NTNU
Norwegian University of
Science and Technology

Using the Signature Quadratic Form Distance for Music Information Retrieval

Håkon Haugdal Hitland

Master of Science in Computer Science
Submission date: June 2011
Supervisor: Magnus Lie Hetland, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Problem Description

The Signature Quadratic Form Distance (SQFD) is a recently introduced distance measure for content-based similarity. It makes use of feature signatures, a flexible way to summarize the features of a multimedia object.

Though feature signatures can represent various types of content, most of the research on SQFD to date has focused on image retrieval.

Investigate how the SQFD can be applied to music information retrieval tasks, e.g., query by humming.

Assignment given:	15. January 2011
Supervisor:	Magnus Lie Hetland

Abstract

This thesis is an investigation into how the signature quadratic form distance can be used to search in music.

Using the method used for images by Beecks, Uysal and Seidl as a starting point, I create feature signatures from sound clips by clustering features from their frequency representations. I compare three different feature types, based on Fourier coefficients, mel frequency cepstrum coefficients (MFCCs), and the chromatic scale.

Two search applications are considered.

First, an audio fingerprinting system, where a music file is located by a short recorded clip from the song. I run experiments to see how the system's parameters affect the search quality, and show that it achieves some robustness to noise in the queries, though less so that comparable state-of-the-art methods.

Second, a query-by-humming system where humming or singing by one user is used to search in humming/singing by other users. Here none of the tested feature types achieve satisfactory search performance. I identify and discuss some possible limitations of the selected feature types for this task.

I believe that this thesis serves to demonstrate the versatility of the feature clustering approach, and may serve as a starting point for further research.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the prevalence of computers, search has become a part of everyday life for most people. The most established ways to search for information today is in structured data, such as a database, or in unstructured text, such as a web search engine.

However, we sometimes want to search for content in ways that do not fit well with the textual search paradigm. Maybe you only remember the melody from a certain song, and want to find the title. Or maybe you want to find all the photos from your vacation that show a beach.

With these applications in mind, research is being done on various techniques for content-based search, where instead of basing the search on manually entered descriptions, the search is done by looking for similarity between the content and the query. Popular content types are images, audio or video.

Research in content based search started picking up pace in the nineties; Ghias et al. [11] published their influential paper on a query-by-humming system in 1995, and this decade was also a turning point for content based image retrieval [24].

We are currently at a stage where this technology is starting to reach consumers. The Shazam music service, which lets people search a music database by recording a sound clip with their phone, was launched in 2002 [9]. Google made a "similar images" feature available as part of their image search in 2009 [15]. We are likely to see more and more of these services appear as advances in the field make them viable.

Often, advances in one field of research can be also be adapted to related problems. The earth mover's distance (EMD) [23] was originally introduced to measure distance between images, but has since been applied to a variety

of content types, including music [20] and text [29].

The signature quadratic form distance (SQFD) [2] is a more recently introduced distance measure with many similarities to the EMD. Advantages over the EMD include indexability [21] and lower computation time. However, no research has as of yet tried to apply the SQFD to music information retrieval.

In this thesis I investigate the suitability of the SQFD for music retrieval tasks.

Specifically I will investigate the following research questions:

Can the SQFD be used to build an audio fingerprinting system?

Can the SQFD be used to build a query-by-humming system?

I chose these problems largely because they can both be implemented with a relatively simple framework around a melodic similarity function. They also have the advantage of allowing quantitative tests against a ground truth, which is not the case for more subjective 'playlist generation' tasks as seen in [20].

My focus will be on investigating the parameters of the system and the effect they have on search quality. I will *not* be considering the speed and efficiency of the search. However, as the distance function I am using is known to support efficient search through indexing, modifying the system to be more efficient should be a relatively straight-forward task.

The structure of the thesis is as follows: Chapter one, the introduction, presents the research questions and gives an overview of the thesis structure. Chapter two, background, aims to give an overview of the concepts that will be used in developing my system. Chapter three, design, gives an overview of the search system, and explains my method for creating feature signatures from audio clips. Chapter four, experiments, explains how I test the performance of the system. Chapter five, results, shows how the system performs with varying parameters. Finally, I present my conclusion and further work in chapter six.

Chapter 2

Background

2.1 Music information retrieval

Ever since the advent of the computer age the amount of stored digital information in the world has been growing at an exponential rate, to the point where our ability to utilise the vast amounts of data available mostly depends on what methods we have to sift and search through it to discover information relevant to us.

This is also the case with music; the large music services iTunes and Spotify both claim over 13 million songs available. These large music databases must also maintain metadata for each piece of music to allow customers to search for the tracks they want, typically by artist, title or genre. But manually maintained textual metadata has its limits. Customers may want to find songs they have heard but do not know the titles of. Or they could be looking for more songs in the style of their current collection.

The field of music information retrieval (MIR) deals with ways to extract data from the music files themselves. The tasks range from focused searches, like searches for a specific song, or detecting plagiarism, to broad classification like tempo detection, genre classification and "similar song" functionality [7].

The music data is typically stored either in symbolic form, like MIDI format, or as a audio recording. While the earliest efforts on the topic mostly focused on symbolic data, MIR tasks in sound data are increasingly a focus [27].

Music tasks relevant for this thesis are:

Audio fingerprinting concerns recognising a previously seen audio clip given a signature of either the whole clip, or a sample of it. Besides simple

search, it is used in applications such as automatic music tagging, and in filtering distribution channels for copyright protected material [7]. Most real-world tasks are complicated because the query will differ from the source file due to noise, distortion and lossy compression [1] [12].

Shazam [30] is one example of an audio fingerprinting system. Shazam is a commercial offering from Shazam Entertainment Ltd., providing identification of recorded music from mobile devices. The service identifies recordings based on local peaks in the frequency spectrum. It uses hashing to index local pairs of such peaks in the music database. When looking up a query, it looks for a sequence of peak pairs in the query that also occur in the same sequence somewhere the database.

Query-by-humming concerns finding a song based on a singing or humming query from an user. For search applications this often more convenient for the users, who will seldom have a recording at hand of the very file they want to find. Unfortunately, this is also a considerably harder task than audio fingerprinting. Users are likely to transpose, sing off-key, and drop or insert extra notes [5].

Most of the early systems, like the one by Ghias et al. [11], try to transcribe the user query to notes before searching the database, which adds the problem of inaccurate transcription. Most systems to date have stored the database in symbolic format, though some attempts have been made at searching in audio databases, with varying results [19, 25].

An interesting approach, used by services like Soundhound and Tunebot [13], consists of matching queries against recordings of other users singing the melodies. This lets them "crowdsource" the problem of transcribing melodies to their users.

The International Society for Music Information Retrieval (ISMIR) holds an annual international conference on the topic of music information retrieval. Coupled with this, an large evaluation campaign known as the Music Information Retrieval Evaluation eXchange (MIREX) is held. MIREX comprises a wide range of music information retrieval tasks, including a query-by-singing/humming task.

2.2 Signature quadratic form distances

The SQFD is a way to measure the similarity between two objects.

The SQFD works on *feature signatures* consisting of sets of points, where

each point has a weight and a set of coordinates. If the points are generated by clustering, they are also called *weighted centroids*. I use the definition of a feature signature from [2], page 697:

Definition 1 *Given any feature space $\mathcal{FS} \subseteq \mathcal{R}^k$, we define a feature signature Q of length n^q as a set of tuples from $\mathcal{FS} \times \mathcal{R}^+$:*

$$Q := \{\langle c_i^q, w_i^q \rangle, i = 1, \ldots, n^q\}$$

The same page also gives the definition of the SQFD itself:

Definition 2 *Given two feature signatures $Q = \{\langle c_i^q, w_i^q \rangle |\ i = 1, \ldots, n\}$ and $P = \{\langle c_i^p, w_i^p \rangle |\ i = 1, \ldots, m\}$, the Signature Quadratic Form Distance $SQFD_A$ between Q and P is defined as:*

$$SQFD_A(Q, P) := \sqrt{(w_q| - w_p) \cdot A \cdot (w_q| - w_p)^T},$$

where $A \in \mathcal{R}^{(n+m)\times(n+m)}$ is the similarity matrix, $w_q = (w_1^q, \ldots, w_n^q)$ and $w_p = (w_1^p, \ldots, w_m^p)$ are weight vectors, and $(w_q|\ -w_p) =$ $(w_1^q, \cdots, w_n^q, -w_1^p, \ldots, -w_m^p)$.

To make the similarity matrix, we must define a similarity function that gives the similarity between two centroids. Beck et al. present three different distance functions, and conclude that of them, the Gaussian function

$$f_g(x, y) = e^{-\alpha \cdot d^2(r_i, r_j)}$$

yield the highest retrieval performance for their tests. [4] For the SQFD evaluations in this thesis I have used the Gaussian similarity function with $\alpha = 1$.

Figure 2.1 gives some insight in the behaviour of the SQFD distance. We can see that if the change we make to the signature is very small, the distance to the original will also be very small, and in general, the distance between "similar" distances will be small.

Several other distance functions for feature signatures exist besides the SQFD. Among the well known ones are the Hausdorff distance [14], the weighted correlation distance [18], and the EMD [23]. A comparative study of such distances can be found in [3].

Beecks, Uysal and Seidl introduced SQFDs as a generalisation of quadratic form distances (QFDs), and applied it to image similarity searching. [2] They

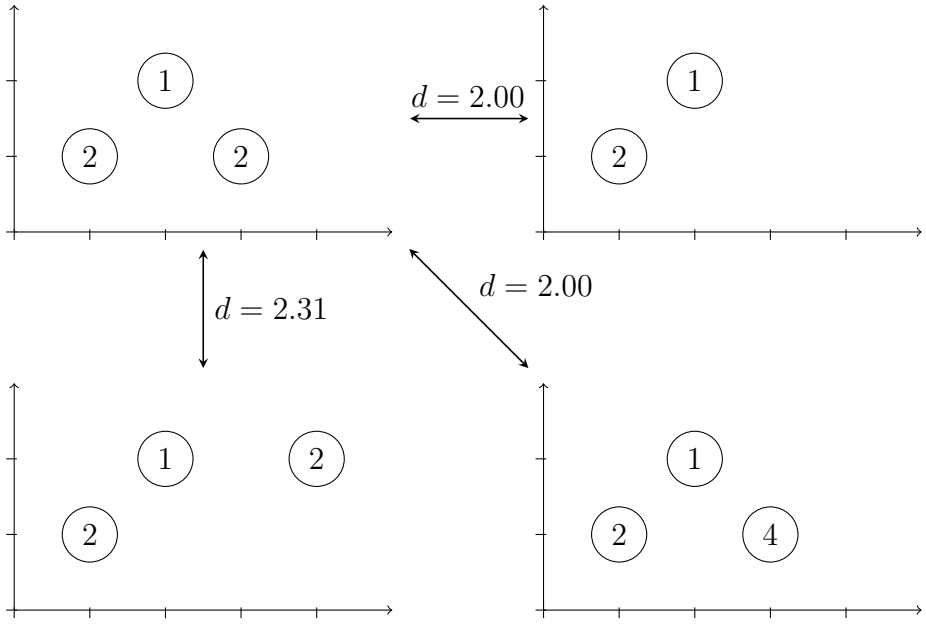


Figure 2.1: Some feature signature and their distances with the SQFD. Centroids are drawn at their position in a two-dimensional feature space, with a number denoting their weight.

show the SQFD measure to be significantly faster to calculate than the EMD while providing comparable search results for the image retrieval task. To generate a feature signature for an image, they use k-means clustering on the individual pixels, where each pixel include position, colour, contrast and coarseness information. The centroid of each clusters become a point, with the number of pixels in the cluster as its weight.

Another relevant use of feature signatures is found in [20], where Logan and Salomon use the EMD to measure similarity between songs. They generate a signature for each song by splitting the song into frames, finding the mel-frequency cepstral coefficients (MFCC) of each frame, and clustering the MFCCs using k-means clustering. The clusters are then converted to a signature just like for the image.

Feature signatures has also been used on symbolic music representations. Typke et al. describe a melodic similarity measure using a variant of the EMD [26], and use this as a part of a query-by-humming system [28]. Their approach differs from the typical histogram-based usage in that they do not generate the signatures by clustering. Instead, they represent each note as a weighted point, where the features of the point are the onset time and pitch

delta, and the weight is the duration of the note.

One property of the SQFD that make it well suited for search applications is that is is *indexable*; this makes it possible to find signatures resembling a query by looking at only a small subset of the signatures in a database. The SQFDs is indexable due to being a *Ptolemaic metric* [21]. This means it satisfies two sets of constraints, both of which can be used to speed up searches.

A distance function $d(x, y)$ is a *metric* if:

- $d(x, y) = 0 \leftrightarrow x = y$; The distance is 0 only for identical objects.
- $d(x, y) = d(y, x)$; The distance is symmetric.
- $d(x, z) \leq d(x, y) + d(y, z)$; The distance satisfies the triangle inequality.

A distance function $d(x, y)$ is a *Ptolemaic distance* if it satisfies *Ptolemy's inequality*:

$$d(x, v) \cdot d(y, u) \leq d(x, y) \cdot d(u, v) + d(x, u) \cdot d(y, v)$$

2.3 Sound processing

The natural format for recording and playback of sound on a computer is a time series of samples from the wave signal, but while this format is easy to record and store, it is not well suited for signal analysis, as slight perturbations can change the signal at this level drastically, and similarities that are apparent to the ear can be hard to spot. The human ear does not sample the waveform, but instead detects sounds as combinations of frequencies. A system that is intended to reflect human perception of melodies should therefore also use the frequency domain representation as a basis.

A well-known way of converting a digital signal from the time domain to the frequency domain is the discrete Fourier transform. By applying the discrete Fourier transform to overlapping windows of the signal, we get a picture of the frequency spectrum over time. The Fourier transform works on complex numbers. When analysing the signal we use only the magnitude, discarding the phase information in the signal, which is encoded in the angle of the coefficient. To reduce *spectral leakage*, distortion in the Fourier signal caused by the windowing, windows should be multiplied by a *windowing function*. One such window is the *Hamming window*.

The Fourier coefficient representation is useful for many audio analysis applications, but when used for music a couple of problems show up: If one frequency is present in the signal, the harmonic frequencies will also appear, which makes it harder to determine the fundamental frequencies. And it does not reflect the logarithmic nature of the octave scale.

The mel cepstrum [22] is intended to solve these problems. After first taking the magnitudes of the Fourier transform, the resulting signal is mapped onto the (roughly logarithmic) mel scale, the power is mapped to the logarithmic scale, and finally the discrete cosine transform, another frequency domain transform, is applied to get the result. The additional frequency transform consolidates the harmonics with their fundamental frequency, and the transform to the mel scale makes the signal more correlated to the experience of a human listener.

A related representation of sound that is also based on human perception is the chromatic system [17, 8]. It uses the musical concepts of an *octave*. To increase a tone by one octave, the frequency is doubled. An octave can be divided into 12 semitones on a *chromatic scale*. Several different chromatic scales have been used in music, but for the purposes of search we use equal temperament, where the frequency ratio between adjacent semitones is a factor of $\sqrt[12]{2}$. Pitches that are one or more whole octaves apart are experienced as very similar, an effect called *octave equivalency*. In the pitch class model, each frequency is mapped to one of the 12 semitones on the scale, discarding octave information.

Chapter 3

Design

In this chapter I describe the music search system I have developed. An overview of the system is shown in Figure 3.1.

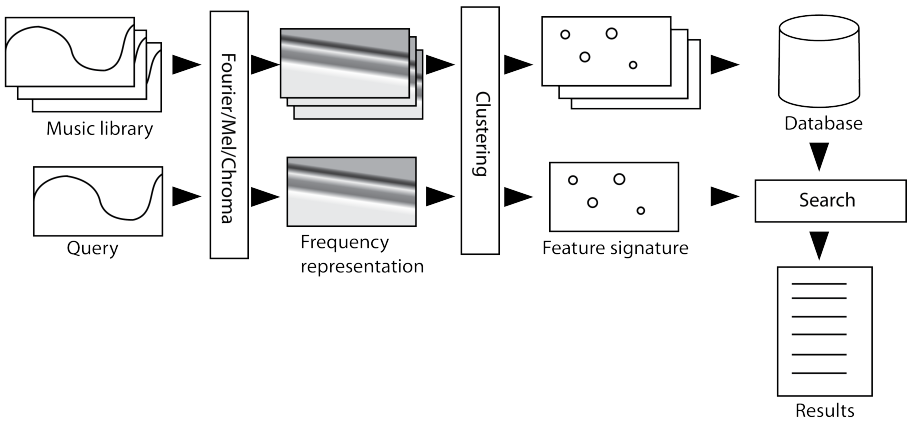


Figure 3.1: An illustration of the search process

The basic framework is quite simple. To perform a search, the query is converted to a feature signature, and all feature signatures in the database is ranked according to their distance to the query signature. The difficulty thus lies in converting the database and query files to signatures.

To use the SQFD to index music, we must first device a way to convert a sound clip into a feature signature. A natural starting point is the paper of

Beecks et al. [2], where they create feature signatures for images by using k-clustering. I have chosen to use a very similar approach, using features from the frequency domain.

A sliding window of length W samples is used to pick windows for frequency analysis. The step size of the window is $\frac{W}{2}$, to smooth features in the spectrum. The window is multiplied with a Hamming window and then frequency transformed. The $V-1$ first values, corresponding to the lowest frequencies, are saved in a vector, together with a position, weighted by P_W. This is similar to how Logan et al. stores coefficients [20]. The vectors from each window are then clustered with k-means clustering, where the number of clusters is C. Finally, the centroid of each cluster becomes a point in the signature, with weight equal to the number of vectors in the point.

3.1 Search modes

The mode of operation is slightly different for the audio fingerprint search, and the query-by-humming search.

3.1.1 Audio fingerprint

For the audio fingerprint task, we would like to be able to identify a snippet from any point in the file. For this I use a sliding window with a fixed step size of 0.25 seconds. I generate one signature at each position, and store it in the database with an association to its originating file.

This overlapping windowing trick is a standard technique for audio fingerprinting [6], however for this application it is typical to use much smaller windows and step sizes, as the features they observe are on a smaller scale.

To perform a search, I compare the query signature with all signatures in the database, and rank each file according to the *closest* of the signatures associated with it.

It is possible to be smarter about this, for instance by windowing the query as well, and searching the database for sequential matches. Some such tricks are detailed in [1]. However, in addition to taking time to implement, this would add complexity to the system, and may obscure the dynamics when investigating the setting of the parameters. As the implementation of these techniques will necessarily have a codependency with the partitioning of the data and the efficency of lookups, I feel that they are more naturally inves-

tigated together with indexing.

3.1.2 Query-by-humming

The specific query-by-humming task I will be investigating is that of comparing the humming of two users. For this task, all queries will start at the same position (the start of the song) and be of the same length. These condition match those of the variants query-by-singing/humming subtask in MIREX 2009 and 2010.

The organization of the query-by-humming database is slightly simpler than for the audio fingerprinting. As the queries I compare all start at the same position and have the same length, only one signature is associated with each query. Excepting that difference, the search proceeds just as with the fingerprint.

3.2 Feature types

I have tested three different *feature types*, all based on the frequency domain.

3.2.1 Fourier

The first of the feature types is based on the Fourier transform.

The Fourier transform is one of the most obvious and well-known ways of transforming a signal to the frequency domain, and therefore serves as a nice baseline.

The window is first transformed using the standard discrete fast Fourier transform. The values are converted from complex to real by taking the absolute value, and then mapped to the logarithmic scale.

$$v_i = \log_{10}(|X_i|)$$

I discard the zeroth value, containing the DC component.

3.2.2 MFCC

The second feature type I have tested is based on the mel frequency cepstral coefficients.

The MFCC is the feature used by Logan et al. in [20], and seems to be a common feature choice for sound search [10, 7].

To get the MFCCs we first do a discrete Fourier transform on the window and take the absolute values of the result, just like for the Fourier based feature. I then run a triangular filter bank over the result to rescale the frequencies to the mel scale. For simplicity I have used an exponential mel curve. The number of mel coefficients is determined by the number of filters in the filter bank. Logan et al. used 40 cepstral coefficients, while Casey et al. report that around 20 is the usual value [7]. I have used a value of 30 for my experiments.

The values from the filter bank is run through a discrete cosine transform to get the final cepstral coefficients.

3.2.3 Chroma

The third feature type maps the frequency spectrum onto the chromatic scale, yielding a *chroma vector*. To generate the chroma vectors I have used Matlab code published by Ellis and Poliner [8].

This feature type was added because I thought it would be better suited for melodic comparison than the Fourier and MFCC. Dividing the spectrum into pitch classes should avoid interference from characteristics of the user's voice in the query-by-humming task.

Note that the W parameter has no effect on this feature type; the code uses its own windowing at its default value. The vector size V has no effect either; all vectors are of length 13 — the position and the twelve pitch classes.

3.3 Parameters

Besides the feature type, I will consider four parameters of the system. I list and discuss them here.

3.3.1 Cluster count

The *cluster count*, C, determines how many clusters to create in the k-clustering stage. As each cluster becomes a point in the feature signature, this parameter will also affect the size of the database and the speed of signature comparisons, which makes a high C parameter somewhat of a tradeoff.

The C parameter seems likely to depend on the length of the query.

The intuitive expectation is that the clustering will roughly correspond to note events. If so, the tempo of the song could affect the optimal parameter here. Possible alternatives to fixed-k-clustering would be clustering with a varying k based on the input data, or to distinguish notes based on a beat tracking algorithm [8].

3.3.2 Vector size

The *vector size*, V, determines how many of the coefficients to store in the centroid. In addition to the temporal position, $V - 1$ coefficients are stored, from the lowest frequency and up.

This parameter is not relevant for the chroma feature type.

3.3.3 Position weight

The *position weight*, P_W, affects the value of the position component of vectors. Centroids $\bar{v}_i$ in a query is assigned a position component of $P_W i$. This lets position play a role in clustering and comparison. When $P_W = 0$, the system is equivalent to a bag-of-features model. When P_W is high, clustering is dominated by the position, and becomes equal to partitioning into equal time slices.

3.3.4 Window size

The *window size*, W, determines how large the window used for the Fourier transform is. This affects both the Fourier and the MFCC feature types, but not the chroma feature. This parameter has two distinct effects. First, it affects the number of vectors before clustering, as a smaller window means a smaller step size, and thus more vectors. Second, it affects the frequencies covered by the vector — the first $V - 1$ coefficients will be a different range if picked from a 128 or a 1024 sized Fourier transform.

3.4 Comparison with other systems

It might be helpful to compare the system directly to the work of Beeck et al. [2] and Logan and Salomon [20]:

3.4.1 Beeck et al.

The main difference compared to my system lies in the feature space. My feature space is audio, in a frequency based representation, with one temporal coordinate. Beeck et al. use a feature space for images, with colour, position, coarseness and contrast information.

3.4.2 Logan and Salomon

The feature representation used is fairly similar to the MFCC feature type in this thesis. An important difference lies in the lack of a temporal coordinate; this makes their clustering a bag-of-features model. They do not generate signatures with a sliding window for exact search, as I do. Instead they make one signature for each song, for the purpose of finding similar songs. Lastly, they use the EMD for comparison while I use the SQFD.

Chapter 4

Experiments

This chapter explain how I have evaluated the performance of the system. This includes what data sets I have used, how I have selected the queries, and what criteria I have used to assess the results.

4.1 Exact search

For testing the exact music search, I used public domain MP3 files of classical music, sourced from the Musopen library[1]. The files were downsampled to 8 kHz and mixed to a single channel. One search query was generated from a random position in each song.

To test the algorithm's robustness against noise, the search is run several times with different levels of white Gaussian noise added to the query. The noise is scaled to achieve the desired signal-to-noise ratio.

The noise levels and query lengths have both been chosen to correspond with those of Wang [30] — queries are tried with signal-to-noise ratios (SNRs) of $-15, -12, -9, -6, -3, 0, +3, +6, +9, +12$ and $+15$ dB with lengths of $5, 10$ and 15 s. This makes for in total $20 \cdot 11 \cdot 3 = 660$ queries.

For each query, the files are ordered from most to least likely, and the rank of the file the query originates from is used to calculate the score.

I use two score functions. One is the average recognition rate. This simply counts the ratio of queries where the correct answer is the top ranked.

The other is the mean reciprocal rank (MRR) of the search results. The

[1] `http://www.musopen.com/`

mean reciprocal rank is defined by the following equation, where n is the total number of queries, and r_i is the ranking of the sound file the ith query was generated from, in a search on that query.

$$\mathrm{MRR} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{r_i} \tag{4.1}$$

The recognition rate has the advantage of being fairly intuitive, while the MRR is defined a bit more opaquely. On the other hand, the MRR has the advantage of being more granular, as it will give partially credit if the correct result appears close to the top.

For tuning the parameters I have selected a fairly small data set, consisting of only 20 songs. This is mostly to keep the running times manageable. Running 660 queries on three different feature types can take hours on a 2.00 GHz Intel Core 2 Duo, and the parameter space to investigate is sizeable. However, this does leave a rather large variance that must be kept in mind when comparing the results.

I had originally intended to verify selected configurations on a much larger test set. The purpose would be to guard against overfitting, and to see how well the results achieved would scale, both of which are serious concerns with such a small data set.

Unfortunately I did not have enough time left at the end to run this larger test. This does negatively influence my ability to draw strong conclusions from my experiments. Even so, many of the effects observed are significant enough to be convincing even at small scales.

4.2 Melody search

To test the melody distance measure, I have used Roger Jang's MIR-QBSH corpus. It consists of 4431 recordings. The recordings are from 195 persons, singing 48 different melodies. All recordings are 8 second long wav files with an 8 kHz sample rate.

The corpus also includes manually transcribed fundamental frequencies for each query, and a MIDI file for each of the 48 melodies. These were not used in the testing.

To test the melodic similarity measures, I want to see if a melody sung by one person can be used to identify recordings of the same melody sung by

other people. For this, 100 of the recordings choosen at random are used as queries to search the rest of the recordings.

The score for a query is the number of the top 10 search results that are the same melody as the query. The total score for a configuration is the mean of the individual query scores. This score should be somewhat comparable to the MIREX 2009/2010 variants QBSH results.

I also calculate the mean average precision (MAP) for the configurations. The definition is as follows, where n is the total number of queries, R is the number of relevant results, and r_j is the ranking of the jth relevant result.

$$\text{MAP} = \frac{1}{n} \sum_{i=1}^{n} \text{AP}_i \tag{4.2}$$

$$\text{AP}_i = \frac{1}{R} \sum_{j=1}^{R} \frac{j}{r_j} \tag{4.3}$$

A result is relevant if the recording is of the same melody as the query.

4.3 Parameters

Testing the effect of each parameter individually will be of little use if the starting parameters are bad. Ideally I would want to start with a set of known good parameters, but the system differs enough from the related work that parameters are unlikely to be directly transferable. To determine an acceptable set of configurations to use as a baseline, I started by best-guessing a set of parameters based on previous work where I could. I then used a hill climbing approach to reach the baseline configurations shown in chapter 5. The starting values were as follows:

With little to go on for the cluster count C, I simply picked the a size of 20, the same as used in [2].

For V, Logan and Salomon's system is roughly comparable. [20] They find the best performance from using 19 MFCC features, leading me to set a starting V of 20.

I chose the starting value 5 for P_W experimentally by running the clustering on a set of features and observing at which point the the position coordinate started dominating the clustering.

For W, Logan and Salomon seem to have choosen a value of 409. I chose to start with a value of 512.

Chapter 5

Results

In this chapter I summarise and discuss the results of the experiments I have performed.

There are two sections, one for the audio fingerprinting task and one for the query-by-humming task.

They are both organised in a similar manner. They start with presenting the maximum obtained scores on the training set, with a discussion of what configuration of parameters showed the best performance. The effect of each parameter is then considered in isolation.

5.1 Audio fingerprinting

The best MRR scores seen during tuning are summarised in Figure 5.1, and the best recognition rate in Figure 5.2. Each graph also shows the expected value and standard error of a purely random ranking.

The MRR and recognition rate can be seen to correspond very closely. In looking at the parameters I will list the MRR, as it is the most granular of the two.

At low levels of noise, most configurations manage a perfect score up to around 0 dB, where the recognition rate drops sharply. This suggests that the scores at noise levels of -3 dB, 0 dB and 3 dB will be the most relevant for discriminating between the configurations.

Figure 5.3 shows the MRRs of all configurations that were tested for these noise levels. One configuration for each feature type was chosen for use in further testing. The scores of the selected configurations are shown in

blue. Configurations that differ from the chosen configurations in only one parameter is shown in yellow, while configurations differing in two or more parameters are shown in red. While there was too much variance to pick any configurations as conclusively 'best', they at least seem to be 'good'.

It is worth noting the presence of several configuration that were outright 'bad'. Picking the parameters hapazardly one might, if one were unlucky, end up with a configuration that does no better than random.

The choosen configurations were:

Feature type	Clusters	Vector size	Position weight	Window size
Fourier	30	20	0.1	256
MFCC	5	20	0.1	256
Chroma	20	-	0.1	-

Table 5.1: Base configurations for audio fingerprint

These are the parameters that are used in the following sections, unless otherwise noted.

5.1.1 Feature type

As can be seen from Figure 5.3, all feature types did well at the 0 and 3 dB signal-to-noise ratio. At -3 dB, the chroma feature did noticeably better than the the Fourier and MFCC features.

5.1.2 Clusters

Table 5.7 shows how the retrieval score varied with the the number of clusters.

	C				
Feature type	1	5	15	20	30
Fourier	0.42	0.74	0.82	0.89	
MFCC	0.96	0.89	0.75	0.63	
Chroma	0.54	0.90	0.98	1.00	1.00

Table 5.2: MRR with varying C at 0 dB

The results here were rather unexpected. While the Fourier and Chroma features seem to benefit from getting grouped into many clusters, the MFCC feature show the completely opposite effect. The MRR score continue to increase significantly all the way up to $C = 1$.

This seems very counterintuitive given the close relation of the Fourier and MFCC. But even though the Fourier features and the MFCC features are similar, they represent quite different ways of viewing the spectrum, and to some degree also different parts of the spectrum. By picking only the lowest frequency coefficients, the Fourier features are dropping a large part of the spectrum, in a way the MFCC features are not. I believe that this could be an issue of global versus local features. The Fourier and chroma features seem to be clustering around note-like events, as expected. Meanwhile, the MFCC could be viewing aspects that change relatively slowly during recording. In that case, clustering will just add noise, and smoothing the feature out by averaging it into a single centroid is preferable.

If this is the case, we would expect the MFCC to fare poorly in the query-by-humming search, as focusing on features of the recording or voice there would be counterproductive.

5.1.3 Vector size

The vector size, V, determines how many components to include for each centroid. The parameter only applies to the Fourier and MFCC feature types — for chroma V is always 13; the position and 12 pitch classes.

	V		
Feature type	15	20	25
Fourier	0.83	0.89	0.98
MFCC	0.92	0.89	0.89

Table 5.3: MRR with varying V at 0 dB

The Fourier feature improves significantly here when adding more coefficients. No significant change is seen for the MFCC.

5.1.4 Position weight

As discussed in subsection 3.3.3, the role of the position weight P_W is to allow the position of features in time to influence the clustering and matching. If

$P_W = 0$, the position has no influence, and the system is effectively a bag-of-features. If P_W is high, the position dominates the clustering and makes it a static partitioning.

	P_W				
Feature type	0.0	0.05	0.1	0.2	5.0
Fourier	0.50	0.86	0.89	0.83	0.27
MFCC	0.78	0.89	0.89	0.90	0.65
Chroma	0.54	0.89	1.00	0.90	0.67

Table 5.4: MRR with varying P_W at 0 dB

The parameter is affecting the scores as you would expect, with middle values significantly outperforming either extreme. It is clear that the addition of the position coordinate offers a considerable advantage over bag-of-features clustering for this task.

5.1.5 Window size

As with the vector size, the window size W applies only to the Fourier and MFCC feature types.

	W			
Feature type	128	256	512	1024
Fourier	0.95	0.89	0.63	0.29
MFCC	0.98	0.89	0.64	0.58

Table 5.5: MRR with varying W at 0 dB

Small vector sizes, at least down to 128, seem to do better than large ones, presumably because more of the spectrum is included in the centroids.

5.1.6 Discussion

By far the most surprising result was in how differently the Fourier feature and the MFCC feature behaved with respect to C. Generally options that tended towards including a larger part of the spectrum did better. For solving a specific task, you could probably perform the selection of discriminating features much more intelligently than just "pick the n first coefficients."

While the results are directly comparable, we can see that Wang [30] gets slightly better performance on a database of 10000 songs.

5.2 Query-by-humming

For the query-by-humming task, my chosen configurations were these:

Feature type	Clusters	Vector size	Position weight	Window size	MAP	Top-10
Fourier	30	20	0.1	512	0.73	3.49
MFCC	30	20	0.1	512	0.43	1.47
Chroma	5	-	0.1	-	0.27	0.77

Table 5.6: Base configurations for query-by-humming

These are the parameters that are used in the following sections, unless otherwise noted.

5.2.1 Feature type

Unexpectedly, the Fourier features is doing quite well here, while the chroma features, which were expected to do well here, do poorly.

5.2.2 Clusters

Table 5.7 shows how the retrieval score varied with the the number of clusters.

	C				
Feature type	1	5	15	20	30
Fourier	0.93	2.74	3.30	3.36	3.49
MFCC	1.72	0.54	0.54	0.64	0.84
Chroma	0.51	0.77	1.39	1.48	1.47

Table 5.7: Top-10 with varying C

5.2.3 Vector size

	V		
Feature type	15	20	25
Fourier	3.54	3.49	3.18
MFCC	0.78	0.84	0.83

Table 5.8: Top-10 with varying V

5.2.4 Position weight

	P_W				
Feature type	0.0	0.05	0.1	0.2	5.0
Fourier	1.05	3.51	33.49	3.24	1.43
MFCC	0.41	0.68	0.84	0.69	0.67
Chroma	0.54	0.66	0.77	0.67	0.66

Table 5.9: Top-10 with varying P_W

5.2.5 Window size

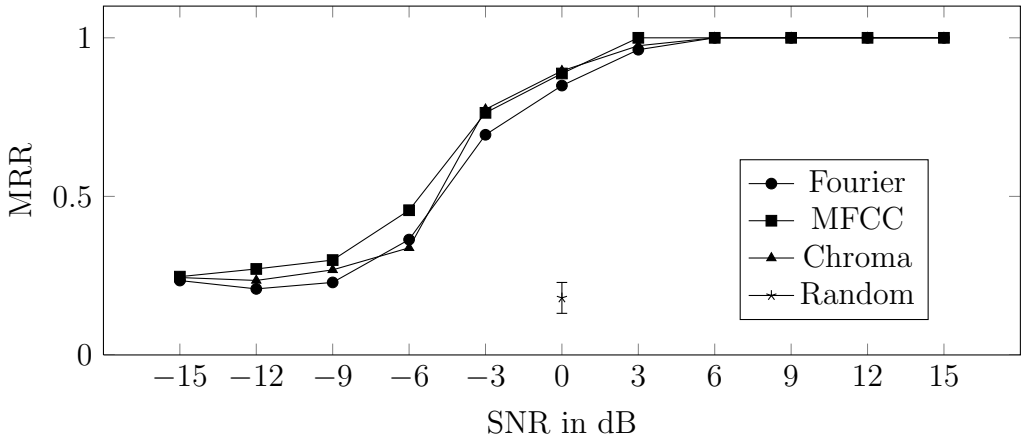
	W	
Feature type	512	1024
Fourier	1.96	3.49
MFCC	0.84	0.57

Table 5.10: MRR with varying W

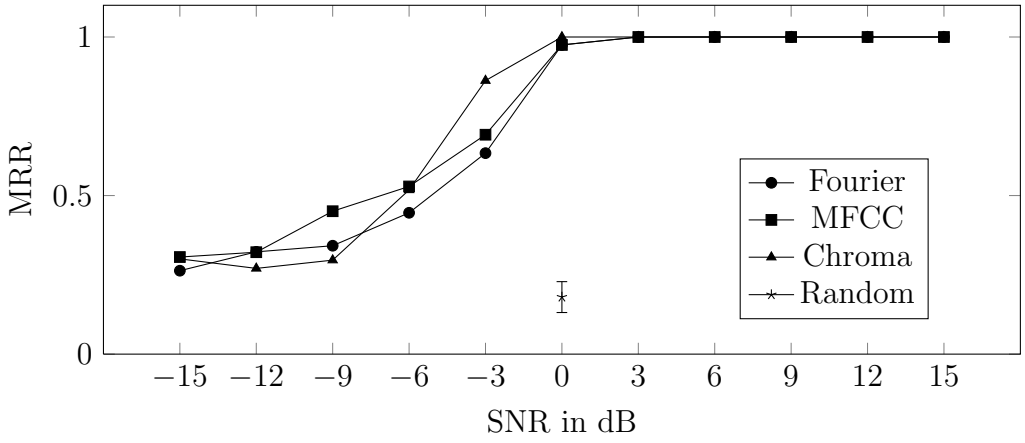
5.2.6 Discussion

Considering the lack of specialisation to the task, the features did very well, especially the Fourier features, unexpectedly enough.

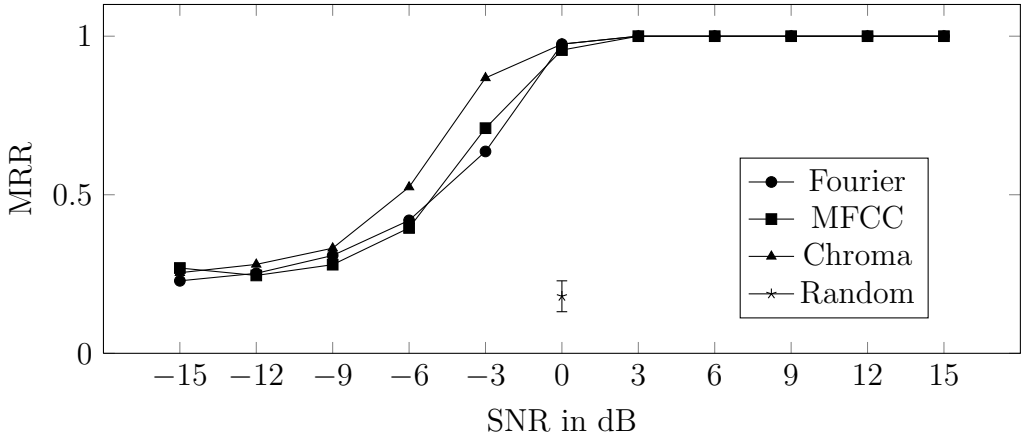
It is worth noting, though, that the top scorer in the comparable MIREX task for 2010 scored 9.135. [16]



(a) 5 second queries

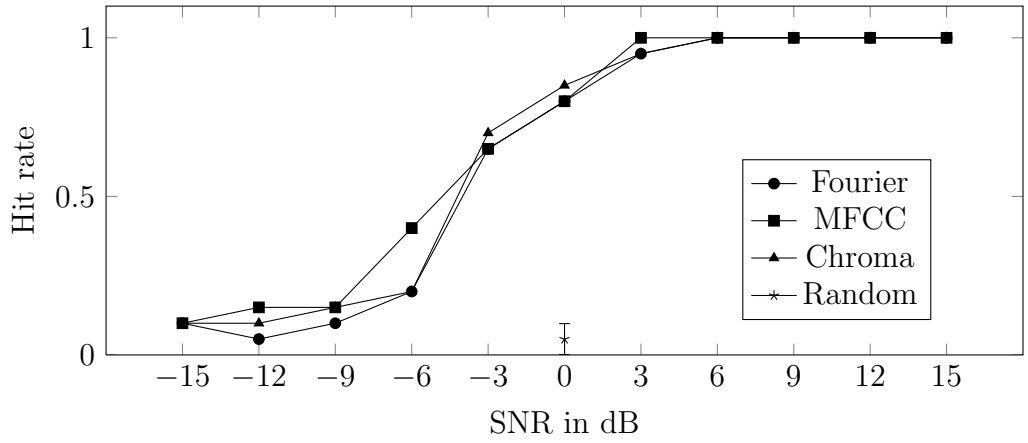


(b) 10 second queries

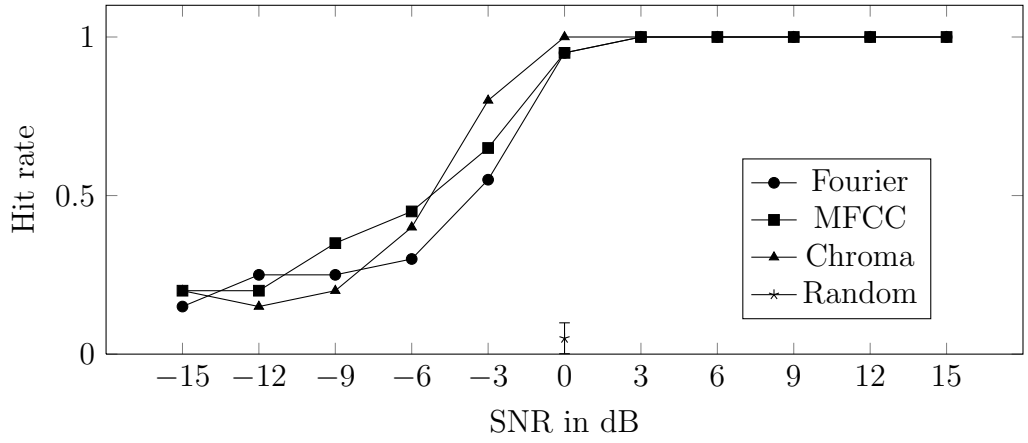


(c) 15 second queries

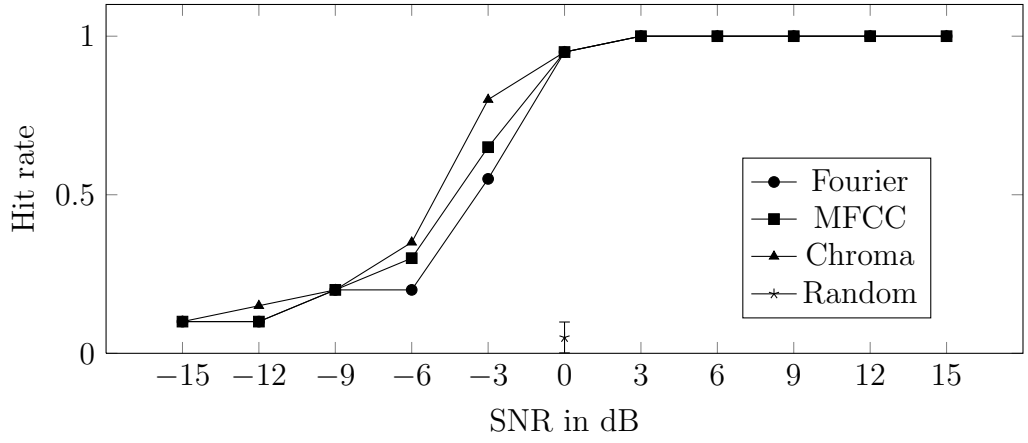
Figure 5.1: MRR by feature type and noise level



(a) 5 second queries



(b) 10 second queries



(c) 15 second queries

Figure 5.2: Recognition rate by feature type and noise level

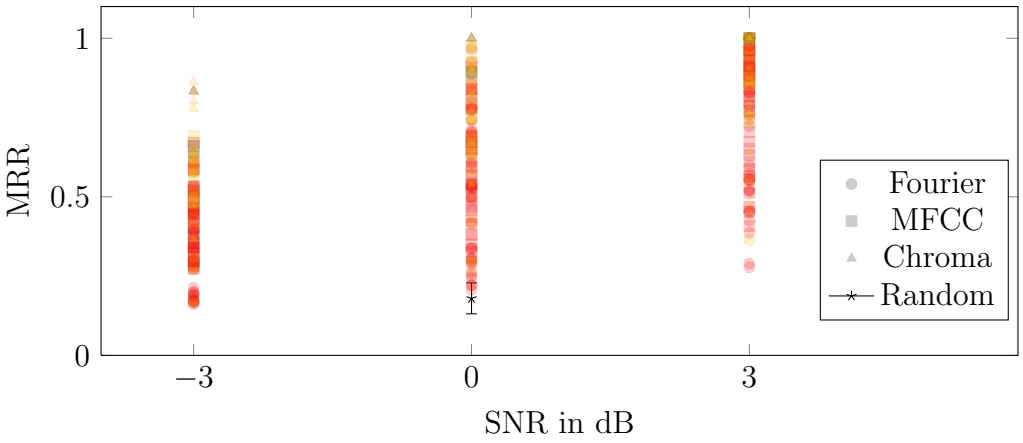


Figure 5.3: Distribution of MRRs for tested configurations, 10 second query

Chapter 6

Conclusion

I have demonstrated a method of searching music by using feature signatures and the signature quadratic form distance, and shown that the method is robust enough to handle noisy queries. I have also applied the method to a query-by-humming task, though the method as-is falls short of the current state of the art.

Through testing with different configurations I have gained some insight into how the parameters affected performance. Of note is the significant change the addition of a position component gave compared to only a bag-of-features model, and in general the importance of accurate clustering. The parameters of the system affect search performance a great deal, and must be chosen with some care to achieve good results. It was also a surprise how geared the MFCC features tended to be towards global feature matching.

Though the signature based method showed satisfactory performance for audio fingerprinting, many well-performing solutions for this problem already exist. For many cases algorithms such as those described in [30], [1] are likely to be a better choice. However, there are cases the SQFD based measure may offer benefits. One example would be a mixed-media database, where the ability to use the same form of signature for indexing both images and sound may make for a simpler system.

The results for the query-by-humming performance also hint that it is generisable to be able to search for more approximate kinds of similarity, and it is relatively easy to customise for different kinds of features in order to search for specific kinds of similarity.

As I was not able to run a large-scale test, I can say little about how well the fingerprinting method will scale to larger databases. With respect to my

research question, though, I believe I have gathered enough data to say that the method is feasible, though some improvements to the speed and accuracy may be required before it is practical.

I had not expected to achieve what one would call a good accuracy on the query-by-humming feature, and this did to some extent prove true. The guiding philosophy in applying these simple features on the problem can be summarised as "try something simple before you try something complex." With this in mind I was positively surprised at how well the system did perform. While I cannot conclusively answer my second research question, I believe the method shows promise also for query-by-humming search, and is worth investigating further.

I see two main directions to take in further work. One is to move on to improve the speed of the search. A natural way to start would involve investigating the indexability of the signatures using metric or Ptolemaic indexing, which has been shown to yield considerable speedups for image data [21]. An effective way to retrieve similar signatures would also allow for testing the audio fingerprinting on much larger databases. Windowing longer queries and looking for sequential matches in time might also give the search greater accuracy

Another direction of inquiry would be looking at different kinds of audio features. The clustering method is quite general, and can be applied to most features that can be extracted from the audio stream. Interesting properties might be invariance to distortion from lossy audio compression. For the signature to be be viable for query-by-humming tasks, particularly desirable properties would be invariance to voice differences, and to transposition of the query. Improvements to the signature could also enable use in other music information retrieval tasks that require approximate melodic matching, e.g. cover song identification.

Appendix A

Acronyms

EMD earth mover's distance

MAP mean average precision

MIR music information retrieval

MFCC mel-frequency cepstral coefficients

MRR mean reciprocal rank

PTD proportional transportation distance

QFD quadratic form distance

SNR signal-to-noise ratio

SQFD signature quadratic form distance

Bibliography

[1] S. Baluja and M. Covell. Content fingerprinting using wavelets. In *Visual Media Production, 2006. CVMP 2006. 3rd European Conference on*, pages 198–207. IET, 2006.

[2] C. Beecks, M.S. Uysal, and T. Seidl. Signature quadratic form distances for content-based similarity. In *Proceedings of the 17th ACM international conference on Multimedia*, pages 697–700. ACM, 2009.

[3] C. Beecks, M.S. Uysal, and T. Seidl. A comparative study of similarity measures for content-based multimedia retrieval. In *Multimedia and Expo (ICME), 2010 IEEE International Conference on*, pages 1552–1557. IEEE, 2010.

[4] C. Beecks, M.S. Uysal, and T. Seidl. Signature quadratic form distance. In *Proceedings of the ACM International Conference on Image and Video Retrieval*, pages 438–445. ACM, 2010.

[5] D. Byrd and T. Crawford. Problems of music information retrieval in the real world. *Information Processing and Management*, 38(2):249–272, 2002.

[6] P. Cano, E. Batle, T. Kalker, and J. Haitsma. A review of algorithms for audio fingerprinting. In *Multimedia Signal Processing, 2002 IEEE Workshop on*, pages 169–173. IEEE, 2002.

[7] M.A. Casey, R. Veltkamp, M. Goto, M. Leman, C. Rhodes, and M. Slaney. Content-based music information retrieval: current directions and future challenges. *Proceedings of the IEEE*, 96(4):668–696, 2008.

[8] D.P.W. Ellis and G.E. Poliner. Identifying cover songs with chroma features and dynamic programming beat tracking. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 4, pages IV–1429. IEEE, 2007.

[9] Shazam Entertainment. About shazam. http://www.shazam.com/music/web/about.html. Online; accessed 2011-05-08.

[10] J.T. Foote. Content-based retrieval of music and audio. In *Proceedings of SPIE*, volume 3229, page 138, 1997.

[11] A. Ghias, J. Logan, D. Chamberlin, and B.C. Smith. Query by humming: musical information retrieval in an audio database. In *Proceedings of the third ACM international conference on Multimedia*, pages 231–236. ACM, 1995.

[12] J. Haitsma and T. Kalker. A highly robust audio fingerprinting system. In *Proc. ISMIR*, volume 2002, pages 144–148. Citeseer, 2002.

[13] A. Huq, M. Cartwright, and B. Pardo. Crowdsourcing a real-world online query by humming system. 2010.

[14] D.P. Huttenlocher, G.A. Klanderman, and WA Rucklidge. Comparing images using the hausdorff distance. *IEEE Transactions on pattern analysis and machine intelligence*, pages 850–863, 1993.

[15] Google Inc. Similar images graduates from google labs. http://googleblog.blogspot.com/2009/10/similar-images-graduates-from-google.html, September 2009. Online; accessed 2011-05-08.

[16] ISMIR. Mirex 2010 qbsh results. http://www.music-ir.org/mirex/wiki/2010:Query-by-Singing/Humming_Results, July 2010. Online; accessed 2011-05-08.

[17] K. Lee. Automatic chord recognition from audio using enhanced pitch class profile. In *Proceedings of the International Computer Music Conference*. Citeseer, 2006.

[18] W.K. Leow and R. Li. The analysis and applications of adaptive-binning color histograms. *Computer Vision and Image Understanding*, 94(1-3):67–91, 2004.

[19] W.N. LIE and C.K. SU. Content-based retrieval of mp3 songs based on query by singing. In *ICASSP*, 2004.

[20] B. Logan and A. Salomon. A music similarity function based on signal analysis. In *ICME 2001*, 2001.

[21] J. Lokoč, M.L. Hetland, T Skopal, and C. Beecks. Ptolemaic indexing of the signature quadratic form distance. In *Proceedings of the Fourth International Conference on SImilarity Search and APplications*, SISAP '11, 2011. To appear.

[22] P. Mermelstein. Distance measures for speech recognition, psychological and instrumental. *Pattern Recognition and Artificial Intelligence*, 116, 1976.

[23] Y. Rubner, C. Tomasi, and L.J. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.

[24] A.W.M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(12):1349–1380, 2000.

[25] J. Song, S.Y. Bae, and K. Yoon. Mid-level music melody representation of polyphonic audio for query-by-humming system. In *International Symposium on Music Information Retrieval*. Citeseer, 2002.

[26] R. Typke, P. Giannopoulos, R.C. Veltkamp, F. Wiering, and R. Van Oostrum. Using transportation distances for measuring melodic similarity. In *Proceedings of the 4th International Conference on Music Information Retrieval (ISMIR 2003)*, pages 107–114, 2003.

[27] R. Typke, F. Wiering, and R.C. Veltkamp. A survey of music information retrieval systems. In *Proceedings of the 6th International Conference on Music Information Retrieval*, pages 153–160. Citeseer, 2005.

[28] R. Typke, F. Wiering, and R.C. Veltkamp. Mirex symbolic melodic similarity and query by singing/humming. *MIREX 2006*, page 19, 2006.

[29] X. Wan and Y. Peng. The earth mover's distance as a semantic measure for document similarity. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 301–302. ACM, 2005.

[30] A. Wang. An industrial strength audio search algorithm. In *International Conference on Music Information Retrieval (ISMIR)*, 2003.