AN AUTOMATIC COMMERCIAL SEARCH APPLICATION FOR TV BROADCASTING USING AUDIO FINGERPRINTING

A Thesis

by

YAOHUA SONG

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2012

Major Subject: Electrical Engineering

AN AUTOMATIC COMMERCIAL SEARCH APPLICATION FOR TV BROADCASTING USING AUDIO FINGERPRINTING

A Thesis

by

YAOHUA SONG

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved by:

Chair of Committee,	Zixiang Xiong
Committee Members,	Jim Ji
	Tie Liu
	Jyh-Charn (Steve) Liu
Head of Department,	Costas N. Georghiades

August 2012

Major Subject: Electrical Engineering

ABSTRACT

An Automatic Commercial Search Application for TV Broadcasting Using Audio Fingerprinting. (August 2012)

Yaohua Song, B.S., University of Science and Technology of China

Chair of Advisory Committee: Dr. Zixiang Xiong

Nowadays, TV advertising is an important part of our daily life. However, it is usually hard for organizations that produce and pay for the advertisements to confirm whether their commercials are broadcasted as required in time and frequency. Consequently, a multimedia file search problem arises and it has drawn more and more attention in the past decade. In this thesis, we propose an automatic commercial search scheme using audio fingerprinting and implement it in a PC-based application.

Our commercial search algorithm is composed of two parts: one for audio feature extraction and another for database search.

For the first part, although the video stream of TV broadcast contains a great deal of intuitive information, we decide to ignore it because it takes much more storage and computations to process. For the audio stream, we have to extract proper audio features which can represent its characteristics and store them in a database for identification. We choose the Normalized Spectral Subband Centroids (NSSCs) as our audio fingerprints and preprocess the known commercials to build the database.

For the second part, we apply a three-step process to search for any matches as the user requests, which comprises candidate search, decision-making and time

verification. This process is performed for every N_1 (N_1=15 in our application) frames if the search result is negative. Once a match is confirmed, we skip the frames left in the commercial and use the frame after it to start a new process.

Our experiment results are satisfactory based on the commercial and TV program data in our database. Moreover, it shows that our PC-based application is robust against degradation during real broadcast and recording.

DEDICATION

To my parents

ACKNOWLEDGEMENTS

First and foremost I would like to express my sincere gratitude to my committee chair Dr. Zixiang Xiong, for his continuous guidance and great patience; he provides not only support throughout my thesis work but also advice on my graduate studies. I would also like to thank Dr. Jim Ji, Dr. Tie Liu and Dr. Jyh-Charn(Steve) Liu for serving on my committee and their valuable comments and time.

Thanks also go to my friends and colleagues in the Wireless Communication Lab of Texas A&M University, especially Dr. Yang Yang, Yifu Zhang, Nan Jiang, Bo Wang and Rengjing Zhang, for the encouragement and insightful discussions. Finally, my whole-hearted appreciation goes to my parents for their unconditional love, encouragement and support.

TABLE OF CONTENTS

Page

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

In the information age, we have access to hundreds of thousands of multimedia files through various means. However, as we have more resources than ever, searching over such an enormous database is becoming more and more challenging. Unlike text-based search, we need to extract features from multimedia files that can represent the distinguishing characteristics of them instead of directly using the content itself. The advantage of using fingerprints is three-fold: the memory/storage requirement is reduced; comparisons are efficient, as perceptual irrelevancies have already been removed; and searches are efficient, as the size of the database to be searched is much smaller.

1.1 Motivation

Nowadays, TV advertising is an essential part of our daily life. It involves two main tasks: 1) producing a television commercial that meets the broadcast standards, and 2) placing the commercial on television via an air time media buy that reaches the target customers. For the latter, it is usually hard for the organizations that create and pay for the advertisement to confirm whether their commercials are broadcasted in time and frequency required. As a matter of fact, TV commercial detection concerns a wide range of users, not only the producers. For instance, the end-users might want to skip the commercial breaks and watch the TV programs only. Marketing agencies can make use

---

This thesis follows the style of *IEEE Transactions on Signal Processing*.

of the results for their marketing analyses, plan or just demonstrate the relevant information to their interested clients.

The most straightforward method is to monitor the TV broadcasting manually and record the occurrences of the commercials. However, this approach is demanding, inefficient and expensive for companies and it is also boring and tedious for the staff. Therefore, an automatic commercial search system is needed. The problem has recently attracted a great deal of attention from both the research community and the industry.

1.2 Problem Formulation

The commercial search problem can be described as follows:

Given a predefined amount of recorded data of TV broadcasting and a set of known commercials, identify the occurrences of some commercials as the users specify or all the commercials stored in the database.

Before proposing an effective algorithm for the search, we need to review the characteristics of TV commercials. A typical TV commercial contains a video stream and an audio stream. A video stream is a sequence of still images representing scenes in motion. It is more intuitive and takes a larger amount of space to store than the audio information. Considering that the searching speed is one of the most important parameters that evaluates the system, only the audio part is used for the commercial identification in this thesis.

A source of difficulty with audio identification stems from the high dimensionality and significant variance of perceptually similar content [1]. The simplest

approach is direct comparison of the digitalized waveforms, which is neither efficient nor effective. It is necessary to extract a perceptual digest of a piece of audio content which summarizes it, i.e., an audio fingerprint. There are several requirements that a good audio fingerprinting system should fulfill:

Robustness: An audio clip should be identified regardless of the possible signal degradation, e.g., compression level, equalization, AD/DA conversion, background noise, etc. This is the reason why perceptual features of a recording are chosen as features. The false negative rate is typically used as a measure of robustness. It is the probability that the fingerprints of two perceptually similar audio clips are so different that a positive match won't be caused.

Reliability: The system should be able to discriminate between different fingerprints to guarantee the accuracy of identification and reduce the false positive rate. The false positive rate is the probability that one commercial is detected as another one.

Computationally Efficiency: Computation time and search speed are key parameters for system evaluation. It is related to the size of the fingerprints, the complexity of the fingerprint extraction, and the search algorithm.

These requirements are not independent. For example, the search speed tends to increase when the system is designed to obtain a more robust fingerprint. This is because audio fingerprint search is an approximate search, i.e., the fingerprint identified as the most similar is not necessarily the perfect match. As a result, the more robust the features are, the less similar they are. Therefore, there is a trade-off between dimensionality reduction and information loss.

2. AUDIO FINGERPRINTING TECHNIQUES

Audio fingerprinting or content-based audio identification (CBID) has been studied since the 1990s. It can be viewed as a special form of dimensionality reduction. While it seems intuitive for a person to tell the differences between various audio clips, it is much more complicated for a computer to accomplish the same task. On one hand, to a computer the audio signal is simply a series of real or digital numbers represented as a function of time. On the other hand, according to studies on the human auditory system (HAS), we know that the human ears decompose an incoming sound into a series of frequency components like a spectrum analyzer. The analyzed results are then passed to the higher processing centers of our brain. In other words, the HAS deals the signals in the frequency domain. Methods in both time and frequency domains are studied and developed for audio fingerprinting.

2.1 Time-domain Methods

The most basic approaches to analyze an audio signal are to examine the waveform and compute the mathematical measurements directly from it. These measurements are also called the physical features. Two commonly used physical features are introduced in the following sections.

2.1.1 Short-time Energy Function

The short-time energy of an audio signal is defined as

$$E = \frac{1}{N}\sum_{m=1}^{N}[x(m)w(m)]^2, \tag{2.1}$$

where $w(m)$ is a window (usually a Hamming window) of a fixed length N. N is also called the length of a frame. The short-time energy represents the signal's amplitude variation over time. It is the basis for differentiating between unvoiced and voiced components of speech since the short-time energy values of the unvoiced parts are much smaller than those of the voiced ones. Sometimes, the short-time energy is used to define loudness [2]

$$L = (\sum_{m=1}^{N}[x(m)w(m)]^2)^{\frac{1}{2}} = E^{\frac{1}{2}}. \tag{2.2}$$

2.1.2 Short-time Average Zero-Crossing Rate (ZCR)

The short-time average zero-crossing rate (ZCR) is calculated as

$$Z = \frac{1}{2}\sum_{m=1}^{N}|sgn[x(m)] - sgn[x(m-1)]|w(m), \tag{2.3}$$

where

$$sgn[x(m)] = \begin{cases} 1, & x(m) \geq 0, \\ -1, & x(m) < 0. \end{cases} \tag{2.4}$$

ZCR is another effective parameter to distinguish voiced portions from unvoiced ones in speech, as the voiced parts generally have much smaller values than the unvoiced parts. In addition to this property, the ZCR curve can serve as an indicator for music and speech. The curve for music is much smoother, with significantly lower variance and mean amplitude, compared to that of speech.

2.1.3 Conclusion

There is very limited success on understanding audio signals' characteristics by applying the time-domain analysis. It only enables us to detect silence as a segment of imperceptible sounds from the whole file, including unnoticeable noise and very short clicks [3].

2.2 Frequency-domain Methods

As mentioned above, HAS performs all the functions in the frequency domain. Thus most research in the audio fingerprinting focuses on features extracted from the frequency domain. It is necessary to preprocess the signal. First the input signal is cut into a series of overlapping frames of length N. This assumes that the signal can be considered as stationary over an interval of a few milliseconds [1]. The overlap between the frames is to ensure the algorithm's robustness to shifts. Then each frame is filtered by a Hamming window to minimize the discontinuities at the beginning and the end.

2.2.1 The Fourier Transform

The Fourier transform of an array of values is defined as:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi k \frac{n}{N}}, k = 0, \dots, N-1, \tag{2.5}$$

where N is the length of the array. To decrease the computational complexity, N is usually chosen to be a power of two so that Fast Fourier Transform (FFT) algorithms can be implemented. From (2.5), we know that X_k normally is a complex number. Due to

the sensitivity of the phase of Fourier transform to frame boundaries and the fact that the human auditory system is relatively insensitive to phase, generally only the power spectral density, calculated as (2.6), is applied for further processing.

$$P_k = |X_k|^2, \tag{2.6}$$

In practice, x_n is always a real number, which results in the symmetry of the $X_k's$. Therefore, it is sufficient to only consider the first half of the spectrum.

In [3] 128-point FFT of the signal was directly used to obtain 65-dimensional vectors as features, which were taken to denote different kinds of timbre and build the Hidden Markov Model (HMM). Clearly, the ratio of the dimension reduction is not satisfying. To get a higher ratio, some research work was done based on several classic algorithms for pattern recognition. Burges, Platt, and Jana [4] proposed an algorithm called Distortion Discriminant Analysis (DDA), which uses oriented Principle Component Analysis (PCA) to project its input (the 128 preprocessed log spectra) into directions maximizing the SNR for a given set of distortions. DDA produces 64 values out of every 243.6ms of audio under the assumption that distorted versions of a set of training signals are available. This algorithm is not suitable for our application since there is only one version of each commercial available in the database and the input size is too large for the database search. So further advanced analysis is required to retrieve more distinct features from the signal's spectrum.

2.2.2 Brightness

Brightness is computed as the centroid of the short-time Fourier magnitude spectra:

$$B_n = \frac{\sum_{m=1}^{N} F_m * m}{\sum_{m=1}^{N} F_m}, \tag{2.7}$$

where

$$F_m = \log(X_m). \tag{2.8}$$

It is a measure of the higher frequency content of the signal, considered as an acoustical feature in [2].

2.2.3 Pitch

Pitch, also called as the short-time fundamental frequency (f_0), is a major auditory attribute of musical tones, along with duration, loudness, and timbre [5]. According to the American National Standards Institute (ANSI), it is officially defined as the attribute of auditory sensation in terms of which sounds may be ordered on a scale extending from low to high. Pitch depends not only on the frequency content but also the sound pressure and the waveform. The fundamental frequency (f_0) of a sound signal is the lowest frequency component that relates well to the other partials [6]. Harmonic Product Spectrum (HPS) is a classic algorithm of pitch detection [7]. For each frame, it measures the maximum coincidence for harmonics according to

$$Y(\omega) = \prod_{r=1}^{R} |X(\omega * r)|, \tag{2.9}$$

$$\hat{Y} = \max_{\omega_i} \{Y(\omega_i)\}, \tag{2.10}$$

where R is the number of harmonics to be considered, and frequency ω_i is inside the range of possible fundamental frequencies. Fig.1 is an example of the HPS with R=5 [8].

If f_0 is below 50 Hz, the audio signal is considered as non-harmonic. Sounds produced by most musical instruments and voiced components in speech are harmonic while unvoiced parts are non-harmonic.

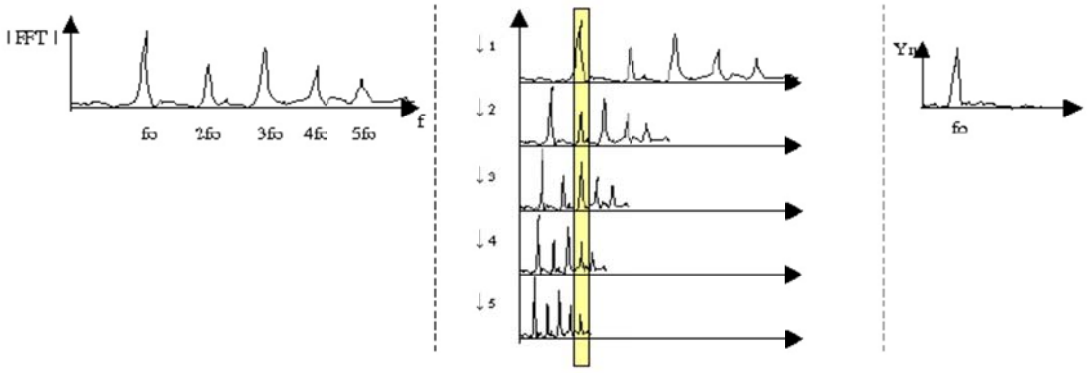

Fig. 1. An example of the HPS with R=5.

Zhang and Kuo [3] combined pitch, the ZCR and the short-time energy to classify the segments of an audio file into different kinds step by step. First, the "silence" part is detected as discussed before. Then some special environmental sounds are separated using the fundamental frequency curve, followed by distinguishing speech and music based on the additional ZCR information. Last, other environmental sounds are further sorted using the log spectrum.

To take advantage of these characteristics, certain fundamental analysis was performed on them. In [2], the authors used statistical values, including means, variances and autocorrelations, of these parameters to increase their reliability. However, this kind of mechanism is only suitable for those sounds with a single timbre. Since a TV

commercial usually contains both music and speech, which are two classes of sounds, this approach is not a good choice for us.

2.2.4 Mel-frequency Cepstral Coefficients (MFCCs)

Mel-scale was proposed by Steven, Volkmann, and Newman [9] as a subjective scale for the measurement of pitch. Compared to the frequency scale, it resembles the human auditory system better because it is constructed from the determinations of the half-value of pitches at various frequencies judged by listeners. The most popular formula of the conversion between hertz (*f*) and mel (*m*) is:

$$m = 2595 log_{10}\left(1 + \frac{f}{700}\right), \tag{2.11}$$

or

$$f = 700 * \left(10^{\frac{m}{2595}} - 1\right), \tag{2.12}$$

Fig.2 is the plot of mel versus Hertz scales. It is seen that we can tell that human hearing is more sensitive in the low frequency region than that in the high frequency.

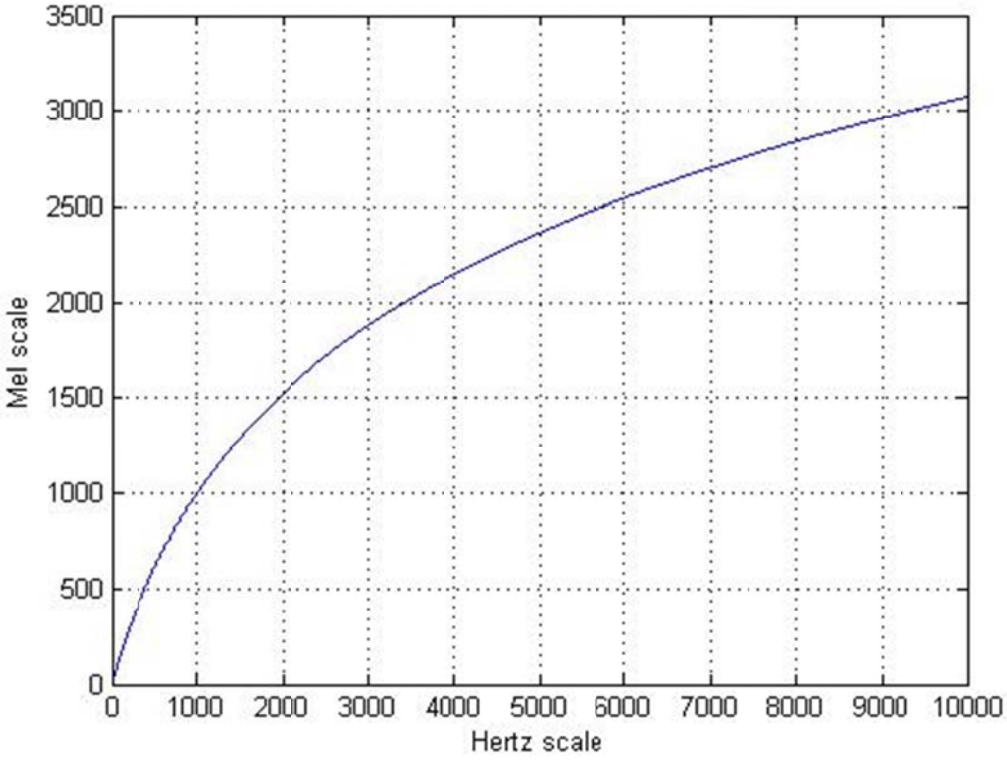

Fig. 2. The plot of mel scale versus Hertz scale

MFCCs are derived as follows:

a) Calculate the FFT for the preprocessed audio data, X_k.

b) Map the log power of the spectrum onto the mel scale, using L windows.,

$$M_n = \sum_{k=1}^{N} MelBank[n][k] * \log(P_k), \qquad n = 1,2,\dots,L, \qquad (2.13)$$

c) Calculate the DCT (discrete cosine transform) of the resulting signal.

$$MFCC_k = \frac{2}{L}\sum_{n=1}^{L} M_n * \cos\left[\frac{\pi}{L}\left(n+\frac{1}{2}\right)n\right], \qquad n = 1,2,\dots l. \qquad (2.14)$$

This procedure produces a l-dimension vector for each frame. The 0^{th} coefficient of the MFCC cepstrum is neglected due to its unreliability. Sometimes, the 1^{st} or 2^{nd} order differential MFCC is used to increase the reliability.

The number and the shape of the windows are not fixed in step b), so the designers can determine the dimension of the fingerprint according to their needs. In [10], the authors compared the performance of various implementations of MFCC by changing the number, shape, and locations of the windows and the usage of energy information. Three different shapes of filters are shown in Fig. 3.

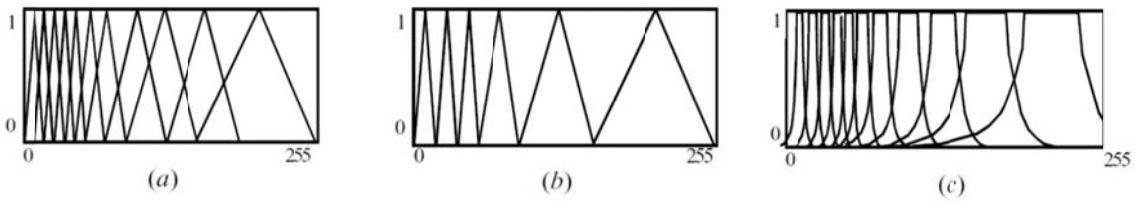

Fig. 3. Three different shapes of filters used in MFCC calculation.

MFCCs were shown to be quite effective for speech recognition, thus they were widely used and discussed in [11-13]. But they do not work so well on more general audio, like music.

2.2.5 Spectral Subband Centroids (SSCs)

Paliwal [14] first proposed Spectral Subband Centroids (SSCs) as new features for speech recognition. The basic idea was to take advantage of formant frequencies to eliminate the cepstral features' sensitivity to the noise distortion. He discussed several

possible strategies by setting the number of subbands, the shape of the subband filter and the type of the scale (hertz scale or mel scale). But no firm conclusions of the parameter selection were given since the author's main purpose was to show that SSC improved the recognition performance as a supplement to cepstral coefficients. It was further investigated and implemented in [15-18].

Chen, Huang, Li, and Paliwal [19] reexamined the SSC's potential as an independent feature set and compared it to MFCC. They conducted a series of experiments to study the effect of the number of subbands on the performance. The result was shown in Fig.4 [19].

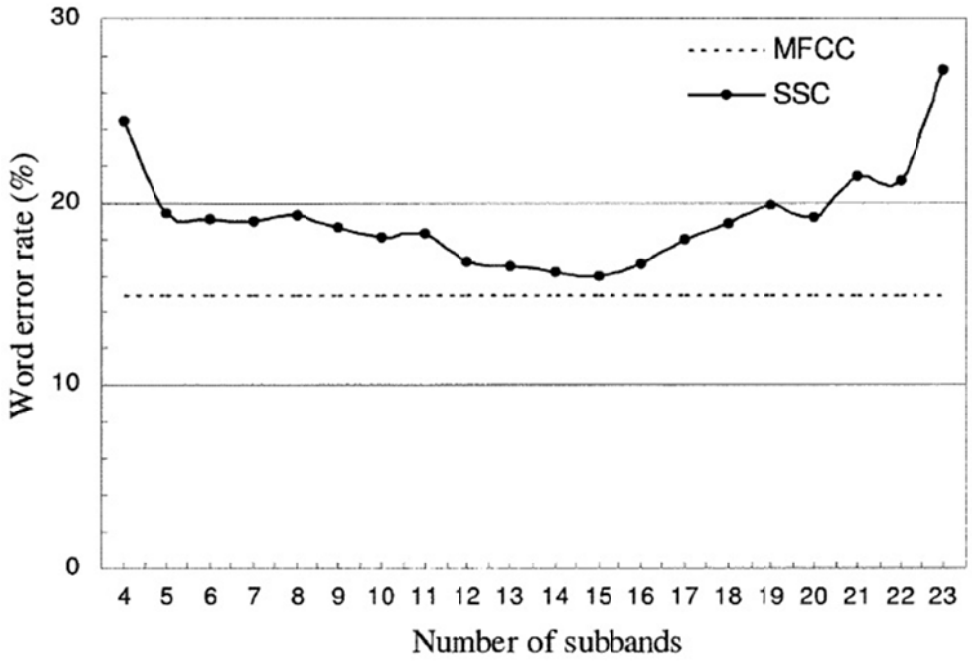

Fig. 4. Recognition performance of isolated spoken alphabet letters versus the number of subbands (without dynamic features).

Their study indicated that the optimal number of subbands was 15, while the SSC features still performed comparably well to the MFCCs when their numbers were between 10 and 20. Furthermore, the SSC coefficients are more robust to the noise than the MFCCs, as shown in Fig.5. In addition to the comparison between MFCC and SSC, the authors further reduced the word error rate by introducing the static, delta, and acceleration vectors of these two features.

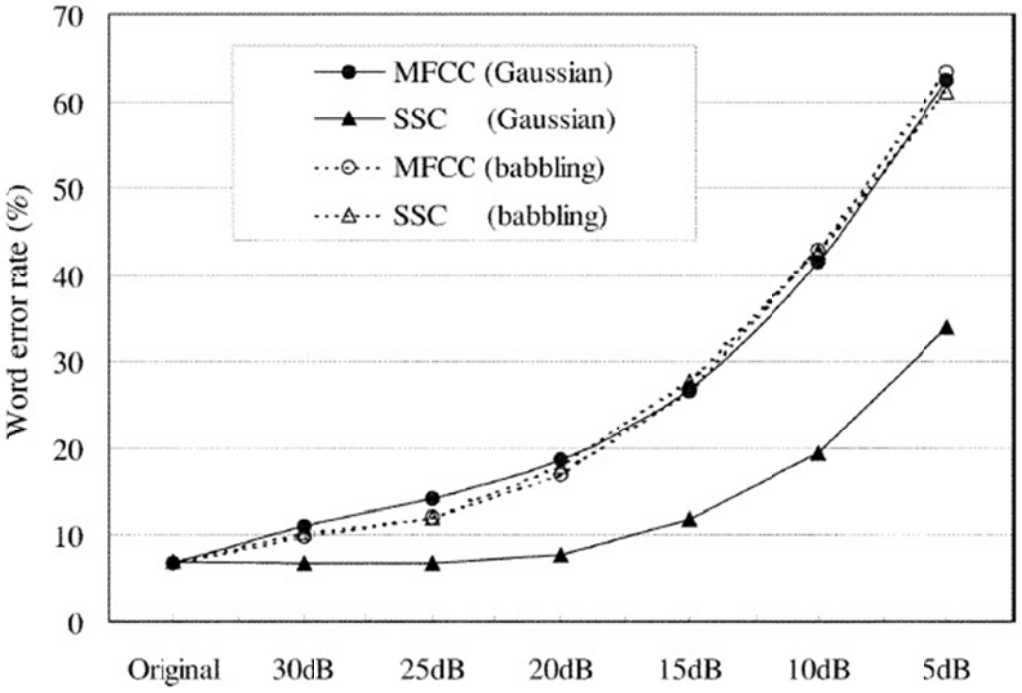

Fig. 5. Recognition performance of isolated spoken alphabet letters as a function of SNR in white Gaussian noise and babbling noise.

SSCs were further discussed in [20] and implemented as a normalized version NSSC in [21] [22].

2.3 Other Methods

Instead of the audio fingerprinting schemes, several video-based approaches have been proposed as well. In [23], the authors presented a prototype application for the TV monitoring only using video information. The visual hash used in [23] was computed from the low-frequency part of both the luminance and chroma frame components. The key assumption for the detection method was that commercials are only broadcasted within the commercial breaks, which are indicated by the bumpers for a given TV station. Under this assumption, there is no need for the authors to distinguish between the TV program and commercials, which greatly simplifies the search task. Unfortunately, the bumper information is unavailable in our system.

3. SEARCHING IN HIGH DIMENSIONS

Thanks to various audio fingerprinting algorithms, we are able to reduce the dimension of the audio data remarkably, typically from thousands to tens. Nevertheless, searching in the feature database is still not as simple as searching in a relational database. In contrast to traditional database applications, the basic function of a feature database is to search for all objects within it that are similar enough to a given object, rather than an exact match. This operation is also called a range query. Formally, given a file of N records (each of which is described by k attributes), a dissimilarity measure function $d(x, y)$, and a predefined threshold δ, the operation is to find all the records in the query range of a query record with specified attribute values. The query range of a record is defined as the region in which all the vectors' distances from the record is no larger than δ. When the database is small, a sequential scan is sufficient to find all the results. When the database is very large and each record is a high-dimension vector in reality, it is essential to apply appropriate multidimensional techniques for efficient search.

3.1 Dissimilarity Measures

Regardless of what technique is used to speed up the search process, a specific dissimilarity measure has to be employed to decide how similar two vectors in the space are. In our problem description, there are no restrictions for the function $d(x, y)$. However, there are several implicit underlying assumptions.

A dissimilarity measure function of k-dimension vectors is defined as:

$$d(x,y) \triangleq F\left[\sum_{i=1}^{k} f_i(x_i,y_i)\right], \tag{3.1}$$

where $\{f_i\}_{i=1}^{k}$ are required to be symmetric

$$f_i(x,y) = f_i(y,x), \qquad 1 \le i \le k, \tag{3.2}$$

and both $\{f_i\}_{i=1}^{k}$ and F are monotonous

$$F(x) \ge F(y), \qquad if\ x > y, \tag{3.3}$$

$$f_i(x,z) \ge f_i(x,y), \qquad if\ x \ge y \ge z\ or\ x \le y \le z, \qquad 1 \le i \le k, \tag{3.4}$$

$\{f_i\}_{i=1}^{k}$ are called the coordinate distance function, which represent the one-dimensional distance along every coordinate. Generally, they are all identical, that is, for $1 \le i \le k$, $f_i(x,y) = f(x,y)$. Then $\sum_{i=1}^{k} f_i(x_i,y_i)$ can be simplified to the linear function $\hat{f}(x,y) = |\vec{x} - \vec{y}|$. Considering the geometrical concept of a vector, a metric distance is commonly a good choice for $d(x,y)$. A metric distance satisfies the triangle inequality

$$d(\vec{x},\vec{y}) + d(\vec{y},\vec{z}) \ge d(\vec{x},\vec{z}). \tag{3.5}$$

The most widely used metric distances are the vector space p-norms

$$d_p(\vec{x},\vec{y}) = \left[\sum_{i=1}^{k} |x_i - y_i|^p\right]^{\frac{1}{p}}. \tag{3.6}$$

The Euclidean distance where p=2 is employed in our system.

3.2 Vector Quantization

To search efficiently, one might think of classifying the set of acquired features into a few groups such that every feature vector can be expressed as the index of its class.

This is also the basic idea of vector quantization and pattern recognition/classification. These two fields have been studied for decades. A number of effective and classical algorithms are proposed and accepted in a wide range. They are potential candidates for this application. We will introduce one mechanism and discuss its feasibility.

The *K*-Nearest Neighbor (*K*-NN) Algorithm is one of the most commonly used methods in pattern recognition. The original application of this method was to search for the most relevant features that could distinguish among the classes. One simple example is the Nearest Neighbor search by choosing *K*=1.

The goal of the *K*-NN algorithm is to minimize the average squared error in every subset. Let $G_k = \{X_1, X_2, \dots X_{|G_k|}\}$ be the subset containing all the members classified as class k; Y_k is the "center" of class k (also known as the codeword). Then the average squared error e_k in G_k can be calculated using

$$e_k = \frac{1}{|G_k|}\sum_{i=1}^{|G_k|}|X_i - Y_k|^2. \tag{3.7}$$

The total average squared error is defined as

$$E = \sum_{i=1}^{K} e_i. \tag{3.8}$$

Now let $X = \{X_1, X_2, \dots X_N\}$ be the training set. To get the codebook $Y = \{Y_1, Y_2, \dots, Y_M\}$, the *K*-NN Algorithm go through the following steps:

a) Initialize the codebook using (3.9), assuming the training data is divided into M subsets of equal size $\frac{N}{M}$ (in practice, it is probably not an integer. For the facility of analysis, $\frac{N}{M}$ is treated as an integer.)

$$Y_i = \frac{M}{N} \sum_{j=(i-1)*\frac{N}{M}+1}^{i*\frac{N}{M}+1} X_j. \tag{3.9}$$

b) Calculate the total average of squared errors $\{E_i\}$ for $\{Y_i\}$ using (3.8).

c) Classify the training data X into different classes G_k based on the NN rule, which is:

$$X_i \in G_k \ \ if\ d(X_i, Y_k) \leq d(X_i, Y_j), \qquad any\ j \neq k, \tag{3.10}$$

where $d(X, Y)$ is defined as the Euclidean distance.

d) Recalculate the Y_i's using the new subsets G_k's to get a new codebook Y'.

e) Calculate the total average squared error $\{E_i'\}$ for $\{Y_i'\}$. Compare $\{E_i\}$ and $\{E_i'\}$. If they are not close enough, let Y' be Y, repeat step b). Otherwise, stop. The current codebook Y is the result.

After creating the codebook Y for the algorithm, the system quantizes every input vector $\vec{x}$ into a class under the scheme below: for an unclassified input, the algorithm searches for the *K* nearest training samples to the query, then assigns it the label that is most frequent among these *K* samples. Obviously, the classification phase of the classic *K*-NN algorithm is too complex. So we adjusted this part to label the input using the index of the nearest codeword as the label, which is the basic quantization concept.

Using this quantization scheme, every feature vector is represented by a number so that an audio clip is expressed as an integer sequence. The problem of searching for commercials in a segment of TV broadcasting is turned into the task of searching for a

short sequence in a much longer one. This transformation seems quite promising; however, there are three problems with this approach:

i) The generation and update of the codebook can be demanding. Review the procedure to get Y. It is a recursive process and its performance depends on the accuracy of the initial codebook as it relates to the final one and the threshold set for the change in $\{E_i\}$. As the size of the training set increases, the necessity of updating the codebook cannot be neglected. Every time the database is updated, the generation procedure needs to be conducted again.

ii) Quantization ensures that the input is quite similar to those training data that label as the same codeword. Nevertheless, it ignores the possibility that there may be other similar data in other subsets. On one hand, imagine a hypersphere centered in the input vector $\vec{x}$ with a radius of δ, which is the given threshold in the initial problem description. Any training data inside this hypersphere is considered as qualified. On the other hand, after quantization, all the candidates selected lie in a hypersphere centered in the codeword Y_i instead of $\vec{x}$. There is a good chance that the search is not complete.

iii) Due to the frame loss phenomenon and the problem discussed in ii), the simple sequence task becomes complicated. First, the frame loss phenomenon requires the consideration of time wrapping during the sequence comparison. Dynamic time wrapping (DTW) is a potential solution. Second, for the issue caused by quantization, the search algorithm needs to be designed for

approximate sequences, rather than exact matches. These two conditions greatly increase the complexity and computations of the sequence search algorithm.

Therefore, vector quantization is not an appropriate choice for our purpose.

3.3 Techniques based on Index Structure

According to the analysis of vector quantization methods above, we realize there are several properties that the search algorithm should meet: 1) the classes are represented by regions instead of points, and 2) the update of the database and the classifications is convenient.

Bohm, Berchtold, and Keim [24] provided an overview of the techniques in querying multimedia databases, which described various index structures and algorithms for query processing in high-dimensional space. Their paper starts with the mathematical analysis of the effects caused by the increase of dimensionality, which are subsumed by the term “curse of dimensionality”. Then the problems were identified and the authors reviewed the proposed methods to overcome these obstacles. Rather than exploring the details of all the approaches, the focus was to understand the basic concepts and common principles shared in them. High-dimensional indexing methods are based on the principle of hierarchical clustering of the data space. Tree structure is resembled and two kind of nodes used for construction (see Fig. 6).

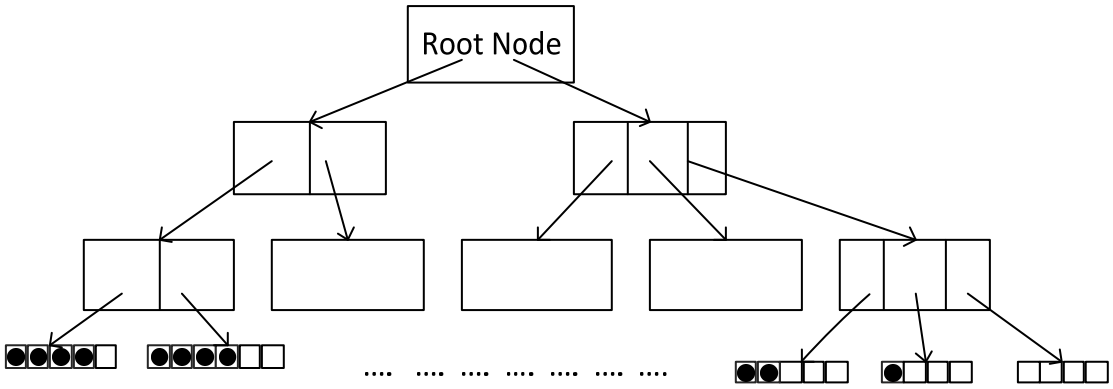

Fig.6 Hierarchical index structures.

The height of the tree is normally the dimension of the vector as only one component is examined at a time. The leaf nodes of the tree are called data nodes, storing the vectors. All the other nodes are the directory nodes, which consist of the partitions inherited from their parent nodes and those passed to the child nodes. In other words, each directory node is a split point for the hyperplane and every leaf node represents a region of the entire high-dimension space. It is easy to see that the region that one directory node defines must always be completely covered by that of its parent. Analogously, all vector objects stored in a subtree lie in the region of its root node. This property enables the system to cut branches of the tree from further processing. For example, if a region does not intersect the with the query range, it is impossible that any sub-region in its subtree intersects with the query range. As a result, only those nodes intersecting with the query range worth further investigation.

It is not surprised that binary tree is a reasonable and proper choice as the framework to organize the high-dimension database, considering the total number of the leaf nodes and the simplicity of the basic operations. A bigger cause of concern is the

partition strategy design to make the best use of the structure. The splitting rule decides the shape, volume, and distribution of all the regions. The shape of a region can be a hypercube, a hypersphere, a multidimensional cuboid, a multidimensional cylinder or a combination of several of the above. A summary of a few leading high-dimensional index structures and their properties was given in [24] (and summarized in Table.1, where MBR stands for minimum bounding rectangle).

Table. 1. High-dimensional index structures and their properties.

Name	Region	Disjoint	Complete	Split Criteria
R-tree	MBR	No	No	(Various algorithms)
R*-tree	MBR	No	No	Surface area Overlap Dead space coverage
X-tree	MBR	No	No	Split History Surface/overlap Dead space coverage
LSD^h-tree	kd-tree region	Yes	No/Yes	Cyclic change of dimension Distinct values
SS-tree	Sphere	No	No	Variance
TV-tree	Sphere with reduced dimension	No	No	Seeds with least common prefix Maximum distance
SR-tree	Intersect sphere/ MBR	No	No	Variance
Space filling curves	Union of rectangles	Yes	Yes	According to space filling curve
Pyramid tree	Trunks of pyramids	Yes	Yes	According to pyramid-mapping

From Table.1, we can see that most structures choose the rectangle or the sphere as the geometrical shape of the regions. The resulting segmentation, which is a primary measure of the performance, is not always disjoint or complete over the entire space. On

one hand, disjoint partitioning guarantees that there is no overlapping between any nodes, thus the decision of which subtree to search is unambiguous. On the other hand, complete partitioning has the dis-advantage that the regions are generally larger than necessary. Taking account of these factors, a modified K-D tree is a reasonable choice for the audio feature search.

The original K-D tree was proposed by Bentley [25], where K is the dimensionality of the space. The partition values were randomly selected and the discriminator for each level was obtained by cycling through the keys in order. Fig. 7 is a 2-D space example. The entire space is cut into the 8 subspaces based on the K-D tree on the right. Several associated basic operation algorithms, e.g. insertion, deletion, and search, were designed and evaluated by average running time.

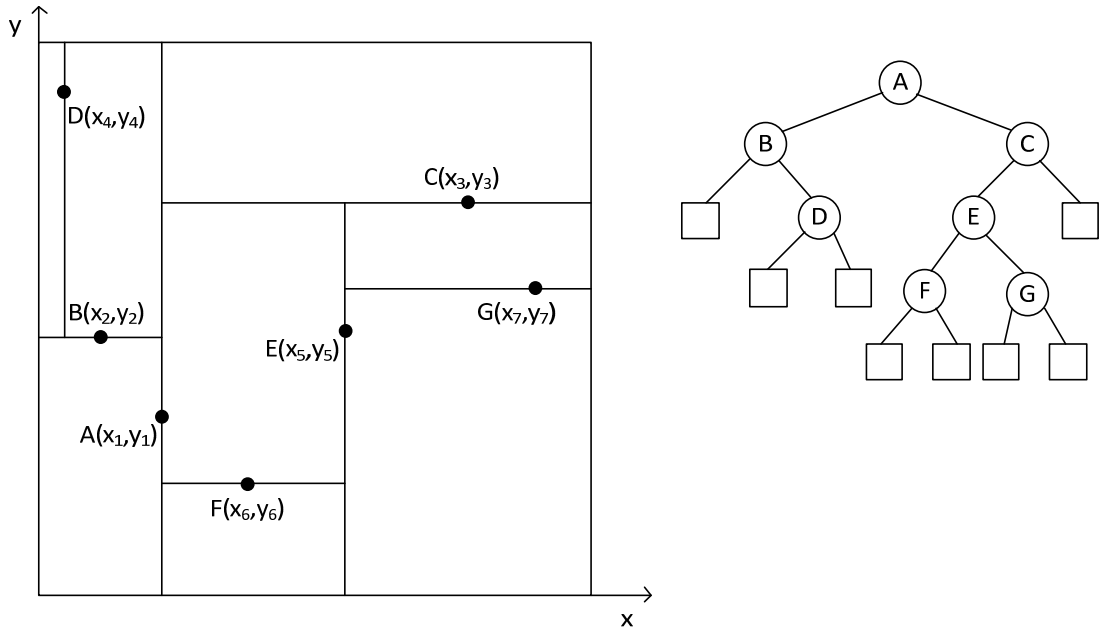

Fig. 7. A 2-D space example of the original K-D tree structure.

It was modified and optimized in [26] with respect to the expected number of records examined in the search process. The modifications are made in three aspects:

a). The height of the tree K is fixed as the number of attributes that one record has. As a result, each component in a feature vector is only checked once in the algorithm. It is reasonable because K is usually quite large. There is no need to check them more than once.

b). The order of the discriminator key is decided by the ranges along different dimensions. The range of a specific component is defined as the difference between the minimum and the maximum values in the database. The dimension with the largest range is checked first, then the second largest one, and so on.

c). The partition value of every discriminator key is defined as the median value of its distribution in the database.

The goal of this selection of the order of the discriminator keys and the partition is to optimize the binary K-D tree for the current database without any knowledge of the distribution of the queries. In particular, it is unnecessary to store every node's bounds in advance as they can be computed in the process using those of the parent node and the information defined in b) and c). The tree is easy to update and this structure also avoids a general binary tree optimization, which can be very time-consuming.

The search algorithm in the modified K-D tree is a recursive procedure. Recall the task of a typical search task: given a predefined threshold δ, and a dissimilarity measure function $d(x, y)$, find all the records in the query range of a given query record with specified attributes in the database. Starting with the root node, the process

examines the geometric boundaries the current node indicated. If they overlap the ball centered at the given record with a radius equal to the dissimilarity threshold δ, then the subtree must be considered as potential candidates and the procedure is called recursively for the child nodes. This is referred to as the "bounds-overlap-ball" test. The details will be introduced in Section 4.3.

At last, the authors analyzed the performance theoretically and experimentally for various combinations of the dimensionality and the dissimilarity function.

Foote [11] applied the modified K-D tree structure for the quantization in his system and used the maximum mutual information (MMI) of every dimension to optimize the partitions. Similarly, every leaf node in the tree is a class i and every non-terminal node represents a hyper rectangular cell j of the high-dimensional space. Let t_d be the partition along dimension d. To calculate the mutual information I(X; I), two probabilities are estimated first. For the region of j , define $P_r(i)$ as the probability of class i and $P_r(a_i)$ as the probability that a member of class i is above t_d. They are computed using

$$P_r(i) \approx \frac{N_{ij}}{N_j}, \qquad P_r(a_i) \approx \frac{A_i}{N_{ij}}, \tag{3.11}$$

where N_{ij} is the number of points in cell j from class i , N_j is the number of points in cell j, and A_i is the number of points from class i with x_d greater than t_d. Then the mutual information is calculated as

$$\mathrm{I(X; I)} = \mathrm{H(I)} - \mathrm{H(I|X)} = -\sum_i P_r(i) log P_r(i) + \sum_i P_r(i) H\big(P_r(a_i)\big)$$

$$\approx -\sum_i \frac{N_{ij}}{N_j} log \frac{N_{ij}}{N_j} + \sum_i \frac{N_{ij}}{N_j} \left[-\frac{A_i}{N_{ij}} log \frac{A_i}{N_{ij}} - \left(1 - \frac{A_i}{N_{ij}}\right) \log \left(1 - \frac{A_i}{N_{ij}}\right) \right]. \quad (3.11)$$

It is a function of t_d need to be maximized. An associative stop rule is set to prevent the optimization process running for ever.

4. PROPOSED SEARCH ALGORITHM

The whole system proposed can be divided into two parts: audio feature extraction and database search. There are two aspects to pay attention to:

First, as mentioned above, TV broadcasting data typically consist of both video and audio parts. To apply the audio feature extraction algorithm, the audio parts should be extracted from the recorded TV broadcasting streams before any data are processed in the system. For our application, the “AoA Audio Extractor” [27] is used to obtain the audio files in the WAVE format.

Second, the database used in this application comprises four parts: the original commercial audio files, the file containing all the features of known commercials, the file recording the basic information of each commercial, and the file containing the key information for the K-D tree. Construction of the database is assumed to be finished offline before the system starting the search process The files are stored in the hard disk in the format of text files. The structure of the database will be explained later in Section 4.2. Fig.8 is the overview of the entire system.

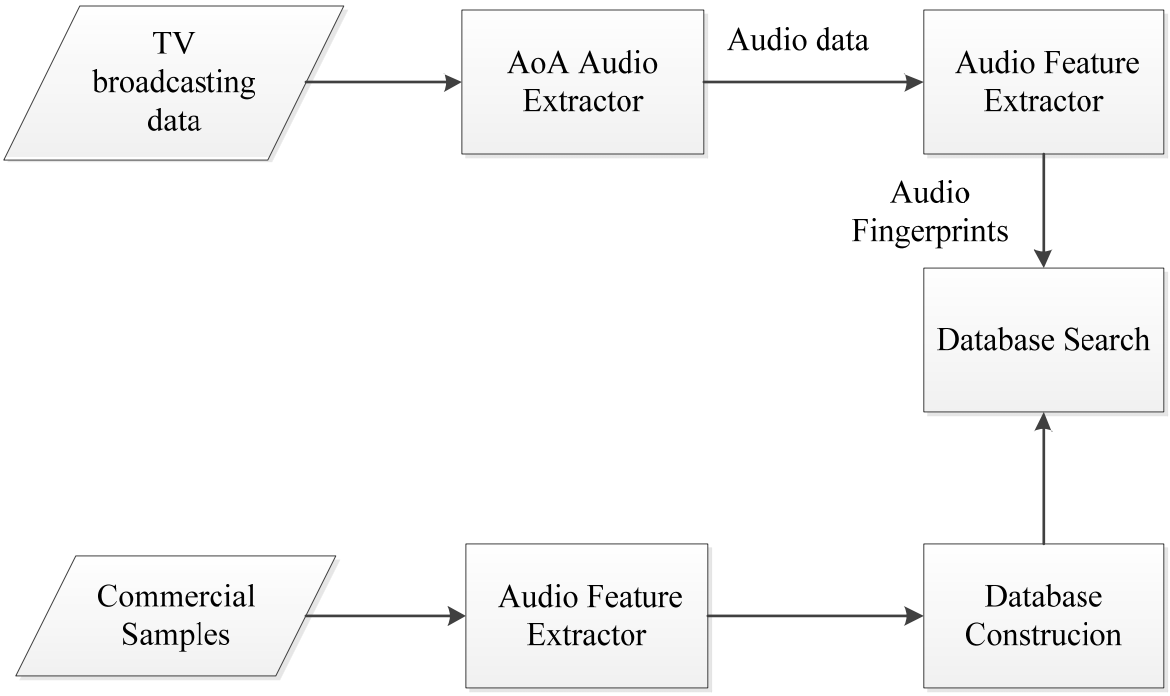

Fig. 8. The overview of the commercial searching system.

4.1 Audio Feature Extraction

Recall the various audio fingerprinting techniques reviewed and discussed in Section 2. The time domain methods do not qualify since they can only classify the audio signals into two basic types: silence and voiced sound. Then we move to the frequency domain approaches.

The Fourier transform itself does not help much since the rate of dimensionality reduction is only one half. The brightness and pitch derived from the Fourier transform coefficients are quite effective for distinguishing different timbres. But for those audio clips containing more than one timbre, which TV commercials usually are, this kind of features cannot supply a good discrimination.

Now consider the MFCC and NSCC. These two audio fingerprinting schemes share the basic idea that instead of directly utilizing the frequency domain information, convert it from hertz scale to other scales that are based on the human hearing's judgment. Mel scale is used in MFCC while bark scale is more appropriate for NSCC. The comparison of the performance of MFCCs and NSCCs has been reviewed in section 2.2.5. If the number of subbands is properly selected (around 15), these two features provide comparable low error rate in speech recognition. Moreover, NSSCs are more robust than MFCCs with respect to the noise. More discussion with regard to the computational complexity is demonstrated in this section.

The generations of the MFCCs and NSSCs begin with a few same steps: down sample the original signal to a predefined frequency, frame it with a specific rate of overlapping, window every frame by a Hamming window, and transform them into the frequency domain coefficients using the FFT. After these procedures, there are two more steps for MFCCs and only one for NSSCs.

To calculate the MFCCs, the power spectrum is mapped onto the mel scale using 20 triangular overlapping windows with the same area, applying the equation (2.13). It is followed by using (2.14) to get 12 MFCCs. This combination of bandpass filters and the dimension of features has been proven to give the highest performance with acceptable computations and storage. On the other hand, NCSSs are obtained by calculating the centroids for K critical bands of the human hearing, which is similar to a filtering process. Hence NSCC is a better choice compared to MFCC. Fig. 9 is an overview of how to calculate the NSSCs of the audio data.

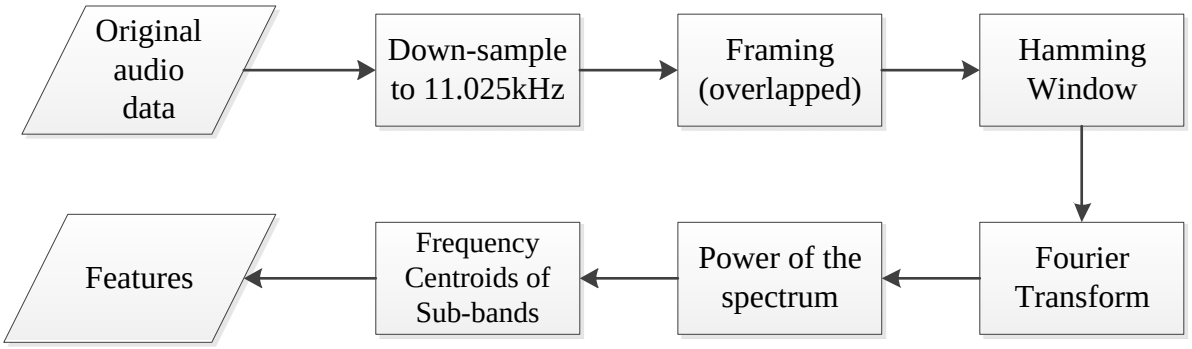

Fig. 9. Overview of audio-based feature extraction.

NSCCs of the audio data are derived as follows:

1. Convert the audio data to mono and down sample it to 11025Hz. It is because the critical subbands we are interested in range from 300Hz to 5300 Hz [28]. The boundaries of the subbands are listed as in Table 2.

Table. 2. The boundaries of 16 critical bands.

Critical band no. (i)	Lower frequency limit (f_l)	Upper Frequency limit (f_u)	Bandwidth (Δf)	Critical band no. (i)	Lower frequency limit (f_l)	Upper frequency limit (f_u)	Bandwidth (Δf)
1	300	400	100	9	1480	1720	240
2	400	510	110	10	1720	2000	280
3	510	630	120	11	2000	2320	320
4	630	770	140	12	2320	2700	380
5	770	920	150	13	2700	3150	450
6	920	1080	160	14	3150	3700	550
7	1080	1270	190	15	3700	4400	700
8	1270	1480	210	16	4400	5300	900

Theoretically, the whole audible frequency ranges from 0 to 16 kHz. It can be divided into 24 subbands according to the HAS. As discussed in [19], the best performance of SSC is achieved when there are about 15 subbands. For this reason, the most relevant 16 of them to the HAS are chosen to be the critical subbands in our algorithm.

2. Segment the signal into overlapping frames. In our application, we choose the width of the frame to be 4096 (approximately 371.5ms) and the overlap ratio is 50%. Overlap must be applied to promise the robustness to shifting.
3. Apply a Hamming window to each frame to minimize the discontinuities at the beginning and the end, shown as Fig. 10.

$$w(n) = 1 - a - a * \cos\left(\frac{2\pi n}{N-1}\right), where\ N = 4096, a = 0.46. \quad (4.1)$$

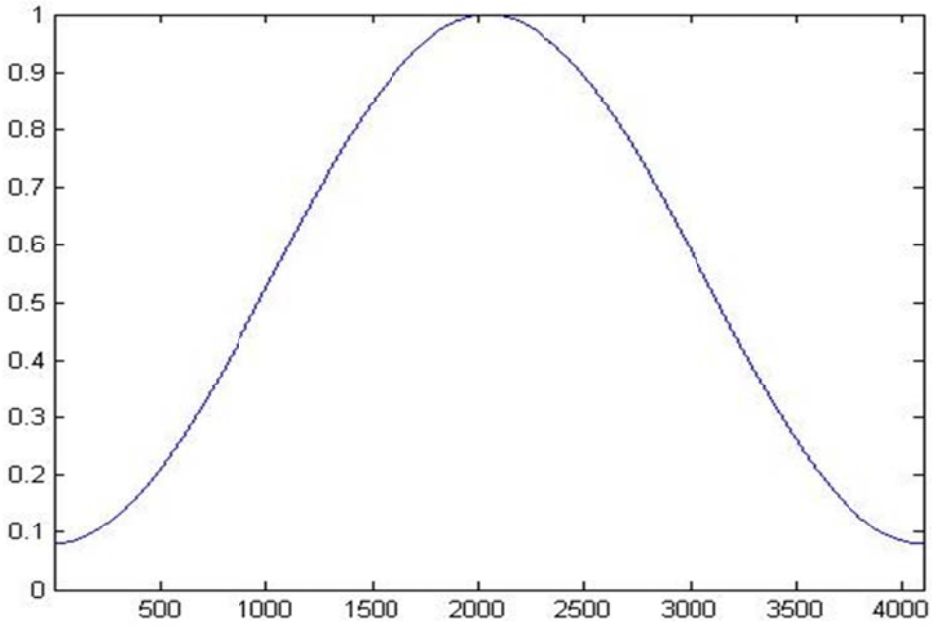

Fig. 10. The 4096-point Hamming window.

4. Transform the signal into the frequency domain using the FFT.

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi k \frac{n}{N}}, k = 0, \dots, N-1, \quad (2.5)$$

5. Take the power of the spectrum.

6. Calculate the normalized frequency centroids of the 16 critical bands. Let C_i be the i-th subband centroid for a frame, resulting

$$C_i = \frac{\sum_{k=CB_i+1}^{CB_{i+1}} k * P_k}{\sum_{k=CB_i+1}^{CB_{i+1}} P_k}, \mathrm{i} = 1, 2, \dots, 16, \quad (4.2)$$

where CB_i is the lower frequency boundary for the i-th critical band, P_k is the audio spectrum. Then it is normalized as follows:

$$NC_i = \frac{C_i - (CB_i + CB_{i+1})/2}{CB_{i+1} - CB_i}. \quad (4.3)$$

Each NSSC is a value between -0.5 and 0.5 regardless of the critical bands. For the computation simplicity and storage efficiency, all the NSSCs are further quantized to 256 values such that each one can be presented using only one byte.

4.2 Database Construction

Based on the audio fingerprint extraction scheme, the database of the known commercials is created for the search. The structure of the database is critical because it is closely related to the effectiveness and efficiency of the search algorithm.

For the first component of the database, the original commercials clips are cut from the TV streams directly. In practice, the source of the known commercials is diversified. They can be the standard samples provided by the companies, the

broadcasted versions modified by the stations, or the degraded files reported by the end-users. In several scenarios proposed before, these audio files were not stored after the features had been calculated for the database in consideration of the space they might take. Nevertheless, the copies are kept in our database so that the users have access to playing them, which is the most direct way to confirm the accuracy of the results.

The second component is the collection of all the features of the commercials in the database by using the extraction algorithm described above. Technically, this is the real “database” for the system. Every 16-dimension NSSC feature is a record and is assigned an index. Combined with the third element, the text file containing the basic information of the ads, the system can indicate which commercial a record comes from and its relative position to it.

The last piece is the file that contains the crucial parameters corresponding to the K-D tree structure and the indexes of the leaf nodes that each record belongs to. These parameters are some statistics data derived from all the features we have, including the maximum, minimum, median values of NSSC of each critical band and the discriminators of each level of the tree. If new commercials are accepted, all the components in the database need to be updated. The computation required to build and update the tree is easily derived. Let N denote the number of records in the database. Two exhaustive scans of the database are necessary and sufficient to carry out the task. The maximum, minimum, median values of 16-dim NSSCs are acquired during the first scan. The computations demanded are proportional to N, N and $N\log N$ respectively. These 48 bytes determine all the nodes of the tree, so in the second scan the leaf node

which a record belongs to is decided by comparing its components to the median values. The time it takes is just O(*N*).

4.3 Database Search

The search algorithm in the system takes three steps to accomplish the result: candidate search, decision, and time verification. Fig. 11 depicts the flow chart of the whole process.

From the flow chart we can see that generally the process is performed for every N_1 (N_1=15 is used in the application) frames when there is no match. Once a commercial is detected and confirmed, it jumps to the frame after the commercial and begins a new search.

For the candidate search, two cases are considered: single commercial and all commercials in the database. If the user selects a specific commercial to search, the system just compares the current frame with all the frames in this commercial using the square of the Euclidean distance to find any qualified candidates. If the user chooses to search all the commercials in the database, it applies the K-D tree method [26] to fulfill the task. The algorithm is depicted in Fig.12.

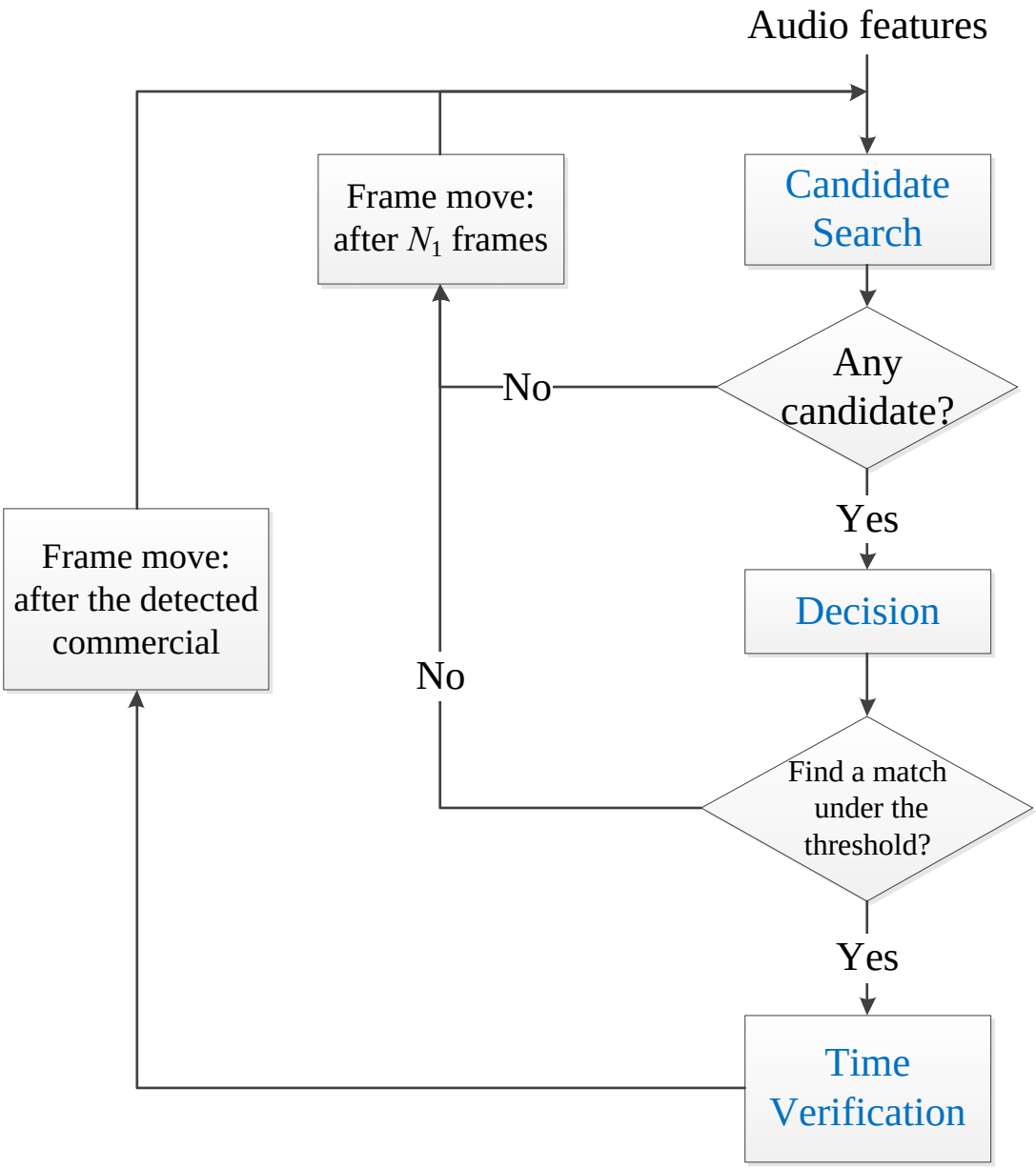

Fig. 11. The flow chart of the search process.

```
Global:
    X[16]: the NSSCs of the current frame
    Upper[16]: the upper bounds
    Lower[16]: the lower bounds
    Discriminator[16]:
    the discriminators for each level, each one is an index of the
    critical bands
    Partition[16]: the partition values for each critical band
Procedure SEARCH(node)
    local  p, d, temp, v;
    let v be the level of current node
    if (v is 16)
       examine all the features in this node, update the candidate set;
    else
       d=Discriminator[v]; p=Partition[v];
       if (X[d]<=p)
          temp=Upper[d]; Upper[d]=p;
          SEARCH(left son of node); Upper[d]=temp;
       else
          temp=Lower[d]; Lower[d]=p;
          SEARCH(right son of node); Lower[d]=temp;
       end
       if (X[d]<=p)
          temp=Lower[d]; Lower[d]=p;
          if (BOUND-OVERLAP-BALL)  SEARCH(right son of node);
          Lower[d]=temp;
       else
          temp=Upper[d]; Upper[d]=p;
          if (BOUND-OVERLAP-BALL)  SEARCH(left son of node);
          Upper[d]=temp;
       end;
    end;

Logical procedure BOUND-OVERLAP-BALL
    local sum, d;
    sum=0;
    for d=1 to d=16
        if (X[d]<Lower[d]) sum+=(X[d]-Lower[d])^2;
           if (sum>δ) return(FALSE);
        else if(X[d]>Upper[d]) sum+=(X[d]-Upper[d])^2;
           if (sum>δ) return(FALSE);
        end;
     end;
```

Fig. 12. The search algorithm for K-D tree.

It consists of two kernel procedures: search (node) and bound-overlap-ball function for the current node. There are six arrays involved in the entire execution path: X [16], Upper [16], Lower [16], Discriminator [16], Partition [16] and Candidate vector. Hence they are defined as global variables. In particular, Upper [16] and Lower [16] are initialized as the maximum and minimum values of the NSSCs of 16 critical bands in the database respectively and will be updated and modified during the recursive search function. On the other hand, X [16], Discriminator [16], Partition [16] are constant for the current frame and the database. Discriminator [16] are the discriminators of different levels of the tree and Partition [16] are the median values of the 16-dim NSSCs in the database.

The function search (node) is called recursively to return all the qualified candidates related to the leaves of the subtree of the node. It is operated as follows:

1. For the given node, calculate its level l in the tree to determine whether it is a leaf node or not. If it is a leaf node (l=16, assuming the level of the root node is 0), check all the candidate frames labeled as this node and return those with smaller distances to the X [16] than the threshold δ, then stop.
2. If the node is a nonterminal one, retrieve the l-th discriminator d and partition p from the Discriminator [16] and Partition [16]. Compare the X [d] with p. If X [d] is smaller than p, store the current Upper [d], change it to p and search the left subtree. Otherwise the current Lower [d] is preserved and updated to p, then search the right one.

3. After searching the selected subtree, the Upper [*d*] or Lower [*d*] is reset to the previous saved one. The procedure applies the bound-overlap-ball function to check the possibility that there are potential candidates in the other tree. The region associated with the other subtree is acquired by updating the Upper [*d*] or Lower [*d*] accordingly. The function bound-overlap-ball calculates the dissimilarity from X [16] to the closest boundary of the subtree under consideration. The distances along the 16 dimensions are computed one at a time. Hence if the region is far from X [16], the subtree can be excluded on the basis of a few NSSCs.

The third step guarantees that all the leaf nodes overlapping the ball centered at X [16] are investigated during the process. For each frame X [16], the system searches the entire database by calling function search (root) and updating the Candidate vector during the process. If the vector is empty, then the system skips to the next N_1-th frame. If there is any candidate, it moves to the next level: decision.

The versatility of this method makes its formal analysis quite difficult. Its performance certainly depends on the total number of records in the database *N*, the dissimilarity threshold δ, the number of records in the leaf nodes and the distribution of the records. Intuitively, as the size of the database *N* becomes bigger, the search will take more time. The number of records in one leaf node is closely related to *N* since the total number of leaf nodes is constant, 2^{16}. Because the median values are chosen to be the partitions, the records are highly possible to evenly distribute in these terminal nodes.

The choice of δ is crucial: a big δ results in a large amount of leaf nodes to be examined and candidates to further investigate in the next step while a small δ probably leaves some real commercial occurrences out mistakenly. Jin S. Seo, Minho Jin et.al [21] performed some analysis on the fingerprints in their database and gave some suggested parameters based on the experimental results. They assumed the SSC sequence as a stationary process and normalized it using (4.4) before modeling.

$$p[n] = \frac{SSC[n] - m}{\sigma}, \quad n = 1,2, \dots M\,, \tag{4.4}$$

where M is the number of SSCs in an audio block (M=848 in [21]), m is the mean and σ^2 is the variance. Then the authors calculated the first-order & second-order autocorrelations of the training data to build a simplified stochastic model as follows:

$$R[k] = E\big[p[n]p[n+k]\big] = a^{|k|}, \tag{4.5}$$

$$Q[k] = E\big[p^2[n]p^2[n+k]\big] = 1 + (\mu_4 - 1)b^{|k|}, \tag{4.6}$$

where a, b and μ_4 were revealed to be 0.59, 0.44 and 3.0 by the experimental results. After the modeling, the square of the Euclidean distance measure D, defined as in (4.7), was analyzed to determine a reasonable threshold for two similar blocks.

$$D = \frac{1}{M}\sum_{n=1}^{M}(p[n] - q[n])^2\,. \tag{4.7}$$

The mean of D is given as

$$E[D] = \frac{1}{M}\sum_{n=1}^{M} E[(p[n] - q[n])^2]$$

$$= \frac{1}{M}\left(\sum_{n=1}^{M} E\left[p^2[n]\right] + \sum_{n=1}^{M} E\left[q^2[n]\right] - 2\sum_{n=1}^{M} E\left[p[n]\right]E\left[q[n]\right]\right) = 2\sigma^2 + 0$$

$$= 2. \tag{4.8}$$

The variance of *D* is calculated as

$$\sigma_D^2 = E[D^2] - (E[D])^2, \tag{4.9}$$

where

$$E[D^2] = \frac{1}{M^2}E\left[\left(\sum_{n=1}^{M} p^2[n] + \sum_{n=1}^{M} q^2[n] - 2\sum_{n=1}^{M} p[n]q[n]\right)^2\right]$$

$$= 2 + \frac{2\mu_4 + 4}{M} + \frac{4}{M^2}\sum_{n=1}^{M-1}(M-n)[1 + (\mu_4 - 1)b^n + 2a^{2n}]. \tag{4.10}$$

Using the experimental values of a, b and μ_4, σ_D is given as 0.1479. Then the false alarm rate P_{FA} was estimated as in (4.11) under the assumption that *D* has a normal distribution if M is sufficiently large and the components in the sum are sufficiently independent.

$$P_{FA} = \int_{-\infty}^{T} \frac{1}{\sqrt{2\pi}\sigma_D} \exp\left(-\frac{(x-2)^2}{2{\sigma_D}^2}\right) dx = \frac{1}{2} erfc\left(\frac{2-T}{\sqrt{2}\sigma_D}\right). \tag{4.11}$$

It turned out that T could be quite large based on the model with a low probability to declare different audio blocks as similar.

For the decision step, the system checks K_1 (K_1=53 is chosen as the length is approximately 10 seconds) consecutive frames for each candidate. As described in Section 4.2, each frame is assigned a number which can indicate the commercial it

belongs to and its relative position to it. We can utilize this number to decide which frames near the candidates to choose. If there are less than K_1 frames in this commercial, all the frames in it are used to calculate the total distance. For those long enough commercial, if the current frame is the first half of it, the first K_1 frames are selected; otherwise the last K_1 ones are chosen. After computing the square of Euclidean distance for all the candidates, the system selects the one with the minimal distance and further verifies if this distance is under the predefined threshold. If it is too large, the process skips to the next N_1-th frame like that in the previous step. If it is small enough, it moves on to the final level: time verification.

At the time verification level, the first frame of the original commercial sample is selected as a header to locate the starting time accurately. After a proper shift, if necessary, it finishes the whole process for one frame.

Compared to the monitoring system proposed in [22], this process can be seen as a simplified version with little impact on the overall performance. Since this application is user-oriented, it is designed to offer more information and options to the users. In addition to the function of searching all the commercials in the database, the user can browse the list and choose any commercial to search in the current file. The result will be labeled directly on the waveform of the TV broadcasting data. Moreover, both the original commercial and the searched clips can be played so that the users can actually hear and confirm occurrences of the commercials.

5. EXPERIMENT RESULTS

5.1 The Interface Design

The application is written with C++ language and MFC. A computer with 2.9 GHz CPU is used in the experiment.

Fig.13 is the startup layout of the designed application. The interface is divided into 5 sub windows. We will introduce their functions and communication between each other.

The leftmost one is to automatically browse and list all the audio files stored under the specified file directory. They are displayed in a tree structure, representing the multi-story catalog. Through the folders' names, it is convenient to figure out when these file are broadcasted and their durations. Users can create new folders to organize their own audio resources. Each leaf node is a name of a WAVE format file. If the user wants to search commercials in one of the clips, just double click the name. Then the application will start loading this file and extracting its audio features. After the processing, the waveform of this audio clip will be displayed in the middle window on the right, as shown in Fig. 14.

The top window on the right is the list of all commercials in the database so that people can select a specific one to search for. It is connected to the left window in the bottom. Double click one commercial then the original version will be displayed in the bottom left. If there is a TV show file loaded before the selection, the application will spontaneously search the commercial in it. Otherwise, a notifying message pops up -“No show file loaded”. The number of matches will be indicated in the message. If there are any occurrences, they will be labeled in the audio file’s waveform, as shown in Fig.15.

There are two ways for the users to confirm the result. Double click the labeled part and the waveform of the short cut will be displayed in the last window, as shown in Fig.16. The user can compare it to that of the original commercial by dragging the horizontal scroll bar to see the whole shape, or just simply play both of them by clicking the “play” button.

Besides the single commercial search, the application supports the search for all the commercials in the database. After loading the TV show file, the user just need to click the “Test” button on the main menu. The format of the result report is shown in Fig.17.

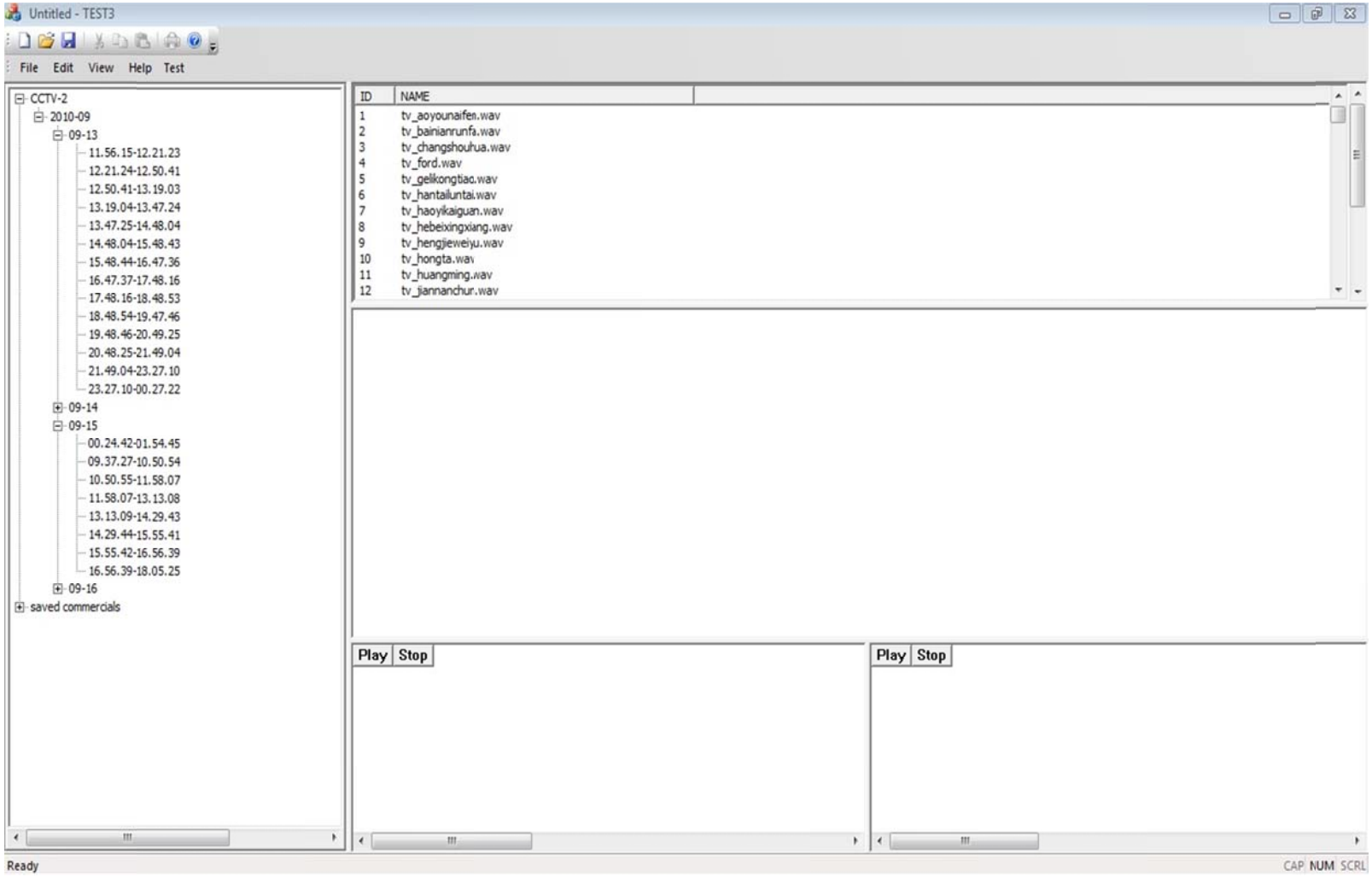

Fig. 13. The startup layout of the application

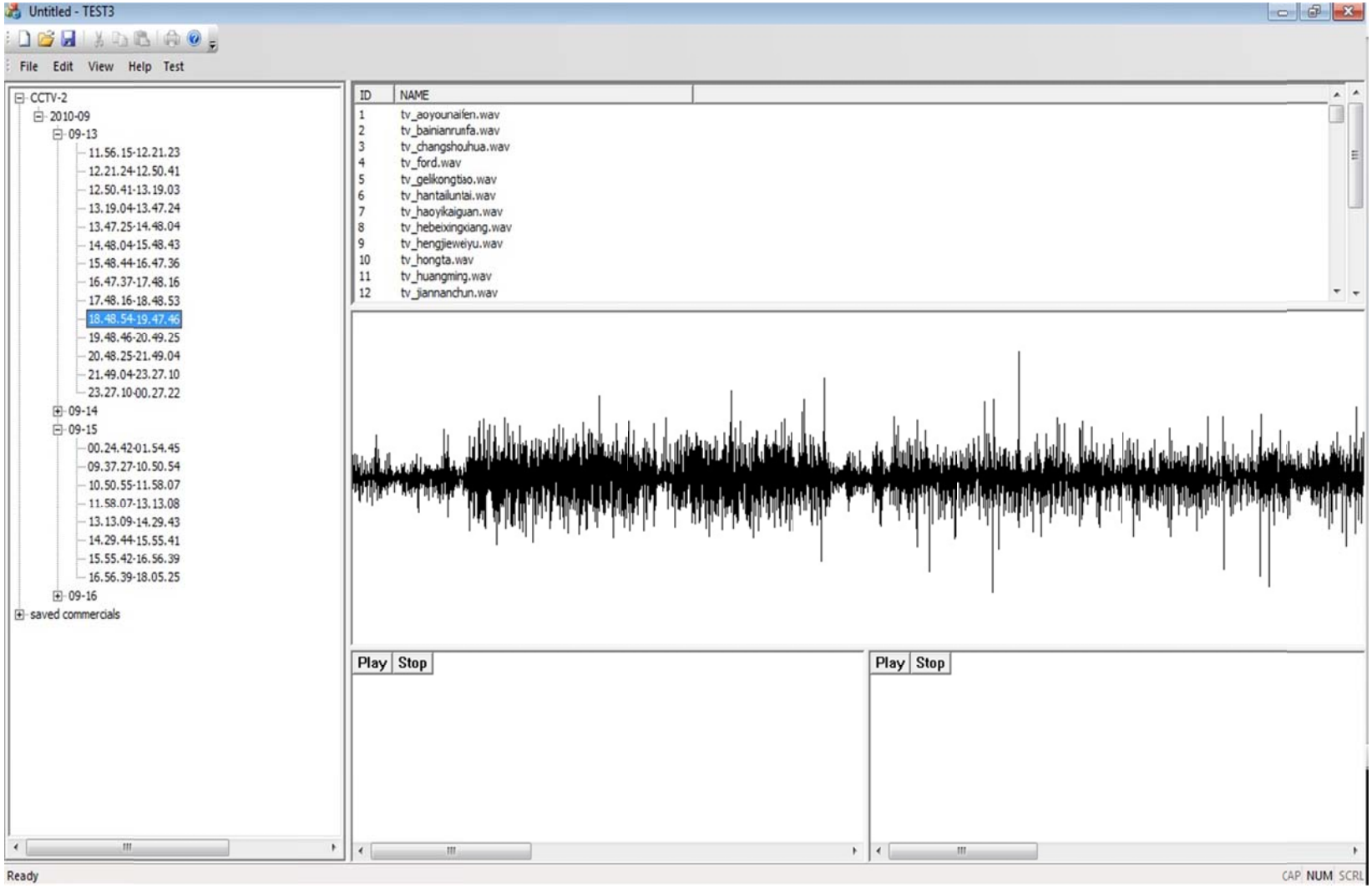

Fig. 14. The interface after loading and processing a selected audio file

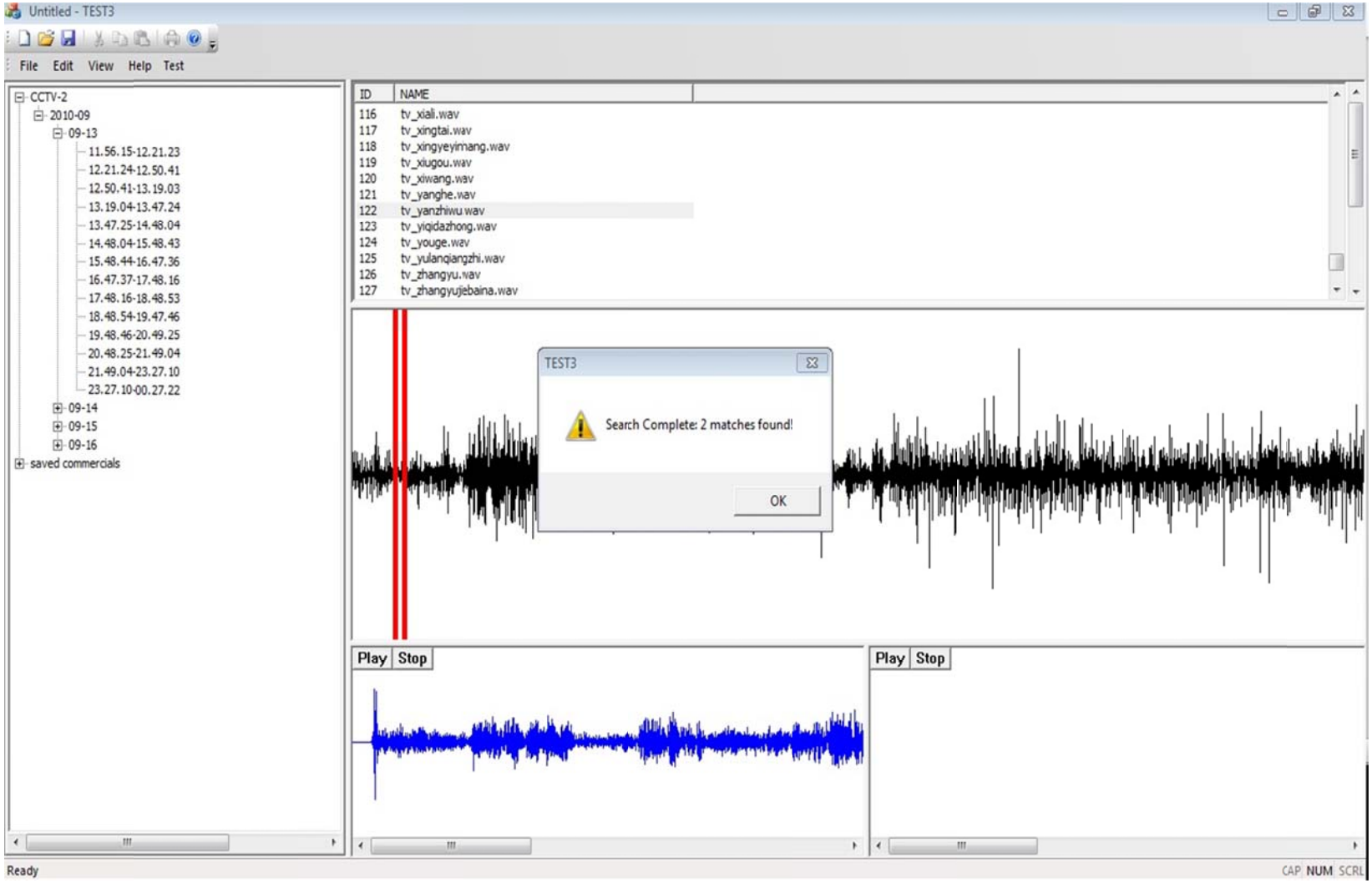

Fig. 15. The display of the search result of one commercial

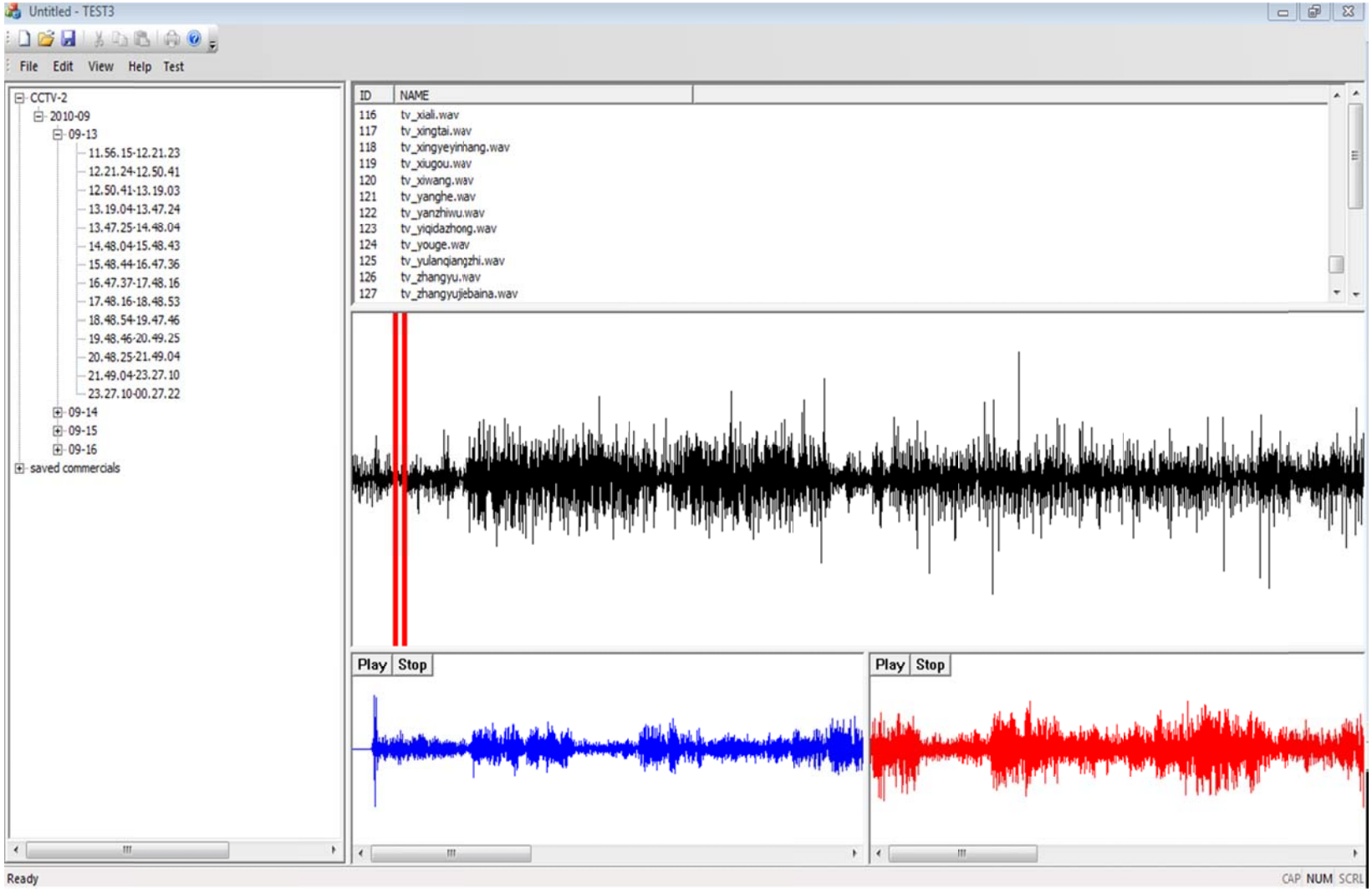

Fig. 16. The display of the result confirmation function.

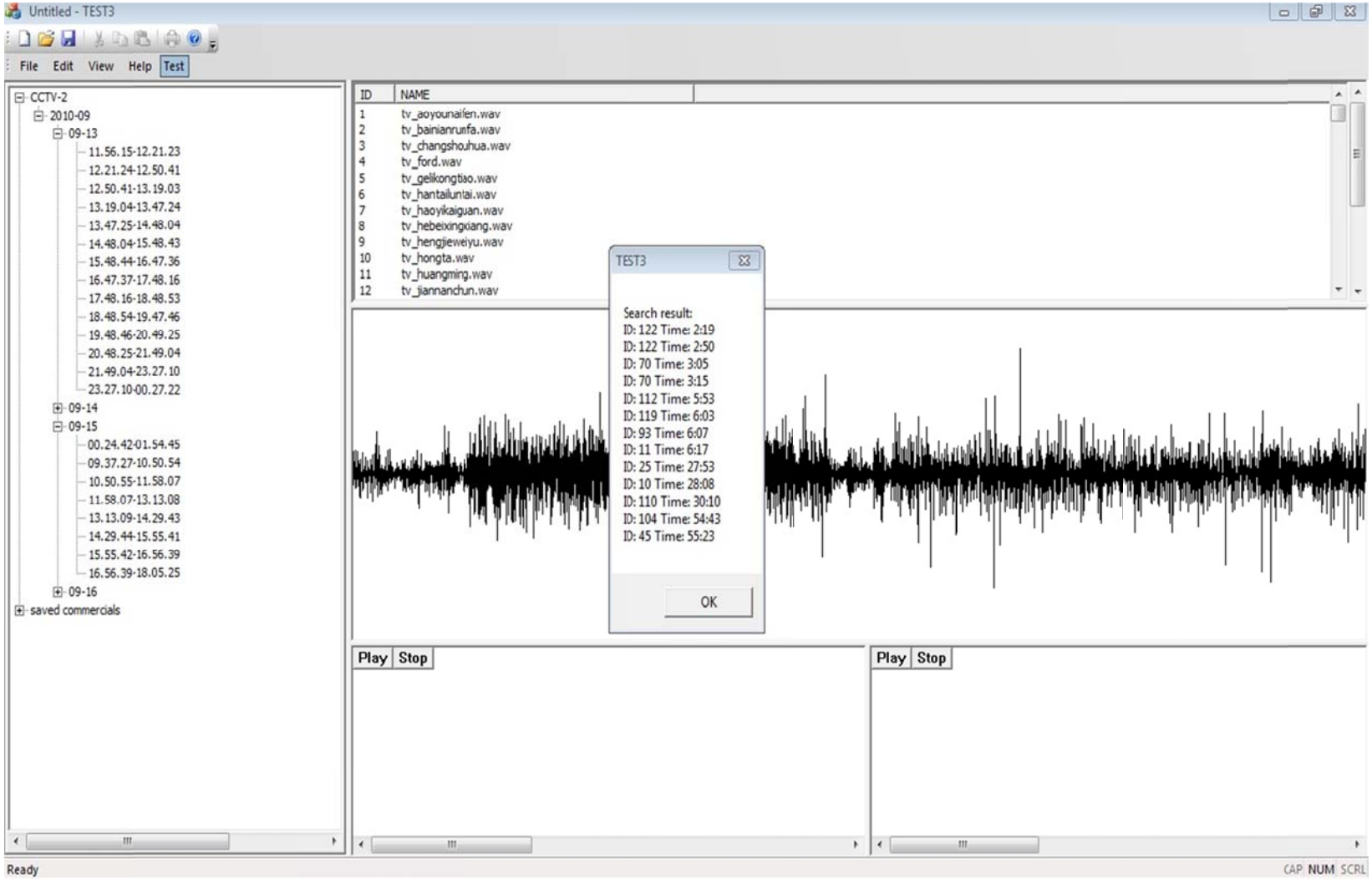

Fig. 17. The display of the search result of all the commercials.

5.2 Result Analysis

There are 126 different commercials in the database and their lengths are 5s, 10s, 15s, 20s or 30s. They are all gathered from the same station, CCTV-2. About 50 hours of recorded real broadcasting data is used for the test.

Based on the experiments results, an average processing time for the audio feature extraction from the audio data of an hour is about 45 seconds. The search speed for one commercial is fast, about 1-2 seconds. The average running time of the entire database search is about 100 seconds.

In the experiments, only the commercials in the database are recognized, and other versions are ignored. They are broadcasted 653 times in total in 50 hours. The system detects 640 among them, which is about 98.00%. No commercial is determined as another one and no TV program clips are recognized as commercials, which means the false alarm rate P_{FA} is 0.00%. Therefore this system is quite reliable. Table.3 lists the commercials that are broadcasted most frequently and their durations, occurrences and detections.

From the table we can see that the detection scheme performs quite well on most commercials. But for some commercials, the detection rates are not good enough.

Table. 3. Statistic results for some commercials.

Commercial Name	Duration(sec.)	Occurrences	Detections	Rate (%)
Yanzhiwu	15	26	26	100
Guomei	15	22	22	100
Shuiyushan	5	18	18	100
Xiali	15	15	15	100
Jiannanchun	15	14	14	100
Qipilang	15	13	13	100
Aoyounaifen	15	13	11	84.62
Shangdaitong	5	12	12	100
Kangjia	5	12	12	100
Dongfengrichan	15	12	12	100
Yangxueqingnao	5	12	11	91.67
Minsheng bank	5	12	11	91.67
Minsheng credit card	5	12	10	83.33
Lanling	10	11	11	100
BYD_F6	10	11	11	100
Xinhuiteng	30	11	11	100
Kangjie	10	11	11	100
Benchi	30	11	11	100
Qicaiyunnan	5	10	10	100
Meidizhenglifang	15	10	10	100

Table. 3. Continued

Commercial Name	Duration(sec.)	Occurrences	Detections	Rate (%)
MDX	30	9	9	100
Aidengbao	15	9	8	88.89
Hongloumengjiu	10	9	8	88.89

Observing the broadcasting time of these commercial, we find out that most of the detection failures happen when the commercial is broadcasted between others and its duration is quite short. Recall several parameters used in the proposed algorithm. In the audio feature extraction part, the audio data is down sampled to 11.025 kHz. The length of every frame is 4096 and the overlap ratio is 50%. So there are actually 2048 new points in a frame, which is about 0.2 sec. This is also the maximum error of time shift for one frame. Although the time verification is applied in the data search part, this error is still likely to propagate due to the fact that the basic unit of search is a frame. When the cumulated error is large enough, the whole commercial may just be skipped because of the short duration.

5.3 Discussion and Future Work

Most existing TV commercial algorithms and products apply the vector quantization in the system to train the broadcasting data. At the expense of more training time on the data, the search speed is really fast. Typically, it takes 150 seconds to preprocess the broadcasting data of an hour and 1 sec to search for all the commercials in the database. The state-of-art product can detect all the occurrences of known

commercials (100% detection) with $P_{FA} = 1.9\%$. Compared to this product, our system sacrifices little robustness for better reliability. However, robustness is more practically important since it is easier to report the false alarm by reviewing than to find out the missing one in the broadcasting stream. Proper adjustment of the dissimilarity threshold is likely to increase the probability of detection.

In practice, considering the cost of broadcasting, companies always tend to adjust the lengths of their commercials for different stations. It leads to various versions of the commercials for the same product. As the system assumes the match is of the same length, it is required to store all the versions in the database to guarantee the search for some specific product is complete.

In this system, the commercials in the database are typical. Every one contains both audio and video components and most of them have background music and speech. However, there are other special cases in the real life. Some commercials only have the video part. Some only contain the speech element, which are quite similar to TV programs, like news report. And others may just play the music part which is used in some different commercial. Under these conditions, this system will be vulnerable. It is necessary to take advantage of the video information for further improvement.

6. CONCLUSION

A commercial search system for TV broadcasting is presented using audio fingerprints. The application selects Normalized Spectral Subband Centroids (NSSCs) as the audio features, and constructs a database and designs search algorithms for different cases. It is implemented based on C++, using Microsoft Visual Studio MFC for the interface. The application can search either one specific commercial as the user request or all the commercials in the database. The detection is quite accurate for most commercials but not good enough for some specific ones. Future work is needed to improve the performance.

REFERENCES

[1] P. Cano, E. Batlle, T. Kalker, and J. Haitsma, “A review of algorithms for audio fingerprinting,” *in Proc. IEEE Workshop on Multi. Signal Process.*, pp. 169-173, Dec. 2002.

[2] E. Wold, T. Blum, D. Keislar, and James Wheaton, “Content-based classification, search, and retrieval of audio,” *IEEE Multimedia*, vol. 3, no. 3, pp. 27-36, Fall 1996.

[3] T. Zhang and C.- C. J. Kuo, “Hierarchical classification of audio data for archiving and retrieving,” *in Proc. ICASSP*, vol. 6, pp. 3001-3004, Mar. 1999.

[4] C. Burges, J. Platt, and S. Jana, “Extracting noise-robust features from audio data,” *in Proc. ICASSP*, vol. 1, pp.1021-1024, May 2002.

[5] Mari Riess Jones, Richard R. Fay, and Arthur N. Popper, *Music Perception*, New York, Springer, 2010.

[6] D. Gernhard, “Pitch extraction and fundamental frequency: history and current techniques,” *Technical Report TR-CS 2003-6*, University of Regina, 2003.

[7] A. Michael Noll, “Pitch determination of human speech by the harmonic product spectrum, the harmonic sum spectrum and a maximum likelihood estimate,” *in Proc. Symp. Comp. Process. Comm.*, vol. 19, pp. 779-797, 1969.

[8] Patricio de la Cuadra, Aaron Master, and Craig Sapp, “Efficient Pitch Detection Techniques for Interactive Music,” *in Proc. Intern. Comp. Music Conf.*, 2001.

[9] S. S. Stevens, J. Volkmann, and E. B. Newman, “A scale for the measurement of the psychological magnitude pitch,” *Journal of the Acoustical Society of America*, vol. 8, no. 3, pp. 185-190, 1937.

[10] Fang Zheng, Guoliang Zhang, and Zhanjiang Song, “Comparison of different implementation of MFCC,” *Journal of Computer Science and Technology*, vol. 16, no. 6, pp. 582-589, Nov. 2001.

[11] Jonathan T. Foote, “Content-based retrieval of music and audio,” *in Proc. SPIE*, vol. 3229, pp. 138-147, Oct. 1997.

[12] Jonathan Foote, “An Overview of Audio Information Retrieval,” *Multimedia Systems*, vol. 7, no.1, pp. 2-10, Jan. 1999.

[13] H. Ozer, B. Sankur, and N. Memon, “Robust audio hashing for audio identification,” *12th European Signal Process.*, vol. 6819, pp. 2091-2094, 2004.

[14] Kuldip K. Paliwal, “Spectral subband centroid features for speech recognition,” *in Proc. ICASSP*, vol.2, pp. 617-620, May 1998.

[15] B. Gajic and K. K. Paliwal, “Robust feature extraction using subband spectral centroid histograms,” *in Proc. ICASSP*, vol. 1, pp. 61-64, 2001.

[16] S. Tsuge, T. Fukada, and H. Singer, “Speaker normalized spectral subband parameters for noise robust speech recognition,” *in Proc. ICASSP*, vol. 1, pp. 285-288, 1999.

[17] D. Albesano, R. De Mori, R. Gemello, and F. Mana, “A study of the effect of adding new dimensions to trajectories in the acoustic space,” in *Proc. EUROSPEECH*, vol. 4, pp. 1503-1506, 1999.

[18] J. Haitsma and T. Kalker, “A Highly Robust Audio Fingerprinting System,” *in Proc. ISMIR 2002*, Oct. 2002.

[19] J. Chen, Y. Huang, Q. Li, and K. K. Paliwal, “Recognition of noisy speech using dynamic spectral subband centroids,” *IEEE Signal Process. Letter*, vol. 11, no. 2, pp. 258-261, Feb. 2004.

[20] Minho Jin and Chang D. Yoo, “Temporal dynamics for spectral subband centroid audio fingerprints,” *Multimedia and Expo, 2007 IEEE Intern. Conf.*, pp. 180-183, July 2007.

[21] J. S. Seo, M. Jin, S. Lee, D. Jang, S. Lee, and C. D. Yoo, “Audio fingerprinting based on normalized spectral subband centroids,” *in Proc. ICASSP 2005*, vol. 3, pp. 213-216, Mar. 2005.

[22] D. Jang, S. Lee, J.S. Lee, M. Jin, J.S. Seo, S. Lee, and C.D. Yoo, “Automatic commercial monitoring for TV broadcasting using audio fingerprinting,” *AES 29th Intern. Conf.*, Seoul, Korea, Sep. 2006.

[23] V. Kitanovski, D. Taskovski, S. Bogdanova, “Application for real-time TV commercial monitoring based on robust visual hashing,” *Visual Inform. Process. (EUVIP), 2010 2nd European Workshop on*, pp. 140-143, July 2010.

[24] C. Bohm, S. Berchtold, and D.Keim, “Searching in a highdimensional spaces: index structures for improving the performance of multimedia databases,” *ACM Computing Surveys*, vol.33, no.3, pp. 322-373, Sep. 2001.

[25] J. L. Bentley, “Multidimensional binary trees used for associative searching,” *Comm. of the ACM*, vol. 18, no. 9, pp. 509-517, Sep. 1975.

[26] J. H. Friedman, J. L. Bentley, and R. A. Finekl, “An algorithm for finding best matches in logarithmic expected time,” *ACM Trans. on Math. Software*, vol. 3, no. 3, pp. 209-226, 1977.

[27] AoAmedia website < http://www.aoamedia.com/audioextractor.htm>, Sep. 2011.

[28] E. Zwicker and H. Fastl, *Psychoacoustics: Facts and Models*, New York, Springer-Verlag, 1999.

VITA

Name: Yaohua Song

Education: Master of Science (August 2009 – May 2012)

Major: Electrical Engineering

Texas A&M University, College Station, TX, 77843

Bachelor of Science (September 2005 – June 2009)

Major: Information Security

University of Science and Technology of China, Hefei, China

Email Address: feicunxun@tamu.edu

Permanent

Address: 236B Zachry Engineering Center, TAMU 3128

College Station, Texas 77843-3128