for audio signals from the extracted features is the main focus of stage three. The features obtained from stage two, which contain unique characteristics of the audio signals, can be regarded as fundamental fingerprints and the hash function of stage thee continues to increase the discrimination power of the fingerprints and to reduce the dimension of the fingerprints for searching efficiency. Templates of audio fingerprints vary depending on the hash function. They can be vectors or matrices of binary or decimal numbers. Some researchers compared values of the extracted features to create

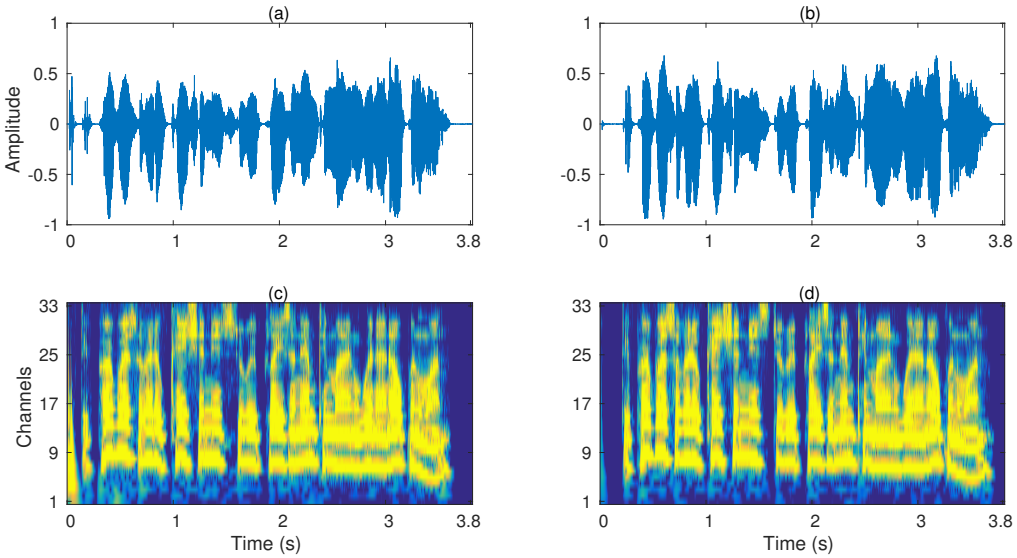

Figure 2.3: Two Speech signals and their corresponding spectrograms. (a) and (b) are speech signals of the same content produced by a same speaker. (c) and (d) are spectrograms corresponding to (a) and (b).

binary encoded fingerprints, others created their final fingerprints by applying pattern recognition methods to capture the relationships of the features [7-12].

2.3 Problems of audio fingerprints in speech

In case of music, the methods mentioned above achieved admirable results but the characteristics of speech have exposed their weaknesses. Block-based coding technique divides a speech signal into blocks having similar arbitrary widths and processes these blocks separately using discrete Fourier transform (DFT) or discrete cosine transform (DCT). This technique is substantially sensitive to signal shifting and phoneme scaling; randomly blocking speech signals does not take into account the alignment of acoustic cues. Given a speech content, it is unlikely for a speaker to produce signals having the same length and phonemes having the same duration. Figure 2.3 shows two speech signals that have the same content and are produced by the same speaker and their corresponding spectrograms. Perceptually, these two speech signals may sound the same but techniquely, the alignments and durations of their acoustic features are considerably different. Therefore, block-based coding will create different spectrograms for speech signals that have the same content and same speaker. One way to overcome the drawback of block-based coding is using filterbank based shift invariant coding but this technique greatly increase the dimension of speech signals because of its convolutional calculations. Data redundancy on the representation models makes it difficult to recognize the underlying structures of speech signals [13].

The sources of problems do not confine only in the signal representation process, they also appear in the features extraction methods. To gather features for the production of speech fingerprints, several methods have been proposed; one used the signs of energies, another used the peaks of spectrogram. Although their methods achieved great experimental results, none of them take into account patterns of the features on the representation model. Given a pair speech signals that has the same content but different speakers and another pair that has the same speaker but different contents, a feature extraction method should realize the similarities and differences of the features on the representation model to improve the quality of speech fingerprints.

An experiment was carried out to evaluate the ability of block-based coding in representing speech signals. A Gammatone filterbank [14] was used in this experiment to process ten speech signals with the same content produced by two speakers. Although this is one of the most well-known techniques used in speech processing, testing the distortion between the original signals and the resynthesized signals could only achieve mean $\mu = 18.9$ dB and standard deviation $\sigma = 1.2$ dB in the signal to noise ratio (SNR) (ranges of SNR are: below 25 dB is low, from 25 dB to 40 dB is high, and above 40 dB is excellent), and mean $\mu = 3.9$ and standard deviation $\sigma = 0.1$ in the perceptual evaluation of speech quality (PESQ)(PESQ scores are: 1 is bad, 2 is poor, 3 is fair, 4 is good, and 5 is excellent).

Based on the work by Ellis [15], another experiment was also conducted to investigate the effectiveness of spectrograms in the process of producing speech fingerprints. The results obtained from this experiment by using the previously described data proved that different fingerprints were generated for speech signals that had the same content and the same speaker despite the fact that this method worked very well with audio fingerprints.

Chapter 3

Method for creating speech fingerprint

3.1 Overall process of creating speech fingerprint

The process used for creating speech fingerprint in this research consists of three stages as depicted in Fig. 3.1. In stage one, sparse coding technique was used to create over-complete representation models for speech signals. To do so, a matching pursuit algorithm and a base spectrum were utilized to generate the desired spikegrams.

In stage two, we aimed to extract the features of the spikegrams that could be related to linguistic content and speaker individuality. At the current stage of this research, we used local binary pattern (LBP) and histogram to analyze texture of our spikegrams.

The purpose of stage three was using a hash function to combine the extracted features to create final fingerprints for speech signals. Details about the configurations of this method and experimental results will be discussed in the following sections.

3.2 First stage – creating spikegram

3.2.1 Overview of sparse coding and spikegrams

Our brains have to process a huge amount of sensory data constantly throughout our lives. We continuously receive visual images from our eyes, sounds from our ears, smells from our noses, tastes from our tongues, and environment temperatures from our skin. The abilities of our brains in processing such enormous amount of stimuli and making senses of them are still mysteries to science. Many researches have been conducted to understand how our brains work and to create machines that are able to mimic the systems that have been developing over millions of years of evolution. Results from these studies show that our brains dedicate a very large number of neurons for processing sensory stimuli but only a small amount of active neurons are used to represent sensory information; such mechanism is called sparse coding [16].

Due to the nature of speech, it is difficult for a speaker to produce the same speech

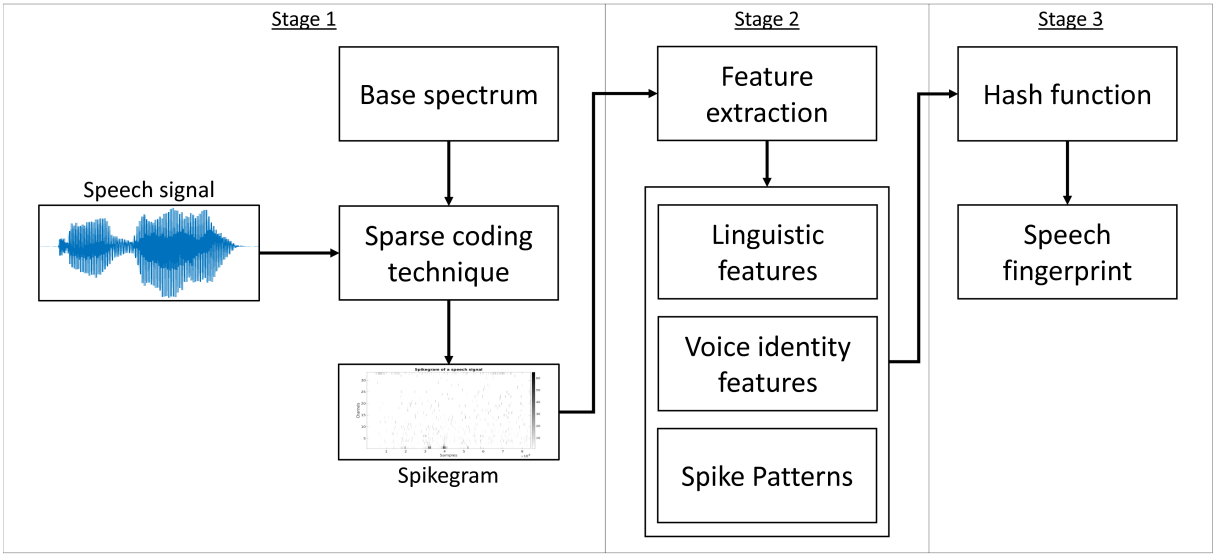

Figure 3.1: Process of creating speech fingerprint

signals that have the same content yet we can recongnize the words that are spoken and distinguish the voice of one speaker from others'. This natural ability of humans is still challenging to speech processing. One explaination for this phenomenon is that acoustical events in speech have relationships in times and frequencies and human hearing system has evolved to understand these kinds of relationships. A part from these relationships, acoustical events vary dramatically in time location, duration, and pitch. Therefore, representing important acoustical events and their relationships is key to improve quality of a speech processing system.

Inspired by the idea, many researchers have tried to represent speech signals using an over-complete representation model, known as spikegrams, in which the number of non-zero elements are significantly smaller than the number of zero elements. Conventional methods used for representing speech signals such as block-based coding or filterbank based shift invariance coding are unable to express the structures of speech signals and the goal of sparse coding is to capture such structures. Sparse coding is a more efficient method for encoding the statistics of speech signals as it is a data-driven technique that can adapt to the variations of speech. Results from several researches showed that using spikegram is better than using spectrogram for representing speech signals [16-20].

Karklin et al. (2012) used spikegram to build a two-layer model to represent speech signals. In the first layer, a spikegram was used to capture a fine structure of an input speech signal. The second layer further reduced the number of spikes to represent the most important acoustic events of the speech signal. Then the output of the second layer was used to resynthesized the speech signal. Testing on TIMIT dataset, results showed that this method had significant improvement over the traditional methods that were based on spectrograms.

Siahpoush et al. (2015) studied different encoding methods to improve the quality of auditory implants. This research first encoded input sound signals with different repre-

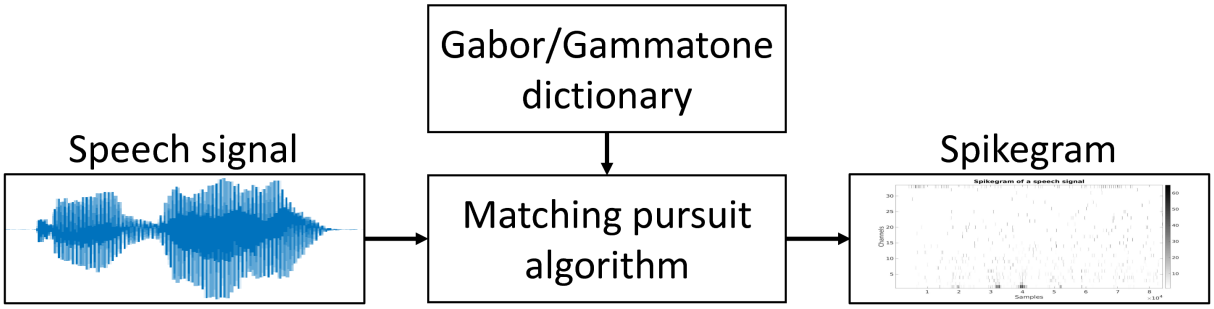

Figure 3.2: Process of creating spikegrams using matching pursuit algorithm and Gabor/Gammatone dictionary.

sentation models such as spectrogram, representation model from Gammatone filterbank, and spikegram. Then Bayesian neural decoding technique was used to reconstruct the sound signals using these kinds of representation models. The distortion between the original signals and the resynthesized signals was measured by signal to noise ratio to evaluate the effectiveness of the decoding technique. Results showed that encoding input signals using spikegrams resulted in highest decoding accuracy.

Erfani et al. (2017) conducted a research about employing spikegram to create a new audio watermarking technique. This research utilized a perceptual matching pursuit algorithm and two Gammatone dictionaries – Gammasines and Gammacosines – to create spikegrams for audio signals. Experimental results showed that this new method produced better performance than other wartermarking techniques.

3.2.2 Process of creating spikegrams

Previous studies have shown that sparse coding outperforms other the block-based coding techniques in decomposing speech signals. Therefore, this research exploits the abilities of a matching pursuit algorithm [21, 22] and two kinds of base spectra – Gabor dictionary and Gammatone dictionary – in representing speech signals as equivalent spikegrams. Results of using each kind of dictionaries will be shown and compare later in this thesis.

The matching pursuit algorithm we use is mathematically based on non-negative matrix factorization (see appendix for more information) that approximates one matrix with two other non-negative matrices, one is a matrix of bases and the other is a matrix of coefficients. Therefore, the quality of the approximation depends on the selection of appropriate bases and the method of finding the coefficients.

Figure 3.2 shows the process of decomposing a speech signal. For a given speech signal, the matching pursuit algorithm uses a Gabor dictionary or a Gammatone dictionary to calculate a coefficients matrix that best approximates the input signal. The resulting coefficients matrix is known as a spikegram.

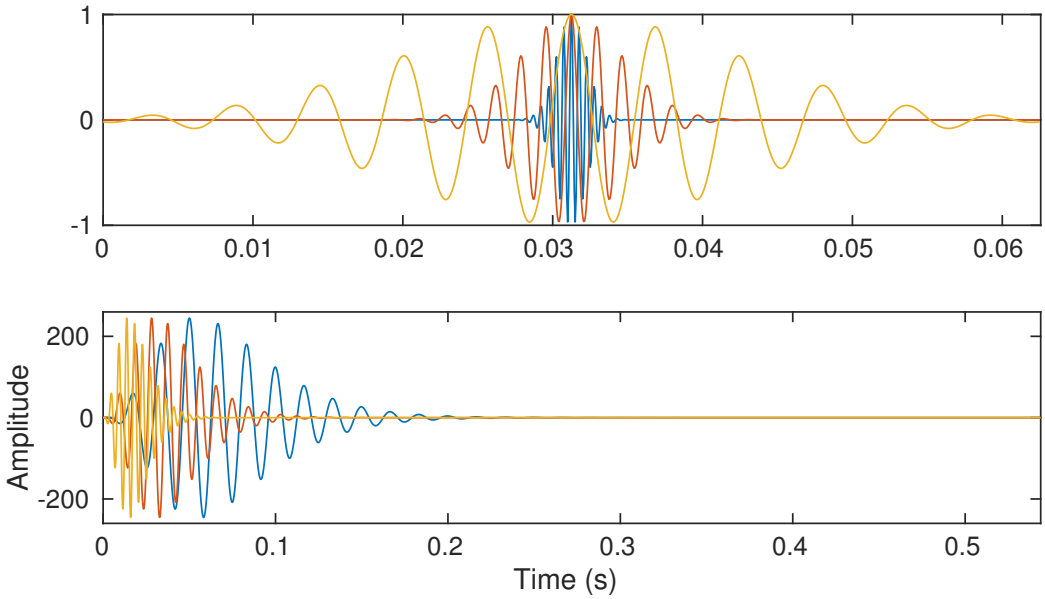

Figure 3.3: Gabor and Gammatone dictionaries. (a) illustrates three kernels of Gabor dictionary and (b) illustrates 3 kernels of Gammatone dictionary.

3.2.3 Base spectrum

Base spectrum is a spectrum of a basis by which a signal can be represented. Namely, in a specific basis, we can express the signal by a unique set of coefficients expanded in that basis. In most vector spaces, the basis is usually orthogonal and linearly independent. However, depending on which properties of the signal we would obtain from its representation in the basis, these constraints of orthogonality and linear dependence alternatively contribute their effects.

A basis can characterize a sparse representation of a signal, which depends on a matching between the properties of the signal and that of the basis. For instance, in an orthogonal basis as Fourier basis $w^{(k)}$, for which

$$w_n^{(k)} = e^{j\frac{2\pi}{nk}}, \tag{3.1}$$

a continuous signal can be sparsely represented, but an impulse cannot. It is realized that on a singular basis, it is difficult to represent sparsity of signals in real situations, besides the speech signal is not an exception. More than one basis are expected to be involved into the representation. The collection of vectors selected from these bases must span the space, which is called a dictionary of vectors. As a result, the vectors in the dictionary is not necessarily linear independent or orthogonal, however, it plays an effect. Also, the appropriate sparsity with both time-frequency is an extremely important target that spikegram is expected to possess.

Therefore, to well capture these sparse characteristics of a speech signal onto spikegram, one of the main components used in creating an over-complete representation of speech

signals is a dictionary of bases. There are two kinds of dictionaries of bases we used in this research, Gabor and Gammatone dictionaries. The impulse response of the Gabor filter can be calculated by multiplying a Gaussian function and a sinusoidal wave. The real part of the Gabor function can be calculated as follow:

$$g(x, y; \lambda, \theta, \psi, \sigma, \gamma) = \exp(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}) \cos(2\pi \frac{x'}{\lambda} + \psi), \tag{3.2}$$

where $x' = x\cos\theta + y\sin\theta, y' = -x\sin\theta + y\cos\theta$, and $\lambda, \theta, \psi, \sigma, \gamma$ are the parameters of the Gabor function. The impulse response of the Gammatone filter is given by:

$$g(t) = at^{n-1}e^{-2\pi bt}\cos(2\pi ft + \phi), \tag{3.3}$$

where a, t, n, b, f, and ϕ correspond to the amplitude, time, order of the filter, bandwidth of the filter, center frequency, and phase.

Figure 3.3 illustrates three kernels of a Gabor dictionary and a Gammatone dictionary. Details about the configurations used to create these dictionaries will be discussed later in this thesis.

3.2.4 Matching pursuit algorithm

As previously mentioned, the vectors in the dictionary are not linearly independent, consequently, it could have more than one representation of a speech signal in the dictionary. Nonetheless, with an over-complete dictionary, we can represent a speech signal in a collection of vectors that perfectly expand to its properties in both time and frequency domain, underlyingly a spikegram. Moreover, we know that it could have several ways to represent the speech signal in the dictionary. The crucial problem is how to find out the best representation. In a mean time, it led a solution of matching pursuit algorithm.

Matching pursuit algorithm was introduced by Mallat and Zhang in 1993 [23]. It identifies the best sparse representation of a signal in a complete or over-complete set of dictionary by performing a nonlinear approximation in the dictionary. Assuming b_i denote the kernels of a dictionary. In this study, the dictionary is over-complete, no single way to express the speech signal in a linear combination of the kernels. If x is the speech signal, an ideal expectation is

$$x = \sum_i w_i b_i, \tag{3.4}$$

To find out the best representation, it is find the collection of b_i producing the biggest absolute inner products with the signal. In another word, finding the best b_i by following the equation

$$\max_i | < x, b_i > | \tag{3.5}$$

for which an optimal kernel is the projection onto the subspace spanned by b_i that yields the largest scalar projection.

By the interpretation above, it implies that matching pursuit locates the optimal K-term expansion of the speech signal in the dictionary. A basic algorithm of matching pursuit for time-shiftable kernels is explained as follows.

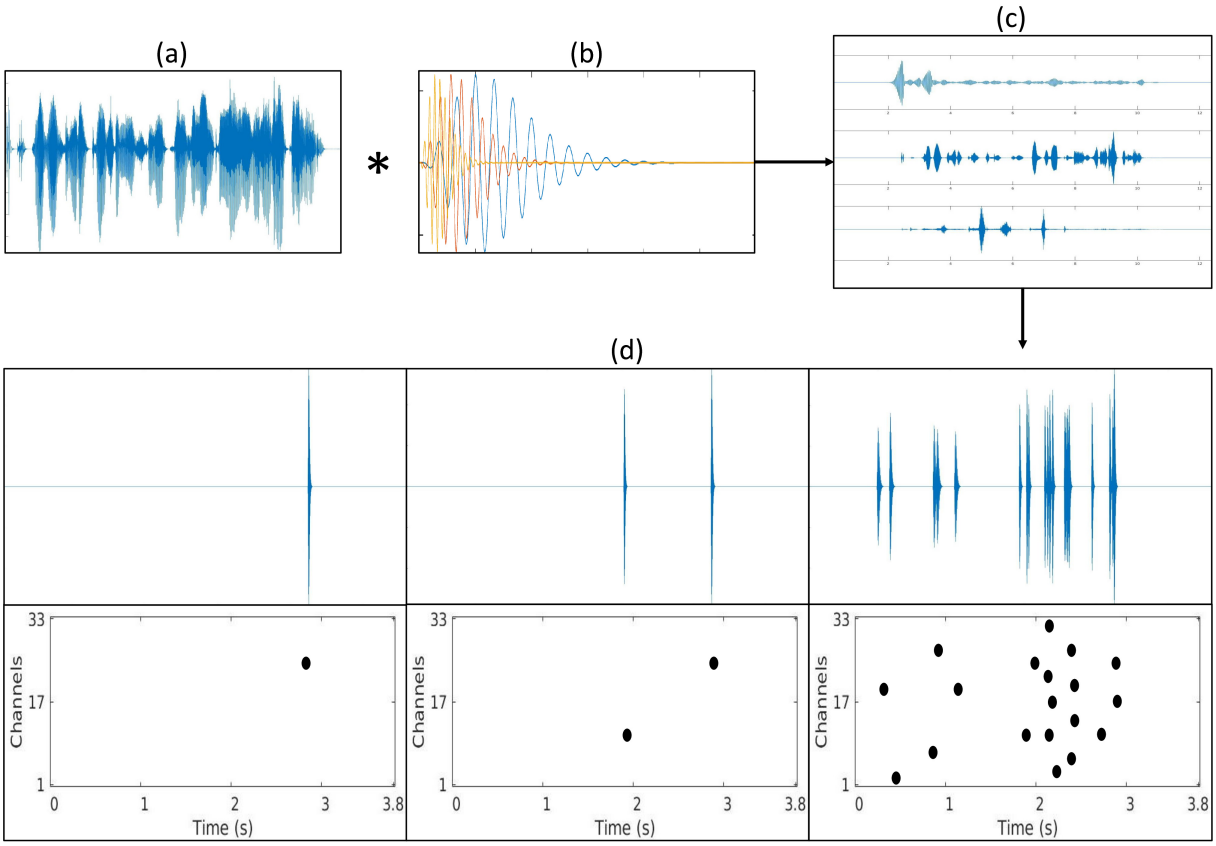

Figure 3.4: Computation steps of matching pursuit algorithm. (a) is a speech signal, (b) is a base spectrum, (c) is the convolution set created by convolving the speech signal and the base spectrum, (d) illustrate the reconstructed signal and spikegram after one, two, and twenty iterations.

- An input signal in the matching pursuit technique can be decomposed into a linear combination:

$$x = \sum_{i}^{K} w_i b_i + r, \tag{3.6}$$

where x is the signal vector, K is the number of iterations, $w_i \in W$ and $b_i \in B$ are the weight and kernel at the i-th iteration, and r is the residual.

- According to the equation Eq. 3.6, minimizing the residual will maximize the approximation of the input signal and its linear combination. Therefore, the matching pursuit algorithm performs a search at each iteration for the largest inner product between the residual and a kernel, i.e.:

$$w_i = \arg\max | < r_i, b_i > |^2, \tag{3.7}$$

where $< r_i, b_i >$ is the inner product between the residual, r_i, and a kernel, b_i, at

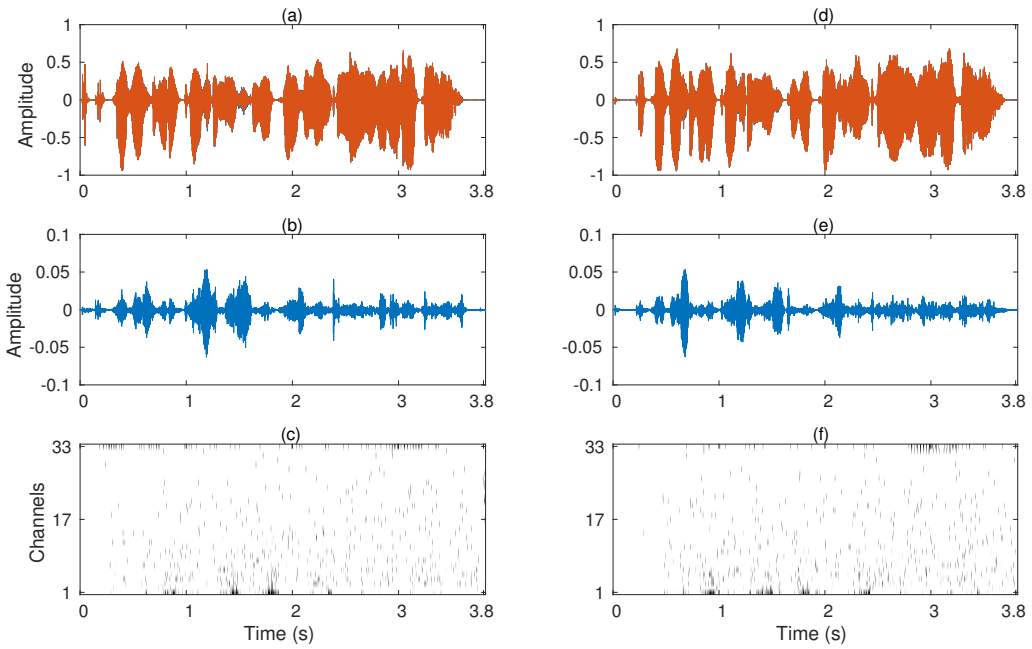

Figure 3.5: Speech signal representation using spikegram. (a) and (d) represent superimposition of the speech signals shown in Fig. 2.3 and their approximations. (b) and (e) are the residual signals after the approximations. (c) and (f) are the spikegrams of the speech signals created using a matching pursuit algorithm and Gammatone dictionary.

the i-th iteration. The residual is then updated as:

$$r_{i+1} = r_i - w_i b_i, \tag{3.8}$$

3.2.5 Creating spikegrams

This research employed a matching pursuit algorithm [7] by using the dictionary of Gammatone kernels as described in the previous section and some computational improvements [8] to create equivalent spikegrams for speech signals. The six overall computation steps are below.

1. There are two most important input arguments of MP, they are input signal x and dictionary of Gammatone kernels B. The MP performs calculation based on these inputs and produce resynthesized signal y and spikegram W.

2. In the initialization step, the MP pre-calculates the correlations between the signal x and each kernel of the Gammatone dictionary B to create a correlation set C. More specifically, all kernels are first inversed by using $B(:, end : -1 : 1)$, then the correlation set is calculated by using $C = conv2(B(:, end : -1 : 1), x)$. The resynthesized signal is set to be a zero vector with the length of the input signal

$y = zeros(1, length(x))$ and the spikegram is set to be a zero matrix with the size of the correlation set $W = zeros(size(C))$.

3. In step one of the main loop, the MP uses the max function to find the maximum value of the correlation set C and to obtain the position of the maximum value denoted as $(maxi, maxj)$.

4. In step two, the spikegram and the resynthesized signal are updated. More specifically, a spike is recored by using $W(maxi, maxj) = W(maxi, maxj)+C(maxi, maxj)$. To update the resynthesized signal, multiplication between the coefficient and the kernel that are corresponding to the maximum value are first calculated as $delta = C(maxi, maxj) \times B(maxi, :)$, then the resynthesized signal is appended with delta as $y = y + delta$.

5. In step three, the correlation set C is updated corresponding to the current maximum value. First, MP calculates the multiplication of the current coefficient and the current kernel corresponding to the maximum value $delta = C(maxi, maxj) \times B(maxi, :)$. Then with the current position $(maxi, maxj)$, a part of the correlation set is subtracted by $delta$. After step 2, the resynthesized signal y is updated; the calculation in step three is performed to reflect the correlation between the residual $(x - y)$ and kernels of the Gammatone dictionary B.

6. Step one of the main loop is repeated until stoping criteria are met.

The matching pursuit algorithm implemented in this research used two stopping criteria; the first was when the SNR reached 40 dB and the second was the total number of iterations.

The top panels in Fig. 3.5 illustrate the original signals and the reconstructed signals from the dictionary of Gammatone kernels B and weight matrix W of two speech utterances. The residuals of the approximation process are depicted in the middle panels. The bottom panels illustrate the spikegrams of these speech utterances.

An experiment was conducted by using the same data as those presented in Section 2 to evaluate the capabilities of the matching pursuit algorithm and Gammatone dictionary in decomposing speech signals. This method achieved mean $\mu = 40.0$ dB and standard deviation $\sigma = 0.0$ dB in SNR, and mean $\mu = 4.3$ and standard deviation $\sigma = 0.2$ in PESQ when the distortion between the original signals and the resynthesized signals were tested.

3.3 Second stage – feature extraction

Local binary pattern is a technique that is widely used in texture analysis [24] and image processing. It was first proposed by Ojala et al. [25] in 1994 as a method used for image classification. In this work, an image was divided into 3×3 blocks, then eight values of the outer ring of each block were compared with the center value of the block. If a value was greater than or equals to the center value, it was encoded with one, and zero otherwise.

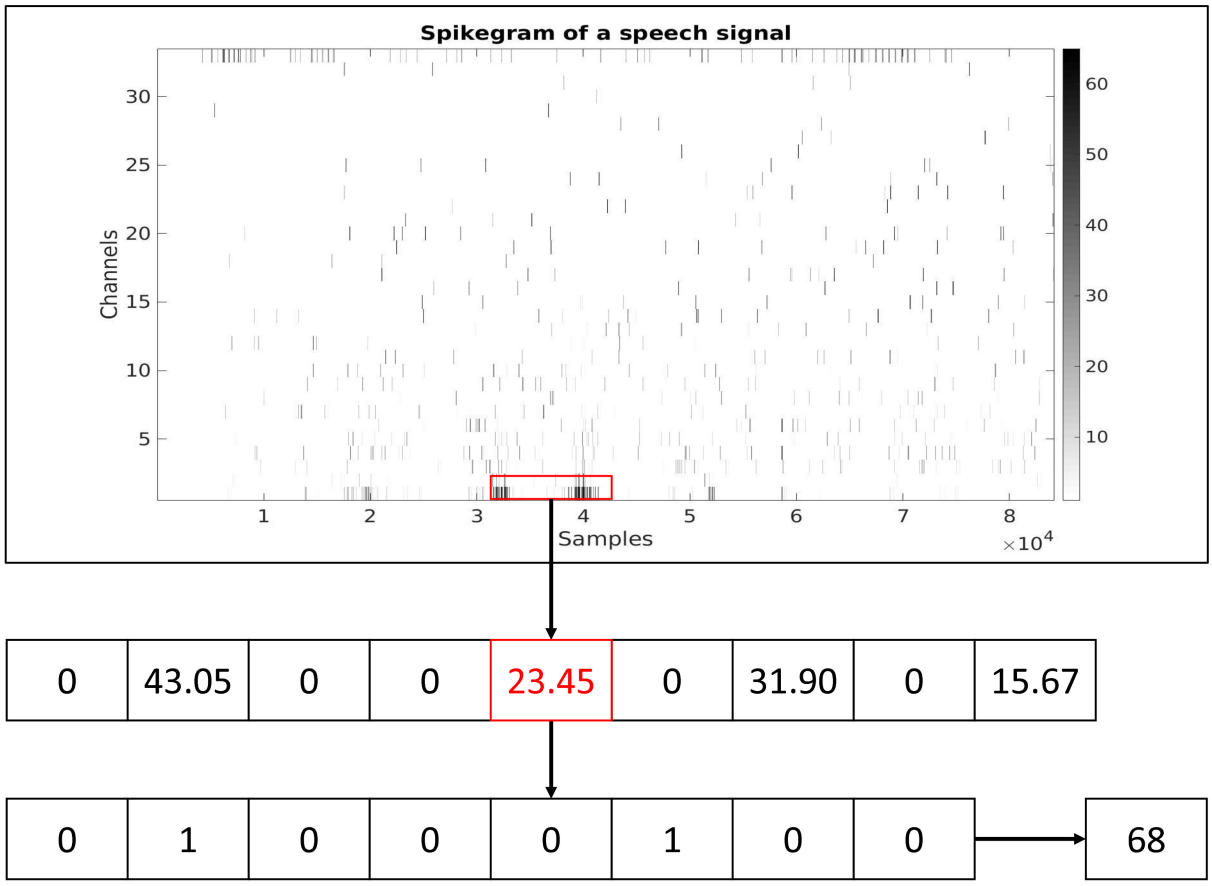

Figure 3.6: Process of converting a spikegram into a LBP matrix. Each spike on the spikegram is selected as the center value along with eight other spikes. Values of the neighboring spikes are compare with the center value to create an eight-bit binary string. Then, the binary string is converted into a decimal value. Finally, the center value is replaced with this decimal value. After this process complete, the spikegram is transformed into a LBP matrix with values ranging from 0 to 255.

After this encoding process, this technique converted pixels an image into a binary matrix and revealed the patterns of the image which was useful for classification tasks. An image classification experiment was conducted to evaluate and compare performances of this technique along with others using various dataset. Results showed that LBP outperfomed other techniques as it achieved lowest error rate.

Over the course of development, LBP was expanded with many variations and applications [26]. The variations of LBP have been proposed for many purposes; some were used for improving image classifying results, others were used for choosing LBP blocks, applying to 3D images, and combining with other methods to create hybrid approaches. One of the most noticeable applications of LBP is face recognition. This application can be summarized in two processes that are representing facial features and classifying them. The facial features representation model plays an important role in the overall quality of

a face recognition application. Therefore, when building a representation model for facial features, one should bare in mind some significant requirements. One of the requirements is that, given images of the same person, features on their representation models should be similar; meanwhile, they should be considerably different for images of different people. Another requirement is that the feature extraction method should be effective for practical application that means the features of raw images should be easy to obtain to avoid computational complexity. Another requirement is that the dimensionality of the representation model should be low so that the classification process could work effectively. LBP happens to satisfy all of the aforementioned requirements. By applying LBP, many researchers, who worked with face recognition problem, achieved astonishing benchmarks in FERET database [27-30].

While analyzing features on spikegrams, there is one obstacle that we need to deal with is the unalignment of the features. Recorded signals have different lengths, they are unaligned, and their phonemes are scaled; as the result, their features on spikegrams are difficult to compare. Further more, the features that represent linguistic content and speaker individuality are encoded in the spike distributions. Therefore, it is necessary to convert spikegrams into a nomalized dimension to compare their features. Considering the robustness of of LBP in representing facial features that can be used for face recognition, this research has tried to utilize the advantages of LBP to analyze the features of spikegrams. One characteristics of LBP that makes it become a candidate for analyzing features of spikegrams is its robustness in texture analysis. Our assumption is that spike distributions of speech signals that have the same content and same speaker should be similar and changes in speech content and or speakers should result in different spike distributions. Therefore, a variation of LBP was applied to this research as shown in Fig. 3.6. Each value on a spikegram is chosen as a center value and eight other values, i.e., four preceding and four trailing the center value, are also selected to calculate the LBP. After this process, each value of a spikegram is converted to a decimal number ranging from zero to 255 and a spikegram is transformed into an LBP matrix. Moreover, histogram of the LBP matrix is calculated to evaluate the texture of the spikegram. Results and evaluations will be discussed later in this thesis.

3.4 Third stage – hash function

Based on the output of the second stage, the hashing technique of this stage further combines the extracted features to create final fingerprints for speech signals. Many researchers have proposed different hashing techniques to create speech fingerprints and designs of the techniques are driven by some requirements

- Speech fingerprints should have strong discrimination power. It means that speech fingerprints created from the speech signals that have the same content and same speaker should be similar together to be considered as the same. On the other hand, speech fingerprints created from speech signals that have different content and or speakers should be considerably different.

- Speech fingerprints should be small in size because there is an enormous amount of speech content that are needed to be hashed. Therefore, the size of speech fingerprints should be as compact as possible to reduce the size of the fingerprint database.

- A direct consequence can be drawn from the above requirement is searching efficiency. A database of fingerprints may contain hundreds of millions of hash sequences, therefore, fingerprints matching should be fast enough to be applicable to practical applications.

Salem et al. (2009) [10] proposed a robust audio hashing system that simulated the vibrations of the basilar membrane and hair cells of our hearing system. In this research, the researchers divided speech signals into frames and applied an energy function to calculate the energies for all the frames. The matrix of energies created by this method was regarded as representation models for the speech signals. Then, two adjacent values on the representation model were selected and subtracted together. If the subtraction result was greater than zero, one was recorded and zero otherwise. After this process, the speech signals were converted into hash strings. Given two hash strings, their bit error rate was calculated and compared with a threshold of 0.25. If the bit error rate was less than threshold, the original two speech signals was regarded as the same and different otherwise. Experimental results showed that this method worked very well against audio attacks and achieved high accuracy.

Chen et al. (2010) [8] first calculated LPCs a speech signal, then the researchers applied NNF to the LPCs to create a features vector for the speech signal, then each value of the features vector was compared with the average value of the entire feature vector. If the value was greater than or equal the average value, one was recorded and zero otherwise. Finally, fingerprint created for the speech signal was a binary vector. The researchers further calculated a threshold that was used distinguishing their speech fingerprints. To compare two speech fingerprints, they calculated their Euclidean distance. If the distance was less than the threshold, the two speech signals were considered as the same and different otherwise. This method was tested with 1000 speech signals and its accuracy was admirable.

Chapter 4

Result and evaluation

4.1 Speech signals representation

This section describes an experiment conducted to evaluate and compare the abilities of representing speech signals of block-based coding and sparse coding. The dataset used for this experiment consists of 75 speech signals of three different types of content including /ohayo-gozaimasu/, /konnichiwa/, and /konbanwa/ produced by five speakers that included three males and two females (each speaker spoke each content five times). A Gammatone filterbank, the representative for block-based coding, was used to create spectrograms for speech signals. To create spikegrams, the matching pursuit algorithm described in the above section and two kinds of dictionaries, namely Gabor dictionary and Gammatone dictionary, was used for the sparse coding technique. Details of the configurations and parameters used for the coding techniques and the results are described in the following section.

4.1.1 Gammatone filterbank spectrograms

In this experiment, a Gammatone filterbank was used to create Gammatone spectrograms. As shown in Table 4.1, the Gammatone filterbank used in this experiment was configured to have 129 channels, the center frequency was 600 Hz, and the frequency of the input speech signals was $16,000$ Hz.

To evaluate the effectiveness of these configurations, the distortions between the original speech signals and the resynthesized speech signals was calculated using SNR and PESQ. Table 4.2 shows the effectiveness results of this method. It can achieve mean $\mu = 19$ dB, standard deviation $\sigma = 1$ dB in SNR and mean $\mu = 3.6$, standard deviation $\sigma = 0.2$ in PESQ.

4.1.2 Gabor spikegrams

Based on the work of Patrick Mineault et al. (2011), a Gabor dictionary was used for the maching pursuit algorithm to create Gabor spikegrams. As shown in Table 4.1, the Gabor dictionary used in this experiment was configured to have 128 kernels, depicted in

Table 4.1: These tables show the configurations of the methods used in this experiment. Parameters of the methods using Gammatone filterbank, Gabor spikegram, and Gammatone spikegram are shown in the top, middle, and bottom tables, respectively.

Parameters of Gammatone filterbank	**Value**
Number of channels	129
Center frequency	600 Hz
Sampling frequency of input signals	16000 Hz

Parameters of Gabor dictionary	**Value**
Number of kernels	128
Sampling frequency of input signals	16000 Hz

Parameters of Gammatone dictionary	**Value**
Number of kernels	33
Center frequency	600 Hz
Sampling frequency of input signals	16000 Hz

Fig. 4.1(b), and the frequency of the input speech signals was $16,000$ Hz. The matching pursuit algorithm was configured to have 2 stopping criteria, one is when the residual signal equals to zero, and another is the maximum iteration limit $(100,000)$.

To evaluate the effectiveness of these configurations, the distortions between the original speech signals and the resynthesized speech signals was calculated using SNR and PESQ. Table 4.2 shows the effectiveness results of this method. It can achieve mean $\mu = 19$ dB, standard deviation $\sigma = 2$ dB in SNR and mean $\mu = 4.4$, standard deviation $\sigma = 0.0$ in PESQ.

4.1.3 Gammatone spikegrams

In this experiment, a Gammatone dictionary was used for the maching pursuit algorithm to create Gammatone spikegrams. As shown in Table 4.1, the Gammatone dictionary used in this experiment was configured to have 33 kernels, depicted in Fig. 4.2(b), the center frequency was 600 Hz, and the frequency of the input speech signals was $16,000$ Hz. The matching pursuit algorithm was configured to have 2 stopping criteria, one is the SNR limit (40 dB), and another is the maximum iteration limit $(500,000)$.

To evaluate the effectiveness of these configurations, the distortions between the original speech signals and the resynthesized speech signals was calculated using SNR and PESQ. Table 4.2 shows the effectiveness results of this method. It can achieve mean $\mu = 39$ dB, standard deviation $\sigma = 1$ dB in SNR and mean $\mu = 4.4$, standard deviation $\sigma = 0.1$ in

Table 4.2: These table shows results of measuring the distortions between the original speech signals and the resynthesized speech signals – created by the methods in this experiment – using SNR and PESQ. Results of the methods using Gammatone filterbank, Gabor spikegram, and Gammatone spikegram are shown in the top, middle, and bottom tables, respectively.

Results of using Gammatone filterbank		
Mesurement	**Mean**	**Standard Deviation**
SNR	$\mu = 19$ dB	$\sigma = 1$ dB
PESQ	$\mu = 3.6$	$\sigma = 0.2$

Results of using Gabor spikegram		
Mesurement	**Mean**	**Standard Deviation**
SNR	$\mu = 30$ dB	$\sigma = 7$ dB
PESQ	$\mu = 4.4$	$\sigma = 0.0$

Results of using Gammatone spikegram		
Mesurement	**Mean**	**Standard Deviation**
SNR	$\mu = 39$ dB	$\sigma = 1$ dB
PESQ	$\mu = 4.4$	$\sigma = 0.1$

PESQ.

4.2 Features of spikegrams

A variation of local binary pattern was employed for this process in the current stage of this research. One characteristic of LBP is that it is able to demonstrate local changes in spike distributions that are important in comparing changes in content and utterers of speech. Each value on a spikegram is chosen as a center value and eight other values, i.e., four preceding and four trailing the center value, are also selected to calculate the LBP. After this process, each value of a spikegram is converted to a decimal number ranging from zero to 255 and a spikegram is transformed into an LBP matrix as shown in Fig. 4.3. Then, histograms for all channels of the LBP matrix are calculated to form a (33×255) histogram matrix, as shown in Fig. 4.4 and Fig. 4.5.

As spikegrams are calculated using a sparse coding technique, a spike in the middle of an ocean of zeros is very common. The stripes in Fig. 4.4 are the results of this pattern; more specifically, they are positioned at $2^0, 2^1, 2^2, 2^3, 2^4, 2^5, 2^6$, and 2^7. Figures 4.4(b) and 4.4(c) are histogram matrices of two speech signals of the same content produced by

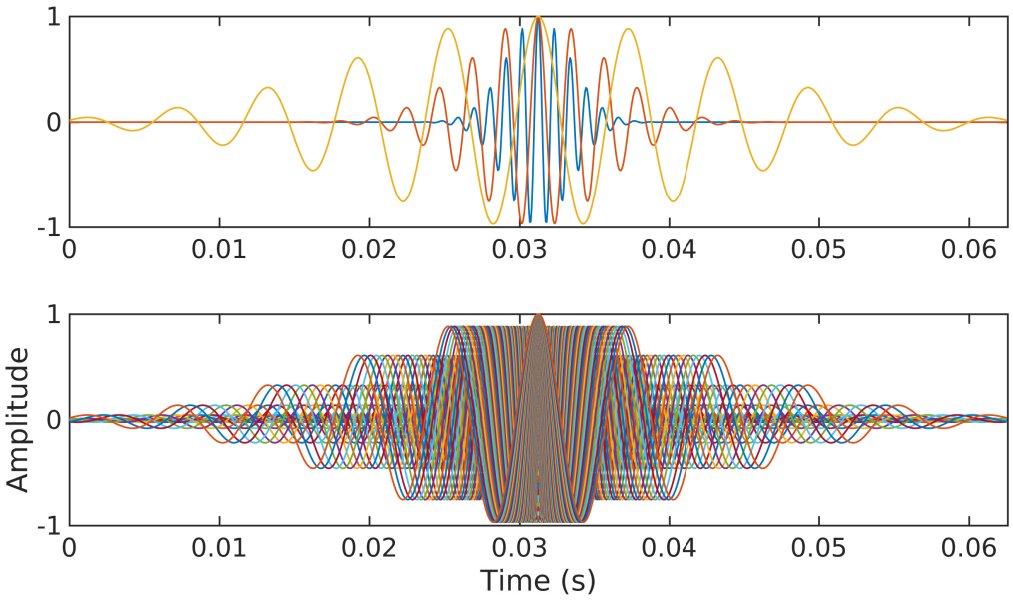

Figure 4.1: Gabor dictionary used for creating Gabor spikegrams. (a) shows 3 kernels of the dictionary and (b) shows full 128 kernels of the dictionary.

two different speakers. Figures 4.5(a), 4.5(b), and 4.5(c) are histogram matrices of three speech signals that are produced by the same speaker and have different content. The speech content includes /ohayo-gozaimasu/, /kon-nichiwa/, and /konbanwa/ shown in (a), (b), and (c), respectively. Color patterns of the LBP histogram matrices are different indicating that texture of the spikegrams are different; this serves as a cue in comparing histogram matrices.

An experiment was conducted to evaluate the capabilities of spikegrams in capturing the underlying structures of speech signals, where 75 speech signals of three different types of content including /ohayo-gozaimasu/, /kon-nichiwa/, and /konbanwa/ produced by five speakers that included three males and two females were recorded. Then, the histogram matrices of equivalent spikegrams of speech signals were calculated using the previously described process.

One method that was used to evaluate the differences in spike distributions involved subtracting the histogram matrices by one another. Figure 4.6(a) shows the results obtained from subtracting the histogram matrices of two speech signals that had the same spoken word produced by the same speaker. The remaining subtraction is almost flat, which indicates that the spike distributions of the original two spikegrams are very similar. Figures 4.6(b), 4.6(c), and 4.6(d) are the respective results from subtracting signals in the same manner that have the same content with different speakers, different types of content with the same speaker, and different types of content with different speakers. The color patterns of these figures indicate that changes in linguistic content or speakers result in very different spike distributions.

The Frobenius norm was another method used for comparing the histogram matrices.

Table 4.3: Comparing Frobenius norm of LBP histograms of different speech utterances to evaluate the effectiveness of spikegram.

	Mean	**Standard Deviation**
Same speaker and same content	$\mu = 408.3$	$\sigma = 20.3$
Same speaker and different content	$\mu = 3878.0$	$\sigma = 194.3$
Different speakers and same content	$\mu = 5072.6$	$\sigma = 878.8$
Different speakers and different content	$\mu = 5260.1$	$\sigma = 2747.2$

Table 4.3 summarizes the mean distances and standard deviations of the results. The mean distance and standard deviation obtained by comparing histogram matrices that have the same content and same speaker are about 10 times lower than the others. The other results demonstrate that changes in content or speakers notably increase the distances between histogram matrices and the farthest distances and largest variations of all of these are between different types of content and different speakers.

4.3 Hash function

We conducted another experiment to evaluate the discrimination ability of the LBP histogram matrices. Given a LBP histogram matrix of a speech signal, we calculated the mean for each column to obtain a (1×256) vector and temporarily called it the speech fingerprint for the speech signal. In this manner, we produced a database containing 75 speech fingerprints. Then we computed Euclidean distances for these 75 fingerprints with the number of permutation was ten. Then we calculated means and standard deviations for the Euclidean distances.

For a new speech signal, we calculated the speech fingerprint for this signal. Then we calculated the Euclidean distance between this fingerprint and other fingerprints in the database. If this distance fell into a distribution calculated earlier, we considered it as a match. Based on the experiment results, we found that there were some cases this comparing method produced wrong results but most of the cases, the querying fingerprints felt into correct distributions.

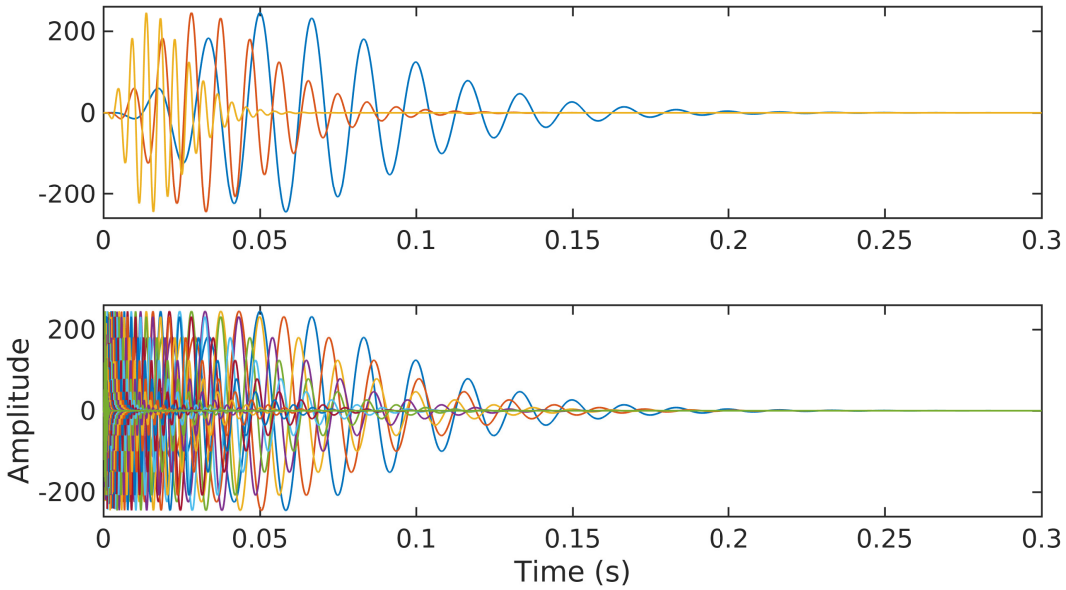

Figure 4.2: Gammatone dictionary used for creating Gammatone spikegrams. (a) shows 3 kernels of the dictionary and (b) shows full 33 kernels of the dictionary.

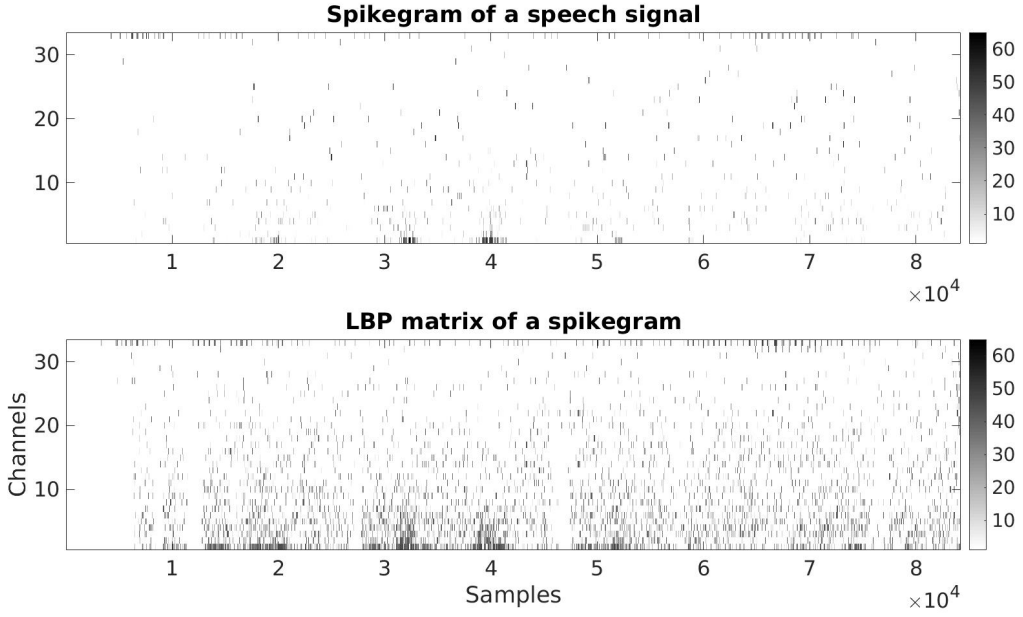

Figure 4.3: A spikegram (top panel) and its LBP matrix (bottom panel).

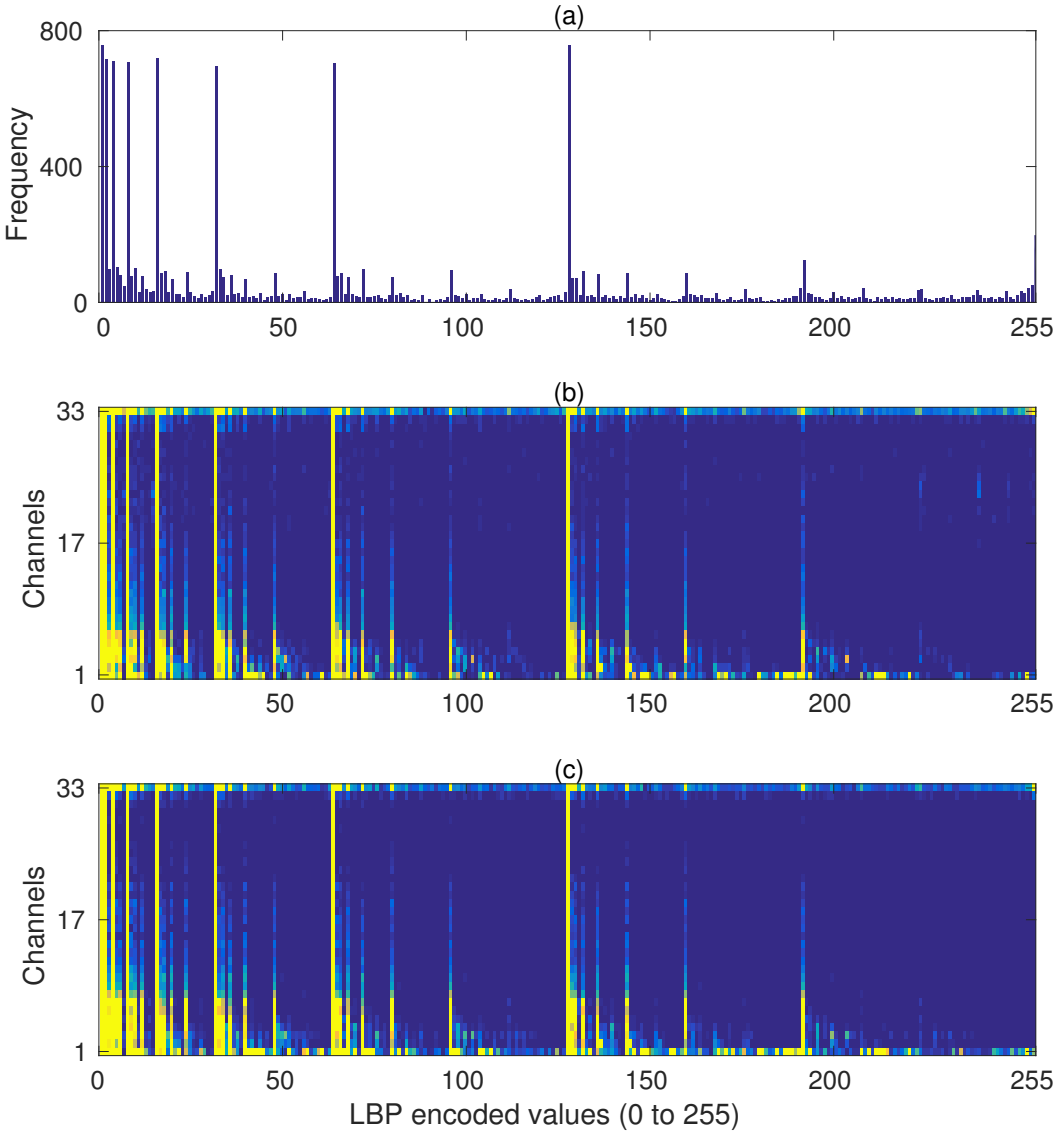

Figure 4.4: LBP histogram matrices. (a) illustrate LBP histogram of one channel of a speech utterance. (b) and (c) are LBP histograms of 33 channels of the speech signal shown in Fig. 2.3 produced by 2 speakers.

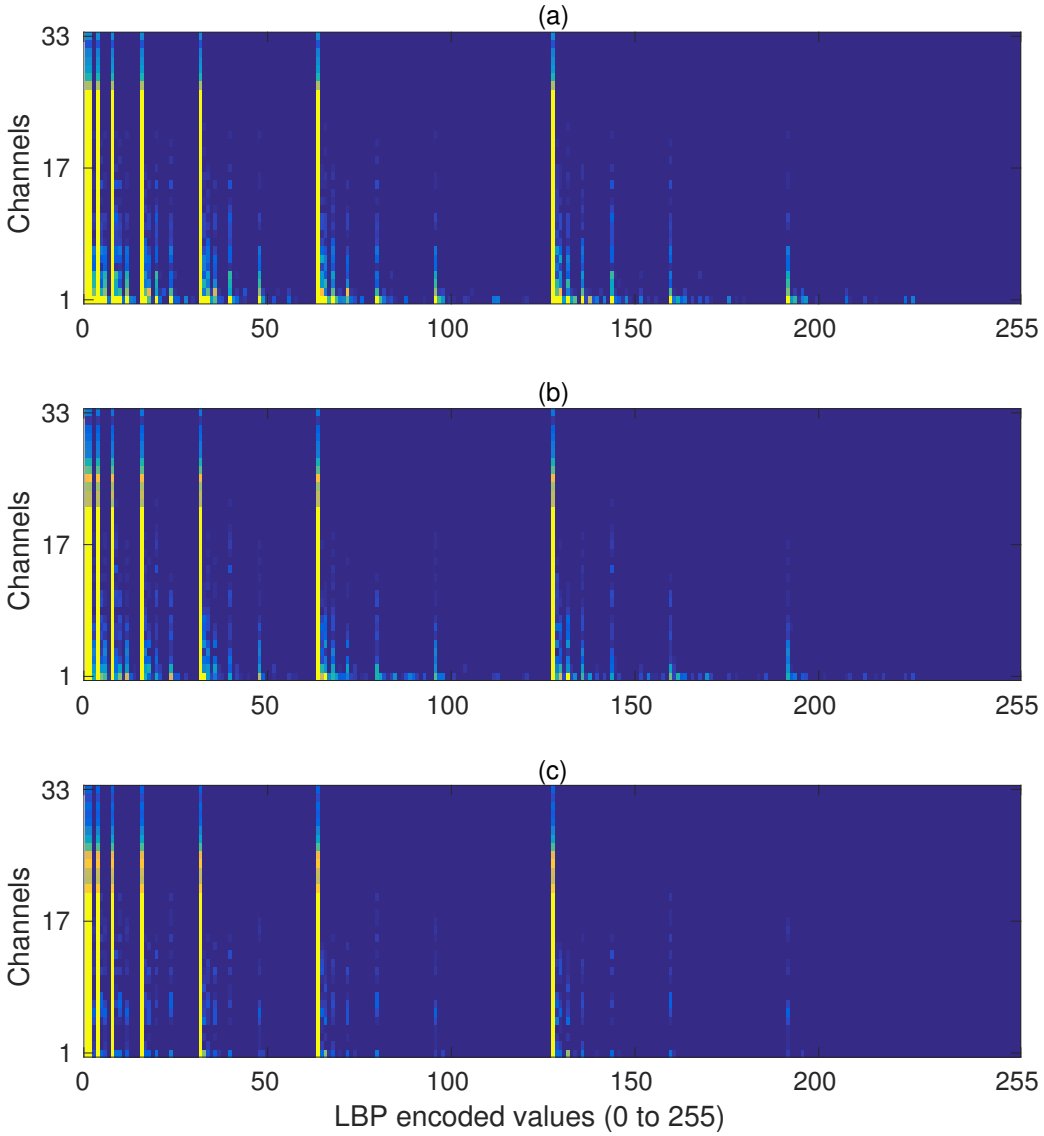

Figure 4.5: LBP histogram matrices of three speech signals that are produced by the same speaker and have different content. Color patterns of the three LBP histogram matrices are different indicating that texture of the spikegrams are different. The speech content includes /ohayo-gozaimasu/, /kon-nichiwa/, and /konbanwa/ shown in (a), (b), and (c), respectively.

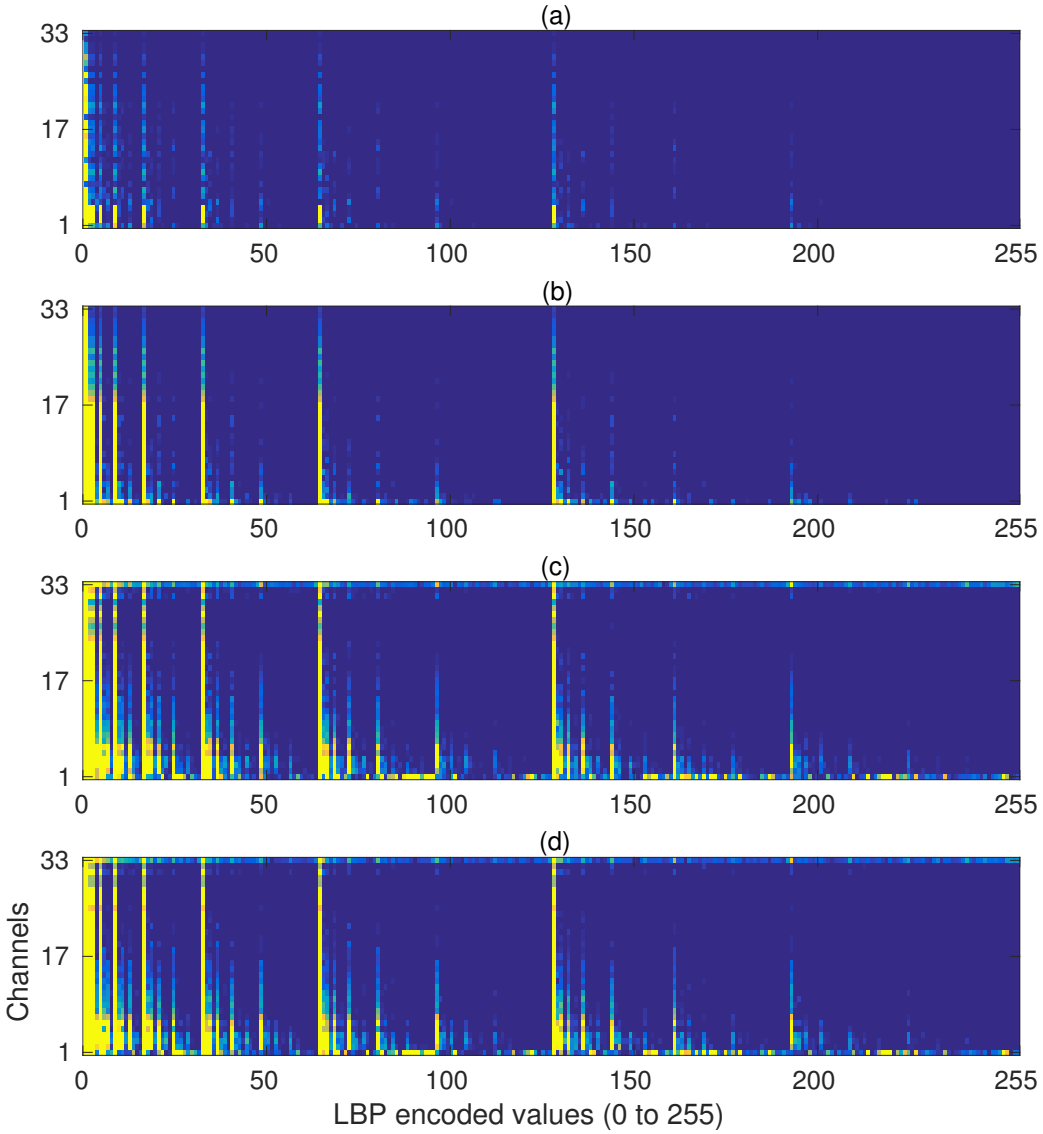

Figure 4.6: Comparing LBP histograms of different speech utterances to evaluate the effectiveness of spikegram. Color patterns in (a) are almost flat, which indicate that 2 LBP histograms are very similar. Color patterns in (b), (c), and (d) are spikier, which indicates that the LBP histograms are significantly different.

Chapter 5

Conclusion

5.1 Summary

We found that the matching pursuit algorithm used with Gabor or Gammatone dictionary outperformed the Gammatone filterbank in decomposing speech signals by evaluating the original speech signals and the resynthesized signals with SNR and PESQ, and sparse coding provided a better representation model for speech signals. More importantly, spikegrams could be used to replace spectrograms as a better alternative in the process of creating speech fingerprints.

Based on the experiments, we concluded that the spike distributions of spikegrams conveyed the linguistic content and voice identities of the utterers of speech signals. These features play a significant role in the accuracy and reliability of speech fingerprints.

5.2 Remaining issues

The aim of stage one is using spikegrams as the representation model for speech signals but there was only one sparse coding technique the matching pursuit algorithm was studied. Other methods used for creating spikegrams should be examined and their results should be compared to evaluate their performances. In addition, Gabor and Gammatone were the only two based spectra that were used in the process of creating spikegrams. Other kinds of base spectrum such as Gammachirp should also be investigated.

In stage 2, local binary pattern was somewhat able to represent characteristics of the spike distribution on spikegrams but its weakness was that it could only reflect the local changes of the spikes. A pattern analysis method that can incorporate all the spikes of an entire spikegram is preferable. Moreover, features that correspond to linguistic content and voice identity of speakers have not been thoroughly studied. Consequently, a model used for extracting these feature has not been built.

In stage 3, calculating the mean of the histogram matrices was too simple for a hash function and using mean and standard deviation to compare the distances of speech fingerprints was unable to achieve good recognition rate. Future development of this stage should target a hash function that could combine the extracted features from stage

2 to create better speech fingerprints. Furthermore, a suitable matching method should be studied to compare speech fingerprints.

So far, the performance of the proposed method was evaluated using a limited dataset. To ensure the quality of the method, a formal dataset should be used and results of each stage should be examine carefully.

5.3 Future work

Futher development of this research will be dedicated to resolve the remaining issues listed in the above section. Although there are still room for improvement in stage one, we belive that features on the spikegrams are sufficient to be analyzed. Therefore, stage two can be carried out with the following tasks:

1. An investigation will be conducted to recognize features that represent linguistic content and voice identity of speakers on the spikegrams.
2. A model will be built to extract these features.
3. The consistency of the features and the effectiveness of the model will be evaluated using a formal dataset.

Stage three can be carried out with the following tasks:

1. A robust hash function will be built to combine the well-tested features from stage 2 to produce speech fingerprints.
2. A suitable method used to compare speech fingerprints to achieve highest recognition rate will be carried on.
3. The robustness of the hash function and the comparing method will be examined using a formal dataset.

A refinement process is necessary to improve the overall quality of the solution. This can be done with the following tasks:

1. In stage 1, other kinds of base spectrum such as Gammachirp, and other kinds of sparse coding algorithms will be explored to compare their performance.
2. In stage 2, various kinds of feature extraction methods will be reviewed to evaluate their effectiveness.
3. In stage 3, different kinds of hash function will be studied to test their effectiveness.
4. Results from each stage will be compared to find the most effective ones.

Appendices

Non-negative matrix factorization

Non-negative matrix factorization (NMF) is a method used for decomposing a matrix into two other matrices that only contain non-negative elements.

The approximation is given by:

$$|V| \approx W \times B, \tag{5.1}$$

where $V \in R^{F \times T}, W \in R^{F \times P}$, and $B \in R^{P \times T}$. The matrix V can be a speech signal, the matrices B and W are commonly known as the dictionary of bases, or the base spectrum, for the former and the weight matrix, or activations, for the latter. The F and T are often referred as a frequency bin and a time bin respectively.

To perform this approximation, it is to minimize the β-divergence, which is defined as

$$D_\beta(c||d) = \begin{cases} \frac{1}{\beta(\beta-1)}(c^\beta + (\beta-1)d^\beta - \beta c d^{\beta-1}) & \beta \in R - \{0,1\} \\ c\log\frac{c}{d} - c + d & \beta = 1 \\ \frac{c}{d} - \log\frac{c}{d} - 1 & \beta = 0 \end{cases} \tag{5.2}$$

Commonly, researchers concentrated on 3 special cases of the β-divergence that are corresponded with $\beta = 0, 1, 2$. Or, they are minimizing the Kullback-Leibler divergence (KL) [31] (Eq. 5.3) minimizing the Itakura-Saito divergence [32] (Eq. 5.3) and minimizing the squared norm (Eq. 5.5) respectively [33, 34].

- The KL divergence

$$(V||WB)_{KL} = \sum_{f,t}(|V_{f,t}|(\log(\frac{V_{f,t}}{(WB)_{f,t}})) + (WB)_{f,t}) \tag{5.3}$$

- The Itakura-Saito divergence

$$(V||WB)_{IS} = \sum_{f,t}(\frac{V_{f,t}}{(WB)_{f,t}} - log\frac{V_{f,t}}{(WB)_{f,t}} - 1) \tag{5.4}$$

- The squared norm

$$||V||WB|| = \sum_{f,t}(|V_{f,t}| - (WB)_{f,t})^2 \tag{5.5}$$

It is seen that it has no direct way to find out the global minimums of the divergences above. Instead, an iterative method is applied. In that process, after the base matrix B and the weight matrix W are initialized by specific values, they are updated in each iteration. We can use the following formulas to update the base and weight vector [32, 34].

- Updating the base and weight matrices in the KL divergence

$$B_{f,p} = B_{f,p} \sum_i \frac{\frac{W_{p,i} V f,i}{(WB)_{f,i}}}{\sum_j W_{p,j}} \tag{5.6}$$

$$W_{p,t} = W_{p,t} \frac{\sum_i \frac{B_{i,p} V i,t}{(WB)_{i,t}}}{\sum_j B_{j,k}} \tag{5.7}$$

- Updating the base and weight matrices in the Itakura-Saito divergence

$$B_{f,p} = B_{f,p} \frac{W^T((WB)^{-2}V)}{W^T(WB)^{-1}} \tag{5.8}$$

$$W_{p,t} = Wp_t \frac{((WB)^{-2}V)B^T}{(WB)^{-1}B^T} \tag{5.9}$$

- Updating the base and weight matrices in the squared norm

$$B_{f,p} = B_{f,p} \frac{(W^T V)_{f,p}}{(W^T W B)_{f,p}} \tag{5.10}$$

$$W_{p,t} = W_{p,t} \frac{(VB^T)_{p,t}}{(WBB^T)_{p,t}} \tag{5.11}$$

When deriving the update equations, it is done on an auxiliary function of the cost function, by which the cost keeps decreasing after every iteration or going down hill in the finding [33, 34]. The auxiliary functions of the cost functions for the squared norm, the KL divergence and the Itakura-Saito divergence can be found in most of the study on NMF [32, 34].

The loop ends when the update is convergent. In another word, a threshold is satisfied (possibly an acceptable value of divergence) or no changes of the base and the weight matrices in further iterations are made. The latter situation of the convergence rarely happens, the preceding case is common in practice.

Bibliography

[1] Dominic Milano, "Content Control: Digital Watermarking and Fingerprinting," White Paper: Video Water Marking and Fingerprinting.

[2] S. E. Siwek, "True Cost of Recorded Music Piracy to the U.S. Economy," Denver, CO, USA: IPI, Aug. 2007.

[3] Pedro Cano, Eloi Batle, Ton Kalker, and Jaap Haitsma, "A Review of Algorithms for Audio Fingerprinting," IEEE Workshop Multimed. Signal Proc., 2002.

[4] Liu Xin and Chang-Chun Bao, "Audio Bandwidth Extension Based on Temporal Smoothing Cepstral Coefficients," EURASIP Journal on Audio, Speech, and Music Processing, vol. 2014, no. 1, 2014.

[5] Liu Xin and Chang-Chun Bao, "Audio Bandwidth Extension Based on Temporal Smoothing Cepstral Coefficients," EURASIP Journal on Audio, Speech, and Music Processing, vol. 2014, no. 1, 2014.

[6] Pedro Cano, Eloi Batlle, Harald Mayer, and Helmut Neuschmied, "Robust Sound Modeling for Song Detection in Broadcast Audio," Convention paper, Audio Engineering Society, 2002.

[7] Ning Chen and Wanggen Wan, "Speech Hashing Algorithm Based on Short-Time Stability," Artificial Neural Networks – ICANN 2009 Lecture Notes in Computer Science, pp. 426–434, 2009.

[8] Ning Chen and Wanggen Wan, "Robust Speech Hash Function," ETRI Journal, vol. 32, no. 2, pp. 345–347, 2010.

[9] Jaap Haitsma, Ton Kalker, and Job Oostveen, "Robust Audio Hashing for Content Identification," 2002.

[10] Abderraouf Ben Salem, Sid-Ahmed Selouani, Habib Hamam, and Jean Caelen, "A Highly Robust Audio Hashing System Using Auditory-Based Front-End Processing," IEEE International Conference on Acoustics, Speech and Signal Processing, 2009.

[11] Yuhua Jiao, Liping Ji, and Xiamu Niu, "Robust Speech Hashing for Content Authentication," IEEE Signal Processing Letters, vol. 16, no. 9, pp. 818–821, 2009.

[12] Avery Li-Chun Wang, "An Industrial-Strength Audio Search Algorithm," 2003.

[13] Ramin Pichevar, Hossein Najaf-Zadeh, Louis Thibault, and Hassan Lahdili, "Auditory-inspired Sparse Representation of Audio Signals," Speech Communication, vol. 53, no. 5, pp. 643–657, 2011.

[14] Masashi Unoki and Masato Akagi, "A Method of Signal Extraction from Noisy Signal Based on Auditory Scene Analysis," Speech Communication, vol. 27, no. 3-4, pp. 261–279, 1999.

[15] Dan Ellis, "Robust Landmark-Based Audio Fingerprinting," `https://labrosa.ee.columbia.edu/matlab/fingerprint/`

[16] Bruno A. Olshausen and David J. Field, "Sparse Coding of Sensory Inputs," Current Opinion in Neurobiology, vol. 14, no. 4, pp. 481–487, 2004.

[17] Evan Smith and Michael S. Lewicki, "Efficient Coding of Time-Relative Structure Using Spikes," Neural Computation, vol. 17, no. 1, pp. 1945, 2005.

[18] Yan Karklin, Chaitanya Ekanadham, and Eero P. Simoncelli, "Hierarchical spike coding of sound," Advanced Neural Information Process System, pp. 3032–3040, 2012.

[19] Yousof Erfani, Ramin Pichevar, and Jean Rouat, "Audio Watermarking Using Spikegram and a Two-Dictionary Approach," IEEE Transactions on Information Forensics and Security, vol. 12, no. 4, pp. 840852, 2017.

[20] Shadi Siahpoush, Yousof Erfani, Thilo Rode, Hubert H. Lim, Jean Rouat, and Eric Plourde, "Improving Neural Decoding in the Central Auditory System Using Bio-Inspired Spectro-Temporal Representations and a Generalized Bilinear Model," Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), 2015.

[21] Patrick Mineault, "Matching pursuit for 1D signals," `https://www.mathworks.com/matlabcentral/fileexchange/32426-matching-pursuit-for-1d-signals`

[22] Smith Evan and Michael S. Lewicki, "Efficient Coding of Time-Relative Structure Using Spikes," Neural Computation, vol. 17, no. 1, p. 19–45, 2005.

[23] Mallat, S.G. and Zhifeng Zhang, "Matching Pursuits with Time-frequency Dictionaries," IEEE Trans. Signal Process., vol. 41, no. 12, pp. 3397–415, 1993.

[24] Dong Chen He and Li Wang, "Texture Unit, Texture Spectrum, And Texture Analysis," IEEE Trans. Geosci. Remote Sens., vol. 28, no. 4, pp. 509-12, 1990.

[25] Timo Ojala, Matti Pietikhen, and David Harwood, "Performance Evaluation of Texture Measures with Classification Based on Kullback Discrimination of Distributions," Proceedings of 12th International Conference on Pattern Recognition, 1994.

[26] Di Huang, Caifeng Shan, Mohsen Ardebilian, Yunhong Wang, and Liming Chen, "Local Binary Patterns and Its Application to Facial Image Analysis: A Survey," IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 41, no. 6, pp. 765–781, 2011.

[27] Timo Ahonen, Jiri Matas, Chu He, and Matti Pietikainen, "Rotation Invariant Image Description with Local Binary Pattern Histogram Fourier Features," Image Analysis Lecture Notes in Computer Science, pp. 61–70, 2009.

[28] Zhen Lei, Shengcai Liao, Ran He, Matti Pietikainen, Stan Z. Li, "Gabor Volume Based Local Binary Pattern for Face Representation and Recognition." 2008 8th IEEE International Conference on Automatic Face and Gesture Recognition, 2008.

[29] Marko Heikkila, Matti Pietikainen, and Cordelia Schmid, "Description of Interest Regions with Center-Symmetric Local Binary Patterns," Computer Vision, Graphics and Image Processing Lecture Notes in Computer Science, pp. 58–69, 2006.

[30] Olli Lahdenoja, Mika Laiho, and Ari Paasio, "Reducing the Feature Vector Length in Local Binary Pattern Based Face Recognition," IEEE International Conference on Image Processing, 2005.

[31] James M. Joyce, "Kullback-leibler divergence," International Encyclopedia of Statistical Science. Springer Berlin Heidelberg, 2011, pp. 720–722.

[32] F. Itakura and S. Saito, "Analysis synthesis telephony based on the maximum likelihood method," In Proc. 6th of the International Congress on Acoustics, Los Alamitos, pp. C17C20, 1968.

[33] Lefevre Augustin, Francis Bach, and Cedric Fevotte, "Itakura-Saito nonnegative matrix factorization with group sparsity," Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on. IEEE, 2011.

[34] Fevotte Cedric, "Majorization-minimization algorithm for smooth Itakura-Saito nonnegative matrix factorization," Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on. IEEE, 2011.

Publications

[1] Dung Kim Tran, Masashi Unoki, "Study on Speech Representation Based on Spikegram for Speech Fingerprints," Advances in Intelligent Information Hiding and Multimedia Signal Processing, vol. 20, Springer 2017.

[2] Dung Kim Tran and Masashi Unoki, "Investigation of spikegram-based signal representation for speech fingerprints," IEICE Technical Report, EMM2017–35, pp. 241–246, Uchiyama–Youkou Show–Room, Tokyo, July 2017.