Music Retrieval
based on
Melodic Similarity

Zoeken naar muziek op basis van melodische gelijkenis
(met een samenvatting in het Nederlands)

PROEFSCHRIFT

ter verkrijging van de graad van
doctor aan de Universiteit Utrecht
op gezag van de rector magnificus,
prof. dr. W. H. Gispen, ingevolge het
besluit van het college voor promoties
in het openbaar te verdedigen op
maandag 19 februari 2007 des ochtends te 10.30 uur

door

Rainer Typke

geboren op 11 april 1973 te Waiblingen, Duitsland

Promotor:	Prof. dr. M. H. Overmars
Co-promotoren:	Dr. R. C. Veltkamp
	Dr. F. Wiering

Contents

Chapter 1

Introduction

1.1 Music Information Retrieval

Music Information Retrieval (MIR) is an emerging, interdisciplinary science [21] that aims at retrieving information from music. It draws on fields like musicology, cognitive psychology, linguistics, library science, and last, but not least, computer science.

Michael Kassler mentioned the term "Musical Information Retrieval" (MIR) as early as 1966 [31]. He describes an assembler-like programming language called MIR which can be used to navigate music scores and find positions that fulfill certain criteria. He realized that his language had somewhat limited capabilities, for example that it would be very difficult to write an MIR program that could recognize whether a musical piece is a parody of another one. Interestingly enough, he claimed that the problem of Optical Music Recognition (OMR) was solvable at a cost of about one million dollars, and that an OMR system would be able to transcribe printed music scores at a rate of several thousand musical symbols per minute. Unfortunately, he did not offer any guess on how many of those symbols would be recognized correctly. Fourty years after Kassler's talk, Don Byrd presented a paper [8] at ISMIR 2006 (see page 8 for an explanation of "ISMIR") that does not mention the transcription speed, which really is not a very serious concern, but shows that even the best contemporary OMR programs still produce double-digit numbers of errors per page including, for example, incorrectly recognized pitches for up to 20 % of the notes, and several percent of incorrectly recognized note durations.

Still, in the late 20th century, along with other areas of Multimedia Information Retrieval such as Image Retrieval or Video Retrieval, Music Information Retrieval started to flourish. Around the end of the 20th century, storage space (hard drives, flash memory) became cheap and abundant, and file formats such as MP3 were developed that made it possible to store vast amounts of music in good quality. These advances, along with the fact that processing power has become cheap and abundant as well, have created both the need for automatically retrieving information from music and new possibilities to address this need. Unlike many other art forms such as painting, music stays very enjoyable even if it is stored or transmitted digitally. This leads to large collections of digital music that can be difficult to manage.

Traditional methods of organizing digital music collections by using metadata quickly reach their limits because metadata are frequently unreliable, missing, or extremely expensive to create (a good example is Pandora, see 2.2.11), and also because it is not always easy to invent good categories one could use for metadata. For example, people with different tastes will need very different sets of music genres for characterizing their collection. For some people, it is important to distinguish between categories such as Trance, House, Dance-Pop, Acid, maybe some more categories and Classical Music, while for others a categorization such as Popular Music, Baroque, Classical Music, Romanticism, and music from the 20th century (just to name a few) would be more useful. Unfortunately, users from these two groups even use similar labels ("Classical") for very different concepts. By not relying on manually attached labels, but rather on the music itself for retrieval and clustering tasks, one can avoid not only a lot of work, but also tricky problems like this.

One can gain a good overview of typical MIR tasks by looking at the MIREX competition[1] for MIR algorithms. The tasks from its first two rounds in 2005 and 2006 can be grouped into classification, feature extraction, alignment, and retrieval.

- Classification:
  - **Audio Artist Identification.** Map audio recordings to labels identifying the artist who created the recording.
  - **Audio/Symbolic Genre Classification.** Determine the musical genre for audio or MIDI recordings.
- Feature Extraction:
  - **Audio Melody Extraction.** Extract the main melodic line from polyphonic audio. This involves two subtasks: Voicing detection (deciding whether a particular time frame contains a "melody pitch") and pitch detection (deciding the most likely melody pitch for each time frame).
  - **Audio Onset Detection.** List the onset times of notes in audio recordings.
  - **Audio Drum Detection.** Determine the onset times and corresponding drum class names of drum events in polyphonic music; that is, unlike the previous task, only detect drum onsets instead of those of any note, and also analyze which class the drum belongs to.
  - **Audio Tempo Extraction.** Determine the perceived tempo for audio recordings.
  - **Audio and Symbolic Key Finding.** Determine the key for a given audio recording or MIDI file.
  - **Audio Beat Tracking.** Unlike the drum detection or tempo extraction task, the problem here is to determine the beat locations in an audio recording.
- Alignment: **Score Following.** Real-time alignment of a music signal (audio or MIDI) to a music score.
- Retrieval:

[1] See http://www.music-ir.org/mirexwiki/index.php/Main_Page

– **Audio Music Similarity and Retrieval.** Calculate a similarity matrix for a list of given audio files.

– **Query by Humming[2], Symbolic Melodic Similarity.** Given a query, search a collection for pieces that contain melodically similar musical material.

This task list shows mainly MIR tasks that are neither satisfactorily solved nor extremely far from a solution. For example, the identification of recordings based on excerpts is not a MIREX task because there are already efficient and effective methods known, such as Shazam's method (see Section 2.2.15) or MusicDNS[3]. On the other hand, some tasks are still far from a satisfying solution, for example the automatic conversion of audio recordings to MIDI files or even an intermediate step towards that goal, the separation of several audio sources (various instruments or voices in a polyphonic piece of music) that are found within one recording.

1.2 Topics of this thesis

The main topic of this thesis is a method for the "Symbolic Melodic Similarity" task, that is, measuring melodic similarity for notated music such as MIDI files. Chapter 2 describes several methods for solving this and similar tasks, along with examples of Music Information Retrieval systems that implement these methods.

In Chapter 3, a music search algorithm is developed and studied which views music as sets of notes that are represented as weighted points in the two-dimensional space of time and pitch. Two point sets can be compared by calculating how much effort it would take to convert one into the other; effort is measured by determining how much weight has to be moved over what distances. The point sets are more similar if there are fewer and smaller movements of weight needed. This transportation-based similarity measure has some desirable properties such as continuity and the ability to match any combination of polyphonic and monophonic music.

To make these point set comparisons efficient enough for searching large databases, the distances between every item (point set) in the database and a small, fixed set of special (vantage) point sets can be pre-calculated. Whenever a new query needs to be compared to the items in the database, one can restrict the search to those items with similar distances to the special point sets. The application of this vantage indexing method to music is described in Chapter 4. Vantage indexing was first suggested for image retrieval [90].

For studying the performance of the transportation-based search algorithm and other, similar ones, the creation of a ground truth for a large music collection (RISM) is described in Chapter 5, along with a performance measure and the application of both the ground truth and the measure for the MIREX algorithm competition.

In Chapter 6, a distance measure is described that is inspired by transportation distances, but puts additional constraints on what flows are possible. If the Earth Mover's Distance (defined in Section 3.1.2) is used for comparing a set containing some points with very large weights to another set with very light points at similar

[2] For a definition of "Query by Humming", see Section 1.5.

[3] MusicDNS (http://www.musicdns.org/)is an open source audio fingerprinting system. Unlike Shazam, it needs more than just a few seconds of audio to identify a track. See Section 2.2.7.

positions, it can happen that the heavy points are partially matched with points that lie far away in the time dimension. This usually does not make musical sense. It can be avoided by constructing a graph with nodes that represent notes and edges that connect notes. For such a graph, the solution of a maximum-flow, minimum-cost problem can be used for comparing point sets in a fashion similar to how the Earth Mover's Distance works. To make unwanted flows less likely, the weight distribution can be normalized before solving the maximum-flow, minimum cost problem. Supporting polyphony can be achieved by constructing the network accordingly.

Chapter 7 concludes the thesis with a brief overview of its contribution, open issues, and some thoughts about the future of Music Information Retrieval.

This thesis contains material that was published in peer-reviewed journals and at international conferences.

- **Chapter 2:** Rainer Typke, Frans Wiering, Remco C. Veltkamp: A Survey of Music Information Retrieval Systems. Proceedings of the Sixth International Conference on Music Information Retrieval (ISMIR), London, September 2005 [81]

- **Chapter 3:**

  - Rainer Typke, Panos Giannopoulos, Remco C. Veltkamp, Frans Wiering, Ren van Oostrum: Using transportation distances for measuring melodic similarity. Proceedings of the International Conference on Music Information Retrieval (ISMIR), pages 107–114, Baltimore, October 2003 [75, 76]

  - Frans Wiering, Rainer Typke, Remco C. Veltkamp: Transportation Distances and their Application in Music-Notation Retrieval. In: Music Query: Methods, Strategies, and User Studies (Computing in Musicology 13, 2004), pages 113-128. CCARH and MIT Press. [93]

  - Remco C. Veltkamp, Frans Wiering, Rainer Typke: Content Based Music Retrieval. In: Encyclopedia of Multimedia, Borko Furht (Ed.), ISBN: 0-387-24395-X, Springer 2006. [89]

  - Rainer Typke, Remco C. Veltkamp, Frans Wiering: Searching notated polyphonic music using transportation distances. Proceedings of the ACM Multimedia Conference, pages 128–135, New York, October 2004 [77]

  - Rainer Typke, Frans Wiering, Remco C. Veltkamp: A search method for notated polyphonic music with pitch and tempo fluctuations. Proceedings of the Fifth International Conference on Music Information Retrieval (ISMIR), pp. 281-288, Barcelona, October 2004 [79]

  - Rainer Typke, Frans Wiering, Remco C. Veltkamp: Transportation distances and human perception of melodic similarity. ESCOM Musicæ Scientiæ, 2007 [83]

- **Chapter 4:** Reinier H. van Leuken, Remco C. Veltkamp, Rainer Typke: Selecting vantage objects for similarity indexing. International Conference on Pattern Recognition (ICPR) 2006, Hong Kong [87]

- **Chapter 5:**

- Rainer Typke, Marc den Hoed, Justin de Nooijer, Frans Wiering, Remco C. Veltkamp: A Ground Truth For Half A Million Musical Incipits. Proceedings of the 5th Dutch-Belgian Information Retrieval Workshop (DIR) 2005, Utrecht, the Netherlands, pages 63–70. [73, 72]
- This publication was selected to also appear in the Journal of Digital Information Management:
  Rainer Typke, Marc den Hoed, Justin de Nooijer, Frans Wiering, Remco C. Veltkamp: A Ground Truth For Half A Million Musical Incipits. Journal of Digital Information Management 3(1), 2005, pages 34–39 [74]
- Rainer Typke, Remco C. Veltkamp, Frans Wiering: A measure for evaluating retrieval techniques based on partially ordered ground truth lists. International Conference on Multimedia & Expo (ICME) 2006, Toronto, Canada [78]
- Rainer Typke, Frans Wiering, Remco C. Veltkamp: Evaluating the Earth Movers Distance for measuring symbolic melodic similarity, MIREX abstract, 2005 [80]
- Rainer Typke, Frans Wiering, Remco C. Veltkamp: MIREX Symbolic Melodic Similarity and Query by Singing/Humming, MIREX abstract, 2006 [82]

1.3 The usefulness of melodic similarity measures

As the list of MIREX tasks shows, one can distinguish two main groups of methods for content-based searching of music databases: methods for audio data and methods for notated music. Some Music Information Retrieval systems combine the two by first converting an audio signal into a symbolic description of notes and then searching a database of notated music. Since many researchers work on tasks for automatically creating symbolic descriptions of audio recordings (such as beat detection, onset detection, melody extraction, and even complete transcription to MIDI of the notes in an audio recording), methods that perform well for symbolic data are eventually going to be useful for audio data as well.

Melody search engines (search engines which serve the information need for pieces of music that contain musical material which is melodically similar to a given query) can be useful for a variety of purposes and audiences:

- Query-by-Humming: in record stores, it is not uncommon for customers to only know a tune from a record they would like to buy, but not the title of the work, composer, or performers. Salespeople with a vast knowledge of music who are willing and able to identify tunes hummed by customers are scarce, and it could be interesting to have a computer do the task of identifying melodies and suggesting records. A Query-by-Humming device would also be interesting for libraries, as a phone service similar to Shazam (see Section 2.2.15), or to aid internet users with the task of retrieving entertaining MP3 files for buying them online.
- A search engine that finds musical scores similar to a given query can help musicologists find out how composers influenced one another or how their works are related to earlier works of their own or by other composers. This task has been done manually by musicologists over the past centuries. If

computers could perform this task reasonably well, more interesting insights could be gained faster and with less effort.

- Copyright issues could be resolved, avoided or raised more easily if composers could easily find out if someone is plagiarizing them or if a new work exposes them to the risk of being accused of plagiarism.

For example, the symbolic melodic similarity retrieval algorithm described in Chapter 3 is used as one of the starting points for Frans Wiering's "Witchcraft" project[4], which aims at creating a search tool that helps test hypotheses about oral transmission of folk songs. This tool is also going to be integrated in the Nederlandse Liederenbank to provide access to the folksong collection of "Onder de Groene Linde", taking into account the problem of oral variation.

1.4 Important features for melody, invariances in perception

Melody is one of the most memorable and characteristic features of Western music. The main topic of this thesis is a retrieval method for melodically similar items, so a few words about what makes melodies similar and what does not are in place.

From the cognitive theory of music [71], we know that melodic motion (characterized by successive pitch intervals) and contour are very important for the perception of a melody. For the rhythmical aspect, patterns are perceived in relation to an underlying pulse that defines the tempo.

Melodic motion and contour as well as the rhythmic patterns do not change if the tempo is changed. Rhythm is always defined in relation to the pulse, and if the pulse gets slower or faster, the rhythm stays the same (unless the tempo change is extreme). For pitch intervals and melodic contour, it is obvious that they are not affected by tempo changes. Also, if a melody is transposed to a different key, these things do not change. Therefore, a basic requirement for a melody search engine is that its distance measures are invariant under transposition and augmentation or diminution.[5] Another musical feature that does not influence the perception of melodies is timbre; making a melody search engine ignore timbre is trivial if it uses notation.

Studies such as Selfridge-Field's article [70] show that melodic similarity is continuous. Local melodic changes such as lengthening a note or moving it up or down a step are usually not perceived as changing the identity of a melody, and by applying more and more changes, the perceived relationship to the original becomes only gradually weaker. Also, melodies are generally quite resistant to the insertion of all sorts of ornamentation. Mozart's variations on "Ah, vous dirai-je, maman" can serve as an illustration for this. See Figure 1.1.

[4] http://www.cs.uu.nl/research/projects/witchcraft/

[5] In Section 3.5.2, we will see that the performance of a search engine can still be improved by attaching a cost to tempo changes. The main reason for this is that there are limits to the tempo invariance of human perception. Snyder [71] explains, for example, that there are some fixed thresholds for time intervals within which events are perceived as happening concurrently; whenever one changes the tempo such that some musical events cross one of these thresholds, perception will change. So, a good distance measure for melodies should not necessarily be always invariant under tempo changes, but allow for controlling the influence of tempo changes on the resulting distances.

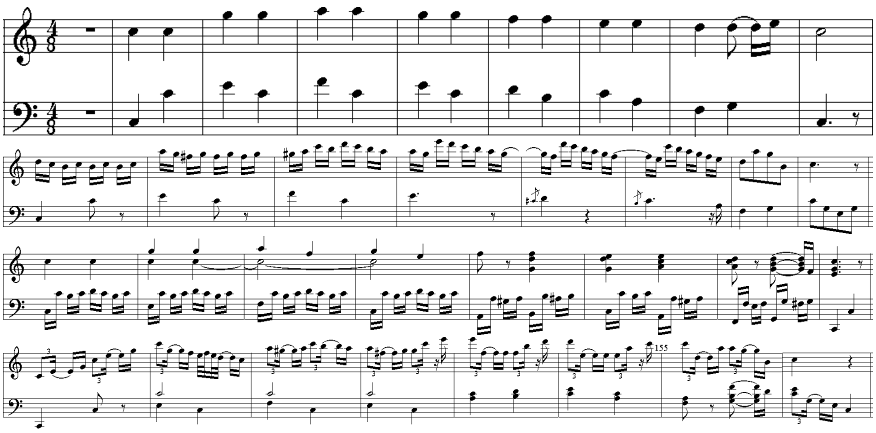


Figure 1.1: A few excerpts from Wolfgang Amadeus Mozart: Twelve variations on "Ah, vous dirai-je, maman", K. 300e.

1.5 Some terms and basic facts

For the rest of this book, we will use the following terms without defining them again:

Pitch The pitch of a sound determines as how "high" it is perceived. Different areas of the cochlea are responsible for detecting different pitches. A musical instrument or singer produces a spectrum of signals with different frequencies. For a pitched instrument, there is a fundamental frequency (often abbreviated as "F0"), accompanied by overtones whose frequencies are multiples of the fundamental frequency. F0 estimation is an important task for pitch detection.

Chroma In a Music Information Retrieval context, chroma is a 12-dimensional vector containing the spectral energy of each of the 12 traditional pitch classes of the equal-tempered scale. This feature takes the close octave relationship in melody and harmony into account as it is prominent in Western music.

Intensity/loudness The amplitude and therefore the energy of the musical signal determines how intense or loud music is perceived.

Timbre Timbre is best defined as what it is not: it is the qualities of sounds that make them distinguishable and that do not fall under either pitch or loudness.

Intervals When speaking about "intervals", we usually mean intervals between pitches. The sequence of intervals between subsequent notes is important for the perception of melody.

Onset time, duration A note usually starts with a very short pitchless attack, followed by the presence of a clearly detectable pitch for a certain amount of time. By "onset time", we mean the point of time when a note starts, and a note's duration is determined by the time period between the onset time and the point of time when the note stops being audible (the offset time).

Melodic contour By "melodic contour", we mean a sequence of interval directions in a melody. If this sequence only distinguishes between "up", "down", or "repeat", we call it "gross contour" to express the fact that the information from the interval sequence is very much reduced.

N-grams We call a group of n subsequent notes an n-gram.

Query by Humming With "Query by Humming", we mean that the user enters a search query by singing, humming, or whistling it into a microphone. That is, the query contains a pitch vector, but not necessarily a measure structure or lyrics. Neither pitches nor tempo are quantized, and onsets or notes need to be detected after the query has been entered if this information is needed by the search algorithm.

1.5.1 Acronyms

The following acronyms will appear frequently when talking about music information retrieval:

ISMIR ISMIR is the main music information retrieval conference. It started as "International Symposium on Music Information Retrieval"; now it is a veritable conference and not just a symposium. For more information, see http://www.ismir.net.

MIREX MIREX is a TREC-like series of comparisons for algorithms. MIREX stands for "Music Information Retrieval Evaluation eXchange". See http://www.music-ir.org/mirex2006/.

RISM Répertoire International des Sources Musicaux. The International Inventory of Musical Scores describes itself as "a cross-country non-profit joint venture which aims at comprehensive documentation of the worldwide existing musical scores". See http://rism.stub.uni-frankfurt.de/. RISM has created a collection of musical incipits and metadata, the "RISM A/II" collection, that covers material from several centuries. We used this collection for comparing the retrieval algorithm that is described in this book to other approaches. Subsets of this collection were used for the "Symbolic Melodic Similarity" MIREX task in 2005 and 2006.

TREC The Text Retrieval Conference (TREC) started yearly competitions for retrieval algorithms in 1992. See http://trec.nist.gov/.

Chapter 2

Related work

In this chapter, we will give a brief overview of different search methods, followed by a list of MIR systems as examples for implementations of these methods. These MIR systems were collected with the help of a website, http://mirsystems.info. We conclude the chapter by mapping the surveyed systems to retrieval tasks, users, and search levels such as genre, work, or instance.

2.1 Search methods for music

While it is possible to search music by working exclusively in the audio domain, many search methods combine some transcription into a symbolic representation, followed by searching symbolic data. We are going to present symbolic and audio techniques in separate sections and describe distance measures and indexing techniques.

2.1.1 Searching symbolic data

String-based methods for monophonic melodies

Monophonic music can be represented by one-dimensional strings of characters, where each character describes one note or one pair of consecutive notes. Strings can represent interval sequences, gross contour, sequences of pitches and the like, and well-known string matching algorithms such as algorithms for calculating editing distances, finding the longest common subsequence, or finding occurrences of one string in another have been applied, sometimes with certain adaptations to make them suitable for matching melodies.

Figure 2.1 illustrates the fact that there are quite common variation techniques that are not naturally supported by string-based methods. Frequently, a variation of a melody is perceived as melodically similar, but contains many more notes, leading to string representations with many more characters. Constructing a similarity measure for strings that still recognizes this kind of similarity without producing false positives is not easy.

Distance Measures Some MIR systems only check for exact matches or cases where the search string is a substring of database entries. For such tasks, standard string searching algorithms like Knuth-Morris-Pratt and Boyer-Moore [34], [16] can be used. Themefinder (see Section 2.2.19) searches the database for entries matching

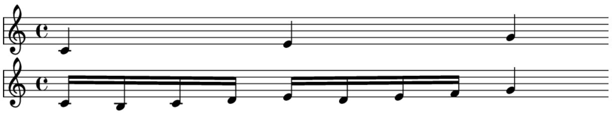


Figure 2.1: Detecting melodic similarity with string-based methods can be difficult in cases like this example (inspired by Geraint Wiggins, ISMIR 2006 [13]).

regular expressions. In this case, there is still no notion of distance, but different strings can match the same regular expression.

For approximate matching, it can be useful to compute an editing distance with dynamic programming. Musipedia is an example of a system that does this (see Section 2.2.8). Simply computing an editing distance between query strings and the data in the database is not good enough, however, because these strings might represent pieces of music with different lenghts. Therefore, it can be necessary to choose suitable substrings before calculating an editing distance.

Indexing For finding substrings that match exactly, the standard methods for indexing text can be used (for example, inverted files, B-trees, etc.). The lack of the equivalent of words in music can be overcome by just cutting melodies into n-grams [18] and indexing those.

For most editing distances that are actually useful, the triangle inequality holds[1]. Therefore, the vantage indexing method described in Chapter 4 can be used for those, but other methods like metric trees or vantage point trees [96] are also possible.

Set-based methods for polyphonic music

Unlike string-based methods, set-based methods do not assume that the notes are ordered. Music is viewed as a set of events with properties like onset time, pitch, and duration. This makes set-based methods suitable to polyphonic music.

Distance Measures Clausen et al. [11] proposed a search method that views scores and queries as sets of notes. Notes are defined by note onset time, pitch, and duration. Exact matches are supersets of queries modulo $\mathbb{Z} \times \mathbb{Z}$-shifts, and approximate matching is done by finding supersets of subsets of the query or by allowing alternative sets.

The method described in this book (see Chapter 3) also views scores and queries as sets of notes, but instead of finding supersets, it uses transportation distances such as the Earth Mover's Distance for comparing sets.

In Section 3.5.1 (page 42), we compare our own method to PROMS by Clausen et al. and to several C-BRAHMS algorithms.

Indexing By quantizing onset times and by segmenting the music into measures, [11] make it possible to use inverted files. [75] exploit the triangle inequality for indexing, which avoids the need for quantizing. Distances to a fixed set of vantage

[1] An example for a not very useful editing distance would be one where any character can be replaced with one special character at no cost. That way, the detour via a string consisting only of that special character would always yield the distance zero for unequal strings of the same length.

objects are pre-calculated for each database entry. Queries then only need to be compared to entries with similar distances to the vantage objects.

Probabilistic Matching

The aim of probabilistic matching methods is to determine probabilistic properties of candidate pieces and compare them with corresponding properties of queries. For example, the GUIDO system (see Section 2.2.5) calculates Markov models describing the probabilities of state transitions in pieces and then compares matrices which describe transition probabilities.

Distance Measures Features of melodies such as interval sequences, pitch sequences, or rhythm can be used to calculate Markov chains. In these Markov chains, states can correspond with features like a certain pitch, interval, or note duration, and the transition probabilities reflect the numbers of occurrences of different subsequent states. The similarity between a query and a candidate piece in the database can be determined by calculating the product of the transition probabilities, based on the transition matrix of the candidate piece, for each pair of consecutive states in the query. See Section 2.2.5 for an example of a MIR system with probabilistic matching.

Indexing: Hierarchical Clustering Transition matrices can be organized as a tree. The leaves are the transition matrices of the pieces in the database, while inner nodes are the transition matrices describing the concatenation of the pieces in the subtree. See Section 2.2.5 or [26] for a more detailed description.

2.1.2 Searching audio data

Extracting perceptionally relevant features

A natural way of comparing audio recordings in a meaningful way is to extract an abstract description of the audio signal which reflects the perceptionally relevant aspects of the recording, followed by the application of a distance function to the extracted information. An audio recording is usually segmented into short, possibly overlapping frames which last short enough such that there are not multiple distinguishable events covered by one frame. Wold et. al. [94] list some features that are commonly extracted from audio frames with a duration between 25 and 40 milliseconds:

- **Loudness:** can be approximated by the square root of the energy of the signal computed from the short-time Fourier transform, in decibels.
- **Pitch:** the Fourier transformation of a frame delivers a spectrum, from which a fundamental frequency can be computed with an approximate greatest common divisor algorithm.
- **Chroma:** From pitch, one can calculate chroma. Müller et al. [47, 48], for example, use chroma for aligning different audio recordings of the same work or finding occurrences of audio queries in a recording.

- **Tone (brightness and bandwidth):** Brightness is a measure of the higher-frequency content of the signal. Bandwidth can be computed as the magnitude-weighted average of the differences between the spectral components and the centroid of the short-time Fourier transform. It is zero for a single sine wave, while ideal white noise has an infinite bandwidth.

- **Mel-filtered Cepstral Coefficients** (often abbreviated as MFCCs) can be computed by applying a mel-spaced set of triangular filters to the short-time Fourier transform, followed by a discrete cosine transform. The word "cepstrum" is a play on the word "spectrum" and is meant to convey that it is a transformation of the spectrum into something that better describes the sound characteristics as they are perceived by a human listener. A mel is a unit of measure for the perceived pitch of a tone. The human ear perceives changes in frequency below 1000 Hz differently from changes above 1000 Hz. Mel-filtering is a scaling of frequency that takes this fact into account.

- **Derivatives:** Since the dynamic behaviour of sound is important, it can be helpful to calculate the instantaneous derivative (time differences) for all of the features above.

Audio retrieval systems such as SuperMBox (see Section 2.2.18) compare vectors of such features in order to find audio recordings that sound similar to a given query.

Audio Fingerprinting

If the aim is not necessarily to identify a work, but a recording, audio fingerprinting techniques perform quite well. See [9] for a review of audio fingerprinting algorithms. Phone-based systems for identifying popular music (e. g., Shazam) use some form of audio fingerprinting. A feature extractor is used to describe short segments of recordings in a way that is as robust as possible against the typical distortions caused by poor speakers, cheap microphones, and a cellular phone connection, as well as background noise like people chatting in a bar. Such features do not need to have anything to do with human perception or the music on the recording, they just need to be unique for different recordings and robust against distortions. These audio fingerprints, usually just a few bytes per recording segment, are then stored in a database index, along with pointers to the recordings where they occur. The same feature extractor is used on the query, and with the audio fingerprints that were extracted from the query, candidates for matching recordings can be quickly retrieved. The number of these candidates can be reduced by checking whether the fingerprints occur in the right order and with the same local timing. Besides the obvious purpose of identifying recordings, audio fingerprints can also be used for estimating the quality of recordings [17].

Set-based Methods

Clausen and Kurth used their set-based method (see Section 2.1.1; a description of the rather similar C-BRAHMS algorithm P1 can be found on Page 42) also for audio data. They use a feature extractor for converting PCM[2] signals into sets that can be treated the same way as sets of notes.

[2] PCM (Pulse Code Manipulation): raw uncompressed digital audio encoding.

Self-Organizing Map

Self-Organizing Map (SOM), a very popular artificial neural network algorithm in the unsupervised learning category, has been used for clustering similar pieces of music and classifying pieces, for example by [63]. Section 2.2.16 describes their system, which extracts feature vectors that describe rhythm patterns from audio, and clusters them with a SOM. A SOM consists of units which are ordered on a low-dimensional grid (usually 2-dimensional, so that it can be nicely put onto a page). A model vector in the high-dimensional data space is assigned to each of the units. During the training, the model vectors are fitted to the data such that the distances between the data items and the corresponding closest model vectors are minimized. The model vectors can contain any features.

2.2 MIR Systems

Table 2.1 gives an overview of the characteristics of some MIR systems. The following subsections contain additional information about these systems.

2.2.1 audentify!

URL: `http://www-mmdb.iai.uni-bonn.de/eng-public.html`
The fingerprints are sequences of bits with a fixed length, where every bit describes one audio window. The collection contains about 15.000 MP3 files (@128kBit/s), approx. 1.5 month of audio data.
Literature: [39], [38], [64], [37], [12]

2.2.2 C-Brahms

URL: `http://www.cs.helsinki.fi/group/cbrahms/demoengine/`
C-Brahms employs nine different algorithms called P1, P2, P3, MonoPoly, IntervalMatching, ShiftOrAnd, PolyCheck, Splitting, and LCTS offering various combinations of monophony, polyphony, rhythm invariance, transposition invariance, partial or exact matching. In Section 3.5.1, we report the results of a comparison of our method with some C-Brahms algorithms. P3 was also submitted to MIREX 2006, see Section 5.3.2, page 92.
Literature: [86], [42], [41]

2.2.3 CubyHum

Edit distances of one-dimensional pattern sequences (here: pitch intervals) are calculated. Nine interval classes are used; intervals above 6 semitones are not distinguished. Filtering is done with the LET algorithm [10] with some heuristic adjustments. CubyHum still looks at every single database entry in every search.
Literature: [57]

Table 2.1: Content-based Music Information Retrieval systems: Input types, matching properties, features, indexing methods, and collection sizes.

	audentify!	C-Brahms	CubyHum	Cuidado	GUIDO/MIR	Meldex/Greenstone	Musipedia	notify! Whistle	Probabilistic "Name that Tune"	PROMS	Cornell's "QBH"	Shazam	SOMeJB	SoundCompass	SuperMBox	Themefinder
Input																
Audio	●		●	●		●	●	●			●	●	●	●	●	
Symbolic		●			●	●	●	●	●	●						●
Matching																
Audio	●			●								●	●			
Symbolic		●	●		●	●	●	●	●	●	●			●	●	●
Exact		●								●		●				●
Approximate	●	●	●	●	●	●	●	●	●	●	●		●	●	●	
Polyphonic	●	●		●			●	●		●		●	●			
Features																
Audio fingerprints	●											●				
Pitch		●			●		●	●		●				●	●	●
Note Duration		●			●		●									
Timbre				●												
Rhythm		●		●	●		●	●		●				●	●	
Contour						●	●			●						●
Intervals		●	●		●	●	●		●							●
Other				●	●				●				●			
Indexing																
	Inverted Files	none	LET	not described	tree of transition matrices	none	Vantage Objects	Inverted Files	Clustering	Inverted Files	none	Yes	Tree	Yes	Hierarchical Filtering	none
Collection Size																
	15,000	278	510	>100,000	150	9,354	¿130,000	2,000	100	12,000	183	> 2.5 million	359	11,132	12,000	35,000

2.2.4 Cuidado Music Browser

Besides similarity measures based on intrinsic audio features such as rhythm, energy, and timbre, there are also similarity measures based on metadata. A co-occurrence matrix keeps track of similar contexts like a radio program, album playlist, or web page. The authors do not describe an indexing method.
Literature: [53], [55], [54]

2.2.5 GUIDO/MIR

URL: `http://www.informatik.tu-darmstadt.de/AFS/GUIDO/index.html`

Queries are a combination of melodic (absolute pitch, intervals, interval types, interval classes, melodic trend (upwards or downwards)) and rhythmic information (absolute durations, relative durations, trend). First-order Markov chains are used for modeling the melodic and rhythmic contours of monophonic pieces of music. There is one Markov chain for each piece and each melodic or rhythmic query type. The states of these chains correspond with melodic or rhythmic features.

Transition matrices are organized as a tree (leaves: pieces; inner nodes: transition matrices describing the concatenation of the pieces in the subtree) with the aim of ruling out data with transition probabilities of zero at an early stage of the search, and heuristically guiding the search.
Literature: [26]

2.2.6 Meldex/Greenstone

URL: `http://www.nzdl.org/fast-cgi-bin/music/musiclibrary`
Meldex uses two matching methods: Editing distance calculation with dynamic programming and a state matching algorithm for approximate searching [95]. With the state matching algorithm, the authors get a much lower complexity than with dynamic programming ($O(kn+m+a)$ instead of $O(mn)$, where n is the length of the pattern, m is the length of the string, and a is the alphabet size) by maintaining k copies of the dynamic programming array, where the ith matrix holds values for matches with up to i errors. The folk song collection is composed of the Essen and Digital Tradition collections.
Literature: [44], [4]

2.2.7 MusicDNS

URL: `http://www.musicdns.org/`
MusicDNS is open-source audio fingerprinting software that aims at identifying tracks of recorded music. Unlike Shazam, it operates on a relatively long sample of audio (2 minutes or the whole track, whichever is shorter) and is therefore not designed to be able to identify tracks based on just a few seconds of audio, and it also cannot pick individual songs out of mash-ups. A fingerprint is calculated in several steps: first, the audio sample is converted to a series of spectra using FFT (Fast Fourier Transform). Frequency bins holding information about the amplitude are viewed as a matrix (with the rows corresponding to time and columns to frequency). This matrix is turned into a much smaller matrix (512 bytes) with a “Singular Value Decomposition”. Finally, “peak trajectories” are determined from the matrix by

looking for prominent frequencies that have continuities from frame to frame. The four strongest trajectories are included in the fingerprint.

Literature: [49]

2.2.8 Musipedia

URL: `http://www.musipedia.org`
The search engine retrieves the closest 100 entries according to either the editing distance of gross contour strings or the Earth Mover's Distance for a given melody or rhythm. The collection can be edited and expanded by any user. For indexing, the vantage object method described by [77] is used for the first 6 characters of the contour string or segments of the point sets that represent melodies or rhythms. Musipedia was known as "Tuneserver" in an earlier development state.
Literature: [58], [75], [77]

2.2.9 Muugle

URL: `http://give-lab.cs.uu.nl/muugle`
The "Musical Utrecht University Global Lookup Engine" is a modular framework that is intended for the purpose of comparing different MIR techniques using the same data collection. Among the implemented search techniques are some of the C-Brahms algorithms as well as the Earth Mover's Distance and the Proportional Transportation Distance.
Literature: [6]

2.2.10 notify! Whistle

URL: `http://www-mmdb.iai.uni-bonn.de/projects/nwo/index.html`
Monophonic queries are matched against polyphonic sets of notes. A rhythm tracker enables matching even if there are fluctuations or differences in tempo. The audio queries can be symbolically edited in pianoroll notation.
Literature: [38]

2.2.11 Pandora

URL: `http://www.pandora.com`
An internet radio website that allows users to say whether they like or dislike a given song, and in response plays only songs that the user should like. For all songs in Pandora's database, a long list of features (chosen by the "Music Genome Project") have been manually extracted. From the combination of these features and the user's responses, Pandora calculates which other songs the user should like. There are hundreds of features, many of them on a rather abstract level and therefore hard to calculate by a computer. As an example, here are some of the features whose names start with "A": Abstract Lyrics, Acousti-Synthetic Sonority, Acoustic Bass Solo, Acoustic Drum Samples, Acoustic Guitar Layering, Acoustic Piano Accompaniment, Acoustic Rhythm Guitars, Acoustic Rhythm Piano, Acoustic Rock Instrumentation, Acoustic Sonority, Afro-Cuban Influences, Aggressive Drumming, Aggressive Female Vocalist, Aggressive Male Vocalist, Altered Female Vocal,

Altered Male Vocal, Altered Piano Timbres, Altered Vocal Sound, Ambient Soundscapes, Ambiguous Lyrics, Angry Lyrics, Angular Melodies, Atmospheric Production, Avant-garde Leanings.
More information:
`http://en.wikipedia.org/wiki/Pandora_(music_service)`

2.2.12 Probabilistic "Name That Song"

This system uses not only music, but also lyrics for matching. All note transitions and words from the query must occur at least once in a piece for it to be considered a match. The pieces in the database are clustered. The probability of sampling is computed for each cluster. A query is then performed in several iterations i. In each iteration, a cluster is selected and the matching criteria are applied to each piece in this cluster until a match is found, which then becomes the rank-ith result.

The clustering makes it unnecessary for the algorithm to visit every single piece in the database.
Literature: [7]

2.2.13 PROMS

URL: `http://www-mmdb.iai.uni-bonn.de/forschungprojekte/midilib/`
PROMS views database entries and queries as sets of notes. Matches are supersets of queries. Queries can be fuzzy (a set of finite, nonempty sets of possible notes instead of a set of notes).

PROMS relies on measure information for segmenting and quantizes pitches and onset times. This makes it possible to use inverted files.
Literature: [11]

2.2.14 Cornell's "Query by Humming"

URL: `http://www.cs.cornell.edu/Info/Faculty/bsmith/query-by-humming.html`
After pitch tracking with autocorrelation, maximum likelihood, or cepstrum analysis, the gross contour is encoded with the alphabet U/D/S (up/down/same). The Baeza-Yates/Perleberg pattern matching algorithm is then used for finding all instances of a pattern string in a text string so that there are at most k mismatches.
Literature: [22]

2.2.15 Shazam

URLs: `http://www.shazam.com`, `http://ismir2003.ismir.net/presentations/Wang.PDF`
Audio fingerprints describe the relative time and pitch distances of future peaks within a fixed-size target zone for a given peak in the spectrum ("landmark"). For all database entries with fingerprints that match some fingerprints in the query, it is checked whether they occur at the correct relative times and at the correct landmarks. This method is very robust against noise and distortion caused by using a mobile phone connection and added background noise.
Literature: [92]

2.2.16 SOMeJB - The SOM-enhanced JukeBox

URL: `http://www.ifs.tuwien.ac.at/~andi/somejb/`
A Self-Organizing Map (SOM) is used for clustering pieces. The SOM consists of units which are ordered on a rectangular 2-dimensional grid. Feature vectors contain amplitude values for selected frequency bands.

Training the neural network, the Growing Hierarchical Self-Organizing Map (GH-SOM), an extension of the SOM, results in a hierarchical organization.
Literature: [63], [56], [62], [61], [60]

2.2.17 SoundCompass

Users first set a metronome to a convenient tempo and then hum their melody so that the beats coincide with metronome clicks. Three feature vectors (Tone Transition, Partial Tone Transition, Tone Distribution) are stored for overlapping windows covering the songs (16 beats long, 4 beats apart from each other). SoundCompass performs Euclidean distance calculations, accelerated with an index.
Literature: [36]

2.2.18 Super MBox

URL: `http://neural.cs.nthu.edu.tw/jang/demo/`
The acoustic input is converted into a pitch sequence with a time scale of 1/16 second. Dynamic time warping is used to compute the warping distance between the input pitch vector and that of every song in the database.
Literature: [29]

2.2.19 Themefinder

URL: `http://themefinder.org`
Themefinder provides a web-based interface to the Humdrum thema command, which allows searching of databases containing musical themes or incipits with string matching algorithms.
Literature: [35]

2.3 Retrieval Tasks

MIR systems can be aimed at solving different MIR retrieval tasks. It is worthwhile to map the systems to these tasks.

Three main audiences can be distinguished that can benefit from MIR:

1. industry: e. g. recording, broadcasting, performance
2. consumers
3. professionals: performers, teachers, musicologists

The level at which retrieval is needed may differ considerably:

1. work instance: the individual score or sound object
2. work: set of instances that are considered to be essentially the same

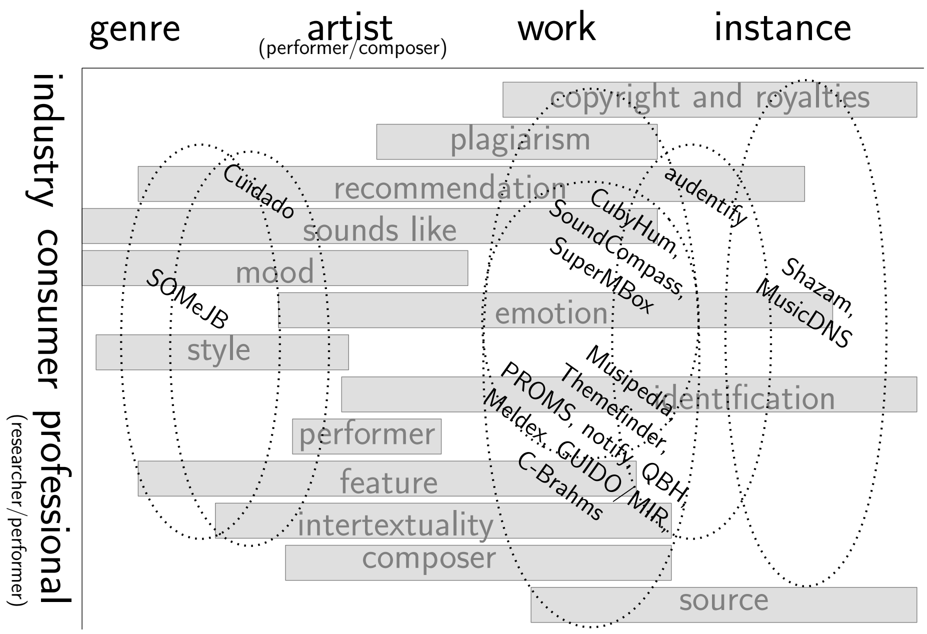


Figure 2.2: A mapping of MIR systems to retrieval tasks. See Section 2.3 for a discussion.

3. artist: creator or performer of work

4. genre: music that is similar at a very generic level, e. g. classical, jazz, pop, world music

This is not a strict hierarchy. Artists perform in different genres, and one work can be performed, even created, by multiple artists. Also, there is rather a continuum. Genres can be divided into subgenres, artists grouped in schools. Even the "work" concept is not a fixed given. Beethoven's Third Symphony, for example is determined by the composer's score, and changing even one note can be a violation of the work, for example the famous "false entry" of the French Horn at the beginning of the recapitulation. On the other hand, different renditions of "I did it my way" are usually considered the same work even though the musical content may be rather different.

MIR retrieval tasks can be characterised by audience and level of retrieval. Often, tasks connect a subrange of the continuum (see Figure 2.2). A non-comprehensive overview of tasks (for typical search tasks and their frequencies of occurence, see also [40]) includes:

- Copyright and royalties: receive payments for broadcast or publication of music (suitable methods: audio fingerprinting for identifying recordings, distance measures for melodic similarity for finding other works with the same musical material).

- Detection of plagiarism: the use of musical ideas or stylistic traits of another artist under one's own name (suitable methods: same as for copyright and royalties).

- Recommendation: find music that suits a personal profile (suitable methods: collaborative filtering, comparison of feature vectors as it is done by Pandora (see Section 2.2.11), classification/clustering methods such as SOM).

- Sounds like: find music that sounds like a given recording (suitable methods: same as for recommendation).

- Mood: find music that suits a certain atmosphere (suitable methods: same as for recommendation).

- Emotion: find music that reflects or contradicts an emotional state (suitable methods: same as for recommendation).

- Style: find music that belongs to a generic category, however defined (suitable methods: same as for recommendation).

- Performer: find music by (type of) performer (suitable methods: same as for recommendation).

- Feature: employ technical features to retrieve works in a genre or by an artist (suitable methods: same as for recommendation).

- Composer: find works by one composer (suitable methods: same as for recommendation).

- Intertextuality: finding works that employ the same material or refer to each other by allusion (suitable methods: same as for copyright and royalties).

- Identification: ascribing a work or work instance to an artist or finding works containing a given theme, query by humming (suitable methods: same as for copyright and royalties).

- Source: identifying the work to which an instance belongs, for example because metadata are missing (suitable methods: same as for copyright and royalties).

Although many basic methods can be used for different purposes (for many tasks, we said that the same methods are suitable as for recommendation), the suitable features differ for the various tasks. For example, recommendation works well with a mixture of abstract features and lower-level features as it is used by Pandora. The "sounds like" task, however, can be performed with more low-level audio features that could possibly be extracted automatically instead of relying on humans. The "composer" task, on the other hand, can be solved successfully by looking at features such as note patterns in scores [3].

Figure 2.2 shows how the MIR systems from Table 2.1 can be mapped to the tasks. Audio fingerprinting systems such as Shazam are particularly good at identifying recordings, that is, instances of works. This task must be based on audio information because in two different performances, the same music might be performed, and therefore only the audio information is different.

Audio data is also a good basis for very general identification tasks such as genre and artist. SOMeJB and Cuidado both use audio features for this purpose. Since it uses metadata, Cuidado can also cover tasks for which it helps to know the artist.

Query-by-humming systems such as SoundCompass, which is intended to be used in a Karaoke bar, make identification tasks easier for consumers who might lack the expertise that is needed for entering a sequence of intervals or a contour in textual form. These systems focus on identifying works or finding works that are similar to a query.

By offering the possibility of entering more complex queries, systems such as Themefinder, C-Brahms, and Musipedia cover a wider range of tasks, but they still can only be used on the work level or for retrieving notational instances of a work. Since they work with sets of notes or representations that are based on sets of notes, they cannot be used for more specific tasks such as identifying specific recordings, and their algorithms are not meant to do tasks on the more general artist and genre levels.

2.4 Observations

We probably covered only a small part of all existing MIR systems (we left some commercial systems out, for example MuscleFish's SoundFisher, because we could not find research papers about them), but we can still make some observations based on this survey.

A great variety of different methods for content-based searching in music scores and audio data has been proposed and implemented in research prototypes and commercial systems. Besides the limited and well-defined task of identifying recordings, for which audio fingerprinting techniques work well, it is hard to tell which methods should be further pursued. This underlines the importance of a TREC-like series of comparisons for algorithms (such as MIREX at ISMIR; for an explanation of these acronyms see Page 8) for searching audio recordings and symbolic music notation.

Audio and symbolic methods are useful for different tasks. For instance, identification of instances of recordings must be based on audio data, while works are best identified based on a symbolic representation. For determining the genre of a given piece of music, approaches based on audio look promising, but symbolic methods might work as well.

Figure 2.2 shows that most MIR systems focus on the work level. There is a gap between MIR systems working on the genre level and those on the work level. Large parts of the more interesting tasks, such as specific recommendation, generic technical features, and intertextuality, fall into this gap. Using metadata might help cover this gap, but this would rule out the possibility of handling data for which the quality of known metadata is not sufficient. Manual annotation quickly gets prohibitively expensive. To fill the gap with completely automatic systems, it might be necessary to find algorithms for representing music at a higher, more conceptual abstraction level than the level of notes.

In the next chapter, we will describe in detail how the problem of retrieving scores with a given melody – or musical material that is similar to it – can be addressed by using transportation distances. This method can be implemented efficiently, and it scored well at MIREX 2006.

Chapter 3

Using transportation distances for measuring symbolic similarity

In this chapter, we describe our method of using the Earth Mover's Distance (EMD) [67] for measuring melodic similarity. The EMD is one particular instance of a transportation distance. We will also describe another one, the Proportional Transportation Distance.

We want to measure melodic similarity in order to solve the problem of retrieving documents that contain musical material which is melodically similar to a given query. We want to search large databases of symbolically encoded music (for example, MIDI files or music scores). The solution to this problem should meet these requirements:

- Humans should agree with the method. In other words, the retrieved items should contain musical material that human listeners or readers would perceive as similar to the query.

- It should be possible to implement the method efficiently. In particular, the response time for answering queries should grow less than linearly with the size of the database to be searched.

- The measure should be robust against pitch and tempo fluctuations. If one would want to use it for "Query by Humming", one would be confronted with queries that are not quantized in either pitch or tempo, and where both can fluctuate.

- The method should not require the data to be searched to be quantized or rhythmically exact. If this requirement is not met, searching real-world MIDI collections would be difficult since many existing MIDI files are recordings of performances, for example on MIDI-enabled pianos.

- It should be possible to search polyphonic music. It is less important, though also desirable, to allow for the queries to be polyphonic as well.

We first present some general ideas on how one can do this. Section 3.1 describes how to represent melodies as weighted point sets, how weighted point sets can be compared using transportation distances, and how one can normalize coordinates and weights to get meaningful results; Section 3.2 lists some advantages of transportation distances. This is followed by some sections about particular situations that require different matching methods, such as matching musical incipits against other incipits or matching a monophonic query against a database with polyphonic pieces. In particular, we study the following cases:

Problem	Method	Evaluation	Section
Matching whole incipits against other whole incipits	No segmenting, only transportation distance	Comparison with earlier efforts of grouping similar incipits together (Schlichte, Howard)	3.3
Matching incipits against other incipits, with improved partial matching in the time dimension	Query segments are aligned with incipits using genetic algorithm	MIREX 2005	3.4
Finding queries in whole pieces	One fixed segment size for both query and database items	Comparison with C-BRAHMS and PROMS	3.5.1
Finding queries in whole pieces	Multiple segment sizes	MIREX 2006	3.5.2

3.1 Representing music as weighted point sets

Representing music as a weighted point set in a two-dimensional space has a tradition of many centuries. Since approximately the 10th century, one popular way of writing music has been to use a set of notes (points) in a two-dimensional space, with time and pitch as coordinates. Varying characteristics are associated with the notes by, for example, using different symbols for different note durations. The look of written music has changed over the last 8 centuries, but the basic idea of representing music as a weighted point set has been followed for almost a millenium, and it has served composers and performers well. Since weighted point sets seem to be so well suited to representing music, it feels natural to measure melodic similarity directly by comparing weighted point sets instead of first transforming the music into one-dimensional abstract representations.

3.1.1 Melodies as weighted point sets

In order to be able to apply a transportation distance measure, we must transform the melodies we want to compare into signatures. By signature, we mean a set of points in the two-dimensional Euclidean space where each point has a weight associated with it. The two dimensions are time and pitch.

When transforming melodies into signatures, we create one point for each note. Rests are encoded implicitly as the time spans that are not covered by points. As a consequence, we do not distinguish between two subsequent quarter rests and one half rest, but we do distinguish between two subsequent quarter notes and a half note; only the latter sounds differently.

The time coordinate

In our database, durations of notes and their positions within measures are specified using divisions of a quarter note, in a way similar to the MIDI format. With every melody, the number of divisions per quarter note is stored. This number is chosen such that the duration of every note in the melody can be specified as a whole number. For example, if there are 96 divisions per quarter note, a quarter note has duration 96, a half note has duration 192, and a sixteenth 24.

We want time coordinates in signatures to be independent of the number of divisions chosen for a particular melody. Therefore, we calculate the time coordinate of a note as the sum of the lengths of measures preceding the note plus the note's position within its measure, divided by the number of divisions for a quarter note. Measure lengths are calculated as follows: for each note or rest in a measure, the duration is added to the position within the measure. The maximum of all of these end points of notes and rests is then taken as the measure length.

In order to skip leading rests – we do not want to distinguish between melodies that differ only in the duration of leading rests –, we then subtract the very first note's time coordinate from all time coordinates, thereby shifting all notes so that the first note starts at time 0.

For a complete example, see Figure 3.1 and Table 3.1.

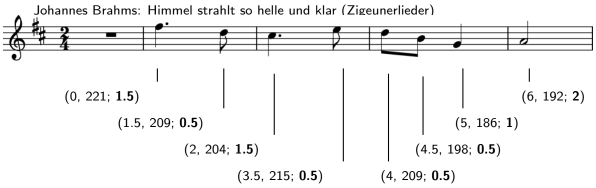


Figure 3.1: An example of music represented with a weighted point set. Format: (Time, Pitch; **Weight**). In this example, the weights only reflect the note durations. Because of this, the time coordinate here equals the sum of the weights of preceding notes. Pitches are specified using Hewlett's [24] base-40 system.

Our method of determining the length of each measure without relying on the time signature ensures that we get sensible coordinates even in cases where the notes in a measure do not match the time signature. This actually happens with the RISM data. See, for example, the bottom incipit in Figure 3.7, where not only the octaves are encoded incorrectly for some notes, but there is also a mismatch of the time signature and the contents of measures.

The pitch coordinate

We work with Walter Hewlett's [24] Base-40 notation, which captures more information about pitch than what is stored in MIDI files. The Base-40 notation distinguishes between notes with the same pitch, but different notations. Like MIDI

Note Number	Measure Number	Pitch40	Duration	Position in bar
1	1	0	1920	0
2	2	221	1440	0
3	2	209	480	1440
4	3	204	1440	0
5	3	215	480	1440
6	4	209	480	0
7	4	198	480	480
8	4	186	960	960
9	5	192	1920	0

Table 3.1: The database contents for the melody shown in Figure 3.1. There are 960 divisions per quarter note, and rests are coded as notes with pitch 0. They are stored in the database, but not represented as weighted points. To arrive at Figure 3.1, we first normalize the time coordinates (i. e., divide them by 960). The durations are then: 1920/960=2, 1.5, 0.5, 1.5, 0.5, 0.5, 0.5, 1, 2. All measure lengths are 1920/960=2. Therefore, the note onset times are: 0, 2, 3.5, 4, 5.5, 6, 6.5, 7, and 8. This still includes the leading rest, which we want to ignore, so finally, we skip the leading rest and subtract its duration from all subseqent notes: 0, 1.5, 2, 3.5, 4, 4.5, 5, 6. These are the time coordinates in Figure 3.1.

pitches, it is a number-line representation of musical pitch notation, but with the added advantage of being interval-invariant. I. e., the difference between any two base-40 pitch numbers will correctly determine the notated interval name between those pitches.

Weights

Increasing a note's weight increases the importance of it having a counterpart of similar weight at the same position in the compared melody. A natural method of using weights is to make them reflect note durations. That way, differing note durations at corresponding positions lead to an increase in the resulting distance. For instance, in Figure 3.1 the note weights reflect only the durations. All results in this chapter were obtained with weights that only depend on note durations. By adding more components, however, additional desirable effects could be achieved. Two promising weight components are stress weight and note number weight.

Stress Weight. There are cases where melodies clearly differ, but a distance measure which ignores the positions of notes within measures fails to distinguish between them. For example, the two melodies in Figure 3.2 would not be distinguished by the simple distance measures used for Figures 3.6 and 3.7. By adding more weight to notes at positions in measures which are usually emphasized, e. g. the first beat, the measure structure can be taken into account as well.

Note Number Weight. In the RISM database, there are no clear rules about how many notes are included in the incipits. Therefore, it happens that very similar or identical melodies differ mainly in the number of notes that are included in the

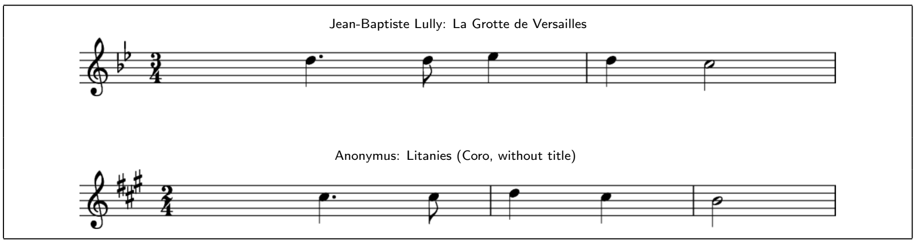

Jean-Baptiste Lully: La Grotte de Versailles
Anonymus: Litanies (Coro, without title)

Figure 3.2: By adding a stress-based weight component, the distance measure can be made to reflect different measure structures. Without that, the distance would be zero for these clearly different melodies, provided that transpositions are allowed.

incipit. As we shall see later, for example in Figure 3.7, there are instances where the distance between melodies becomes very large because one of them is cut off after fewer notes, not because they contain very different musical material. One possible way of addressing this problem is to add an extra weight component to each note that depends on how many notes precede it. That way, notes close to the beginning are made more important than extra notes at the end which might not be present in all occurrences of a melody in the database.

In Section 3.1.3, we will describe some adjustments of the signatures which we do before applying a distance measure.

3.1.2 Dissimilarity Measures for Weighted Point Sets

For a dissimilarity measure (formally speaking, a function on a set S, $d : S \times S \to \mathbb{R}^+ \cup \{0\}$), the following properties are usually desirable:

i. *Self-identity*: For all $x \in S, d(x, x) = 0$.

ii. *Positivity*: For all $x \neq y$ in $S, d(x, y) > 0$.

iii. *Symmetry*: For all $x, y \in S, d(x, y) = d(y, x)$.

iv. *Triangle inequality*: For all $x, y, z \in S$, $d(x, z) \leq d(x, y) + d(y, z)$.

A measure with all of these properties is called a metric, while a measure with only properties i, iii, and iv is called a pseudo-metric. Depending on the application, different properties are relevant. For measuring melodic similarity in a way that agrees with human perception, our measure should have the self-identity property. Symmetry might be useful as well, but it is less clear whether humans really apply a symmetric dissimilarity measure to melodies. The triangle inequality is useful for efficiently searching the database [5]. Positivity is not necessarily always desired. The EMD's partial matching property, which is closely related to its lack of positivity, can be useful.

In the following subsections, we will describe the two transportation distances which we used.

The Earth Mover's Distance (EMD)

The Earth Mover's Distance between two weighted point sets measures the minimum amount of work needed to transform one into the other by moving weight.

Intuitively speaking, a weighted point can be seen as an amount of earth or mass; alternatively it can be taken as an empty hole with a certain capacity. We can arbitrarily assign the role of the supplier to one set and that of the receiver/demander to the other one, setting, in that way, the direction of weight movement. The EMD then measures the minimum amount of work needed to fill the holes with earth (measured in weight units multiplied with the covered ground distance). See Cohen's Ph.D. thesis [14] for a more detailed description of the EMD.

Definition Let $A = \{a_1, a_2, .., a_m\}$ be a weighted point set such that $a_i = \{(x_i, w_i)\}, i = 1, .., m$, where $x_i \in \mathbb{R}^k$ with $w_i \in \mathbb{R}^+ \cup \{0\}$ being its corresponding weight. Let W=$\sum_{j=1}^{n} w_i$ be the total weight of set A.

The EMD can be formulated as a linear programming problem. Given two weighted point sets A, B and a ground distance d, we denote as f_{ij} the elementary flow of weight from x_i to y_j over the distance d_{ij}. If W, U are the total weights of A, B respectively, the set $\mathcal{F}$ of all possible flows $F = [f_{ij}]$ is defined by the following constraints:

1. $f_{ij} \geq 0, i = 1, ..., m, j = 1, ..., n$
2. $\sum_{j=1}^{n} f_{ij} \leq w_i, i = 1, ..., m$
3. $\sum_{i=1}^{m} f_{ij} \leq u_j, j = 1, ..., n$
4. $\sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij} = min(W, U)$

These constraints say that each particular flow is non-negative, no point from the "supplier" set emits more weight than it has, and no point from the "receiver" receives more weight than it needs. Finally, the total transported weight is the minimum of the total weights of the two sets.

The flow of weight f_{ij} over a distance d_{ij} is penalized by its product with this distance. The sum of all these individual products is the total cost for transforming A into B. The EMD(A, B) is defined as the minimum total cost over $\mathcal{F}$, normalized by the weight of the lighter set; a unit of cost or work corresponds to transporting one unit of weight over one unit of ground distance. That is:

$$\mathrm{EMD}(A, B) = \frac{\min_{F \in \mathcal{F}} \sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij} d_{ij}}{min(W, U)}$$

Properties and Computation. The most important properties of the EMD can be summarized as follows:

1. The EMD is a metric if the ground distance is a metric and if the EMD is applied on the space of equal total weight sets.
2. It is continuous, in other words, infinitesimal small changes in position and/or weight of existing points cause only infinitesimal change in its value. Moreover, the addition of a point with an arbitrarily small weight, i. e. noise (which can be seen as increasing its weight from zero to a positive value) leads to an arbitrarily small change in the EMD's value.
3. It does not obey the positivity property if the sums of the weights of the two sets are not equal. In that case, some of the weight of the heavier distribution

remains unmatched. Therefore, the EMD allows for partial matching. As a result, there are cases where it does not distinguish between two non-identical sets. Sometimes this can be useful, for example when two incipits contain identical melodies which are cut off after different numbers of notes. On the other hand, this also leads to effects like the one we see with incipit number 12 in Figure 3.6, where the EMD yields a relatively low distance. Here the surplus of weight is not all concentrated at the end of the melody, but distributed over several rests and other places, which leads to a false positive.

4. In the case of unequal total weights, the EMD does not obey the triangle inequality. A simple counterexample would be three melodies called A, B, and AB. Let us assume that AB is the concatenation of A and B, and let us assume that A and B are chosen so that the EMD yields a distance of 1 between them. If A and B are positioned accordingly, both the distance between A and AB and the distance between B and AB can be zero (because both A and B are parts of AB). Then, $d(\text{A, B}) > d(\text{A, AB}) + d(\text{AB, B})$.

   As a result, methods that rely on the triangle inequality for speeding up database retrieval cannot be used in conjunction with the EMD.

The EMD can be computed efficiently by solving the corresponding linear programming problem, for example by using a streamlined version of the Simplex algorithm for the transportation problem [25]. We used Rubner's [66] EMD function, which implements Hillier's and Lieberman's Simplex algorithm. It is possible that the Simplex algorithm performs an exponential number of steps. One could use polynomial algorithms like an interior point algorithm, but in practice that would outperform the Simplex algorithm only for very large problem sizes. Since the transportation problem is a special case of the minimum cost flow problem in networks, a polynomial time algorithm for that could be used as well.

The Proportional Transportation Distance (PTD)

Giannopoulos and Veltkamp [52] proposed a modification of the EMD in order to get a dissimilarity measure based on weight transportation such that the surplus of weight between two point sets is taken into account and the triangle inequality still holds. They call this modified EMD the "Proportional Transportation Distance" (PTD) because any surplus or shortage of weight is removed in a way that the proportions are preserved before the EMD is calculated. The PTD is calculated by first dividing, for both point sets, every point's weight by its point set's total weight, and then calculating the EMD for the resulting point sets.

The PTD is defined as follows: Let A, B be two weighted point sets, W, U the total weights of A and B, and d a ground distance. The set $\mathcal{F}$ of all feasible flows $F = [f_{ij}]$ from A to B is defined by the following constraints:

1. $f_{ij} \geq 0, i = 1, ..., m, j = 1, ..., n$
2. $\sum_{j=1}^{n} f_{ij} = w_i, i = 1, ..., m$
3. $\sum_{i=1}^{m} f_{ij} = \frac{u_j W}{U}, j = 1, ..., n$
4. $\sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij} = W$

The PTD(A, B) is given by:

$$\mathrm{PTD}(A, B) = \frac{\min_{F \in \mathcal{F}} \sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij} d_{ij}}{W}$$

Constraints 2 and 4 force all of A's weight to move to the positions of points in B. Constraint 3 ensures that this is done in a way that preserves the old percentages of weight in B.

The PTD is a pseudo-metric. In particular, it obeys the *triangle inequality*. It still does not have the *positivity* property since the distance between positionally coinciding sets with the same percentages of weights at the same positions is zero. However, this is the only case in which the PTD distance between two non-identical point sets is zero. The PTD will distinguish between two sets B and B' which differ in only one point. It has all other properties of the EMD for equal total weight sets.

3.1.3 Ground distance, adjustments of coordinates and weights

The ground distance

For all results in this book, we used the Euclidean distance as ground distance, unless specified otherwise. That is, the distance between two notes with the coordinates (t_1, p_1) and (t_2, p_2) is $\sqrt{(t_1 - t_2)^2 + (p_1 - p_2)^2}$.

An interesting variation, especially for polyphonic music, would be to make the distance in the pitch dimension depend on harmony instead of just calculating the difference of pitches.

Adjustments of coordinates and weights

Before applying one of the dissimilarity measures for weighted point sets described above, we adjust the signatures of the two melodies we want to compare in several ways:

- In order to be able to recognize augmented or diminished versions of a melody as similar (like for example in Figure 3.7, second group, melody 4 in comparison with the melody at the top in the same figure), it can be necessary to normalize the range of time coordinates. We chose to stretch the melody with the smaller maximum time coordinate over a longer time such that after the adjustment, both melodies' maximum time coordinates equal the larger maximum time coordinate before the adjustment. Note that with a less careful normalization, e. g. the adjustment of a randomly chosen melody to the other melody's length, one can easily lose the symmetry property. We did this alignment of durations when we used the PTD, where there is no partial matching; when we used the EMD, we compared the distances with and without duration alignment and took the minimum. We left the weights unchanged for the EMD.

- It is desirable to make the distance measure independent of transpositions. This could be done by moving one of the two melodies up or down in pitch to a position where the distance is minimal [50]. Since finding the optimum transposition would require the repeated application of the dissimilarity measure, which would take a lot of time, we chose to transpose one of the melodies so that the weighted average pitch is equal. This way, the dissimilarity measure for weighted point sets needs to be applied only once, but this is not always the optimum solution. However, this approximation usually works well enough for transposed versions of the same melody to appear closer than other melodies from the database, see Figures 3.6 and 3.7.

- When the transportation distance is calculated, the transportation of weight from one note to another can happen in the time dimension, the pitch dimension, or a combination of the two. Therefore, the range of numbers in both dimensions affects the results. For the comparison of our method with Howard's and Schlichte's, we multiplied the time coordinates with 3 in order to avoid all points to be placed in a very narrow, long strip like in Figure 3.1, where the pitch ranges from 186 to 221 (a range of 35), while the time only ranges from 0 to 6. An arrangement of the points like in Figure 3.1 would make it too cheap to move weight in the time dimension in comparison to the pitch dimension, which would lead to notes being matched with notes that do not really correspond with them.

  The optimum factor for the time coordinate needs to be determined experimentally such that the distance measure agrees well with human ideas of melodic similarity.

An example

Figure 3.3 shows the weight flow for signatures of two melodies after adjusting them as described above. Unlike the signature shown in Figure 3.1, the time coordinates are now multiplied with 3 so that weight transportation in the time and pitch dimensions are similarly expensive. In Figure 3.1, the range of pitches is much larger than the range of time coordinates so that transportation distance measures would match notes which do not correspond with one another. Also, the top melody in Figure 3.3 was stretched so that the maximum time coordinates are both 28.5, and the top melody was also slightly shifted in the pitch dimension so that the weighted average pitches are the same.Without the pitch and duration alignments, the distance between these two melodies would be 5.41825 instead of 0.739529. For the sake of simplicity, we treated the grace note in the bottom melody like any other eighth note, thereby overemphasizing it and influencing the time coordinates of subsequent notes. A special treatment of grace notes would probably lead to better results.

In Figure 3.3, an arrow indicates the flow and the transported weight for each pair of weighted points between which any weight is transported. Consider, for example, the first two notes of both melodies. Since the second melody starts with a dotted eighth note and a sixteenth note, while the first one starts with two eighth notes, half of the weight of the second note of the first melody is transported to the first note of the second melody, while the other half goes to the second note. The quarter note which is represented as a hollow circle is only partially matched. It has a capacity of 1, but only 0.5 weight units are transported into it.

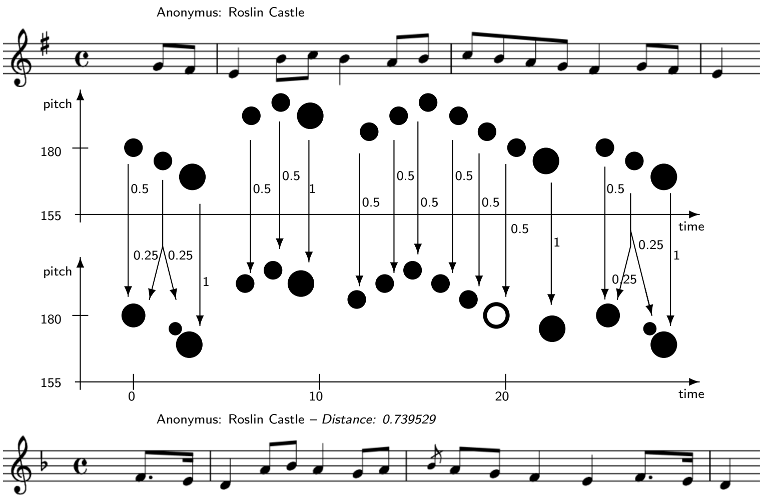


Figure 3.3: An illustration of a weight flow with the EMD; the coordinates are adjusted as described in Section 3.1.3. These two melodies are taken from the query result shown in Figure 3.6. The signatures of melodies 1 and 11 from Figure 3.6 after the adjustments, shown in the format (Time, Pitch; **Weight**): Top: (0, 180.138; **0.5**), (1.58333, 175.138; **0.5**), (3.16667, 169.138; **1**), (6.33333, 192.138; **0.5**), (7.91667, 197.138; **0.5**), (9.5, 192.138; **1**), (12.6667, 186.138; **0.5**), (14.25, 192.138; **0.5**), (15.8333, 197.138; **0.5**), (17.4167, 192.138; **0.5**), (19, 186.138; **0.5**), (20.5833, 180.138; **0.5**), (22.1667, 175.138; **1**), (25.3333, 180.138; **0.5**), (26.9167, 175.138; **0.5**), (28.5, 169.138; **1**) Bottom: (0, 180; **0.75**), (2.25, 175; **0.25**), (3, 169; **1**), (6, 192; **0.5**), (7.5, 197; **0.5**), (9, 192; **1**), (12, 186; **0.5**), (13.5, 192; **0.5**), (15, 197; **0.5**), (16.5, 192; **0.5**), (18, 186; **0.5**), (19.5, 180; **1**), (22.5, 175; **1**), (25.5, 180; **0.75**), (27.75, 175; **0.25**), (28.5, 169; **1**)

3.2 Advantages of transportation distances

From the definition, we can already see that using transportation distances for measuring melodic similarity has the following advantages:

- **Continuity:** If differences between queries and database documents are small, transportation distances deliver accordingly small values. When a query is distorted, there is no point at which the distance would suddenly become larger.

- **Support for many distortions:** Many kinds of differences such as grace notes, differences in pitch, note durations, and rhythm are taken into account by transportation distances without the need for their explicit anticipation.

- **Partial Matching for any combination of polyphonic and monophonic music:** With some transportation distances, examples of which include the EMD, any combination of monophonic and polyphonic music can be matched. If one of the two compared point sets has fewer notes, the EMD automatically picks the notes that match best and ignores those notes from the larger point set that do not have corresponding notes in the smaller point set.

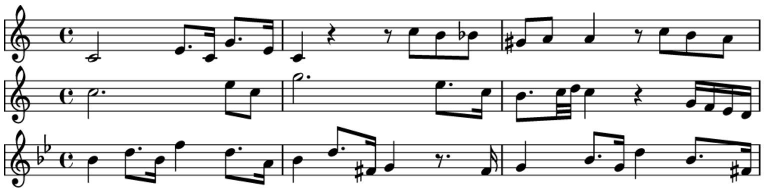


Figure 3.4: Three incipits that all start with a sequence of the intervals ascending major third/descending major third/ascending fifth/descending minor third. Schlichte [69] uses this example for pointing out that interval sequences alone are not suitable for finding melodically similar RISM incipits. The composers: top: Schröter, Johann Samuel; middle: Schuster, Joseph; bottom: Eichner, Ernst.

- **Flexibility:** Transportation distances can be fine-tuned to genres and human perception by modifying the weighting scheme and ground distance.

In the following sections, we will look at how various variants of our distance measure perform in comparison with other methods.

3.3 Matching incipits against other incipits

In this section, we compare the performance of the transportation distance method for measuring the similarity of RISM incipits to earlier attempts of comparing RISM incipits. For different problems, such as finding short queries in long pieces of music, we need to develop the method further. This will be discussed in subsequent sections.

3.3.1 Comparison with Schlichte's "Frankfurt Experience"

Joachim Schlichte [69] describes an attempt of grouping similar incipits together. This work was based on 83,243 incipits from the RISM A/II collection. Schlichte shows that omitting "insignificant" musical phenomena immediately leads to useless results, even among the small subset of 83,243 out of the 476,000 incipits that are currently available to us. The methods he classified as useless are:

- Converting the incipits into strings of intervals and comparing those, thus ignoring rhythm, absolute pitch, and rests, leads to distances of zero between very different pieces. Schlichte gives some examples, one of which is shown in Figure 3.4.

- Comparing strings of pitches, ignoring only rhythm and rests, still leads to too many false positives. Transposing all pieces into the same key before applying this method makes matters worse.

Schlichte therefore only looked for identical incipits, which already give music scholars valuable pointers to interesting facts. One of the interesting applications is the identification of anonymous pieces. Schlichte writes that among the 14,000 anonymous works in his data collection, 292, i. e. about 2 %, can be automatically associated with a composer by looking for identical incipits. This comparison is based on the Plaine & Easie encoding [27] in which all RISM A/II incipits are stored.

We compared approximately 80,000 incipits by unidentified composers in the RISM A/II collection to all other incipits; the result can be seen at http://give-lab.cs.uu.nl/MIR/anon/idx.html. About 13 % of these incipits lie within a distance of less than 1 (PTD; weights: duration only; base-40 pitches; time coordinates multiplied with 3) from other incipits. This includes trivial cases where the incipits are identical, but also more interesting cases like the one shown in Figure 3.5, where added notes, augmentation, transposition, and differences in rhyhtm, contour and the sequence of intervals make it more difficult to recognize the similarity.

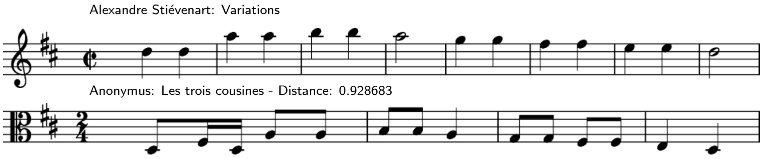

Alexandre Stiévenart: Variations
Anonymus: Les trois cousines - Distance: 0.928683

Figure 3.5: These two versions of the "Ah! vous dirai-je Maman" theme are recognized as similar (with PTD, weights: duration only). Note the extra notes in the second to last measure of Stiévenart and the first measure of Anonymus, which lead to differences both in the sequence of intervals and the contours, and the fact that the Stiévenart version is an augmented and transposed version.

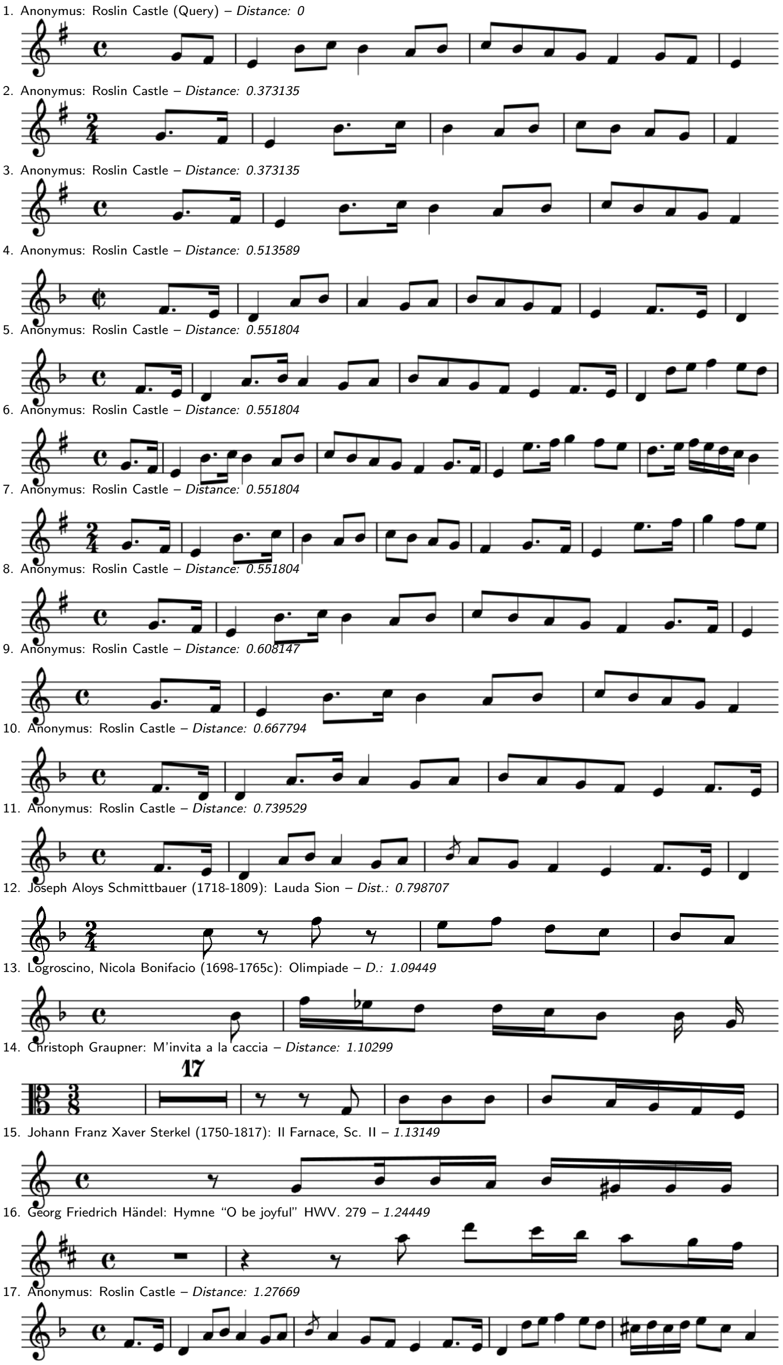
In order to see how many of these matches are actually useful and would not have been found by just looking for identical pieces, we manually checked 100 randomly chosen search results with distances below one. 55 % of these works only match with other anonymous works. For 19 %, a composer could be found because the compared incipits are identical. This is similar to Schlichte's result – 19 % of 13 % are 2.47 %, while Schlichte's figure is 2.08 %. We expect a slightly higher percentage because we do not compare the Plaine & Easie encodings, but our database contents as described in Section 3.1.1, which means that we view more melodies as identical than Schlichte did. For example, we ignore beaming. For another 11 %, we could determine the composer although the incipits are not identical. Therefore, our method leads to the identification of about 3.9 % of all anonymous pieces instead of Schlichte's 2.08 %.

3.3.2 Comparison with Howard's "Harvard Experience"

Howard [28] describes a later attempt of grouping together similar incipits from the RISM A/II collection. This work was based on a subset of our collection with at most half as many[1] incipits. The U. S. RISM officials did not, like their Frankfurt colleagues, compare Plaine & Easie encodings, but converted the incipits into the DARMS [27] encoding language. They compared sorting results of five encoding types:

1. the complete encoding with all parameters,
2. the complete encoding transposed to a common pitch register,
3. the encoding stripped of such features as beaming, bar lines, and fermatas,

[1] Howard does not clearly say how many, but from the introduction to his paper it can be inferred that the number was probably below 230,000.



1. Anonymus: Roslin Castle (Query) – *Distance: 0*

2. Anonymus: Roslin Castle – *Distance: 0.373135*

3. Anonymus: Roslin Castle – *Distance: 0.373135*

4. Anonymus: Roslin Castle – *Distance: 0.513589*

5. Anonymus: Roslin Castle – *Distance: 0.551804*

6. Anonymus: Roslin Castle – *Distance: 0.551804*

7. Anonymus: Roslin Castle – *Distance: 0.551804*

8. Anonymus: Roslin Castle – *Distance: 0.551804*

9. Anonymus: Roslin Castle – *Distance: 0.608147*

10. Anonymus: Roslin Castle – *Distance: 0.667794*

11. Anonymus: Roslin Castle – *Distance: 0.739529*

12. Joseph Aloys Schmittbauer (1718-1809): Lauda Sion – *Dist.: 0.798707*

13. Logroscino, Nicola Bonifacio (1698-1765c): Olimpiade – *D.: 1.09449*

14. Christoph Graupner: M'invita a la caccia – *Distance: 1.10299*

15. Johann Franz Xaver Sterkel (1750-1817): Il Farnace, Sc. II – *1.13149*

16. Georg Friedrich Händel: Hymne "O be joyful" HWV. 279 – *1.24449*

17. Anonymus: Roslin Castle – *Distance: 1.27669*

Figure 3.6: Query results for "Roslin Castle" among 476,000 melodies. The top 17 matches of an EMD-based query contain 12 occurrences of "Roslin Castle". For a discussion of the differences between EMD (this figure) and PTD (Figure 3.7), see Section 3.3.2.

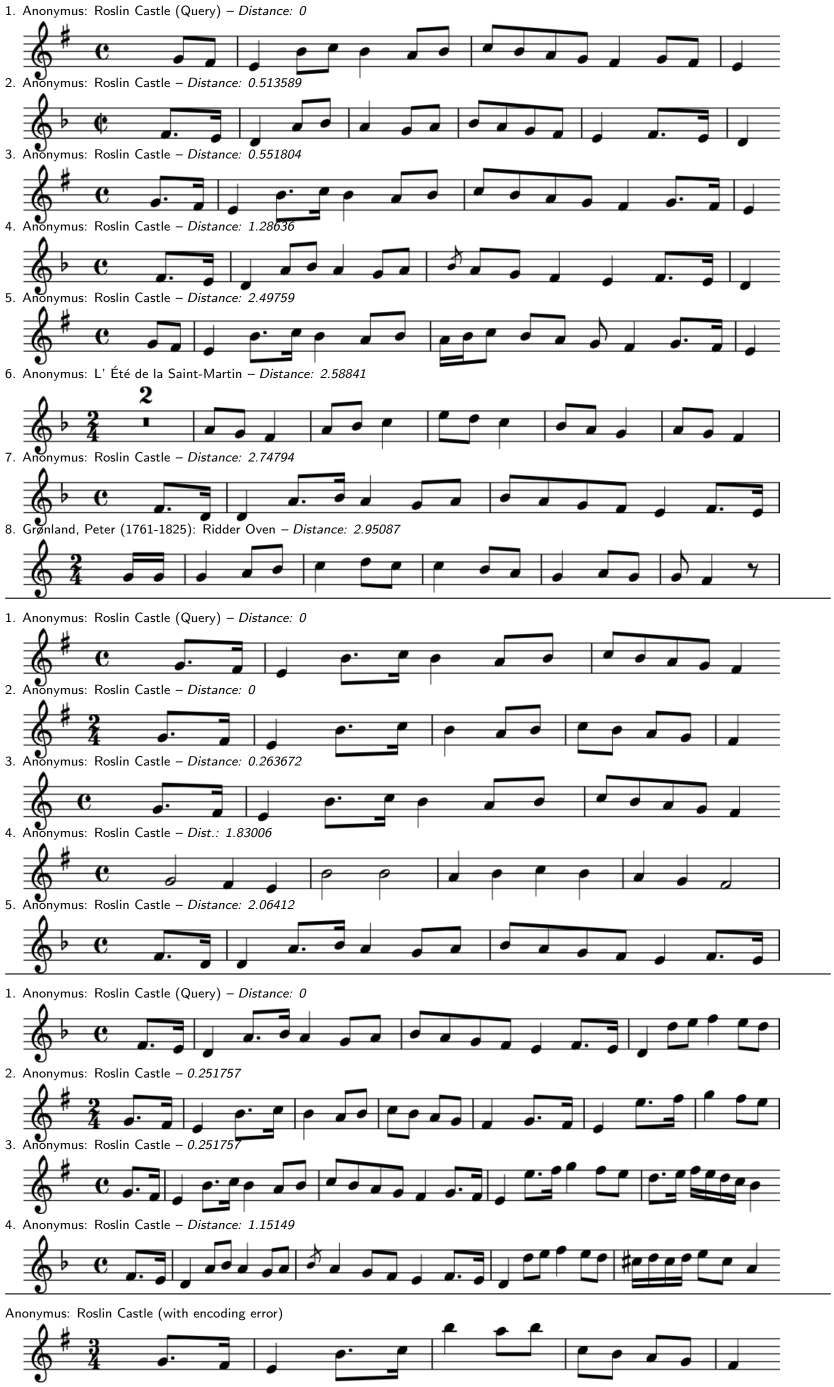

1. Anonymus: Roslin Castle (Query) – *Distance: 0*
2. Anonymus: Roslin Castle – *Distance: 0.513589*
3. Anonymus: Roslin Castle – *Distance: 0.551804*
4. Anonymus: Roslin Castle – *Distance: 1.28636*
5. Anonymus: Roslin Castle – *Distance: 2.49759*
6. Anonymus: L' Été de la Saint-Martin – *Distance: 2.58841*
2
7. Anonymus: Roslin Castle – *Distance: 2.74794*
8. Grønland, Peter (1761-1825): Ridder Oven – *Distance: 2.95087*
1. Anonymus: Roslin Castle (Query) – *Distance: 0*
2. Anonymus: Roslin Castle – *Distance: 0*
3. Anonymus: Roslin Castle – *Distance: 0.263672*
4. Anonymus: Roslin Castle – *Dist.: 1.83006*
5. Anonymus: Roslin Castle – *Distance: 2.06412*
1. Anonymus: Roslin Castle (Query) – *Distance: 0*
2. Anonymus: Roslin Castle – *0.251757*
3. Anonymus: Roslin Castle – *0.251757*
4. Anonymus: Roslin Castle – *Distance: 1.15149*
Anonymus: Roslin Castle (with encoding error)

Figure 3.7: Out of all 16 known occurrences of "Roslin Castle", 15 were retrieved with 3 PTD-based queries whose results are separated with horizontal lines. There is an encoding error which prevents the 16th occurrence from being shown in other query results.

4. the encoding stripped of the items given in (3) plus grace notes,
5. the encoding stripped of the items given in (3) and (4) plus rhythmic values, rests, and ties, with the transposition to a common register (2) but with preservation of repeated notes.

None of the five encoding types lead to more than 6 out of 13 known occurrences of a song called "Roslin Castle" being sorted together among not more than 230,000 incipits.

Figures 3.6 and 3.7 show the results of some queries for the same song with the EMD (3.6) and PTD (3.7). The EMD query groups together 11 out of 16 known occurrences among 476,000 incipits, i. e. a larger percentage among more than twice as many potential false positives in comparison with the "Harvard experience". If one does not count the 16th occurrence, shown at the bottom right, because it is not encoded correctly, our method compares even more favourably (73 % versus 46 %). The PTD result shown in Figure 3.7 does not group together more occurrences, but at least the false positives are musically very similar.

Figures 3.6 and 3.7 also illustrate the different properties of the EMD and PTD:

- The EMD groups more occurrences together in just one query result. Among the 16 known occurrences of "Roslin Castle", there are 3 groups with similar numbers of notes. The fact that the EMD allows partial matching, while the PTD matches all notes, leads to a clear distinction of these groups by the PTD, but not the EMD.
- Because of the weight normalization, the PTD recognizes augmented or diminished versions of the same melody as similar. In the second group of melodies in Figure 3.7, melody number 4 is recognized as similar to the other melodies in the group, while the use of the EMD leads to a rather large distance. With the EMD, the different durations mean that aligning the time coordinates (as described on page 29) would be necessary for recognizing the similarity here. However, unfortunately, these two incipits do not cover corresponding portions of the melodies, so this adjustment fails to make the similarity recognizable. In later sections, we show how problems like this can be overcome by segmenting.
- The false positives in Figure 3.7 (numbers 6 and 8 in the first group) are more similar to the musical material in the rest of the query result than the false positives in Figure 3.6 (numbers 12 to 16). The reason is that the EMD allows an unmatched weight surplus to be spread over the whole melody. In other words, this distance measure does not distinguish between a few extra or missing blocks of notes on the one side and differences between many individual notes or rests on the other side. When the PTD is used, blocks of extra or missing notes lead to the wrong notes being compared to one another, which usually dramatically increases the distance. Any differences between individual notes are also penalized.

3.4 Matching query segments against whole incipits

The melody comparison algorithm as it is described in the previous sections relies on the compared incipits having roughly the same length. Only then does the alignment to a fixed length make musical sense. In order to overcome this restriction, we modified the algorithm such that the query is segmented, and the incipits in the collection are searched for occurrences of translated and scaled versions of the query segments. The resulting algorithm was submitted for the Symbolic Melodic Similarity task at MIREX 2005; see Section 5.3.1 for a discussion of its performance.

3.4.1 Segmenting the query

Segmenting queries can lead to an improvement of partial matching in the time dimension and an increase of robustness against pitch and/or tempo fluctuations, even if the segments do not make musical sense. If queries are entered by humans, the pitch and/or tempo frequently fluctuate. While such fluctuations can greatly distort a query, they either do not have a large impact on short segments, or only on a few of them. The partial matching in the time dimension is improved if the smaller point set is translated and scaled such that the transportation distance to the larger point set is minimized.

For our experiments, we worked with segment lengths in the range from 6 to 9 consecutive notes. Segments of this length are usually distinctive enough so that we did not get too many matches from pieces that are not really similar, but still short enough for getting the desired robustness against tempo and pitch fluctuations.

Our segmenting algorithm must fulfill certain conditions for our method to work properly. We would like to be able to process manually recorded MIDI queries with free, possibly fluctuating tempo and unknown measure structure. Also, we want the segmenting results to be largely independent of how many voices are present at the same time. Therefore, we cannot just take a certain number of notes and declare them a segment, and we also cannot rely on the measure structure. Instead, we look at a certain number of consecutive notes.

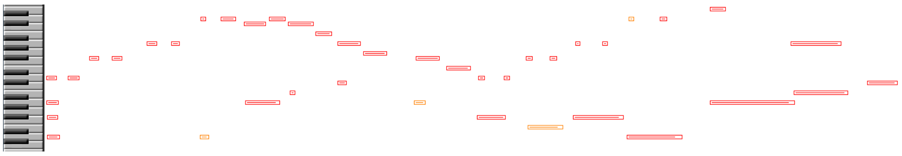


Figure 3.8: A polyphonic query for Bach's Brandenburg concerto No. 5 (violin part plus the left hand of the cembalo). This query was played by hand on a MIDI keyboard, which lead to slightly inexact rhythm.

We count consecutive notes in monophonic or polyphonic music as follows:

1. First, we set a pointer to the onset time of the first note that is to become part of the next segment. This is the beginning time of a new segment.

2. Then, we move the pointer to the next end of any note whose onset time lies within the current new segment, then to the next beginning of a note. We do this until we have the desired number of consecutive notes in the segment.

3. We include all notes with an onset time within the closed interval from the beginning of the segment to the current pointer position in the next segment.

For example, segment number 1 in Figure 3.9 is found like this: First, we move a pointer to the onset time of note number 1, the first note we want to include in the

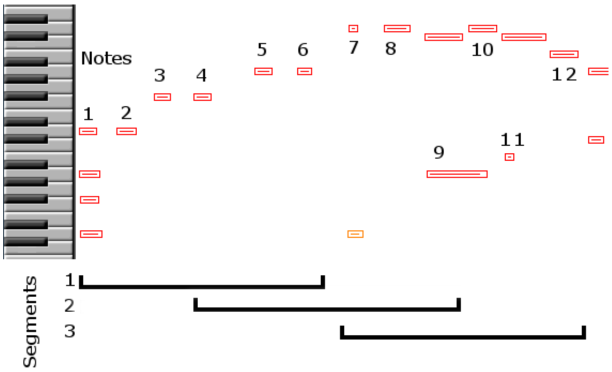


Figure 3.9: The first three segments of the polyphonic query shown in Figure 3.8.

segment. Then, we move the pointer to the end of the longest note in the beginning chord (the lowest note), because that is the next ending of any note whose onset time lies in the current segment. Now we move the pointer to the beginning of note 2 since this is the next onset time after the pointer. This way, we have included the whole chord at the beginning in the new segment, but count it as only one out of six steps. The next five steps work the same way (go to the next end of any note after the pointer, then to the next onset time). As a result, we identify the first segment as shown in Figure 3.9 with its 9 notes as a segment with 6 consecutive notes.

In order to correctly recognize consecutive legato notes in MIDI queries as consecutive (for getting a legato effect, the player releases piano keys only after the following note has started), it was sufficient to treat all notes as if they were only 80% of their length for the purpose of segmenting.

By segmenting queries, we increase the number of comparisons of point sets that are necessary for answering a query. On the other hand, the individual comparisons become simpler since smaller point sets need to be compared, and the size of point sets is bounded. The number of comparisons grows linearly with the query length.

3.4.2 Search algorithm

The segmented search for incipits works as follows:

1. Represent melodies as weighted point sets as described in Section 3.1.1.

2. Segment the query into possibly overlapping segments with a given number of consecutive notes. The length of these segments is largely independent on the number of voices since we do not count all notes in the segment, but the number of notes that follow one another. For details on this segmenting method, see Section 3.4.1.

3. Find a good alignment of the two point sets to be compared. Scaling in the time dimension and translation of a point set in both time and pitch dimensions are allowed since neither tempo changes nor transposition or the position of a melody within a piece fundamentally change the character of a melody. For aligning two point sets such that the EMD is minimized, we use

the evolutionary optimization function evofunc [45] of the library "Reusable Evolutionary Algorithms in Shape Matching" (REALISM), part of the Shape Matching Environment (SHAME). To find a good alignment of two point sets A and B, Evofunc creates a population of individuals; each individual is a transformation of point set A, characterized by three parameters that specify a combination of translation in the time and pitch dimension and scaling in the time dimension. The fitness of each individual is calculated by determining the distance between the corresponding point set (a stretched and translated version of A) and B. New individuals are created from the old ones in a fashion similar to simulated annealing, and this process of assessing individuals and creating new ones is repeated until a certain threshold of quality or quality improvement is reached. See Figures 3.10 and 3.11 for an illustration of the alignment problem.

4. Use the EMD between the optimally aligned point sets as distance measure.

5. Now we have one result list of matching documents and their distances for every query segment. Combine them as follows:

   - Calculate normalized distances as follows: Let e be the EMD distance between a query segment and a document. Let c be a large cutoff distance that is larger than a distance observed when there is some meaningful melodic similarity. Pick ε with $0 < \varepsilon < 1$. The normalized distance is 1 for EMD distances $\geq c$, and it is $\varepsilon + \frac{(1-\varepsilon)e}{c}$ otherwise. This ensures that the normalized distance is in the interval $[\varepsilon, 1]$.
   - For every document that occurs in any list of matches for a query segment, construct a list that contains: query segment number, normalized distance to the document, and the onset times of the first and last query note within the matched document. This list might contain the same query segment number multiple times (if it matches at more than one place within the document), and it might contain different query segment numbers (if more than one query segment matches the document).
   - For every document, calculate an overall distance score by finding the minimum product of normalized distances for a legal combination of segments that match with this document. By "legal combination", we mean that:
     - No segment number occurs twice
     - Segments with a higher number are matched at a later position within the document
     - For segments with consecutive numbers: the overlap of the matched areas corresponds to the overlap of the segments within the query (if there is any), and the matched areas are not too far apart (within plus or minus 10 % of the values one would expect from the query).

The performance of this algorithm was compared to other state-of-the-art melody search algorithms at MIREX 2005. Our algorithm ended up in the middle of the field, but was the highest-ranking algorithm that would also support polyphony and distorted queries. However, the runtime was rather high: it took 51240 seconds to search 581 incipits for 11 queries on a Dual AMD Opteron 64 1.6GHz machine. See Section 5.3.1 for details of the MIREX 2005 results.

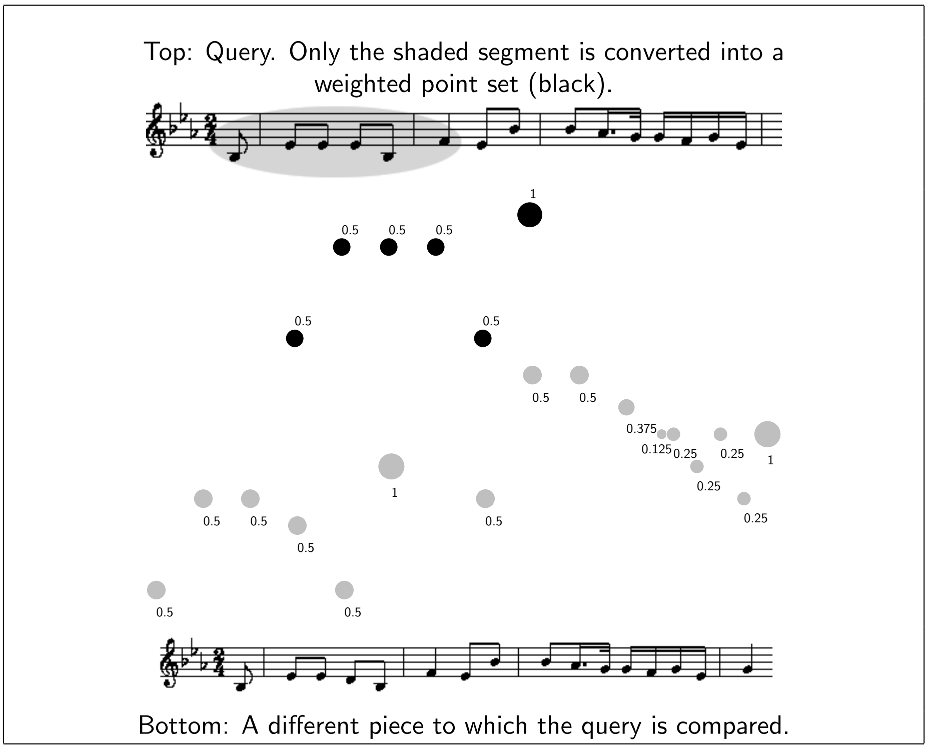

Top: Query. Only the shaded segment is converted into a weighted point set (black).
1
0.5 0.5 0.5
0.5 0.5
0.5 0.5
0.375
0.125 0.25 0.25 1
0.25
1
0.5 0.5 0.5
0.5
0.25
0.5 0.5
Bottom: A different piece to which the query is compared.

Figure 3.10: Problem: Before calculating the EMD, we need to somehow find out that the black point set should be moved to the beginning of the grey one for minimizing the EMD.

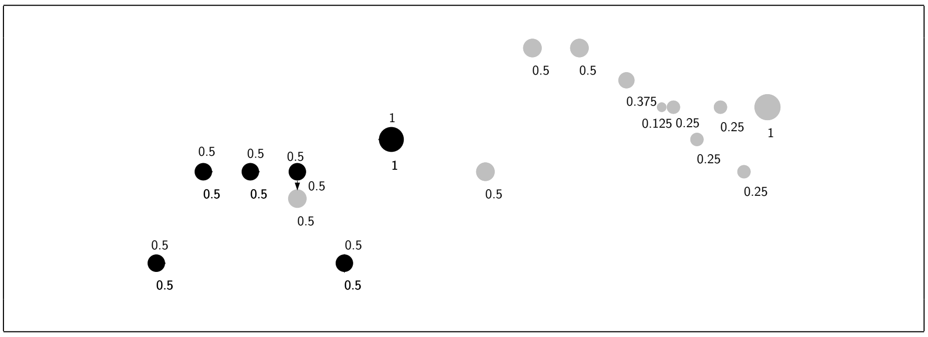


Figure 3.11: The optimum alignment of the two point sets from Figure 3.10 so that the EMD is minimized. The black points without arrows hide grey points.

3.5 Segmenting both the query and database documents

Much of the runtime our algorithm needed at MIREX 2005 was spent on finding the optimum alignment of query segments and incipits such that the EMD was minimized. The fact that this is an NP-complete problem makes this tolerable only as long as the size of the compared point sets is bounded by a low constant, as is the case with musical incipits. However, the simple fact that finding a good alignment takes some effort is not the only problem in this context: another problem is that allowing combinations of translation in the pitch and time dimensions and of scaling in the time dimension also makes it impossible to rely on the vantage indexing method (see Chapter 4).

Both problems can be avoided, and the method can be made suitable for searching complete pieces of music instead of just incipits, by not only segmenting queries, but also the pieces to be searched. While this approach increases the number of items to be compared, it avoids the need for expensive alignments. If the segments are created in the same way, the corresponding point sets can just be scaled so that they cover the same amount of time and translated (now only in the pitch dimension) such that the EMD is minimized. Therefore, one does not need to find an optimum combination of translation and scaling anymore. The increase in the number of items to be compared can be addressed with indexing techniques such as vantage indexing.

In the next sections, we will study several different variants of the algorithm that use different combinations of segment sizes. The next section covers the case of one fixed segment size for both query and database items (using the segmenting algorithm described in Section 3.4.1), and Section 3.5.2 deals with the case of multiple segment sizes for database items. For the latter variant, we first split polyphonic pieces into monophonic voices and then cut the voices into monophonic segments such that the Proportional Transportation Distance can be used in combination with vantage indexing, which guarantees that the indexing does not cause false negatives. If one works with sufficiently many different segment sizes for database items, one also has the option of not segmenting the query at all. We will also report experimental results with this variant.

3.5.1 A single segment size for database items and query

By segmenting not only the query, but also the database items using the segmenting algorithm described in Section 3.4.1, we avoid the need for finding an optimum combination of translation and scaling that would align a query with a matching piece such that the distance is minimized. Translation is still supported since every query segment is compared to segments from any position within the database items, and scaling (varying tempo) is supported because every query segment and every segment in the database are scaled to a fixed duration before being compared.

Algorithm description

The algorithm involves the following steps:

- Segment all items in the database using the segmenting algorithm from Section 3.4.1.
- When a query needs to be answered:
  - Segment the query in the same way.
  - For each query segment, retrieve the most similar segments from the database.
  - Combine the results of the segment searches into one overall score.

Each segment search yields a list of pieces that contain at least one matching segment. The overall result should be a list of pieces with many closely matching segments. For this, we need to compute a distance for each piece that occurs in at least one segment search result. To do this, we first determine the maximum distance M that occurs in any segment search result. For each segment search result in which a piece P occurs, we add the distance of the highest ranked segment of P to the overall score for P. For each segment search result in which P does not occur, we do not know the distance of the corresponding segments because it was high enough for the segment of P to not occur in this result list. Therefore, it is at least the maximum distance in this result list, but probably clearly higher. We get good results if we add twice M to the overall score for P in such cases. For each segment search result without a segment from P that is both preceded and followed by segment search results with segments from P, we add 4 times M to the overall score for P. If the query is really a subset of the database document P, there should not be a section within P that does not match, therefore there should be a higher penalty for missing segments within the query than for missing segments at the beginning or end of a query.

The resulting overall score is a distance measure. It is zero if for every segment of the query a matching segment with distance zero was found in the same database document P. The distance measure grows with the individual distances of segments and with the number of segments for which no matches were found. Although the transportation distances are symmetric, the combined distance measure is not. Also, the triangle inequality does not hold, and it is not always positive for unequal pieces of music. Therefore, it is not a metric.

Adjusting the search radius for different segments For each segment, we perform an n nearest-neighbours search up to a given maximum search radius m.

When using the vantage indexing method (see Chapter 4), we cannot directly search for n nearest neighbours, but need to work with a search radius. This radius has to be different for different segments if we want to retrieve similar numbers of neighbours. For typical musical patterns, like many repeated notes within one segment, there tend to be many more neighbours within a small radius around the segment than for very distinctive patterns of notes.

To select an appropriate search radius for each segment, we proceed as follows: The search starts with a given low initial value which is unlikely to be too large for any segment. If during the search we find more than n neighbours with distance zero, the segment is not distinctive enough to be considered at all, and this segment search can be stopped immediately. There are segments that do not contain enough characteristic musical material for being helpful. If at the end of the search, not enough matches (less than the n nearest neighbours we are looking for) were found within the search radius, we increase the radius and search again. In this case, it is sufficient to search the area outside the original search radius, but within the new, enlarged one. We do this only while the search radius is less than the given maximum search radius m.

Comparison with PROMS and C-BRAHMS

Lemström et al. [41], [42], [85] propose a number of algorithms for searching notated music. Their search engine prototype, which uses these algorithms, is called "C-Brahms". We compare our search results to those produced by three of their algorithms that support polyphonic queries. The other C-BRAHMS algorithms mentioned in Section 2.2.2 were excluded from this comparison because they cannot be used for matching polyphonic data with other polyphonic data. These C-BRAHMS algorithms were included in our comparison:

- **P1:** Find translations of the query pattern such that *all* onset times and pitches of notes in the query match with some onset times and pitches of notes in the database documents. Time complexity: $O(mn)$ (where m is the number of notes in the query and n the number of notes in the database document).

- **P2:** Find translations of the query pattern such that *some* onset times and pitches of the query match with some onset times and pitches of database documents. Time complexity: $O(mn \log m)$.

- **P3:** Find translations of the query pattern that give longest common shared time (i. e., maximize the times at which query notes sound at the same time and with the same pitch as notes from the database documents). This algorithm does not take into consideration whether onset times match. Time complexity: $O(mn \log(mn))$ or, for a constant, finite number of possible pitch levels: $O(mn \log m)$.

PROMS [11] is an efficient system for finding supersets of queries, where queries are sets of notes described by onset time and pitch. Therefore, it essentially performs algorithm P1 in an efficient way, so we do not need a separate comparison with the results that PROMS would deliver for the same queries and database. In 2006, Clifford et al. [13] proposed a randomized maximum subset matching algorithm with complexity $O(n \log n)$ that solves a task similar to P3. It works, like PROMS, on quantized score encodings and is faster than the minimum effort of $O(n^2)$ for an exact solution thanks to the randomization and because it only looks for an approximation.

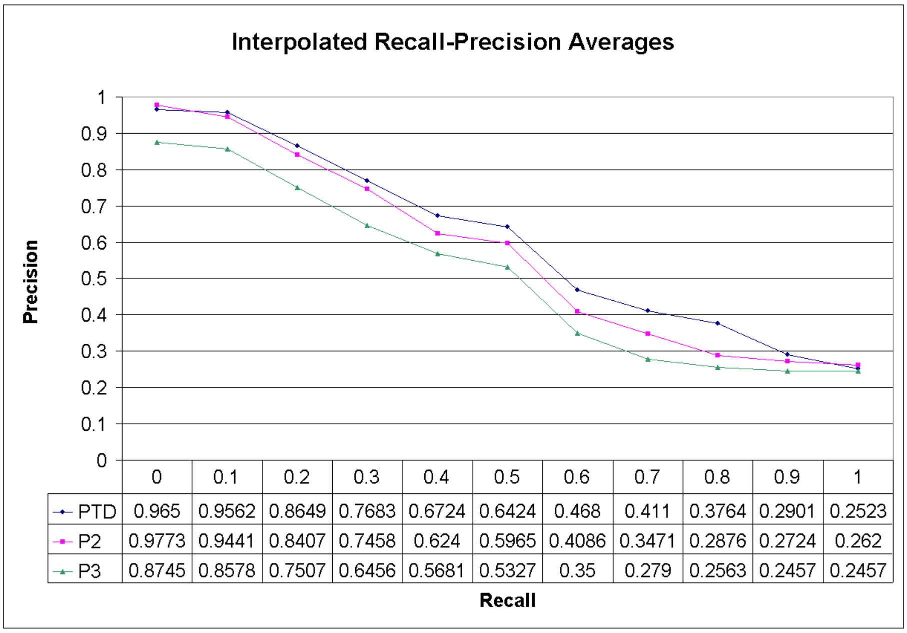
Lemström et al. do not propose any indexing method for their algorithms, so for any search, the query has to be compared with each database document. PROMS uses an inverted file index that requires a quantization of note onset times and the knowledge of the measure structure of queries and database documents. Our method can be used with the vantage indexing method described in Chapter 4, without requiring the measure structure to be known or the onset times to be quantized beyond what is needed for representing real numbers using a computer's finite memory.

Experiment We randomly selected 44 incipits from our database of about 476,000 incipits and used them as queries. We cut off every result list at position 25 and, if the 26th document was ranked the same as the 25th, removed all documents of that rank (otherwise we would have made a very arbitrary selection of matches since there were cases with hundreds of documents with the same rank as the 25th). This gave us a total of 3563 matches (4 methods, 44 queries, and up to 25 matches each), for each of which we decided whether it was melodically similar and therefore relevant.

Using van Zwol's testbed [88], we decided about the relevance in a way that minimized the influence of any bias towards one method. For each query, we created one combined result list containing all documents that were returned by any search method. Those that were returned by more than one method were listed only once. These lists were not sorted by method or by the ranks of documents, but by the library holding the source manuscripts. Therefore, for every relevance decision it was very hard to tell which method had retrieved the document in question. We had four people make the relevance decisions, two of whom had not worked on our project before. Each person covered half of the 44 queries, and every match was judged by two people.

Generally, P1, P2, and P3 do not properly match patterns if for some notes, the pitch is slightly distorted, and they also do not work if the tempo is not constant. Since for our comparison, we searched the RISM A/II collection for pieces similar to queries taken from the same collection, there were no pitch or rhythm distortions. But even if we do not take this advantage of using transportation distances into account, our method still compares favourably with P1, P2 and P3 (and therefore also PROMS).

See Figure 3.12 for a recall-precision graph and numbers of retrieved relevant documents. For the purpose of this graph, we assumed that all relevant documents were retrieved by some method. We do not have a ground truth for our 44 queries. There are probably some relevant documents that were not retrieved by any method. Therefore, the actual precision is probably lower for all methods, but a comparison is still possible. Out of all 275 relevant documents found by all methods combined, our method retrieved 179, P1 only 76, and P2, the second-best method, 166. Thus, our method found 8 % more relevant documents than P2 and 136 % more than P1. For every method, we looked at queries where it performed badly, and describe the individual weaknesses in the following subsections.



	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
PTD	0.965	0.9562	0.8649	0.7683	0.6724	0.6424	0.468	0.411	0.3764	0.2901	0.2523
P2	0.9773	0.9441	0.8407	0.7458	0.624	0.5965	0.4086	0.3471	0.2876	0.2724	0.262
P3	0.8745	0.8578	0.7507	0.6456	0.5681	0.5327	0.35	0.279	0.2563	0.2457	0.2457

Figure 3.12: P1 is not included in the graph because it does not rank matches. Among all documents that any method retrieved, 275 were judged to be relevant. Out of these, P1 found 76, P2 found 166, P3 found 125, and PTD found 179.

P1 (all onset times match) Algorithm P1 finds only supersets of the query. For most queries, only database documents containing exactly the same notes as the query turned out to be supersets. Whenever this was not the case, the retrieved document tended to be a false positive like the example shown in Figure 3.13. For this query, P1 only found two occurrences of the piece containing the query and two occurrences of the same false positive. The problem here is that P1 completely ignores any non-matched notes which occur between the query notes.

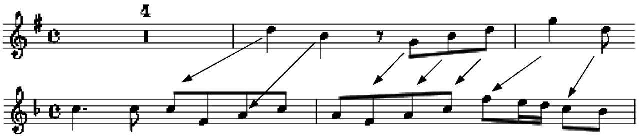


Figure 3.13: Top: Joseph Eybler: "Dies sanctificatus" (Query). Bottom: False positive, Pietro Guglielmi: "Domine ad adjuvandum". Arrows connect the notes that P1 matched with one another.

P2 (some onset times match) We ranked the P2 matches by the number of matching onset times. The highest-ranked matches are always identical to the P1 matches. Therefore, the problem illustrated with Figure 3.13 also applies to P2. For this query, Joseph Eybler's Gradual "Dies sanctificatus", P2 returns more than 500 matches, but fails to find two occurrences of Gregor Rösler's sacred song "Dies sanctificatus", which is melodically similar and has the same title and therefore should be ranked highly (see Figure 3.16). The Rösler incipits were missed by P2 because they are diminished, one note is a third higher, and there is a slight variation in rhythm. This is a good example for alterations that do not change the character of a melody very much, but greatly decrease the number of matching onset times.

For an example of a match that was found by P2, but missed by P1, see Figure 3.14.

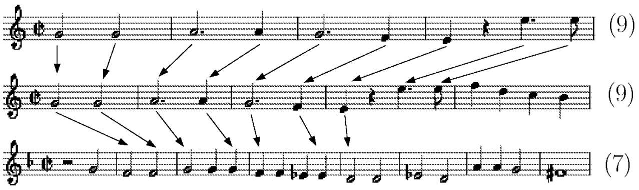


Figure 3.14: The top three matches for a Kyrie by Jan Vitásek, as returned by P2. The query is shown at the top, and the number of matched onset times is written in parentheses. There are many more matches with 7 matching onsets. The partial match shown here (a song titled "En jungfru stålter haver lag") is clearly not the same piece, but it is similar enough to be relevant.

P3 (maximize shared common time) Like P1 and P2, P3 finds all matches that are a superset of the query. False positives are caused by a problem somwhat similar to P1's: P3 does not take additional onsets into account which greatly change the perception of a melody, but not necessarily the overlap. An illustration of this problem can be found in Figure 3.15, which shows a query for which P3 finds

113 quite different matches, all with the same maximum shared common time. In the bottom incipit (Palestrina), the extra notes between matching notes make the melody sound quite different from that of the top incipit (Haydn).

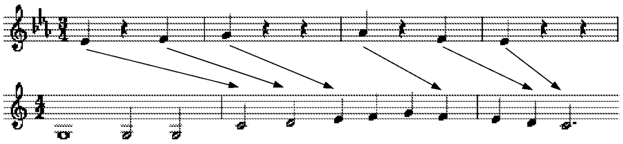


Figure 3.15: Top: Joseph Haydn: Symphony in E flat, Hob. 1.91, first movement (query). Bottom: G. P. da Palestrina: Offertorium "Confitebuntur caeli". This is one of the 113 matches where the common shared time is maximal (equal to the total time of the query). Most of the other 112 matches have just as little in common with the query.

PTD (transportation distance) We used the PTD for comparing our segmented search method since for the almost exclusively monophonic RISM A/II collection, the EMD's partial matching capability is not needed.

Like P1, P2, and P3, the segmented PTD searches always resulted in top ranks for the exact matches. An example of a true positive found by PTD, but missed by P1, P2, and P3 is shown in Figure 3.16.

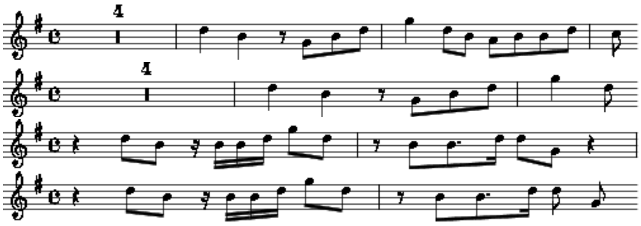


Figure 3.16: The first four matches for Joseph Eybler's "Dies sanctificatus", returned by a segmented PTD search. The distance is 0 for the top two matches, which are both occurrences of the Eybler piece, and 1.333 for the next two, which are both occurrences of Gregor Rösler's sacred song "Dies sanctificatus".

Although the PTD performs better than the other methods, P2 found some relevant documents that the PTD missed. Usually, the reason was the same as for the example shown in Figure 3.14. Here, our segmenting method does not extract corresponding groups of notes from the query and the database document. This can be addressed by creating multiple segments with different numbers of consecutive notes, all starting at the same note, instead of only one segment with 6 consecutive notes. This technique is described in more detail in Section 3.5.2.

Conclusions The comparison of our method with PROMS and C-Brahms indicates that our method delivers better results, is faster than C-Brahms for realistically large numbers of documents (hundreds of thousands or more), and puts fewer constraints on data and queries.

- **Better results:** For the RISM A/II collection, our method finds more relevant documents than any of the three polyphonic C-Brahms algorithms P1, P2 and

P3. Our method finds more than twice as many relevant documents as P1, of which PROMS is essentially an efficient implementation. However, we did not evaluate the possibility of fuzzy queries that PROMS/notify offers. Those are equivalent to sets of non-fuzzy queries.

- **More efficient algorithm:** If we use the vantage indexing method described in Chapter 4, our algorithm needs $O(k \log n)$ L$_\infty$ norm calculations plus k transportation distance calculations (for point sets representing music segments whose duration is bounded by a constant), where k is the number of reported point sets and n the number of music segments in the database. This is better than the complexity of the best C-Brahms algorithm in our comparison, P2. Because of the lack of a suitable indexing structure, P2 needs $O(d)$ comparisons, each of them with a complexity of $O(ij \log i)$, where d is the number of documents, i the query length, and j the length of the compared document. Even if there was a suitable indexing structure which would reduce the number of comparisons, P2 would still need more time for longer documents, while for our method, only the sum of all document sizes matters, not the sizes of individual documents. In practice, with half a million of database documents on a 1-GHz machine with 1 GB of memory, answering queries with the C-Brahms algorithms takes hours instead of seconds or minutes with transportation distances.

- **Fewer constraints on data:** Unlike PROMS/notify or C-Brahms, our method is continuous, and it supports both tempo and pitch fluctuations. The "notify" system supports tempo fluctuations, but no pitch fluctuations, and C-Brahms works only if neither tempo nor pitch fluctuate. Our database only contains pieces without tempo or pitch fluctuations, but when used with the EMD, our method can correctly match the distorted query shown in Figure 3.8 with Bach's Brandenburg concerto No. 5.

  With our method, the query does not have to be a subset of database documents like for PROMS/notify.

3.5.2 Multiple segment sizes

Cutting both the query and the database items into segments of one fixed size and then only matching those segments eliminates a useful degree of freedom that was still present in the variant where only the query is segmented (Section 3.4): the possibility of matching different numbers of notes against one another. Often, variations of the same melody contain different numbers of notes, for example in the case shown in Figure 3.17.

This problem can be addressed by segmenting the database documents into segments of multiple sizes and then matching each query segment against all segments (with multiple sizes) at once. Figure 3.18 shows an example of how a hummed query can be cut into overlapping segments which are then matched against segments like those shown in Figure 3.19.

The MIDI file which was converted into the sheet music shown in Figure 3.19 does not contain the upbeat that the query contains (the very first note). The first six note segment from the query therefore has no counterpart with corresponding six notes in the MIDI file. However, because on each note, segments of length 5,

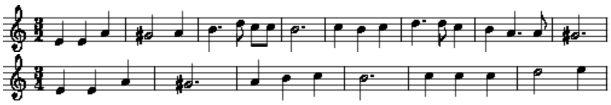


Figure 3.17: An example for the usefulness of segmenting both the query and the database items and using multiple segment sizes for the latter. Neither the whole incipits nor n-grams with a fixed n contain corresponding melodic material, but the whole incipits are still melodically similar. (Top: John Dowland, "If fluds of tears could clense my follies past". Bottom: a violin duet by Josephus Fodor.)

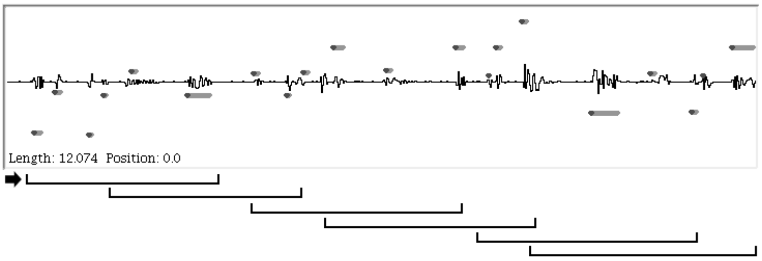


Figure 3.18: A hummed query. The monophonic query is cut into overlapping segments of length 6. The wavy line in the background shows the volume over time, while the horizontal bars represent the recognized notes.

6, and 7 start, there is still a rather similar segment – the segment with 5 notes starting on the very left of Figure 3.19 (marked with a little arrow).

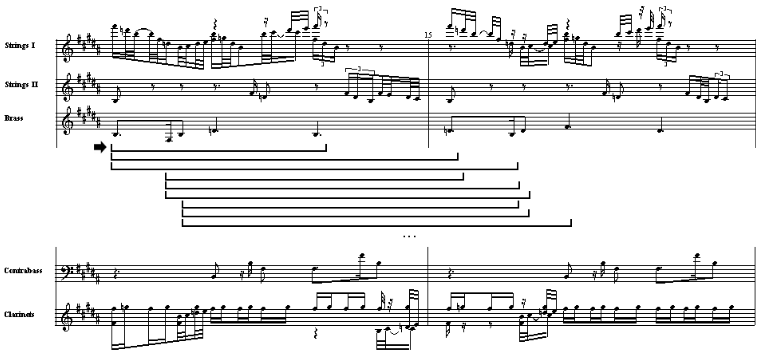


Figure 3.19: Each voice is made monophonic and then cut into segments of lengths 5, 6, and 7. On every note, three segments start. The number of segments is still in $O(N)$ (N=number of notes in the piece). This figure shows just a few of the many segments that are created from these notes (an excerpt from Wagner's "Ride of the Valkyries").

Different combinations of segment sizes for queries and database items are suitable for different circumstances. If one has enough space for letting the segments get as long as the longest expected query, one can even consider not segmenting the query at all, but instead cutting the database items into segments with lengths that cover the entire supported range for query lengths.

Generally, when deciding on how exactly to segment queries and database items, one needs to take the following effects into consideration (here we assume that the pieces in the database are always segmented):

Parameter	Choice	Advantages	Disadvantages
Segmenting the query	Yes	Increases the robustness against *major distortions* of the query in a *few places,* such as the insertion of random notes with random pitches (this can happen if a Query-by-Humming system's note recognition fails) or user input errors such as unintended modulations	Increases computational complexity (multiple searches for queries, consolidation of results)
	No	Increases the robustness against *minor distortions* of *many* query notes (this can happen if the user of a Query-by-Humming system has poor intonation or rhythm and gets almost every query note wrong; he might still be able to produce a query whose overall shape is approximately correct).	Increases space complexity since the database items need to be split into segments of many different lengths to support the wide range of possible query lengths.
Query segment length	long	Longer segments are more distinctive and lead to fewer matches from the database. This reduces the effort for consolidating multiple results for query segments into one overall result.	Longer segments reduce the robustness against few major distortions since larger portions of the query are affected.
	short	A shorter query segment length implies a shorter minimum query length.	Segments are less distinctive (the extreme case of query length 1, for example, would mean that anything matches perfectly).

Continued on next page

– continued from previous page

Parameter	Choice	Advantages	Disadvantages
Query segment lengths	one fixed length	A fixed query segment length means that consolidating results is slightly simpler since every query segment can be considered equally important.	If the query length is not a multiple of the segment length minus the segment overlap, one needs to either ignore some notes at the end of the query or over-represent some notes.
	multiple lengths	With multiple query segment lengths, one can try to combine the robustness advantages of long and short query segments.	More query segments lead to higher computational complexity, and more query segment lengths imply a higher space complexity for the database since the database items will also need to be cut into segments with more different lengths.
Database segment lengths	one fixed length	To cover the possibility of ornamentations, one could also use a single size for all database documents but multiple, overlapping query segment sizes. Advantage: lower space complexity.	Higher computational complexity (quite a few segment searches since multiple segments need to start at every query note).
	multiple lengths	The case of ornamentations is covered without the need to split the query in many overlapping segments of different lengths. This means a lower computational complexity.	Higher space complexity.

For the Symbolic Melodic Similarity and Query by Singing/Humming tasks at MIREX 2006, we tried several combinations of these choices for different situations. For a description of the different subtasks and how the various algorithms performed, see Section 5.3.2. The following subsections describe how the different subtasks were addressed with different combinations of segmenting queries and database items.

Multiple segment sizes for both query and data (MIREX 2006, RISM collection)

For the RISM collection of musical incipits, the task was to find incipits that are similar to a given incipit. The incipits differ in length, and it is quite possible that the query is longer than some desired matches. Therefore, both the query and the database items need to be segmented. If only the database items, but not the query would be segmented, matches that are shorter than the query would be hard to find.

The incipits were split into segments of 5 to 16 consecutive notes, while the query was split into segments of varying sizes from 5 consecutive notes up to either 16 or the length of the query, whatever is lower. That way, shorter incipits can be matched to parts of the query, but the whole query can still be matched to longer incipits in one comparison.

The algorithm from Section 3.5.1 needs to be adjusted so that it takes different segment lengths into consideration, which has an impact especially on how partial search results are combined. The following algorithm was used for the MIREX submission:

- Segment all items in the database using the segmenting algorithm from Section 3.4.1. At every note that has at least $x - 1$ subsequent notes, create a segment of length x, where x ranges from 5 to 16.
- When a query needs to be answered:
  - Segment the query: create segments that start with the first note of the query and are 5 to N notes long, where N=16 or the query length, whichever is lower. If the query is longer than 16 notes, also create segments of length 6 and an overlap of 3 that cover the whole query.
  - For each unique query segment, retrieve candidates for the most similar segments from the database by using a vantage index (see Chapter 4). This results in a list of segments that probably have a low EMD distance, along with an estimate for that distance. However, the actual distances are not calculated. The estimate is based on how similar the distances to the set of vantage objects are.
    Since we know that the collection contains only musical incipits, not whole pieces, we only consider matches where the first query segment matches near the beginning. A threshold of 3.5 seconds in the MIDI rendition (about four quarter notes) for the offset of the matching region worked well.
  - Combine the results of the segment searches into one overall score. For every combination of matching segments from the same piece whose relative positions in the piece match those in the query, calculate the score as follows:
    - ∗ Determine the expected number of notes that should be covered by matching segments. This is either the query length or the length of the matching piece, whichever is lower.
    - ∗ Count the actual number of notes that are covered by at least one matching query segment.
    - ∗ Add the vantage index-based estimate for the EMD distance of every matching segment to the overall score.

    - Add a penalty p to the score for every note that should be covered but is not. For MIREX 2006 and the RISM collection, p was 1. Its value determines whether the coverage or the distances of individual segments have a higher impact on the overall distance scores.
    - So far, the dissimilarity scores were just estimates based on the vantage index, which are easy and fast to calculate, but may underestimate the actual EMD distances (see Chapter 4 for an explanation of how vantage indexing can lead to false positives). To eliminate such false positives, re-calculate the overall scores based on actual EMD distances for the top 50 results.

  - At this stage of the search, the result is already quite reasonable. To further improve precision and recall near the top of the result list, it is, however, useful to slightly lower the scores for items that contain especially many notes which exactly match their counterparts in the highest-ranked match. For the 50 items with the lowest scores, this is done by multiplying the score with a factor f which is calculated as follows: $f = 1/3 + \frac{2(1-m/c)}{3}$, where m is the number of notes that exactly match their counterparts in the query, and c is the total number of notes considered (that is, the minimum of query length and the length of the matching incipit).

    Whether this last step is useful or not depends on whether the searched collection is quantized or not. Only if it is quantized, the concept of an exactly matching note is helpful. For counting matching notes, the incipits were compared to the highest-ranking incipit instead of to the query because the query was not always quantized, while the highest-ranking incipit was both quantized and very similar to the query.

This method gave good results at MIREX 2006. No competing algorithm gave better results, and only one algorithm needed a few seconds less for answering queries. For a detailed comparison of the results, see Section 5.3.2.

The possibility of matching a long query to a shorter incipit introduces another problem: The scoring system described above does not distinguish between perfect matches of a whole incipit that is shorter than the query (distance: zero) and perfect matches of the whole query with parts of a longer incipit (here the distance is also zero). Assigning different distances (a lower distance to the latter case, where more notes match), would be an improvement, but one would need to investigate how large such a difference should be in order to agree with human ideas of similarity.

No query segmenting, but multiple segment sizes for the data (MIREX 2006, Karaoke collection)

Not segmenting the query but instead using more different segment lengths for the data to be searched lowers the response time for queries. Only one segment search is necessary for answering a query, and there is no need to combine search results for different query segments. The main drawback is that more segment sizes for the pieces in the database mean that the index gets larger. Another small drawback is that queries cannot be matched to pieces that are shorter than the query.

For the MIREX 2006 Karaoke collection, which contains 1,000 complete pieces, the advantages seemed to outweigh the disadvantages.

This is the – somewhat simpler – algorithm when the query is not segmented:

- Cut the polyphonic pieces in the database into voices. It is assumed that every voice is stored in either its own track or channel. In a second step, a skyline algorithm is applied to make each of these extracted voices monophonic. By "skyline algorithm", we mean that whenever multiple notes sound at the same time, all but the highest note are deleted. This way, only the "skyline", the sequence of highest notes, remains.

- Segment all voices in the database using the segmenting algorithm from Section 3.4.1. At every note that has at least $x - 1$ subsequent notes, create a segment of length x, where x ranges from 5 to 16. Since pop music tends to be repetitive, only the first 80 notes in each voice are indexed.

- When a query needs to be answered:
  - If the query contains more than 16 notes: remove notes at the end such that the query is 16 notes long. Otherwise, do not change the query.
  - Retrieve candidates for the most similar segments from the database by using a vantage index (see Chapter 4). This results in a list of segments that probably have a low EMD distance, along with an estimate for that distance.
    The matching segments can come from different voices and from anywhere in the piece. Therefore, the algorithm supports cases where a melody wanders across voices, and it can find occurrences of a melody anywhere in a piece.
  - To eliminate false positives stemming from the vantage indexing (see Chapter 4), re-calculate the overall scores based on actual EMD distances for the top 50 results.
  - At this stage of the search, the result is already quite reasonable. To further improve precision and recall near the top of the result list, it is, however, useful to add a cost to noticeable tempo differences. Even though melodies generally do not change their character if the tempo is varied, if we completely ignore tempo, we will sometimes match fast accompanying figures to slower melodic lines. We divide the duration of the matching area in the piece by the duration of the query; if this quotient is less than 1, it is inverted so that the resulting tempo difference factor is greater than or equal to 1. If the tempo is really noticeably different (by a factor of more than 1.2), the calculated distance is multiplied with the tempo difference factor. Otherwise, it is left unchanged.

This method gave very good results at MIREX 2006, clearly better than all competing algorithms. Only one algorithm needed a few seconds less for answering queries. For a detailed comparison of the results, see Section 5.3.2.

Single segment size for the query, multiple segment sizes for the data (MIREX 2006, mixed polyphonic collection)

The mixed polyphonic collection at MIREX 2006 was too large for indexing it in the same way as the Karaoke collection and still keep the entire index in memory. The machines had just 2 GB of main memory, which we already filled with just three segments starting at every note. So, in order to not let the response times for queries get too long, the collection was cut into segments of 5, 6, and 7 consecutive notes only. This meant that queries had to be segmented using only a fixed segment size of 6 consecutive notes in order to support note insertions and deletions.

The algorithm is a combination of the two previous ones – polyphonic pieces are cut into voices like for the Karaoke collection, and results for different query segments are combined like for the RISM collection:

- Cut the polyphonic pieces in the database into voices. It is assumed that every voice is stored in either its own track or channel. In a second step, a skyline algorithm is applied to make each of these extracted voices monophonic.

- Segment all voices in the database using the segmenting algorithm from Section 3.4.1. At every note that has at least $x - 1$ subsequent notes, create a segment of length x, where x ranges from 5 to 7. Since pop music tends to be repetitive, only the first 80 notes in each voice are indexed.

- When a query needs to be answered:

  – Cut the query into segments containing 6 consecutive notes. Two consecutive query segments have an overlap of 3 notes, with the possible exception of the last query segment, which is placed such that it covers the last 6 notes of the query, even if this means that its overlap with the second to last segment is not 3. Every note in the query is part of at least one query segment.

  – For each unique query segment, retrieve candidates for the most similar segments from the database by using a vantage index (see Chapter 4). This results in a list of segments that probably have a low EMD distance, along with an estimate for that distance. However, the actual distances are not calculated. The estimate is based on how similar the distances to the set of vantage objects are.
    Like in the variant described in the previous section, the matching segments can come from different voices and from anywhere in the piece. Therefore, the algorithm supports cases where a melody wanders across voices, and it can find occurrences of a melody anywhere in a piece.

  – Combine the results of the segment searches into one overall score. For every combination of matching segments from the same piece whose relative positions in the piece match those in the query, calculate the score as follows:

    ∗ Determine the expected number of notes that should be covered by matching segments. This is either the query length or the length of the matching piece, whichever is lower.

    ∗ Count the actual number of notes that are covered by at least one matching query segment.

    ∗ Add the vantage index-based estimate for the EMD distance of every matching segment to the overall score.

    ∗ Add a penalty p to the score for every note that should be covered but is not. Like for the RISM collection, a value of 1 for p works well.

    - ∗ So far, the dissimilarity scores were just estimates based on the vantage index, which are easy and fast to calculate, but may underestimate the actual EMD distances (see Chapter 4 for an explanation of how vantage indexing can lead to false positives). To eliminate such false positives, re-calculate the overall scores based on actual EMD distances for the top 50 results.

  - – Like for the Karaoke collection, precision and recall near the top of the result list are improved by multiplying the overall distance score with the tempo change factor if that factor is greater than 1.2.

Like for the Karaoke collection, this method gave very good results at MIREX 2006, clearly better than all competing algorithms. Only one algorithm needed a few seconds less for answering queries. For a detailed comparison of the results, see Section 5.3.2.

A comparison of the runtimes for the 1000-item Karaoke collection and the 10,000-item mixed polyphonic collection shows that our algorithm scales well. Although the variant without query segmenting is computationally less expensive (no partial results need to be combined, and there is only one segment search), searching 10,000 polyphonic MIDI files with query segmenting took only 2.8 times as much time as searching 1000 without query segmenting. The fastest competing algorithm needed 8.5 times longer for 10,000 items.

3.5.3 Time and space complexity

If one is only interested in a given number of items from the database that are most similar to the query, our method can give an answer in $O(i \log n + k)$ time using $O(n)$ storage space, where n is the total number of notes in all items to be searched, i is the input size (query length), and k is the number of returned items. In practice, i and k will be bounded, so the time complexity for searching is essentially just logarithmic.

We will show this in detail for the most complex variants of our algorithm, the versions for MIREX 2006 (see Section 3.5.2).

Effort for indexing

The indexing method with vantage objects is described in Chapter 4. It involves computing the transportation distance between every item in the database and a fixed, small number of vantage objects. In our case, the items in the database are point sets representing segments of pieces from the database. The number of these segments lies in $O(n)$ for every variant of the algorithm since the segment length, the number of different segment lengths, and the number of segments beginning on any note are all bounded for all algorithm variants (we used, for example, lengths of 5 to 16 notes for the "Karaoke" task at MIREX 2006, and therefore 12 segments began at every note).

Since the number of vantage objects is a constant and the number of segments lies in $O(n)$, the size of the index table containing the distance to each vantage object for every item in the database lies in $O(n)$. To make searching this table efficient, we create a B-tree index for it, whose size is also $O(n)$, and whose construction can be done in $O(n^2 \log n)$ [16].

Although the algorithm for computing the EMD (we use the Simplex algorithm) sometimes can perform an exponential number of steps, this does not matter for us since we only run it on segments with constant sizes (for example, between 5 and 16 consecutive notes), so calculating the EMD (or PTD) between two segments always lies in $O(1)$.

Overall, for building an index, the space complexity lies in $O(n)$, and the time complexity is dominated by the effort of building a B-tree for the table containing the distances to vantage objects, $O(n^2 \log n)$. Building that table involves a constant number of EMD calculations that each take constant time for each segment; the number of segments in $O(n)$.

Effort for searching

For all variants from Section 3.5.2 (the MIREX 2006 variants), we need to consider the following steps:

- Segmenting the query: $O(i)$ (i is the query length).
- Calculating the distances to each vantage object for every query segment: $O(i)$ (since the number of query segments lies in $O(i)$ and their lengths and the number of vantage objects are bounded, and the vantage objects have constant sizes).
- For all query segments, finding candidates for matches using the vantage index: $O(i \log n)$ for a i range queries on the vantage table with length n using a B-tree.
- The length of the desired result list is k, therefore the number of matches to consider for each query segment lies in $O(k/i)$. Thus, identifying false positives from the vantage index and calculating overall scores for the bounded number of candidates lies in $O(k)$ (segment sizes are still bounded).

Overall, the effort for searching is $O(i + i \log n + k)$, where n is the total number of notes in the database, i is the query length, and k is the desired number of returned items. In practice, i and k will always be small. If the number of items to search is increased, the search time only grows logarithmically.

3.5.4 Conclusions

We have shown that our method for measuring melodic similarity performs well in terms of result quality as well as time and space complexity. At MIREX 2006, no algorithm got better results for monophonic music, and all other algorithms performed worse for polyphonic music. Space complexity lies in $O(n)$ and queries can be answered with an effort of $O(i + i \log n + k)$ (n is the number of notes in the database, i is the query length, and k is the desired number of returned items), so the response time for searching only grows logarithmically if more items are added to the database.

The fact that our distance measure is continuous and does not need quantized data or a known measure structure contributed to the good result quality even for distorted (hummed) queries and non-quantized data at MIREX 2006.

Chapter 4

Indexing music with vantage objects

The RISM A/II collection [1] contains about half a million musical incipits. For the simplest variant of our algorithm, without any segmenting, we therefore need to perform about half a million EMD calculations for one search. As we have mentioned in Section 3.1.2, the Simplex algorithm is a good choice for calculating the EMD. The Simplex algorithm has exponential time complexity in the worst case [33]; polynomial algorithms exist, but for typical problem sizes when comparing segments of music, they are unlikely to outperform the Simplex algorithm. Half a million of EMD calculations take many hours on a normal PC with a 1 GHz CPU and 1 GB of main memory. Therefore, for this algorithm to be useful in practice, the search needs to be sped up with a suitable indexing technique. This need becomes even greater for variants of the algorithm that involve segmenting of the query or the database items because this greatly increases the number of required comparisons of point sets; for example for MIREX 2006, the algorithm variant that works with three different segment lengths needed over 12 million segments to represent the first 80 notes of each voice from 10,000 polyphonic MIDI files. This chapter describes how one can search millions of segments within seconds while still using an expensive EMD-based distance measure.

4.1 Vantage indexing

Vleugels and Veltkamp [90] first suggested vantage indexing for image retrieval. The basic idea is that if the triangle inequality holds for a distance measure, one can save a lot of effort by pre-calculating the distances between a small set of vantage objects and each item in the database; whenever items need to be retrieved that are close to a given query item, one can then limit the database search to those items whose distances to the vantage objects are similar to those of the query item.

Figure 4.1 illustrates a simple case with just one vantage object. Before any query takes place, for every database item P_i, we calculate the distance to the vantage object V and store these distances along with the objects P_i in the database. To find the items that are closer to a query Q than some search radius r, it is now no longer necessary to calculate the distances between Q and all objects P_i; instead it is sufficient to calculate the distance between Q and V (let us call this distance

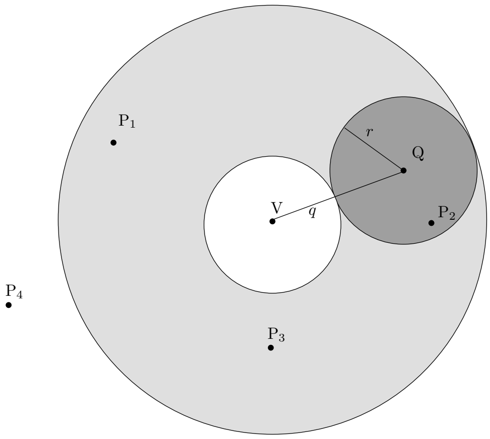


Figure 4.1: Wanted: all objects P within the dark grey circle around Q with radius r. For all objects P$_i$ in the database, the distance to V has already been calculated. For answering the query Q, we need to calculate Q's distance to the vantage object V. Now we can quickly exclude all objects outside the light grey ring around V.

q) and then restrict the search to those items in the database whose distance to V lies between $q - r$ and $q + r$. In Figure 4.1, we can exclude all points that do not lie within the light grey ring around V, such as for example P_4. Only for the points within the light grey ring, we still need to calculate the actual distance to determine whether these points lie within the dark grey area of interest.

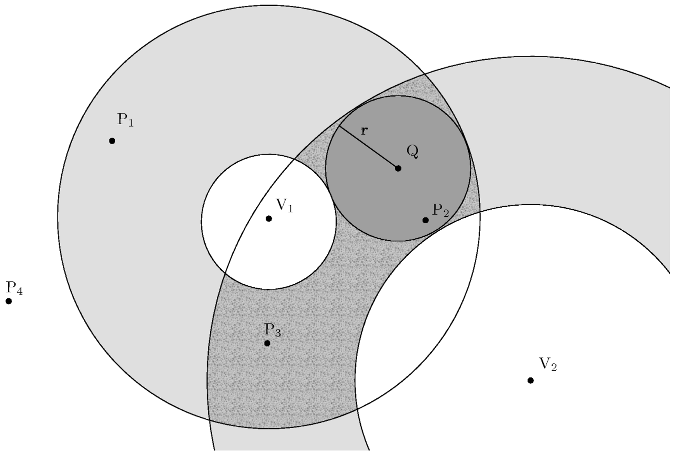


Figure 4.2: By using multiple vantage objects, one can exclude all objects outside the intersection of rings around the vantage objects.

As Figure 4.1 shows, the vantage index cannot reliably exclude all items outside the search radius from the search. There can be false positives such as P$_1$ and P$_3$ which lie inside the light grey ring, but not inside the dark area we are interested in. To reduce the number of false positives, we can work with multiple vantage objects.

Figure 4.2 shows how the situation changes if we add another vantage object. Now we need to pre-calculate and store two distances for every item in the database, but in return for this increased effort, we can exclude even more items from the search: only items that lie in the intersection of the two light grey rings around V_1 and V_2 are now candidates for matches. For example, this excludes P_1 from the set of false positives; now the only false positive is P_3. See Section 4.4 for experimental results about the effect of the number of vantage objects on search performance.

4.2 The EMD and the triangle inequality

The Earth Mover's Distance obeys the triangle inequality only if the total weight sums are the same in the two point sets. Otherwise, not even a weak triangle inequality such as EMD(A,B) $\leq k$ (EMD(A,C) + EMD(C,B)) holds (with $k \geq 0$). Counterexamples exist where EMD(A,B)>0, EMD(A,C)=0, and EMD(B,C)=0; see Figure 4.3.

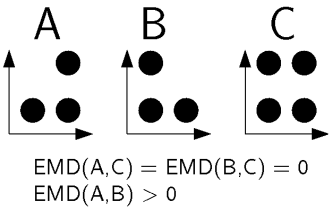


Figure 4.3: For the EMD, not even the weak triangle inequality holds. In this example, EMD(A,B) $> k$ (EMD(A,C) + EMD(C,B)) for all $k \geq 0$.

However, the experiments described in Section 3.3.2 indicated that with the RISM A/II collection, when using the vantage indexing method with the EMD, probably most matches within a third of the search radius are retrieved. Hence if the search radius is increased accordingly, the vantage indexing method can still be used for polyphonic searches with the EMD, albeit without a guarantee for the completeness of the matches. It is hard to quantify the necessary increase of the search radius or the probability of still retrieving all matches since this depends on the data collection. It is possible to construct a data collection where the radius needs to be much larger than three times that for the PTD, which does satisfy the triangle inequality.

4.3 Search radius versus nearest neighbour search

Vantage indexing can be used to support either nearest-neighbour searches[1] or searches within a given search radius. The introduction of vantage indexing in Section 4.1 illustrates the latter, but the principle of exploiting the triangle inequality can also be used in conjunction with kd-trees.[2] For an exact nearest-neighbour

[1] In a nearest-neighbour search, an item, a search space, a distance measure, and a number n is given. The desired result are the n items that are closest to the given item in the search space according to the distance measure.

[2] A kd-tree is an index structure that accelerates searches in a k-dimensional space. The space is partitioned by a hierarchy of splitting planes, each of which is associated with a node in a

search, querying a v-dimensional kd-tree would involve $O(n^{1-\frac{1}{v}} + k)$ L$_\infty$ norm calculations, where k is the number of reported points (each representing a set of notes). Arya et al. [2] describe an approximate nearest-neighbour search which would need $O(k \log n)$ L$_\infty$ norm calculations [2]. Their method uses a balanced box decomposition tree which recursively subdivides space into cells; each leaf cell is associated with one point. First they find the cell that contains the query point (with a simple descent through the tree); then they identify the closest cells and calculate the distances of the contained points from the query point. Once a point is found that is closer to the query point than the distance of any other cell times $(1 + \epsilon)$, this point can be reported as an approximate nearest neighbour, and the search can be terminated.

By using the vantage indexing method in combination with B-trees[3], it is easy to retrieve all segments that lie within a given search radius around a query segment with an effort of $O(\log n)$, with n the number of segments to search.

For music information retrieval, a search radius and a nearest neighbour search both have advantages and disadvantages.

If one works with a constant search radius for every segment (or incipit), one can end up with very many matches for segments that contain common musical material (for example, repeated notes or repetitions of triads) and, for a different query segment and the same search radius, with very few matches. Too many matches for a query segment slow down the search process, and too few matches may lead to false negatives.

On the other hand, if one considers the same number of nearest neighbours for every segment, this can also lead to very unequal treatment of the query segments. For segments that contain common musical material, one would only consider matches that are extremely similar, while for unusual segments, one would risk considering very different matches that are so far away from the query segment that a human would not see any melodic similarity.

Since both a constant search radius and a constant number of nearest neighbours are inappropriate in their pure forms, we use a combined approach. First we count how many segments lie within a certain initial search radius r_1 (this can be done efficiently with a B-tree). We pick this search radius so that it reflects a distance within which a human can recognize melodic similarities. For determining a good initial radius, it is useful to have a human-generated ground truth and also the results of the ground distance measure for some segments. If the number of matches within this radius is higher than c, the maximum number of segments we are willing to consider, we lower the search radius in several steps until either the number of matches drops below c or we reach a minimum search radius r_2 that reflects a minimum variation we want to permit in our search. If even this minimum

binary tree. For retrieving items from the search space that are close to a given position in the search space, one traverses the kd-tree by deciding at each node which side of the splitting plane to search (because the given position is on that side of the splitting plane) and therefore which subtree to traverse. A kd-tree supports nearest-neighbour searches because by traversing the tree, one finds points that are close to the given position. Depending on how many nearest neighbours are desired, one still has to do more than just traversing the tree from its root to a leaf in order to find them all. For more information about kd-trees, see, for example, [16].

[3] A B-tree is an access structure for sorted data. It uses a tree of logarithmic height for supporting range queries or queries for an item with a given key. Each internal node can have multiple pointers to child nodes with certain ranges of the key value. Thus, accessing the leaf node that contains the item or range of items of interest takes $\log n$ steps (with n the number of items). Nested B-trees can be used for searching data based on multiple different key values such as distances to multiple vantage objects. For more information about B-trees, see [16].

search radius gives us too many matches, the segment contains very uncharacteristic musical material and is ignored. The number c of segments we are willing to consider depends on the number of segments in the query (if there are very few, we can spend more effort on every segment), on the speed of our computer and on the maximum response time we are willing to tolerate.

4.4 The number of vantage objects

Increasing the number of vantage objects influences the search performance in various ways:

- The number of false positives is lowered (see Figures 4.1 and 4.2). This makes the next step, sifting through the candidates for matches, less computationally expensive.
- Searching the vantage index becomes computationally more expensive since the range query affects more columns of the index table.
- Indexing becomes computationally more expensive; for each item that is added to the index, more distances to a vantage object need to be calculated. However, this does not affect the asymptotic time complexity.
- The space complexity is accordingly increased; for each item in the index table, more distances need to be stored, and the index for the index table (for example, a B-tree of B-trees) also grows accordingly.

With the number of false positives, the number of vantage objects also influences the search radius that can be searched if the effort spent on identifying false positives is to remain constant, which is needed to keep the overall complexity of the search algorithm within $O(\log(n))$. Therefore, the optimum number of vantage objects depends not only on the collection and the choice of particular vantage objects, but also on the maximum effort that may be devoted to identifying false positives. In this section, we present some experimental results about what happens if the effort spent on identifying false positives is kept constant, but the number of vantage objects is varied. The next section will mention some results on how one can choose good vantage objects.

For our experiments, we used the UK subset of the RISM collection, which was also used for MIREX 2006, and for which a ground truth was determined as part of MIREX 2006 (see Section 5.3.2).

The left part of Figure 4.4 shows the result quality if the number of vantage objects is varied from 1 to 8. The maximum number of considered segments was kept constant at 20,000. That is, for every segment search, the search radius was automatically chosen such that the vantage index search lead to at most 20,000 candidates for matching segments. This explains why the result quality varies: with only one vantage object, there were so many false positives that very few, if any, truly similar segments happened to be among the 20,000 retrieved candidates for segment searches, and as a consequence, not a single relevant document was retrieved for the six queries from MIREX 2006. However, with two vantage objects, this already improved considerably, and the vantage index reached its optimum performance in the range from 5 to 8 vantage objects (the threshold of 20,000 was chosen with 8 vantage objects in mind). One should not take the small changes in

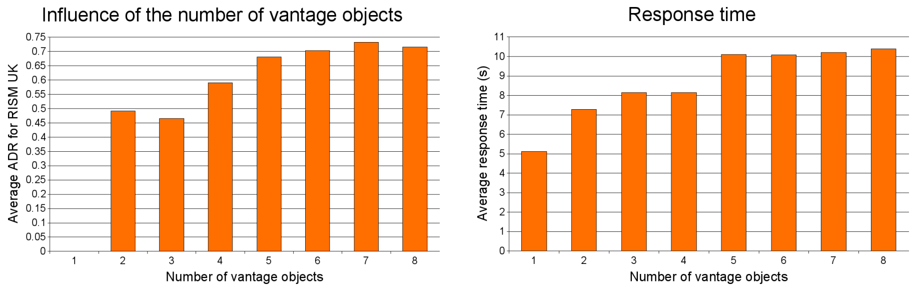


Figure 4.4: The influence of the number of vantage objects on Average Dynamic Recall (ADR; see Section 5.2) and response time if the effort for identifying false positives is kept constant. ADR and runtime were averaged over the 6 queries for the RISM UK collection from MIREX 2006. The ground truth from MIREX 2006 was used for calculating ADR.

result quality in that range too seriously, especially not the decrease for 8 vantage objects instead of 7. This effect can be caused by the fact that the variations of the search radius are done with discrete steps to keep the number of range queries low. In other words, the algorithm does not always retrieve exactly 20,000 candidates, and if one is unlucky with one particular query, the actual radius can be lower with 8 vantage objects than with 7, leading to some more false negatives with more vantage objects.

The number of 20,000 was chosen by first experimenting with a larger threshold for 8 vantage objects, establishing the best ADR the algorithm can reach with a large threshold for a few sample queries, and then lowering the threshold until the effect of lowering the threshold becomes visible in the ADR score. If one would start out with fewer vantage objects, one would have to work with a higher threshold for the number of considered candidates to get a similarly good result quality. Since the threshold was chosen such that its existence did not degrade the result quality for 8 vantage objects, one should expect that increasing the number of vantage objects would not improve the ADR scores. However, it would make it possible to lower the threshold.

As one might expect, the total response time increases with the number of vantage objects since the index table is searched with range queries that affect more columns. For searching the index table, the relational database system (MySQL) uses a nested B-tree whose nesting depth equals the number of vantage objects. Traversing a larger tree takes more time. Also, calculating the overall scores for database items becomes computationally more expensive if more matching segments from the same item are found, which is more likely if the number of false positives is decreased. However, the runtime information needs to be taken with a grain of salt since cache effects can have influenced it.

4.5 Choosing good vantage objects

In this section, we will present results about selecting good vantage objects based on two criteria, spacing and correlation. This work was coauthored by Reinier van Leuken and Remco C. Veltkamp [87]. We experimented with real-world polyphonic

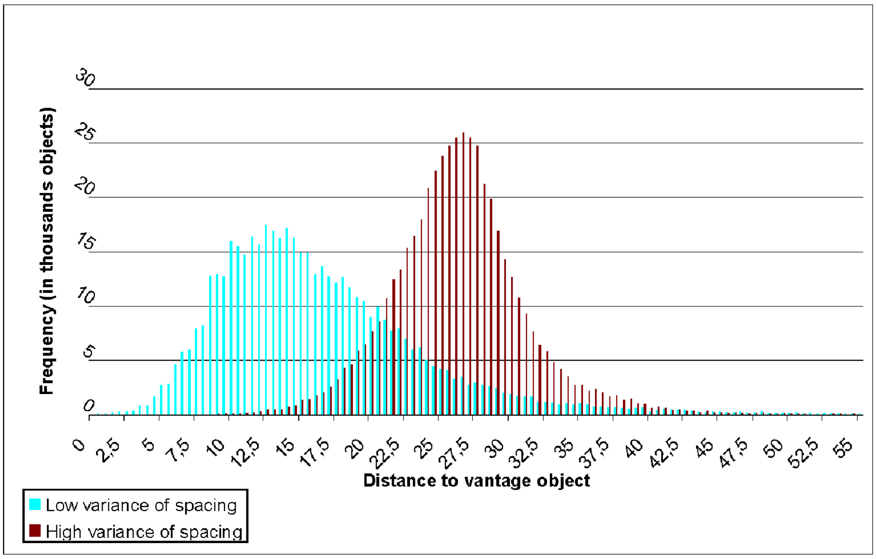


Figure 4.5: Distance distributions for vantage objects with a high and low variance of spacing.

MIDI data from the same collection that was also used for MIREX 2006. Whenever we mention "database items" in this section, we mean monophonic segments of length 5 from this collection.

4.5.1 The spacing criterion

Not all vantage objects are created equal when it comes to the number of false positives they produce. False positives occur whenever two objects happen to have similar distances to a vantage object although they are rather different. As is illustrated in Figure 4.2, an easy way of eliminating false positives is to use multiple vantage objects. However, this solution comes at a cost – it costs space to store more distances to vantage objects, and it costs time to compute those distances. Therefore, it can be useful to carefully choose vantage objects so that one does not have to deal with too many false positives to begin with and therefore can do with fewer vantage objects for reaching a reasonable number of false positives.

One can formalize the discriminative power of a vantage object by using the variance of spacing. If, for one given vantage object, one marks its distances to all items in the database on a vantage axis, one can measure its discriminative power by measuring how evenly spaced the marks on the axis are. The more clusters occur, and the denser those clusters are, the more false positives one has to expect to be caused by this vantage object. This can be measured with the variance of spacing. Figure 4.5 illustrates how the variance of spacing and the distance distributions for vantage objects are connected. The distribution of distances of all database objects to two vantage objects is visualized in histograms. One can observe that the database objects have a wider variety of distances to the vantage object with a low variance of spacing than to the vantage object with a high variance of spacing.

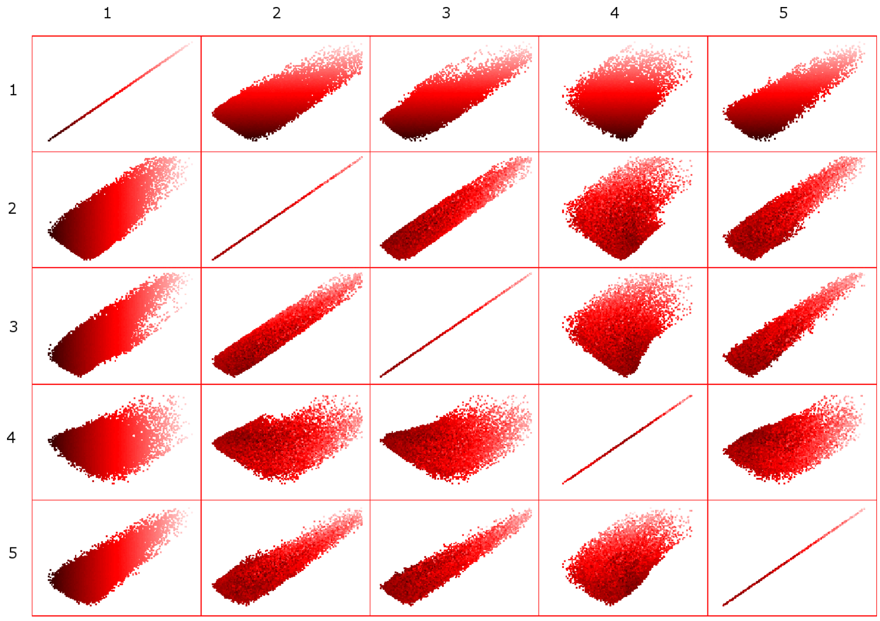


Figure 4.6: A scatterplot matrix for 5 randomly selected vantage objects. There are 25 scatterplots for each combination of two vantage objects. For every combination of two vantage objects, the scatterplot shows database items as dots at coordinates that correspond with the distances to the two vantage objects. The diagonal of the scatterplot matrix shows combinations of vantage objects with themselves, therefore we only see diagonal lines in those scatterplots. The larger the area that is covered by dots in a scatterplot, the better it is to combine the two vantage objects.

4.5.2 The correlation criterion

Even very well chosen vantage objects still can produce false positives. Therefore, one still needs the principle of eliminating them with multiple vantage objects, as is illustrated in Figure 4.2. However, when adding vantage objects, it is not only important that they are good in the sense of producing a low variance in spacing, but also that they do not produce the same false positives as the set of vantage objects to which they are added. The more closely the distances to all database items are correlated for two vantage objects, the less useful it is to use the second vantage object in addition to the first one; if it has similar distances to the database items, it is likely to produce just the same false positives which it is supposed to eliminate.

To measure the usefulness of adding a given new vantage object to a given set of vantage objects, one can compute the Pearsons Correlation coeffcient. Figures 4.6 and 4.7 illustrate the benefits of picking vantage objects with low correlation. These scatter plots indicate how the distances of all items in the database to two vantage objects are correlated. Maximum correlation results in a simple diagonal line, which one can see on the diagonal of the matrices, where each vantage object is compared to itself. For Figure 4.6, five vantage objects were randomly chosen, while the five vantage objects for Figure 4.7 were selected such that the correlation

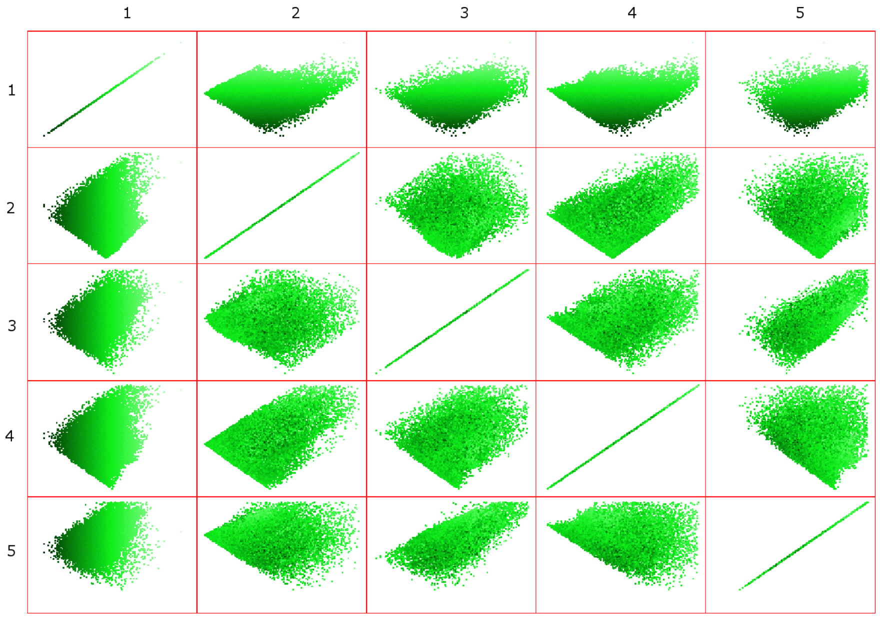


Figure 4.7: Scatterplot matrix for 5 vantage objects with low correlation

coefficient between every possible pair is low.

4.5.3 Some speculation about what makes a vantage object good

The experiments with selecting good vantage objects – good in the sense that spacing variance and inter-vantage object correlation are low – produced some concrete examples of bad and good vantage objects, which inspire the following hypotheses about good vantage objects for weighted point sets and transportation distances:

- If a lot of weight is concentrated in a small percentage of the area covered by a weighted point set, this makes the point set a bad vantage object since it makes it likely that many different point sets will have similar distances to this vantage object. See Figure 4.8 for an example. Let us assume for Figure 4.8 that the two points with weight 50 in the left light point set are each a distance of d away from the point with weight 100. On the right side, the middle point with weight 34 is also d away from the point with weight 100, while the two points with weight 33 are equally far away from the middle point with weight 34. In such a case, the overall distance to the bad vantage object is exactly the same for two considerably different point sets. More points in the vantage object, and more evenly distributed weights, would make such cases much less likely.

- However, a certain healthy dose of asymmetry is needed in a vantage object to make it sufficiently different from other vantage objects. Otherwise, the correlation between vantage objects is too high.

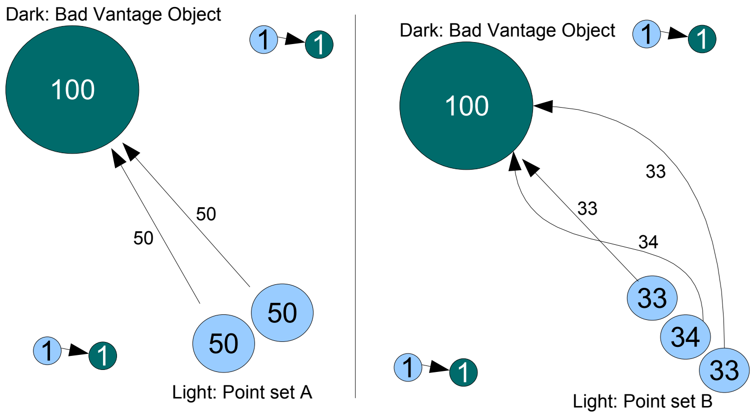


Figure 4.8: An example for how uneven weight distribution in the vantage object can lead to flow components that span a large ground distance. These dominate the transportation distance for point sets A and B, which makes the distance to the vantage object about the same for A and B and thus masks the noticeable differences between these two point sets.

For constructing good sets of vantage objects, it therefore seems promising to aim at well-balanced objects that still have large distances between them. If one views vantage objects as music, this could translate into a number of successive chords of about evenly weighted notes (for creating an even weight distribution), but some of the vantage objects should have narrow chords and others wider chords (for causing large distances between the vantage objects).

4.6 Splitting the segment table

Indexing segments by using vantage objects involves a large segment table which, for each segment, contains the distances to the vantage objects as well as the information to which document a segment belongs and at what position within the document it occurs. For finding segments that are similar to a given segment from a query, we first calculate the distances between the query segment and each vantage object. Then we retrieve all segments with similar distances to the vantage objects from the segment table, which can be done efficiently with a range query that is supported by a B-tree. While the B-tree nicely avoids the need to look at every single record in this table, it does not prevent the table from growing linearly with the number of indexed documents. By splitting this table, we can benefit from the less than linear growth of the part of the table that contains the distances to vantage objects. The distances to vantage objects take up most of the space in each record.

Real-world music databases contain many segments that are almost identical, even if they belong to different pieces. Our algorithm works much faster and uses less space if we store every unique set of distances to vantage objects only once. That is, we split the segment table into one table that contains unique sets of distances to

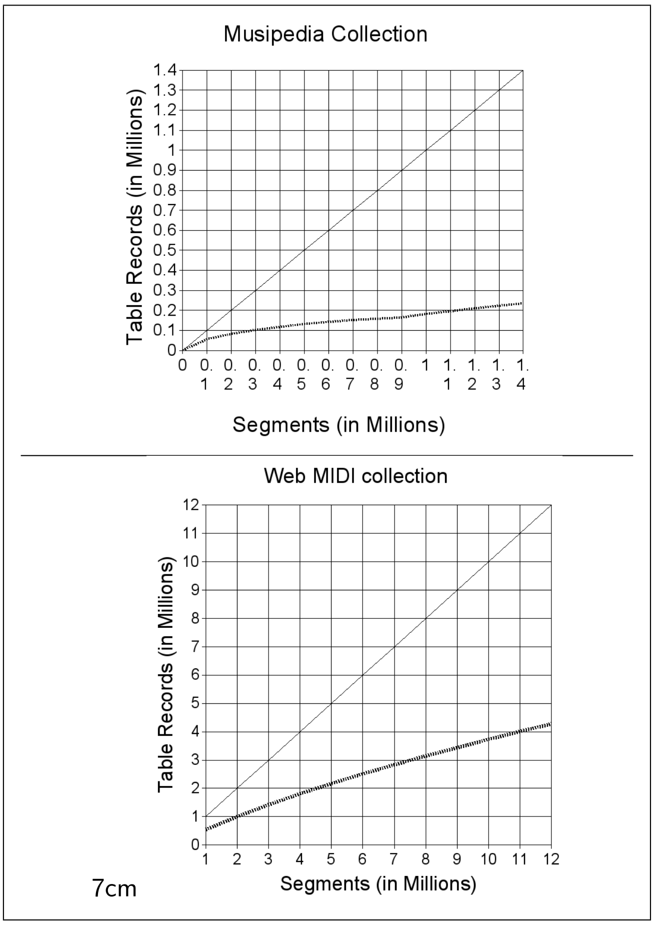


Figure 4.9: The number of characteristic segments (fat dotted line) versus the total number of segments (thin solid line) for the Musipedia collection (top) and a collection of about 50,000 MIDI files collected from the Web (bottom).

vantage objects (which we call "characteristic segments"), and another one that, for every segment, contains the information in which document it occurs and at what position, as well as a reference to its set of distances to vantage objects.

Figure 4.9 shows the number of characteristic segments in comparison to the total number of segments for two different collections. The Musipedia collection (from the collaborative directory of melodies http://www.musipedia.org) is a mostly monophonic collection of about 30,000 musical themes which are rhythmically quantized, while the Web MIDI collection contains about 50,000 mostly polyphonic and not rhythmically quantized pieces. The rhythmic quantization makes it more likely to encounter the same segment in different pieces, which explains why the number of characteristic segments is so much lower for the Musipedia collection even though this collection contains much more varied musical material.

How exactly the number of characteristic segments grows depends on the collection of music scores. For quantized collections such as Musipedia, there are only finitely many possibilities for segments, so the curve will eventually approach a limit. The number of possibilities for segments is limited because there are finitely many pitches and note durations, and the segment length is constant. For example, for 120 possible pitches, 20 different note durations, and a segment length of 5, one cannot get more than about $(120{\cdot}20)^5 \approx 8{\cdot}10^{16}$ different segments. In practice, the number of different segments is lower because most of them are musical nonsense and therefore do not occur in real life. Even for the Web collection, the number of characteristic segments grows more slowly than linear (for example, 1 million segments can be described with 0.54 million characteristic segments, while for 3 million segments, the number of characteristic segments is less than half, namely 1.41 million, and for 12.2 million, it is closer to a third: 4.3 million).

One can control the growth of the number of characteristic segments by relaxing the uniqueness criterion and treating segments whose distances to the vantage objects do not differ by more than a threshold ϵ as the same. In order to not create false negatives as a result, one would then need to increase the search radius by ϵ, and one would have to take into consideration that the reported distance can be distorted by the fact that in each distance to a vantage object, there can be an error up to ϵ. In other words, one can trade a loss of accuracy (of a known magnitude) for additional saved space and the introduction of a limit to the number of segments that are treated as different. These space savings would be in addition to those shown in Figure 4.9 – those characteristic segments are actually unique (ϵ is zero).

4.7 Conclusions

We have shown how vantage indexing, which was originally suggested for image retrieval, can be adapted for Music Information Retrieval. For our approach of searching for melody segments, it is suitable to neither do a pure nearest-neighbour search nor search a fixed search radius, but combine elements of both. We have shown that for music, splitting the vantage table into one table with the distances to vantage objects for "characteristic segments" and another one with the rest of the segment information can noticeably decrease the storage space requirements because there are many very similar segments. This effect is still there, although less pronounced, for non-quantized music. The asymptotic time complexity for searching is not affected by the reduction of space for the index because the table split makes a join operation necessary whose asymptotic time complexity is the same as that needed for searching the table if it is not split. However, in practice, the amount of data one can search in a few seconds is largely determined by the size of main memory. So, if the table split cuts the amount of needed memory in half, this does not make the search twice as fast, but it means that we can search twice as much data in a reasonable amount of time.

Chapter 5

Evaluating MIR systems by using a ground truth generated by humans

For evaluating the performance of a music retrieval system, one needs a ground truth for its data collection and some given queries. In other words, for the given queries, it should be known what the ideal search result is. The music retrieval systems we have in mind serve the information need for music that is melodically similar to a given query.

In this chapter, we will describe how we created a ground truth for the RISM A/II collection of about half a million musical incipits. This involved a filtering step for finding as many good candidates for matches as possible for a selected set of queries, the collection of expert opinions about which candidates are similar to the queries (and what the right ranking would be), and the consolidation of many expert opinions into one ground truth.

Since our ground truth contains a reliable ordering of groups of items, not necessarily individual items, we introduce the new measure "Average Dynamic Recall" (ADR) that can handle such a ground truth. Both our ground truth and the measure were used for the MIREX 2005 competition of melodic similarity algorithms. ADR was also used for SHREC 2006[1].

At the end of this chapter, we report the MIREX 2005 and 2006 results for "symbolic melodic similarity".

5.1 A ground truth for the RISM A/II collection

The RISM A/II collection [1] contains 476,600 incipits (in the 2002 edition), short excerpts of notated music from the beginnings of manuscripts in libraries, archives, cloisters, schools, and private collections worldwide. This collection is useful for content-based music retrieval because of its size and the fact that it contains real music written by human composers. A music retrieval system that does not work

[1] SHREC stands for "3D Shape Retrieval Contest"; see http://www.aimatshape.net/event/SHREC.

well with this collection probably also does not perform well for real-world applications in general. Our ground truth can serve as a benchmark for deciding how well a music retrieval system works with the RISM A/II collection.

In TREC [91], relevance assessments are mostly binary ("relevant" or "not relevant"). Only in more recent TREC web tracks such as at TREC-9 [23], this was extended to ternary ("irrelevant"/"relevant"/"highly relevant").

Because of the continuity of melodic similarity (see Section 1.4), there are no sensible criteria for assigning one out of a few distinct degrees of relevance to a melody, so any relevance assessment with a given scale length seems inappropriate. Instead, we asked human experts to rank all incipits where they saw any similarity to the query. Our ground truth therefore does not consist of sets of highly relevant, relevant and irrelevant documents, but of ranking lists of documents.

A valid way of establishing such a ground truth would be to ask a number of human experts to look at all possible matches for a given query (carefully making sure that they stay concentrated long enough) and order them by similarity. Since we cannot expect our human experts to sift through half a million melodies, we needed to filter out incipits of which we can be reasonably sure that they do not resemble the query.

5.1.1 Filtering Melodies

To be able to exclude incipits that are very different from our selected queries, we calculated some features for every incipit in the database. Filtering could then easily be done by issuing SQL statements with selections based on those features.

- **Pitch range:** the interval between the highest and lowest note in the incipit.

- **Duration ratio:** the duration of the shortest note (not rest), divided by the duration of the longest note (not rest). The result is a number in the interval (0,1], where 1 means that all notes have the same duration, while a very small number means a very high contrast in durations.

- **Maximum interval:** the largest interval between subsequent notes. Rests are ignored.

- **Editing distance between gross contours:** the editing distance between two character strings is the sum of the costs of the cheapest possible combination of character insertion, deletion, and replacement operations that transform one string into the other. We determined the gross contour as a string of characters from the alphabet U ("up"), D ("down"), and R("repeat") and calculated the distance to every query for each incipit in the database, using the editing distance described by Prechelt and Typke in [58]. They had optimized the costs for the insertion, deletion, and replacement operations for gross contour strings such that the resulting similarity measure corresponds well with human perception.

- **Editing distance between rhythm strings:** we also represented the incipits as rhythm strings with one character from a three-character alphabet for each pair of subsequent notes: longer, shorter, and same duration.

- **Interval histogram:** the number of occurrences for each interval between subsequent notes, normalized with the total number of intervals. With this feature, we can base selections on things like "incipits with many thirds".

- **Interval strings:** one string of diatonic intervals and one string of chromatic intervals for every incipit. This makes it possible to select incipits that contain a certain sequence of intervals.

- **Motive repetitions:** in order to be able to select things like "all incipits with at least three repeated notes in two different places", we collected sequences of intervals that were repeated at least once, along with their number of occurrences, for every incipit. The repetition detection algorithm maximizes the motive length, even if this means fewer repetitions.

We used different filtering steps and features for every query since every query has its own characteristic features. Every filtering step had the aim of reducing the number of candidates for matches for a given query by excluding incipits with features that make them very different from the query. As long as this holds for every filtering step, different people should arrive at similar candidate lists even if they apply different filtering steps. However, they need to have similar notions of melodic dissimilarity (also similar to those of the human experts whose input determines the actual ground truth).

For example, we used the following filtering steps for the "White Cockade" incipit whose ground truth is shown in Table 5.3:

- Exclude incipits whose pitch range is less than an octave or greater than a minor tenth. This excluded 78 % of the incipits in the database.

- Exclude incipits whose maximum interval between subsequent notes is less than a minor sixth or greater than a diminished seventh.[2] This excluded 79 % of the remaining incipits.

- Exclude incipits with a duration ratio greater than 0.51, i. e. incipits where all notes have quite similar durations. This excluded a further 4 % of incipits.

- Exclude incipits that do not contain at least one of the two interval sequences "fifth up, third down, unison, sixth up" or "third up, unison, unison, sixth up".[3] This left us with 88 incipits.

Because of the dangers of filtering too strictly and thereby accidentally excluding incipits that are similar to the query, we stopped the filtering process once the number of remaining incipits had fallen below 300. To arrive at the desired number of about 50 candidates,[4] we manually excluded those remaining incipits that we perceived as most different from the query.

As an additional measure to limit the error introduced by accidentally filtering out similar incipits, we used our prototype of a search engine based on transportation distances (see Chapter 3; we used the algorithm without segmenting) as well as two algorithms from [41] for finding incipits that are similar to the query. The latter two algorithms, called P2 and P3 by their authors (for an explanation of P2 and P3,

[2] A "minor sixth" is the smaller of two musical intervals that span six diatonic scale degrees. It is an interval of 8 semitones. A "diminished seventh" is an interval of 9 semitones.

[3] A fifth is an interval of 7 semitones. This means that the frequency ratio of the two involved notes is rather simple, namely 2/3, which is perceived as very consonant. "Unison" means an interval of size zero, and a third and sixth span three or six diatonic scale degrees, respectively.

[4] Depending on the incipit, we usually viewed a number of 50 candidates as desirable since this number allows us to include some not very similar items while still being manageable for the human experts.

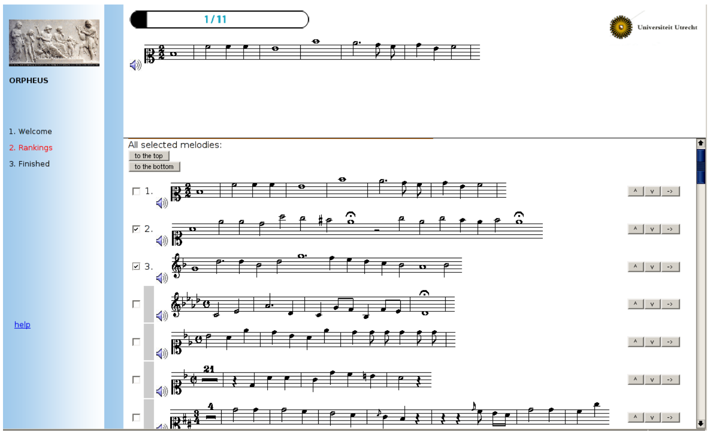


Figure 5.1: The user interface for the experiment. MIDI files are provided for listening to incipits. In the bottom half of the screen, the subjects can change the order of the candidate incipits, while the query always remains visible at the top.

see Section 3.5.1), find incipits containing transpositions of the query where many onset time/pitch combinations match, and incipits containing transpositions of the query with maximum common duration with matching pitch. From these search results, we included candidates that we considered similar although they had been filtered out. Also, we used the metadata in the RISM A/II collection. For example, for "Roslin Castle" (see Table 5.1), we made sure that every incipit whose title contains the word "Roslin" was included. With these methods, we found between 0 and about 8 additional candidates for each query, with an average of about 4. In a comparison of algorithms based on the ground truth, one needs to avoid favouring the algorithms that were used for finding additional candidates against other algorithms. If other search algorithms find more sensible matches for a query that were incorrectly excluded in the filtering steps, they need to be included in the ground truth, ideally by rebuilding the ground truth with a panel of human experts that is shown the more complete list of candidates for matches.

Once we had filtered out the vast majority of incipits that are not similar to the query, we also removed incipits that were either identical to other incipits or to parts of other incipits. Including identical incipits multiple times in the candidate list would have amounted to asking our experts the same question multiple times, and we wanted to put their time to a more productive use. As a result, only 6 versions of "Roslin Castle" occur in our ground truth in Table 5.1 although we list 16 known occurrences of this melody in our Section 3.3, for which we used the same 2002 version of the RISM database.

5.1.2 Experiment Design

Notated music, MIDI files

Our goal was to establish a ground truth for the incipits that are contained in the RISM A/II collection. These incipits can be exported from the database in the "Plaine & Easie" format [27] and then rendered in common music notation. In order to prevent differences in the rendition of the notated music from having an

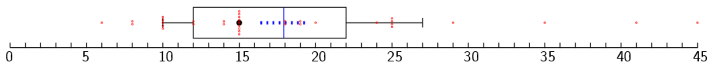


Figure 5.2: The experience of our experts (instrument playing or studying music), in years. The box extends from the first to the third quartile. The whiskers mark the bottom and top 10 percent. Every data point is shown as a little dot. The median is marked with a fat dot, the mean is shown as a vertical line. The dashed horizontal line around the mean marks one standard deviation below and above the mean.

impact on the ground truth, we used the software that is included with the RISM A/II database [1] for rendering the music notation bitmaps and took screen shots of the results. Only in cases where the RISM software fails to show the whole incipit because it is too long for fitting on the screen, we rendered the notated music ourselves by converting the Plaine & Easie data into the Lilypond [51] format. In addition to the notated music, we also provided MIDI files generated from the Plaine & Easie data as an illustration of the incipits.

The metadata from the RISM A/II collection (composer, work title, title of the movement, instrumentation etc.) was not shown to the human experts. They only saw the notated music of the incipits and could listen to a MIDI rendition, as can be seen in Figure 5.1.

Experts

Müllensiefen et al. point out [46] that music experts tend to have stable similarity judgements, in other words, do not change their mind on what is melodically similar when asked to perform the same judgements a few weeks apart. Subjects with stable similarity judgements, in turn, seem to have the same notion of melodic similarity (however, there also were some music experts with unstable notions of melodic similarity). In order to establish a meaningful ground truth, we therefore tried to recruit music experts as our experimental subjects. We asked people who either have completed a degree in a music-related field such as musicology or performance, who were still studying music theory, or who attended the International Conference on Music Information Retrieval Graduate School in Barcelona 2004 to participate in our experiment. For organizational and budgetary reasons, we did not test the stability of their notions of melodic similarity. Instead, we try to ignore outliers with statistical methods as described below.

All of our experts play at least one instrument or sing, most play several instruments. See Figure 5.2 for a box-and-whisker plot showing their musical experience in years.

Instructions, tasks

We asked the subjects to rank all candidates that resemble the query by their melodic similarity to the query. Candidates that seemed completely different from the query could be left unranked. The ranking was to be done by reordering the given candidates such that the candidate most similar to the query was at the top, followed by less and less similar candidates, and finally a number of candidates without any assigned ranks that did not resemble the query at all. By asking people to reorder a list instead of picking a rank from a scale, we avoided suggesting how long the ranked list should be, and we also made it easy for the experts to judge whether they ranked all candidates correctly by looking at a local ordering only. It was sufficient to ensure that for each pair of consecutive candidates in the ranked part of their reordered list, the incipit that was ranked higher was more similar to the query than the other incipit of the pair.

We asked the experts to regard transpositions of a melody as identical, as well as melodies that are notated slightly differently, but in a way that does not affect the way they sound. For example, melodies that are notated with different clefs, but are otherwise the same, should not be viewed as different. In cases where two incipits were taken from similar pieces, but covered different amounts of musical material, we asked the subjects to only consider the common parts of the two incipits for the comparison. Since the conversion of RISM incipits to MIDI removed some information, we told the experts that the definitive source for similarity judgements is the notated music, and that the MIDI files only serve as an illustration.

We asked every subject for about 2 hours of his time and presented up to 11 queries. We asked the experts to work carefully, even if that meant that they could not finish all 11 queries within two hours. After collecting 30 expert opinions for a query, we stopped showing it to other experts. For the next expert, we picked 11 queries from the set of queries for which we still had fewer than 30 expert opinions. Overall, 37 experts worked on these 11 queries. For MIREX 2005, another 11 queries were selected and presented to another 30 experts. The ground truth for the first 11 queries was used as training data, and the other 11 queries as test data for the 2005 "Symbolic Melodic Similarity" task.

Threats to the validity of results

- **Filtering errors.** It is possible that we filtered out some incipits although they are similar to the query. Our ground truth, therefore, could be incomplete. However, this does not threaten the validity of the ranking of those candidates that we did include.

- **Sequence effects.** The initial order of candidates as well as the order in which queries are presented to the experts could have an impact on the results. Experts could be tempted to leave the order similar to the initial order, and they get more tired and at the same time more skilled at using our interface over the course of the experiment. We addressed these problems by randomizing the order of queries for every participant, and we also put the candidates in a new random order whenever a new query appeared on the screen.

- **Carelessness of experts.** For some queries, such as the "White Cockade" shown in Table 5.3, we included the query itself among the candidates. Careful experts should put it at the very top of the ranked list. Not everybody did, but enough of the experts were careful. This query was recognized as most similar to itself with high statistical significance: the Wilcoxon rank sum test, which we used as described in Section 5.1.3, shows that for every candidate that was not identical to the query, the probability of the null hypothesis is ≤ 0.0001123.

- **User interface limitations.** Most of the time, we presented more candidates than what would fit onto one screen. Creating a good ranked list demanded some discipline and patience from the experts, who first had to move all similar items close to the top of the list and then compare them with one another to create the final ranking.

5.1.3 Results

Evaluation methodology

For every query, the subjects were asked to choose and rank as many of the candidates for matches as they thought had some similarity to the query. Those candidates without any similarity could be left unranked. This gives us a list of ranks for every candidate. These lists tend to be longer for the candidates that are more similar to the query.

To obtain a ground truth, we ordered the candidates by their median rank and then by their mean rank. In addition, for every ranked candidate, we applied the Wilcoxon rank sum test to the ranked candidate and every incipit that was ranked higher. The Wilcoxon rank sum test, given two samples, determines the probability of the null hypothesis (p-value), that is, the hypothesis that the median values are the same for the whole two populations from which the samples were taken (here, the population would be the group of all Western music experts). We used it to find out how likely it is that the differences in ranking observed by us are only caused by our choice of 37 people out of the whole population of music experts. A low p-value resulting from the Wilcoxon test means that the difference in medians is probably not a coincidence. A large p-value does not necessarily mean that the medians are the same, but just that we do not have compelling evidence for them being different.

The resulting ground truth tables

We visualize the ranks assigned to each candidate with a box-and-whisker plot. The box extends from the first to the third quartile. The whiskers mark the bottom and top 10 percent. Every data point is shown as a little dot. The median is marked with a fat dot, the mean is shown as a vertical line. The dashed horizontal line around the mean marks one standard deviation below and above the mean. The numbers on the scales reflect ranks.

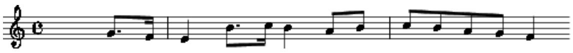
Below every box-and-whisker plot except for the first one, we visualize the Wilcoxon rank sum test results with a horizontal bar that is composed of one square for every incipit which is ranked higher than the current one. Each of these squares has a dark upper area and a lower area with a lighter colour. The size of the dark upper area reflects the p-value.

For incipits where every square in the Wilcoxon visualization is almost entirely light-coloured, we can be reasonably sure that all preceding incipits should indeed be ranked higher. Wherever this is the case, we draw a horizontal line immediately above the incipit. For Table 5.1, we set the threshold for the maximum probability for the null hypothesis at 0.25. In other words, we draw a horizontal line above every incipit where the p-value is less than 0.25 for every single incipit that appears higher in the list. Most actual probabilities are much lower than that, as the visualization of the Wilcoxon tests in Table 5.1 shows.

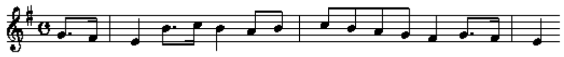
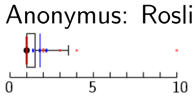
For "Roslin Castle" (Table 5.1), we find five clearly distinguishable groups that way. The incipit with median rank 1 is generally considered the most similar incipit to the query. For the incipit with median rank 2, the Wilcoxon test shows that the probability for the null hypothesis is p=0.00006722. Therefore, we consider the difference in median values statistically significant and separate the second incipit from the first with a horizontal line. For the incipit with median rank 3, the difference in medians is statistically significant for the comparison with the first incipit (p=0.0002765), but not for the comparison with the second incipit (p=0.6363). This is reflected in the Wilcoxon visualization bar, which consists of one almost entirely light-coloured square on the left for the comparison of the third incipit with the first one, and one mostly dark square on the right for the comparison of the third incipit with the second one. Since there is no statistically significant difference between the second and third incipit, we group them together and do not separate them with a horizontal line. The third group consists of the incipit with median rank 4. The highest of its three p-values resulting from the Wilcoxon tests for its three predecessors is 0.07633. The fourth group again consists of one single incipit, while for all other incipits, there are no statistically significant differences in median ranks. Either we did not have enough subjects who ranked these incipits, or people simply do not consider the dissimilarities between the remaining incipits and the query significantly different.

The tables shown in this book are not complete. We cut them off a bit after the last detected border between clearly distinguishable groups because the ranking becomes less reliable and therefore less interesting towards the bottom of the tables. The complete data are available online at `http://rainer.typke.org/mirex05.0.html`.

Table 5.1: Ground truth for "Roslin Castle". Table contents: median rank, incipit with title and RISM A/II signature, box-and-whisker plot showing the ranks assigned by our subjects, and a bar composed of squares visualizing the Wilcoxon rank sum test results for every preceding incipit. For details see Section 5.1.3.

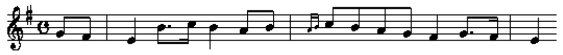
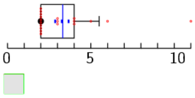
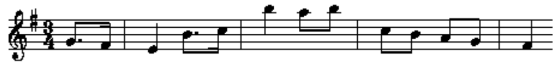
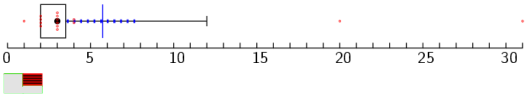
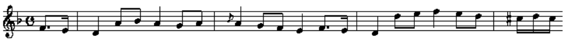
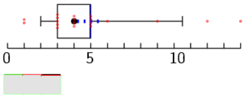
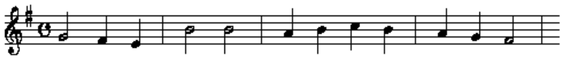
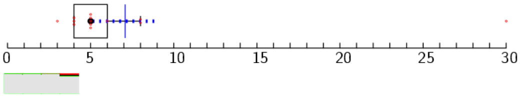
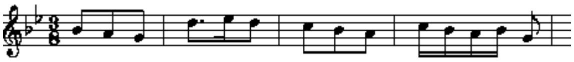
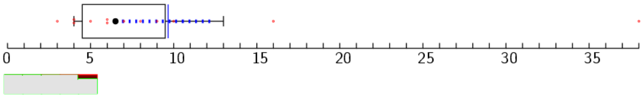
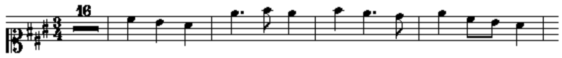
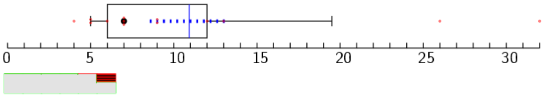


Query: Anonymus: Roslin Castle, RISM A/II signature: 800.000.193

Median Rank	Candidate Incipit, Composer, Title, RISM A/II signature, Ranks, Wilcoxon Test Results (p-values: dark upper area)
1	 Anonymus: Roslin Castle. 000.109.446 

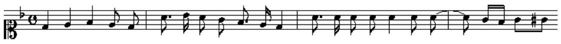
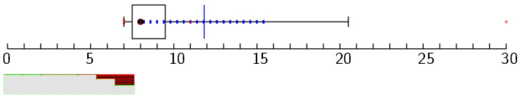
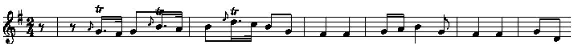
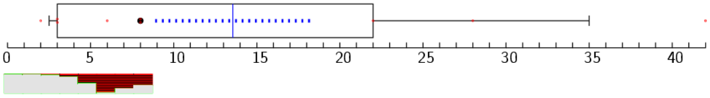
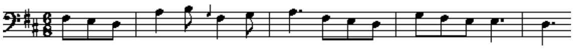
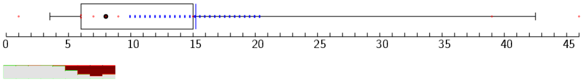
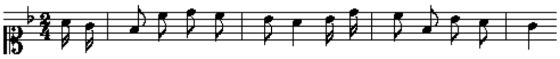
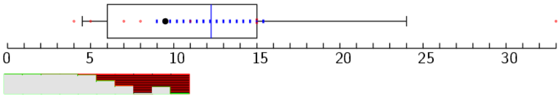
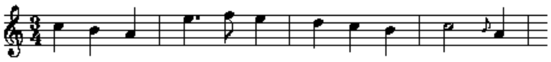
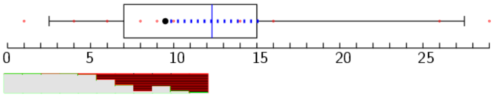
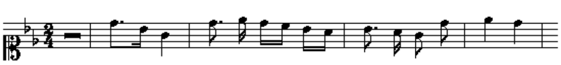
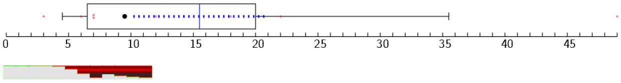
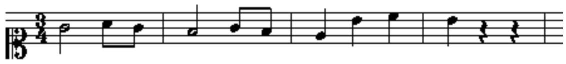
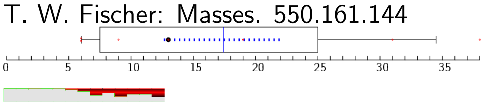
Continued on next page

– continued from previous page

Median Rank	Candidate Incipit, Composer, Title, RISM A/II signature, Ranks, Wilcoxon Test Results (p-values: dark upper area)
2	 Anonymus: Roslin Castle. 000.111.779 
3	 Anonymus: Roslin Castle. 000.112.692 
4	 Anonymus: Roslin Castle. 000.132.330 
5	 Anonymus: Roslin Castle. 000.112.625 
6.5	 Anonymus: Allegro. 704.000.704 
7	 Anonymus: Care Jesu. 400.196.546 

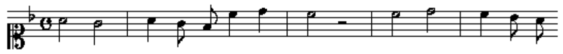
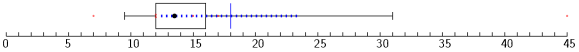
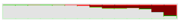
Continued on next page

– continued from previous page

Median Rank	Candidate Incipit, Composer, Title, RISM A/II signature, Ranks, Wilcoxon Test Results (p-values: dark upper area)
8	 Robert Führer: Vesperae. 220.000.909 
8	 Florian L. Gaßmann: Trios. 702.001.807 
8	 Anonymus: De France 700.008.178 
9.5	 Anon.: Vernünftige Lust. 000.114.257 
9.5	 Anon.: Pour me venger. 000.109.156 
9.5	 G. Molinari: Mecum enim 850.503.217 
13	 T. W. Fischer: Masses. 550.161.144 

Continued on next page

– continued from previous page

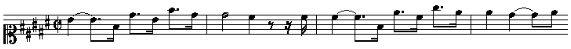
Median Rank	**Candidate Incipit,** **Composer, Title, RISM A/II signature,** **Ranks,** **Wilcoxon Test Results (p-values: dark upper area)**
13.5	 G. J. Werner: Puer natus. 530.004.292  

5.1.4 Musical properties of the identified groups

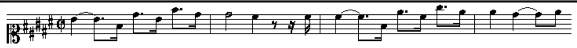
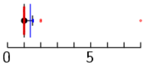
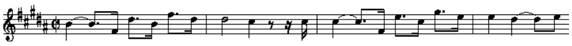
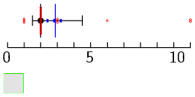
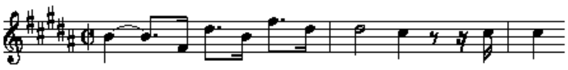
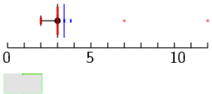
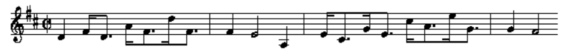
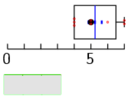
In Table 5.1 ("Roslin Castle"), the candidate with the highest rank looks as if it would begin with the query and therefore should, according to our instructions, be regarded as identical to the query since only the common part should be considered. If one looks more closely, however, one notices that the key signatures are different. The resulting differences in two notes, however, are not big enough for our experts to consider it very different from the query. The incipits with median ranks 2 and 3 constitute the second group. Both begin differently from the query - the incipit with median rank 2 has slight differences in rhythm at the beginning and two grace notes added in the second measure, while the incipit with median rank 3 has its second measure transposed by an octave. Otherwise their beginnings are the same as the query. Our experts agree that these incipits are both less similar than the incipit with median rank 1, but they disagree on whether the transposition of a measure by an octave or the modified rhythm and added grace notes should be seen as a greater dissimilarity. Because of this, these two incipits are combined into one group. The experts agree that the incipit with median rank 4 is significantly different from those preceding it. This is justified by a minor difference in rhythm in measure 1 and a major difference in measure two – the first note is just a grace note, so there is no group of four descending eighth notes in that measure as in all preceding incipits. The incipit with median rank 5 is again significantly different. The rhythm is changed in several ways, leading to a very noticeable difference in measure 3. The third note in this measure corresponds to the first note in measure 2 of all preceding incipits. Because here this note is not at the beginning of the measure, it is much less emphasized, which changes the character of the melody. The last statistically significant border between groups is that between the incipits with median ranks 5 and 6.5. The latter is the first incipit of a different piece, and it also has a different time signature, so we would expect a border between groups here. Another border could be expected between the second and third incipit with median rank 9.5 because the interval sequence at the beginning changes noticeably here. However, at this point in the ranked list, we do not have enough votes per incipit for finding a statistically significant difference.

Table 5.2 shows that the top three candidates for J. A. Hasse's "Artemisia" are very similar. The incipit with median rank 1 is identical to the query, that with median rank 2 is written with a different clef, but otherwise identical to the query, and the incipit with median rank 3 is identical to the first half of the query. Although they were instructed to disregard such differences, our experts still agreed that simply notating the query differently changes it less than omitting the second half, leading to statistically significant differences in the rankings. The incipit with median rank 5 is a somewhat similar melody by the same composer, but from a different work ("Tito Vespasiano"). It is similar to the query because it also begins with a motive built from notes from a triad (the tonic) and with a dotted rhythm, followed by a variation of the same motive that is based on another triad, the dominant. However, the rhythm is inverted in "Tito Vespasiano". All other candidates are ranked lower, but without further statistically significant differences in rank. The next candidates also begin with a triad that is split up in a similar way, sometimes also with a dotted rhythm, but not followed by a similar motive based on the dominant.

Table 5.2: Ground truth for an Aria by Johann Adolf Hasse as query. For details see Section 5.1.4.

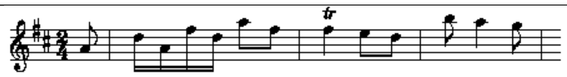
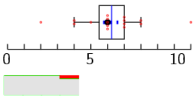
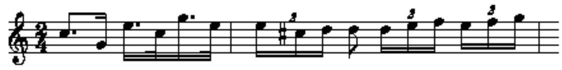
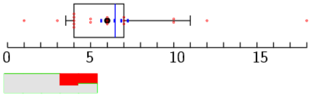


Query: J. A. Hasse: Artemisia, Aria no. 16, Andantino/Allegretto, RISM A/II signature: 270.000.749

Median Rank	**Candidate Incipit,** **Composer, Title, RISM A/II signature,** **Ranks,** **Wilcoxon Test Results (p-values: dark upper area)**
1	 J. A. Hasse: Artemisia. 270.000.749 
2	 J. A. Hasse: Artemisia. 270.000.746 
3	 J. A. Hasse: Artemisia. 270.000.748 
5	 J. A. Hasse: Tito Vespasiano. 270.000.530 

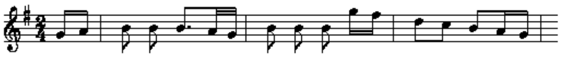
Continued on next page

– continued from previous page

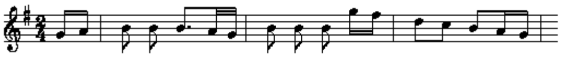
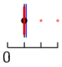
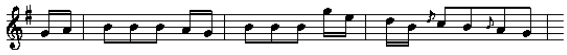
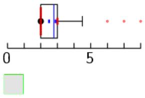
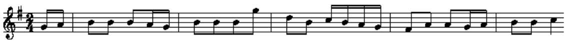
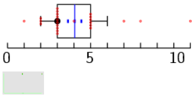
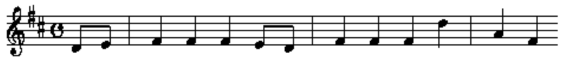
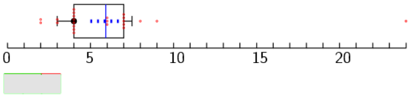
Median Rank	Candidate Incipit, Composer, Title, RISM A/II signature, Ranks, Wilcoxon Test Results (p-values: dark upper area)
6	 A. C. Rezel: Ihr die ihr mit vergnügtem Blick. 240.003.707 
6	 J. Touchemoulin: Sonatas. 706.000.461 

The ground truth for “The White Cockade” by J. F. Latour (Table 5.3) shows that our experts correctly recognized that the incipit that is most similar to the query is the query itself. The incipit with median rank 2 has some minor differences in rhythm, some added grace notes, and two different eighth notes instead of one quarter note in the last measure. Surprisingly enough, the incipit with median rank 3, about which we could say pretty much the same as about that with median rank 2, is ranked lower, and this difference is statistically significant. The remaining incipits of “The White Cockade” or a German version of the same song, “Der verurteilte Hochlandsmann”, are all ranked lower, but without statistically significant differences in their median ranks. In that group, there is one incipit from a different piece (Anonymus: “Cotillons”) that is melodically very similar. There is again a noticeable border between the incipit with median rank 7 and that with median rank 10. The latter is a sonata by Friedrich II, where only the first five notes are similar.

Table 5.3: Ground truth for "The White Cockade" by J. F. Latour as query. Only one out of the top nine pieces, "Cotillons", is not the same piece as the query. As one should expect, the Wilcoxon rank sum test results warrant a separator between the first nine incipits and the tenth, which is from a different piece and at the same time clearly different from the preceding incipits. For details see Section 5.1.4.

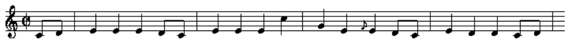
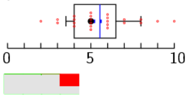
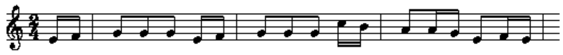
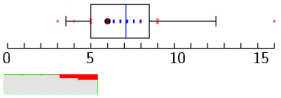
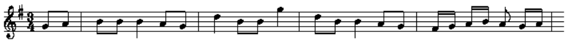
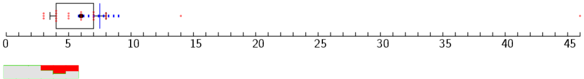
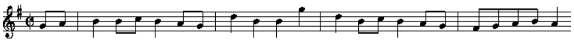
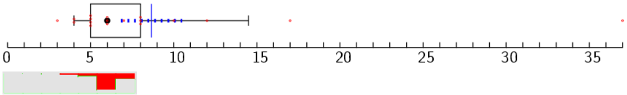
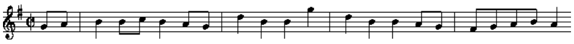
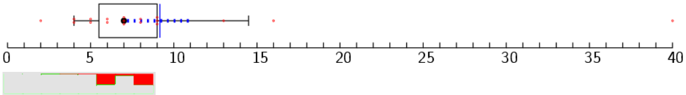


Query: J. F. Latour: The White Cockade, RISM A/II signature: 000.111.706

Median Rank	**Candidate Incipit,** **Composer, Title, RISM A/II signature,** **Ranks,** **Wilcoxon Test Results (p-values: dark upper area)**
1	 J. F. Latour: The White Cockade. 000.111.706 
2	 Anon.: White cockade. 000.113.506 
3	 J. F. Latour: The White C. 000.116.073 
4	 E. Hille: Der verurteilte Hochlandsmann. 451.503.814 

Continued on next page

– continued from previous page

Median Rank	Candidate Incipit, Composer, Title, RISM A/II signature, Ranks, Wilcoxon Test Results (p-values: dark upper area)
5	 J. F. Latour: The White C. 000.113.932 
6	 Anonymus: Cotillons. 190.018.612 
6	 Anon.: White Cockade. 000.135.676 
6	 Anon.: White Cockade. 000.127.493 
7	 Anon.: White Cockade. 000.132.448 

5.2 A measure for comparing search results: Average Dynamic Recall

Our ground truth that is described in Section 5.1 was used at the “1st Annual Music Information Retrieval Evaluation eXchange” (MIREX) 2005 for comparing various methods for measuring melodic similarity for notated music. In order to compare different algorithms, a measure was necessary that compares every algorithm's performance with the ground truth. The measure that was used for ranking the algorithms is described in this section.

Our ground truth does not give one single correct order of matches for every query. One reason is that limited numbers of experts do not allow statistically significant differences in ranks for every single item. Also, for some alternative ways of altering a melody, human experts simply do not agree on which one changes the melody more, so even increasing the number of experts might not always avoid situations where the ground truth contains only *groups* of matches whose correct order is reliably known, while the correct order of matches within the groups is not known.

Kekäläinen and Järvelin suggested graded relevance assessment measures based on cumulated gain [32], [30], which are related to traditional measures such as expected search length [15], average search length [43], and normalized recall [65], [68].

We propose a measure (called "average dynamic recall") that measures, at any point in the result list, the recall among the documents that the user should have seen so far. Unlike Kekäläinen's and Järvelin's measures [30], this measure only requires a partially ordered result list as ground truth, but no similarity scores, and it works without a binary relevance scale.

5.2.1 Some existing measures

For situations where for a given query, there are just two kinds of items, relevant and irrelevant, a whole range of measures for search results has been proposed.

- Two of the most important measures are **precision and recall.** Precision is the percentage of relevant items among a certain number of retrieved items, while recall is the percentage of relevant items that have been retrieved. For example, if 15 relevant items exist, and the top ten retrieved items include 8 relevant items, the precision is 0.8 (8 out of 10 considered items are relevant), while the recall is 0.5333 (8 out of 15 relevant items have been retrieved).

- **Average Precision:** At every relevant document in the result list, the precision is measured. The average precision is the mean of the precisions at the positions of relevant documents in the result list.

Kekäläinen and Järvelin's cumulated gain measures [30] can be used if graded relevance assessments are available. The measures estimate the cumulative relevance gain the user receives by examining the retrieval result up to a given rank. The authors present three measures: the first one simply adds the relevance scores up to the given rank. The second measure does the same, but it also applies a discount factor which depends on the position in the result list. This is meant to take into account the fact that relevant items are more useful if they are close to the beginning of the result list. The third measure computes the relative-to-the-ideal performance.

5.2.2 Motivation for introducing a new measure

Because of the restrictions of binary scales, and also because the ground truth we used is not based on a finite relevance scale and does not contain relevance scores for the documents, we are proposing a new measure for our comparison. We try to meet the following criteria:

1. To make comparisons easy, the measure should deliver one number, for example in the range from 0 to 1, where 0 denotes a completely useless result and 1 a result that completely agrees with the ground truth.

2. In the ground truth, we know only the correct order of groups of matches, not necessarily of every single match. The measure should be able to use the existing information without requiring the ground truth to be completely ordered.

3. There are no relevance scores known for the documents in the ground truth, which only consists of a partially ordered list. The measure should therefore not depend on relevance scores.

4. The measure should not have any parameters one could use to dramatically alter the results (such as a freely chosen discount function for the purpose of rewarding returning highly relevant matches early, arbitrarily chosen thresholds, and the like).

5. The measure should reward putting matches in the right order, as far as that order is known. Therefore, differences in the order within groups should not influence the result, but differences in the order across group boundaries should.

6. In a similar fashion, violations of the correct order should be punished if they happen across group boundaries.

7. False positives in the result should lead to a lower measure, even if the order of the true positives is correct.

8. Both true and false positives that occur close to the beginning of the result list should have a higher influence on the measure than those occurring closer to the end of the list.

9. Since the group sizes do not mean much (they are influenced, for example, by the threshold for statistical significance that was chosen when the groups were established [74]), they should not have a high influence on the measure.

5.2.3 Definition

Our measure is the average recall over the first n documents, where n is the number of items in the ground truth, and the recall is calculated over a dynamic set of relevant documents. Because of this, we call it "average dynamic recall" (ADR). At the beginning of the result list, only the most similar document is counted as relevant (or all documents of which it is not known that they are less similar than the most similar one). The set of relevant documents grows with the position in the result list. Since there are groups of documents in the ground truth where no differences in relevance are known, the dynamic set of relevant documents does not always grow just by one single new relevant document. Rather, at each group boundary it grows by all elements of the next group, and it does not grow between group boundaries. However, at each position in the result list, we still divide the number of found relevant items at that position by the position number, not by the number of all items that would count as relevant.

More formally, consider a result list

$$\langle R_1, R_2, \ldots \rangle$$

and a ground truth of g groups of items

$$\langle (G^1_1, G^1_2, \dots, G^1_{m_1}), (G^2_1, \dots, G^2_{m_2}), \dots, (G^g_1, \dots, G^g_{m_g}) \rangle$$

(with m_i denoting the number of members of group i) where we know that $rank(G^i_j) < rank(G^k_l)$ if and only if $i < k$, but we do not know whether $rank(G^i_j) < rank(G^i_p)$ for any i (unless $j = p$)[5]. We propose to calculate the result quality as follows. Let $n = \sum_{i=1}^{g} m_i$ be the number of matches in the ground truth and c the number of the group that contains the ith item in the ground truth ($\sum_{v=1}^{c} m_v \geq i \wedge \sum_{v=1}^{c-1} m_v < i$). Then we can define r_i, the recall after the item R_i, as:

$$r_i = \frac{\#\{R_w | w \leq i \wedge \exists j, k : j \leq c \wedge R_w = G^j_k\}}{i}.$$

The result quality q is then defined as:

$$q = \frac{1}{n} \sum_{i=1}^{n} r_i.$$

As an example, consider $\langle (1, 2), (3, 4, 5) \rangle$ as ground truth and the result list $\langle 2, 3, 1, 5, 7, 8, 9, 4 \rangle$. That is, while we do not know whether item 1 or item 2 should be at the top of the list, we know that both should be ranked higher than any of the items 3, 4, and 5. In this case, the result quality q is calculated as follows:

Pos.	encountered	relevant	#found	recall
1	2	1, 2	1	1
2	2, 3	1, 2	1	0.5
3	2, 3, 1	1, 2, 3, 4, 5	3	1
4	2, 3, 1, 5	1, 2, 3, 4, 5	4	1
5	2, 3, 1, 5, 7	1, 2, 3, 4, 5	4	0.8

The overall result quality here is (1+0.5+1+1+0.8)/5 = 0.86.

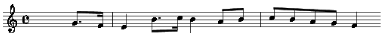
If there would be an additional false positive at position 2, say, ⟨2, 10, 3, 1, 5, 7, 8, 9, 4⟩, the result quality would be lower: 0.7433. False positives lower the result quality in two ways: by shifting subsequent true positives to lower ranks and possibly by shifting true positives out of the scope altogether. Both true and false positives have higher impacts if they occur closer to the beginning of the result list since they influence all subsequent recall values. This illustrates how the criteria number 7 and 8 are met. Criterion 1 is obviously met, and so are criteria 2, 3, and 4. Criteria 5 and 6 are met because of the way r_i is defined: at every group boundary, the set of items that count as relevant is extended by all elements in the next group. Therefore, it does not matter in which order group members are found, as long as they are found before the group boundary.

A more complex example can be found in Table 5.4, which shows the ADR for a sample result of our EMD-based algorithm.

5.2.4 Comparison with normalized discounted cumulative gain

The average dynamic recall (ADR) shares many advantages with the cumulative gain measures introduced by Järvelin and Kekäläinen [30], who state that their

[5] The $rank$ function determines the position of an item within the result list. It is 1 for the first element, 2 for the second one and so forth.



Query: Anonymus: Roslin Castle, RISM A/II signature: 800.000.193

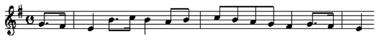
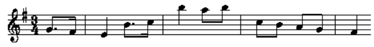
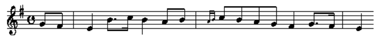
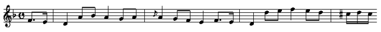
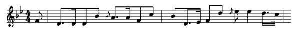
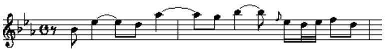
Distance	Found Incipit, Composer, Title, RISM A/II signature	Found	Recall	ADR
0.0010	 Anonymus: Roslin Castle. 000.109.446	1	1	1
0.0012	 Anonymus: Roslin Castle. 000.112.692	2	1	1
0.0016	 Anonymus: Roslin Castle. 000.111.779	3	1	1
0.002	 Anonymus: Roslin Castle. 000.132.330	4	1	1
0.0102	 Anonymous: Quemadmodum desiderat cervus. 702.004.201	4	0.8	0.96
0.0107	 Sarti, Giuseppe: Gli amanti consolati 240.003.908-1.36.1	4	0.6667	0.9111

Table 5.4: A sample search result with the ADR calculation. In the"found" column, we list the number of relevant documents found so far. Note that although the second and third match are listed in the inverse order when compared to Table 5.1, they are still both counted as relevant since they belong to the same group.

measures are, among other things, obvious to interpret, are based on recall bases instead of only on retrieved lists, systematically combine document rank and degree of relevance, and, in their normalized forms, support the analysis of performance differences.

- ADR is obvious to interpret: at any number of retrieved items, it gives the average recall among the documents that the user should have seen so far. It can be calculated not only for the first n documents, if n is the number of items in the ground truth, but also for other numbers of documents.
- ADR is based on an absolute ground truth, not on retrieved lists alone, and therefore does not vary uncontrollably if the considered retrieved lists change.
- ADR systematically combines actual document rank and desired document rank.
- ADR supports the analysis of performance differences of different IR methods since it is normalized.

An important difference between ADR and cumulated gain-based measures is that ADR does not rely on relevance scores and therefore does not take them into consideration. This avoids the problem of correctly choosing a discount function for a discounted cumulative gain measure. By choosing the discount function for the normalized discounted cumulative gain (nDCG) [30] accordingly, one can sometimes invert the result of performance analyses. Different discount functions put, for instance, different emphasis on the beginnings of result lists. Because of this, it is possible to construct pairs of result lists that differ at the beginning in a way such that with, for example, log_2 as discount function, the first list gets a better nDCG score than the second one. With log_3 as the discount function and the same pair of lists, the nDCG score of the second list can be better than that of the first list.

Besides the discount function, the relative differences between relevance scores also have a high impact on nDCG results. Changing the relevance scores can also lead to opposite comparison results. So, to make nDCG results meaningful, one has to know exactly how the value of a relevant item decreases with a growing position in the result list – this determines the discount function –, and also exactly how relevant every document is in relation to the other documents. The ADR, on the other hand, only requires a partially ordered list as a ground truth for delivering meaningful results.

A weakness of the ADR is that situations can arise where different documents are both counted as relevant or both as irrelevant, with no distinction between them, although it is known which one of the two should be ranked higher.

As an illustration of this problem, consider a ground truth of $\langle(1),(2),(3),(4)\rangle$ and the result lists $\langle 4,3,5,6\rangle$ and $\langle 3,4,5,6\rangle$. It would be nice if the second result list would get a better score because it is known that item 3 should be ranked higher than item 4. But the ADR does not distinguish between them since at the second position, both item 3 and item 4 are not yet in the dynamic set of relevant documents, and at the third position, it is too late to treat them differently because both item 3 and item 4 are already in the set of encountered documents. In a similar way, one can construct examples where pairs of relevant items from different groups in the ground truth are encountered so late in a result list that both are counted as relevant, no matter in which order they appear, although it is known which one of the two should be ranked higher.

Problems like this can be caused in two ways during the calculation of the ADR: by items which are first counted as irrelevant and later as relevant (like item 3 in the example above), or by items which are encountered at a higher position than the end of the group to which they belong in the ground truth. Therefore, one could break ties like this by calculating an ADR score based on a list containing those problematic items and an inverted ground truth. In this constructed list, all other items are replaced with one item from the most highly ranked group.

In the example above, items 3 and 4 fulfill the condition for inclusion in the constructed list, while items 5 and 6 do not, so we would construct the lists $\langle 4,3,1,1\rangle$ and $\langle 3,4,1,1\rangle$. If we now calculate the ADR on these constructed lists using the inverted ground truth (here: $\langle(4),(3),(2),(1)\rangle$), the problem with items being treated the same although it is known that they should be ranked differently cannot occur anymore (because of the way the list was constructed). The ADR calculated from these constructed lists and the inverted ground truth could be used to break ties. However, to have a measure that is obvious to interpret, we simply used the ADR as described in Section 5.2.3 for our comparison of melodic similarity algorithms.

Table 5.5: Result quality for all MIREX symbolic melodic similarity submissions. ADR=Average Dynamic Recall, NRGB=normalized recall at group boundaries, AP=non-interpolated average precision, and PN= precision at N documents (N is the number of relevant documents).

Rank / Participant	ADR	NRGB	AP	PN
1 Grachten, Arcos & Mantaras	65.98%	55.24%	51.72%	44.33%
2 Orio	64.96%	53.35%	42.96%	39.86%
3 Suyoto & Uitdenbogerd	64.18%	51.79%	40.42%	41.72%
4 Typke, Wiering & Veltkamp	57.09%	48.17%	35.64%	33.46%
5 Lemström, Mikkila, Mäkinen & Ukkonen (P3)	55.82%	46.56%	41.40%	39.18%
6 Lemström, Mikkila, Mäkinen & Ukkonen (DP)	54.27%	47.26%	39.91%	36.20%
7 Frieler & Müllensiefen	51.81%	45.10%	33.93%	33.71%

5.3 MIREX, Symbolic Melodic Similarity

5.3.1 MIREX 2005: queries with exact rhythm and pitches, RISM incipits

The Symbolic Melodic Similarity task at MIREX 2005 was to retrieve the most similar incipits from a subset of the RISM A/II collection, given one of the incipits as a query. For each algorithm, the result lists for 11 queries were compared to a ground truth that was established as described in Section 5.1. The results were evaluated with four measures: Average dynamic recall (see Section 5.2.3), normalized recall at group boundaries, average precision, and precision at N documents (where N is the number of relevant documents).

We submitted the algorithm that is described in Section 3.4. It compares melodies by transforming notes into a two-dimensional weighted point set. For each note, the coordinates are the onset time and pitch values, and the weight is the duration. The weighted point sets are then compared with the Earth Mover's Distance (EMD). The EMD is continuous and provides partial matching. By changing the weighting scheme and ground distance, one can tune it for different purposes. Its continuity makes it suitable for matching queries that are generated by humans (sung or played on a MIDI piano) with entries of a database of symbolic music,without the need for quantizing, time warping, or any other form of tempo or pitch tracking. This strength does not matter in the MIREX 2005 task of matching notated music against other notated music. However, our method still ranks in the middle of the other methods, which are not designed to work with distorted queries.

Result quality measures

Table 5.5 shows various measures of result quality for all algorithms that were submitted for the symbolic melodic similarity task at MIREX 2005.[6] The measures can be split into two groups: those that work with a dynamic set of relevant documents and those that work with one fixed set of relevant documents. The former measures view some documents as relevant only from a certain position on. For example, a document that is somewhat similar to the query, but clearly less similar than two other documents, would be viewed as relevant only beginning with position 3 in the result list. The following measures work with a dynamic set of relevant documents:

- **Average Dynamic Recall:** This measure is described in Section 5.2.3. At any number of retrieved items, it gives the average recall among the documents that the user should have seen so far. For this comparison, it was measured at position N (where N is the number of relevant documents).
- **Normalized Recall at Group Boundaries:** The ground truth from [74] (described in Section 5.1) does not give one ideal ordering of results, but rather an ordering of groups where the ideal order within groups is not known. This measure is based on the recall at the boundaries of those groups.

Average precision and precision at N documents view all relevant documents as equally relevant, even if they belong to different groups according to the ground truth constructed as described in Section 5.1. See Section 5.2.1 for a definition of average precision and precision.

Result quality of our algorithm

Table 5.5 shows that our algorithm is ranked at the median position of all participants at MIREX 2005 for the measures which distinguish between different degrees of relevance. It would rank lower if the two measures which view all relevant items as equally relevant would be used.

Even though our algorithm did not get top results at MIREX 2005, it is still interesting for certain applications because it has some desirable properties that most of the other algorithms lack:

- Our algorithm can handle cases where the query is distorted by tempo variations or pitch variations. This could make it suitable for query-by-humming applications without the need for an extra algorithm for pitch quantisation or tempo tracking.
- Our distance measure is continuous.

5.3.2 MIREX 2006: distorted queries, RISM incipits, and complete polyphonic pieces

Tasks and participants

In 2006, the “Symbolic Melodic Similarity” task was to retrieve MIDI files that contain material which is melodically similar to a given MIDI query. Unlike 2005,

[6] The complete results can be found here: http://www.music-ir.org/evaluation/mirex-results/sym-melody/index.html; the ground truth that was used for training and for evaluation is published at http://rainer.typke.org/mirex05.0.html.

only half the queries were quantized in rhythm and pitch, while the other half was only quantized in pitch but not in rhythm. The latter half was created by singing melodies.

There were three subtasks that differ mainly in the collection of data to be searched:

- Approximately 16,000 incipits from the UK subset of the RISM collection, almost exclusively monophonic. Six queries (three of them quantized).
- 1000 polyphonic Karaoke files. Five queries (two of them quantized; the three sung queries include two versions of the same melody).
- 10,000 randomly chosen MIDI files that were harvested from the Web, most of them polyphonic. Six queries (three of them quantized).

The participating methods are:

- RT – the method described in Section 3.5.2, however, with equal weights for all notes (that is, the duration was ignored);
- FH – an editing distance for quotiented trees by Pascal Ferraro and Pierre Hanna [19];
- KF – a hybrid distance measure by Klaus Frieler and Daniel Müllensiefen [20];
- AU – Alexandra Uitdenbogerd's Start-Match Alignment technique [84], and
- NM – the geometric "P3" algorithm by Kjell Lemström, Niko Mikkilä, Veli Mäkinen and Esko Ukkonen.

For a more detailed description of these methods, see the MIREX abstracts, available from http://www.music-ir.org/mirex2006/index.php/Symbolic_Melodic_Similarity_Results.

Ground Truth

Due to the size of the collections, no ground truth was known in advance. From every participating algorithm, the top ten matches were put into a pool, and human graders judged the relevance.

The raw ground truth data consisted of a rough and a fine relevance score for every item that was returned by an algorithm. For the rough score, a scale of "very similar", "somewhat similar", and "not similar" was used, while the fine score was a number between 0 (not similar) and 10 (very similar). For the polyphonic tasks, algorithms reported where in a MIDI file a match was found, and the graders heard only those excerpts of MIDI files instead of the whole MIDI files. Even if multiple algorithms created excerpts from the same MIDI file, separate relevance scores were collected for each excerpt.

For some measures, an ordered list of relevant items is necessary. These ordered lists were created as follows from the collected relevance scores:

- Calculate average scores for every MIDI file; the rough scores were counted as follows: 0 for "not similar", 1 for "somewhat similar", and 2 for "very similar".

- For each query, order the matches first by the rough and, in case of ties, by the fine score.

- Group together matches with the same average rough score. For example, all items that were rated "very similar" by all graders were put in the top group, followed by another group of items where at least one grader had selected "somewhat similar", and so forth.

- Only include items with average rough scores of better than "somewhat similar". There were very many "somewhat similar" items in the database, therefore an algorithm should not be rewarded too much for putting one of them into one of the top ten positions.

- If the resulting list is longer than 10, remove whole groups at the end until at most 10 items remain; there was one exception where the top group had 11 "very similar" items. This step is aimed at reducing the influence of arbitrary classifications on the overall result. Since we know very little about the correct order within one group, we should not include part of some group and exclude another part of the group. Doing so would reward algorithms that are lucky enough to have returned items that happened to be accidentally included in the ground truth and punish other algorithms that found items which were about as relevant, but got accidentally excluded.

Result quality measures

In addition to the measures used in 2005 (ADR = average dynamic recall, NRGB = normalized recall at group boundaries, AP = average precision and PND = precision at N documents), the following measures were also calculated. These measures do not rely on an ordered list, but just on the rough and fine scores:

- Fine = Sum of fine-grained human similarity decisions (0-10).

- PSum = Sum of human rough similarity decisions: not similar = 0, somewhat similar = 1, very similar = 2.

- WCsum = 'World Cup' scoring: not similar = 0, somewhat similar = 1, very similar = 3 (rewards "very similar").

- SDsum = 'Stephen Downie' scoring: not similar = 0, somewhat similar = 1, very similar = 4 (strongly rewards "very similar").

- Greater0 = not similar = 0, somewhat similar = 1, very similar = 1 (binary relevance judgement).

- Greater1 = not similar = 0, somewhat similar = 0, very similar = 1 (binary relevance judgement using only "very similar").

All measures were normalized so that they lie in the range from 0 to 1.

Result quality of the submitted algorithms

For the monophonic task, there were no significant performance differences between our method and the editing distance for quotiented trees by Pascal Ferraro and Pierre Hanna [19], while all other methods performed worse, no matter which

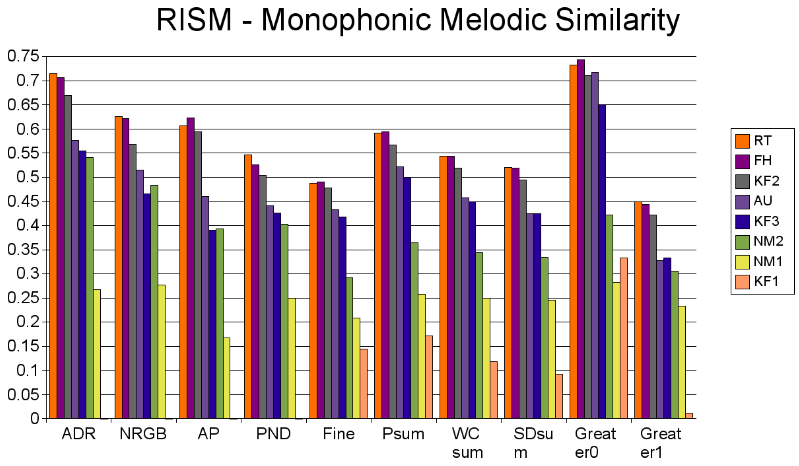


Figure 5.3: Task I: RISM Overall Summary. See Section 5.3.2 for an explanation of the measures and a list of participants.

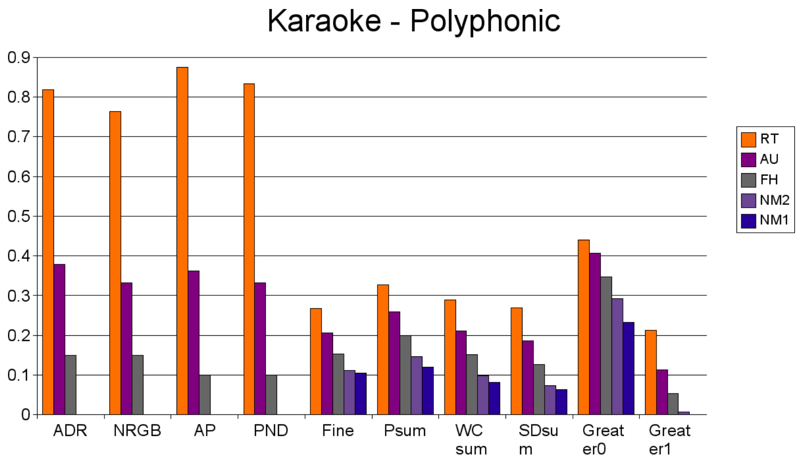


Figure 5.4: Task IIa: Karaoke Overall Summary

measure is used. When looking at the actual result lists – they are available at http://rainer.typke.org/mirex06.0.html, and two examples are shown in Figures 5.6 and 5.7 – the main difference between the results of these two methods seems to be that the latter is more likely to retrieve rather short matches (compare, for example, our result in Figure 5.6 with Ferraro/Hanna's result for the same query, shown in Figure 5.7). In some cases, this might have lead to a lower average precision or average dynamic recall, like for example in the case of http://rainer.typke.org/qr6-fh.0.html, where short matches pushed nice longer ones down to lower ranks.

For both polyphonic tasks, our method clearly outperforms the other methods.

Besides the obvious difference in the number of notes that can sound at the same time, the polyphonic collections also differ in other ways from the monophonic RISM collection:

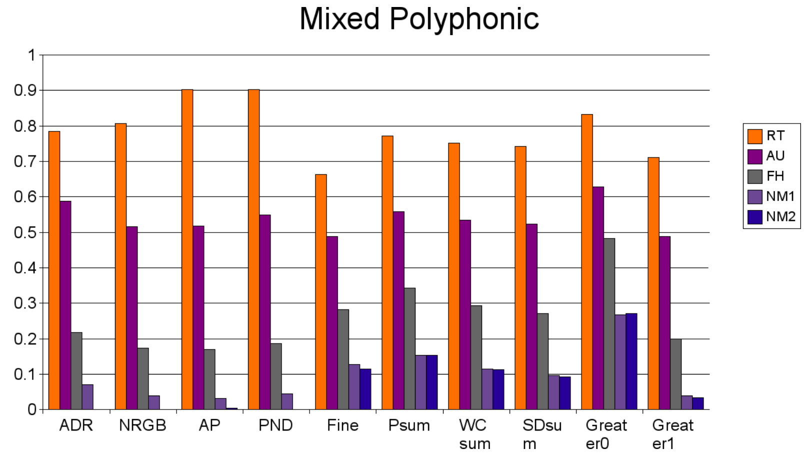


Figure 5.5: Task IIb: Mixed Polyphonic Overall Summary

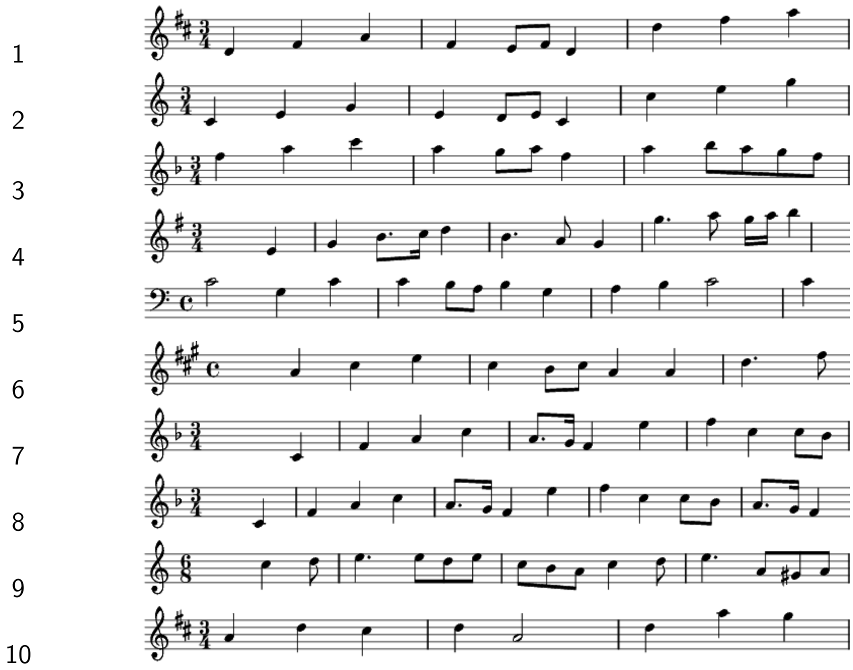


Figure 5.6: A sample result list for the monophonic RISM task, created by our own algorithm. The query is identical to the top-ranked incipit.

- Both polyphonic collections were random sets of files that were harvested from the Web. Because of this, the encoding quality was not as homogeneous as for the RISM collection. Some files in the polyphonic collection were not even syntactically correct MIDI files.

- While the RISM collection was created from plaine&easie code and therefore

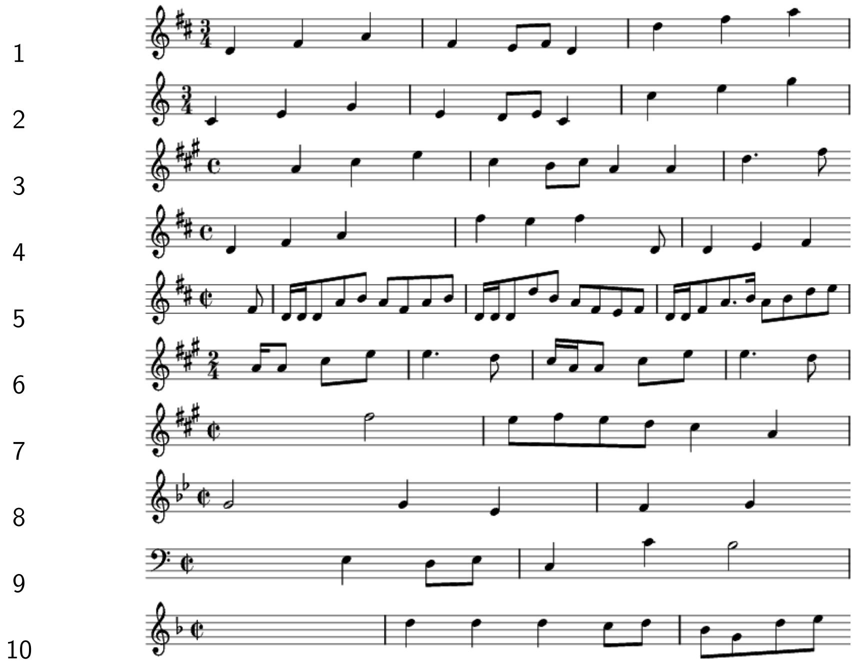


Figure 5.7: Ferraro's and Hanna's result for the same query as in Figure 5.6.

rhythmically quantized, the polyphonic collections contained both quantized music and renditions of performances, where neither onset times nor note durations were exactly what a written score would suggest, and the tempo was not necessarily always stable.

When looking at Figures 5.4 and 5.5, it is quite noticeable that the six rightmost measures, which were not based on the ranked lists described in Section 5.3.2, indicate a much worse performance for the Karaoke task for all algorithms. The reason is that the Karaoke collection was much smaller (1000 items instead of the 10,000 items in the mixed collection) and therefore contained fewer good matches to begin with. Even an ideal algorithm can therefore not reach a score of 1 for measures such as "Fine" for the Karaoke collection. The four measures on the left side, on the other hand, compare the algorithms' outputs with the ground truth lists.

Query response times

Thanks to the vantage index (see Chapter 4), the EMD-based method has short response times – only one method, Alexandra Uitdenbogerd's, needs a few seconds less for answering a query – and it scales well. For all other algorithms, the response time grows by a larger factor when comparing the polyphonic "Karaoke" task with its collection size of 1000 and the polyphonic "Mix" task with a collection of 10,000 items.

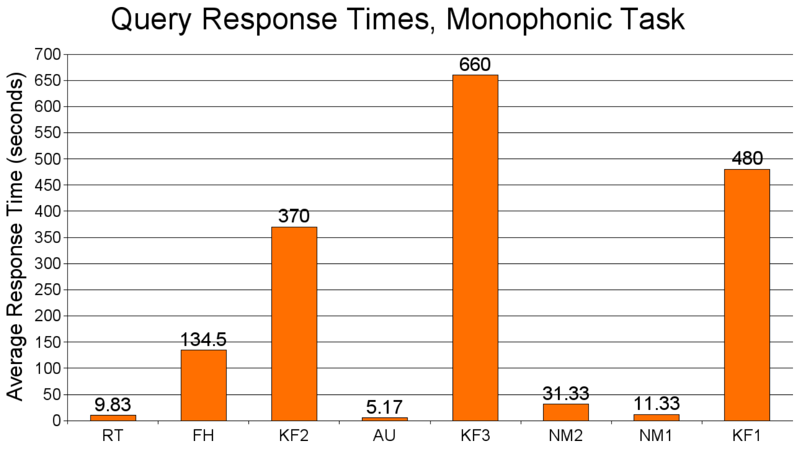


Figure 5.8: Query response times for the monophonic task

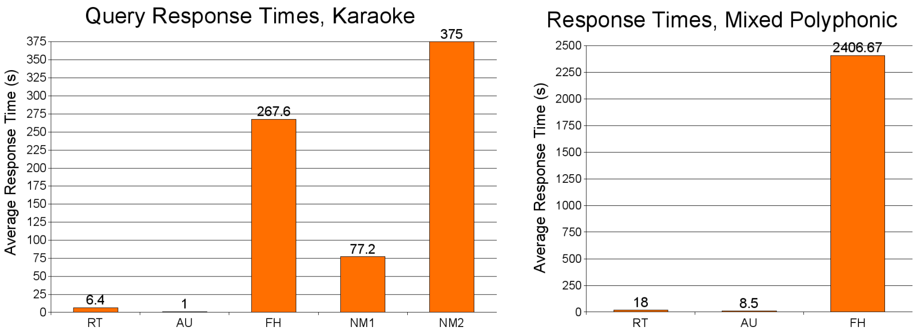


Figure 5.9: Query response times for the polyphonic tasks. The NM method is not shown for the polyphonic mix because the runtimes for indexing and answering queries were not separately available.

5.3.3 Conclusions

Our own comparisons with other algorithms for symbolic melodic similarity – the comparison with Schlichte's and Howard's incipit comparisons in Section 3.3 and the comparison with PROMS/C-Brahms in Section 3.5.1 – have shown that using transportation distances for measuring melodic similarity is a promising approach. The results from MIREX 2005 and 2006 are probably more meaningful than our own comparisons since they involve many more competing algorithms, the procedure for comparing them and measuring the results was discussed and agreed on by all participants, and the actual comparison was done by a neutral third party (Stephen Downie's IMIRSEL team at the university of Illinois at Urbana-Champaign). All of this leaves less room for any bias towards one method. The main reasons for our algorithm getting much better results at MIREX 2006 in comparison to 2005 include:

- The "Symbolic Melodic Similarity" task in 2006 was harder and took advantage of strengths of transportation distances. Half the queries were rhythmically distorted in 2006, while all queries were exact transcriptions of the notation in 2005. In 2006, the polyphonic data consisted largely of non-quantized music, while in 2005, all data to be searched was quantized notated music.

- In 2005, we relied on a genetic algorithm for finding an optimum alignment of query segments with matching incipits. This algorithm was not guaranteed to find the optimum, and it was rather slow. In 2006, we segmented both the query and the database documents, and we used multiple segment lengths for the database. This allowed us to not only align the query segments more efficiently with the matching documents from the database, but also to better control the allowed scaling and translations, and the algorithm was guaranteed to find the optimum within the given constraints.

Chapter 6

A network flow distance for chords

Although, as we have seen in Chapters 3 and 5, the Earth Mover's Distance (EMD) and the Proportional Transportation Distance (PTD) have quite nice properties, they are not perfect. One particularly noticeable problem with the EMD is the fact that with everything else being equal, if one increases the weight of one point in the lighter point set by a large enough amount, one can reach a point where more points from the other point set must be matched to it. There can be similar effects when one uses the PTD: if one increases the weight proportion that is associated with one point by a large enough amount, more points from the other point set are matched to it. This does not directly correspond to music perception – there is no reason why a more important note should be matched with more notes than a less important note. What is really intended – and also achieved – by attaching more weight to certain notes is to increase their influence on the overall distance. See Figure 6.1 for an example of a flow where some weight flows between two points that are musically unrelated.

It would be nice to have a distance measure that preserves as much of the desirable qualities of transportation distances as possible – in particular, continuity and the triangle inequality at least for some cases –, but avoids the effect of matching an important note with unrelated notes.

Another desirable feature would be to normalize weights as it is done for the Proportional Transportation Distance, but still have the possibility of partial matching in the pitch dimension.

This chapter describes a measure that comes closer to these two goals than either the Earth Mover's Distance or the Proportional Transportation Distance.

6.1 Describing transportation distances with maximum flow/minimum cost network flow problems

The Earth Mover's Distance and Proportional Transportation Distance can both be described as special cases of a network flow distance [59], where a certain total amount of weight has to be moved across a network such that the flow is maximized

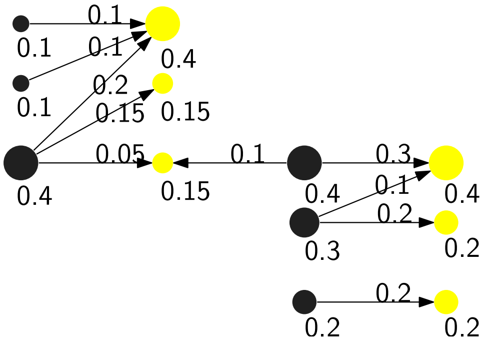


Figure 6.1: An example for unwanted flows with the EMD. Both the dark and the light point set represent two successive chords. To make the flow better visible, the point sets have been pulled apart horizontally; the onset times of the first chords are actually identical. Since the total weight in the two chords differs between the point sets, some of the weight is carried from one chord to the other, which does not make much musical sense. With a local weight normalization, this can be avoided (see Figure 6.5).

and the costs are minimized. The network contains one source node and one sink node. There is a connection from the source node to each node representing a point in the first point set A; these connections each have a capacity that corresponds to the target point's weight, and the cost for using those connections is zero. Each point in point set A is connected to each point in point set B with a connection whose capacity corresponds to the minimum of the two points' weights. Every connection between the point sets A and B has a cost that depends on the ground distance between the two connected points. The points in point set B are each connected to the sink node with connections whose capacities correspond to the weight of the point set in B, and whose costs are zero. The problem of finding a maximum flow at minimum cost through such a network is equivalent to calculating a transportation distance between the two point sets. See Figure 6.2 for an illustration. We simplify the network suggested by Ramon and Bruynooghe [59] by omitting the two extra nodes for carrying surplus weight because this is not needed if we want to normalize both weight sums and enforce that all weight is transported between nodes that represent notes.

It is useful to enforce that all weight flows through the network. If one allows some weight to remain unmatched without a penalty, it is very hard to construct a measure for which the triangle inequality holds. The network flow distance by Ramon et al. [59] obeys the triangle inequality without enforcing all weight to be matched, but it heavily penalizes any unmatched weight by forcing it to flow via an expensive detour in the network. So even without enforcing all weight to be matched, they still force all of it to flow through the network. For the purposes of measuring melodic similarity, penalizing unmatched weight is generally not a good idea, especially for polyphonic music, because the presence or absence of additional voices does not have a very large influence on the identity of a melody. We want to neither allow weight to remain unmatched without penalty (because that would completely break the triangle inequality), nor do we want to penalize unmatched weight like Ramon et al., because that would be inadequate for our application to music.

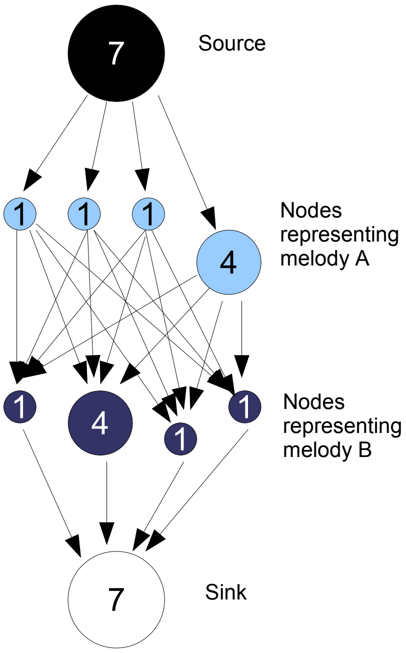


Figure 6.2: Formulation of a transportation distance as a network flow. The black circle at the top is the source, the white one at the bottom is the sink. Source and sink are both connected to each point in one of the two point sets with connections whose capacities correspond to the connected points' weights, and which involve costs of zero. The connections between the two point sets, on the other hand, carry costs that correspond to the ground distance between the connected points.

Enforcing all weight to be matched is also an elegant way of avoiding discontinuities. If, under certain circumstances, we allow some weight to remain unmatched, we would have to take special precautions to ensure that at the point where a point set changes from a situation where all weight is matched to a situation where some weight remains unmatched, the matched amount does not change in a discrete step. It could change in a discrete step, for example, if we would remove a connection with a non-zero capacity from the network. If discontinuities exist, one can always construct point sets that are rather similar but have quite different distances to the same vantage object. Therefore, even if the triangle inequality holds, vantage indexing can become much less useful if there are large discontinuities.

6.2 A network flow distance with normalized weights for chords

6.2.1 First aim: realistic matches for notes

We would like to create a transportation distance that does not have the problem described in the introduction to this chapter, that is, increasing the weight of one particular point should not necessarily lead to it being matched to more points in the other point set, in particular not to points that represent notes with very different onset times.

An obvious idea might be to remove connections according to some rule that specifies under what circumstances notes may be matched with one another. This way, one might, for example, want to modify the network shown in Figure 6.2 such that it is turned into the network shown in Figure 6.3 before calculating the maximum flow with minimum cost.

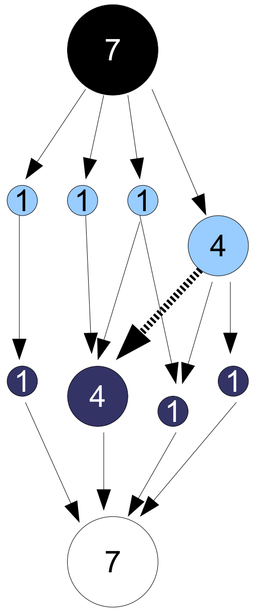


Figure 6.3: A naïve approach to enforcing rules about acceptable matches of notes is to just remove unwanted connections. For example, doing this to the network shown in Figure 6.2 could result in a network like this one. If one would also remove the dashed arrow (because the rightmost long note in the top point set really should not be matched with the last three notes in the bottom point set), solving the flow problem becomes infeasible.

However, there are two problems with this naïve approach of removing unwanted connections: if it is possible to avoid bad matches, the algorithm for finding the maximum flow at minimum cost will always do so anyways (because "bad" connections will always be associated with large ground distances and therefore high costs), and therefore by removing these connections, one does not gain anything.

Secondly, even worse, if the removal of some connections has any effect, it will be the effect that solving the network flow problem is no longer feasible. This would, for example, happen if the algorithm for removing connections would decide that the dashed arrow in Figure 6.3 should be removed. We would still want to move the entire weight of 7 from the source to the sink over the network, but this would not be possible, and therefore there would be no minimum cost, and the distance between the two point sets would be undefined.

Instead of somwhat arbitrarily removing connections, one can also approach the aim of reducing unrealistic flows by ensuring a certain balance in the weight distribution before calculating the optimum flow. The following constraint can do this:

- The proportion of the total weight present between the onset and the offset of any note (or chord) should equal the proportion of the time covered by this note (or chord).

Note that this constraint implies that for monophonic music, the weight simply encodes the duration of a note. For polyphonic music, even with this constraint, one can still use the weight for encoding different degrees of importance for different notes, independent of the note duration.

If this constraint is in place, a single note can still be matched with multiple other notes, but only as long as the durations and the onset times of the involved notes make this plausible.

6.2.2 Second aim: polyphony, fuzzy queries

By adding arcs between nodes that represent the same point set, we can build some more desirable properties into the network flow distance measure. In order to support polyphonic matching of sequences of chords or fuzzy monophonic queries that contain several possibilities for some notes, one can connect all notes that belong to the same chord with arcs of cost zero and a capacity that allows the complete weight of the lighter of the two points to flow to the other point. Notes can be defined to belong to the same chord if their overlap in time exceeds a certain threshold such as 75 % of the duration of the shorter note. Figure 6.4 shows an example.

Overall, we construct networks like this:

1. Represent the two segments of music that are to be compared as weighted point sets, as described in Section 3.1.

2. Identify chords in both point sets and connect any two notes that belong to the same chord with an arc whose cost is zero and whose capacity equals the weight of the lighter of the two points. Notes are said to belong to the same chord if their overlap in time exceeds a certain percentage of the duration of the shorter of the two notes.

3. For each chord, calculate onset and offset time (find the earliest onset time of any note in the chord, and the latest offset time).

4. For each chord, set the total weight to the duration of the chord (the difference of onset time and offset time), preserving the weight proportions within the chord.

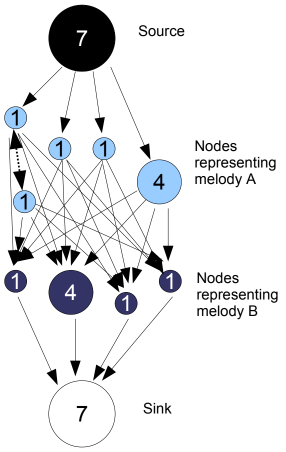


Figure 6.4: An example for how chords can be modeled by adding arcs within point sets. Point set A here contains two notes which overlap in time. In such cases, we add a connection whose capacity allows the entire weight of the lighter of the two points to flow to the other point, and whose cost is zero.

5. For each point set, set the total weight to 1, preserving the weight proportions within the point set.
6. Add a special source node with weight 1 and connect it to every point in the first point set. Each of these arcs has cost zero and a capacity that equals the weight of the node representing a note.
7. Add a special sink node with weight 1 and connect it to every point in the second point in a similar fashion.
8. Connect each point in the first point set with each point in the second point set with a connection whose cost corresponds to the ground distance between the two points and whose capacity equals the weight of the lighter of the two connected points.

6.3 Properties

The new network flow distance with balanced weights and partial matching in the pitch dimension (BWPP) is constructed in a way such that it has the following properties:

1. **The BWPP is continuous under weight modification, the addition or removal of points, and changes in the pitch coordinate.** Neither the weight normalization nor the added arcs within point sets introduce discontinuities if weights are modified or points are added or removed. If weights are modified, this either is undone by the weight normalization, or it has an impact on the overall distance that is proportional to the modification of the weight. If points are added or removed, the change in the overall distance

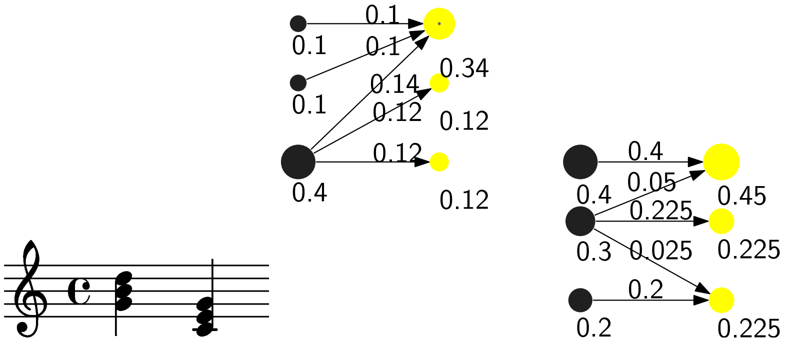


Figure 6.5: By normalizing the weights of chords, we can avoid unwanted flows like in Figure 6.1 without giving up the principle that all weight should be matched. Since flows within a chord do not incur costs, the proposed distance measure would result in a distance of zero between these two pairs of chords, which makes sense because they consist of identical notes.

can be zero, or it is proportional to the weight of the added or removed point. If a point is moved up or down in pitch, this causes changes in the ground distance that are proportional to the change in the coordinate.

2. **The BWPP is not always continuous if the onset time of notes is changed.** If the time coordinate of a point is changed such that it starts or stops being part of a chord, even a small change of the time coordinate can cause a discontinuity because suddenly, all points that are part of the chord in question have one additional possible connection to the other point set, or lose one. However, this kind of discontinuity can only occur if there are chords; as long as all point sets are monophonic, changes in the coordinates of points do not cause discontinuities.

3. **The BWPP obeys the triangle inequality if there are no chords.** Since we normalize the total weight for both point sets, the BWPP obeys the triangle inequality for the same reasons the Proportional Transportation Distance obeys the triangle inequality. However, as soon as we introduce arcs within point sets, it becomes possible to construct point sets for which the BWPP does not obey the triangle inequality anymore. Figure 6.6 shows an example where this occurs.

4. **The BWPP is suitable for fuzzy queries or for matching arbitrary combinations of polyphonic and monophonic music.** Thanks to the added arcs within chords, one can try several alternatives for one note in the query at once. For example, if it is not known what the second note in a monophonic query should be, but there are several possible candidates, one could put all possible notes into a chord where the second note would be, and the distance measure would find matches for all possibilities.

6.3.1 Experimental evaluation with MIREX 2006 data

We created a vantage index using the new distance measure for the “Karaoke” collection of 1000 MIDI files from MIREX 2006 and calculated Average Dynamic

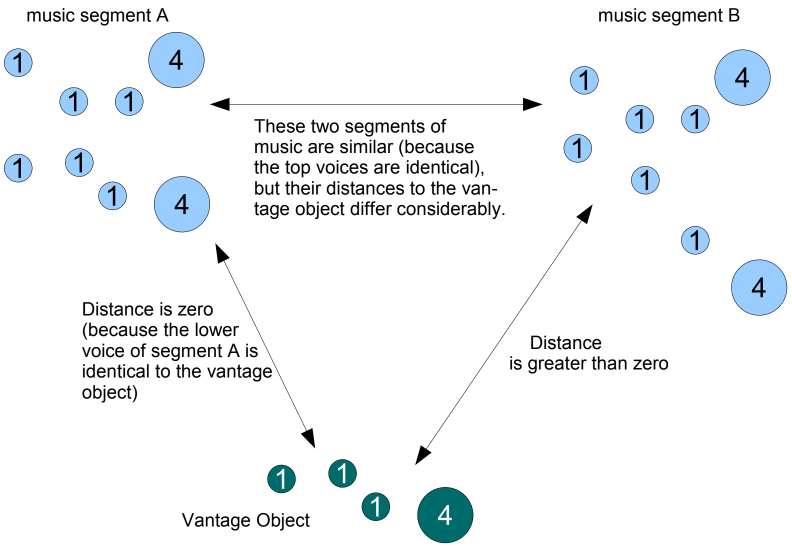

music segment A
music segment B
4
1
These two segments of music are similar (because the top voices are identical), but their distances to the vantage object differ considerably.
Distance is zero (because the lower voice of segment A is identical to the vantage object)
Distance is greater than zero
Vantage Object

Figure 6.6: An example for how the triangle inequality can be violated for point sets representing polyphonic music.

Table 6.1: Result quality for the MIREX 2006 symbolic melodic similarity "Karaoke" task. ADR at MIREX=Average Dynamic Recall for the measure we submitted to MIREX 2006, ADR with new measure=the ADR score of the measure described in this chapter.

Query number	**ADR at MIREX**	**ADR with new measure**
1	1	0.67
2	0.54	0
3	0.56	0
4	1	0.5
5	1	0
Average	0.82	0.23

Recall (ADR) at rank 10 for the same queries that were used at MIREX 2006. The results can be seen in Table 6.1.

Since we only counted true positives that were ranked among the top ten items, we did not count two matches for one query which the new measure found but our MIREX 2006 distance measure did not. The new measure placed them between the tenth and the twentieth position, so they were still ranked higher than 98 percent of the 1000 pieces in the collection. However, this does not change the fact that the new network flow distance performs much worse than the PTD.

An important difference between the new measure and the MIREX version is that at MIREX, all points were created with equal weights, that is, all notes were given exactly the same importance. For this experiment with the new distance measure, however, we gave the points weights that were proportional to the note durations. Therefore, one cannot deduct from this table that the new network distance always performs worse than the PTD. This result does indicate, however, that it was probably a good decision to give all points equal weights at MIREX 2006.

There is still a large class of cases where even with intra-point set weight distributions that are normalized according to the new measure, unwanted flows occur that do not make much musical sense. The next section describes this problem and how it can be solved.

6.4 Using inter-onset time instead of note durations for distributing weights

Even if for every chord (or, for monophonic melodies, for every note), the weight is proportional to the time that is covered by the chord (or note), situations can arise where very unrealistic flows can occur. Figure 6.7 shows an example (two slightly different renditions of a motive from Johannes Brahms, "Ein deutsches Requiem", first movement). The cause of the problem is that imbalances are not prevented by this constraint if the inter-onset intervals of notes are noticeably larger than their durations. With real-world data such as the MIREX Karaoke collection, this happens quite frequently.

The top point set in Figure 6.7 has a much heavier first note than the bottom point set, but all but the last notes have the same weights. Because of this, the weight surplus is forced to flow all the way to the last note of the bottom point set. If weight is forced to flow from the first to the last note like this, the resulting distance depends on the difference in onset times of two largely unrelated notes, which does not make much musical sense. The coordinates and weights for the top point set are (in the format <onset time><pitch><weight>): 0 7.2 0.5/0.75 7.4 0.125/1 7.6 0.25/1.5 7.4 0.25/2 7.2 0.25/2.5 7.1 0.25/3 6.9 0.25/3.75 7.1 0.125. The bottom point set is 0 7.2 0.375/0.8 7.4 0.125/1.25 7.6 0.25/1.75 7.4 0.25/2.25 7.2 0.25/2.5 7.1 0.25/3 6.9 0.25/3.5 7.1 0.25. As we can see, the inter-onset interval is always larger than the note duration.

Weight imbalances can be prevented in a larger number of cases if we replace the constraint from Section 6.2.1 with:

- The proportion of the total weight present between the onset and the offset of any note (or chord) should equal the proportion of the time covered by its inter-onset interval (or, for the last note or chord, its duration).

If we normalize weights for the two point sets from Figure 6.7 by applying this constraint, we get the two new point sets 0 7.2 0.375/0.75 7.4 0.125/1 7.6 0.25/1.5 7.4 0.25/2 7.2 0.25/2.5 7.1 0.25/3 6.9 0.375/3.75 7.1 0.125 (top) and 0 7.2 0.4/0.8 7.4 0.225/1.25 7.6 0.25/1.75 7.4 0.25/2.25 7.2 0.125/2.5 7.1 0.25/3 6.9 0.25/3.5 7.1 0.25 (bottom). We leave everything else equal, but get the much more realistic flow that is shown in Figure 6.8. This flow does not only look more realistic, but it involves much lower costs: 0.167 instead of 0.33 (with the Euclidean distance as ground distance), which is desired because the melodic similarity is very pronounced.

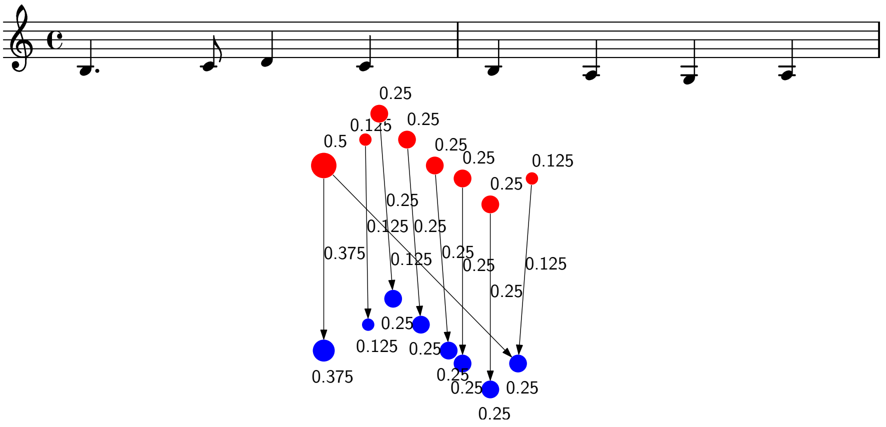


Figure 6.7: Making the weight proportional to the time proportion that each note covers does not always prevent the optimum flow from being unrealistic. The top and bottom point sets are slightly different renditions of the same motive from Brahms' "Ein deutsches Requiem", first movement, and should therefore be recognized as rather similar.

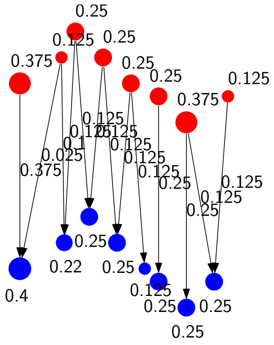


Figure 6.8: By using inter onset intervals instead of note durations for enforcing an even distribution of weight, the problem shown in Figure 6.7 can be avoided.

6.5 Conclusions

By using the more general formulation of network flows, one can construct transportation distances with certain desired properties. One can use weight normalizations, arc capacities and costs, and additional arcs to modify the properties of the resulting dissimilarity measures. We have shown one possible measure that reduces unrealistic matching of notes and that supports fuzzy queries and polyphonic matching. The introduction of chords within which free weight flows are possible, however, means that the triangle inequality does no longer hold.

For making the violations of the triangle inequality less problematic for vantage indexing, one might want to experiment with multiple small groups of vantage objects; for each group, one would search the intersection of rings around vantage objects, as shown in Figure 4.2. This would limit the number of false positives, which, as we have seen in Section 4.4, already drops significantly with small numbers of vantage objects. The false negatives that are introduced by the problem illustrated in Figure 6.6 could then be counteracted by looking at the union of the candidate sets produced by different sets of vantage objects. If the vantage objects in different groups are sufficiently different, the violations of the triangle inequality for any item should differ for each set of vantage objects. A false negative would then only occur if one is unlucky enough to encounter a severe enough violation of the triangle inequality with every group of vantage objects.

Chapter 7

Conclusions and future work

For the last decade, the possibilities for Music Information Retrieval (MIR) research have been much better than fourty years ago, when Kassler gave his talk [31] about an assembler-like language called MIR. At Kassler's time, using computers for processing audio signals was so time-consuming that he did not even mention this possibility in his optimistic talk. The abundance of cheap storage space and processing power is not just a gradual, but a qualitative change since it allows a whole new range of problems to be addressed. This is probably an important reason why the last ten years have seen so much more progress than the previous thirty.

Part of that progress was the development of new ways of measuring melodic similarity. In the late nineties, most publications about query-by-humming systems described string-based methods for comparing melodic contours by using editing distances. While it is possible to efficiently find nearest neighbours when using such distance measures, for example with vantage indexing, using such distance measures for polyphonic music it is rather awkward, and these methods also do not naturally support ornamentations very well where single notes need to be matched to multiple notes.

Early geometric algorithms that are able to match arbitrary groups of notes, possibly polyphonic, to other groups of notes, suffered from a lack of suitable indexing methods. Also, distance measures that rely on counting coinciding note onsets are usually not continuous and work well only for quantized music.

This book shows that by using transportation distances, one can overcome many of the limitations of both other geometric algorithms and of string-based methods. Transportation distances can be designed such that they are continuous and support the triangle inequality. At MIREX 2006, it was shown that the result quality, response times, and scalability of our method compare favourably with those for string-based methods or other state-of-the-art geometric algorithms. However, especially for collections with a noticeable continuum of melodic similarity such as the RISM UK collection of musical incipits, the overlap of relevant items found by different algorithms was not very large, and therefore there is still quite some room for improvement.

To make methods for measuring melodic similarity more useful, it would be desirable to combine audio feature extraction methods with symbolic approaches like the distance measures that are the topic of this book. One of the big open problems of MIR, automatic polyphonic audio-to-MIDI transcription, is probably going to remain unsolved for quite a while. However, it is conceivable that with transportation distances, even imperfect transcription results could be made searchable in a useful way thanks to the robustness against various types of errors one can achieve with transportation distances.

Transportation distances could be further improved in various ways, for example:

- To take full advantage of variants of transportation distances that obey the triangle inequality, but are only suitable for monophonic music, such as the Proportional Transportation Distance, one should work on better voice splitting algorithms and on recognizing which voices are only accompaniment and therefore not worth searching for melodies.

- To still be able to use vantage indexing for distance measures with desirable properties for measuring melodic similarities, but with the disadvantage of not guaranteeing that the triangle inequality always holds, it could be interesting to experiment with combinations of multiple vantage indices (for lowering the number of false negatives) and with vantage objects that are constructed with the aim of either making violations of the triangle inequality less likely, or at least very different from those encountered with other vantage objects.

If the recent trends in MIR research continue, we should have very nice retrieval algorithms at our disposal fourty years from now. And maybe even the pesky old Optical Music Recognition problem which Kassler thought would just require a million dollars and negligible time will eventually be solved as well.

Bibliography

[1] *Répertoire International des Sources Musicales (RISM). Serie A/II, manuscrits musicaux après 1600.* K. G. Saur Verlag, München, Germany, 2002.

[2] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Wu. An optimum algorithm for approximate nearest neighbor searching. In *Proceedings of the Fifth ACM-SIAM Symposium on Discrete Algorithms*, pages 573–582, 1994.

[3] E. Backer and P. van Kranenburg. On musical stylometry – a pattern recognition approach. *Pattern Recognition Letters*, 26:299–309, 2005.

[4] D. Bainbridge, S. J. Cunningham, and J. S. Downie. Greenstone as a music digital library toolkit. In *ISMIR Proceedings*, pages 42–43, 2004.

[5] J. Barros, J. French, W. Martin, P. Kelly, and M. Cannon. Using the triangle inequality to reduce the number of comparisons required for similarity-based retrieval. In *Proceedings of SPIE, Storage and Retrieval for Still Image and Video Databases IV*, pages 392–403, 1996.

[6] M. Bosma, R. C. Veltkamp, and F. Wiering. Muugle: A modular music information retrieval framework. In *ISMIR Proceedings*, 2006.

[7] E. Brochu and N. de Freitas. "Name That Song!": A probabilistic approach to querying on music and text. *NIPS.Neural Information Processing Systems: Natural and Synthetic*, 2002.

[8] D. Byrd and M. Schindele. Prospects for improving OMR with multiple recognizers. In *ISMIR Proceedings*, 2006.

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

[10] W. I. Chang and E. L. Lawler. Sublinear approximate string matching and biological applications. *Algorithmica*, 12(4/5):327–344, 1994.

[11] M. Clausen, R. Engelbrecht, D. Meyer, and J. Schmitz. PROMS: a web-based tool for searching in polyphonic music. In *ISMIR Proceedings*, 2000.

[12] M. Clausen and F. Kurth. A unified approach to content based and fault tolerant music identification. In *International Conference On Web Delivering of Music.*, 2002.

[13] R. Clifford, M. Christodoulakis, T. Crawford, D. Meredith, and G. Wiggins. A fast, randomised, maximum subset matching algorithm for document-level music retrieval. In *International Conference on Music Information Retrieval (ISMIR)*, 2006.

[14] S. Cohen. *Finding Color and Shape Patterns in Images*. Stanford University, Department of Computer Science, 1999. Ph.D. thesis.

[15] W. S. Cooper. Expected search length: A single measure of retrieval effectiveness based on weak ordering action of retrieval systems. *Journal of the American Society for Information Science and Technology*, 19(1):13–41, 1968.

[16] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press and McGraw-Hill, 2001.

[17] P. J. O. Doets, M. Menor Gisbert, and R. L. Lagendijk. On the comparison of audio fingerprints for extracting quality parameters of compressed audio. In *Security, steganography, and watermarking of multimedia contents VII, Proceedings of the SPIE*, 2006.

[18] J. S. Downie. *Evaluating a simple approach to music information retrieval: Conceiving melodic n-grams as text.* PhD thesis, University of Western Ontario, London, Ontario, Canada, 1999.

[19] P. Ferraro and C. Godin. An edit distance between quotiented trees. *Algorithmica*, 36:1–39, 2003.

[20] K. Frieler and D. Müllensiefen. Extended abstract for submissions to mirex 2006: Three algorithms for symbolic similarity computation. 2006. `http://www.music-ir.org/evaluation/MIREX/2006_abstracts/SMS_frieler.pd% f`.

[21] J. Futrelle and J. S. Downie. Interdisciplinary communities and research issues in music information retrieval. In *ISMIR Proceedings*, 2002.

[22] A. Ghias, J. Logan, D. Chamberlin, and B. C. Smith. Query by humming - musical information retrieval in an audio database. In *Proceedings ACM Multimedia*, 1995.

[23] D. Hawking. Overview of the TREC-9 Web Track. In *Proceedings of the 9th Text Retrieval Conference TREC-9*, Gaithersburg, MD, 2001.

[24] W. B. Hewlett. A base-40 number-line representation of musical pitch notation. *Musikometrika*, 4:1–14, 1992. Retrieved April 1, 2003, from http://www.ccarh.org/publications/reprints/base40/.

[25] S. Hillier and G. J. Lieberman. *Introduction to Mathematical Programming*. McGraw-Hill, 1990.

[26] H. Hoos, K. Renz, and M. Görg. GUIDO/MIR - an experimental musical information retrieval system based on guido music notation. In *ISMIR Proceedings*, pages 41–50, 2001.

[27] J. Howard. *Plaine and Easie Code: A Code for Music Bibliography*, volume Beyond MIDI: The Handbook of Musical Codes. Harvard, 1997. Editor: E. Selfridge-Field.

[28] J. B. Howard. Strategies for sorting melodic incipits. *Computing in Musicology, Melodic Similarities: Concepts, Procedures, and Applications*, 11:119–128, 1998.

[29] J.-S. Jang, H.-R. Lee, and J.-C. Chen. Super MBox: An efficient/effective content-based music retrieval system. In *9th ACM Multimedia Conference*, pages 636–637, 2001.

[30] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, 20(4):422–446, 2002.

[31] Michael Kassler. Toward musical information retrieval. *Perspectives of New Music*, 4(6), 1966.

[32] J. Kekäläinen and K. Järvelin. Using graded relevance assessments in IR evaluation. *Journal of the American Society for Information Science and Technology*, 53(13):1120–1129, 2002.

[33] V. Klee and G.J. Minty. How good is the simplex algorithm? In O. Shisha, editor, *Inequalities*, volume III, pages 159–175. Academic Press, New York, NY, 1972.

[34] D. Knuth, Jr J. H. Morris, and V. Pratt. Fast pattern matching in strings. *SIAM Journal on Computing*, 6(2):323–350, 1977.

[35] A. Kornstädt. Themefinder: A web-based melodic search tool. In W. Hewlett and E. Selfridge-Field, editors, *Melodic Similarity: Concepts, Procedures, and Applications, Computing in Musicology*, volume 11. MIT Press, Cambridge, 1998.

[36] N. Kosugi, Y. Nishihara, T. Sakata, M. Yamamuro, and K. Kushima. A practical query-by-humming system for a large music database. In *Proceedings ACM Multimedia*, pages 333–342, 2000.

[37] F. Kurth. A ranking technique for fast audio identification. In *International Workshop on Multimedia Signal Processing.*, 2002.

[38] F. Kurth, A. Ribbrock, and M. Clausen. Efficient fault tolerant search techniques for full-text audio retrieval. In *112th Convention of the Audio Engineering Society*, 2002.

[39] F. Kurth, A. Ribbrock, and M. Clausen. Identification of highly distorted audio material for querying large scale data bases. In *112th Convention of the Audio Engineering Society*, 2002.

[40] J. H. Lee and J. S. Downie. Survey of music information needs, uses, and seeking behaviours: Preliminary findings. In *ISMIR Proceedings*, pages 441–446, 2004.

[41] K. Lemström, V. Mäkinen, A. Pienimäki, M. Turkia, and E. Ukkonen. The C-BRAHMS project. In *ISMIR Proceedings*, pages 237–238, 2003.

[42] K. Lemström and J. Tarhio. Transposition invariant pattern matching for multi-track strings. *Nordic Journal of Computing*, 2003.

[43] R. M. Losee. Text retrieval and filtering: analytic models of performance. *Kluwer Academic, Boston.*, 1998.

[44] McNab, Smith, Bainbridge, and Witten. The New Zealand digital library MELody inDEX. *D-Lib Magazine*, May 1997.

[45] P. Min. Evofunc: evolutionary optimisation function of the library "Reusable Evolutionary Algorithms in Shape Matching" (REALISM). `http://www.cs.uu.nl/centers/give/multimedia/matching/shame.html`.

[46] D. Müllensiefen and K. Frieler. Measuring melodic similarity: Human vs. algorithmic judgments. In R. Parncutt, A. Kessler, and F. Zimmer, editors, *Proceedings of the Conference on Interdisciplinary Musicology (CIM04) Graz/Austria*, 2004.

[47] M. Müller, F. Kurth, and M. Clausen. Audio matching via chroma-based statistical features. In *ISMIR proceedings*, pages 288–295, 2005.

[48] M. Müller, H. Mattes, and F. Kurth. An efficient multiscale approach to audio synchronization. In *ISMIR proceedings*, 2006.

[49] MusicIP/predixis.com. Open fingerprint architecture whitepaper. `http://musicip.com/themes/predixis/downloads/ Open_Fingerprint_Architecture_Whitepaper_v1.pdf`.

[50] D. Ó Maidín. A geometrical algorithm for melodic difference. *Computing in Musicology, Melodic Similarities: Concepts, Procedures, and Applications*, 11:65–72, 1998.

[51] H.-W. Nienhuys and J. Nieuwenhuizen. Lilypond, a system for automated music engraving. In *Proceedings of the XIV. Colloquium on Musical Informatics (XIV CIM 2003)*, 2003. See http://www.lilypond.org.

[52] P. P. Giannopoulos and R. C. Veltkamp. A pseudo-metric for weighted point sets. In *Proceedings of the 7th European Conference on Computer Vision (ECCV)*, pages 715–730, Copenhagen, Denmark, 2002. Springer-Verlag.

[53] F. Pachet. Content management for electronic music distribution. *CACM*, 46(4):71–75, 2003.

[54] F. Pachet, A. Laburthe, and J.-J. Aucouturier. The Cuidado Music Browser: An end-to-end EMD system. In *Proceedings of the 3rd International Workshop on Content-Based Multimedia Indexing*, 2003.

[55] F. Pachet, A. Laburthe, and J.-J. Aucouturier. Popular music access: The Sony Music Browser. *Journal of American Society for Information Science*, 2003.

[56] E. Pampalk, A. Rauber, and D. Merkl. Content-based organization and visualization of music archives. In *Proceedings of ACM Multimedia*, pages 570–579, 2002.

[57] S. Pauws. CubyHum: a fully operational query by humming system. In *ISMIR Proceedings*, pages 187–196, 2002.

[58] L. Prechelt and R. Typke. An interface for melody input. *ACM Transactions on Computer-Human Interaction*, 8(2):133–149, 2001.

[59] J. Ramon and M. Bruynooghe. A polynomial time computable metric between point sets. *Acta Informatica*, 37:765–780, 2001.

[60] A. Rauber and M. Frühwirth. Automatically analyzing and organizing music archives. In *Proceedings of the 5. European Conference on Research and Advanced Technology for Digital Libraries*, Lecture Notes in Computer Science. Springer, 2001.

[61] A. Rauber, E. Pampalk, and D. Merkl. Content-based music indexing and organization. In *Proceedings of the 25. ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 409–410, 2002.

[62] A. Rauber, E. Pampalk, and D. Merkl. Using psycho-acoustic models and self-organizing maps to create a hierarchical structuring of music by musical styles. In *ISMIR Proceedings*, pages 71–80, 2002.

[63] A. Rauber, E. Pampalk, and D. Merkl. The SOM-enhanced jukebox: Organization and visualization of music collections based on perceptual models. *Journal of New Music Research (JNMR)*, 32(2):193–210, 2003.

[64] A. Ribbrock and F. Kurth. A full-text retrieval approach to content-based audio identification. In *International Workshop on Multimedia Signal Processing*, 2002.

[65] J. J. Rocchio. *Document retrieval systems - Optimization and evaluation. PhD dissertation.* Harvard, 1966.

[66] Y. Rubner. 1998. Source code for the Earth Mover's Distance software retrieved April 1, 2003, from http://robotics.stanford.edu/˜rubner/emd/default.htm.

[67] Y. Rubner, C. Tomasi, and L. J. Guibas. A metric for distributions with applications to image databases. In *Proceedings of the 1998 IEEE International Conference on Computer Vision*, pages 59–66, 1998.

[68] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

[69] J. Schlichte. Der automatische Vergleich von 83243 Musikincipits aus der RISM-Datenbank: Ergebnisse - Nutzen - Perspektiven. *Fontes artis musicæ*, 37:35–46, 1990.

[70] E. Selfridge-Field. Conceptual and representational issues in melodic comparison. *Computing in Musicology*, 11:3–64, 1998.

[71] B. Snyder. *Music and memory: an introduction*. MIT Press, 2000.

[72] R. Typke, M. den Hoed, J. de Nooijer, F. Wiering, and R. C. Veltkamp. A ground truth for half a million musical incipits. Techical report UU-CS-2004-060, Universiteit Utrecht, 2004.

[73] R. Typke, M. den Hoed, J. de Nooijer, F. Wiering, and R. C. Veltkamp. A ground truth for half a million musical incipits. In *Proceedings of the 5th Dutch-Belgian Information Retrieval Workshop (DIR)*, pages 63–70, 2005.

[74] R. Typke, M. den Hoed, J. de Nooijer, F. Wiering, and R. C. Veltkamp. A ground truth for half a million musical incipits. *Journal of Digital Information Management*, 3(1):34–39, 2005. Ground truth data available at http://rainer.typke.org/mirex05.0.html.

[75] R. Typke, P. Giannopoulos, R. C. Veltkamp, F. Wiering, and R. van Oostrum. Using transportation distances for measuring melodic similarity. In *ISMIR Proceedings*, pages 107–114, 2003.

[76] R. Typke, P. Giannopoulos, R. C. Veltkamp, F. Wiering, and R. van Oostrum. Using transportation distances for measuring melodic similarity. Techical report UU-CS-2003-024, Universiteit Utrecht, 2003.

[77] R. Typke, R. C. Veltkamp, and F. Wiering. Searching notated polyphonic music using transportation distances. In *Proceedings of the ACM Multimedia Conference*, pages 128–135, New York, 2004.

[78] R. Typke, R. C. Veltkamp, and F. Wiering. A measure for evaluating retrieval techniques based on partially ordered ground truth lists. In *International Conference on Multimedia & Expo (ICME)*, 2006.

[79] R. Typke, F. Wiering, and R. C. Veltkamp. A search method for notated polyphonic music with pitch and tempo fluctuations. In *Proceedings of the Fifth International Conference on Music Information Retrieval (ISMIR)*, pages 281–288, 2004.

[80] R. Typke, F. Wiering, and R. C. Veltkamp. Evaluating the earth mover's distance for measuring symbolic melodic similarity. 2005. `http://www.music-ir.org/evaluation/mirex-results/ articles/similarity/typke.pdf`.

[81] R. Typke, F. Wiering, and R. C. Veltkamp. A survey of music information retrieval systems. In *ISMIR Proceedings*, 2005.

[82] R. Typke, F. Wiering, and R. C. Veltkamp. MIREX symbolic melodic similarity and query by singing/humming. 2006. `http://www.music-ir.org/evaluation/MIREX/2006_abstracts/SMS_QBSH_typke.pdf`.

[83] R. Typke, F. Wiering, and R. C. Veltkamp. Transportation distances and human perception of melodic similarity. *ESCOM Musicæ Scientiæ*, 2007.

[84] A. L. Uitdenbogerd. Variations on local alignment for specic query types. 2006. `http://www.music-ir.org/evaluation/MIREX/2006_abstracts/SMS_uitdenboge% rd.pdf`.

[85] E. Ukkonen, K. Lemström, and V. Mäkinen. Geometric algorithms for transposition invariant content-based music retrieval. In *ISMIR 2003: Proceedings of the Fourth International Conference on Music Information Retrieval*, pages 193–199, 2003.

[86] E. Ukkonen, K. Lemström, and V. Mäkinen. Sweepline the music! *Computer Science in Perspective*, pages 330–342, 2003.

[87] R. H. van Leuken, R. C. Veltkamp, and R. Typke. Selecting vantage objects for similarity indexing. In *International Conference on Pattern Recognition (ICPR)*, 2006.

[88] R. van Zwol and H. Oostendorp. Google's "I'm feeling lucky", truly a gamble? In *Proceedings of the Fifth International Conference on Web Information Systems Engineering*, Brisbane, Australia, 2004.

[89] R. C. Veltkamp, F. Wiering, and R. Typke. Content based music retrieval. In Borko Furht, editor, *Encyclopedia of Multimedia*. Springer, 2006.

[90] J. Vleugels and R. C. Veltkamp. Efficient image retrieval through vantage objects. *Pattern Recognition*, 35(1):69–80, 2002.

[91] E. M. Voorhees and D. K. Harman. *Overview of the eighth Text REtrieval Conference (TREC-8)*. 2000.

[92] A. Wang. An industrial strength audio search algorithm. In *ISMIR Proceedings*, Baltimore, 2003.

[93] F. Wiering, R. Typke, and R. C. Veltkamp. Transportation distances and their application in music-notation retrieval. *Music Query: Methods, Strategies, and User Studies (Computing in Musicology)*, 13:113–128, 2004.

[94] E. Wold, T. Blum, D. Keislar, and J. Wheaton. Content-based classification, search, and retrieval of audio. *IEEE Multimedia*, 3(3):27–36, 1996.

[95] S. Wu and U. Manber. Fast text searching allowing errors. *CACM*, 35(10):83–89, 1992.

[96] Peter N. Yianilos. Excluded middle vantage point forests for nearest neighbor search. Technical report, NEC Research Institute, Princeton, NJ, July 1998.

Samenvatting

Voor het zoeken naar muziek op het Web, in een bibliotheek, of in een persoonlijke verzameling van muziek kan het nuttig zijn om een zoekmachine te hebben die de muziek zelf doorzoekt en niet alleen meta-gegevens. De huidige zoekmachines werken vooral op basis van titel en artiestnamen, en niet met melodieën of andere muzikale inhoud. In dit proefschrift wordt op een fundamentele manier ingegaan op de vraag hoe men kan uitrekenen hoe melodieën (of meer algemeen, verzamelingen van muzieknoten) op elkaar lijken.

Een overzicht van systemen voor Music Retrieval

In hoofdstuk 2 (pagina 9) worden verschillende bestaande systemen en algoritmes voor het zoeken naar muziek beschreven. Het bevat ook een overzicht van taken en gebruikers voor MIR (Music Information Retrieval) systemen. Er blijkt een gat te bestaan tussen systemen die op een heel algemeen niveau zoals genre werken en systemen voor heel specifieke taken zoals identificatie van een werk of een opname. Taken zoals intertekstualiteit of de identificatie van een artiest worden niet door vele systemen ondersteund. Waarschijnlijk is het nodig om daarvoor muziek op een hoger, meer abstract niveau dan noten te representeren.

Een afstandsmaat voor melodieën

Elke muzieknoot wordt omgezet in een twee-dimensionaal punt met een gewicht dat de belangrijkheid van de noot aangeeft. Hoe belangrijker de noot in een melodie is, hoe groter het gewicht. De dimensies zijn het tijdstip waarop een noot begint en de toonhoogte. Zie figuur 3.1 op pagina 24.

Met de zogenaamde Earth Mover's Distance wordt uitgerekend hoe goed twee verzamelingen van dat soort punten op elkaar lijken. Dit gaat als volgt. De ene verzameling wordt opgevat als een stel bergjes aarde, de andere als een stel gaten in de grond. Vervolgens wordt uitgerekend op welke manier de gaten het efficiëntst gevuld kunnen worden met de aarde. De (minimale) hoeveelheid moeite die het vullen van de gaten kost, is een maat voor de gelijkenis van de twee puntenverzamelingen: als er weinig verschil is, dan liggen de bergjes en gaten dicht bij elkaar, en dan kost het weinig moeite om de gaten te vullen. Figuur 3.3 op pagina 31 geeft een illustratie van het berekende transport van de bergjes (bovenste rij) naar de gaten (onderste rij). De melodieën in dit voorbeeld lijken muzikaal sterk op elkaar, en ook het berekende transport is klein.

Deze methode is uitgetest op een collectie van bijna 500.000 'incipits', korte muziekfragmenten die opgenomen zijn in een catalogus van muziekhandschriften (RISM A/II). Een aanzienlijk deel van de handschriften is anoniem overgeleverd. Met deze methode zijn kandidaat-componisten opgespoord voor 18.000 anonieme muziekstukken. Zie figuur 3.5 op pagina 33 voor een voorbeeld. Daarvoor was het wel nodig om een index te bouwen over de collectie. Stel dat een enkele vergelijking tussen twee puntenverzamelingen een milliseconde zou duren, dan had het vergelijken van alle anonieme stukken meer dan een jaar geduurd. Met de ontwikkelde indexeringsmethode was dat een kwestie van minuten.

Indexeringsmethode: Vantage Objects

Vleugels en Veltkamp hebben vantage indexing ontwikkeld voor het zoeken in objectruimten waarin de driehoeksongelijkheid geldt. Als een afstandsmaat aan de driehoeksongelijkheid voldoet, is het mogelijk om een zoekopdracht met behulp van vantage objects te versnellen. Om naar een nieuw object te zoeken, worden de afstanden naar de vantage objecten berekend, en met deze informatie kan op een efficiënte manier een verzameling van objecten met ongeveer dezelfde afstanden uit de database geselecteerd worden. Slechts deze objecten worden dan met de zoekvraag vergeleken.

In dit proefschrift wordt beschreven hoe vantage indexing voor muziek kan worden toegepast. Wij splitsen de muziek in segmenten en gebruiken vantage indexing om segmenten te vinden die op een segment uit de zoekvraag lijken. Voor het zoeken naar melodieën is het niet altijd goed om alleen naar "nearest neighbours" te zoeken, maar ook het zoeken in een vaste zoekradius heeft nadelen. Wij werken met een variabel zoekradius om de voordelen van een "nearest neighbour" search en het zoeken binnen een bepaalde radius te combineren.

Omdat in verschillende muziekstukken vaak dezelfde segmenten voorkomen (tenminste als de segmenten kort genoeg zijn), is het mogelijk om veel geheugen te besparen door de tabel met muzieksegmenten en afstanden naar vantage objects te splitsen in een tabel met unieke segmenten en hun afstanden naar vantage objects en een tweede tabel met de informatie waar en in welken muziekstukken de segmenten voorkomen.

Een belangrijke vraag is hoeveel vantage objects er voor een optimale zoeksnelheid nodig zijn. Een groter aantal vantage objects betekent minder false positives en dus minder werk voor het uitrekenen van de werkelijke afstanden voor de nearest neighbours, maar het betekent ook meer werk voor het indexeren van gegevens, en de index neemt meer ruimte in beslag. We hebben voor een constante aantal kandidaten experimenten met 1 tot 8 vantage objects gedaan.

Niet alleen het aantal, maar ook de kwaliteit van vantage objects heeft een grote invloed op hoe goed het zoeken werkt. In een goede verzameling van vantage objects zitten erg verschillende vantage objects; als twee vantage objects te sterk op elkaar lijken, levert het tweede vantage object geen extra informatie op en kan worden weggelaten. Voor ieder vantage object is ook belangrijk dat voor verschillende muzieksegmenten de afstand naar het vantage object ook niet te gelijk is.

Evaluatie van zoekalgoritmes voor muziek

MIREX is een internationale muziekretrievalcompetitie. In 2005 werd voor MIREX in de categorie vergelijking van genoteerde melodieën een "ground truth" gebruikt die in dit onderzoek was vervaardigd. Voor enkele zoekvragen werden ongeveer 50 kandidaten uit de RISM A/II collectie geselecteerd. Menselijke experts bepaalden dan een ideale volgorde van deze incipits. In de praktijk produceren verschillende algoritmes verschillend geordende resultaten; met de nieuw ontwikkelde maat voor de kwaliteit van zoekresultaten, "Average Dynamic Recall" (ADR), kunnen deze worden vergeleken met de ideale volgorde. Met ADR is het mogelijk om "ground truths" te gebruiken die slechts partieel geordend zijn. Als de ideale volgordes van meerdere menselijke experts tot een algemene ideale volgorde verwerkt worden, gebeurt het vaak dat de rangen van groepen van melodieën met een hoge statistische significantie verschillen, maar niet de rangen van individuele melodieën. Met ADR kan de betrouwbare informatie over verschillen tussen groepen worden gebruikt zonder dat de minder betrouwbare informatie over verschillen binnen deze groepen invloed op de uitslag heeft.

Bij MIREX 2006 behaalde de in dit proefschrift beschrevene vergelijkingsmethode de eerste plaats voor genoteerde melodieën.

Een algemenere manier om transportafstanden te beschrijven: Network Flows

In hoofdstuk 6 (pagina 98) wordt beschreven hoe transportafstanden zoals de "Earth Mover's Distance" op een algemenere manier kunnen gemodelleerd worden. Met "Network Flows" is het mogelijk om nieuwe afstandsmaten met bepaalden eigenschappen te ontwikkelen. Wij beschrijven als een voorbeeld een afstandsmaat voor akkoorden dat het mogelijk maakt om iedere combinatie van monofone en polyfone muziek te vergelijken. Met dit maat treedt ook een probleem niet op dat met de Earth Mover's Distance wel kan optreden: voor sommige combinaties van noten wordt met de Earth Mover's Distance gewicht van een noot naar een andere noot getransporteerd die veel later of eerder voorkomt. Deze noten zijn dan niet goed vergelijkbaar. Zie figuur 6.7 (pagina 107) voor een illustratie.

Acknowledgements

I thank my promotor Mark Overmars and the co-promotors Remco C. Veltkamp and Frans Wiering for all the work they put into this project. Many thanks also go to my coauthors Reinier H. van Leuken, Marc den Hoed, and Justin de Nooijer for the valuable input they have provided for our papers. I am grateful for the time and effort that Louis Grijp, Michael Clausen, Geraint Wiggins, Arnold Smeulders, and Linda van der Gaag have invested into being part of my reading committee. Gisela Gerritsen, with her excellent Dutch class, has made Utrecht a much more welcoming place than it otherwise would have been, as have friends like Henriette, Junying, and Ona, playing with orchestras such as the Orkest van Utrecht and the Nederlands Sinfonietta, and playing chamber music with Henk, Andries, and Lucie. Last but definitely not least, the support of my dear wife Agatha and her and my parents Zbigniew, Krystyna, Gudrun, and Juergen has been vital for successfully completing this thesis.

Curriculum Vitae

Rainer Typke

11. 04. 1973	born in Waiblingen, Germany
1984–1992	**Gymnasium in der Taus,** Backnang, Germany
1992, 1994	**IBM Deutschland Entwicklung GmbH**, Böblingen, Germany: Software development
1992–1993	Community Service (instead of mandatory military service): Long-term day-to-day care and support for severely disabled patients
1994–1999	**Universität Karlsruhe**, Germany: 5 year studies terminating with "Diplom" in Computer Science (Similar to an undergraduate course combined with an MS)
1997	**Duke University**, Durham, North Carolina: Instructor for Duke's TIP Program (see http://www.tip.duke.edu).
1999–2003	**Accenture**, Frankfurt, Germany: Analyst, on March 1, 2001 promotion to Consultant.
2003–2007	**Universiteit Utrecht**: Assistent in opleiding (PhD student)
From 2007	Researcher at the **Austrian Research Institute for Artificial Intelligence**, Vienna

Advanced School for Computing and Imaging

This work was carried out in the ASCI graduate school.
ASCI dissertation series number 137.

ISBN-10: 90-393-4441-8
ISBN-13: 978-90-393-4441-5