ISMIR 2001

October 15-17, 2001

2nd Annual

International Symposium on Music Information Retrieval

Indiana Memorial Union

Indiana University

Bloomington, Indiana, USA

Edited by J. Stephen Downie and David Bainbridge

ISMIR 2001

2nd Annual International Symposium on Music Information Retrieval

October 15-17, 2001

Sponsored by:

The National Science Foundation
4201 Wilson Boulevard, Arlington, Virginia 22230, USA
Tel: 703-292-5111, FIRS: 800-877-8339 | TDD: 703-292-5090
This meeting is supported by the National Science Foundation under Grant No. 9909068. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

INDIANA UNIVERSITY
Office of the Vice President for Information Technology and CIO
http://www.indiana.edu/~ovpit/
University Information Technology Services
http://www.indiana.edu/~uits/
Indiana University Digital Library Program
http://dlib.indiana.edu/

National Center for Supercomputing Applications
University of Illinois at Urbana-Champaign
http://www.ncsa.uiuc.edu/

Edited by J. Stephen Downie and David Bainbridge

ISMIR 2001

October 15, 2001

Welcome friends and colleagues to the 2nd Annual International Symposium on Music Information Retrieval – ISMIR 2001.

Following on the heels of last year's groundbreaking inaugural conference, we're convening with colleagues this year at the beautiful campus of Indiana University, Bloomington. We hope the information exchange fostered by this conference will facilitate innovation and enhance collaboration in this dynamic area of research.

This year's program is rich in content and variety. We are honored to have David Cope present this year's keynote address. The presentations by our four invited speakers, Roger Dannenberg, Jef Raskin, Youngmoo Kim, and Adam Lindsay provide added depth and breadth to an already dynamic and diverse program.

This document includes the texts of the accepted papers along with the extended abstracts of the invited talks and poster presentations. All are also available on the ISMIR 2001Web site at: http://ismir2001.indiana.edu/

As with last year, we were very encouraged by the number and quality of submissions. Response to our Call for Papers was remarkable. Selecting the twenty papers for presentation (out of 40 submissions) and the eighteen posters for exhibition was no easy task. I'd like to personally thank all those that gave of their time to help review submissions.

Unending appreciation and thanks must be extended to the Program Committee: David Bainbridge (Program Chair), Gerry Bernbom, Donald Byrd, Tim Crawford, Jon Dunn, and Michael Fingerhut.

Additionally I'd like to thank Dr. Stephen Griffin of the National Science Foundation, for helping us secure the foundational funding that made this symposium possible. Dr. Radha Nandkumar of the National Center for Supercomputing Applications, University of Illinois at Urbana-Champaign, must also be thanked for providing financial support to help us augment our student stipend program.

Finally, several departments and individuals at our host institution, Indiana University Bloomington, deserve thanks. A great deal of the planning that went into this conference was done by Diane Jung, Charles Rondot, David Taylor, and Les Teach, of the Communications and Planning Office under the Office of the Vice President for Information Technology and CIO, and by Tawana Green and the staff of the IU Conference Bureau. Their assistance is much appreciated. In addition, the Indiana University School of Music generously supplied an extraordinary instrument, a fortepiano, for the mini-recital at the Mathers Museum.

Sincerely,

J. Stephen Downie
Symposium Chair
Graduate School of Library and Information Science
University of Illinois at Urbana-Champaign

ISMIR 2001
International Symposium on Music Information Retrieval

Program Schedule

* E-mail room computers will be configured with Telnet and Web browsers. Check with your network and/or security personnel on any special arrangements you may need to connect to your home or institutional resources.

ISMIR 2001

Monday, October 15, 2001

2:00-5:00pm	Poster Session Setup, Frangipani Room
2:00-5:00pm	Tour of IU Musical Arts Center and Cook Music Library: leaves from registration table in East Lounge at 2:00pm.
3:00-9:00pm	E-mail room: IMU088 *
4:00-8:00pm	Registration: East Lounge
6:00-8:00pm	Reception/Poster Sessions: Frangipani Room
	Using Long-Term Structure to Retrieve Music: Representation and Matching
	Jean-Julien Aucouturier, Mark Sandler
	Statistical Significance in Song-Spotting in Audio
	Pedro Cano, Martin Kaltenbrunner, Oscar Mayor, Eloi Batlle
	Music Information Retrieval Annotated Bibliography Website Project, Phase 1
	J. Stephen Downie
	Computer-supported Browsing for MIR
	Mikael Fernström, Donncha Ó Maidín
	MIRACLE: A Music Information Retrieval System with Clustered Computing Engines
	J.-S. Roger Jang, Jiang-Chun Chen, Ming-Yang Kao
	A Three-Layer Approach for Music Retrieval in Large Databases
	Kjell Lemström, Geraint A. Wiggins, David Meredith
	Gamera: A Structured Document Recognition Application Development Environment
	Karl MacMillan, Michael Droettboom, Ichiro Fujinaga
	Melody Matching Directly From Audio
	Dominic Mazzoni, Roger B. Dannenberg
	Usage of the MELDEX Digital Music Library
	John R. McPherson, David Bainbridge
	Addressing the Same But Different – Different But Similar Problem in Automatic Music Classification
	Unjung Nam, Jonathan Berger
	Musical Information Retrieval for Delta and Neumatic Systems
	D. Politis, P. Linardis, H. Kotopoulos, A. Alygizakis
	Statistical Analysis in Music Information Retrieval
	William Rand, William Birmingham
	Adaptive User Modeling in a Content-Based Music Retrieval System
	Pierre-Yves Rolland
	Discovering Themes by Exact Pattern Matching
	Lloyd Smith, Richard Medina
	Melodic Resolution in Music Retrieval
	Timo Sorsa, Jyri Huopaniemi
	Sound Spotting – a Frame-Based Approach
	Christian Spevak, Richard Polfreman
	Music Database Retrieval Based on Spectral Similarity
	Cheng Yang
8:00-9:00pm	Posters tear down and move to back of Alumni Hall

* E-mail room computers will be configured with Telnet and Web browsers. Check with your network and/or security personnel on any special arrangements you may need to connect to your home or institutional resources.

Using Long-Term Structure to Retrieve Music: Representation and Matching.

Jean-Julien Aucouturier
SONY Computer Science Labs, Inc.
Rue Amyot, 75005 Paris, France
jjaucouturier@caramail.com

Mark Sandler
Department of Electronic Engineering,
Queen Mary, University of London,
Mile End Road, London E14NS, UK
mark.sandler@elec.qmw.ac.uk

ABSTRACT

We present a measure of the similarity of the long-term structure of musical pieces. The system deals with raw polyphonic data. Through unsupervised learning, we generate an abstract representation of music - the "texture score". This "texture score" can be matched to other similar scores using a *generalized edit distance*, in order to assess structural similarity. We notably apply this algorithm to the retrieval of different interpretations of the same song within a music database.

1. MOTIVATION

Motivation for this system is our belief that a bird-eye-view of a song's long-term structure is often a sufficient description for music retrieval purposes. In particular, our system doesn't use any "transcription" information such as pitch or rhythm. Thus, it can deal with polyphonic music without the problem of instrument separation.

A similar approach has already been illustrated by Foote in [1], where the author designs an algorithm to retrieve orchestral music based on the energy profiles. A drawback of his system is that it requires music with high dynamic variations. To address this problem, our approach is rather based on spectral variation: we uncover and match the succession over time of abstract "spectral textures".

2. REPRESENTATION

A piece of polyphonic music can be viewed as the superposition of different instruments playing together, each with its own timbre. We call "texture" the polyphonic timbre resulting of this superposition. For example, a piece of rock music could be the succession over time of the following textures: {drums}, then {drums + bass + guitar}, then {drums + bass}, then {drums + bass + guitar + voice}, etc…

The front-end for our system is based on work done by the authors in [2]. The musical signal is first windowed into short 30ms overlapping frames. For each of the frames, we compute the short-time spectrum. We then estimate its spectral envelope using Mel Cepstrum Coefficients [3]. A Hidden Markov Model (HMM) [4] is then used to classify the frames in an unsupervised way: it learns the different textures occurring in the song in terms of mixtures of Gaussian distributions over the space of spectral envelopes. The learning is done with the classic Baum-Welsh algorithm. Each state of the HMM accounts for one texture. Through Viterbi decoding, we finally label each frame with its corresponding texture.

Our "texture score" representation is just the succession over time of the textures learned by the model (figure 1). It reveals much of the structure of the song: phrases succeed to phrases, common patterns are repeated every verse and chorus, instrument solos stand out clearly and echo the introduction and ending, etc.

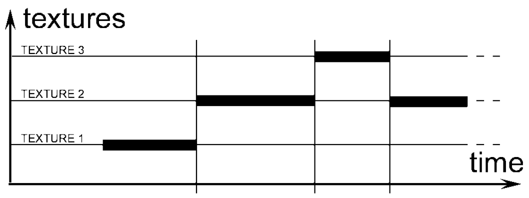


Figure 1: The texture score representation for a few seconds of music.

One interesting property of this representation is that the spectral signification of the textures has been discarded by the HMM. The texture score of figure 1 could correspond to {drums} - {guitar + drums} - {guitar + drums + voice} -{guitar + drums}, but could also well be {cello} - {cello + violin} - {cello + violin + voice} - {cello + violin}, etc. We only know about the *succession* of the textures, not about the textures *themselves*. We will use this property to match different interpretations of the same song (i.e. same long-term structure) which use different instrumentations (i.e. the spectral content of the textures is different).

3. MATCHING

In order to assess the structural similarity of pieces of music, we've designed an appropriate string-matching algorithm to compare texture scores. Each score is a simple string of digits out of a small alphabet: if we've identified 4 textures in the song, the score will be of the form …11221333441… out of the alphabet {1,2,3,4}.

There are three issues that the string-matching algorithm needs to solve:

- *Noise:* similar structures can differ quite a lot locally, so the matching can only be approximate.
- *Time Warping*: two different performances with the same structure can have a different rhythm or expressivity (rubato…).
- *Permutations*: the numeration of the textures by the front-end is arbitrary. This means that a texture which is referred

to as “1” in one song, could be referred to as “3” in another. Therefore, the two strings “112133” and “331322” should be considered to be the same (as they differ only by the following permutation {(1,3), (2,1), (3,2)}).

The first two issues are classically dealt with using dynamic programming to compute an edit distance (also called *Levenshtein distance*) [5]. It gives the minimal number of local transformations (insertion, deletion, substitution) needed to transform – or “edit”- one string into one other.

However, the third issue has not received much coverage in the string matching literature. To avoid the brute force approach consisting of $n!$ distance measures for all permutation of the alphabet, Baker in [6] suggests an interesting factorization method. Unfortunately, it is mainly designed for exact matching (without noise), and is also very dependent on the time scale.

Our integrated solution to these three issues is a *generalized edit distance*, where we progressively adapt the cost of the each elementary substitution as the edit distance between two strings is computed. At the beginning of the process, we “charge” every substitution of one symbol into another, except the identity. By the end of the measure, the costs have changed to “learn” the best permutation between the two strings: we “charge” every substitution (including identity) except the ones corresponding to the permutation between the two strings.

4. TWO APPLICATIONS

4.1 Clustering covers of the same songs

Figure 2 shows the texture scores for the beginning of two versions of the same song, with different instrumentations: the first one is a male singer and an accompaniment based on accordion; the second one has a female singer and violins. Since we have freed ourselves from these spectral differences by using the texture scores, we are able to notice that the two pieces show some similarity. We have applied our algorithm on a database containing different versions of different songs (notably 3 versions of a French song from the 50’s by A. Bourvil, J. Greco and I. Aubret, 4 versions of a Bob Dylan tune, with acoustic or electric guitar, studio or live recording, etc.), and the results are encouraging: the edit distance between “covers” is generally small, and the distance between different songs is big, which allows us to cluster the different interpretations.

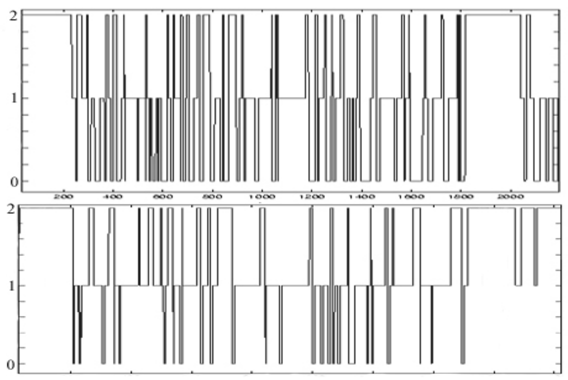


Figure2: Comparison of the texture score representations of two different interpretations of the same song.

4.2 Clustering songs of the same genre.

We have also applied our algorithm to cluster a database containing acoustic blues (3 Robert Johnson tunes, 2 Son House and 2 Tommy Johnson), folk (4 songs by Nick Drake) and country pieces (4 songs by Woody Guthrie). As most of the blues tunes show a common phrase structure (AAB), we are able to gather and separate them from the other pieces. Once again, a bottom-up spectral approach can’t easily succeed in this task, since all the pieces contain mostly the same instrumentation (voice + guitar).

5. CONCLUSION

The texture score is a good representation to study the long-term structure of polyphonic musical signals. In the context of string matching, it provides an efficient retrieval tool to cluster songs with the same structure. Two applications are covers of the same tune, and pieces of the same “structural” genre.

This tool is especially useful since it disregards the spectrum content of the signals. Obtaining the same assessment of structural similarity from the extraction of “transcription” features such as pitch, instrumentation and rhythm would actually require very sophisticated high-level knowledge.

The generation of the texture score involves a machine-learning algorithm, which is quite intensive for a database application (processing a piece of music takes about real time), but once extracted, the score can be stored as metadata, and the retrieval can be performed in reasonable times (it is just an *edit distance*).

Further work includes generating “cleaner” texture scores (for issues on the front-end, see [2]), and optimizing the computation of our *generalized edit distance*. The scheme still has to be tested on a large corpus of tunes and genres to measure a meaningful precision rate, but we believe that these results already show the relevance of this alternative approach to Music Retrieval.

6. ACKNOWLEDGMENTS

Thanks to Maxime Crochemore from Institut Gaspard Monge, Marne-La-Vallee, France, and to Yoan J. Pinzon from King’s College London, UK, for discussions on string matching. This work has been partially supported by the British Council, UK

7. REFERENCES

[1] Foote J., “ARTHUR: Retrieving Orchestral Music by Long-Term Structure”. In Proc. 1st ISMIR, October 2000.

[2] Aucouturier, J-J, and Sandler, M., “Segmentation of Musical Signals Using Hidden Markov Models”. In Proc. AES 110th Convention, May 2001.

[3] Rabiner, L.R. and Juang, B.H., “Fundamentals of speech recognition”. Prentice-Hall 1993

[4] Rabiner, L., “A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition”. In Proc. IEEE, vol. 77, no. 2, 1989.

[5] Crochemore, M., and Rytter, W., “Text Algorithms”. Oxford University Press, 1994.

[6] Baker, B.S., “Parameterized Pattern Matching: Algorithms and Applications”, to appear in J. Comput. Syst. Sci

Statistical Significance in Song-Spotting in Audio

Pedro Cano, Martin
Kaltenbrunner, Oscar Mayor,
Eloi Batlle
Music Technology Group
IUA-Pompeu Fabra University
Barcelona, Spain
34-935422176
{pedro.cano,modin,oscar.mayo
r,eloi.batlle}@iua.upf.es

ABSTRACT

We present some methods for improving the performance a system capable of automatically identifying audio titles by listening to broadcast radio. We outline how the techniques, placed in an identification system, allow us detect and isolate songs embedded in hours of unlabelled audio yielding over a 91% rate of recognition of the songs and no false alarms. The whole system is also able of working real-time in an off-the-shelf computer.

1. INTRODUCTION

A monitoring system able to automatically generate play lists of registered songs can be a valuable tool for copyright enforcement organizations and for companies reporting statistics on the music broadcasted. The difficulty inherent in the task is mainly due to the difference of quality of the original titles in the CD and the quality of the broadcasted ones. The song is transmitted partially, the speaker talks on top of different fragments, the piece is maybe playing faster and several manipulation effects are applied to increase the listener's psycho-acoustic impact (compressors, enhancers, equalization, bass-booster, etc…). An additional difficulty is that there are no markers in broadcasted radio informing when the songs start and end.

In this scenario, the article focus on the pattern matching techniques that, given a sequence of audio descriptors, are able to locate a song in a stream avoiding false alarms. Shortly the whole system works as follows, off-line and out of a collection of music representative of the type of songs to be identified, an alphabet of sounds that describe the music is derived. These audio units are modeled with Hidden Markov Models (HMM). The unlabelled audio and the set of songs are decomposed in these audio units. We end up then with a sequence of letters for the unlabelled audio and a database of sequences representing the original songs. By approximate string matching the song sequences that best resembles the audio the most similar song is obtained. We point out the importance of assessing statistical relevance on the best matching song found in order to avoid false positives. We end up explaining how these techniques can be applied to continuous stream of audio and commenting the results.

2. AUDIO PERCEPTUAL UNITS

From an acoustic point of view, music can be described as a sequence of acoustic events. To be able to identify titles it is relevant to extract information about the temporal structure of these sequences. The first step converts the acoustic signal into a sequence of abstract acoustic events. Speech events are described in terms of phones. In music modeling this is not so straightforward. Using, for instance notes would have disadvantages: Often notes are played simultaneously (accords, polyphonic music) and music samples contain additional voices or other sounds. The approach therefore followed is learning relevant acoustic events, that is, finding the set of "fundamental sounds" in which we can decompose audio and representing them with a letter. The alphabet of audio perceptual units is derived through unsupervised clustering using cooperative HMM from a database of several thousand titles [1].

3. SEQUENCE ALIGNMENT

Having derived HMM models for the audio perceptual units, we can decompose the songs into a symbolic representation. Instead of comparing raw audio, for identifying titles, we compare the sequence of letters of unknown audio against the sequences corresponding to all the songs to identify. The search for a sequence in a database similar to the query sequence is performed by approximate string pattern matching [2]. A measure of the difference between two sequences is the edit distance, defined as the minimum number of character insertions, deletions and substitutions needed to make them equal. An arbitrary weight can be associated with every edit operation, as well as with a match.

The dynamic programming algorithm is guarantied to find the best alignment between a pair of sequences given a particular choice of scoring matrix and gap penalties [3]. There are several variants of the dynamic programming algorithm that yield different kinds of alignments. The Neddleman and Wunsch is a global alignment, that is to say, it aligns the entire length of both sequences. For our particular case this is not suitable since it is typical that a song in the radio is broadcasted partially. The variant known as the Smith-Waterman algorithm yields a local alignment. It aligns the pair of regions within the sequencesIn our application, since the query audio sequence must be compared to several thousand titles, we run a heuristic approximation to the Smith-Waterman algorithm that allows us perform the matching much faster named FASTA[4].

3.1 The choice of substitution scores

The weighted scores for substitutions of the edit distance are calculated to account for bias in the replacement of symbols between the original and the broadcasted song sequences. A set of original CD and corresponding radio songs are selected and manually edited by cutting pieces so that the pieces of audio are synchronized. Then a similarity ratio, R_{ij} is computed for the symbols in the sequences

$$R_{ij} = \frac{q_{ij}}{p_i p_j}$$

where q_{ij} is the relative frequency with which the symbols i and j are observed to replace each other in the manually aligned sequences. p_i and p_j are the frequencies at which the symbols i and j occur in the set of songs in which the substitutions are observed. Their product, $p_i\ p_j$, is the frequency at which they would be expected replace each other if the replacements were random. If the observed replacement rate is equal to the theoretical replacement rate, then the ratio is one ($R_{ij} = q_{ij} / p_i p_j = 1.0$). If the replacements are favored with the manipulative effects above described the ratio will be greater than one and if there is selection against the replacement the ratio will be less than one. The similarity reported in the similarity score matrices S_{ij} is the logarithm to this ratio.

4. STATISTICAL SIGNIFICANCE

Considering the possible uses of the system, a great concern in the similarity searching above described is a false-positive error. We would not like to include in a play list for a copyright enforcement association a song that has not been played. Any two sequences composed of letters from the same alphabet can be aligned to show some measure of similarity. Typically alignment scores of unrelated sequences are small, so that the occurrence of unusually large scores can be attributed to a match. However, even unrelated sequences can occasionally give large scores in the local alignment regime. Although these events are rare, they become important when one attempts a search of a big and expanding sequence database. How often will an event at least as extreme as the one just observed happen if these events are the result of a well defined, specific, random process? It is imperative to understand the statistics of the high-scoring events, in order to estimate the statistical significance of a high-scoring alignment.

In the case of gapless alignment, it is known rigorously [6] that the distribution of alignment scores of random sequences is the Gumbel or extreme value distribution (EVD), which has a much broader tail than that of the Gaussian distribution. For the case of gapped alignment, there is no theory available to predict the distribution of alignment scores for random sequences. It has been conjecture that the score distribution is still of the Gumbel form. Also our tests on sequence of descriptors extracted from audio seem to show a good fit to the Extreme Value Distribution. The EVD is of the form:

$$E = Kmne^{-\lambda S}$$

where E is the expected number of hits with score $>=S$, m is the size of the query sequence, n is the size of the database. λ and K the are Gumbel constants and must be estimated from a large scale comparison of random sequences. The FASTA or various implementation of the SW algorithm, produce optimal alignment scores for the comparison of the query sequence to sequences in the database. Most of these scores involve unrelated sequences, and therefore can be used to estimate λ and K.

5. ON-LINE SYSTEM

We have then a method for comparing fragments of audio against a database of songs for a best match and statistical method for assessing its goodness. Both the symbolic extraction and the matching against the database run fast on a normal machine. The approach for, having a continuous stream of broadcasted audio, identify songs consists in sending hypothesis to match against the database every few seconds. That is, the superstring resulting from the conversion of the raw audio to symbols is windowed with overlap. So every 10 seconds, a sequence corresponding to two and a half minutes of sound is compared to the database. As a result of each comparison a set of candidates is shown along with its expectation (E-value). A candidate with sufficiently low E-value suggests that the query is related to that candidate sequence and therefore can be added to the play list. Along with the candidate sequence, an alignment with the query is provided. With the timing associated to the query sequence an estimation of the beginning and ending time of the song broadcasted can be obtained and printed in the play list.

6. RESULTS

The system has been tested with 24 hours of radio recorded from 10 different stations against a database of around 2500 songs of commercial music. The radio data contains among music, jingles commercials... 147 songs registered in the system (its original version is in the database). The system yields a result of 133 (little over a 91%) songs recognized and no false positive. By lowering the threshold of acceptance of a candidate raises the results to 135 correctly identified but false positives appear as well. When working on-line, the delay between the moment a song starts sounding and it is added correctly to the play list is about one minute as average. The system runs in more than real-time in a Pentium III 500Mhz.

7. REFERENCES

[1] Batlle, E., Cano, P., Automatic Segmentation for Music Classification using Competitive Hidden Markov Models, Proceedings International Symposium on Music Information Retrieval (2000)

[2] Gusfield, D., Algorithms on Strings, Trees and Sequences, Cambridge University Press (1997)

[3] Smith, T.F. and Waterman, M.S., Identification of common molecular subsequences, Journal of Molecular Biology. (1981), 195-197.

[4] Pearson, W.R. and Lipman, D.J. Improved tools for Biological Sequence Comparison. Proc. Natl. Acad. Sci. (1988) 85 : 2444-2448.

[5] Nicholas, H.B., Deerfield D. W., Ropelewski, A.J. A Tutorial on Searching Sequences Databases and Sequence Scoring Methods (1997)

[6] Karlin, S. And Altschul, S.F. Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. Proc. Natl. Acad. Sci. USA 87 (1990), 2264-2268.

Music Information Retrieval Annotated Bibliography Website Project, Phase I

J. Stephen Downie

Graduate School of Library and Information Science
University of Illinois at Urbana-Champaign

501 East Daniel St.
Champaign, IL 61820

jdownie@uiuc.edu

ABSTRACT

Music information retrieval (MIR) as a nascent discipline is blessed with a multi-disciplinary group of people endeavoring to bring their respective knowledge-bases and research paradigms to bear on MIR problems. Communication difficulties across disciplinary boundaries, however, threaten to impede the maturation of MIR into a full-fledge discipline. The principal causes of the communications breakdown among members of the MIR community are a) the lack of bibliographic control of the MIR literature; and, b) the use of discipline-specific languages and methodologies throughout that literature. This poster abstract reports upon the background, framework, goals and ongoing development of the *MIR Annotated Bibliography Website Project.* This project is being undertaken to specifically address and overcome these bibliographic control and communications issues.

1. INTRODUCTION

The problems associated with the creation, deployment, and evaluation of robust, large-scale, and content-based (i.e., music queries framed musically) music information retrieval (MIR) systems are far from trivial. Music information is inherently multi-faceted, multi-representational (i.e., can be represented in many different ways), multi-modal (i.e., experienced in many different ways), and multi-cultural. The complex interaction of Pitch, Temporal, Harmonic, Timbral, Editorial, Textual, Bibliographic, Representational, Experiential, and Cultural facets makes music information difficult to store, and then retrieve, in any robust, large-scale, and comprehensive manner. Simply put, this dizzyingly complex interaction *is* the “MIR problem”.

Because MIR is such a complex and multi-dimensional research problem, many diverse groups of scholars, researchers, and interested parties have begun to explore MIR issues within the frameworks of their particular disciplines. These groups include music and digital librarians, computer scientists, audio engineers, music publishers and retailers, musicologists, information retrieval specialists, intellectual property lawyers, music hobbyists, music psychologists, educators, Internet content providers, broadcasters, and business managers. Students, representing all the aforementioned disciplines, at levels ranging from undergraduate to post-doctorate, are also seeing MIR issues as fruitful and interesting areas of study.

2. THE PROBLEM

A recurring theme brought to the fore by participants at the *International Symposium on Music Information Retrieval* (23-25 October, 2000) was the ignorance many participants felt about MIR work being done in disciplines other than their own. This ignorance had two manifestations. First, participants of discipline ***W*** bemoaned the fact that they did not know that members of discipline ***X*** had been working on a given problem ***Y*** and publishing their findings on ***Y*** in the ***X*** literature for years. For example, computer scientists, and others, were not aware of the extensive musicology literature dealing with music representation codes. Second, participants of discipline ***X*** were distressed by their inability to fully comprehend, and thus evaluate fairly, the contributions being made by members of discipline ***W*** because the contributions of ***W*** were so deeply rooted its discipline-specific language and methods. For example, the music librarians, and others, struggled with the audio engineering presentations because they did not have the educational background needed to evaluate the application of Fast Fourier Transforms and other highly mathematical techniques to a particular MIR problem.

The MIR corpus is scattered willy-nilly across the scholastic landscape with important papers found in the musicology, computer science, information retrieval, information science, and engineering literatures, to name but a few sources. Because of this scattering, it is nowhere uniformly represented in any one of the traditional indexing sources. For example, the musicology-based MIR work is found in various music, arts, and humanities, indexes but not necessarily in the computer science and engineering indexes. Similarly, important engineering-based papers are missing from the arts and humanities indexes, and so on. Since researchers are generally unaware of the differences in scope of the various discipline-based indexes, they tend to focus upon those with which they are most familiar and thus overlook the contributions of those based in other disciplines. Unfamiliarity with the wide-range of vocabularies used by the various disciplines further compounds the communication difficulties by making it problematic for MIR investigators to conduct thorough and comprehensive searches for MIR materials. Until these issues are addressed, MIR will never be in a position to fully realize the benefits that a multi-disciplinary research and development community offers, nor will it be able to develop into a discipline in its own right.

3. PROPOSED SOLUTION

The creation of a Web-based, two-level, collection of annotated bibliographies will overcome many of the communications problems currently plaguing the MIR community (Fig. 1). The first level, or core bibliography, will bring together those items identified as being germane to MIR as a nascent discipline. Thus, the core bibliography will comprise only those papers dealing specifically with some aspect of the MIR problem, such as MIR system development, experimentation, and evaluation, etc. The second level, or periphery bibliographies, will comprise a ***set*** of discipline-specific bibliographies. Each discipline-specific bibliography in the set will provide access to the discipline-specific background materials necessary for non-expert members of the other disciplines to comprehend and evaluate the papers from each participating discipline. For example, an audio engineering bibliography could be used by music librarians and others to understand the basics of signal processing (i.e., Fast Fourier Transforms, etc.). Another example would be a musicology bibliography that computer scientists could draw upon in an effort to understand the strengths and weaknesses of the various music encoding schemes, and so on. Thus, taken together, the two-levels of the MIR bibliography will provide:

a) the much needed bibliographic control to the MIR literature; and,

b) an important a mechanism for members of each discipline to comprehend the contributions of the other disciplines.

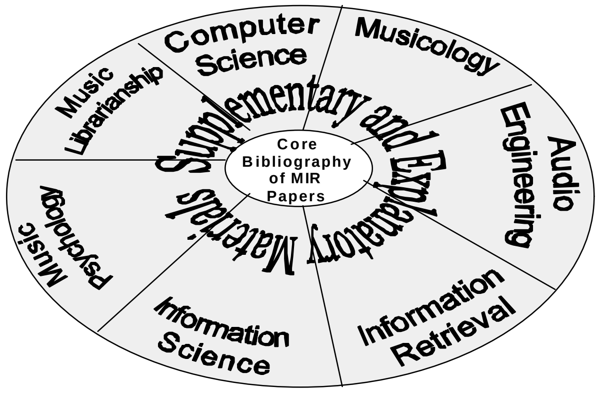


Figure 1. Project Schematic

4. PHASE I COMPONENTS

An important operating principle of the project is the use of non-proprietary formats and software. We are committed to the ideals of the Open Source Initiative [6] and the GNU General Public License [2] and thus intend to make our innovations freely available to others. In keeping with this commitment, we have chosen the *Greenstone Digital Library Software* (GSDL) package [5], the *Apache HTTP* server [1], the PERL scripting language [7], and the Linux operating system [4] to create the basic technological foundation of the project. We have purchased copies of the commercial bibliographic software package, *ProCite* [3] for initial, in-house, data-entry. *ProCite* also provides us with a representative instance of commercially available software that many end-users might utilize in manipulating the records they retrieve from our bibliography.

We have acquired the domain name ***music-ir.org*** under which access to the bibliography will be located (http://www.music-ir.org). At present, there are two central components of project undergoing development and alpha testing:

a) the bibliographic search and retrieval interface using the GSDL package; and,

b) the Web-based end-user data entry system.

For both of these, the goal is to create a system that will permit ongoing viability of the bibliography by minimizing—but not necessarily eliminating—the amount of human editorial intervention required. Item A issues being addressed include modifications to the basic GSDL system to permit the importation of specially structured bibliographic records and their subsequent access through a variety of field selection options. Item B is a CGI-based input system that guides end-users through the process of constructing well-structured bibliographic records through a series of step-by-step interactions and the on-the-fly generation of input forms specifically designed to provide the appropriate fields for the various types of bibliographic source materials (i.e., journal articles, conference papers, theses, etc.).

4.1 Next Steps

Now that the general framework for the core bibliography has been laid, we are moving forward on the acquisition of the supplementary and explanatory materials. For these we will be drawing upon the expert advice of those that have graciously signed on as advisors to the project. These advisors will not only lend their disciplinary expertise but will also afford us a very important multinational perspective on the potential uses and growth of the bibliography.

5. ACKNOWLEDGMENTS

I would like to thank Dr. Suzanne Lodato for helping the project obtain its principal financial support from the Andrew W. Mellon Foundation. I would also like to thank Dr. Radha Nandkumar and the National Center for Supercomputing Applications Faculty Fellows Programme for their support. The hard work and insightful advice of Karen Medina, Joe Futrelle, Dr. David Dubin, and the other members of the Information Science Research Laboratory at the Graduate School of Library and Information Science, UIUC, is gratefully appreciated. Those members of the MIR research community who have volunteered to act as project advisors are also thanked.

6. REFERENCES

[1] Apache Software Foundation. 2001. *Apache project*. Available at: http://httpd.apache.org/ .

[2] GNU Project. 2001. *Free software licenses*. Available at: http://www.gnu.org/licenses/licenses.html

[3] ISI Researchsoft. 2001. *ProCite, Version 5*. Available at: http://www.procite.com .

[4] Linux Online. 2001. *The Linux homepage*. Available at: http://www.linux.org/ .

[5] New Zealand Digital Library Project. 2001. *About the Greenstone software*. Available at: http://www.nzdl.org/cgi-bin/library?a=p&p=gsdl .

[6] Open Source Initiative. 2001. *The open source definition, version 1.9*. Available at: http://www.opensource.org/docs/definition.html

[7] Perl Mongers. 2001. *Perl Mongers: The PERL advocacy people*. Available at: http://www.perl.org/ .

Computer-supported Browsing for MIR

Mikael Fernström
Interaction Design Centre,
Department of Computer Science and Information Systems,
University of Limerick
353(0)61202606
mikael.fernstrom@ul.ie

Donncha Ó Maidín
Centre for Computational Musicology and Computer Music
Department of Computer Science and Information Systems,
University of Limerick
353(0)61202606
donncha.omaidin@ul.ie

ABSTRACT

The word 'browser' has come to acquire an additional meaning to 'one who browses'. 'Browser' is frequently used to describe a piece of software that enables a human, the browser, to engage interactively in visualising and exploring representation of objects in a computer. Users of record or bookshops and readers of newspapers engage in browsing. This activity differs radically from the traditional IR strategy of using a query to select subsets or extract properties from data sets. Browsing is an alternate IR strategy that can be effective for exploration, especially in the cases of unfamiliar domains, or in cases where a well-defined query is not desirable or possible. The provision of rich information spaces allows people to develop an understanding of the space, or the objects within that space, and of the relations between the objects. Two existing browsing prototypes are presented that illustrate how some browsing techniques may be applied to music. They are of particular interest in that they explore the potential for incorporating combined auditory and visual information spaces.

1. INTRODUCTION

With computers, we can extend the abilities of our minds, just as mechanical devices, motors and electricity have enabled us to extend our mechanical and motor abilities. When showing a musicologist a *sonic browser* prototype, he expressed great surprise, *"I've never seen a collection this way before"* (see Figure 1). This is a very important comment as he had been working for a couple of years re-cataloguing the collection, from index cards to database, via desktop publishing tools to its final form – a paper based product, ready to print. Still, in paper or data base format, one cannot get a visual spatial overview of, for example, '*here* are the jigs and *there* are the reels'.
Marchionini and Shneiderman [5] describe browsing as:

- an exploratory, information seeking strategy that depends upon serendipity.
- especially appropriate for ill-defined problems and for exploring new task domains.

This can be the case when musicologists are searching for melodies in a collection. Melodies collected through fieldwork can often be different to older original versions. They can still be the same melodies but with the addition of an individual performers style. This sometimes makes it difficult to use computer-based search algorithms or formal queries.

2. BROWSING MUSIC

To create a *sonic browser* that provides an efficient way for musicologists to find tunes in collections through browsing, we need to:

- Provide users with an overview of the data set.
- Facilitate recognition of information of interest.
- Give details on demand about objects in the data set.
- Provide visual and auditory representations of the data set.
- Provide an interaction style that makes users feel engaged and in control.

3. USE SCENARIOS

To develop an understanding of how musicologists use collections of tunes and the difficulties they experience we interviewed and observed musicologists at work. Most musicologists stated that when they read a score, they use their 'inner ear' to listen to an internal representation of what the melody would sound like. Others, who don't have such a strong ability for internal representation, would often hum or whistle the melody they are reading. Our use scenario also corresponds quite well to user needs described by Alain Bonardi [6], where he lists some key features for a musicologist's workstation.

4. THE SONIC BROWSER

Music is best perceived by listening and the *sonic browser* is focused on the affordances of direct and interactive sonification, i.e. making sound files that are in the user's focus of attention play, with the shortest possible interaction sequence. With the Sonic Browser [2], the user only has to move the cursor around in the screen and soundscape to see and hear the contents or representation of the data set. The application utilises our ability to switch our attention between sounds in the auditory scene, making use of the *cocktail party effect* [1]. With multiple auditory streams it is interesting to note that people have a different ability to differentiate between the number of concurrent sound sources. A metaphor for a user controllable function that makes it visible to the user is the application of an *aura* [3]. An *aura*, in this context, is a function that defines the user's range of perception in a domain. The aura is the receiver of information in the domain. See Figure 1.

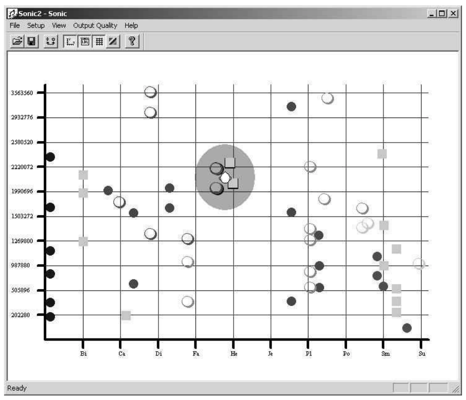


Figure 1: Sonic Browser, with aura around cursor

Using the sonic browser, properties of the melodies can be mapped to arbitrary features of the visual display. File size can be mapped to size of visual symbols, genre to colour, symbol shape time signature, horizontal location to time of collection and vertical location to key signature. Users can at any time change these arbitrary mappings, to suit their needs.

5. C.P.N. BROWSE

A second system is under development. It represents corpora as a weighted graph, where vertices represent tunes, and each weighted edge a significant relationship between the tunes. Building the graph is a process separate from browsing. Building is O(n^2) in time, and can in practice take many hours to compute. The resulting relations are stored for later use by the browser. The system enables one to browse between melodies, where at each stage the closest melodies to the current one are represented sonically and visually (See Figure 2). The cursor enables the user to move between melodies using a semi-acoustic perspective [7].

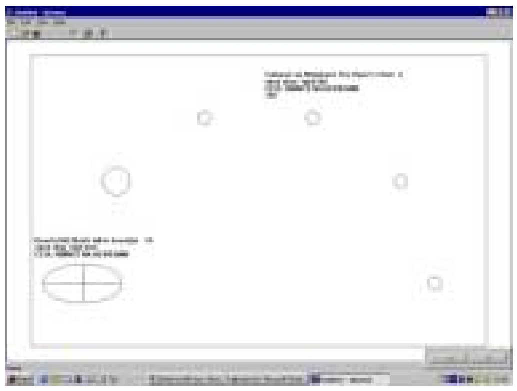


Figure 2: C.P.N. Browse

6. EVALUATION

In user testing with 10 musicologists from the Irish World Music Centre at the University of Limerick, we found that:

- Users performed browsing tasks substantially faster (c. 27%) when multiple-stream audio was used, compared to single-stream audio.
- Users remembered where (in the screen/soundscape) they heard a melody before.

The interaction sequence was made easier and more engaging in the sonic browser than with typical standard tools, as unnecessary mouse clicks were avoided. When the cursor and aura was over symbols representing melodies, all melodies within the aura were be heard. Users could rapidly switch the aura on or off with a single keystroke, to make it easier to pinpoint a particular melody in high-density clusters.

7. CONCLUSION

Using interactive visualisation and sonification techniques can allow us to make new discoveries. Our ability to see and hear new patterns and relations emerge can be supported by many of the visualisation techniques described in this paper. An ideal MIR system needs to support both algorithmic approaches as well as interactive possibilities that extend our human abilities [5]. While different forms of representation, algorithms, heuristics, neural networks, etc., can assist us in automating categorisation, feature extraction, comparison, differentiation, etc., the fact still remains that music is best perceived and understood through listening. Sonic browsing techniques can also be used for example when investigating copyright issues, searching for new repertoire, or, for pure enjoyment.

8. ACKNOWLEDGMENTS

Our thanks to Prof. Mícheál Ó Súilleabháin and Paul & Colm McGettrick of the Irish World Music Centre at the University of Limerick for giving us access to the "A Key to Sources of Irish Traditional Music" (also known as the Fleischmann Collection).

9. REFERENCES

[1] Arons, B. (1992). "A Review of the Cocktail Party Effect." Journal of the American Voice I/O Society 12: 35-50.

[2] Fernström, J. M. and L. J. Bannon (1997). Explorations in Sonic Browsing. BCS HCI '97, Bristol, UK, Springer Verlag, London.

[3] Fernström, J. M. and C. McNamara (1998). After Direct Manipulation - Direct Sonification. ICAD '98, Glasgow, Scotland, Springer-Verlag.

[4] Marchionini, G. and B. Shneiderman (1988). "Finding facts versus browsing knowledge in hypertext systems." IEEE Computer 19: 70-80.

[5] Wegner, P. (1997) Why Interaction is More Powerful than Algorithms, Communications of the ACM, May 1997, Vol. 40, 5, pp.80-92

[6] Bonardi, A. (2000) IR for Contemporary Music: What the Muscologist Needs, Proceedings of the Int. Symposium on Music Information Retrieval, Oct 23-25 2000, Plymouth, MA.

[7] Ó Maidín, D and M. Fernström (2000) The Best of Two Worlds: Retrieving and Browsing, Proceedings of the COST G-6 Conference on Digital Audio Effects (DAFX-00), Verona, Italy, December 7-9, 2000.

MIRACLE: A Music Information Retrieval System with Clustered Computing Engines[1]

J.-S. Roger Jang, Jiang-Chun Chen, Ming-Yang Kao

Multimedia Information Retrieval Laboratory
Computer Science Department, National Tsing Hua University, Taiwan
{jang, jtchen, gao}@cs.nthu.edu.tw

Keywords
Content-based Music Retrieval, Audio Indexing and Retrieval, Dynamic Programming, Dynamic Time Warping, Query by Singing, Cluster Computing, Parallel & Distributed Computing

1. Introduction

This paper presents a music information retrieval system based on parallel and distributed computing. The system, called MIRACLE (Music Information Retrieval Acoustically with Clustered and paralleL Engine), can take a user's acoustic input (about 8 seconds) and perform similarity comparison on a group of clustered PCs. The system is a paradigm of client-server distributed computing since the pitch tracking is performed at the client computer while the time-consuming process of similarity comparison is performed at the server. Moreover, the similarity comparison is handled by a master server which partitions the comparison task into subtasks and dispatches these subtasks to a collection of slave servers. Currently there are more then 10,000 songs in MARACLE and the average retrieval time for "match beginning" is less than 1 seconds. The top-10 recognition rate for "match beginning" is 72%, and for "match anywhere", 56%. Extensive tests and performance evaluation demonstrates that MIRACLE is a feasible system that suits common people's needs of music information retrieval.

2. Related Work of MIR Systems

As the needs for music information retrieval rises, there are many MIR systems reported in the literature, including QBH (Query by Humming) by Ghias et al. [1], MELDEX (Melody Indexing) by Bainbrideg et al. [6], SoundCompass by Kosugi et al. [5], Super MBox by Jang et al. [2], Themefinder by Kornstadt et al.[3], MELODISCOV by Rolland et al.[7], etc. However, most of the above systems do not allow acoustic input from users directly; therefore the usability of those systems is significantly limited. Even within those MIR systems based on acoustic input, only MELDEX and Super MBox (a precursor of MIRACLE) have web deployment, which allows general public access. Particularly, as far as we know, MIRACLE is the first MIR system that is based on cluster computing.

The authors have also published their work on a content-based MIR system called Super Mbox [2]. The focus of the publications is on the use of dynamic programming techniques for elastic match in the comparison engine. A significant advantage of using DTW is that users are not required to singing "ta" to facilitate note segmentation, as required by MELDEX and SoundCompss. Being a precursor of MIRACLE, Super MBox only allows the use of DTW on a single processor. MIRACLE, on the other hand, adopts a two-step hierarchical filtering method (HFM) that filters out 90% candidates using an efficient linear-scaling comparison, and then employs DTW to compare the survived 10% candidates.

3. Distributed and Parallel Computing

MIRACLE is composed of a client and a server component. The client component takes users' acoustic input and transforms it into a pitch vector. The resulted pitch vector is then send to the server for similarity comparison. At the server side, the request is first handled by a master server which partitions the whole song list into partial lists, and then dispatches these partial lists to 18 client PC servers (ranging from Pentium 166 MHz to 1 GHz). Once a slave server receives its candidate list from the master server, it starts similarity comparison and return top-20 most likely candidate songs to the master server. The master server then combines and reorders the top-20 lists from all slave servers to generate the overall top-20 ranking list.

The initial length of the comparison song list for slave server p, denoted by l_p, is proportional to the clock rate of the slave server, namely, $l_p = l * \frac{C_p}{\sum_{k=1}^{18} C_k}$, where C_k is the clock rate of slave server k and l is the length of the total song list. To adaptively change the formula for l_p at request i, denoted by $l_p(i)$, can be expressed as $l_p(i+1) = l_p(i) * \exp\left\{\frac{t_{ave}(i) - t_p(i)}{k}\right\}$, where $t_p(i)$ is the response time of slave server p at request i; $t_{ave}(i) = \frac{1}{18}\sum_{j=1}^{18} t_j(i)$ is the average response time over all slave servers at request i; and k is a constant used to control the sensitivity of the adaptation. The goal of the adaptive load balancing strategy is to eventually have a same response time for each slave server. In fact, the above expression for $l_p(i+1)$ is not the final value for our experiment since $l_p(i+1)$ must fulfill the

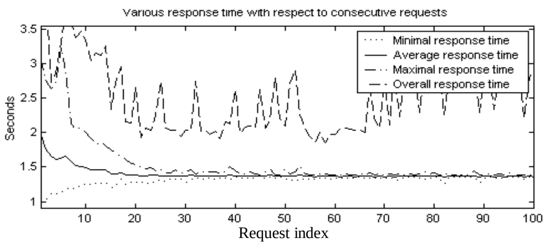
[1] This is an on-going project supported by Cweb Inc. (http://www.4music.com.tw) and the Excellency Project sponsored by the National Science Council at Taiwan, where the authors express many thanks.

constraint that the total length should be equal to l. Hence the final value of $l_p(i+1)$ is $l_p(i+1) = l * \frac{l_p(i+1)}{\sum_{j=1}^{18} l_j(i+1)}$.

4. Performance and Discussions

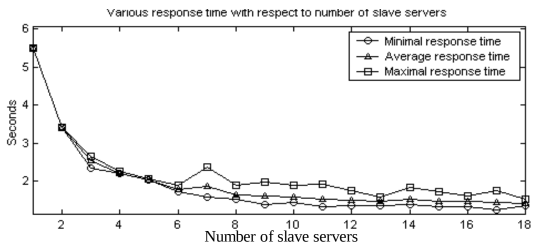
To test the recognition rate of MIRACLE, we have a large collection of 1550 vocal clips from 20 persons. 1054 of the vocal clips are from the beginning of songs, while the other 496 are from the middle. Some of the recordings are obtained via regular handsets or mobile phones to test the robustness of the pitch tracking algorithm. For the case of "match beginning", we sent the 1054 files to MIRACLE that employs two-step HFM as the comparison procedure. The average response time is about 2.29 seconds. The top-3 recognition rate is 65.75%; top-10 is 70.68%. If we choose DTW instead of two-step HFM, the top-3 recognition rate is 65.56% and top-10 72.58%. It is obvious that DTW and two-step HFM have comparable recognition rates. However, DTW's average search time is about 5 seconds, which is much longer than that of two-step HFM. For "match anywhere", we sent the 496 vocal clips to the master server and the average response time is about 5 seconds. The top-3 recognition rate is 43.29%; top-10 is 49.42%.

To test the adaptive strategy for load balancing, we measured various response time for 100 consecutive requests of “match anywhere” sent to the master server. The plot of various response time with respective to request indices can be shown in the following figure:



Obviously our adaptive strategy can effectively balance the loads such that the response time of each slave server approaches the same. Since the slave servers are not dedicated to MIRACLE only, we can see some sudden increases in the slowest response time.

The following plot shows the curves of various response time (after 100 requests, and taking the average of the last 10 requests) with respect to number of slave servers:



From the above plot, we can observe that the average response time levels off when the number of slave servers is 10 or more. We can conclude that for a MIR system with 10,000 songs, 10 clustered PCs are qualified for the requests of “match anywhere”.

To test drive MIRACLE, please follow the link of “Online demo of Super Mbox” at the author’s homepage at:

http://www.cs.nthu.edu.tw/~jang

5. References

[1] Ghias, A. J. and Logan, D. Chamberlain, B. C. Smith, “Query by humming-musical information retrieval in an audio database”, ACM Multimedia ’95 San Francisco, 1995. (http://www2.cs.cornell.edu/zeno/Papers/humming/humming.html)

[2] Jang, J.-S. Roger and Gao, Ming-Yang "A Query-by-Singing System based on Dynamic Programming", International Workshop on Intelligent Systms Resolutions (the 8th Bellman Continuum), PP. 85-89, Hsinchu, Taiwan, Dec 2000.

[3] Kornstadt, Andreas. "Themefinder: A Web-based Melodic Search Tool," *Melodic Comparison: Concepts, Procedures, and Applications, Computing in Musicology* 11 (1998), 231-236.

[4] Kosugi, N. Y., Kon'ya, Nishihara, S., Yamamura, M. and Kushima, K. "Music Retrieval by Humming – Using Similarity Retrieval over High Dimensional Feature Vector Space," pp 404-407, IEEE 1999.

[5] Kosugi, N., Nishihara, Y., Sakata, T., Yamamuro, M., and Kushima, K., "A practical query-by-humming system for a large music database," In Proc. ACM Multimedia 2000, November 2000.

[6] McNab, R. J., Smith, L. A. and Witten, Jan H. “Towards the Digital Music Library: Tune Retrieval from Acoustic Input” ACM, 1996.

[7] Rolland, P.-Y. “Discovering patterns in musical sequences,” Journal of New Music Research, vol. 28, no. 4, pp 334-350, 1999.

A Three-Layer Approach for Music Retrieval in Large Databases

Kjell Lemström
Department of Computer Science, University of Helsinki
PO Box 26, FIN-00014
University of Helsinki, Finland
klemstro@cs.Helsinki.Fi

Geraint A. Wiggins
Department of Computing, City University, London
Northampton Square, London
EC1V 0HB, England
geraint@soi.city.ac.uk

David Meredith
Department of Computing, City University, London
Northampton Square, London
EC1V 0HB, England
dave@city.ac.uk

ABSTRACT
An effective music information retrieval (MIR) system should provide fast queries to music databases taking into account musical features relevant to the task, such as *transposition invariance*, *polyphony* of music and the fact that there might be some 'extra intervening musical elements' (such as grace notes) within the database occurrence of the query pattern. The importance of efficiency is due to the need to find musical documents in possibly enormous databases. In this paper, we introduce an approach using three different matching layers each of which is possible to find transposition invariant occurrences of given musical query patterns in polyphonic music databases. The advantage of the layers is in sorting them in an order of decreasing efficiency as regards to the speed, but in an increasing order of carefulness put in the searching process. Thus, a repeatedly occurring musical pattern used as a query pattern should be found very efficiently, while searching for a query pattern corresponding to a rare database occurrence with arbitrarily many extra intervening elements is allowed to take more time.
Key words: music retrieval, multi-layer approach, polyphonic databases.

1. INTRODUCTION
The rapid growth of Internet-based services, multimedia technology, and development of new standards, such as MPEG-7, have made content-based retrieval and analysis of multimedia information an increasingly important field of research. Traditionally, the research has focused on the indexing and retrieval of texts and images. Nevertheless, music is at least as important a part of our culture and, consequently, as important a constituent of the multimedia domain, which can be seen in the current effort being put into developing theories and practical methods for its retrieval for use in, for example, Internet search and digital libraries. Content-based retrieval of music requires specific techniques that have not been employed for other media.

In this paper, we suggest a multi-layer approach for music information retrieval (MIR) in large-scale databases. Our approach contains three different matching layers, each of which is capable of finding transposition invariant occurrences of a given query pattern within a polyphonic database. The matching layers are executed in order of decreasing efficiency. The same ordering, however, puts the layers in order of increasing number of possibilities inspected. Having executed one layer, the algorithm of that layer outputs the occurrences it has found (if any), after which it is down to the user whether the next matching layer is to be executed.

A specific property of two of the layers (the first and the third) is that in a constituent of a musical pattern there may appear any finite number of other events between any two events included in the query pattern. Thus, an occurrence could be found even if the database version had, for example, some kind of musical decoration (such as different ornamentations or grace notes) that is absent in the query pattern. Next we briefly describe the three layers one by one.

2. DESCRIPTION OF THE LAYERS
The First Layer
The first matching layer is based on indexing. A particularly time-efficient indexing technique, based on *suffix structures*, for the matching can be performed in linear time with respect to the length of the query pattern. There are several ways to implement a suffix structure. An attractive choice is the so-called *suffix tree* because its space complexity is linear in the size of the database. Moreover, it can be constructed in linear time [7]. However, a suffix tree storing all the MIDI files available on the Internet would require an impossible amount of main memory [4]. Because of this problem, some MIR researchers have started to consider the possibility of finding *musically meaningful* patterns in the musical documents residing in the database. One of the first ideas was to extract only the melodies, or only the melodies of the choruses of the musical documents, these being the parts of musical documents most frequently used in content based queries [1].

Another current idea for indexing music [3] is to use a multiple viewpoint system [2]. The idea is to put in the structure such subsequences from any of the considered viewpoints whose frequency, in practice, exceeds significantly the expected frequency. Both of the methods above, as well as all the others previously suggested for MIR indexing, have been alike in that they have been based on sequences of musical events, and the sequences to be considered have been fixed beforehand.

Our current pattern induction algorithm, SIATEC (Structure Induction Algorithm with Translational Equivalence Classes) [6], can be used to find the frequently occurring musical patterns. SIATEC works in two phases. The first phase, SIA, computes every maximal repeated pattern in the given music database. The second phase takes the output of SIA as input and then computes all the occurrences of each of the maximal repeated patterns computed by SIA. The patterns to be inserted in the indexing structure are selected out of these occurrences of maximal repeated patterns, and can be seen as the *Longest Common Transposition Invariant Subsequences* (or LCTS) [4] of subsequences appearing in the database. Thus, we are able to find in this most efficient first phase patterns that are musically meaningful (since they are those that recur in the database) even if they are decorated unexpectedly. We cannot afford, however, to index all such recurring patterns of a large-scale database, because of the huge amount of main memory required for the indexing structure. Therefore, the following matching layers would be needed in many situations.

The Second Layer

If the pattern being sought cannot be found in the index, the second layer of our approach can be invoked. This layer applies the fast bit parallel MonoPoly filtering algorithm [5]. With most patterns (any pattern whose length is shorter than the size of one computer word), the main phase scans through the database in linear time (with respect to the length of the database). The main phase reports possible locations for occurrences, which are to be checked with another, somewhat slower algorithm.

The advantage of this layer is that it does not need an excessively large main memory to be able to search the occurrences of the pattern anywhere in the database. Furthermore, even though it is not as efficient as the previous layer, it is still very fast compared to the conventional dynamic programming algorithm computing edit distance, frequently applied to MIR (see e.g. [4] for a summary of some current methods). The found occurrences, however, for this layer are always sequential. Therefore we have one further layer.

The Third Layer

The final third layer applies the SIA(M)ESE matching algorithm based on the SIA algorithm [6]. This layer is the slowest of the three, but it allows a broader definition of what counts as a possible match than the two previous layers. More precisely, it does the same kind of LCTS matching as the first layer, but because it does not need the indexing structure, it is not restricted to matching against frequently occurring patterns only. Moreover, SIA(M)ESE is capable of *multi-resolution searching*, *i.e.*, the matching can be done on any desired level of detail (*cf. e.g.*, Schenkerian analyses). This is obtained by definition without any extra cost on the performance; the resolution of the search is defined by the details of the user-given query pattern.

3. CONCLUDING REMARKS

In this paper, we have introduced a three-layer approach to MIR, which is capable of finding transposition invariant occurrences of a given query pattern within large-scale polyphonic music databases. The occurrences of the pattern can be found even if the database version has, for example, different musical decorations than the query pattern. The contribution of our approach is in having three distinct searching algorithms each of which are efficient and effective already in themselves, but that work particularly well together when sorted in an order of decreasing performance and in an increasing order of thoroghness and detail in the searching process. In this way, a repeatedly occurring pattern can be found very efficiently, whereas finding another query pattern corresponding to a rare database occurrence with possibly many intervening elements takes more time, but is still possible.

Denoting the number of musical events in the query pattern and music database by m and n, respectively, and the number of chords in the database and the maximum size of a chord by $\overline{n}$ and q, the running times of the different phases of our multi-layer algorithm are as follows. Before any query execution, a preprocessing phase is required; the indexing structure for the first layer is constructed in $O(n^2 \log n)$ time, and the structures required for the second layer are built in $O(\overline{n}q)$ time. The three matching layers are executable in $O(m)$, $O(\overline{n})$ ($O(\overline{n}q(q+m))$ for the checking phase), and $O(mn\log(mn))$ times, respectively, the second requiring the restriction that the length of the query pattern is no more than the size of the computer word in bits.

4. REFERENCES

[1] D. Bainbridge. Private Communication. October 2000.

[2] D. Conklin and I. Witten. Multiple viewpoint systems for music prediction. *Journal of New Music Research* 24 (1995), pp. 51–73.

[3] D. Conklin and C. Anagnostopolou. Representation and discovery of multiple viewpoint patterns. In *Proceedings of the 2001 International Computer Music Conference*, Havana, Cuba, October 2001.

[4] K. Lemström. *String Matching Techniques for Music Retrieval.* PhD thesis, University of Helsinki, Faculty of Science, Department of Computer Science, 2000. Report A-2000-4.

[5] K. Lemström and J. Tarhio. Detecting monophonic patterns within polyphonic sources. In *Content-Based Multimedia Information Access Conference Proceedings (RIAO'2000)*, pages 1261–1279, Paris, 2000.

[6] D. Meredith, G.A. Wiggins and K. Lemström. Pattern induction and matching in polyphonic music and other multidimensional datasets. In *Proceedings of the Fifth World Multiconference on Systemics Cybernetics and Informatics (SCI2001)*, pages 61–66 (vol X), Orlando, FL, 2001.

[7] E. Ukkonen. On-line construction of suffix-trees. *Algorithmica* 14 (1995), pp. 249–260.

Gamera: A Structured Document Recognition Application Development Environment

Karl MacMillan, Michael Droettboom, and Ichiro Fujinaga

Peabody Conservatory of Music
Johns Hopkins University
1 East Mount Vernon Place, Baltimore MD 21202
email: {karlmac,mdboom,ich}@peabody.jhu.edu

ABSTRACT

This paper presents a new toolkit for the creation of customized structured document recognition applications by expert users. This open-source system, called Gamera, allows a user, with particular knowledge of the documents to be recognized, to combine image processing and recognition tools in an easy to use, interactive, graphical scripting environment. Additionally, the system can be extended through a C++ module system.

1. INTRODUCTION

This paper[1] presents a new toolkit for the creation of domain-specific structured document recognition applications by expert users. This system, called Gamera, allows a knowledgeable user to combine image processing and recognition tools in an easy to use, interactive, graphical scripting environment. The applications created by the user are suitable for use in a large-scale digitization project; they can be run in a batch processing mode and easily integrated into a digitization framework. Additionally, a module (plug-in) system allows experienced programmers to extend the system. This paper will give an overview of Gamera, describe the user environment, and briefly discuss the plug-in system.

2. MOTIVATION AND GOALS

Gamera is being created as part of the Lester S. Levy Sheet Music Project (Phase Two) (Choudhury et al. 2001). The Levy collection represents one of the largest collections of sheet music available online. The goal of the Levy Project is to create an efficient workflow management system to reduce the cost and complexity of converting large, existing collections to digital form. From the beginning of the project, optical music recognition (OMR) software was a key component of the workflow system. The creation of a flexible OMR tool is necessary because of the historical nature of the Levy collection; existing OMR systems are not designed to handle the wide range of music notation found in the collection or deal with the potentially degraded documents. OMR alone is not sufficient for the complete recognition of the scores in the Levy collection as they are not comprised only of musical symbols. Text is also present as lyrics, score markings, and metadata. It was hoped, however, that an existing optical character recognition (OCR) system would be able to process such text. Early trials of existing systems revealed there are many problems with the current generation of OCR software within this context.

[1] Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page.

To address the need for OCR in the Levy project the Gamera system was created. Gamera is an extension of the existing OMR system to a general symbol recognition system. By creating a general symbol recognition system it is possible to use the same technology that allows the OMR system to perform well on the musical portions of the Levy collection to recognize the text. In addition to serving the needs of the Levy project, we hope that the system may be used in the future for the recognition of historical documents and any other structured documents that current recognition systems do not adequately address.

In addition to generalizing the system, a graphical programming environment has been added to ease the adaptation of the system by users with expert knowledge of the documents to be recognized. This environment provides an easy-to-learn scripting language coupled with a graphical user interface. The goal is to allow the user to experiment easily with algorithms and recognition strategies during the creation of custom scripts for the recognition process. This will allow users to leverage their knowledge of the documents to customize the recognition process. It is hoped that users without extensive computer experience can effectively use this environment with a small amount of training. The scripting environment does contain, however, a complete, modern programming language that will allow advanced users considerable flexibility and power.

3. ARCHITECTURE OVERVIEW

Gamera is primarily a toolkit for the creation of applications by knowledgeable users. It is composed of modules (plug-ins), written in a combination of C++ and Python, that are combined in a very high-level scripting environment to form an application. The overall design is inspired by systems like MathWorks Matlab, CVIP tools (Umbaugh 1998), or spreadsheet macros. In Gamera, modules perform one of five document recognition tasks:

1. Pre-processing
2. Document segmentation and analysis
3. Symbol segmentation and classification
4. Syntactical or semantic analysis
5. Output

Each of these tasks can be arbitrarily complex, involve multiple strategies or modules, or be removed entirely depending on the specific recognition problem. Additionally, keeping with the toolbox philosophy of Gamera, the user of the system has access to a range of tools that fall within the general category of these tasks. The actual steps that make up the recognition process are completely controlled by the user.

In addition to flexibility Gamera also has several other goals that are important to the Levy project and to large-scale digitization projects in general. These are:

1. A batch processing mode to allow many documents to be recognized without user intervention.
2. Open-source so that the software can be customized to interact well with the other parts of the digitization framework.
3. The system designed to run on a variety of operating systems including Unix, Microsoft Windows, and Apple MacOS.
4. Recognition confidence output so that collection managers can easily target documents that need correction or different recognition strategies.

The first three goals have been achieved while the last goal is currently being developed.

4. USER ENVIRONMENT

The goal of the user environment is to leverage the knowledge and skills of the user about the documents being recognized. This is accomplished by creating a dynamic scripting environment and graphical user interface where users can experiment with various Gamera modules.

4.1 Scripting Environment

Gamera includes a complete scripting environment that a user can use to create custom recognition systems quickly and easily. The scripting environment tries to be easy to use, flexible, and extensible.

4.1.1 Ease of Use

Perhaps the most important aspect of the Gamera scripting environment is ease of use by users with limited computer programming experience. As previously stated, the targeted user is a person with expert knowledge of the documents to be recognized that may or may not have computer programming experience. In order to meet this goal Python was chosen as the foundation and extensions were written that are as easy to use as possible.

In order to transform Python from a general purpose scripting language to scripting environment tailored to the needs of Gamera users, a set of extensions was written in a combination of Python and C++.

4.1.2 Flexibility

Flexibility is the second most important goal for the scripting environment. Again, this aspect of the scripting environment is facilitated by the choice of Python. Because Python is a general-purpose programming language, a large portion of the system can be implemented directly in standard Python. In general, only those algorithms that need direct access to image pixels are written in C++. This allows users to customize existing modules written in Python, combine the low-level building blocks into new modules, or write modules from scratch.

4.1.3 Extensibility

Despite the flexibility of the scripting environment, not all algorithms can be suitably implemented in Python. For this reason, a C++ module system for use by experienced programmers has been developed. Some of the features of this system are:

1. Automatic binding of C++ code to Python.
2. Runtime addition of C++ modules as methods to Python classes.
3. Abstraction of the data storage format of image data using C++ templates to allow convenient access to compressed images.
4. Flexible programming interface allows the easy conversion of existing C and C++ code that uses a variety access methods to image data.

4.2 Graphical Interface

In addition to the scripting environment Gamera includes a graphical user interface that allows the interactive display and manipulation of images using the scripting environment. This can be as simple as displaying the results of a pre-processing algorithm or as complex as complete interface for training. Again, like the scripting environment, the graphical interface is created with standard tools entirely in Python allowing users to extend and modify the system.

5. CONCLUSION

A graphical programming environment for the creation of document recognition applications was described. This system, called Gamera, is designed to be used by people with expert knowledge of the documents to be processed. These users are not required to have extensive computer experience; the system can be effectively used with a small amount of training. Users with considerable programming experience can also create custom modules in Python or C++ to extend the system. The applications created by this system are suitable for large-scale digitization projects because they can be run in batch mode and integrated into the digitization framework.

6. ACKNOWLEDGEMENTS

The second phase of the Levy Project is funded through the NSF's DLI-2 initiative (Award #9817430), an IMLS National Leadership Grant, and support from the Levy Family.

7. REFERENCES

Choudhury, G. S., T. DiLauro, M. Droettboom, I. Fujinaga, and K. MacMillan. 2001. *Strike up the score: Deriving searchable and playable digital formats from sheet music.* D-Lib Magazine 7 (2).

Umbaugh, S. E. 1998. *Computer vision and image processing: A practical approach using CVIPtools.* Upper Saddle River, NJ: Prentice Hall.

Melody Matching Directly From Audio

Dominic Mazzoni
Carnegie Mellon University
5000 Forbes Ave
Pittsburgh, PA 15213
(412) 268-3069
dmazzoni@cs.cmu.edu

Roger B. Dannenberg
Carnegie Mellon University
5000 Forbes Ave
Pittsburgh, PA 15213
(412) 268-3827
rbd@cs.cmu.edu

ABSTRACT

In this paper we explore a technique for content-based music retrieval using a continuous pitch contour derived from a recording of the audio query instead of a quantization of the query into discrete notes. Our system determines the pitch for each unit of time in the query and then uses a time-warping algorithm to match this string of pitches against songs in a database of MIDI files. This technique, while much slower at matching, is usually far more accurate than techniques based on discrete notes. It would be an ideal technique to use to provide the final ranking of candidate results produced by a faster but lest robust matching algorithm.

1. INTRODUCTION

Today, a musician who wishes to locate a particular song by melody can use a number of different search programs that allow one to input a few notes from the song (in any key), or even just the melodic contour [1-6]. Realistically, most people are not musically literate and are not capable of transcribing a melody they are hearing in their heads into normal music notation. Even identifying whether the next note in a sequence goes up, down, or stays the same, is beyond the capabilities of many potential users. That is the motivation behind creating an interface where the user only needs to hum the melody he or she would like to search for.

It is not sufficient to rely on a melody transcription algorithm to convert a digital recording of the hummed query into a sequence of notes to search for in a song. Common problems include regions where the pitch tracker cannot lock onto any frequency, octave errors, and segmentation errors (two consecutive notes mistranscribed as a long note or vice versa).

Instead we propose searching for a melody based on the best estimate of the continuous pitch contour derived directly from the audio recording. Speech recognition researchers have discovered time and time again that in the many steps necessary to go from a recording of speaking to the textual transcription, making hard decisions at any step can be disastrous. Guided by this experience, we try to eliminate the transcription steps that quantize pitches and segment them into discrete notes, as this process is certain to introduce errors.

This work is guided by the model of query-by-humming systems. In such a system, the user hums, sings, or whistles (we will refer to any of these simply as "humming"), and the system finds matching entries in a music database. An entry matches if it contains a close match to the hummed query. Since songs are generally considered to be equivalent when performed at a speed or in a different key, the system should be invariant with respect to transposition and tempo.

2. METHODOLOGY

Our idea for searching based on the pitch contour is very straightforward. First use a pitch transcription system to compute the continuous pitch contour of the hummed query. (We distinguish between pitch transcription systems, which simply attempt to determine the pitch being hummed at each point in time, and full melody transcription systems, which attempt to extract a discrete series of notes, each with its own pitch, onset time, and duration.) Overlay the pitch contour on top of every possible place in the song, for every possible pitch offset, and for a range of reasonable time scaling factors. For each position, offset, and time scale, approximate the integral of the difference between the instantaneous pitch at each point in time and the pitch of the song at that point, giving a simple distance measure between the two. The song that contains the minimum distance measure is the one that best matches the query. Because hummed queries are not likely to have a perfectly consistent tempo, we use a dynamic time warping algorithm to allow for small rhythmic differences.

This method is very computationally intensive, and even with heavy optimization it is not likely to be fast enough to be a complete melody-matching solution. However, note that pitch and rhythm are taken into account without relying on pitch quantization, beat induction, or note segmentation. We believe this contributes to the improved accuracy of this method.

Here are the details of our implementation. We segment the query and candidate melody into frames of 100 ms. (100 ms was chosen as a compromise between efficiency and accuracy.) Then we run the pitch transcription algorithm on each frame of the audio recording of the humming. The pitch transcription algorithm that we use is based on the *enhanced autocorrelation* algorithm described by Tolonen and Karjalainen [7]. We investigated many other pitch transcription programs, including spectral-based approaches, other autocorrelation methods, and commercial products, but found that choosing the peak of the enhanced autocorrelation signal worked as well if not better than anything else when the goal was simply to come up with one target pitch for each frame. We represented pitches as MIDI note numbers, allowing fractions, so for example 60.13 stands for a pitch 13 cents above middle C. Other details, such as pitch ranges for different singers and silence thresholds, can be obtained directly from our source code, which is freely available on the Internet [8].

The songs in our database are all MIDI files, so they also require some preprocessing before we perform our melody-matching algorithm. To compute a string of 100 ms pitch frames from a MIDI file, we consider each MIDI channel separately, and find the note that is most contained in each time frame. If multiple notes are found, we choose the one with the highest pitch. Also, because note releases seem to be much less important perceptual cues than note onsets, and because note releases are performed inconsistently, we extend all notes to the beginning of the next note, thereby eliminating rests in the melody. This mirrors the technique of defining a note's duration as the inter-onset time used in almost all note-based melody matching algorithms. Because the tempo of the query may not have exactly matched the tempo stored in the MIDI file, we repeat this process with different time scaling factors from 0.5 to 2.0, allowing for an opportunity to match a hummed melody from half the speed up to twice the speed.

At this point we have a string of n pitches for the query, so for every possible sequence of about $2n$ frames from every channel of our MIDI file, we match the query against the database clip using a dynamic programming-based time-warping algorithm, exactly the same as would be found in a limited-vocabulary speech recognition system. To limit the amount of rhythmic variation between the query and the song from the database, we use a *beam width* of $n/10$, ensuring that only paths that do not stray too far from the straight diagonal are allowed. Finally, we run this time warping algorithm 24 times, once for each possible quartertone offset.

3. EXPERIMENTAL RESULTS

In order to compare our approach against other techniques for melody retrieval, we collected a database of MIDI files in different genres and recordings of various people humming melodies from these MIDI files. We used the algorithm discussed in the previous section to compare the query to each song in our database and arrive at a distance between the query and each song. We then ranked the songs according to distance, smallest first, and looked at the rank of the intended song.

Our preliminary results were based on two small databases of MIDI files, one containing 77 big band swing songs, and one containing 18 Beatles songs. (For more recent results, see our website.) Our results were quite promising. Out of Beatles song queries, 9/11 times the correct song had a rank of one, and all 11 times the correct song appeared in the top three. Out of queries of big band songs, 13/20 times the correct song had a rank of one, and 16/20 times the correct song was in the top three. We also implemented a number of more traditional matching algorithms based on strings of discrete notes, and none of these performed as well, mostly because they returned a large number of false positives. The best note-based algorithm we implemented (which incorporated both pitch and rhythmic information) only got the correct song first 5/11 times for Beatles songs, and only got it in the top three 8/11 times. For big band songs, the note-based algorithm got the correct match first 5/20 times, and got it in the top three 6/20 times. This does not mean that it would not be possible for a better note-based algorithm to do much better, and in fact we are making our queries and our database available to any researchers who would like to try, but we feel that no approach of this form is likely to outperform our frame-based method unless there is a major breakthrough in melody transcription software.

4. CONCLUSIONS AND FUTURE WORK

Our frame-based approach shows a lot of promise. It works better than any note-based approach we were able to implement, and more importantly, there are compelling reasons why one would expect this approach to be more accurate.

In spite of these advantages, our approach is not perfect. One potential problem is that singers may change pitch in the middle of a query, and our approach does not currently deal with this as well as an interval-based algorithm. Perhaps the biggest criticism of our work is that it is clearly a brute-force approach and it is very slow. Rather than move from dynamic programming toward sub-linear retrieval algorithms suitable for large databases, we are advocating strings that are much longer than the number of notes. Our searches run orders of magnitude slower than typical note-based searches, and as a result, this algorithm could not be used by itself to drive a content-based music retrieval system.

Still, our approach could also be used behind the scenes to improve faster algorithms: when our frame-based algorithm fails, it is often because the query itself was not particularly good. Thus a researcher could use our more robust algorithm to distinguish between cases where the query was simply no good and cases where a prototype algorithm failed for a different reason.

In the future we would like to improve the speed by using two or more levels of refinement. We would begin with a fast but imprecise algorithm to narrow the search to a small subset of the database, then use successively more precise but more expensive algorithms to arrive at the final result. In addition, we would like to experiment with searching audio data instead of MIDI.

The authors would like to thank Kjell Lemstrom for answering questions about SEMEX and providing some valuable insight. This material is based upon work supported by NSF Award #0085945, an IBM Faculty Partnership Award, and an NSF Graduate Research Fellowship.

5. REFERENCES

[1] R.J. McNab, L. A. Smith, D. Bainbridge and I.H. Witten. The New Zealand Digital Library MELody inDEX (MELDEX). *D-Lib Magazine*, May 1997.

[2] A. Kornstädt. “Themefinder: A web-based melodic search tool.” *Computing in Musicology*, v11, pp. 231-36, 1998.

[3] D. Huron et. Al. *Themefinder* (website). http://www.themefinder.org/

[4] R. Typke. *Tuneserver* (website). http://wwwipd.ira.uka.de/tuneserver/

[5] K. Lemström. “String Matching Techniques for Music Retrieval.” Ph.D. thesis, University of Helsinki, Finland, Nov., 1999.

[6] K. Lemström and S. Perttu. “SEMEX – An Efficient Music Retrieval Prototype.” *Proceedings of the ISMIR 2000.*

[7] T. Tolonen, M. Karjalainen. “A computationally efficient multi-pitch analysis model.” *IEEE Transactions on Speech and Audio Processing*, Vol. 8, No. 6, Nov. 2000.

[8] CMU Music group website: http://www.cs.cmu.edu/~music/

Usage of the MELDEX Digital Music Library

John R. McPherson
Department of Computer Science
University of Waikato
Hamilton
New Zealand
jrm21@cs.waikato.ac.nz

David Bainbridge
Department of Computer Science
University of Waikato
Hamilton
New Zealand
davidb@cs.waikato.ac.nz

ABSTRACT

Online Digital Music Libraries are becoming increasing common and more sophisticated; however, availability of information on how users access and navigate these resources is limited. This type of information is crucial for improving user interface design and for providing users with better supported services.

Here we present an analysis of the logs for our digital music library, Meldex, for a 1 year period to discover patterns of usage.

1. INTRODUCTION

Our Melody Index [1] is part of the New Zealand Digital Library project (`nzdl.org`). Users can access songs in two ways: they can see the results of a query, or they can browse the song titles alphabetically. Queries can either be melodic or textual. Melodic queries are submitted by either uploading (posting) a short recording of sung or played notes, or by providing a Uniform Resource Locator (URL) to such a recording. Our demonstration page provides some sample recordings. Textual queries are matched against song metadata, such as title or author, and lyrics.

Songs are returned in a variety of different audio formats, such as WAV, MIDI, and Audio Interchange File Format (AIFF). Some collections can also have results returned as an image of the original sheet music. For example, our "Fake Book" collection is built from the results of running optical music recognition over sheet music. Copyright considerations restrict which collections return full-length audio files and images.

Our oldest collection is known as the "Folksong" collection. Based on the Essen and Digital Tradition databases, it consists of 9,354 folk songs which are divided into geographical regions (Chinese, German, Irish and North American). The "Fakebook" collection (mentioned above) consists of 1,235 songs. The "Midi" collection is built from 9,763 MIDI files sourced from the Web, and supports textual and melodic querying. The "MidiMax" collection indexes 17,799 MIDI songs and is more sophisticated, also allowing the indexing and retrieval of motifs. It has been available since October 2000, while the other collections have been online since November 1999.

2. A SELECTION OF STATISTICS

In reviewing prior work for usage analysis, the Variations music library [2] at Indiana University is notable for providing daily statistics online. Given the context of the Variations project, these focus on aggregate performance-oriented statistics such as number of songs retrieved, and maximum, minimum and average retrieval times.

Here we present an analysis of the usage logs of our digital library service for the 12 month period 1 April, 2000 to 31 March, 2001. Most of the results given here that are not for the whole library are for the Folksong collection, as this data set reflects patterns observed across the other collections.

Figure 1 shows the number of daily hits received (the line represents a rolling 7-day average). There is not a noticeable trend here, although there is a drop-off over the Christmas and New Year holidays. There are also several brief periods of server outages.

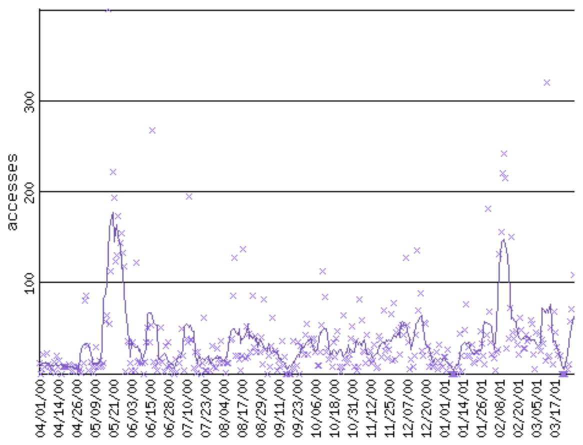


Figure 1: Daily accesses for the folksong collection

Table 1 shows the distribution of visitors to the folksong collection. This is based on all the web pages generated by

Table 1: Top 10 visitor domains for 'folksong'

Domain	Accesses	%age of total
.net	3,827	29.67
.com	2,128	16.50
Europe	2,102	16.30
unknown	2,090	16.20
.edu	1,001	7.76
Sth. Pacific	661	5.12
Asia	435	3.37
Nth. America	235	1.82
Australia	188	1.46
Sth. America	73	0.57
Totals:	12740	98.77

Table 2: Results pages generated — All collections

Page type	Number
Own audio file with text query	104
Own audio file only	588
Demo audio file with text query	105
Demo audio file only	89
Text query only	1539
Browse titles	1070
Total:	3495

the music library and includes help and query pages, for example, in addition to requests for songs from the collection.

Around 2000 of the hits for the folksong collection are from one site, which appears to have been crawling part of our library website. This accounts for slightly over half of the visits from the `.net` top-level domain, and also accounts for the two spikes observed in Figure 1. That particular internet address has been filtered from the remaining statistics given here. In addition, the addresses used by Meldex's principal researchers over this period have been filtered out of all statistics in this report.

Assuming that any accesses from the same IP address with less than five minutes of separation are part of the same "visit", the average amount of time spent per visit over all Meldex collections was slightly over 2 minutes, and consisted of an average of 3.4 page views. Visits came from just over 1,900 different internet addresses.

Table 2 shows how users get to song listings. Just under 70% of song listings are generated as a result of a query, either audio or textual (or possibly both), with the remainder generated as alphabetical listings of titles.

Table 3: Users' preferred Audio file format

Audio Format	%age
MIDI	48.5 (834)
WAV	23.8 (410)
Real Audio	17.2 (295)
AIFF	04.0 (69)
Soundblaster VOC	02.6 (45)
Sun u-law	02.3 (39)
Sun AU	01.6 (28)
Total	100 (1720)

Three of the available file formats account for nearly 90% of users' preference settings, with the MIDI format accounting for nearly half. These settings are listed in Table 3.

Table 4: Folksong Hit Parade - Top 10

Accesses	Name
80	"Auld Lang Syne" [from demo page]
72	"Aéire cinn bó rúin"
62	"The Ash Grove" [from demo page]
52	"Abdul Abulbul Ameer"
49	"Ai erwa"
36	"Three Blind Mice" [from demo page]
30	"A New England Ballad"
25	"Abilene"
25	"A-Beggin' I Will Go"
22	"Adam and Eve"

Table 4 gives the titles of the 10 most frequently requested songs for the folksong collection. Of the 9,354 songs indexed in this collection, 2,395 (25.6%) have been accessed at least once, and about 1,700 have been accessed exactly once, suggesting that these downloads are the results of users' individual queries.

3. SUMMARY

Around thirty percent of song lists generated are alphabetical title listings, and most accesses from these lists are for songs that start with the letter 'A'. We conjecture this is because new users to the library have a strong desire to discover what sort of music is contained in the collection, and accessing songs by titles is the easiest route currently available in the interface.

A result that took us initially by surprise is that forty-four percent of all listings are the result of a text query alone. While it is possible that a wide range of Web users are conditioned to typing queries into a text box, it should not be overlooked that the overhead of entering a music query in our current interface might be too high for many users. Also, analysis of our own group members has shown that for large MIDI collections, a browsing habit that had formed was to enter a text query on some vague topic (for example, "fire") and see which tunes popped up.

Our Meldex service is available at `www.nzdl.org/musiclib`.

4. REFERENCES

[1] Rodger J. McNab, Lloyd A. Smith, David Bainbridge, and Ian H. Witten. The New Zealand Digital Library MELody inDEX. *D-Lib Magazine*, May 1997.

[2] Jon W. Dunn and Constance A. Mayer. Variations: A digital music library system at Indiana University. In *Proceedings of the Fourth ACM Conference on Digital Libraries*, Berkeley, California, 1999. ACM.

Addressing the *Same but different - different but similar* problem in automatic music classification

Unjung Nam
CCRMA
Stanford University
Stanford, CA 94305
1 650 723 4971
unjung@ccrma.stanford.edu

Jonathan Berger
CCRMA
Stanford University
Stanford, CA 94305
1 650 723 4971
brg@ccrma.stanford.edu

ABSTRACT
We present a hybrid method in which we classify music from a raw audio signal according to their spectral features, while maintaining the ability to assess similarities between any two pieces in the set of analyzed works. First we segment the audio file into discrete windows and create a vector of triplets respectively describing the spectral centroid, the short-time energy function, and the short-time average zero-crossing rates of each window. In the training phase these vectors are averaged and charted in three-dimensional space using k-means clustering. In the test phase each vector of the analyzed piece is considered in terms of its proximity to the graphed vectors in the training set using k-Nearest Neighbor method. For the second phase we apply Foote's (1999) similarity matrix to retrieve the similar content of the music structures between two members in the database.

1. ANALYSIS METHODS

1.1 Spectral Centroid

The spectral centroid is commonly associated with the measure of the brightness of a sound. The individual centroid of a spectral frame is defined as (here, $F[k]$ is the amplitude corresponding to bin k in DFT spectrum..)

$$Spectral\ \ Centroid = \frac{\sum_{k=1}^{N} kF[k]}{\sum_{k=1}^{N} F[k]}$$

Figure 1 presents the weighted average spectral centroids of the two analyzed sound examples. The lower (magenta) band is an excerpt of the Kremlin Symphony's recording of Mozart's Symphony 25 (K. 183) and the upper (cyan) band is a rock style arrangement of the same musical segment. The high frequency components in the pervasively percussive rock version accounts for its higher placement on the graph.

1.2 Short-Time Energy Function

The short-time energy function of an audio signal is defined as: (where $x(m)$ is the discrete time audio signal, n is time index of the short-time energy, and $w(m)$ is a rectangular window.)

$$En = \frac{1}{N}\sum_{m}[x(m)w(n-m)]^2 \qquad w(x) = \begin{cases}1, & 0 \le x \le N-1,\\ 0, & \text{otherwise.}\end{cases}$$

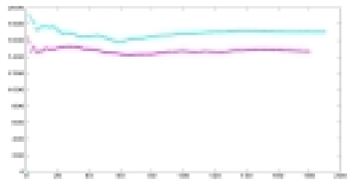


Figure 1.

It provides a convenient representation of amplitude variation over time. Patterns of change over time suggest the rhythmic and periodic nature of the analyzed sound. Figure 2 is the short-time energy change of the same excerpts. The highly fluctuating rock version (cyan) resulting from the persistent drum beats compared to the more subdued but highly contrasting symphonic version suggests one possible determinant for genre classification.

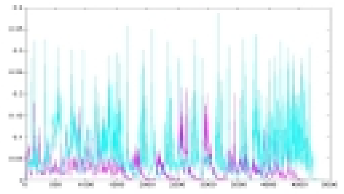


Figure 2.

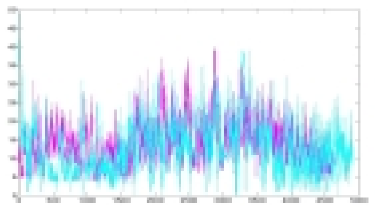
1.3 Short-Time Average Zero-Crossing Rate

In the context of discrete-time signals, a zero crossing is said to occur if successive samples have different signs. The short-time averaged zero-crossing rate (ZCR) is defined as

$$Zn = \frac{1}{2}\sum_{m}\left|\operatorname{sgn}[x(m)] - \operatorname{sgn}[x(m-1)]\right| w(n-m),$$
$$where\ \ \operatorname{sgn}[x(n)] = \begin{cases}1, & x(n) \ge 0,\\ -1, & x(n) < 0,\end{cases}$$

Figure 3 is the ZCR over time of the same two sound examples, as before, the classical version is magenta and the rock version is cyan. Compared to that of speech signals, the ZCR curve of music has much lower variance and average amplitude and when averaged, shows significantly more stability over time. ZCR curves of music generally have an irregular small range of amplitudes.

Short-time
ZCR



Time (samples)

Figure 3.

1.4 Foote's Similarity Method

Foote (1999) represents acoustic similarity between any two instants of an audio recording in a 2D representation, Figure 4 shows the 'similarity matrix' analyzed for the two music samples. The parameterization was done with a Mel-frequency cepstral coefficient function with frame size 30. Both samples are about 16 seconds long and sampled at 11025hz, 16 bits. The analysis visualizes the tripartite segmentation of the phrase in the 16 second excerpt (seconds 1-5, 5-12, and 12-16) in both the classical version (fig 4.1) and the classical version (fig 4.2). Despite the stylistic disparity between the two examples the musical similarity in terms of pitch and rhythmic structure is well represented.

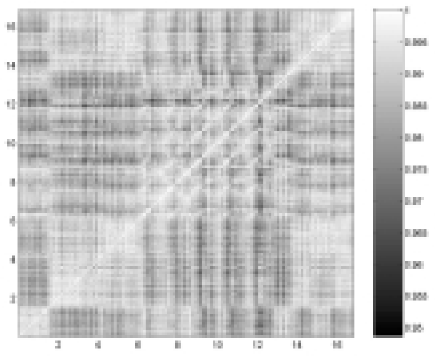


Figure 4.1. orchestral version.

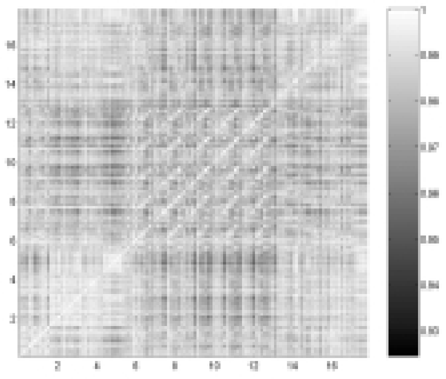


Figure 4.2. rock version

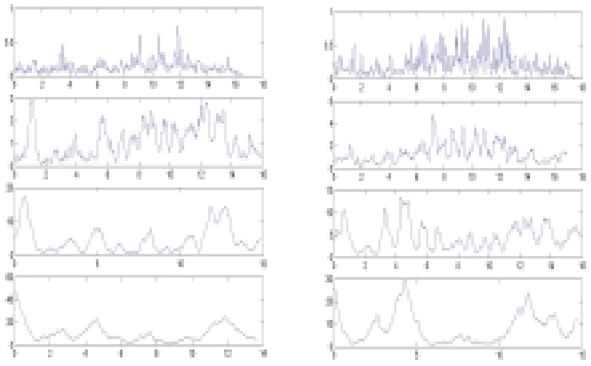


Figure 5. Novelty scores of
orchestral (left) and rock (right) version

Figure 5 presents the novelty scores over time(second) of the two examples. In each figure the outputs with kernel sizes, from top down, 10, 20, 60 and 96. The graph of kernel size 96 displays three high peaks corresponding to the tripartite musical structure. The smaller the kernel size the greater the detail represented. This facilitates detection of discrete musical events. We are currently considering heuristics to find optimal kernel sizes to track appropriate novelty information.

2.CONCLUSION

In this paper we explored a computational model that combines classification and comparison of raw audio signals to explore the perceived similarity between musical recordings. Foote's (1999) similarity matrix retrieved the similar content of the music structures between two music samples even though their spectral components are different. Future research will focus on quantitative measurement of the degree of musical similarity between two works, as well as genre classification by statistical clustering.

3. ACKNOWLEDGMENTS

Our thanks to Professor Julius O. Smith, Jonathan Foote and Malcolm Slaney for their insights and assistance.

4. REFERENCES

Foote, J. (1999) "Visualizing Music and Audio using Self-Similarity." In Proceedings of ACM on Multimedia.

Scheirer, E. and Malcolm Slaney. (1997) "Construction and Evaluation of A Robust Multifeature Speech/Music Discriminator." In Proceedings of IEEE ICASSP.

Musical Information Retrieval for Delta and Neumatic Systems

D. Politis, P. Linardis
Department of Informatics
Aristotle University of Thessaloniki
Thessaloniki 540 06
GREECE
+ 30 31 998406
{dpolitis, linardis}@csd.auth.gr
http://www.csd.auth.gr

H. Kotopoulos, A. Alygizakis
Department of Music Science & Art
University of Macedonia
156 Egnatia Str., Thessaloniki 540 06
GREECE
+ 30 31 891399
koxri@compulink.gr http://www.uom.gr

ABSTRACT

In this paper an alternate to Western Music musical system is presented. This system has flourished for more than 15 centuries in the areas of Byzantine Empire and it implements Neumatic and Delta Interfaces in order to represent musical structures. Recently, a remarkable revival and propagation of this system has been recorded worldwide. The motivation for this paper has been given from a joint effort of the Department of Informatics at the Aristotle University and the Department of Music Science and Art at the University of Macedonia to register the musical content not only of contemporary manuscripts but also to record and correlate morphologically the evolutionary stages from the neumatic origin to the final Delta Analytical method.

KEYWORDS

Alternate Musical Interfaces, Neumatic and Delta Systems, Byzantine Music, Morphology, Information Retrieval.

1. INTRODUCTION: DELTA AND NEUMATIC MUSIC NOTATIONS

The world of music is not uniform nor unified; it consists of various segmented systems diversified on matters of scales, rhythms and transitional phenomena [1]. The Common Music Notation (CMN) scheme along with the MIDI specification are Western Music oriented. As a result, they are not able to clearly depict alternate musical forms and traditions. The methodology described in this paper implements an indexing scheme based on signatures characterizing the content they point to. Also, emphasis is given on content extraction mechanisms concerning the morphology of the melodies. The musical database selected for the application of this method is a Delta musical notation system

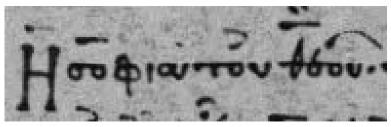


Figure 1. Byzantine Music Manuscipts. Detail of the so-called 'Chartres fragment' with musical notation, beginning of a sticheron in Mode 8, Monumenta Musicae Byzantinae, University of Copenhagen, Denmark.

known as Byzantine Music (Figure 1).

The problem with Delta symbols is that the same sequence of symbols may yield a different melodic content, depending on the scales of the Mode in which a melody is deployed [2] (Figure 2).

2. PROBLEM FORMULATION

The major issues in MIR for Byzantine music melodies are: (a) how to locate specific sequences of symbols (b) how to associate morphological metadata with the content.

Although the answer to this question may sound obvious, that by forming any melodically meaningful text databases [3] we can use IR systems available for Free Text Retrieval, things are not that simple. IR systems appropriate for this purpose are the *Inverted File* and the *Signature File* indexing schemes, both used extensively for indexing Free Text Databases.

The Signature File indexing schemes have a simple structure and require significantly less storage overhead. In Figure 3 is presented the structure of a signature indexing scheme, as modified here to handle a Musical Database. The Audio Data File is a collection of original melodic data blocks. These blocks may

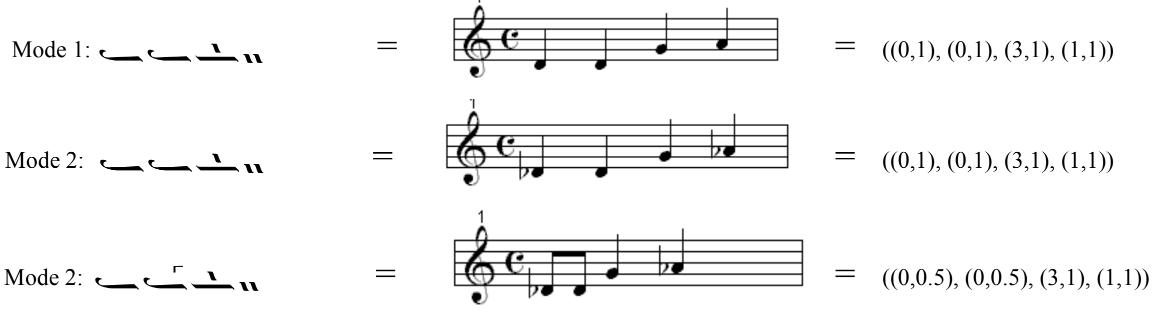


Figure 2. Variations of a motive in Modes 1 and 2 with D^4 serving as a melodic basis.

contain both semantic data and comments appended in the form of text [4].

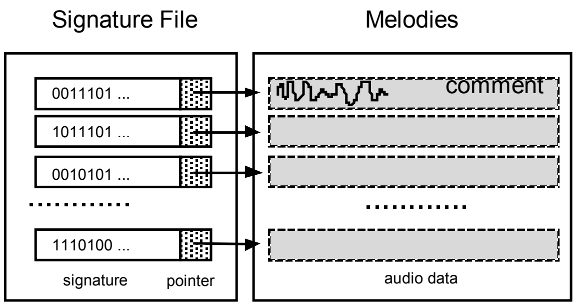


Figure 3. A Signature File scheme for digitally transcribed manuscripts.

In the Signature File are stored the signature records of the audio blocks and to each record is attached a pointer to the corresponding audio block. The Signature File and the Audio Data File may be kept and processed separately. The Signature File, which is of much smaller size, may be copied and distributed to many processors so that either many workers can take advantage of it or a form of parallel processing may be applied. In our case the signatures of a Delta musical file are attached to it as an appendix.

For a given query the Signature File is searched for **signature** records conforming to this query, then the pointers attached to these records are used to locate in the Audio Data File the corresponding audio block. It should be pointed out that queries scan for *motives* rather then isolated Delta notes which are meaningless by themselves (see again Figure 2).

In order to build a Database, an extension of the signature file method described in Figure 3 has been adopted which was originally presented for free text bases, the so-called *S-Index* scheme [5].

S-Index is a hybrid indexing scheme that combines many of the merits of Inverted File and Signature File schemes. Its performance is tunable between two extreme ends. At one end S-Index turns into a Signature File and at the other end it becomes an Inverted File. One advantage of the adopted indexing method is that frequently queried terms or certain user selected terms may be indexed via an Inverted File method, for speed, whereas the bulk of the terms may be indexed in the form of a tree of signature segments, which requires a lower storage overhead and also is more suitable for multiple term queries.
Since most RDBMS do not support direct use of binary variables or Boolean operators on binary variables, a table was created simulating the behavior of the proposed index. This architecture yields a binary tree of signature segments. Each node of this tree has the following structure:

```
SINDEXnn        (block_no       INTEGER,
                aa              INTEGER,
                node_no         INTEGER,
                sig             CHAR(k))
```

The value of parameter *k* depends on layers of the S-Index structure. For instance, for SINDEX4 it is $k=2^4=16$. Every table SINDEXnn (0 <= nn <= 14) records: (a) pointers to the audio file packages (b) pointers to the nodes of the internal tree structure and (c) the binary signature itself.
Apart from this RDBMS-centric methodology, the authors of this paper are seeking a method to encode digitally Byzantine melodies in a MIDI-like specification and to add to these files as accompanying metadata the *signatures* of each melody.

3. RESULTS

Following the analysis methodology described in the previous section, we have used Full Text Retrieval systems (BRS SIRSI and SQL Server 2000) along with custom made programs that calculate the probability of appearance for sequence 7 - 10 of symbols.
Some results are presented in Table 1. For the sake of simplicity, conditional probabilities for 3 symbol sequences are presented.
For these sequences digital signatures are built which accompany the digitally encoded manuscript.

Table 1. 3rd order stochastic sequences for the 3 more frequently appearing Delta symbols. P(So) is the probability for S_i as an initial symbol.

Symbol S	P(S)	P(So)	P(S1*S2*S3)
S1: ˎ	0,354	0	P(S1*S1*S3) 0,019
S2: —	0,149	0.05	P(S2*S1*S1) 0,0154
S3: ˋ—	0,146	0.55	P(S3*S1*S1) 0,018

Some results excluded from the statistical analysis tables:
- (a) motives are on average 9 Delta symbols long.
- (b) Non-terminating motive endings are declared by increased durations by one time unit for 95% of examined cases.
- (c) If intermediate (i.e. non terminating) segmentation takes place, motive length drops to 8 Delta symbols.
- (d) The terminating endings of a thesis are less than ten.

4. REFERENCES

[1] Lerdahl, F., Jackendoff, R., *A generative theory of tonal music*, MIT Press, Cambridege (Ma.), 1983.

[2] Spyridis, H.C. and D. Politis. 1990. “Information Theory Applied to the Structural Study of Byzantine Ecclesiastical Hymns.”*ACUSTICA*, 71(1): 41-49.

[3] Downie, J.S., “Music Retrieval as Text Retrieval: Simple Yet Effective”. *Proceedings of SIGIR'99*. Berkley, CA, USA. pp. 297-298.

[4] Jagadish H., Faloutsos C., Hybrid Index Organizations for Text Databases, *Proceedings of the Extending Database Technology Conference (EDBT)*, pp. 310-327, March 1992.

[5] Dervos D., Linardis P., Manolopoulos Y., S-Index: a Hybrid Structure for Text Retrieval, *Proc. First East-European Symposium on Advances in Databases and Information Systems* (ADBIS’97) St. Petersburg, September 1997.

Statistical Analysis in Music Information Retrieval

William Rand and William Birmingham
Electrical Engineering and Computer Science Department
The University of Michigan
1101 Beal Ave
Ann Arbor, MI 48109
1-734-763-1561

wrand@umich.edu, wpb@umich.edu

ABSTRACT
We introduce a statistical model of music that allows for the retrieval of music based upon an audio input. This model uses frequency counts of state transitions to index pieces of music. Several methods of comparing these index values to choose an appropriate match are examined. We describe how this model can serve as the basis for a query-by-humming system. The model is also shown to be robust to several kinds of errors.

1. INTRODUCTION AND RELATED WORK
Recently, researchers have developed systems that can retrieve a piece of music from a musical database using an aural query (e.g., a sung or hummed query). Of course, a query sounds very different from a full audio recording, which typically contains much more information. Moreover, the generally accepted wisdom is that users remember the major themes from pieces, and use these themes as their queries [1]. Thus, most music-retrieval systems cast the retrieval problem as matching an abstracted representation of each piece in their database, typically the major theme(s), against the query.

Some researchers approach the retrieval problem using string-matching techniques [2] [3] [4] [5] [6]. In these approaches, the theme and query are treated as a melodic contour and a string is derived from the alphabet S, U, D, (same, up, and down) for the interval change between notes. Some researchers, however, have used different mechanisms. In other related work, Brunelli and Messelodi examined different metrics of comparison when examining images [7]. Alamkan et al, used Markov models of music to generate new pieces in the style of those composed used to train the models [8].

Our work is inspired by Alamkan's work; we induce Markov models of themes in our database (all pieces are in MIDI). Similarly, we induce a Markov model for a query. We then use several statistic metrics to compare models to find the theme that most closely matches the query. In this paper, we overview our approach and provide summary experimental results.

2. THE MODEL
Music unfolds over time. Thus, we can model a piece of music as a series of states from an alphabet Σ. One statistic that we can calculate about these states is the frequency that state $\alpha \in \Sigma$ transitions to another state $\beta \in \Sigma$, without considering history [8]. Moreover, we have a function $\tau(\alpha,\beta)=\rho$, where ρ is the probability of transitioning from the state α to state β.

We use a state descriptor called the "concurrency." [8] This descriptor records all the notes that are sounding at any particular time, as well as the duration during which they are sounding. If we assume monophonic input and wrapping all notes into one pitch, this class yields 12 possible states for notes; based on our observations, we found that our duration space tends to be around a dozen different possible lengths. Thus, $|\Sigma|=12*12$ or $|\Sigma|=144$. In addition, we can totally order state space based on pitch classes and duration.

3. CORRELATION METRICS
We posit that the statistical model presented here captures some important elements of the style of a piece. Therefore, we should be able to compare τ's created from different musical pieces to determine style similarity. This gives the basis for a retrieval mechanism: we take a τ derived from a query and compare it to τ's induced from the database to determine their similarity. We assume that τ 's based on similar musical pieces will have higher correlations than those from disparate works. Here, we examine two measures for calculating similarity:

- *Correlation Coefficient* – This is the standard Pearson's correlation coefficient [10]. This technique indicates how well an entry in one matrix predicts the value of an entry in the other matrix.
- *Count Correlation* - We can observe the frequency of arriving in a state rather than examining the frequency of transitions. This is called a count array. We can then apply a correlation technique to the array. This procedure is not a true correlation measure, but can best be described as a weighted correlation, where the weights are the frequencies of state visitations. This measure makes a zero-order Markov assumption (our use of Pearson's makes a first-order assumption). In other words, the probability of transitioning to any state is the probability of being in that state. This technique is modified from a method originally developed by Alamkan [9].
- *Modified Scoring Metric* - We found through experimentation that both correlation measures work equally well. Moreover, it appears that the instances where they do not work are mainly disjoint. Thus, we combined measures to create a new measure, which takes the average of the scores from the correlation measures. We found that this combined measure has better results than either method alone.

We note that these techniques can be computed quickly, and thus searching is quite fast. The comparison between pieces is independent of the size of piece, and is dependent on the size of the state space. Of course, fast on-line computation is not without cost: storing a piece in our highly abstract format requires substantial computation time compared with search.

4. EXPERIMENTAL SETUP

To test our measures, we created a database of 188 MIDI pieces, about half consists of Western classical pieces; the other half consists of everything from modern popular tunes to Broadway show tunes to traditional folk songs.

We ran these pieces through a thematic extractor called MME [12]. MME extracts up to ten of the most prominent themes from a piece of music. These themes were used as the target for the queries. We then converted all of these pieces into the transition table (τ) representation.

Our search engine takes a query in MIDI format and induces a transition table (τ) representation for it. The engine then compares this transition table to all transition tables (themes) in the database. It then returns correlation measures for all pieces, sorted by most to least highly correlated.

We modified the themes to simulate the errors that a user might introduce when querying the system [5]. The three types of errors we examined were:

- *Duration-Change Error* - The user may not sing a note's correct duration. This was simulated by changing the delta time of an event in the MIDI file. Changes were made in increments of standard note lengths, from a 32nd note to a whole note.
- *Pitch-Change Error* – The user may sing the wrong note. These errors were simulated by varying the pitch of a note up and down as much as 20 MIDI numbers.
- *Note-Drop Error* - The user may forget notes. This was simulated by removing notes from the query. A 10% *note-drop* error rate indicated that 10% of the notes were removed from the query.

5. RESULTS AND DISCUSSION

For each run, one class of errors was varied, while the rest were held constant. The error rate was varied from 0% to 100% at 10% increments. 100 themes were randomly chosen from our database and manipulated each time. The themes were not changed during the experiment. The net result was 28 runs of 100 themes each time. A query for each test was created from one of the 100 themes. A correct match means that the top result is from the same piece from which the query was derived.

The results indicate that the system is robust to two types of errors, duration change and note drop. The system does not perform, however, well on pitch-change errors. We believe this is due to a combination of factors. First, if the themes extracted from the original piece are not very long or do not have a sufficient repetitive structure, it is difficult for the system to find statistical regularity. Second, the system tries to identify similar states between the query and the database. If a change in the theme alters the state table in such a way that it is disjoint from the original state, then the measures will not work

The duration-error results are a little surprising in that the system performs well when the duration error is at 100%. The result, however, make sense. The input that we are basing our database upon is highly quantized. This means that many times even when we change the duration by several MIDI ticks, it still is close to the original duration.

The least surprising result is from the note-drop error injection. Here, the quality of results decreases gradually as the error percentage increases. The fact that the system performs as well as it does at 50% error is promising. After 50% error rate, the accuracy of the system quickly decays.

6. ACKNOWLEDGEMENT

This research is supported by NSF Award #0085945. The views expressed here are solely those of the authors.

7. REFERENCES

[1] W. J. Dowling. Scale and contour: Two components of a theory of memory for melodies. Psychological Review, 85(4):341–54, 1978.

[2] J. Chih-Chin, L. Arbee, and L. Chen. An approximate string-matching algorithm for content-based music data retrieval. In *Proc. of IEEE International Conference on Multimedia Computing and Systems*. IEEE, 1999.

[3] M. Clausen, R. Englebrecht, D. Meyer, and J. Schmitz. Proms: A web-based tool for searching in polyphonic music. In *Proc. of the International Symposium on Music Information Retrieval*, 2000.

[4] Kjell Lemström and Smai Perttu. Semex - an efficient music retrieval prototype. In *Proc. of the International Symposium on Music Information Retrieval*, 2000.

[5] Rodger J. McNab, Lloyd A. Smith, Ian H. Witten, Clare Henderson, and Sally Jo Cunningham. Towards the digital music library: Tune retrieval from acoustic input. In *Digital Libraries '96: Proceedings of the ACM Digital Libraries Conference*, 1996.

[6] J. S. Downie. *Evaluating a Simple Approach to Music Information Retrieval*. PhD thesis, University of Western Ontario, 1999.

[7] R. Brunelli and S. Messelodi. Robust estimation of correlation with applications to computer vision. *PR*, 28(6):833–841, June 1995.

[8] Claude Alamkan, William P. Birmingham, and Mary H. Simoni. Stylistic structures: An initial investigation of the stochastic generation of tonal music. Technical Report CSE-TR-395-99, University of Michigan, August 1999.

[9] Claude Alamkan. Stochastic analysis and generation of music. Master's thesis, University of Michigan, May 1999.

[10] Irwin Miller and Marylees Miller. *John E. Freund's Mathematical Statistics*. Prentice Hall, 6th edition, 1999.

[11] Colin Meek and William P. Birmingham. Thematic extractor. The Proceedings of ISMIR-2001, Bloomington, IN, October, 2001.

Adaptive User Modeling in a Content-Based Music Retrieval System

Pierre-Yves ROLLAND
Laboratoire d'Informatique de Paris 6 (LIP6)
and Atelier de Modélisation et de Prévision, Université d'Aix-Marseille III,
15/19 allée Claude Forbin, 13627 Aix en Provence cedex 1, France
Tel: +33 61 19 19 19 7
P_Y_Rolland@yahoo.com

1. INTRODUCTION

While it is clear that user modeling could be valuable in many music retrieval contexts [1], the focus in this paper is on content-based music retrieval as in the WYHIWYG (*What You Hum Is What You Get*) paradigm [5], also referred to as Query-by-Humming. Desirable data to include in such a user model include:

- Musical preferences, expressed by the user by answering the system's questions (examples can be found in [1]).
- History-oriented information computed by the system, e.g. the music genre most often retrieved by the user *recently* (or up to now).

Such data can indeed be utilized by a CBMR system to bias its search results, hopefully making the latter more accurate. Based on user data the system builds expectations which are used to filter candidate results. For instance, if a song belonging to the user's most usually retrieved music genre matches the user's query, it would be advantaged by the system over a matching song of a genre the user never retrieved before. Similarly for songs belonging or not to a genre declared by the user as being among his/her favorite. A number of methods have been proposed for collecting, representing and updating these more traditional kinds of user data. These are out of the scope of this paper.

In this paper the focus will be on presenting concepts and techniques for **modeling a user's sense of musical similarity**, which I see as absolutely central. However the similarity model which will be proposed throughout the rest of this paper is seen as part of a larger user model including the more 'traditional' data types mentioned above. Whatever the user modeling paradigms used, I suggest that user models in CBMR should be **adaptive**. This means that the model is continuously enhanced throughout the successive interactions with the user. While it can be desirable to initialize user X's model by asking X a series of questions, the adaptive paradigm allows to instead initialize the model of every new user to a default and then automatically, incrementally personalize it based on the user's feedback.

In the next section I explain why modeling of a user's sense of musical similarity is seen as central in a CMBR system.

2. RATIONALE

The rationale behind this work lies in the following points:

- **A.** Most — if not all — content-based retrieval systems for music use similarity searching.
- **B.** Similarity search is based on an explicit, or sometimes implicit, **formal model of musical similarity as it is perceived by human listeners**. In this paper, to avoid any confusion between 'musical similarity model' and 'user model' the former will be designated by the more restrictive term 'melodic similarity *function*'. Such a mathematical function is designed to automatically compute the similarity between two melodic pieces or passages, often entailing a whole algorithm such as a dynamic programming one. Many different such functions have been proposed. Each is underlied by a sequence comparison scheme that is either exact or approximate (strict vs. error-tolerant matching), binary or gradual (boolean vs. gradual similarity function), etc. — see e.g. [4] for a review
- **C.** Human judgments of musical similarity are multidimensional. This means that the perceived degree of similarity of, say, two passages not only derives from the absolute pitch and duration of the notes heard but generally from a far larger set of musical characteristics of the two passages.
- **D.** The relative importance of descriptions in the overall similarity judgment can be different from one description to another. As a well-known example, it has been established that two isochronous passages having exactly the same underlying interval sequence are often judged more similar than two isochronous passages having mostly the same absolute pitches sequence but with several mismatches.
- **E.** Last but not least, **the relative importance of one given description can vary from one individual to another.** For instance, for certain human subjects rhythmic aspects contribute more strongly to similarity than pitch aspects, and vice-versa.

For all these reasons I suggest it is desirable that CBMR systems should use not only general user models, but also models of users' sense of musical similarity. Since there is no way to (entirely) predict the parameter values of such a similarity model beforehand — i.e. based on general user characteristics of the user — the model should be **adaptive.** In other words, the system adjusts the similarity model based on user feedback received during successive interactions with the user (search sessions).

There is a trade-off between search speed and search quality. Very fast search techniques have been developed for CBMR (e.g. [2]), which is convenient for allowing the user to carry out for instance a broad 'initial screening' of a music content database. However these techniques easily lack recall (or even precision) because their time efficiency relies on the simplicity of musical similarity models and, correlatively, on the strictness of match criteria. Similarly to standard text-based search engines, it is seen as important that CBMR systems should also offer 'advanced' or 'specialized' search modes based, among others, on richer musical similarity models and more flexible match criteria. The ideas and techniques presented in this paper should prove even more useful for these slower search paradigms.

3. USER MODELING PARADIGM

The proposed user modeling paradigm is intrinsically linked to the CBMR framework in general, and to that of melodic similarity assessment schemes it uses. These frameworks are presented in the first three subsections.

3.1 CBMR Framework

The CBMR paradigm in which we place, viz. that of the Melodiscov system [5] will now be briefly described. Schematically, pattern-matching techniques are used to match the user's query against the target collection of music pieces (see Figure 1). The underlying similarity function (see 3.2) being *gradual*, search results are returned under the form of a ranked list of matches, by decreasing order of match quality. In the typical querying mode, the user can hum, sing (with lyrics), whistle or play an acoustic instrument, in which case Melodiscov's *transcription* module transforms the audio query into a MIDI-like music structure called *raw symbolic query*. (Here the adjective 'symbolic' is used to distinguish between direct content, viz. digital audio signal, and abstract content such as MIDI or score representations).

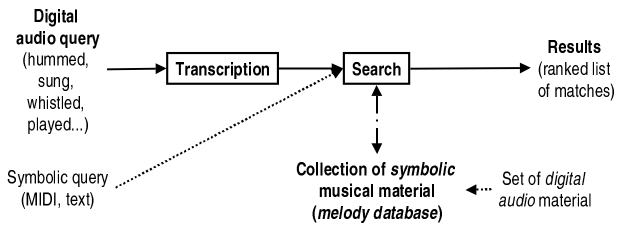

Digital audio query (hummed, sung, whistled, played...)
Transcription
Search
Results (ranked list of matches)
Symbolic query (MIDI, text)
Collection of symbolic musical material (melody database)
Set of digital audio material

Figure 1. CBMR paradigm (Melodiscov system)

Although some of the concepts and techniques proposed in this paper would apply to other kinds of CBMR schemes, it is assumed here that the target content database is a collection of musical works such as MIDI songs, called *melodic database*. In that phrase as well as in the rest of the paper, 'melodic' is used to distinguish from other kind of musical content, e.g. harmonic (chord) sequences. However 'melodic' does not imply monophonic material or mere pitch sequences with no rhythmic information.

3.2 Melody Representation

In Melodiscov music is represented using multiple characteristics (called *descriptions* henceforth). These are derived from the immense body of work that has been carried out in the areas of music psychology, music perception and cognition, and music theory at large. The various descriptions for melodic material are categorized according to their horizontal span (examples given below do not necessarily fit all melody retrieval contexts — a far more complete list can be found in [3]):

- *Individual* descriptions correspond to individual notes (or rests, or chords). Examples: 'absolute pitch', 'forward interval direction', 'backward chromatic interval', 'backward duration ratio', 'forward metric variation'…
- *Local* descriptions correspond to groups of notes (or rests/chords). Examples: 'ascending pitch contour', 'gap-fill', 'phrase-based grouping'…
- *Global* descriptions correspond to a whole melody (viz. one song in the searched database or the hummed/sung/whistled/… query). Examples: 'pitch histogram', 'average note duration' 'time signature', 'overall tonality'.

The raw symbolic query output by the query transcription module is, roughly speaking, a MIDI melody. Similarly, currently in Melodiscov the melody database is initially made of standard MIDI files. This initial, MIDI-type, representation comprises only three descriptions: the individual descriptions absolute pitch (or Midipitch 0-127), relative duration (in number of beats) and absolute amplitude (0-127). An algorithmic step is required to compute the *final* representation from the *initial* one. An automated *representation enrichment phase* is inserted in the CBMR algorithmic scheme. In an incremental process, descriptions are derived one after the other from basic descriptions and/or already derived descriptions, in a specific order (see [3] for more details).

3.3 Melodic Similarity Assessment

3.3.1 Overview

The melodic similarity function used in Melodiscov is based on the Multidimensional Valued Edit Model or MVEM (see e.g. [4]). MVEM has been designed to accommodate the multiplicity of musical descriptions, each possibly with a different horizontal span. MVEM generalizes the basic string edit distance framework and allows to carry out soft matching, i.e. allows a level of discrepancy between the query and a candidate passage in a target music work. Such error tolerance is fundamental in CBMR systems because errors in CBMR result from many possible causes:

- Users' inaccurate remembering of the searched melody
- Monophonic rendering, by users, of inherently polyphonic queries (think for instance of a theme with some bi-phonic passages)
- Transcription process, either because the query is physically too inaccurate (wrong pitches and/or rhythm) or because of technical shortcomings in the transcription algorithm.

3.3.2 MVEM in greater detail

MVEM will now be described in more technical terms, using as an example the similarity computation between the two passages shown in Figure 2. These two passages illustrate typical ornamentation cases that can be encountered in CBMR.

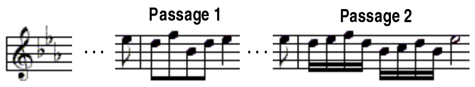


Figure 2. Two melodic passages (Joseph Haydn, Concerto for Trumpet in Eb major)

To compare two passages, the optimal correspondence scheme between their respective elements is determined. A 'correspondence scheme', called *alignment*, is a series of *pairings*, with each pairing meaning that two groups of notes and/or rests are put in correspondence (see Figure 3 and Figure 4). For instance, the second pairing in Figure 3 puts in correspondence notes 2 and 3 of passage 1 with notes 2, 3, 4 and 5 of passage 2. This is depicted in gray on the figure as two ellipses connected by an oriented link. I have introduced the notion of pairing to define alignments as it provides a richer and more flexible formalism than the traditional 'edit operations' formalism. Each group in a pairing may contain only one note or even zero note (see below). The succession of pairings forming an alignment between the two passages is interpreted as a *transformation* of passage 1 into passage 2. For instance, in Figure 4 it is considered that the final

Eb (quarter note) in passage 1 is *replaced* by the final Eb (half note) of passage 2. Similarly, in passage 2 the second Eb is said to be *inserted*.

For any pairing the number of notes in each group determine the *pairing type*. Let the *signature* of a pairing be the integer pair (#G1,#G2) where #G1 (resp. #G2) is the number of notes and/or rests in the pairing's first group (resp. second group). In the above example, the pairing's signature is (2,4).

- Pairings with a signature of that form, i.e. (r, s) where r > 1 and s > 1 are called generalized replacements.
- Pairings with signature (1,1), such as the first pairing in Figure 3, are called [individual] replacements.
- (0,s) pairings, where s > 1, are called generalized insertions.
- (r,0) pairings, where r > 1, are called generalized deletions.
- (0,1) pairings are called [individual] insertions.
- (1,0) pairings are called [individual] deletions.

As can be seen, such important musical notions as ornamentation or variation can be neatly dealt with in this framework. One other powerful feature of MEVM is that it can *elicit* (or explain) the similarity between two passages P and P', but this is out of the scope of this paper.

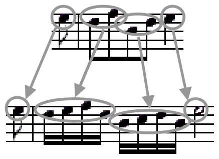


Figure 3. One possible alignment between the two passages, using 2 individual replacements and 2 generalized replacements

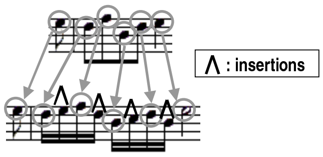


Figure 4. Another possible alignment between the two passages, using 6 individual replacements and 4 insertions

3.4 Valuation

In a **valued** edit model, a *similarity contribution function* (in short 'contribution function') is associated to each pairing type. Every pairing *p* in an alignment gets a numerical evaluation *contrib(p)* reflecting its individual contribution to the overall similarity. The contribution may be positive or negative. The various descriptions in the representation are simultaneously taken into account in contribution functions using a weighted linear combination paradigm, as shown in Equation 1. *contrib(p)* is the sum, over all descriptions belonging to the music representation *R*, of terms $w_D \times contrib_D(p)$, where:

- w_D is the weight attributed to description *D*, a real number in [0;1]. A weight has value zero iff the associated description is not taken into account in the model, at least at that particular moment.
- $contrib_D(p)$ is the contribution of *p* seen only from the point of view of description *D*. Suppose for example that p is the replacement of the final Eb (quarter note) in passage 1 by the final Eb (half note) of passage 2 in Figure 4. Consider the basic descriptions *D1*: 'degree of note in overall tonality' and *D2*: 'relative duration of note in beats'. We can expect $contrib_{D1}(p)$ to have a strong positive value because the degree is the same for both notes. Conversely, $contrib_{D2}(p)$ can be expected to have a (moderately) negative value because the duration of the 'replacing' note is double that of the 'original' note.

The *value* of an alignment is the sum of all of its constitutive pairings contributions (Equation 2). Finally, the **similarity** between passage 1 and passage 2 is defined as being the **greatest value of all possible alignments between the two passages**. There are several techniques, based on dynamic programming, for computing that greatest value as well as, if needed, the corresponding alignment(s). In the case of CBMR, the *matching quality* (or 'matching score', etc.) between a query and a music work is the greatest value of all possible alignments between the query and a passage of the work. The search results are made of the list of works, ordered by decreasing matching quality, whose matching quality is above a predefined threshold.

$$contrib(p) = \sum_{D \in R} w_D \times contrib_D(p)$$

Equation 1

$$Value(A) = \sum_{p \in A} contrib(p)$$

Equation 2

3.5 Model Representation and Adaptation

3.5.1 Description weight vector

A user's sense of melodic similarity is modeled using a scalar vector which we call *description weight vector* (DWV). The DWV contains the weight w_D of each description D in the music representation, each weight being able to vary throughout user interaction — primarily search sessions. This user model underlies a melodic similarity function that emulates as closely as possible the user's sense of melodic similarity.

The numeric vector format of the user model allows it to be shared by different applications or agents. In fact, a musical similarity function is inherently modular, other musical software such as navigational interfaces, pattern extraction programs and so on can directly reuse it for the omnipresent purposes of melodic comparison. Also, a DWV can directly be merged with a sharable user model such as the one suggested in [1].

3.5.2 Initialization and interaction

The first time a user uses Melodiscov, the DWV is initialized by setting all of its components (description weights) to a default value of 0.5.

Every time the user is presented with a ranked list of search results, s/he gives feedback to the system in two possible fashions:

- In the simplest interaction mode, *single match feedback*, s/he just tells the system which of the found matches is correct, i.e. corresponds to the music work actually looked for. In case that music piece does not appear in the system's list of matches, the user may request that lower quality matches, if any, should be displayed. These are matches whose similarity scores are below a given constant threshold specific to the CBMR system.
- In a more complex one, *ranked match feedback*, s/he gives her/his own ranking of some or all of the matches. Think for instance of a content database containing several variations of a target song; these variations could be designated by the user to the system as 'reasonable secondary matches'.

3.5.3 Model update

Unless the user has confirmed the system's best match (*single match feedback*) or best matches (*multiple match feedback*), the user's DWV is updated in the following manner. (For the sake of simplicity it will be assumed that *single match feedback* mode has been used; feedback in the other mode is dealt with similarly).

The system's ranked list $\{M_1,…, M_m\}$ of matches is separated in three groups (in decreasing order of match quality as computed by the system) :

- the group $FP=\{FP_1, ..., FP_f\}$ of all the 'false positive' matches (i.e. all matches reported by the system with a better matching score than the correct match. In other words, if the user selects Mi the list's then $FP=\{M_1..M_{i-1}\}$)
- the correct match C
- the group S of all subsequent matches proposed by the system

For each, the contribution of each description to the matching score is computed. The weights of some, if not all, weights in the user's DWV are then adjusted in such a way that, after adjustment, the new ranking gets closer to what it should be, i.e. C should be ranked first. The actual update algorithm in detail is not important here, what is key is the underlying idea. The latter will be presented through two characteristic cases:

- For each description D such that the term $v_{C,D}$ is greater than its vertically homologous terms in FP_1, … FP_f (viz. $v_{FP1,D}$, … $v_{FPf,D}$), W_D is increased in the following manner :

  $$w_D = (1+k)w_D$$

  Intuitively, this is based on the observation that, if a term $v_{C,Di}$ contributes more to the similarity in the correct match than in the false positive matches, it should be reinforced via an increase of its weight.

- Conversely, for each description D such that the term $v_{C,D}$ is lesser than its corresponding terms $v_{FP1,D}$, … $v_{FPf,D}$:
  W_D is decreased in the following manner :

  $$w_D = w_D/(1+k)$$

  This is based on the observation that, if a term $v_{C,D}$ contributes more to the dissimilarity in the false positive matches than in the correct match, it should be attenuated via a decrease in its weight.

The positive real number k is called *update rate*. Of course, k values close to 0 induce weak updates while higher values induce more drastic updates. In order to force model convergence (stabilization), k can also be made a decreasing function of time, tending to zero. This is similar to the temperature function used in simulated annealing algorithms.

What has just been presented is the normal, 'ongoing' interaction scenario: model update occurs based on feedback the user gives throughout successive search sessions. In addition, the user can, initially but also at any time, enter *system learning sessions* that are directed toward fast user model learning. In these sessions, instead of carrying out CBMR searches the user makes direct similarity judgments about melodic material presented by the system. In the simplest setting, the user is presented with melodic passages A, B and B' and must tell the system which of the pairs A-B and A-B' is more similar. Of course, every (A,B,C) triplets is chosen (by the system's designer) so as to emphasize the contrast between two particular descriptions. The results of the successive similarity rankings made by the user during such a learning session are finally aggregated, resulting in an appropriate update in the user's DWV.

As can be observed, the current weight update scheme uses a fixed strategy similar to error gradient feedback in neural networks. It should be remarked that other strategies such as evolutionary algorithms could be other appropriate candidates.

4. RELATED AND FUTURE WORK

The techniques proposed in this paper are currently being implemented within the Melodiscov system. Melodiscov uses a core set of object-oriented classes and methods allowing to represent music with multiple, individually weighted descriptions. Using that same representation platform, the influence of description weight adjustment has been experimented in the context of automated musical pattern extraction [3], a problem area directly connected to that of CBMR. Additionally, the outcome of the work presented in [6] may be very useful.

The current priority is on completing implementation and experimenting with the system. One future work direction concerns model initialization. Currently the initial user model is a default that is the same for every user. I wish to investigate whether more appropriate initial models could be generated for every user based on the characteristics recorded in a standard, 'general' user model. For instance, suppose that user X's general model says that X *has received significant musical education*. Stronger weights would then be given to the more abstract descriptions in user X's initial model (e.g. harmony-oriented ones) than if X was known to *have received no musical education*. While this simple example relies on common sense, more probabilistically accurate initialization strategies for user models could be designed based on statistical analysis of evolved and stabilized user models of perceived musical similarity. Another future work direction is investigating aggregation/fusion strategies for synergetically mixing the DWV-based model with more 'traditional' user models such as the ones mentioned in the first section of this paper.

5. REFERENCES

[1] Chai, Wei and Vercoe, Barry. 2000. Using user models in music information retrieval systems. Proc. International Symposium on Music Information Retrieval, Oct. 2000.

[2] Lemstrom, K. and S. Perttu. 2000. SEMEX - An efficient Music Retrieval Prototype. Proceedings ISMIR'00.

[3] Rolland, P.Y. 1999. Discovering Patterns in Musical Sequences. *Journal of New Music Research* 28:4, December 1999.

[4] Rolland, P.Y., Ganascia, J.G. 1999. Musical Pattern Extraction and Similarity Assessment. In Miranda, E. (ed.). *Readings in Music and Artificial Intelligence*. New York and London: Gordon & Breach - Harwood Academic Publishers.

[5] Rolland, P.Y., Raskinis, G., Ganascia, J.G. 1999. Musical Content-Based Retrieval : an Overview of the Melodiscov Approach and System. In Proceedings of the 7th ACM International Multimedia Conference, Orlando.

[6] Shmulevich, I., Yli-Harja, O., Coyle, E., Povel D., Lemström, K. 2001. Perceptual issues in music pattern recognition. In Rolland, P.Y., Cambouropoulos, E., Wiggins, G. (editors). Pattern Processing in Music Analysis and Creation. *Computers and the Humanities* 35(1), journal special issue.

Discovering Themes by Exact Pattern Matching

Lloyd Smith and Richard Medina
New Mexico Highlands University
Las Vegas, NM 87701
+1 505 454-3071
{las,richspider}@cs.nmhu.edu

1. INTRODUCTION

Content-based music information retrieval provides ways for people to locate and retrieve music based on its musical characteristics, rather than on more familiar metadata such as composer and title. The potential utility of such systems is attested to by music librarians, who report that library patrons often hum or whistle a phrase of music and ask them to identify the corresponding musical work [5, 7].

Content-based MIR systems operate by taking a musical query (i.e., a string of notes) from the user and searching the music database for a pattern closely matching the query. The search may be carried out by exhaustively matching the database [5] or by matching an index of n-grams created by passing a sliding window of length n over the database [2].

While these exhaustive search methods are adequate for relatively small music databases they do not scale well to large collections such as thousands of symphonies and other major works.

Fortunately, for large classical works, such as sonatas and symphonies, it is possible to avoid exhaustive search by using an index of themes. A theme, in classical music, is a melody that the composer uses as a starting point for development. A piece of music may have several themes; each of them will repeat and may be slightly changed (a “variation”) by the composer on repetition. Using such an index greatly condenses the search on a database of classical major works. Furthermore, themes are the musical phrases likely to be remembered by listeners, so a theme index helps focus the search on the parts of the database most likely to match a query.

One well known theme index is that produced by Harold Barlow and Sam Morgenstern [1]. This is a print book containing approximately 10,000 themes from the classical music genre. Each theme is identified by composer, title of work and section of appearance (movement for symphony, act for ballet, overture for opera, and so forth). In addition to its print edition, this theme index can also be searched online [3].

Unfortunately, it is a monumental task to manually compile such a theme index. Because themes are, by definition, recurring patterns in a piece of music, it should be possible to automate the discovery of musical themes in order to create a theme index over a given database.

Our goal is to perform this automation – to analyze a piece of music and automatically determine its themes. Some work has been done on finding themes in music. Mongeau and Sankoff, for example, suggested the use of dynamic programming for finding recurring sections in a piece of music [6]. Their method, however, was somewhat cumbersome, relying on a closeness threshold to determine the beginnings and ends of recurring musical patterns.

2. DISCOVERING THEMES IN MUSIC

Because a theme, by definition, is a melody that the composer uses as a starting point for variation, most researchers have assumed that a theme discovery system must use approximate string matching [6].

Our approach takes the view that a theme dictionary may be constructed using an exact match of the musical sequence against itself. Our hypothesis is that a significant part of a theme is likely to repeat at least once, and that smaller chunks of a theme are likely to repeat multiple times. The basic idea is similar to that followed by Liu, et al. [4], but, where they build theme candidates by joining small repeating patterns into larger ones, we start with the longest repeating patterns and look for continuations and substrings.

Theme discovery is essentially a search for self-similarity in a piece of music. For that reason, we begin by creating a self-similarity matrix. For a musical piece of *n* notes, this is an *n x n* matrix representing where each interval exactly matches another interval in the piece – a repeated interval is represented by a *1* in the matrix. We use intervals in order to make our analysis independent of key. This does not make the analysis independent of mode – our algorithm is not likely to find repetitions of a major-key theme in minor mode, for example, or vice versa. If, however, a variation repeats multiple times, our algorithm will discover those patterns.

From the self-similarity matrix, we build a lattice showing all repeating patterns longer than a predetermined length and their relationships. At this point we are analyzing all repeating patterns of four or more notes. This lattice is further processed and used to determine which patterns to keep as candidate themes.

3. CASE STUDY

As a simple test of our algorithm, we used it to identify repeating patterns in Bach’s Fugue No. 7, from the Well Tempered Clavier. We used only the top (soprano) part of the fugue – this is to find out whether our algorithm can find the theme without seeing its reiterations when the second and third voices enter. When processing the entire piece, we simply concatenate all parts to form one long sequence – this enables the algorithm to capture themes from repetitions in different voices, but does not attempt to discover themes split among more than one voice – the algorithm is not expected to find a theme, for example, that starts in the first violin and migrates into the cello.

Figure 1 shows part of the lattice built from the top part of Fugue No. 7. As stated above, we ignore repeating patterns of fewer than 4 intervals.

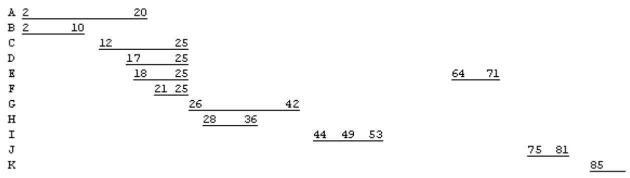


Figure 1. Partial lattice for fugue no. 7.

The lattice clearly shows a pattern of 18 notes, labeled A, near the beginning of the piece, starting with the second interval. This is not surprising, given the structure of a fugue, with the introductory statement of the subject. Pattern B is a substring of A, while pattern C overlaps A. Patterns D, E and F are substrings of C. This is only part of the lattice; pattern A repeats at position 311, B repeats at 93 and 311, C repeats at 103, and so forth. The lattice shows one repeat of pattern E at 64; E also repeats at 109 and 133. Pattern I overlaps itself, starting at position 44 and ending – and starting again – at position 49. A total of 20 repeating patterns were found.

We envision the theme index being searched by approximate search on user queries. For that reason, it is unecessary to keep patterns that are substrings of other patterns. Furthermore, we extend candidate themes by combining overlapping patterns. Figure 2 shows the lattice after discarding substrings and short patterns that occur only twice. Overlapping patterns A and C have been combined into one longer pattern. In fact, AC is the theme of the fugue – minus the first interval – as listed by Barlow and Morgenstern [1].

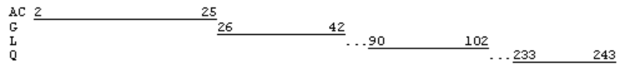


Figure 2. Lattice for fugue no. 7 after pruning.

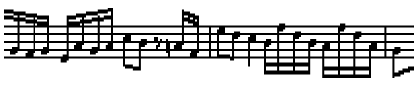
Pattern G appears twice in the fugue, it is long, and it has a substring that repeats, so it is included in the final list of patterns to be added to the theme dictionary.

Pattern L is mostly a substring of AC, and could be eliminated. However, it adds two (tied) notes to the beginning of AC and, at this point, is left in the list because of that extension. It is, in fact, a theme variation.

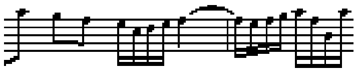
Pattern Q remains in the list because of its length. It appears only twice and has no repeating substrings. Inspection of Q shows that it is another variation on part of the theme, and it leads into a repetition of pattern C. Q is very similar to pattern B (the intervals up to the rest in AC), but introduces an E-natural in place of the expected E-flat.

Given the fact that a theme index is expected to be searched using approximate matching, it is likely that patterns L and Q should not be included – the first part of pattern AC is close enough to both of them to allow user queries to retrieve this particular fugue.

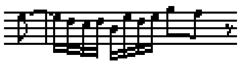
Figure 3 shows musical notation, extracted from the score, for patterns AC, G, L and Q,.



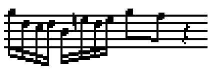
(a) Pattern AC



(b) Pattern G



(c) Pattern L



(d) Pattern Q

Figure 3. Candidate themes from Fugue No. 7.

4. CONCLUSION

This paper describes a method for automatically discovering themes in music. A program based on this algorithm can generate a theme index from a music database.

At this point, we have tested the algorithm using simple musical structures – namely, fugues. This provides a suitable beginning test because we can easily analyze whether the program is acting appropriately. In developing the algorithm, we have not made use of any musical knowledge regarding the structure of fugues, but have let the algorithm discover what it regards as themes. Our immediate goal now is to test the algorithm over a much wider range of music. A longer term goal is to produce a music analysis system based on the algorithm and to incorporate more sophisticated approximate matching to complement the base algorithm.

5. REFERENCES

[1] Barlow, H. and S. Morgenstern. *A Dictionary of Musical Themes,* Crown Publishers, NY, 1948.

[2] Downie, J. S. and M. Nelson. “Evaluation of a simple and effective music information retrieval method,” Proc ACM SIGIR 2000, 73-80, 2000.

[3] Huron, D. *Themefinder,* http://www.themefinder.org/.

[4] Liu, C. C., J. L. Hsu and A. L. P. Chen. “Efficient theme and non-trivial repeating pattern discovering in music databases,” Proc. IEEE Intl. Conf. on Data Engineering, 14-21, 1999.

[5] McNab, R. J., L. A. Smith, I. H. Witten and C. L. Henderson, “Tune retrieval in the multimedia library,” Multimedia Tools and Applications 10, 113-132, 2000.

[6] Mongeau, M. and D. Sankoff. “Comparison of musical sequences,” Computers and the Humanities 24, 161-175, 1990.

[7] Salosaari, P. and K. Jarvelin. “MUSIR -- a retrieval model for music,” Technical Report RN-1998-1, University of Tampere, Department of Information Studies, 1998.

Melodic Resolution in Music Retrieval

Timo Sorsa and Jyri Huopaniemi
Nokia Research Center,
Speech and Audio Systems Laboratory
P.O.Box 407, FIN-00045
Nokia Group, Finland

{timo.sorsa,
jyri.huopaniemi}@nokia.com

ABSTRACT

This poster considers the use of different levels of melodic resolution in acoustically driven music retrieval systems from the viewpoint of search-key lengths. A query-by-humming application was constructed to evaluate the dependencies between the melodic resolution, database size and the search-key length in order to consider the optimal level of melodic resolution in music retrieval applications.

1. INTRODUCTION

Acoustically driven retrieval systems (often referred to as query-by-humming applications[1]-[3]) are a recent approach for efficient and flexible music retrieval. These acoustically driven music retrieval systems use a hummed, whistled or played sample of a melody as a search-key to search matching database entries from a music database. Current efforts in standardization such as MPEG-7 [4] are a clear indication of the research and commercial interest on the topic.

In the general case, when the user generates the input to a melody retrieval process by humming, whistling or playing an instrument, the input is noisy. Noise, meaning errors in the input melody in relation to the database entries, affects the accuracy of the process. To overcome this some approximation can be introduced to the retrieval process.

It is customary to use two methods to introduce approximation to query-by-humming applications. First, approximate string matching algorithms are used. Second, different levels of melodic resolution are used. By lowering the resolution, i.e. using fewer intervals to represent melody-lines, the system can eliminate some of the interval errors included in the input signals. But it is a trade-off; the lower the resolution is the lower is the disparity between the database entries. In this poster the emphasis is on the use of different levels of melodic resolution.

2. SYSTEM ARCHITECTURE

A melody retrieval application was constructed for the purpose of evaluating the concept of query-by-humming in general and to test the effects of using different levels of melodic resolution in the retrieval process. In the following the two main functional blocks of the system are presented.

2.1 Acoustical front-end

The acoustical front-end transcribes the input melody into an inner representation (IRP)[5]. The transcribed melody is used as a search-key by the database engine.

The input signal is segmented into notes with amplitude-based segmentation. A normalized signal level is compared to two constant thresholds. A higher threshold is used for detecting note onsets and a lower one for detecting note offsets.

The fundamental frequencies of the frames within segmented notes are determined with an autocorrelation-based pitch tracker [6]. On a note level the pitch is estimated as the median of the pitch values of the frames within a note. The presented retrieval system does not use rhythmic information and therefore note duration is not detected.

The acoustical front-end is designed to accept input generated by humming, whistling and playing an acoustic instrument. The system also has an option to take typed search-keys as inputs. The user can type the names of the notes of the melody in question and that pattern is then used as a search-key.

2.2 Database engine

The core of the database engine has been implemented at the Department of Computer Science at the University of Helsinki. (See [7] for details.) A fast bit-parallel dynamic programming algorithm by Myers is used for approximate string matching [8]. Transcription invariance is assured by the use of intervals in the matching process.

3. MELODIC RESOLUTION

Melodic resolution refers to the accuracy of the representation of the melody-lines. Essentially different levels of melodic resolution are achieved by using different number of intervals to represent the melody.

The use of lower melodic resolution is motivated by the approximation that it offers for the user input. When using lower resolution the intervals in the input melody do not have to be as accurate as with higher resolution. On the other hand the use of lower resolution forces the user to use longer search-keys for successful retrieval.

In the developed system five different levels of melodic resolution can be used in the matching process. The highest resolution presents the melody with semitone accuracy using 25 intervals (12 up and down and the prime). The lowest resolution is the so-called sign-based contour representation that represents the melody with only three intervals (see for example [9]).

3.1 Required Length of the Search-key

From the users point of view the search-key length is a relevant parameter when considering the experienced quality of a retrieval system.

For the purpose of studying the effect of using different melodic resolutions the system was tested with correct typed inputs. That is, for every search-key used, there was an exact match in the database. The melody lines of 13 different tunes were used as search-keys. Typed input was used in this test so that the accuracy of the audio analysis would not have an effect on the test results.

The tests showed that on average, in order to find a unique match, the sign-based contour representation (3 intervals) requires about 1.7 times longer search-keys than the semitone resolution.

In *Figure 1* the relationship between the database size and the required search-key length is presented. *Figure 1* shows the average required search-key lengths for four different databases with five different levels of melodic resolution.

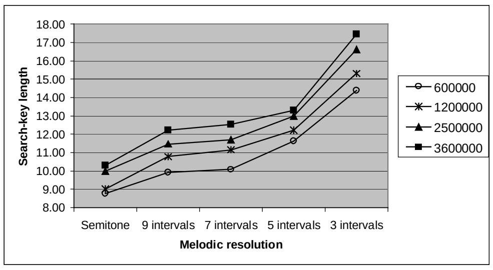


Figure 1. The average search-key lengths required for successful unique retrieval using different levels of melodic resolution in test MIDI databases of 0.6, 1.2, 2.5 and 3.6 million notes with correct search-keys.

The search-key length becomes even more relevant when the errors in the input signals are considered. This was studied with another test. In this test the same search-keys were used as in the previous test but one error was included in each of the keys by removing one random note from the pattern.

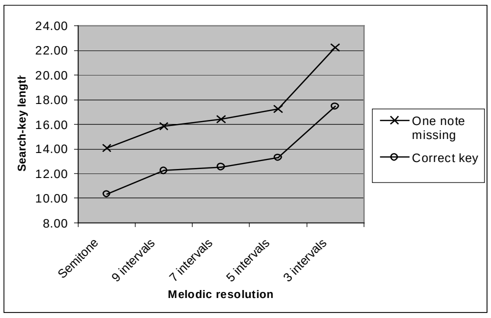


Figure 2. The average search-key lengths required for successful unique retrieval in a test MIDI database of 3.6 million notes. Lower values for correct search-keys and higher values for search-keys with one error (one random note missing).

In *Figure 2* the required search-key lengths for successful unique retrieval in a test database are presented for five different levels of melodic resolution. The search-key lengths are calculated for an optimal case where the search-key is exactly like the corresponding melody-line in the database and for a more realistic case where there is one error in the search-key.

4. CONCLUSIONS

The tests reported above imply that for large databases acoustically driven melody retrieval is not, in the general case, accurate enough. This is mainly due to the errors included in the search-keys generated by the users. This is not to say that query-by-humming type applications are not practical and user-friendly but it implies that the effective size of a large database has to be relatively small if query-by-humming type algorithms are applied.

The tests give an indication of the optimal level of melodic resolution from the point of view of search-key lengths. 9-, 7-, and 5-interval representations are relatively equal from this viewpoint, whereas the 3-interval representation requires significantly longer search keys than the other representations.

The concept of query-by-humming is in many ways a user friendly and efficient method for melody retrieval but the tests indicate that, from the database size point-of-view, it has some, relatively low upper limits beyond which the retrieval process becomes too inaccurate and impractical. When these upper limits are exceeded some other means for classifying the database entries has to be considered. One straightforward method of reducing the effective database size is the use of key words to identify relevant parts of the database in which the actual melody based retrieval process is executed.

5. REFERENCES

[1] McNab, R.J., Smith, L.A., Bainbridge, D., and Witten, I.H., "The New Zealand digital library MELody inDEX", *D-Lib Magazine, May 1997.*

[2] Sonoda, T. and Muraoka, Y., "A WWW-based melody-retrieval system- an indexing method for a large melody database", *Proceedings of ICMC 2000, pp. 170-173, Berlin, Germany, 2000.*

[3] Ghias, A., Logan, J., Chamberlin, D., and Smith, B.C., "Query by humming – musical information retrieval in an audio database", *ACM Multimedia '95, San Francisco, USA, 1995.*

[4] MPEG Requirements Group, "Overview of the MPEG-7 Standard", *Doc. ISO/MPEG N4031, MPEG Singapore Meeting, March 2001 edited by Martinez, J.M.,* http://www.cselt.it/mpeg/standards/mpeg-7/mpeg-7.htm.

[5] Lemström, K. and Laine, P., "Musical information retrieval using musical parameters" *Proceedings of ICMC'98, pp. 341-348, Michigan, USA, 1998.*

[6] Rabiner, L.R., Cheng, M.J., Rosenberg, A.E. and McGonegal C.A., "A comparative performance study of several pitch detection algorithms", *IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 24, no. 5, pp. 399-418, July 1976.*

[7] Lemström, K. and Perttu, S., "SEMEX - An efficient music retrieval prototype", *Proceedings of Music IR 2000, Plymouth, MA, USA, Oct. 2000.*

[8] Myers, G., "A fast bit-vector algorithm for approximate string matching based on dynamic programming", *Proceedings of the 9th Annual Symposium on Combinatorial Pattern Matching, Piscataway, USA, 1998.*

[9] Dowling, W.J., "Scale and contour: two components of a theory of memory and melodies", *Psychological Review, Vol. 85 no.4 pp. 341-354, 1978.*

Sound Spotting – a Frame-Based Approach

Christian Spevak
Music Department
University of Hertfordshire
College Lane, Hatfield, AL10 9AB, UK
+44 1707 28 4442
c.spevak@herts.ac.uk

Richard Polfreman
Music Department
University of Hertfordshire
College Lane, Hatfield, AL10 9AB, UK
+44 1707 28 6473
r.p.polfreman@herts.ac.uk

ABSTRACT

We present a system for content-based retrieval of perceptually similar sound events in audio documents ('sound spotting', using a query by example. The system consists of three discrete stages: a front-end for feature extraction, a self-organizing map, and a pattern matching unit. Our paper introduces the approach, describes the separate modules and discusses some preliminary results and future research.

1. PROBLEM

The possibility of storing large quantities of sound or video data on digital media has resulted in a growing demand for content-based retrieval techniques to search multimedia data for particular events without using annotations or other meta-data. This paper presents an approach to a task that can be described as *sound spotting:* the detection of perceptually *similar* sounds in a given document, using a *query by example*, i.e. selecting a particular sound event and searching for 'similar' occurrences. The proposed system could be applied to content-based retrieval of sound events from broadcasting archives or to aid transcription and analysis of non-notated music.

A particular problem is posed by the definition of *perceptual similarity:* sound perception comprises so many different aspects that it is very hard to define a general perceptual distance measure for a pair of sounds. Even if the variability is restricted to timbre alone, it is still uncertain how to construct a *timbre space* with respect to any underlying acoustical features. Within the scope of our system we decided to focus on the spectral evolution of sounds by calculating a time-frequency distribution and splitting the signal into a series of short-time frames. Similarity can then be assessed by comparing sequences of frames.

2. APPROACH

Our concept builds on various connectionist approaches to modelling the perception of timbre that have been investigated over the last ten years [1, 2, 3]. These systems typically consist of some kind of auditory model to preprocess the sounds, and a self-organizing map to classify the resulting feature vectors. The reported experiments involved the classificaton of a small number of test sounds equalized in pitch, duration and loudness. To extend these models towards evolutions of timbre, pitch and loudness we have pursued a dynamic, frame-based approach involving three stages.

First the raw audio data is preprocessed by an auditory model performing a *feature extraction*. The signal is divided into short-time frames and represented by a series of feature vectors. In the current system we use a parametric representation adopted from automatic speech recognition, mel-frequency cepstral coefficients (MFCC).

Second a *self-organizing map* (SOM) is employed to perform a vector quantization while mapping the feature vectors onto a two-dimensional array of units. The SOM assigns a best-matching unit to each input vector, so that a sound signal corresponds to a sequence of best-matching units.

Finally a pattern matching algorithm is applied to search the entire source for sequences 'similar' to a selected pattern. Currently we refer to the SOM units simply by discrete symbols (disregarding the associated weight vectors and topological relations) and perform an *approximate matching* on the resulting sequences.

3. SYSTEM COMPONENTS

3.1 Feature Extraction

Besides their application in speech recognition mel-frequency cepstral coefficients have been successfully utilized for timbre analysis [4] and music modeling [5]. MFCC calculation involves the following steps: the signal is divided into short frames (10-20 ms), a discrete Fourier transform is taken of each frame and its amplitude spectrum converted to a logarithmic scale to approximately model perceived loudness. The spectrum is smoothed by combining Fourier bins into outputs of a 40 channel mel-spaced filterbank (*mel* being a psychological measure of pitch magnitude). Finally a discrete cosine transform is applied to extract principal components and reduce the data to typically 13 components per frame.

3.2 Self-Organizing Map

Self-organizing maps constitute a particular class of artificial neural networks, which is inspired by brain maps such as the tonotopic map in the auditory cortex [6]. A self-organizing map can be imagined as a lattice of neurons, each of which possesses a multidimensional weight vector. Feature vectors are mapped onto the lattice by assigning *a best-matching unit* to each vector.

Self-organization of the map takes place during a training phase, where the entire data is repeatedly presented to the network. The SOM 'learns' the topology of the input data and forms a set of ordered discrete reference vectors, which can be regarded as a reduced representation of the original data.

To enable an efficient pattern matching process in the third stage of the system we represent the best-matching units by their index number only and disregard their mutual relations. A sound sample then corresponds to a string of symbols, which can be further processed by means of string searching algorithms.

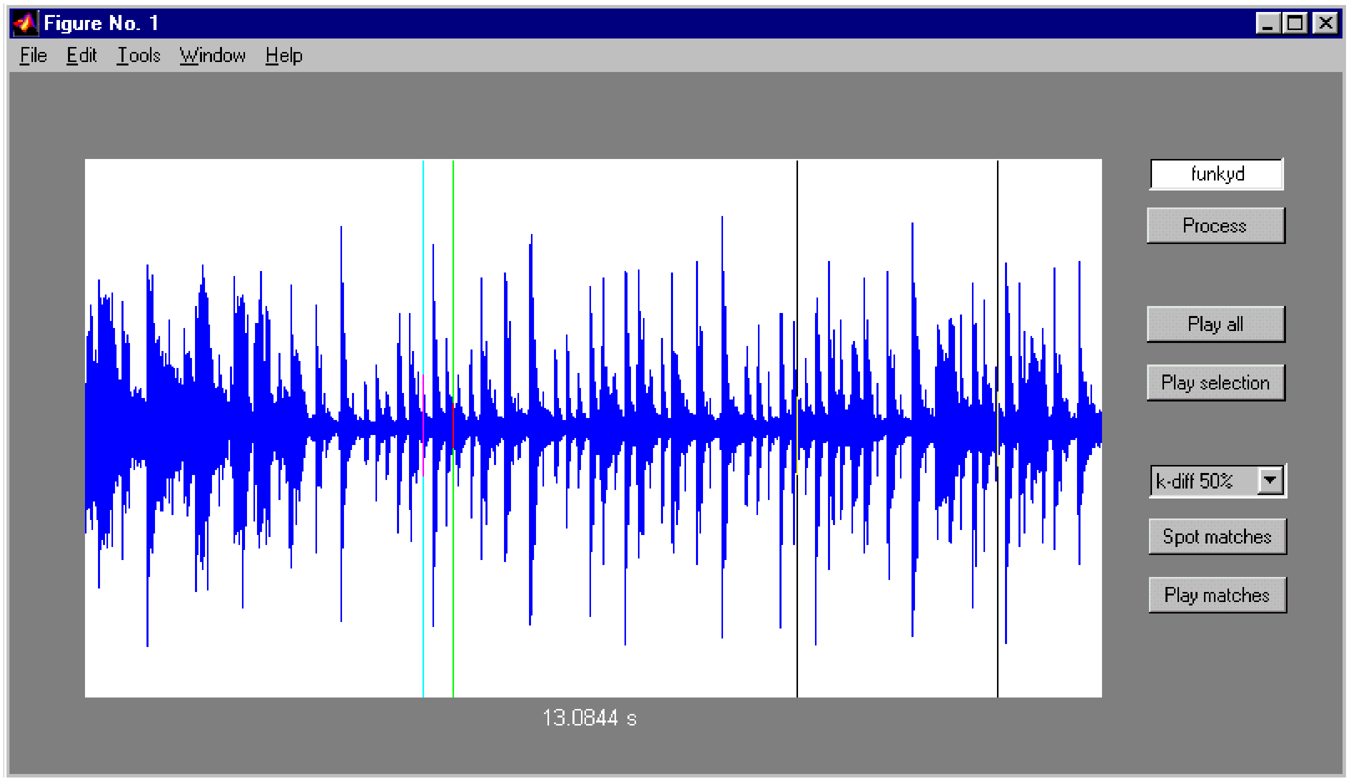


Figure 1. Graphical user interface of the prototype implementation.

3.3 Pattern Matching

We use a *k-difference inexact matching* algorithm to retrieve approximate matches of a selected *pattern* from the entire *text* [7]. The algorithm retrieves matches differing by an *edit distance* of at most *k*, where edit distance denotes the minimum number of operations needed to transform one string into another, permitting insertion, deletion and substitution of symbols. *k* can be specified with respect to the pattern length (e.g. 40%).

4. DISCUSSION

Initial experiments conducted with a MATLAB prototype implementation (see Figure 1) have demonstrated varying degrees of success in retrieving perceptually similar sounds. Encourageing results have been obtained for instance with drum loops, where similar sounds could easily be detected. Difficulties arise when an event has to be detected in a mixture of different sounds. The reduction of the multidimensional feature vectors to index numbers and the use of a simple string matching algorithm clearly entails a significant loss of potentially important information, which could be avoided by a more sophicated distance measure in conjunction with a suitable pattern matching algorithm. These issues will be addressed in future research.

5. REFERENCES

[1] Piero Cosi, Giovanni De Poli and Giampaolo Lauzzana (1994). Auditory modelling and self-organizing neural networks for timbre classification. *Journal of New Music Research* 23(1): 71-98.

[2] Bernhard Feiten and Stefan Günzel (1994). Automatic indexing of a sound database using self-organizing neural nets. *Computer Music Journal* 18(3): 53-65.

[3] Petri Toiviainen (2000). Symbolic AI versus connectionism in music research. In: Eduardo Reck Miranda (ed.) *Readings in Music and Artificial Intelligence*, 47-67. Harwood Academic Publishers.

[4] Giovanni De Poli and Paolo Prandoni (1997). Sonological models for timbre characterization. *Journal of New Music Research* 26: 170-197.

[5] Beth Logan (2000). Mel frequency cepstral coefficients for music modeling. *Proceedings of the International Symposium on Music Information Retrieval (ISMIR)*.

[6] Teuvo Kohonen (2000). *Self-Organizing Maps*. Third edition. Springer.

[7] Graham A. Stephen (1994). *String Searching Algorithms*. World Scientific Publishing, Singapore.

Music Database Retrieval Based on Spectral Similarity

Cheng Yang[*]
Department of Computer Science
Stanford University
yangc@cs.stanford.edu

Abstract

We present an efficient algorithm to retrieve similar music pieces from an audio database. The algorithm tries to capture the intuitive notion of similarity perceived by human: two pieces are similar if they are fully or partially based on the same score, even if they are performed by different people or at different speed.

Each audio file is preprocessed to identify local peaks in signal power. A spectral vector is extracted near each peak, and a list of such spectral vectors forms our intermediate representation of a music piece. A database of such intermediate representations is constructed, and two pieces are matched against each other based on a specially-defined distance function. Matching results are then filtered according to some linearity criteria to select the best result to a user query.

1 Introduction

With the explosive amount of music data available on the internet in recent years, there has been much interest in developing new ways to search and retrieve such data effectively. Most on-line music databases today, such as Napster and mp3.com, rely on file names or text labels to do searching and indexing, using traditional text searching techniques. Although this approach has proven to be useful and widely accepted, it would be nice to have more sophisticated search capabilities, namely, searching by content. Potential applications include "intelligent" music retrieval systems, music identification, plagiarism detection, etc.

Most content-based music retrieval systems operate on score-based databases such as MIDI, with input methods ranging from note sequences to melody contours to user-hummed tunes [2, 5, 6]. Relatively few systems are for raw audio databases. Our work focuses on raw audio databases; both the underlying database and the user query are given in .wav audio format. We develop algorithms to search for music pieces similar to the user query. Similarity is based on the intuitive notion of similarity perceived by humans: two pieces are similar if they are fully or partially based on the same score, even if they are performed by different people or at different tempo.

See our full paper [12] for a detailed review of other related work [1, 3, 4, 7, 8, 9, 10, 11, 14].

2 The Algorithm

The algorithm consists of three components, which are discussed below.

1. Intermediate Data Generation.

   For each music piece, we generate its spectrogram, and plot its instantaneous power as a function of time. Next, we identify peaks in this power plot, where peak is defined as a local maximum value within a neighborhood of a fixed size. Intuitively, these peaks roughly correspond to distinctive notes or rhythmic patterns, with some inaccuracy that will be compensated in later steps. We extract the frequency components near each peak, taking 180 samples of frequency components between 200Hz and 2000Hz. This gives us n spectral vectors of 180 dimensions each, where n is the number of peaks obtained. After normalization, these n vectors form our intermediate representation of the corresponding music piece.

2. Matching.

   In this step, two music pieces are compared against each other by matching spectral vectors in the intermediate data. We associate a "distance" score to each matching by computing the sum of root-mean-squared errors between matching vectors plus a penalty term

[*]Supported by a Leonard J. Shustek Fellowship, part of the Stanford Graduate Fellowship program, and NSF Grant IIS-9811904.

for non-matching items. A dynamic programming approach is used to find the best matching that minimizes this distance. Furthermore, a "linearity filtering" step is taken to ensure that matching vectors reflect a linear scaling based on a consistent tempo change.

3. Query Processing.

   All music files are preprocessed into the intermediate representation of spectral vectors discussed earlier. Given a query sound clip (also converted into the intermediate representation), the database is matched against the query using our minimum-distance matching and linearity filtering algorithms. The pieces that end up with the highest number of matching points are selected as answers to the user query.

See [12] for details and analysis of the algorithm.

3 Experiments and Future Work

We identify five different types of "similar" music pairs, with increasing levels of difficulty:

- Type I: Identical digital copy
- Type II: Same analog source, different digital copies, possibly with noise
- Type III: Same instrumental performance, different vocal components
- Type IV: Same score, different performances (possibly at different tempo)
- Type V: Same underlying melody, different otherwise, with possible transposition

Sound samples of each type can be found at `http://www-db.stanford.edu/~yangc/musicir/`.

Tests are conducted on a dataset of 120 music pieces, each of size 1MB. For each query, items from the database are ranked according to the number of final matching points with the query music, and the top 2 matches are returned. For each of the first 4 similarity types, retrieval accuracy is above 90%. Type-V is the most difficult, and better algorithms need to be developed to handle it.

We are experimenting with indexing schemes [13] in order to get faster retrieval response. We are also planning to augment the algorithm to handle transpositions, i.e., pitch shifts.

References

[1] J. P. Bello, G. Monti and M. Sandler, "Techniques for Automatic Music Transcription", in *International Symposium on Music Information Retrieval*, 2000.

[2] S. Blackburn and D. DeRoure, "A Tool for Content Based Navigation of Music", in *Proc. ACM Multimedia*, 1998.

[3] J. C. Brown and B. Zhang, "Musical Frequency Tracking using the Methods of Conventional and 'Narrowed' Autocorrelation", *J. Acoust. Soc. Am.* 89, pp. 2346-2354. 1991.

[4] J. Foote, "ARTHUR: Retrieving Orchestral Music by Long-Term Structure", in *International Symposium on Music Information Retrieval*, 2000.

[5] A. Ghias, J. Logan, D. Chamberlin and B. Smith, "Query By Humming – Musical Information Retrieval in an Audio Database", in *Proc. ACM Multimedia*, 1995.

[6] R. J. McNab, L. A. Smith, I. H. Witten, C. L. Henderson and S. J. Cunningham, "Towards the digital music library: Tune retrieval from acoustic input", in *Proc. ACM Digital Libraries*, 1996.

[7] E. D. Scheirer, "Pulse Tracking with a Pitch Tracker", in *Proc. Workshop on Applications of Signal Processing to Audio and Acoustics*, 1997.

[8] E. D. Scheirer, *Music-Listening Systems*, Ph. D. dissertation, Massachusetts Institute of Technology, 2000.

[9] A. S. Tanguiane, *Artificial Perception and Music Recognition*, Springer-Verlag, 1993.

[10] G. Tzanetakis and P. Cook, "Audio Information Retrieval (AIR) Tools", in *International Symposium on Music Information Retrieval*, 2000.

[11] E. Wold, T. Blum, D. Keislar and J. Wheaton, "Content-Based Classification, Search and retrieval of audio", in *IEEE Multimedia*, 3(3), 1996.

[12] C. Yang, "Music Database Retrieval Based on Spectral Similarity", *Stanford University Database Group Technical Report* 2001-14. `http://dbpubs.stanford.edu/`

[13] C. Yang, "MACS: Music Audio Characteristic Sequence Indexing for Similarity Retrieval", in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2001.

[14] C. Yang and T. Lozano-Pérez, "Image Database Retrieval with Multiple-Instance Learning Techniques", *Proc. International Conference on Data Engineering*, 2000, pp. 233-243.

ISMIR 2001

Tuesday, October 16, 2001

7am-9pm	E-mail room: IMU088 *
7:30-8:30am	Registration and Continental Breakfast: Alumni Hall
8:30-8:45am	Welcome/Opening Remarks

8:45-10:20am, Systems and Interfaces 1

8:45-9:30am	Invited Speaker, Jef Raskin: *Making Machines Palatable*
9:30-9:55am	*GUIDO/MIR - an Experimental Musical Information Retrieval System based on GUIDO Music Notation* Holger H. Hoos, Kai Renz, Marko Görg
9:55-10:20am	*A Naturalist Approach to Music File Name Analysis* François Pachet, Damien Laigre

10:20-10:45am	Break/Refreshments

10:45-12:00n, Systems and Interfaces 2

10:45-11:10am	*Score Processing for MIR* Donncha S. Ó Maidín, Margaret Cahill
11:10-11:35am	*An Audio Front End for Query-by-Humming Systems* Goffredo Haus, Emanuele Pollastri
11:35-12:00n	*MUSART: Music Retrieval Via Aural Queries* Willian P. Birmingham, Roger B. Dannenberg, Gregory H. Wakefield, Mark Bartsch, David Bykowski, Dominic Mazzoni, Colin Meek, Maureen Mellody, William Rand

12:05-1:20pm	Lunch

1:20-2:55pm, Musicology and Extraction of Musical Information 1

1:20-2:05pm	Keynote, David Cope: *Computer Analysis of Musical Allusions*
2:05-2:30pm	*Musical Works as Information Retrieval Entities: Epistemological Perspectives* Richard P. Smiraglia
2:30-2:55pm	*The JRing System for Computer-Assisted Musicological Analysis* Andreas Kornstädt

2:55-3:20pm	Break/Refreshments

3:20-5:00pm, Musicology and Extraction of Musical Information 2

3:20-3 :45pm	*Automated Rhythm Transcription* Christopher Raphael
3:45-4:10pm	*Melody Spotting Using Hidden Markov Models* Adriane Swalm Durey, Mark A. Clements
4:10-4:35pm	*Thematic Extractor* Colin Meek, William P. Birmingham
4:35-5:00pm	*Figured Bass and Tonality Recognition* Jerome Barthélemy, Alain Bonardi

5:00-7:00pm	Free Time
5:30-6:15pm	Informal Discussion of Testbed Databases: Distinguished Alumni Room
7:00-9:00pm	Dinner/Awards

* E-mail room computers will be configured with Telnet and Web browsers. Check with your network and/or security personnel on any special arrangements you may need to connect to your home or institutional resources.

ISMIR2001 Invited Address

Making Machines Palatable

Jef Raskin
Human-Computer Interaction Author,
Designer, and Consultant
8 Gypsy Hill
Pacifica CA 94044
jefraskin@aol.com

ABSTRACT

In a way, this talk is addressed to anybody that is involved with developing systems to aid human in intellectual pursuits. That I have been using computers to notate, edit, store, and perform music since the 1960s is almost irrelevant to my being here, except that it makes me especially sympathetic to the needs of other musicians**.**

1. INTRODUCTION

There is a sad truth I wish to bring forcibly to our attention, and surprisingly, it is one that we all know, though you may have put in the back of your mind -- for if you thought about it often, it would either make you mad or drive you mad. And that truth is that our computers and our software are too hard to use. This would not be a primary concern of mine if the difficulty was a necessary consequence of computer technology and the tasks we are trying to accomplish, but most of the difficulties are unnecessary. They can be fixed. Computer software is not beyond our control; unlike the weather, we can do something about it.

The obvious question is this: Considering that computers were made to help humans with tasks once thought to require intellect for their successful accomplishment, why were they so difficult to use? And why hasn't the problem been rectified?

2. DISCUSSION

A good deal of the reason has to do with history. In the computer industry the term "legacy" is often applied to things developed long ago and still in use. We have a great deal of this in music. Some of the notations of music, the layout of the keyboard, the use of key signatures, our rich variety of clefs, that we have transposing instruments in the orchestra, and a myriad of other details of musical life and nomenclature could be streamlined to great advantage. Yet we are tied into our history, we are trained from youth in our arcane skills. Because it is training, secured by repetition upon repetition, it is difficult to dislodge them. If I may be permitted a personal note, I presently watch my son becoming professionally competent on the French horn. We are both horrified that hornists are expected to be able to play in a profusion of transpositions, a legacy from the days that horn players slipped in a crook to change key. Why do most modern editions and orchestral copyists preserve the old notation, now a relic useful only for tripping over? Because that was the way it used to be done.

A second reason is fashion. Does anybody find a splash screen on a computer program useful? Why do we waste pixels on making window boarders shaded, as if they had thickness (did anybody not see them when they were flat-looking?). Where legacy represents ideas that once made sense carried into a world where they no longer do, fashion is intended to be primarily decorative when introduced, and then persists mindlessly. Fashion and legacy are an especially dangerous couple, the Bonnie and Clyde of usability. An example is the present desktop metaphor for a computer operating system. The concept is a harmful legacy, the presentation a miserable fashion. There is not room in this abstract to detail the reasons.

A third reason is ignorance. Most developers of software are simply not aware of the strides that have been made in understanding how humans interact with their electronic servants. It is challenge enough to have competence in, say, music, acoustics, and computer science without also having to become a student of *cognetics*. Cognetics is the application of cognitive psychology to the mental side of human-machine interaction (much as ergonomics is applied physiology, essential in designing the physical side of human-machine interaction). However, there is little excuse for not finding a colleague, expert in the field, who can help out.

There are superb reasons for paying attention to interface design. If done incorrectly a facility can be made difficult and time-consuming to use. For example, most music-entry systems (in fact, all that I have seen) are unnecessarily complex and slow. They are created on the basis of legacy, fashion, and incompetence. Published methods for predicting keystroke counts and time to complete tasks have been long available [1], popular yet reliable treatments of the foibles of poor design have appeared, e.g. [2], and a deeper understanding of the ways we work with interfaces (or instruments) based on recent results in cognition research has developed in the past few years, along with quantitative measures of efficiency [3].

If you are going to create software or hardware that interacts with humans then even a small spark of consideration and humanity would seem to demand that you acquaint yourself with the basics of what is known. Otherwise you will, through ignorance, saddle your users with unnecessarily high error rates and extra work (leading to lowered productivity, higher risk of repetitive stress injuries, and the pressures of frustration).

3. CONCLUSION

If you use software or hardware, then you should know that what you are saddled with is not decreed from on high. You will learn that there is no real excuse to not doing a lot better, and you can work more effectively with vendors and programmers. In either case, allow me to point you to the two brief books [2] and [3] on my list of references. Whether designer, programmer, or user, you will find yourself better able to deal with technology after a brief immersion in this most practical corner of evidence-based psychology.

4. REFERENCES

[1] Card, Moran, and Newell. *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum Associates, 1983

[2] Norman. *The Design of Everyday Things*. Basic Books, 1990

[3] Raskin. *The Humane Interface*. Addison Wesley, 2000

GUIDO/MIR — an Experimental Musical Information Retrieval System based on GUIDO Music Notation

Holger H. Hoos[*] [†] and **Kai Renz**[†] and **Marko Görg**[†]

Abstract

Musical databases are growing in number, size, and complexity, and they are becoming increasingly relevant for a broad range of academic as well as commercial applications. The features and performance of musical database systems critically depend on two factors: The nature and representation of the information stored in the database, and the search and retrieval mechanisms available to the user. In this paper, we present an experimental database and retrieval system for score-level musical information based on GUIDO Music Notation as the underlying music representation. We motivate and describe the database design as well as the flexible and efficient query and retrieval mechanism, a query-by-example technique based on probabilistic matching over a clustered dataset. This approach has numerous advantages, and based on experience with a first, experimental implementation, we believe it provides a solid foundation for powerful, efficient, and usable database and retrieval systems for structured musical information.

1 Introduction

Multimedia databases play an important role, especially in the context of online systems available on the World Wide Web. As these databases grow in number, size, and complexity, it becomes increasingly important to provide flexible and efficient search and retrieval techniques. When dealing with musical data, two main difficulties are encountered: Firstly, the multidimensional, often complex structure of the data makes both the formulation of queries and the matching of stored data with a given query difficult. Secondly, there is often a considerable amount of uncertainty or in accuracy in the query and/or the data, stemming from limitations of the methods used for obtaining queries, such as "Query-By-Humming" [12], or for acquiring musical data, such as automated performance transcription, as well as from simple human error when entering data.

While there is a strong and increasing interest in database and retrieval systems for sound and sound-level descriptions of music, many application contexts (particularly in musical analysis, composition, and performance) benefit from or require higher-level, structured music representations. Consequently, there is a growing body of research on musical databases and music information retrieval based on structured, score-level music representations (see, e.g., [3; 21; 8]). In this work, we focus on content-based music information retrieval from a database of score-level musical data based on the query-by-example approach [2]. The main contributions of our work, can be summarised as follows:

1. We use GUIDO Music Notation [16] as the music representation underlying the database as well as for formulating queries. Compared to the use of MIDI and various other music representation formats, this approach has a number of conceptual and practical advantages which will be discussed in detail in the following sections. We find that GUIDO is particularly suitable for formulating queries in a query-by-example approach, and we outline how a small and natural extension of GUIDO allows the explicit and localised representation of uncertainty associated with a given query.
2. We introduce a novel music retrieval mechanism based on probabilistic models and a hierarchically clustered musical database. Using probabilistic models for musical information retrieval has the advantage of offering natural, elegant, and flexible ways of scoring exact and approximate matches between pieces in the database and a given query. While in this work, we introduce and illustrate this general concept using rather simple probabilistic models, the approach can be easily generalised to

[*]Corresponding author. University of British Columbia, Department of Computer Science, 2366 Main Mall, Vancouver, BC, V6T 1Z4, Canada, hoos@cs.ubc.ca

[†]Technische Universität Darmstadt, Fachbereich Informatik, Wilhelminenstr. 7, D-64283 Darmstadt, Germany, renz@iti.informatik.tu-darmstadt.de

more complex probabilistic models.

3. We present an experimental database and retrieval system which implements the design and techniques proposed in this paper. This prototypical system, which is available on the WWW, supports various combinations of melodic and rhythmic query types for retrieving information from a database of pieces of varying complexity. The system is implemented in Perl [1] and highly portable; the underlying, object-oriented and modular design facilitates the implementation of different search and retrieval techniques and the investigation of their behaviour.

In the following, we present and discuss our overall approach in more detail. We start with a brief introduction of GUIDO Music Notation and discuss its use in the context of musical database and retrieval systems. In Section 3, we outline our experimental musical database design and implementation. Section 4 is the core of our work; it motivates and describes our approach to music information retrieval. Related approaches are briefly discussed in Section 5, and Section 6 presents some conclusions and outlines a number of directions for future research.

2 Why GUIDO?

GUIDO Music Notation[1] is a general purpose formal language for representing score level music in a platform independent, plain-text and human-readable way [16]. The GUIDO design concentrates on general musical concepts (as opposed to only notational, i.e., graphical features). Its key feature is *representational adequacy*, meaning that simple musical concepts should be represented in a simple way and only complex notions should require complex representations. Figure 1 contains three simple examples of GUIDO Music Notation and the matching conventional music notation.

The GUIDO design is organised in three layers: Basic, Advanced, and Extended GUIDO Music Notation. *Basic GUIDO* introduces the basic GUIDO syntactical structures and covers basic musical notions; *Advanced GUIDO* extends this layer to support exact score formatting and more sophisticated musical concepts; and *Extended GUIDO* introduces features which are beyond conventional music notation. GUIDO Music Notation is designed as a flexible and easily extensible open standard. Thus, it can be easily adapted and customised to cover specialised musical concepts as might be required in the context of research projects in computational musicology. GUIDO has not been developed with a particular application in mind but to provide an adequate representation formalism for score-level music over a broad range of applications. The intended application areas include notation software, compositional and analytical systems and tools, musical databases, performance systems, and music on the WWW. Currently, a growing number of applications is using GUIDO as their music representation format.

[1]GUIDO Music Notation is named after Guido d'Arezzo (ca. 992-1050), a renowned music theorist of his time and important contributor to today's conventional musical notation.

GUIDO vs. MIDI

Currently, virtually every (content-based) MIR system works on MIDI files. The two main reasons for that are:

- the enormous amount of music available as MIDI files on the WWW
- the lack of a commonly used and accepted representation format for structured music

Although Standard MIDI File (SMF) format is the most commonly used music interchange format, it does not *adequately* support activities other than playback. MIDI was never intended to be the notation (and music) interchange format that it has become today.

There are several reasons, why MIDI is not very well suited for MIR. A MIDI file contains a low level-description of music which describes only the timing and intensity of notes. Since structural information such as chords, slurs or ties cannot be stored in a Standard MIDI file[2], a high- or multilevel description is not possible. Some of the basic limitations of a MIDI file are the lack of differentiation between enharmonic equivalents (e.g.. C-sharp and D-flat), and lack of precision in the durations between events (which are expressed in MIDI-ticks).

Our MIR system has been implemented using GUIDO as its underlying music representation language. To still be able to use the huge body of MIDI files on the WWW, our group has developed converters between GUIDO and MIDI[3].

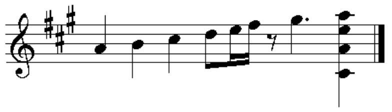
GUIDO vs. XML

XML is a simplified subset of SGML, a general markup language that has been officially registered as a standard (ISO8879). Because of its increasing popularity, there have been quite a number of attempts to use it for storing musical data [14; 6] and as well as for Music Information Retrieval [26]. XML has obvious and undeniable strengths as a general representation language: it is platform independent, text-based, human-readable and extensible. Additionally, by using XML to represent music, one gains the advantage of using a standardised metalanguage for which a growing number of tools are becoming available. To our knowledge none of the approaches to music representation using XML published so far has yet gained wide acceptance. One

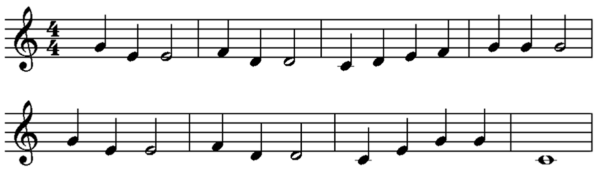
[2]Using non-standard techniques, it is possible to store additional information in MIDI files; however, these mechanisms are not part of the standard

[3]GMN2MIDI and MIDI2GMN are available at our web site `http://www.salieri.org/guido`

```
[ \key<"A"> a1/4 h c#2 d/8 e/16 f#16 _/8
  g#*1/4. {a1/4,c#,e2,a2} ]
```



```
[ \key<"C"> \meter<"4/4"> g1/4 e e/2
f/4 d d/2 c/4 d e f g g g/2 g/4 e e/2
f/4 d d/2 c/4 e g g c/1 ]
```



```
{ [ \tempo<"Vivace">
  \meter<"5/8"> \intens<"p"> \sl(\bm(g1*1/8 a b)
  \bm(b& c2) \bm(c# b1 a b& a&)) ],
[ \meter<"5/8"> \sl(g1*3/8 d/4 c#*3/8 d/4) ] }
```

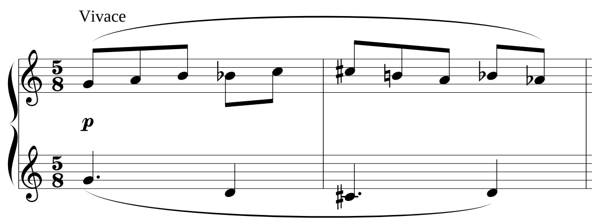


Figure 1: Simple examples of GUIDO Music Notation; more complex examples can be found in [17,18].

of the reasons for this seems to lie in the complexity of musical structure; just using a “new” format does not automatically lead to a simple and easy to use data structure.

To allow the use of XML-tools where needed, we have developed GUIDO/XML, a XML compliant format that completely encapsulates GUIDO within a XML structure. Using GUIDO/XML is simple: we provide tools that convert GUIDO Music Notation files into GUIDO/XML files and vice versa. Using this approach, we can continue to use GUIDO Music Notation and its associated tools (SALIERI, NoteAbility, NoteServer, ParserKit, etc.) but are also free to use any current or emerging XML tool.

One advantage of XML is its ability to store so called meta data. A piece of music can be associated with a composer, a title, a publisher, a publishing date and even version information. One can easily add new meta data fields encoding additional musical information (like for example performance-related data for a piece). Using GUIDO/XML in conjunction with a set of meta data information can lead to complete XML-compatible descriptions of structured music.

Using GUIDO Music Notation for Musical Databases and MIR

As we have shown, GUIDO Music Notation offers an intuitive yet complete approach for representing musical data. Using GUIDO in musical databases is therefore a straight forward task: because it is a plain-text format, no additional tools are necessary to create, manipulate or to store GMN files. It is also possible to use standard text-compression tools to minimise storage space (the size of compressed GMN files compares to the size of MIDI files). By using existing tools like the GUIDO NoteServer[25], one can create conventional music notation from GUIDO descriptions quickly.

Because of its representationally adequate design, GMN is also very well suited for MIR: Queries can be written as (enhanced) GUIDO strings. Users with a background in GUIDO can specify even complex queries in an easy way. By using additional tools like a virtual piano-keyboard, even novice users are able to build queries quickly. In Section 4 it will be shown, how using GUIDO as the underlying music representation language simplifies the task of building query-engines and we also demonstrate, how a slight extension to GUIDO leads to an intuitive approach to approximate matching.

Other representation formats (like XML) do not provide this feature: a new query language has to be created in order to access the stored information. As there is no standard for musical queries (like SQL is for relational databases systems) a whole range of different musical query languages will be proposed in the future.

3 The Experimental GUIDO Database

As was shown in the previous section, GUIDO Music Notation is well suited as a general music representation language. Our prototypical MIR system is build on the basis of an experimental GUIDO Database that will be described in this section.

The GUIDO Database contains musical pieces stored as GMN files along with some additional information which is used for efficient retrieval (this will be discussed in more detail in the next section). Instead of building our musical database based on a conventional database system, we decided to implement it in Perl [1], using the regular file system for information storage. This design offers a numbers of advantages:

- The Perl language has good support for manipula-

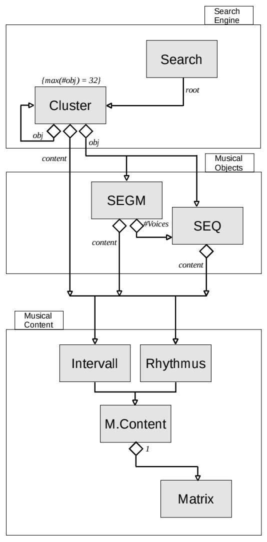


Figure 2: Overview of the object-oriented design of our experimental database and information retrieval system.

ting textual data (such as GUIDO or HTML data) and is well suited for rapid prototyping.

- Using PERL allows for very easy integration in online systems.
- Disk storage is cheap, and textual data can be compressed efficiently using general file compression techniques; furthermore, modern operating systems allow time-efficient file access through caching.
- It is easy and reasonably efficient to build index structures on a file system.
- Maintenance and updating of the database is relatively easy, since functionality of the operating system and underlying file system can be used.

One drawback of this approach is the fact that standard database functionality, such as concurrent write access, the implementation of access control, and transaction control would have to be implemented separately and are currently not supported. However, it should be noted that in the context of music information retrieval, write operations (i.e., modifications of or additions to the database) are relatively rare compared to read access (such as retrieval) and usually restricted to selected users. Interestingly, the same holds for many large and heavily used online biomedical and literature database systems. Our model is based on an off-line update mechanism, where pieces are added to the database by taking the database off-line and generating / updating the index structure while no other access is permitted.

Our implementation follows an object oriented design which is graphically summarised in Figure 2. Details of the implementation can be seen from the sources of our Perl modules, which are publicly available from `http://www.salieri.org/guido/mir`.

Currently, our database system contains about 150 files, most of which have been converted to GUIDO from other formats like abc and MIDI. Because the conversion from MIDI to GUIDO is a complex task that sometimes needs manual interaction, extending this corpus is time-consuming. However, we expect that based on recent improvements of our conversion tools, we will be able to extend our database to a much larger body of files. Our experimental system is not optimised for speed, and we are quite certain that we will need to increase its efficiency when operating on a much larger database.

4 The Experimental MIR Engine

Our music information retrieval approach is based on the "Query by Example" (QBE) paradigm [2]. QBE has the advantage that queries can be formulated in an easy and intuitive way. In many search and retrieval situations, users appear to prefer the QBE approach over the use of query languages, which support more complex queries like boolean expressions, wildcards, or regular expressions.

Query Types

Many music information retrieval systems are primarily based on melodic, i.e., pitch-related information[4]. Types of melodic information that can be used for queries are absolute pitches, pitch-classes, intervals, interval classes (e.g., large/small intervals) and melodic trends (e.g., up/down/equal). Alternately or additionally, rhythmic information can be used as a basis for retrieval. Again, various types of rhythmic information can be distinguished: absolute durations, relative durations (or duration ratios), or trends (e.g., shorter, longer, equal).

Our prototypical MIR Engine supports queries that arbitrarily combine one out of five types of melodic information with one out of three types of rhythmic information. The melodic query features are the following: absolute pitch (such as c1, d#2, etc.), intervals (such as minor third, major sixth, etc.), interval types (such as second, fourth, etc.), interval classes (equal, small, medium, large), melodic trend (upwards, downwards, static). The three currently supported rhythmic features are absolute durations (such as 1/4, 1/8., etc.), relative durations (such as 1:2, 4:3, etc.), and rhythmic trends (shorter, longer, equal). All these features are determined for individual notes or pairs of notes, respectively, such that a query effectively specifies sequences of these features.

Since we are following a QBE approach, these various query types (and their combinations) correspond merely to different interpretations of the same musical query. For instance, the GMN fragment `[g1/4 e1/4 e1/4]` can be used as a purely melodic query, using ab-

[4]see [21] for an overview of MIR systems and their pitch representations

solute pitch. In this case, only the melodic sequence [g1 e1 e1] would be matched, regardless of rhythm. The same fragment, used as a purely rhythmical query would also match [e1/4 e1/4 e1/4], and even [_/4 _/4 _/4]. For information retrieval based on the QBE approach, this paradigm of "query = data + feature selection" is very natural; this applies particularly to multi-dimensional, complex data such as musical or graphical objects.

We can also distinguish exact retrieval, where the task is to find exact occurrences of the information specified in the query, or approximate (or error-tolerant) retrieval, where a certain amount of deviation between the query information and the data to be retrieved is permitted. Here, we first consider exact retrieval, and later discuss briefly an extension of our approach to approximate retrieval.

Probabilistic Models

The music information retrieval approach taken here is based on the general idea of characterising and summarising musical structure using probabilistic models. Searching for a fragment with a specific musical structure (specified in a query) can then be done by probabilistic matching using these models. Here, we propose a rather simple approach, which is based on first-order Markov chains for modeling the melodic and rhythmic contours of a monophonic piece of music [15; 9]. Currently, we focus on horizontal queries only, i.e. queries which only involve monophonic music, and treat pieces with multiple voices (or chords) as collections of monophonic pieces.

Intuitively, a (discrete time) first-order Markov chain is a probabilistic model for a process which at each time is in a state, and at each time step probabilistically changes into a successor state (which can be the same as the current state) with a probability that only depends on the present state. Hence, first-order Markov chains are characterised by the transition probabilities t_{ab} for entering state b as the next state, when the current state is a.[5] The transition probabilities characterising a first-order Markov chain can be written in form of a square matrix $T = (t_{ab})$ whose rows and colum indices correspond to the states of the chain. It should be noted that first-order Markov chains with a finite set of states correspond to non-deterministic finite state machines (FSMs), and can also be seen as a special case of Hidden Markov Models (HMMs) where emissions for all states are deterministic [24].

In the application considered here, we conceptually use one first-order Markov chain for each melodic and rhythmic query type and each given monophonic piece. The states of these chains correspond to pitches for absolute pitch queries, to intervals for interval queries, to relative durations for relative rhythmic queries, etc. The corresponding transition probabilities are determined from frequency counts over neighbouring pitches, intervals, note durations, etc. which are normalised to obtain proper probabilities.

Figure 3 shows the transition probability matrices for the Markov chains characterising the sequences of absolute pitches and durations for the "Hänschen Klein"[6] example from the second row of Figure 1 as well as the corresponding representations as non-deterministic finite state machines; the latter representation is often more intuitive and concise.

The transition probabilities of these first-order Markov chains summarise statistical properties of the pieces in the musical database. When trying to find exact matches between a given query and pieces in the database, we can make use of the following simple observation: If for a given piece p, a transition that is present in the query (e.g., an upward fifth followed by a downward third) has probability zero, there is no exact match of the query in p. Unfortunately, the converse is not true: There are cases where there is no exact match of the given query in p, yet for any neighbouring features in the query, the corresponding transition probabilities in p are greater than zero.

Generally, the key idea of information retrieval based on probabilistic models is the following: Given a piece p and a probabilistic model $M(p)$ for this piece, this model can be used to generate pieces p' with properties similar to p. Here, these properties are the transition probabilities of the first-order Markov chains we use for characterising sequences of features. To assess the potential of a match given a query sequence $q = q_1, q_2, ..., q_n$ (where the q_i are individual features such as pitches) and a candidate piece p from the database, we determine the probability $P(q|M(p))$ that the probabilistic model of p, denoted $M(p)$, generates the feature sequence corresponding to the given query q. For our simple probabilistic model, $M(p)$ is characterised by a matrix of transition probabilities t_{ab}, and the probability of generating the query sequence given that model is given by the product of the transition probabilities t_{ab} which correspond to all neighbouring features in the query sequence:

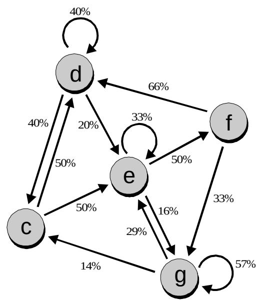
$$P(q|M(p)) = \prod\{t_{ab}|a = q_i, b = q_{i+1}, 1 \leq i < n\}$$

Intuitively, this probability score will be higher for pieces which contain many exact matches than for pieces which contain few exact matches, and as explained above, it will be zero for pieces which do not contain any exact matches at all. Since the probabilistic model we use is very simplistic and is certainly far from capturing all relevant (statistical) features of the pieces in the database, we cannot expect this intuition to be fully met. However, our practical experience with

[5]In this work, we only use homogenous Markov chains, i.e. chains, for which the transition probabilities do not change over time. In Section 6 we briefly discuss how and why a more general approach equivalent to using inhomogeneous chains might be advantageous.

[6]"Hänschen Klein" is a popupar German childrens song

$$t_{\text{pitch}} = \begin{array}{c|ccccc} & c & d & e & f & g \\ \hline c & 0 & 0.5 & 0.5 & 0 & 0 \\ d & 0.4 & 0.4 & 0.2 & 0 & 0 \\ e & 0 & 0 & 0.33 & 0.5 & 0.16 \\ f & 0 & 0.66 & 0 & 0 & 0.33 \\ g & 0.14 & 0 & 0.29 & 0 & 0.57 \end{array}$$



$$t_{\text{rhythm}} = \begin{array}{c|ccc} & \frac{1}{4} & \frac{1}{2} & \frac{1}{1} \\ \hline \frac{1}{4} & 0.67 & 0.28 & 0.06 \\ \frac{1}{2} & 1 & 0 & 0 \\ \frac{1}{1} & 0 & 0 & 0 \end{array}$$

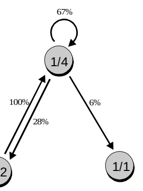


Figure 3: Transition probability matrices and Finite State Machines for absolute pitch and absolute rhythm

the experimental system described here indicates that even when using this simplistic probabilistic model, the correlation between probability scores and pieces containing exact matches is sufficient to be used as a basis for a music information retrieval mechanism.

Obviously, the transition probability matrices corresponding to related features, such as absolute pitches and intervals, are not independent, and in fact two matrices (one for absolute pitches, one for absolute durations) are sufficient as a basis for handling any type of query. However, in practice, there is a trade-off between the amount of pre-computed statistical data (transition probabilities), and the time required for matching a given query against a probabilistic model that might not be explicitly available.

Note: The techniques presented here do not directly support the efficient search of matches *within* a given piece (which might have been selected based on a high probability score for a given query). To efficiently search matches within a piece, conventional techniques, such as suffix trees (see, e.g., [20]) can be used. Alternatively, pieces can be segmented (manually, or automatically, using any suitable segmentation algorithm; see, e.g., [22]), and probabilistic modelling and matching can be applied to the segments individually.

Hierarchical Clustering

The probabilistic matching technique described before can help to reduce search effort by eliminating some of the pieces that do not match a given query, and more importantly, by identifying promising candidate pieces based on their transition probability matrices only. However, a naive search for good candidates based on probability scores would still require to evaluate the query against the probabilistic models for all pieces in the database. For very large databases, or when short response times are required, this might be too time-consuming.

One way of addressing this problem is to organise the database in form of a tree, where each leaf corresponds to one element (i.e., piece) of the musical database. For a given query, we could now start at the root and follow the paths leading to the leaves which contain pieces which match the query. This would allow us to retrieve matches in time proportional to the height of the tree, i.e., logarithmic in the number of leaves for a balanced tree. In order to do this, we need a mechanism that at each node of the tree allows us to identify the subtree that is most likely to contain a match.

As a first approximation to such a mechanism, we use combined probabilistic models which summarise the properties of all sequences in a given subtree. Note that our first-order Markov chain model can be easily generalised to sets of pieces instead of single pieces: Given two pieces p_1, p_2, we combine the two transition probability matrices $T(p_1), T(p_2)$ derived from their respective interval sequences into one joint matrix $T(\{p_1, p_2\})$ by computing a weighted sum such that the resulting transition probabilities are equivalent to those that would have been obtained by deriving a transition probability matrix from the concatenation $p_1 \cdot p_2$ of the two sequences:

$$\begin{aligned} t(\{p_1, p_2\})_{ab} &= t(p_1 \cdot p_2)_{ab} \\ &= \frac{|p_1| t(p_1)_{ab} + |p_2| t(p_2)_{ab}}{|p_1| + |p_2|} \end{aligned}$$

This method generalises to the case of combining the models of more than two sequences in a straightforward way. Matching for these combined probabilistic models works exactly as for single pieces, and the probability scores thus obtained can be used to guide the search for matches in a tree structured index of the database. Figure 4 shows how the tree structure of transition matrices is build; the search for a pattern begins at the root matrix and then continues at the descendant matrices as long as these match the transition probabilities of the query.

Obviously, the topology of the tree as well as the decision how pieces and sets of pieces are grouped together can have a large impact on the efficiency of the proposed search mechanism. One potentially very fruitful approach for deriving tree structures is the use of hierarchical clustering techniques [10]. However, it is presently not clear whether similar pieces should be clustered together or whether clustering dissimilar pieces together would be more beneficial; the former approach might make it easier to identify larger sets of promising candidates for matches early in the

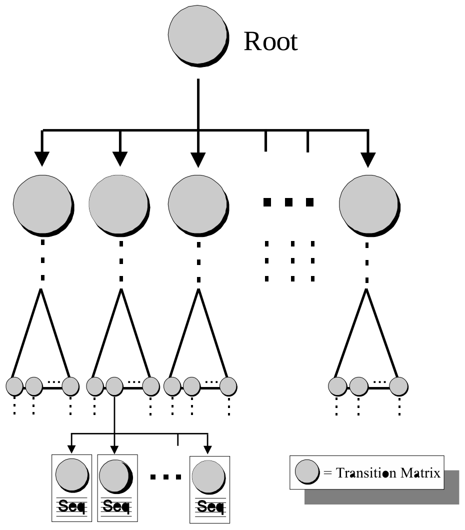


Figure 4: Tree structured index of the database

search, while the latter should facilitate selecting the most probable match from a set of pieces.

The issues arising in this context are rather complex and require thorough empirical analyses; we plan to further investigate and discuss these elsewhere. For our present prototype, we use a simple and rather arbitrary hierarchical clustering resulting in a balanced tree where each node has up to 32 children.[7] Furthermore, to speed up the search within this tree, for each node we store three bit matrices whose entries indicate whether the transition probabilities in the probabilistic model for the cluster corresponding to that node exceeds thresholds of 0, 0.15, and 0.3, respectively. Those threshold matrices are used for rapidly selecting the most promising subcluster at each internal node of the cluster tree that is visited during the search. (For details of this mechanism, see [13].)

Once again, it should be noted that the mechanisms introduced here serve a double purpose: They can potentially prune large parts of the search, for which exact matches cannot be encountered (based on the occurrence of transition probabilities with value zero), and they also heuristically guide the search such that promising candidate pieces are identified early in the search.

Approximate Matching and Error-tolerant Search

Often, queries are inaccurate or may contain errors, and relevant matches cannot expected to be perfect matches. In other cases, a user querying a musical database system might be interested in "almost matches", which might indicate interesting musical similarities. One way of addressing this situation is to use exact matching in combination with "fuzzy" queries that support features such as melodic or rhythmic trends or interval classes. But this is not always the most appropriate approach, and many music approximate retrieval mechanisms instead or additionally support true error-tolerant search, which allows (penalised) mismatches when matching queries against pieces from the given musical database[8].

Our retrieval mechanism based on probabilistic models, although primarily developed for exact matching, quite naturally extends to a certain type of error-tolerant search. To that end, both the search for good candidate sequences, as well as the search within the sequences need to be modified. While we cannot discuss the technical details of these extensions here, we will outline the general ideas and provide a detailed description elsewhere.

To localise candidate sequences in an error-tolerant way, we could modify the probabilistic models associated with the individual pieces in the database with prior information by factoring pseudo-observations into all transition probabilities (this is a standard method in machine learning, which is applied frequently when probabilistically modelling sequence data, see, e.g., [4]). Intuitively, this would reflect a certain degree of uncertainty about the pieces in the database. The hierarchical clustering of the probabilistic models and the search process based on scoring the query sequence using these probabilistic models remains unchanged, but the whole process now supports imperfect matches, which are still penalised, but no longer ruled out.

The same effect can achieved by factoring the prior information into the query; this corresponds to allowing for errors or inaccuracies in the query. The mechanism is exactly the same, only now the prior is associated with the query and gets dynamically factored into the probabilistic scoring process rather than folded statically into the transition matrices stored in the database. Under this view, it is possible to allow the uncertainty associated with particular aspects of the query to be explicitly specified. For example, a user might be absolutely certain about the first and the second pitch of a melodic fragment used as a query, but less certain about the third pitch, and very uncertain about a fourth one.[9] We devised an extension of GUIDO Music Notation that allows to express such local uncertainties in a simple and intuitive way by using a the symbols "?" and "!". An instance of the example given above could thus be specified as `[g1! e1! e1? f1??]`. We are currently working on extending this concept to all melodic and rhythmic features supported by our MIR Engine.

[7]The number 32 is chosen in order to allow bit-parallel operations to be used on this data, see also [21].

[8]See [21] for an overview of MIR system and their approximate matching types.

[9]Note that this information need not necessarily be explicitly entered by the user — it could theoretically be added automatically based on a learned model of typical errors made by (particular) users, e.g., in the context of a Query-by-Humming approach.

The second stage of error-tolerant retrieval, locating approximate matches within candidate pieces, can be handled in many different ways, including standard methods based on edit-distances as well as techniques closely related to the one we discussed for finding candidate pieces in an error-tolerant way. The latter approach appears to be conceptually more elegant; we are currently developing a unified approximate retrieval mechanism based on this idea, which will be discussed in detail elsewhere. (The current implementation of our experimental Retrieval Engine contains a more ad hoc method for error-tolerant search, which we intend to replace with the theoretically more solid approach outlined above.)

5 Related Work

Over the last few years, a substantial amount of work on music database and retrieval systems has been published. While we cannot nearly cover all relevant approaches, we will outline and discuss similarities and differences between the key ideas of our approach and some recent and earlier work in the field.

Lemström and Laine recognised early that music representations which are more expressive than MIDI provide a better basis for certain retrieval tasks (see [20]; this paper also contains a nice overview of earlier work in music information retrieval). Recently, a number of musical database and retrieval systems have been developed in which music representations other (and more expressive) than MIDI are used (see, e.g., [5]); however, we believe that our use of GUIDO goes one step further than most of these in using a uniform formalism for representing pieces in the database and for formulating queries which is powerful enough to capture basically any aspect of a musical score. Although our present system only supports queries based on primary melodic and rhythmic features, we feel that the ability to extend this in a natural way to other musical concepts, such as key, metre, or barline information, is an important advantage of our approach.

Recently, a number of XML-based music representations have been proposed (see, e.g., [14; 26; 6]. While these offer some advantages by allowing the use of standard XML tools and certainly have the potential to represent arbitrary aspects of score-level music, we are not aware of any existing musical database and retrieval system based on an XML-representation. As discussed in Section 2 of this paper, XML-based representations share many desirable features with GUIDO. Aside from tool support, we cannot see any features which would make XML intrinsically suitable for content-based music information retrieval. While XML-based representations are typically much too verbose and syntactically complex to be used directly for musical queries, many aspects of our work (particularly our retrieval technique) are independent from the use of GUIDO as the underlying music representation, and can be easily applied to a broad range of other formats.

Sonoda et al. have been developing a WWW-based system for retrieving musical information from an online musical database based on the "Query-by-Humming" approach [19; 28]. Their system is based on MIDI as the underlying music representation, and their indexing and retrieval method, which uses dynamic programming for matching, has recently been optimised for efficient retrieval from large musical databases [29]. Similar to their approach, we follow the "Query-by-Example" paradigm (using GUIDO instead of MIDI) and acknowledge that matching against large databases, using dynamic programming or similar techniques, can be prohibitively inefficient, particularly in the context of an on-line system. Our probabilistic matching technique is fundamentally different from their "Short Dynamic Programming". Their technique requires very large indeces (compared to the size of the database), while our probabilistic models are relatively compact. Their retrieval technique is a rather efficient stand-alone method for finding matches in the given database.[10] In contrast, we mainly focus on a technique for identifying promising candidate pieces in the database, which can be combined with various methods for identifying matches within a given piece (e.g., dynamic programming). Another difference between their approach and ours is the fact that they focus on melodic information alone, while we support queries that can combine various melodic and rhythmic features. Evidence for the importance of supporting such combined queries is given in [8], who use a fixed time-grid for rhythmical structure (in contrast to our more flexible rhythmical query types) and a retrieval method based on inverted file indexing.

An interesting approach to music information retrieval which has recently gained some popularity is the use of text-retrieval methods on suitably encoded music representations [23]. Although our system uses a text-based music representation, our approach to music information retrieval is radically different, and actually more related to techniques for biomolecular sequence analysis and genomic information retrieval (see, e.g., [11; 4]). It is our belief that musical information is in many ways inherently different from text, and that specific properties of musical data should be exploited for music information retrieval. To that end, sequence retrieval methods developed for text data can potentially provide a valuable starting point (as has been the case for biomolecular sequence analysis), but ultimately will have to be complemented and augmented by techniques specifically developed for musical data. The probabilistic matching approach we propose provides a basis for such techniques, and the overall design of our system facilitates such extensions. Furthermore, text-based methods can be used in the context of our approach for locating matches within candidate pieces identified by our probabilistic matching technique.

[10] Since the only evaluation of their approach we are aware of is based on a database of mainly random pieces, we feel that the accuracy of the method in practice is hard to assess.

Generally, our probabilistic modelling approach is based on characterisations of the underlying musical data which can be potentially useful for purposes other than information retrieval, such as analysis or composition (see, e.g., [9]).[11] In this sense, our approach is related to work by Thom and Dannenberg [31; 30], who use probabilistic models and machine learning techniques for characterising melodies.

Finally, let us point out a general problem with almost any work on content-based music information retrieval we are aware of (including our own work presented here): the lack of a corpus of music for testing the efficiency and accuracy of music retrieval systems. Part of the reason for this is the lack of a commonly used and widely supported music interchange format. We believe that GUIDO Music Notation has the potential to remedy this situation, and we are currently working on translating various collections of musical material into GUIDO, in order to integrate these into our experimental musical database.

6 Conclusions and Future Work

In this paper we have presented the concept of a database system for structured, score-level musical information and introduced a query-by-example mechanism for retrieving information based on a variety of melodic and rhythmic search criteria. The underlying music retrieval method uses probabilistic models and a hierarchical clustering of the database for pruning and heuristically guiding the search. We also presented an extension of GUIDO Music Notation, the music representation language we use for the pieces in the database as well as for queries, which allows expressing localised uncertainty in musical queries; and we briefly described an extension of our retrieval mechanism that uses such extended queries for approximate probabilistic matching.

A first prototype of the database system and retrieval engine has been implemented and tested on a set of about 150 relatively simple musical pieces in GUIDO Notation Format. This experimental system has been equipped with a WWW interface and is available online at `http://www.salieri.org/guido/mir/`. Our experience with this small prototype suggests that the approach presented here can provide a solid foundation for larger and more complex database and information retrieval systems for structured musical data.

Conceptually as well as with respect to the implementation, this work is still in a relatively early stage, and many aspects of it will be further explored and refined in the future. On the practical side, an obvious extension of our work is to test our system and methods on larger musical databases. To that end, we have begun to include a broad range of structured musical data, including the "Essen Folksong Collection" [27] into our dataset. Finally, we hope to get access to the data coming out of Fujinaga et al.'s "Optical Music Recognition System" [7], where a large collection of American sheet music is automatically converted into GUIDO descriptions. With this additional data, we hope to be able to conduct some tests with thousands to ten-thousands of pieces in GUIDO Music Notation in the near future. We also intend to improve the integration with the experimental database/MIR system with other GUIDO tools and applications, in particular with the latest version of the GUIDO NoteServer [25] (for visualising the musical data), converters (in particular GUIDO-to-MIDI for playback), and analysis tools which are currently under development.

Another direction we would like to explore in the near future is to support queries which allow the use of GUIDO tags in addition to melodic and rhythmic information. Clearly, the information represented by tags in the GUIDO data comprising the elements of the database can be musically very meaningful, and in many contexts we consider it desirable to include such information in musical queries. This could be very useful, e.g., in order to support the specification of tonality, metre, or instrument information in a query; similarly, constraints on the metric position within bars could be expressed in queries by including barlines, and including expressive markings or dynamic information could help to make approximate queries more specific. The probabilistic matching mechanism presented here can be extended in various ways to accommodate queries including tag information, and determining a theoretically elegant and practically effective solution to this problem is a challenging problem for future research.

Even when just considering the melodic and rhythmic query types supported in our present system, it might be interesting to investigate more powerful probabilistic models as a basis for the characterisation of the musical data which is at the core of our retrieval mechanism. Obviously, higher-order Markov models could be used to capture more of the local structure, and additional statistical information which better resembles aspects of the global structure of larger pieces could be used in addition to simple Markov chains. Furthermore, larger pieces can be more appropriately handled by segmenting them into smaller fragments (using standard segmentation approaches), for which probabilistic models are then constructed individually. This way, local structure can be captured more adequately and probabilistic matching based on the fragment models will be more accurate.

Finally, we are interested in extending our approach beyond purely horizontal queries by allowing polyphonic features to be included in queries. We see two fundamental approaches for such an extension: Allowing chords and possibly tags referring to harmonic context to be included in monophonic queries, or supporting full polyphonic queries that specify simultaneous monophonic voices. We believe that our general approach should in principle be applicable to either type

[11]The simple probabilistic models used here, as well as more complex models, can be used for generating statistically similar fragments of music.

of polyphonic query, but clearly, substantial further investigation will be required to devise and implement the corresponding retrieval algorithms. Overall, we are convinced that the work presented here will provide a good basis for these and other generalised retrieval tasks.

References

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

[2] For a definition of "Query by Example" look at `http://www.whatis.com/definition/0,289893,sid9_gci214554,00.html`

[3] Bainbridge D. The Role of Music IR in the New Zealand Digital Library project. Proceedings ISMIR 00.

[4] Baldi P.; Brunak S. Bioinformatics: the machine learning approach. MIT Press, 1998.

[5] Boehm C.; MacLellan D.; Hall C. MuTaDeD'II – A System for Music Information Retrieval of Encoded Music. Proceedings ICMC 00. 174–177.

[6] Castan G. NIFFML: An XML Implementation of the Notation Interchange File Format. *in* Computing in Musicology (12). MIT Press, 2001.

[7] Choudhury G.; DiLauro T.; Droettboom M.; Fujinaga I.; Harrington B.; MacMillan K. Optical Music Recognition System within a Large-Scale Digitization Project. Proceedings ISMIR 00.

[8] Clausen M.; Engelbrecht R.; Meyer D.; Schmitz J. PROMS: A Web-based Tool for Searching in Polyphonic Music. Proceedings ISMIR 00.

[9] Dodge, C.; Jerse T. Computer Music: Synthesis, Composition, and Performance, 2nd ed., Shirmer Books: New York, 1997. 361–368.

[10] Duda R.O.; Hart P.E. Pattern Classification and Scene Analysis. New York: Wiley, 1973.

[11] Durbin; Eddy; Krogh; Mitchison Biological sequence analysis: Probabilistic models of proteins and nucleic acids. Cambridge University Press, 1998.

[12] Ghias A.; Logan J.; Chamberlin D.; Smith B.C. Query by Humming: Musical Information Retrieval in an Audio Database. Proceedings of ACM Multimedia 95, 231–236.

[13] Görg M. GUIDO/MIR – Ein GUIDO basiertes Music Information Retrieval System. Masters Thesis, TU Darmstadt, Fachbereich Informatik, 1999.

[14] Good M. Representing Music Using XML. Proceedings ISMIR 00.

[15] Grimmett, G.R.; Stirzaker, D.R. Probability and Random Processes, 2nd ed. Clarendon Press: Oxford, 1994. Chapter 6.

[16] Hoos H.; Hamel K.; Renz K.; Kilian J. The GUIDO Notation Format – A Novel Approach for Adequately Representing Score-Level Music Proceedings ICMC 98. 451–454.

[17] Hoos H.; Hamel K.; Renz K.; Using Advanced GUIDO as a Notation Interchange Format. Proceedings ICMC 99. 395–398.

[18] Hoos H.; Hamel K.; Renz K.; Kilian J. Representing Score-Level Music Using the GUIDO Music-Notation Format. Computing in Musicology, Vol 12, Editors: W.B.Hewlett, E.Selfridge-Field; MIT Press, 2001.

[19] Kageyama T.; Mochizuki K.; Takashima Y. Melody Retrieval with Humming. Proceedings ICMC 93. 349–351.

[20] Lemström K.; Laine, P. Musical Information Retrieval using Musical Parameters. Proceedings ICMC 98, 341–348.

[21] Lemström K.; Perttu S. SEMEX – An Efficient Music Retrieval Prototype. Proceedings ISMIR 00.

[22] Melucci M.; Orio N. The use of melodic segmentation for content-based retrieval of musical data. Proceedings ICMC 99. 120–123.

[23] Pickens J. A Comparison of Language Modeling and Probabilistic Text Information Retrieval – Approaches to Monophonic Music Retrieval. Proceedings ISMIR 00

[24] Rabiner L.R.; Juang B.H. An introduction to hidden Markov models. IEEE ASSP Magazine, January 1986, 4–15.

[25] Renz K.; Hoos H. A WEB-based Approach to Music Notation using GUIDO. Proceedings ICMC 98. 455–458.

[26] Roland P. XML4MIR: Extensible Markup Language for Music Information Retrieval. Proceedings ISMIR 00.

[27] Schaffrath, H. The Essen Folksong Collection in the Humdrum Kern Format. D. Huron (ed.). Menlo Park, CA: Center for Computer Assisted Research in the Humanities, 1995.

[28] Sonoda T.; Goto M.; Muraoka Y. A WWW-based Melody Retrieval System. Proceedings ICMC 98. 349–352.

[29] Sonoda T.; Muraoka Y. A WWW-based Melody-Retrieval System – An Indexing Method for a Large Melody Database. Proceedings ICMC 00. 170–173.

[30] Thom B. Learning Models for Interactive Melodic Improvisation. Proceedings ICMC 99. 190–193.

[31] Thom B.; Dannenberg R. Predicting Chords in Jazz. Proceedings ICMC 95. 237–238.

A Naturalist Approach to Music File Name Analysis

François Pachet
Sony CSL-Paris
6, rue Amyot
75005 PARIS
France
pachet@csl.sony.fr

Damien Laigre
Sony CSL-Paris
6, rue Amyot
75005 PARIS
France
dlaigre@csl.sony.fr

Abstract:
Music title identification is a key ingredient of content-based electronic music distribution. Because of the lack of standards in music identification – or the lack of enforcement of existing standards – there is a huge amount of unidentified music files in the world. We propose here an identification mechanism that exploits the information possibly contained in the file name itself. We study large corpora of files whose names are decided by humans without particular constraints other than readability, and draw various hypotheses concerning the natural syntaxes that emerge from these corpora. A central hypothesis is the local syntactic consistency, which claims that file name syntaxes, whatever they are, are locally consistent within clusters of related music files. These heuristics allow to parse successfully file names without knowing their syntax *a priori*, using statistical measures on clusters of files, rather than on parsing files on a strict individual basis. Based on these validated hypothesis we propose a heuristics-based parsing system and illustrate it in the context of an Electronic Music Distribution project.

1 Introduction

The recent progress of digital audio technologies and the availability of easy and cheap Internet access have led to the proliferation of music files on the planet.
Efficient digital audio compression format such as mp3 have made possible the distribution of music on a large scale, using all sorts of broadcasting techniques and supports, such as peer-to-peer communication systems. This proliferation of music data around the globe is not incidental, and may be seen as a sign of the huge pressure for Electronic Music Distribution (EMD) from the community of music listeners.
EMD, however, is more than just representing music as audio files. Confronted to large databases, users can only access what they know, and content-based management techniques are acknowledged to be a necessary ingredient to fulfil the target of true, personalized music distribution. Content-based music access requires, between other things, the ability of extracting features from the signal, of gathering descriptions of various source of textual information, of modelling user profiles and matching these profiles to music descriptors, etc. (see Pachet, 2001a for a survey). Among these requirements, one key issue is music identification: how to identify in a non-ambiguous way music files. This identification is crucial to allow the management of metadata, copyrights, profiles, recommendation systems, etc. Without a solid identification mechanism, EMD may well turn into a gigantic and serendipitous adventure for users, content providers and distributors.
Various standardization efforts have been conducted to define universal codes for music titles. The most famous is probably the ISRC (International Standard Recording Code), developed by ISO (ISO 3901) to identify sound and audio-visual recordings. ISRC is a unique identifier of each recording that makes up an album. Unfortunately it is not followed by all music production companies, and hardly used in unofficial music sources such as peer-to-peer communication systems.
Another problem is that, even when a code could be used, it is not: for instance, digital music encoded in the audio CD format usually does not contain information on the music identification. Strangely enough, it is not possible to get the track listing information from a CD. External databases of track listings for commercial CDs have been developed, such as CDDB. CDDB works by associating track-listing information to audio signatures of CDs. To allow scaling up, CDDB is a collective effort: the database is made up by the users themselves. While this collaborative aspect does allow scaling up (there are more than 4 millions CDs registered on CDDB), there is an obvious drawback to this enterprise: the track listing information is not guaranteed, which leads to many errors, duplications and to the difficulty of identifying correctly music titles.
There are other sectors of the music production chain that are concerned with music title identification, such as radios (which display their track listing on Internet for instance) or copyright associations (which have to keep track of broadcasted titles to compute the payment of royalties). In each case, ad hoc and proprietary schemes have been devised, but there is no convergence of music identification methods.
There are several approaches to the identification of audio music sources. The most straightforward one consists in analysing the signal, typically a portion of the whole music title, to extract an audio signature. This signature is then matched against a database of pre-recorded music signals. This task is, for instance, addressed by technologies such as Broadcast Data Systems (US) or MediaControl (Germany), and is used

by copyrights management companies to infer radio play lists. The techniques used to perform the identification range from usual pattern matching to more elaborate statistical methods based on characterization of the evolution of spectral behaviours. In all cases, the identification requires a database of all music files created beforehand. Such a global database is far from realistic in the near future sot the approach can work only within limited contexts.

Another approach consists in exploiting external information about the music source. For instance, the Emarker system (Emarker, 2001), exploits the geographical and temporal location of a radio listener requesting a song, and then queries a large database containing all radio stations programs by time and location. The approach is of course much lighter than the signal based approach since no signal processing is required, and can scale-up to recognize virtually any number of titles. It works of course only for titles played on official radio stations.

In this paper we describe another approach, more suited to personal music file management systems, for which no radio track listing is available for identification, and which does not require the management of a global, universal database of music titles. This approach is based on the analysis of actual music file names.

More precisely, we consider the context of popular music titles, and therefore seek to identify two main information for a music source: the artist (or performer) identification, and the actual name of the music title. We consider music file names coming from natural sources, such as personal hard disk drives (usually filled with audio files coming from peer-to-peer communication systems), track listing databases (such as CDDB), or radio track listings. In all these cases, the file names are input by users who do not follow any constraint, other than human readability.

We consider here information contained in music file names, and not identification from the signal, or from other external sources of information (such as ID tags in mp3 files, see Hacker, 2000). These other methods are orthogonal to the method proposed here. In an ideal case, music identification could exploit all these methods collaboratively.

We will first introduce the context of our study, and the corpora analysed (Section 2.1). We then propose several assumptions for guiding the analysis process, the main assumption being a local consistency assumption (Section 2.2). We perform a statistical analysis of these corpora to validate the assumptions and draw corresponding heuristics. Finally, we describe FNI, a system that implements our heuristics, and illustrate how it performs in the context of a real world Electronic Music Distribution system developed at Sony CSL, within the European CUIDADO IST-funded project.

2 Popular Music file names

Music file names may contain various types of information about a music title. In our context we focus on popular music, for which two information are of interest: the artist or interpreter identifier, and the actual title name. In some cases, file names can also contain other information such as the album or track number. In the case of Classical music, the notion of artist is more complex, and identification may contain both composer and performer identifier. Additionally, various identifiers may also be present, such as the version (instrumental, remix, etc). Several statistical approaches have been proposed to parse text automatically into coherent segments, corresponding for instance to different topics in news transcripts (see e.g Beeferman et al., 1999). In our case, the textual data considered is much shorter and the domain (music works) is narrower, so we show that it is possible to derive heuristics to implement an efficient parsing system without a learning component, at least as a first approximation.

2.1 Corpora studied

Music files names are typically found in the following locations: 1) personal storage systems such as hard disks, 2) radio program track listings and 3) repositories of musical metadata. For the purpose of our study, we identified three such databases: a subset of 22, 302 album track listings from the CDDB database containing track listings for about 4 millions CD albums, 2) a 1-year listing of a radio station broadcasting music in a large variety of styles (Fip/RadioFrance) and 3) a set of file listings (about 3000 files) of personal hard-disks of intensive users of peer-to-peer music communication systems.

These three cases share a common characteristic: the file names they contain have been specified by individuals on whom no particular syntactic constraint was enforced, other than human readability, i.e. the fact that these names should be understood easily by other individuals of the same community. The individuals name their files as they wish, and these personal conventions are simply spread through the community without modification. In CDDB, the principle of the database is collaboration: albums track-listings are given by the users themselves. Although the editors for entering track-listing information may in some case force some structure (e.g. differentiate title and artists), there is no unique syntax valid for all track listings, as illustrated below. In the case of radio stations broadcasting their programs, there may more be cohesion since these programs are entered by a smaller number of individuals, but, similarly, the syntax will not be constant, and will differ from a radio station to another one. However, the case of radios is simplified by the fact that the syntax of file names is usually constant for a given radio.

To illustrate our study, we give below some typical examples of file names coming from the sources at hand.

```
The Rolling Stones - Angie
The Beatles - Oh! Darling
Eagles - Hotel California
Simon & Garfunkel - The Sound Of Silence
Kansas - Dust In The Wind
America - The Last Unicorn
Creedence Clearwater Revival - I Put A Spell On
You
The Beatles - Let It Be
The Tremeloes - Silence Is Golden
Hollies - He Ain’t Heavy He’s My Brother
Zz Top - Blue Jeans Blues
Simon & Garfunken - El Condor Pasa (It I Could)
Bee Gees - Massachusetts
```

```
Omega - The Girl With The Pearl’s Hair -
Featuring Gabor Presser, Ann
```

Figure 1. File names found on the CDDB database, for an album entitled *“Golden Rock Ballads V.1”*

```
d:\mp3\CSL2-1\Various - Animals - The house of
the rising sun.mp3
d:\mp3\CSL2-1\Various - The Mindbenders - A
groovy kind of love.mp3
d:\mp3\CSL2-1\Various - Hollies - The Air That I
Breathe.mp3
d:\mp3\CSL2-1\Various - The Beatles - Ain’t she
sweet.mp3
d:\mp3\CSL2-1\Various - Bee Gees -
Massachusetts.mp3
d:\mp3\CSL2-1\Various - The Moody Blues - Nights
in white satin.mp3
d:\mp3\CSL2-1\Simon and Garfunkel – El Condor
Pasa (If I Could).mp3
d:\mp3\CSL2-1\Simon and Garfunkel – The Sound of
Silence.mp3
d:\mp3\CSL2-1\Bee Gees - Saturday Night
Fever.mp3
d:\mp3\CSL2-1\Beastie Boys - Song for Junior.mp3
d:\mp3\CSL2-1\Beach Boys - Good Vibrations.mp3
d:\mp3\CSL2-1\01 - The Beatles - Doctor
Robert.mp3
d:\mp3\CSL2-1\05 - The Beatles - Sgt Pepper’s
Lonely Hearts.mp3
d:\mp3\CSL2-9\Various - Rock F.M\Original Rock
N°5 - Crack The World Ltd - Fine Young Cannibals
- She Drives Me Crazy.mp3
d:\mp3\CSL2-9\Various - Rock F.M\Original Rock
N°5 - Crack The World Ltd - The Beach Boys - I
Get Around.mp3
d:\mp3\Jazz\STAN_GETZ\MENINA_MOCA.mp3
d:\mp3\Jazz\STAN_GETZ\SAMBA_DE_UMA_NOTA_SO.mp3
```

Figure 2. File names found on a personal hard disk.

```
17:54 OH DARLING, THE BEATLES
      ABBEY ROAD (1969 EMI)
17:57 I BELONG TO YOU, LENNY KRAVITZ
      5 (1998 VIRGIN)
18:01 FATIGUE D ETRE FATIGUE, LES RITA MITSOUKO
      COOL FRENESIE (2000 DELABEL)
18:09 IT AIN T NECESSARILY SO, MILES DAVIS
      BESS (1958 CBS)
18:14 ENTRE VOUS NOUVIAUX MARIES, ALLA FRANCESCA
      BEAUTE PARFAITE / ALLA FRANCESCA (1997
OPUS 111)
18:16 FOR EMILY WHENEVER I MAY FIND HER, SIMON
AND GARFUNKEL
      COLLECTED WORKS (1966 CBS)
```

Figure 3. A typical radio program on Fip/Radio France.

```
9:28 Bach: Concerto #4 in A, BWV 1055 (Glenn
Gould, piano, Columbia SO/Golschmann) CBS 38524
9:50 Bach/Manze: Toccata & fugue in d, BWV 565
(Andrew Manze, solo violin) Harmonia Mundi
907250.51
10:04 Jaromír Weinberger: Polka & fugue from
Schwanda the Bagpiper (Philadelphia O/Ormandy)
Sony 63053
10:21 Shostakovich: Piano concerto #2 in F,
Op.101 (Mikhail Rudy, St. Petersburg PO/Jansons)
EMI Classics 56591
10:49 Dvorák: Bagatelles, Op.47 (Takács Qrt.)
London 430 077
11:14 Falla: El sombrero de tres picos (Three-
Cornered Hat), part 1 (Jennifer Larmore, Chicago
SO/Barenboim) Teldec 0630-17145
```

Figure 4. Another typical radio program on WFCR/Western New England.

2.2 Clusters

An important remark to be made is that the music files considered are usually organized in different levels. In CDDB, there is only one level which is the album, itself containing tracks. On personal hard disks, there may be any number of levels, represented by the directory structure of file systems. For the sake of generality, we consider that the database of file names is structured by clusters – possibly – recursively. Clusters may contain either other clusters of file names.

As we can see, there is no universally valid syntax, either at the lexeme level (morphology of informative elements) or the music file level (actual syntax). However, these file names are not totally random, and some regularities can be identified, in particular at the cluster level. In the next section, we examine more closely the regularities found in these various sources, from which we will draw a set of heuristics for an automatic file name recogniser.

2.3 An Empirical Analysis

A manual analysis of a subset of our databases was performed, to identify the most salient characteristics of file names. This manual analysis of some examples yields a number of regularities:

1) Regularities at the file name level. There is a small number of delimiters that are used for separating artist and title information. Based on these delimiters, there are some syntaxes with a higher degree of probability than others. For instance: “artist – title “ such as “The Beatles **-** Oh! Darling”, “title – artist” such as “Oh Darling, The Beatles”, or “constant term – artist – title” such as “Various **-** The Beatles **-** Ain't she sweet”, etc.
2) Regularities at the word level. Artist names are usually found under a restricted number of syntactic forms, such as: “Paul McCartney”, “McCartney, Paul”, “Mc Cartney”, or “The Beatles”, “Beatles, the”, “Beatles”.
3) Most importantly, regularities at the *cluster level*. It appears that syntaxes, as cumbersome as they may sometimes be, are not distributed uniformly: within a cluster, it is often the case than all titles follow the same syntax, or, at least, a small number of syntaxes. This remark is at the core of our proposal, as we will see below.

Based on these remarks, we propose the following four hypotheses relative to music file name analysis:

Delimiter Hypothesis:
This hypothesis states that the artist and title name information are indeed separated by delimiters, which are special characters within a given, small set of characters. As a special case, we consider that a file name using no separators is a title name without reference to its artist.

Constant Term Hypothesis:
Several syntaxes may contain constant terms, which are not directly relevant. A constant term can be for instance the album name, a date, or key words such as “Various Artists” (see Figure 2). The notion of constant term here is augmented by integrating possibly varying numerals, to

handle cases such as track numbers ("Track 1," Track 2", etc. see Figure 2).

Word Morphology Hypothesis:
Artist names and title names have statistically different morphologies. For instance, the number of words for artist names is less important than the number of words used for title names. Additionally, artist names often make use of a limited number of specific heuristics related to first name (McCartney, Paul" is the same than "Paul McCartney" or "McCartney, P."). These heuristics may be used to determine whether a piece of information denotes an artist or a title name.

Local Syntactic Consistency Hypothesis
This hypothesis asserts that syntaxes of file names are consistent within a given cluster (what we call a syntax will be defined more precisely below). In reality, the hypothesis is weakened by the fact that this consistency may not actually occur entirely within a cluster. For instance, Figure 2 shows a directory listing containing four main syntaxes (for a total 13 titles, which is indeed an extreme case). We weaken this hypothesis by considering sub-clusters sharing the same syntax, and showing that only a small number of sub-clusters is needed – in general – to perform the analysis correctly.

In the next Section, we show the results of an automatic analysis performed on our databases to assess the validity or our hypothesis.

3 Statistical Analysis of File Name Corpora

3.1 *Delimiter Hypothesis*

We call here a delimiter a character used to separate different type of information in a given segment. The hypothesis states that there are indeed delimiters: these special characters are - most often - used as separators, rather than significant characters for artists or title names. The most encountered delimiters in the corpora are the following: '-', '/', '(', ')', '[', ']', '{', '}', ';', ':'.
To validate the Delimiter Hypothesis, we have to show that the file names use delimiters to separate artist and title information. To do this systematically would require a thorough check of over 300.000 titles, which is too hard a task to be done manually. Instead, we show here that delimiters are used in a consistent manner within each cluster. Although this check does not guarantee that delimiters are indeed used to separate, e.g. artist and title information, it does a give strong indication that there is a consistent use of these characters as syntactical elements rather than significant characters.
More precisely we call "common delimiter" a character delimiter found in *all* the segments of a given cluster. This delimiter indicates in most of the case a separation between different information types. As the following table shows, many (64.4%) though not all clusters have one common delimiter. Some clusters have no delimiters (7.2%), which corresponds to cases where the file name only contains the title information (the artist name is then most often contained in the album name for CDDB, or in the super directory for personal files). In the remaining cases, several delimiters are found in given clusters. We then look for the minimum number of delimiters that "cover" the whole cluster. What the table shows is that there is, in most of the cases, a small number of such covering delimiters, which is once again a strong indication that these delimiters are used for syntactical purposes.

	Nb clusters:	**Percentage:**
no delimiter:	1615	7.2 %
1 common delimiter :	14354	64.4 %
2 delimiters cover the cluster :	4763	21.3 %
3 delimiters cover the cluster :	1338	6.0 %
4 delimiters cover the cluster :	215	1.0 %
5 delimiters cover the cluster :	17	0.1 %
Total :	22302	100.0 %

Figure 5 Analysis of delimiters in our CDDB play lists.

3.2 *Word Morphology Hypothesis*

The word morphology hypothesis asserts that artist names are shorter on average than title names. Although this hypothesis is certainly not always true (e.g. the group named "Everything but the girl" has recorded a song named "Angel"), it is true in average, and in particular within clusters.
An analysis of about 17,000 titles from CDDB yields an average of 1.6 words per artist names against an average of 3.2 words for title names, i.e. a ratio of 2 times more words in artist names. Similarly, an analysis of 19,648 titles from the FIP radio program yields 2.1 words for artist names against 3.1 words for the title names, i.e. a ratio of 1.5.
This shows clearly that titles names are, on average, longer than artist names. As we will show below, this heuristic may be used when no other clue allows to infer whether a string is an artist or title name.

3.3 *Constant Term Hypothesis*

The constant term hypothesis asserts that clusters may happen to contain constant terms in *all* their segments. These constant terms can refer for instance to the artist name, but also to information which is useless in our context.
The analysis of our CDDB database yields 800 constant terms, of which about 20% are not artist names. As an indication, here are the 10 most frequent useless constant terms retrieved from this list:

```
Sampler
Various Artists
Various
Passion
Unknown
Fabulous
Success
Dreams
Memories
Mixery
```

Figure 6 Most Frequent constant terms in CDDB play lists

These constant terms are used in our system to differentiate between useless information that can be discarded from useful information such as artist names.

3.4 Local Syntactic Consistency Hypothesis

This hypothesis is the most important in our study, since it will allow us to determine the syntax according to the analysis of a group of titles, rather than individual titles only. To validate this hypothesis, we need to estimate the average number of different syntaxes a cluster contains.
To do so we introduce the notion of syntax as follows. For a given file name string, we replace all the token strings encountered by an alphabetic letter incremented automatically (a, then b, then c, etc.) and we replace all numbers by a digit (0, 1, 2; etc.). We let the delimiters unchanged. The resulting string may be seen as a canonical representation of the syntax of the file name.

Here are some examples of file names and their associated syntax as extracted by our analysis:

File name	Syntax
`Various - Bee Gees - Massachusetts.mp3`	a-b-c
`Simon and Garfunkel – El Condor Pasa (If I Could).mp3`	a-b(c)
`The Beatles - 05 - Sgt Pepper’s Lonely Hearts Cl.mp3`	a-0-b
`Original Rock N°5 - Crack The World Ltd - The Beach Boys (I Get Around).mp3`	a-b-c(d)
`All you need is love.mp3`	a

Figure 7 Canonical syntaxes for various music file names.

As an illustration of the process, here are the most frequent syntaxes retrieved (in number of lines):

a	69277	a-b	66637
a/b	64584	a(b)c	30569
a-b(c)d	15351	a/b(c)d	19191
a:b	5561	a(b)-c	4198
a-b-c	3050	a/b-c	2508
a(b)/c	1959	a,b	1918
a-b/c	1708	a(b-c)d	1319
a[b]c	1317	a--b	1128
a/b,c	954	a,b/c	930
a:b(c)d	906	a-b,c	727
a-b[c]d	703	a:b-c	673
a,b-c	589	a/b[c]d	556
(a)b	524	a/b(c-d)e	517
a-b-c(d)e	506	a(b)(c)d	500

Figure 8 Main syntaxes found in our CDDB play lists

Once syntaxes are extracted, we compute, for each cluster, the number of different syntaxes it contains. This computation simply consists in comparing syntaxes using string comparison operators. The following table shows the result of this computation.

	Nb clusters:	**Percentage :**
1 common syntax :	4871	21.8 %
2 syntaxes in the cluster :	7702	34.6 %
3 syntaxes in the cluster :	5160	23.1 %
4 syntaxes in the cluster :	2691	12.1 %
5 syntaxes in the cluster :	1162	5.2 %
6 syntaxes in the cluster :	409	1.8 %
7 syntaxes in the cluster :	180	0.8 %
8 syntaxes in the cluster :	51	0.2 %
9 syntaxes in the cluster :	27	0.1 %
Over 9 syntaxes:	49	0.2 %
Total:	22302	100.0 %

Figure 9 Analysis of syntaxes in our CDDB play lists

These results clearly show that there is indeed a syntactic consistency in most of the clusters encountered. This consistency, in turn, will be used to parse file names according to the most prominent syntax within clusters, as shown in the next section.

4 The FileNameInterpreter (FNI) System

The hypotheses we made and validated have been exploited to design and implement a file name interpreter, in the context of an EMD application. This application is part of *Cuidado*, a large European project for content-based music access (see Pachet, 2001b). In this section, we describe the overall design of this system, and show its performance on real world examples.

4.1 Overview

The input of our system is a file containing a structured list of file names. The output is a file containing the analysed artist and title name information. This analysis is performed by applying heuristics as described below. To allow flexibility, the user always has the possibility to correct manually the analysis proposed, and this correction is then substituted to the analysis in the output file, and retrieved in later analysis to avoid repeating corrections.

4.2 Initialization

A pre-processing phase is applied systematically to the input list of music file names. This pre-processing consists in:

1) Grouping together file names having the same syntax into sub-clusters,
2) Chunking related file names into segments according to delimiters.

For instance, if we consider the input file as given in Figure 2, considering only the first cluster we obtain the following:

1) The syntaxes found in this corpus are: `"a-b-c", "0-a-b", "a-b", "a-b(c)"`.

2) The lists relative to the syntaxes are then the following ("|" indicates separation between recognized segments):

Syntax: a-b-c
```
Various | Animals | The house of the rising sun
Various | The Mindbenders | A groovy kind of love
Various | Hollies | The Air That I Breathe
Various | The Beatles | Ain't she sweet
Various | Bee Gees | Massachusetts
Various | The Moody Blues | Nights in white satin
```

Syntax: 0-a-b
```
01 | The Beatles | Doctor Robert
05 | The Beatles | Sgt Pepper's Lonely Hearts
```

Syntax: a-b
```
Simon and Garfunkel | The Sound of Silence
Bee Gees | Saturday Night Fever
Beastie Boys | Song for Junior
Beach Boys | Good Vibrations
```

Syntax: a-b(c)
```
Simon and Garfunkel | El Condor Pasa (If I Could)
```

Each of these three sub clusters is now treated using the implementation of the heuristics as described below.
In the next sections, we consider each sub cluster as an array. The lines of the array match the lines of the sub cluster, and the columns of the array match the segments in each line of the sub cluster.

4.3 Management of Identifiers

In order to take into account differences in the spelling of Proper names (artists) and title names, we implement retrieval mechanisms based on a canonical representation of identifiers. This representation is computed so that different spellings of a given identifier yield the same representation.
The principle is to build a unique String composed only of the significant characters of a given identifier, removing blanks, spaces, and non-standard characters.

Additionally, there is a specific provision for managing artist names: artist names may have several attributes such as "firstName", or "group prefix" (e.g. "The" or "Les" in French). These attributes are specified in a lazy mode, that is as they are encountered.
For instance, the first time we encounter the artist spelled as "McCartney, Paul", we create an entry in the artist directory, with a canonical representation being "mccartney", a first name being "paul".
When we encounter another occurrence of McCartney, but with a different ordering or spelling, such as "McCartney" or "Paul McCartney", we are able to retrieve the previously entered occurrence by trying several all the possible combinations.

Lastly, this specific procedure is augmented with a fuzzy matching algorithm to take into account possible misspelling and errors. This procedure is not discussed here for reasons of space.

4.4 Implementation of the heuristics

We describe here how we implement and prioritise the different heuristics to infer the artist and title information from a given sub cluster in which all titles share the same syntax. We do not describe the whole analyser here, but only highlight its main structure.

4.4.1 Case 1, implicit information

If the sub cluster contains only one segment, the only hypothesis we can make is that 1) the segment denotes the title name, and 2) the artist information is contained in the super cluster (super directory usually). For instance, if the corpus is a directory from a personal database of music file names, the artist name can be the name of the directory containing the music files. This is the case with the 2 "Stan Getz" files in Figure 2 for instance.

4.4.2 General Case

As illustrated in Section 2.2, about 93% of the play lists analysed from CDDB have at least two segments. We therefore assume that these segments contain at least both the title name and the artist name. The problem is now to determine which segment is the artist name, which one the title name, and which ones are useless groups of words such as constant terms, dates, etc.

Here is the ordering of the heuristics to identify properly the artist and title information.

1) Constant term heuristics,
2) Artist names heuristics,
3) Title name heuristics.

4.4.2.1 Constant Terms Heuristics

This heuristics is applied only if the syntax of the sub cluster considered contains at least two segments.
We first check if the array contains any constant terms in a whole column. If a column contains the same constant term, there are two possible interpretations:

- The array contains two columns: the column containing the constant terms is assumed to be the artist name column.
- The array contains more than two columns: we must check if the constant term belongs to a list of known constant terms as illustrated in section 3.3. The list of well-known constant terms we use has been retrieved from our CDDB database. We cannot determine whether or not a constant term is an artist name if it does not belong to our list. If the constant term belongs to our list of constant terms, we will not take into account the column relative to this constant term anymore in the title identification and consider that the artist and title names are the remaining columns of the array.

4.4.2.2 Artist Names Heuristics

To determine if a column is an artist name, we consider the following information in the following order:

1) Number of comas.
One heuristic is to consider that the column containing the most commas in its strings is the artist names column. Indeed, even if the percentage of cases where the artist name is written with a comma (ex: “Beatles, the”, “Mc Cartney, paul”) is not very high, this is a first way to retrieve the artist name.

2) Known artists
Then, if the artist column has not been found, we propose to take into account the artists already known by the system. If a known artist is found in a column, this is the artist column, following our local consistency hypothesis (all the artist names are in the same column).

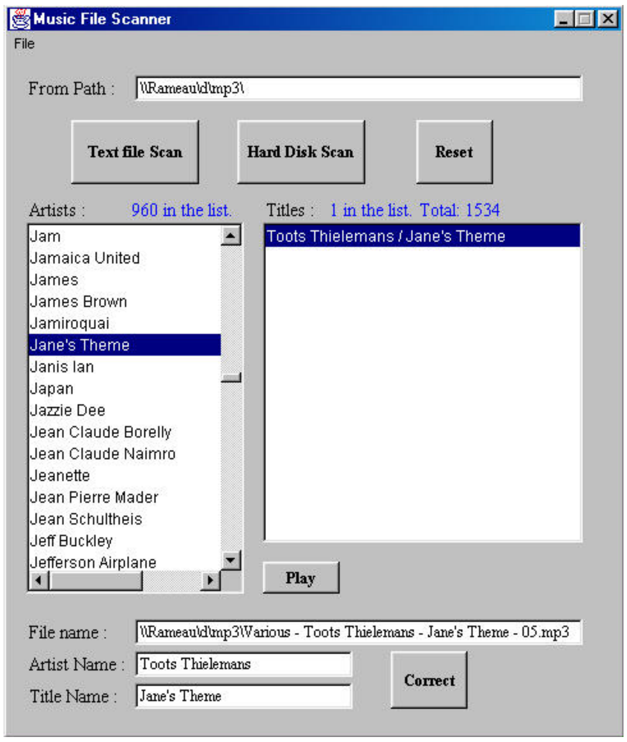
3) Number of different words
Once the elimination of columns containing useless terms has been performed and if there are only two columns left in the array considered, we count the number of different words in all the valid columns of the array. If the number is smaller in one of the columns, we assume this column represents the artist names.

4.4.2.3 Title Name Heuristics

At this step of the identification, the column of the array containing the title names may be inferred in most of the cases by elimination since there is most often only one column remaining.
However, if we have more than one column, we apply again the heuristics about the number of words: if the number of different words is greater in one of the columns, we considered it as the title names column.

5 Experimentations

Our system has been tested and validated on our three databases. To validate the system, we made about 1500 random experiments, by drawing a random title, and checking manually whether the parse was correct or not. 95 % of the cases where correctly analysed. We assume the most frequent cases have been encountered.


Music File Scanner
File
From Path : \\Rameau\d\mp3\
Text file Scan | Hard Disk Scan | Reset
Artists : 960 in the list.
Titles : 1 in the list. Total: 1534
Jam
Jamaica United
James
James Brown
Jamiroquai
Jane's Theme
Janis Ian
Japan
Jazzie Dee
Jean Claude Borelly
Jean Claude Naimro
Jeanette
Jean Pierre Mader
Jean Schultheis
Jeff Buckley
Jefferson Airplane

Toots Thielemans / Jane's Theme
Play
File name : \\Rameau\d\mp3\Various - Toots Thielemans - Jane's Theme - 05.mp3
Artist Name : Toots Thielemans
Title Name : Jane's Theme
Correct

Figure 10. The interface of FNI.

The incorrect cases are most often non interpretable file names. For instance, as illustrated in Figure 10, “Various - Toots Thielemans - Jane's Theme - 05” has too many segments. The “05” is duly recognized as a constant term, but the system cannot determine which segment refers to the title name and which one refers to the artist name. In this case, even a human could not infer the right syntax, unless he/she would know the track listings and albums names of Toots Thielmans.
A few cases were not correctly analysed because the syntax exceptionally did not match our heuristics. Example: “Johnny Lee Hooker - Boom, Boom.mp3”. The artist name has more words than the title name, and the title name contains a comma. However our system allows to correct manually the wrong file names (see Figure 10). Additionally, the list of “known artists” is updated automatically, so mistakes are only done once.
FNI is integrated in *Personal Radio*, a working EMD system that has already been tested on over 100 users. More tests are being conducted within the Cuidado European project (Pachet, 2001b).

6 Conclusion

We described a method for parsing music file names without knowing their syntax *a priori*. The method is based on a set of justified heuristics which are validated by a prior analysis of a large corpora of “natural” file names, and by a working system integrated in a large scale EMD project. The success of the approach lies mainly in the local consistency hypothesis, which states that syntaxes are usually consistent within related groups of music files. This hypothesis allows to solve a number of ambiguity by making choices based on statistical properties of file clusters rather than on individual files. Extensions of the approach for handling other types of music (e.g. Classical) or non-Western filenames are under study, and may require different sets of heuristics, but we believe the approach in general is still valid. Lastly, we plan to integrate a learning module to FNI that is able to learn automatically new syntaxes from errors, in the spirit of (Petasis et al., 2001).

7 References

Beeferman, D. Berger, A. Lafferty, J. (1999) Statistical Models for Text Segmentation, Machine Learning, 34, 1-3, Feb. 1999.

Hacker, Scott (2000) *MP3, the definitive guide,* O'Reilly.

Maurel, D. Piton, O. Eggert, E. (2001) Automatic Processing of Proper Nouns Vol. 41 N.3, February 2001.

Pachet, F. (2001a) Content management for Electronic Music Distribution: The Real Issues, submitted to Communications of the ACM, 2001.

Pachet, F. (2001b) Metadata for music and sounds: The Cuidado Project, Content-Based Multimedia Indexing Workshop, Brescia (It).

Petasis, G. Vichot, F. Wolinski, F. Paliouras, G. Karkaletsis, V. Spyropoulos, C. (2001) Using Machine Learning to Maintain Rule-based Named-Entity Recognition and Classification Systems, Association for Computational Linguistics, ACL.

SCORE PROCESSING FOR MIR

Donncha S. Ó Maidín
Centre for Computational Musicology and Computer Music
Department of Computer Science and Information Systems
University of Limerick
353(0)61202705
donncha.omaidin@ul.ie

Margaret Cahill
Centre for Computational Musicology and Computer Music
Department of Computer Science and Information Systems
University of Limerick
353(0)61202759
margaret.cahill@ul.ie

ABSTRACT

The focus of this paper is on the design and use of a music score representation. The structure of the representation is discussed and illustrated with sample algorithms, including some from music information retrieval. The score representation was designed for the development of general algorithms and applications. The common container-iterator paradigm is used, in which the score is modelled as a container of objects, such as clefs, key signatures, time signatures, notes, rests and barlines. Access to objects within the score is achieved through iterators. These iterators provide the developer with a mechanism for accessing the information content of the score. The iterators are designed to achieve a high level of data hiding, so that the user is shielded from the substantial underlying complexity of score representation, while at the same time, having access to the score's full information content.

1. INTRODUCTION

The focus of this paper is on representing music scores. The music score is the primary document for practically all music of the past. It holds a primary place in literacy, in education, in composition and performance and in music theory.

In the computer era, two main sources for digital versions of scores arise. The first of these is from digitising initiatives, such as those at Center for Computer Assisted Research in the Humanities at Stanford University. The second comes as a by-product of music publishing. These activities have resulted in the production quantities of machine-readable scores. Unfortunately, not all of these efforts are readily usable in IR research. Lack of agreed open standards and lack of openness on the part of notation software developers form some of the main barriers to more general use.

In practice MIDI has become the representation used in much of music information retrieval research. The MIDI standard was invented to capture the gestures of a keyboard player. Its ability to provide the pitch and duration content of music has resulted in its acceptability for music research. However basic MIDI representation is radically different from score representation. Rests, slurs, barlines, staccatos, trills, ornaments, triplets and chromaticisms, are examples of concepts that are not explicit in MIDI.

An alternate prospect to using MIDI is that of having an availability of music scores, encoded to professional editorial standards, together with appropriate tools. One consequence of this approach is that the music IR researcher may work with the full information content of a score, rather than a simplified view of pitch and duration. Additionally this approach will serve to facilitate cooperation between the music IR researcher and music theorists, who deal primarily with the score.

2. CONTAINER-ITERATOR

Some endeavours in Computer Science have been concerned with discovering useful ways for organising collections of data. One of the most basic ways of organising data is to conceive of a complex object as a collection of objects that are included in a **container**. Objects within the container are accessed by means of iterators. Iterators are often characterised as 'safe' pointers – that is that they can point to objects within a container, and can be safely manipulated, or moved about so as to make all of the internal objects accessible. The success of this approach has led to the development of many libraries, the most widely used one being the C++ Standard Template Library that was developed at Hewlett-Packard Labs by Alexander Stepanov and Meng Lee[1][2] and was adapted as part of ANS and ISO standards in the 90's. The containers in the STL allow the user to structure the data as vectors, lists, deques, sets, maps, stack and queues. Most of these data structures are one-dimensional. Their associated iterators are built so that all objects in the more general containers may be accessed in a left-to-right fashion, and possibly in a reverse order. Additionally some iterator/container combinations allow random access to the contained objects.

It is appropriate to consider how the music score may be modelled as a container. Items in a score may be represented as objects within the container. Objects are used to represent notes, barlines, key and time signature. Iterators allow the software developer access to the information content of a score.

A strict adherence to the S.T.L. model has proved inappropriate for score representation and manipulation. In S.T.L., one of the main functions of iterators is to allow access to container members. In C.P.N.View, the iterator is used to carry out the substantial scoping resolution. Automatic scorping resolution is essential in order to achieve an appropriate level of abstraction or complexity hiding. Iterators must be able to randomly access objects (e.g. the uppermost object half ways through bar no 22 in the second violin line). Iterators may be required to move vertically, to access harmony or horizontally to follow a melody. The iterator must, at the same time keep track of scoping information about aspects of the current context such as clef, key signature, time signature, metronome settings, tempo indications, accidental alterations and location within a bar. This is necessary in order to free the user from the considerable scoping complexities, that would otherwise tend to get in the way of the development.

3. COMMON PRACTICE NOTATION VIEW

C.P.N.View[3][4] is a score representation written in C++, that was developed in the mid-1990s. It implements a representation of scores as containers and provides iterators for use with algorithms. A score object is created either algorithmically or by using one of the components for converting from various file-based representations (ALMA, *kern, NIFF, EsAC).

Creating a score object:

Score s(filename);

One or more score iterators may be created in order to gain access to the internal objects in the score as

ScoreIterator si(s);

For the initial examples we will assume that the score is monophonic.

A number of functions exist to move the score iterator about the score. Random access is achieved using the **locate** member function. This moves the iterator to an arbitrary place in the score

si.locate(NOTE, 23);

will move the iterator **si** to the 23rd note of the score. This function returns a TRUE/FALSE value to signal the success or otherwise of the operation. For example if we call this function on a score that has only 22 notes, we get a FALSE result. Almost all of the functions in C.P.N.View return a TRUE/FALSE result, where appropriate.

si.locate(BAR, 20);

moves the iterator **si** to the start of the 20th bar of the score, if it exists.

Relative movement of the iterator is achieved by the **step** member function. This function may take a parameter, indicating the kind of object that it is required to move to. The following code fragment may be used to traverse all of the notes of the score contained in **filename**.

```
Score s(filename);
ScoreIterator si(s);
while (si.step(NOTE))
{
            doSomething(si);
}
```

Iterators in C.P.N.View are used to directly extract information about the objects in the score. C.P.N.View iterators carry out domain level processing. Much of this processing is involved with resolving contextual information. For example, where a score iterator points to a note, the member function **pitch12()** returns the chromatic note number (effectively the MIDI note number). C.P.N.View performs the following operations automatically. (1) key signature in use; (2) checks if any accidental alterations are present since the start of the bar in which the note in question resides, and (3) calculates relevant adjustments to the final pitch of the current note. Using all of this information the correct pitch12 value is returned.

The following fragment illustrates how the constructs discussed so far can be used to identify and print out the highest and lowest note of a piece and to calculate the pitch range of the piece.

```
Score s(filename);
ScoreIterator si(s);
si.step(NOTE);
ScoreIterator highest = si, lowest = si;
while (si.step(NOTE))
{
if ( si.getPitch12() < lowest.getPitch12()) lowest = si;
if (si.getPitch12() > highest.getPitch12()) highest = si;
}
std::cout << "\nHighest note is " << highest;
std::cout << "\nLowest note is " << lowest;
std::cout << "\nRange is " << highest.getPitch12() – lowest.getPitch12()
<< " semitones.\n";
```

Similarly, a program to locate the longest and shortest note in a piece, excluding grace notes,

```
Score s(filename);
ScoreIterator si(s);
si.step(NOTE);
ScoreIterator longest = si, shortest = si;
while (si.step(NOTE))
{
   if (!si.hasAttribute(GRACE_NOTE))
   {
            if ( si.getRDuration() < shortest.getRDuration()) shortest = si;
            if (si..getRDuration( ) > highest.getRDuration()) longest = si;
   }
}
std::cout << "\nLongest note is " << longest;
std::cout << "\nShortest note is " << shortest;
```

A question arises of what happens if we run these programs on a polyphonic score? The answer is that these will work and produce meaningful results.

To explain what happens it is necessary to consider two cases. The first one is where a score consist of a single stave, but has simultaneous notes. Some examples of this will be found in string music in which multiple stopping occurs. More complicated examples happen in scores where two lines occupy the same stave.

An iterator in C.P.N.View has an internal mode setting of MONO or POLY. In MONO mode the iterator traverses the uppermost notes on each stave only, and skips others. In POLY mode the iterator traverses all of the notes, moving vertically, from top to bottom, where possible, and moving to the next highest rightmost object otherwise.

Functions exist for setting and querying the scanning mode of an iterator.

si.putScanMode(MONO);

si.putScanMode(POLY);

getScanMode();

Figures 1 and 2 show iterators in MONO and POLY mode operating on a single stave piece.

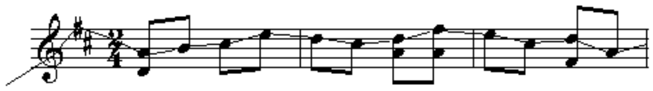


Figure.1 Single stave traversal in MONO mode

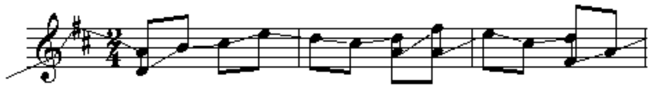


Figure 2 Single stave traversal in POLY mode

By default, the iterators created above will have MONO scan modes if the score contained in **filename** has one stave.

On the other hand, if the score is a multistave score, the scan mode will default to POLY, and the iterator will scan all of the objects in the score. To understand how the iterators scan across multiple staves, it is necessary to regard the score as being divided vertically into windows. Each window has an associated width, corresponding to a time span. The left and right borders of each window correspond to onsets or offsets of notes or rests. A window may not contain internal onsets or offsets.

Successive calls to the **step()** function moves the iterator vertically, wherever possible, from the uppermost object on the top stave in a window to objects on the lowermost stave in the same window. Where the score iterator points to the lowermost allowable object, a call made to **step()** moves the iterator to the uppermost object in the next adjacent window to the right. Figure 3 shows an such an iterator. Figure 4 demonstrates the concept of dividing the score into vertical windows.

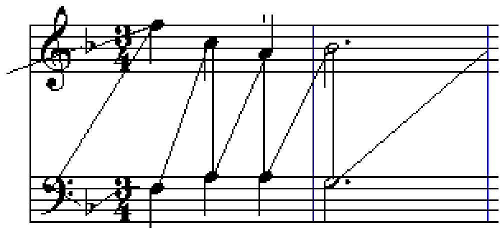


Figure 3 Multi-stave traversal in POLY mode

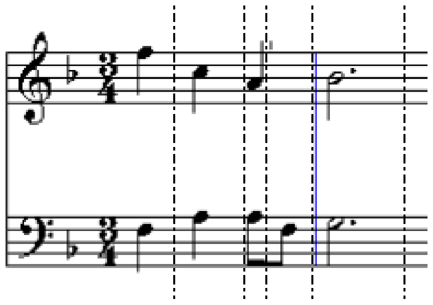


Figure 4 Multi-stave score divided into vertical windows

The program fragments above will work correctly with a polyphonic score and will search through all of the notes present. The occurrence of different clefs, changes in key and accidental changes in the score are all dealt automatically.

In cases where we want to scan individual staves of a polyphonic score, the constructor for the ScoreIterator takes an additional parameter corresponding to the stave number. The uppermost stave is numbered 0. An iterator created in this way will have a default scanning mode of MONO. To access objects on the last stave of a polyphonic score that has 10 staves, the following iterator could be used.

ScoreIterator si(s, 9);

In the previous examples, we have seen use of the member functions **locate**, **step**, **getPitch12**, **getRDuration**, **getScanMode** and **putScanMode**. A design strategy arises in creating such information retrieving functions. One could aim to design a minimal set of functions to retrieve the basic information content from the score. Such a minimal set will compromise convenience. The opposite approach of providing functions to retrieve every conceivable form will make things more difficult for the user, who will have a larger set to sift through and remember. The current set of 131 functions and operators might appear to lean towards the second approach. However many of these operators and functions cluster into families and thereby reduce the cognitive load in familiarization. Also many of these group into meaningful pairs. For example most member functions that start with 'get' have a counterpart that starts with 'put'. Additionally some of these are more frequently used than others. For example the **step** and **locate** functions are the main navigation mechanisms. There is very little additional to learn. The **getPitch12**, **getRDuration** deliver information similar to that available in a basic MIDI file. A short review of some of the main functions is given below.

The **getTag** member function give the type of object that is current.

If the current object is a note or rest, duration values may be retrieved

getHead() returns the head value,

getDots() returns the dot count,

getRDuration() retrieves the rational time value of the note or rest, including the resolution of groupette scoping.

Pitch information can be retrieved in many forms, including

getAlpha() returns the alphabetic note name,

getOctave() returns the octave number,

getAccid() returns details of any accidental placed directly on the note,

getPitch12() returns the MIDI pitch number of the note, with all necessary scoping resolved,

Many function return scoping information. These include the self obvious **getKeySig(), getTimeSig(), getClef()**, **getBarNo()**.

getBarDist() returns the distance of the current position from the start of the bar.

A facility exists for annotating any score object, and for querying these annotations.

4. IR EXAMPLES

The following illustrates the use of C.P.N.View in an IR context. It contains a complete implementation of the dynamic programming algorithm as documented in Sequence-Based Melodic Comparison: A Dynamic-Programming Approach[5]. The two encoded score fragments used in the algorithm have been encoded in two files that appear in lines 1 and 2. These are “Innsbruck ich muss dich lassen” and “Nun ruhen alle Waelder”.

This algorithm is based on the concept of a minimal edit cost of transforming a source melody into a target melody, using operations of insert, delete and replace. The cost matrix d, represents the minimal cost of transforming the notes of the source tune, represented in rows, into notes of the target tune represented across the columns. The recurrence equations for generating the matrix are

$$d_{00} = 0 \qquad \text{a1}$$

$$d_{i0} = d_{i-1,0} + w(a_i, \phi), i \geq 1 \qquad \text{a2}$$

$$d_{0j} = d_{0,j-1} + w(\phi, b_j), j \geq 1 \qquad \text{a3}$$

$$d_{ij} = \min\{d_{i-1,j} + w(a_i, \phi), d_{i-1,j-1} + w(a_i, b_j), d_{i,j-1} + w(\phi, b_j)\} \qquad \text{a4}$$

Where

$w(a_i, \phi)$, is the cost of inserting note a_i,

$w(\phi, b_j)$, is the cost of deleting b_j,

both of these have value 4 in this example.

$w(a_i, b_j)$, is the cost of substituting a_i with b_j,

This cost is calculated by taking the absolute value of the difference in MIDI note number, then adding half the absolute value of the difference in duration measured in 16th note units.

```
Score s1("C:\\Mdb\\Others\\inns.alm");                          // 1
Score s2("C:\\Mdb\\Others\\nur.alm");                           // 2
ScoreIterator si1(s1);                                          // 3
ScoreIterator si2(s2);                                          // 4
ScoreIterator siAr1[100] = {ScoreIterator()},
              siAr2[100]= {ScoreIterator()};                    // 5
int length1 = 0;                                                // 6
while ( si1.step(NOTE)) siAr1[++length1] = si1;                 // 7
int length2 = 0;                                                // 8
while ( si2.step(NOTE)) siAr2[++length2] = si2;                 // 9
double diffMatrix[100][100];                                    // 10
int i, j;                                                       // 11
diffMatrix[0][0] = 0.0;


for ( i = 1; i <= length1; i++)                                 //12
        diffMatrix[i][0] = diffMatrix[i-1][0] + 4.0;            //13
for ( j = 1; j <= length2; j++)                                 //14
        diffMatrix[0][j] = diffMatrix[0][j-1] + 4.0;            // 15

for ( i = 1; i <= length1; i++)                                 // 16
{
 for ( j = 1; j <= length2; j++)                                // 17
 {
  diffMatrix[i][j] =                                            // 18
   min3(
    diffMatrix[i-1][j] + 4.0,                                   // 19
    diffMatrix[i][j-1] + 4.0,                                   // 20
    diffMatrix[i-1][j-1] +                                      // 21
     fabs(siAr1[i].getPitch12() - siAr2[j].getPitch12()) +
     0.5*16*fabs(siAr1[i].getRDuration()- siAr2[j].getRDuration()));
 }
}
```

Lines 1 to 9 two arrays siAr1 and siAr2 are created, and contain iterators that point to each note of the score

Lines 12 to 15 correspond to equations a1, a2 and a3.

Lines 16 to 21 implement a4.

The function min3 is not documented here.

Table 1. The difference matrix diffMatrix.

		A	F	G	A	B	C	B	A
	0	4	8	12	16	20	24	28	32
F	4	6	6	10	14	18	22	26	30
F	8	8	6	8	12	16	20	24	28
G	12	10	10	6	10	14	18	22	26
A	16	14	14	10	9	13	17	19	22
C	20	18	18	14	13	14	15	19	22
B	24	22	22	18	17	16	18	15	19
A	28	26	26	22	21	20	21	19	15

Each cell in the matrix represents the difference at a particular point between the two fragments of "Innsbruck ich muss dich lassen" and "Nun ruhen alle Waelder". The total distance between the two melodies is represented by the value in the bottom right corner, 15. The combination of edit operators that yield this result may also be determined by tracing the best path from the bottom, rightmost cell to the top left corner.

In this example, it will appear that little is to be gained from using the score representation instead of MIDI. However it is worth emphasising that the with score representation this algorithm may be refined since all of the information content of the score is available. Some examples of using score information, instead of MIDI may be gleaned from the following possibilities. Stressed notes may be distinguished by their position in the bar, using information for **getBarDist** and **getTimeSig** functions. Using these, one could allow greater weights for inserting and deleting stressed notes. Pitch information can be dealt with at a finer level by distinguishing between enharmonic versions of the same note. Hence a C sharp may be treated differently than a D flat. A policy on handling rests will be necessary, if this algorithm is to have general applicability. The algorithm is not explicit on how grace notes are handled. Are they to be treated as part of the pitch contour? Perhaps a researcher may wish to treat them as providing extra emphasis on the following note.

A final example will illustrate a partial implementation of this same basic algorithm, as it was designed by Mongeau and Sankoff[6]. They used a more sophisticated model for the calculation of replacement costs than in the previous example. The previous example was encoded above as part of line 21, by calculating the absolute value of the two MIDI note number.

fabs(siAr1[i].getPitch12() - siAr2[j].getPitch12())

Mongeau and Sankoff used the pitch differences between pairs of notes to calculate difference weights, based on consonances of their intervals. This involves a simple table look-up for diatonic notes. However for comparing pairs of notes in cases where one or other note involved is chromatic, an alternate table is used, as is reflected in the following algorithm. The array **deg** holds the difference weights for the diatonic table and **ton** holds the corresponding weights for chromatic intervals.

The algorithm checks if the notes involved are diatonic, and applies the appropriate transformation. The function **pitchD** is a simplified version of the production algorithm, that is applicable to music is in a major key only.

The function **noteInKey** was developed specifically for this application. Its coding is given below.

```
int ScoreIterator::noteInKey()
{
 int key = currentKs;
 int actual12 = getPitch12();
 int actual7 = getPitch7();
 int nrOctaves7 = actual7/7;
 int octaveDisp12 = nrOctaves7*12;
 int scaleStep7 = actual7%7;
 int diatonicSteps[] = {0,2,4,5,7,9,11};
 int unalteredPitch12 = octaveDisp12 + diatonicSteps[scaleStep7] +
                                              getKeySigAdjust();
 if ( unalteredPitch12 == actual12 ) return TRUE;
 return FALSE;
}
```

```
double pitchD(ScoreIterator &si1, int major1, ScoreIterator & si2)
{
  double pitchDist = 0;
  if ( inKey(si1, major1 ))
  {
   double deg[] = {0.0, 0.9, 0.2, 0.5, 0.1, 0.35, 0.8};
   double ton[] = {0.6, 2.6, 2.3, 1.0, 1.0, 1.6, 1.8, 0.8,
                        1.3, 1.3, 2.2, 2.5};
   if ( si1.noteInKey() && si2.noteInKey() )
   {
    int diatonicSteps = fabs(si1.getPitch7() –
                            si2.getPitch7());
     diatonicSteps = diatonicSteps % 7;
     pitchDist = deg[diatonicSteps];
    }
    else
    {
     int chromaticSteps = fabs(si1.getPitch12() –
                            si2.getPitch12());
     chromaticSteps = chromaticSteps % 12;
     pitchDist = ton[chromaticSteps];
   }
  }
  return pitchDist;
}
```

5. CONCLUSION

Processing of music scores gives the prospect for accessing ever increasing corpora that have been created to high editorial standards. The Container/Iterator model gives an appropriate tool for algorithmic construction. Experience with C.P.N.View raises some interesting issues. An illustration of one such, that has been mentioned earlier in this paper is on devising an optimal set of operations to include in C.P.N.View. A minimal set, makes it easier for anyone to learn to use C.P.N.View. A more extensive set of operations, make it easier to write algorithms. A case in point is the **noteInKey** function above. This was not developed initially as part of C.P.N.View, but instead formed part of the implementation of the Mongeau and Sankoff algorithm. It was added to C.P.N.View, on the basis that it provided potential for reuse in other algorithms.
C.P.N.View provides a sufficiently abstract model of a score that it is potentially useable with a wide range of score representations, including some representations from notation packages. Currently C.P.N.View can accept input from score codings in ALMA, NIFF, *kern and EsAC. Some incomplete work has been done with SCORE and Enigma files.

6. REFERENCES

[1] Stepanov, A.A., and Lee, M. The Standard Template Library, Technical Report HPL-95911, Hewlett-Packard Laboratories, Palo Alto VA, February 1995.

[2] Plauger, P.J., Stepanov, A.A., Lee, M., and Musser, D.R. The C++ Stamdard Template Library, Prentice-Hall, NJ, 2001.

[3] Ó Maidín, D. Common Practice Notation View: a Score Representation for the Construction of Algorithms, Proceeding of the 1999 ICMC (Beijing,1999), ICMA, San Francisco, 248-251.

[4] Ó Maidín, D. “Common Practice Notation View User’ Manual” Technical Report UL-CSIS-98-02, University of Limerick, 1998.

[5] Smith, L., McNab, R., Witten, I. , Sequence-Based Melodic Comparison: A Dynamic-Programming Approach, in Melodic Similarity, Concepts, Procedures and Applications,Computing in Musicology 11. Hewlett W., Selfridge-Field, E. (eds). MIT Press 1998, 101-117.

[6] Mongeau, M.,Sankoff,D., Comparison of Musical Sequences, Computers and the Humanities 24,Kluwer Academic Publishers 1990, 161-175.

An Audio Front End for Query-by-Humming Systems

Goffredo Haus Emanuele Pollastri
L.I.M.-Laboratorio di Informatica Musicale, Dipartimento di Scienze dell'Informazione, Università Statale di Milano
via Comelico, 39; I-20135 Milan (Italy)
+39-02-58356222 +39-02-58356297
haus@dsi.unimi.it pollastri@dsi.unimi.it

ABSTRACT
In this paper, the problem of processing audio signals is addressed in the context of query-by-humming systems. Since singing is naturally used as input, we aim to develop a front end dedicated to the symbolic translation of voice into a sequence of pitch and duration pairs. This operation is crucial for the effectiveness of searching for music by melodic similarity. In order to identify and segment a tune, well-known signal processing techniques are applied to the singing voice. After detecting pitch, a novel post-processing stage is proposed to adjust the intonation of the user. A global refinement is based on a relative scale estimated out of the most frequent errors made by singers. Four rules are then employed to eliminate local errors. This front end has been tested with five subjects and four short tunes, detecting some 90% of right notes. Results have been compared to other approximation methods like rounding to the nearest absolute tone/interval and an example of adaptive moving tuning, achieving respectively 74%, 80% and 44% of right estimations. A special session of tests has been conducted to verify the capability of the system in detecting vibrato/legato notes. Finally, issues about the best representation for the translated symbols are briefly discussed.

1. INTRODUCTION
In the last few years, the amount of bandwidth for multimedia applications and the dimension of digital archives have been continuously growing, so that accessibility and retrieval of information are becoming the new emergency. In the case of digital music archive, querying by melodic content received a lot of attention. The preferred strategy has been the introduction of query-by-humming interfaces that enable even non-professional users to query by musical content. A number of different implementations has been presented since the first work by Ghias et al. [4] and a brief overview is introduced in the next section. In spite of this fact, the digital audio processing of an hummed tune has been tackled with naive algorithms or with software tools available on the market. This fact results in a poor performance of the translation from audio signals to symbols. Furthermore, previous query-by-humming systems can be hardly extended to handle sung queries (i.e. with lyrics) instead of hummed queries.

The quality of a query-by-humming system is strictly connected to the accuracy of the audio translation. It is well known that the amount of musical pieces retrieved through a melody grows when the length of the query decreases [8, 12, 13, 22]. Employing representations like the 3-level contour will further lengthen the list of matched pieces. In the same time, we can not expect users to search through very long queries (more than twenty notes long) or to sing perfectly, without errors and approximations. Interval representations show another source of errors, since a misplaced note propagates to the contiguous one. Thus, an accurate translation of the input is surely a basic requirement for every query-by-humming system.

In this paper, we propose an audio front end for the translation of acoustic events into note-like attributes and dedicated to the singing voice. We will focus on the post-processing of the voice in order to minimize the characteristic errors of a singer. In other words, the audio processing will be conducted in a user-oriented way, that is, trying to understand the intention of the singer. This work follows the one presented in [5] where some preliminary work and experiments have been briefly illustrated.

2. RELATED WORK
There are many techniques to extract pitch information from audio signals, primarily developed for speech and then extended to the music domain. The detection of pitch from monophonic sources is well understood and can be easily accomplished through the analysis of the sampled waveform, the estimation of the spectrum, the autocorrelation function or the cepstrum method.

Previous query-by-humming systems employed some basic pitch tracking algorithms with only little pre- and post- processing, if any. For example, Ghias et al. performed pitch extraction by finding the peak of the autocorrelation of the signal [4], McNab et al. employed the Gold-Rabiner algorithm [12], while Prechelt and Typke looked for prominent peaks in the signal spectrum [16]. Rolland et al. [19] applied an autocorrelation algorithm with heuristic rules for post-processing. Some works focused mainly on the matching and indexing stages of the query-by-humming, using software tools available on the market for the audio translation [3,7].

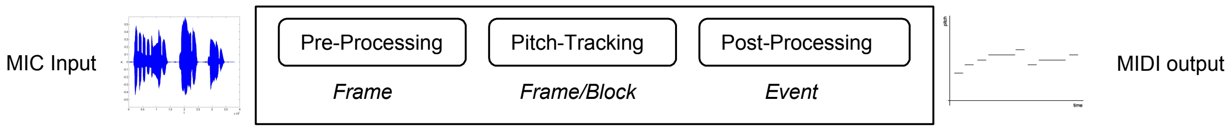


Figure 1. Architecture of the system developed.

Outside of the Music Information Retrieval community, the analysis of the singing voice constitutes an established research field, especially in the framework of voice analysis/re-synthesis. Typical examples are the voice morphing system by Loscos et al. [10], the structured audio approach to singing analysis score driven by Kim [6] and the synthesis of voice based on sinusoidal modeling by Macon et al. [11].

3. BACKGROUND

Despite its monophonic nature, singing has proved to be difficult to analyze [21]. The time-varying spectral characteristics of voice are similar during speech and singing. In both cases, we can divide the generated sounds in voiced and unvoiced[1]. In order to have an approximate idea of this property, we can think of the former kind of sounds as consonants[2] and the latter as vowels. Since voiced sounds are constituted by periodic waveform, they are easier to analyze, while unvoiced sounds have a state similar to noise. Luckily, during singing the voiced properties are predominant and contain what we call musical pitches. However, the information held by unvoiced regions are important as well, since they often contain the rhythmic aspect of the performance. Unlike speech, the singing voice shows a slowly-changing temporal modulation both in the pitch and in the amplitude (vibrato). In addition to these acoustic properties, singing voice analysis should deal with human performance that is typically affected by errors and unstable. Previous researches revealed that errors remain constant regardless of the note distance in time and in frequency [9]. We will follow these findings in the post-processing step of the proposed front end.

4. VOICE PROCESSING

An audio front end for a query-by-humming/singing system should contain all the elements needed to perform the transformation from audio to symbols, where audio is the singing voice and symbols are the most likely sequences of notes and durations. It should be able to adapt to the user automatically, i.e. without any user-defined parameter settings. Further, it should not require a particular way of singing, like inserting some little pause between notes or following some reference musical scale or metronome. In a query-by-singing application, the last requirements are important to avoid limiting the number of potential users, who are expected to be most non-professional users [5].

[1] A more rigorous definition is the following: "speech sounds can be voiced, fricative (or unvoiced) and plosive, according to their mode of excitation" [18]. In the present paper, plosive and fricative sounds will be grouped into the unvoiced category.

[2] with the exception of [m][n][l] which are voiced.

We suggest to elaborate the audio signal at three different levels of abstraction, each one with a particular set of operations and suitable approximations:

1- event

2- block

3- frame

At the event level, we estimate starting/ending points of musically meaningful signal, signal gain and, as a last step of computation, pitches and durations. At the block level, a background noise threshold is determined, voiced-unvoiced segments are isolated and pitches are approximated; eventually, effects of vibrato or bending are eliminated. At a frame level, we estimate spectrum, zero crossing rate, RMS power and octave errors. From the above observations, we derived an architecture (Figure 1) in which every uncertainty about the audio signal is resolved with subsequent approximations. The elaboration path is divided into three stages; details of each stage are presented in the following sections. The system developed is designed for offline voice processing and is not currently developed for real-time operations. Thus, audio is captured from a microphone, stored as wave file with sampling frequency of 44100 samples/sec and 16 bit of quantization, and then analyzed.

4.1 Pre-Processing

The first purpose of the audio front end is to estimate the background noise. We evaluate the RMS power of the first 60 msec. of the signal; a threshold for the Signal/Noise discrimination is set to a value of 15% above this level (S/N threshold). If this value is above –30dB, the user is asked to repeat the recording in a less noisy room. Otherwise, two iterative processes begin to analyze the waveform, one from the beginning and another from the end. Both the processes perform the same algorithm: the RMS power of the signal is calculated for frame 440 samples long (about 10 msec.) and compared with the S/N threshold. To avoid the presence of ghost onsets caused by impulsive noise, the value of the n-th frame is compared to the (n+4)-th. The value of 40 msec. is too long for such noise and it is not enough to skip a true note. The forward and backward analysis are then composed giving respectively a first estimate of the onset and offset points. The fragments of signal between each onset and offset represent the musically meaningful events.

Before localizing voiced and unvoiced regions, we calculate the derivative of the signal normalized to the maximum value, so that the difference in amplitude is emphasized. This way, it will be easier detecting the voiced consonants since their energy is most likely to be lower than the energy of vowels. A well-known technique for performing the voice/unvoiced discrimination is derived from speech recognition studies and relies on the estimation of the RMS power and the Zero Crossing Rate [1, 18]. Plosive sounds show high values of zero crossing rate because the spectral energy is almost distributed at higher frequencies. Mean experimental

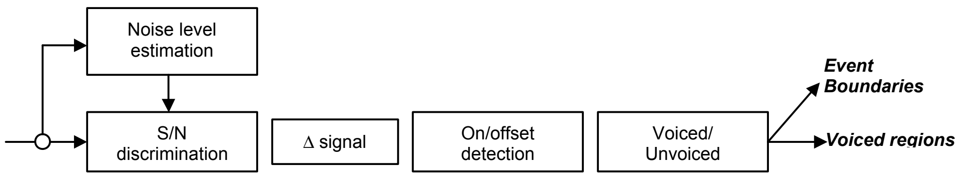


Figure 2. The pre-processing stage of the system developed. An audio signal given in input is segmented into musically meaningful events. Each event is characterized by its location in time (event boundaries) and by its voiced region.

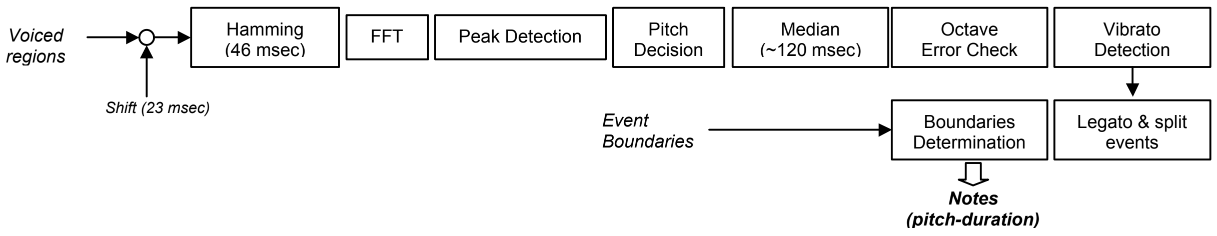


Figure 3. The proposed pitch-tracking stage; pitch detection is followed by a quantization step in which median approximation, vibrato suppression and legato detection are applied. The output is a sequence of pitches and durations.

values of average number of zero crossings are 49 for unvoiced sounds and 14 for voiced sounds in a 10 msec window. The task is not trivial for other speech utterance like weak fricatives. A better technique employs mean and standard deviation of the RMS power and zero-crossing rate of both background noise and signal as thresholds. Moreover, heuristic rules about the maximum duration admitted for each utterance are used. For example, events longer than 260 msec can not be unvoiced. These methods are applied to the derivative of the signal, detecting voiced consonants, unvoiced sounds and vowels. Thanks to this procedure, we can refine the on/offset estimation. In Figure 2 the process explained so far is illustrated.

4.2 Pitch-Tracking

As we said, the pitch of a sung note is captured by its voiced region and in particular by vowels. Thus, we will estimate pitch only on those fragments. Compared to unvoiced sounds, voiced sounds exhibit a relatively slowly-changing pitch. Thus, the frame size can be widen. For each voiced fragment identified in the segmentation step discussed above, the signal is divided into half-overlapping Hamming windows of 46 msec (2048 samples) (see Figure 3). A FFT algorithm is performed for each frame and the most prominent peaks of the estimated spectrum are passed to the next step. Here, it is taken the decision of pitch at the frame level. The algorithm is a simplified version of the one presented in [15]. The basic rule is quite simple: the candidate peak centered at a frequency in the range 87 Hz – 800 Hz that clearly shows at least two overtones is the fundamental frequency. Then, fundamental frequencies within an event are mediated along each three subsequent frames (median approximation) and are checked for octave errors. A group of four contiguous frames with similar fundamental frequencies constitutes a block. This further level of abstraction is needed to look for vibrato and legato (with glissando), which are slowly changing modulations in pitch and in amplitude. In the case of singing, vibrato is a regular modulation with rate of 4/7 Hz (i.e. with a 150/240 msec period or about 1/2 blocks) and depth between 4% and 15% [14, 20]. Legato is detected when adjacent blocks have pitches more than 0.8 semitones apart. This former case is resolved generating two different events; otherwise, the adjacent blocks are joint to form an event. For each event, pitch values are set to the average of the pitches of the constituting blocks. These information are gathered with the relative positions of consonants and the exact bounds of each note are estimated.

4.3 Post-Processing

The most critical stage is the post processing where the information captured by earlier stages are interpreted as pitch and duration. The intonation of the user is rarely absolute[3] and the transcription process has to take into account a relative musical scale. Pitches are measured in fraction of semitones to improve the importance of the relative distance between tones in the frame of the tempered musical scale. We use the definition of MIDI note; the number resulting from the following equation is rounded off to three decimal places:

$$Note_{MIDI} = \frac{1}{\log \sqrt[12]{2}} \log \frac{f}{f_0} \qquad \textbf{Eq. 1}$$

[3] Only about 1 in 10,000 people claim to have tone-Absolute Pitch [17]

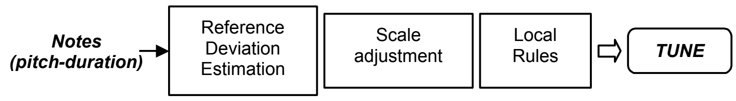


Figure 4. The post-processing stage of the system; the sequence of notes estimated in the previous stages is adjusted by means of a relative scale and four local rules. The definitive tune is given in output.

where f_0 is the frequency in Hertz associated to the MIDI note zero, that is:

$$f_0 = 440 / 2^{69/12} \cong 8.1758 Hz \qquad \textbf{Eq. 2}$$

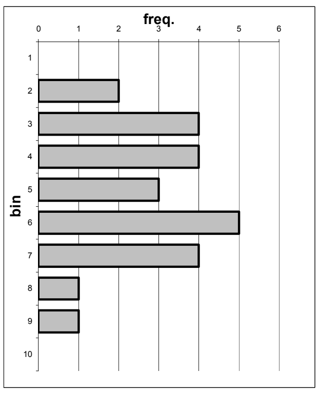
To our knowledge, only Mcnab et al. [12] introduced a procedure to adjust the scale during transcription. They used a constantly changing offset, initially estimated by the deviation of the sung tone from the nearest tone on the equal tempered scale. The resulting musical scale is continuously altering the reference tuning, in relation to the previous note. They relied on the assumption that singers tend to compress wide leaps and expand sequences of smaller intervals, suggesting that errors accumulate during singing. On the contrary, in this work we assume to deal with constant sized errors in accordance with the experiments conducted by Lindsay [9].

The tune estimated by the pitch-tracking is adjusted by means of three different steps: estimation of a reference deviation, scale adjustment and local refinement (Figure 4). The construction of a relative scale is based on the following idea: every singer has its own reference tone in mind and he/she sings each note relatively to the scale constructed on that tone. There are two important consequences: errors do not propagate during singing and are constant, apart some small increases with the size of the interval. These observations suggest to look for the reference tone of the singer through the estimation of his/her most frequent deviations from any given scale. In order to estimate the reference value for the construction of a relative scale, the semitone is divided into ten overlapping bins, each one being 0.2 semitone wide with an overlapping region of 0.1 semitone. We compute the histogram of the deviations from an absolute scale, which are the decimal digits of the estimated MIDI notes. The mean of the deviations that belong to the maximum bin is the constant average distance in semitones from the user's reference tone. Thus, the scale can be shifted by this estimated amount. An example is illustrated in Figure 5.

With the relative scale just introduced, we achieved results always better than rounding to the nearest MIDI note or implementing the algorithm by McNab et al. [12] (see next section for quantitative results). It is worth noting that the minimization of error has been obtained out of the whole performance of a singer. A further refinement is possible considering some local rules. When the reference deviation is between 0.15 and 0.85 semitones or there is more than one maximum bin in the histogram, the approximation introduced by the relative scale could be excessive. In particular, notes that have a deviation from 0.3 to 0.7 semitones on the relative scale are said to be critical. In this case, other four hypothetic melodies are considered; they reflect the following assumptions:

- a singer tends to correct its intonation from a note to the following one.
- some singers show a stable, even if slight, sharp or flat tuning with larger intervals (5 semitones and higher).
- rounded off absolute pitches and rounded intervals can further adjust an imperfect intonation

A very simple rule allows to remove single-note mistakes. The rounded melody on the relative scale is compared with the ones just calculated: a note *n* on the relative scale is replaced by the *nth* value given by three of the four representations above, when this value is the same in the three.



	Bin Range	Notes within a bin	Average Deviation
Bin 1	0.0-0.199		0
Bin 2	0.1-0.299	61.255; 58.286	0.27
Bin 3	0.2-0.399	61.255;58.286;58.328; 63.352	0.305
Bin 4	0.3-0.499	58.328;63.352;56.423;56.435	0.384
Bin 5	0.4-0.599	56.423;56.435; 61.537	0.465
Bin 6	0.5-0.699	61.537; 56.693; 56.623; 56.628; 56.644	**0.625**
Bin 7	0.6-0.799	56.693; 56.623; 56.628; 56.644	0.647
Bin 8	0.7-0.899	60.872	0.872
Bin 9	0.8-0.999	60.872	0.872
Bin 10	0.9-0.099		0

Figure 5. Example of calculation of the reference deviation (in bold style). The deviations (right) within the highest bin (left) are averaged.

5. REPRESENTATION ISSUES

The proposed front end can translate an acoustic input into a sequence of symbols represented by MIDI note numbers and durations in absolute time (milliseconds). This representation could not be best suited for querying by melodic content because it is not invariant to transpositions and different tempi. Musical intervals are the most likely representation for searching by similarity, as it is normally implemented by current query-by-humming systems. Since we expect to have a good approximation of the absolute pitches for each note, intervals can be naturally obtained as difference between a pitch value and the previous one.

A different matter concerns the rhythmic information, for which an acceptable representation is not known. Singers are likely to make great approximations on tempo, probably larger than the errors introduced by an imperfect estimation of note boundaries. Thus, the introduction of a stage for tempo quantization should be encouraged. For example, the measured lengths in msec of each note event could be smoothed by means of a logarithmic function. We suggest to use the following definition that is invariant to different tempi:

$$ratio\,(i) = round\left(10 \cdot \log_{10}\left(\frac{duration\ \ (i+1)}{duration\ \ (i)}\right)\right) \qquad \textbf{Eq. 3}$$

The alphabet is constituted by integers in the range [–10÷10]; for instance, the value 6 corresponds to the transition from a sixteenth note to a quarter note and –6 is the reverse transition; equal length transitions are represented by the symbol 0. Since it leads to a very detailed description of rhythm, this definition can be easily relaxed for handling approximate or contour-based representation of durations.

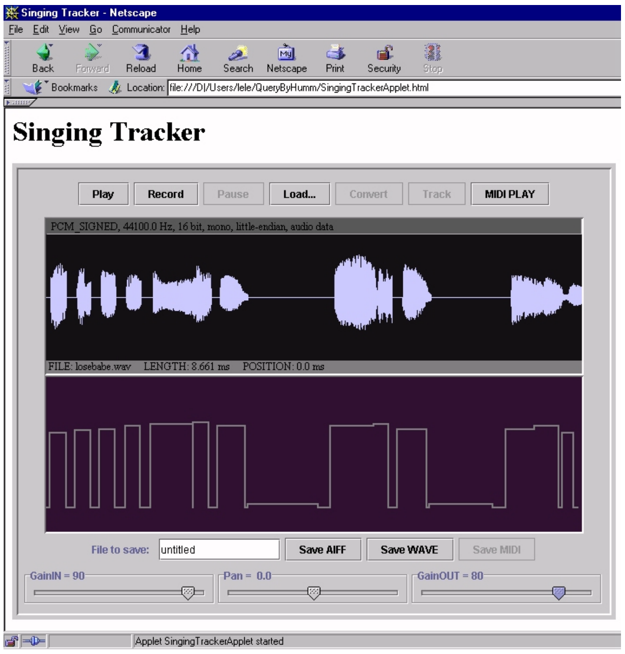

Singing Tracker - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: file:///D|/Users/lele/QueryByHumm/SingingTrackerApplet.html
Singing Tracker
Play Record Pause Load... Convert Track MIDI PLAY
PCM_SIGNED, 44100.0 Hz, 16 bit, mono, little-endian, audio data
FILE: losebabe.wav LENGTH: 8.661 ms POSITION: 0.0 ms
File to save: untitled Save AIFF Save WAVE Save MIDI
GainIN = 90
Pan = 0.0
GainOUT = 80
Applet SingingTrackerApplet started

Figure 6. Screen capture of the applet Java developed running in Netscape Navigator. It is illustrated an example of the translation of melody 1 (MIDI note on vertical axis; value 0 indicates pauses; value 5 represents silence).

6. EXPERIMENTAL RESULTS

The audio front-end illustrated in section 4 has been implemented in Matlab code and Java applet. The first prototype allowed us to adjust a proper set of parameters, while the second one has been employed with human subjects. The Java applet provides a graphical interface that allows users to record their voice through the sound card and to store it as a wave file. The recorded audio can be tracked and the translation can be stored as midifile or played. The system produces two warnings in case of too low or too high recording gain. Input and output gain can be easily adjusted by means of two sliders. The audio waveform and a graphical representation of the melody are given in output on the screen (see Figure 6).

Five subjects were asked to participate to the experiment. None of them was a musician or experienced singer, although they declared to feel themselves inclined to sing. Three subjects were male and two were female. The subjects sung in the tonality they preferred but without lyrics (i.e. singing ‘na-na’, ‘ta-ta’, ‘pa-pa’), simulating a query-by-humming session at home. The choice of removing lyrics was suggested to take out another possible source of errors that is difficult to quantify. Note segmentation is too much dependent on the way users remember the metric of a song. The experiment has been conducted in a non acoustically treated room, thus, in medium noisy condition. The audio card was a Creative Sound Blaster Live and the microphone was a cheap model by Technics.

Four simple melodies were chosen among the ones that the subjects proved to remember very well. The knowledge of the melodies doesn’t hold a particular importance; it assures the equivalence of the tunes to be evaluated. After the recording sessions, a musician was asked to transcribe the melodies sung by all the subjects without taking care of his memory of the melodies. The aim was to keep as much as possible of the original intention of the singers, and not their ability in remembering a music fragment. A different interpretation occurred only with a subject in three ending notes of a tune, but the same amount of notes has been sung and rhythm has been preserved; for this reason, that performance was included in the test. The transcriptions constitute the reference melodies for the evaluation of the front end. Each tune has been chosen for testing a particular block of the system. The main characteristics of each melody are described in the following:

- Melody number 1: three legato notes (with bending)
- Melody number 2: three sustained notes
- Melody number 3: very long, but well-known as well, sequence of notes (28 notes)
- Melody number 4 (“Happy Birthday”): well-known tune always sung in a different key

For each tune, the output of the front end is compared with the transcriptions by the musician. Each different pitch accounts for an error. In the case of round-interval tunes, the comparison has been made on the transcribed intervals. For the evaluation of the estimated durations, we consider only segmentation errors (i.e. a note split into two or more notes and two or more notes grouped into a note).

Table 1-2. Comparison of different methods for approximating an estimated tune. With the exception of the last row, values indicate the absolute number of wrong notes.

ALL SUBJECTS	Round MIDI	Moving Tuning	Round Intervals	Proposed without local rules	Proposed with local rules
Melody 1 (13 notes)	15	38	7	5	3
Melody 2 (8 notes)	4	15	4	3	3
Melody 3 (28 notes)	38	89	34	24	21
Melody 4 (12 notes)	19	30	8	8	4
ALL MELODIES					
Subject 1	22	26	12	8	10
Subject 2	9	42	8	8	5
Subject 3	14	35	12	8	6
Subject 4	17	32	10	6	3
Subject 5	14	37	11	10	7
Average Error (%)	**24.9%**	**56.4%**	**17.4%**	**13.1%**	**10.2%**

Tests have been carried out with different approximation methods for a direct comparison of the proposed one with the following: rounded midi note, McNab's moving tuning [12] and rounded intervals. The proposed method has been also tested without local rules (see Section 4.3), in order to assess their contribution. Results are illustrated in Table 1 and 2, ordered respectively by melody and by subject and without considering segmentation errors. In Table 3 the overall error rates (with segmentation errors) are summarized.

As previously noticed, the relative scale introduced in this work is always better than any other approximation method. The moving scale developed by McNab et al. [12] has the worst performance (56.4% of wrong approximations), confirming that errors do not accumulate. Rounding to the nearest tone on an absolute scale (round-MIDI) lead to an error in 26.6% of the sung notes,

Table 3. Summary of performances for the five methods employed. Error rates account for both pitch and segmentation errors

	Round MIDI	Moving Tuning	Round Intervals	Proposed without local rules	Proposed with local rules
Nr.of Wrong Notes (total number of notes=310)	81	177	63	45	36
Error Rate (%)	**26.1%**	**57.1%**	**20.3%**	**14.5%**	**11.6%**

showing a performance comparable to the proposed method only in the second melody. Here, the deviations from the MIDI scale are close to zero, thus indicating the simple round as a valid approximation. The round-interval tunes perform better as expected (17.4% of wrong approximated notes), since it confirms the previous work by Lindsay [9]. However, the segmentation errors have an unwanted side effect on intervals, since a single error propagates. Thus, the overall error rates increase more than the number of the segmentation errors, going from 17.4% of wrong pitches to 20.3% of wrong notes (pitch and segmentation).

The introduction of the four local rules bring some benefits, in fact the error rate is reduced from 13.1% to 10.2%. In absolute terms, these heuristic rules permit us to make the right approximation for ten notes more and introduce wrong approximations for only a note.

The recognition of note events has been very successful: only 5 notes were split into two events, thus identifying a total number of 310 notes instead of 305. Such a negligible error rate can be easily fixed by a somewhat fuzzy algorithm for melody comparison and retrieval, for example in a hypothetic next stage of the audio front end. As already said, in the case of round-interval the segmentation errors lead to heavier costs.

An example of translation is reported in Table 4. It is shown the

Table 4. Approximation of melody 4 (first twelve notes of "Happy Birthday"); actual notes come from the transcription by a musician. Sung melody represents the sequence of pitches given in output by the second stage of the front end.

Actual Melody	56	56	58	56	61	60	56	56	58	56	63	61
Sung Melody	56.693	56.623	58.328	56.628	61.255	60.872	56.423	56.435	58.286	56.644	63.352	61.537
Round-MIDI Melody	**57**	**57**	58	**57**	61	**61**	56	56	58	**57**	63	**62**
Dev. Round-MIDI Melody	-0.307	-0.377	0.328	-0.372	0.255	-0.128	0.423	0.435	0.286	-0.356	0.352	-0.463
Adjusted Melody	56.068	55.998	57.703	56.003	60.63	60.247	55.798	55.81	57.661	56.019	62.727	60.912
Round Adjusted Mel.	56	56	58	56	61	60	56	56	58	56	63	61
Dev. Adjusted Melody	0.068	-0.002	-0.297	0.003	-0.37	0.247	-0.202	-0.19	-0.339	0.019	-0.273	-0.088
Rounded Intervals		0	2	-2	5	**0**	-4	0	2	-2	7	-2
Moving Tuning	**57**	56	58	**57**	**62**	**61**	56	56	58	**57**	63	62

Variance Dev. Round MIDI	*0.134*
Reference Dev. Melody	*0.625*

Variance Dev. Adjusted Mel.	*0.036*

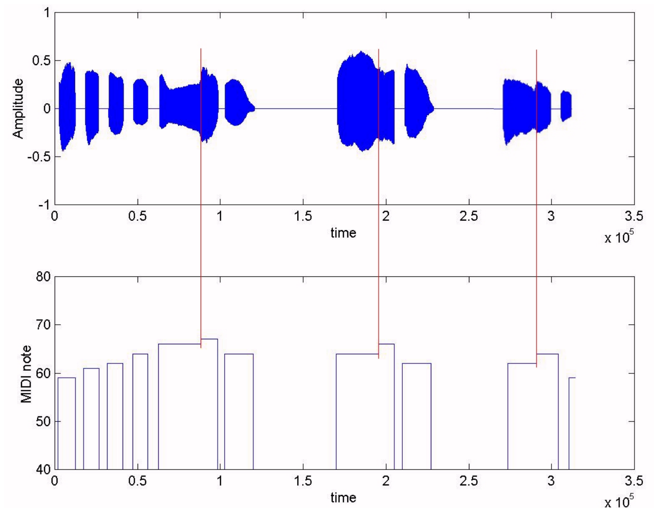


Figure 7. Example of legato notes detection (melody 1).

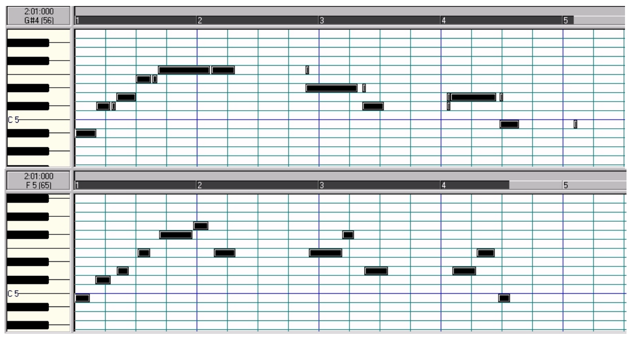


Figure 8. Transcription of melody 1 by a software tool available on the market and by the system developed here. Actual notes coincide with the sequence on the bottom.

reference deviation on which the relative scale is built; errors are indicated in bold type. Without employing any local rules, the melody is perfectly approximated on the relative scale, while the round interval and moving-tuning approximations account respectively for an error and six errors.

A hard problem for pitch-tracking algorithms are notes sung legato, for which there is neither a noticeable change in energy nor an abrupt modification in pitch. In Figure 7, the sampled waveform of the melody nr.1 is depicted with its translation. Three vertical lines highlight the estimated legato tones. The approximation introduced by the front end is able to capture the performance, splitting the legato notes in a natural way. The same file has been translated by means of Digital Ear® by Epinoisis Software [2]. Since this software tool allows smart recognition of onsets and recovery of out-of-tune notes, different settings have been employed. In Figure 8, one of the resulting MIDI files (top figure) is compared to the translation obtained with our system (bottom figure). Although it is not made clear in the figure, the actual notes coincide with the latter tune; a number of errors both in the segmentation and pitch-tracking can be noted in the former translation.

7. CONCLUSION AND FURTHER WORK

The need of dedicated singing voice processing tools strongly arises in the context of query-by-humming systems. The translation of the acoustic input into a symbolic query is crucial for the effectiveness of every music information retrieval system.

In the present work, well-known signal processing techniques have been combined with a novel approach. Our goal is the realization of an audio front end for identifying, segmenting and labeling a sung tune. The labeling stage constitutes the novelty; it enables to adjust a human performance out of a set of hypothesis on the most frequent errors made by singers. The adjustment follows two steps: global tuning and local rules. Both methods have been tested with twenty human performances (four tunes, five singers). We achieved the detection of some 90% of right notes with both steps. Previously employed methods like rounding to the nearest absolute tone or interval, and the moving tuning by McNab et al. [12], were outperformed, since they respectively accounted for about 74%, 80% and 44% of right notes. A special session of tests has been carried out to verify the ability of the pitch tracking stage in detecting vibrato and legato effects. An example has been reported in comparison with a software tool available on the market. The proposed front end roughly identified all the notes sung legato in our dataset. Quantitative results could not be presented, since it is impossible to classify as right/wrong the splitting point between two legato tones.

Much work needs to be done in different directions. First, we are developing a new pre-processing stage for the detection of noise. The aim is twofold: improving the estimation of the background noise level and filtering the noisy sources from the singing voice. This pre-process should be very robust since we are looking to applications like query-by-singing by cellular phones or other mobile devices.

In the post-processing stage, we relied on assumptions derived from the cited work of Lindsay [9]. Although these assumptions have been confirmed, a more rigorous model should be formalized. Moreover, we employ four local rules that have been introduced from experimental results but we don't know how these rules can be arranged in a more general model.

Query-by-singing is a straightforward extension of querying through hummed tones. Preliminary tests show that the task is not trivial and should need further experiments for the detection of note boundaries. As we said, language articulation could cause a wrong estimation of both the number of events and the rhythmic aspects of a performance.

Finally, current implementation suffers from the known performance deficiencies of Java. The computation time is about the same of the play time (i.e. length of the audio file) on a Pentium III, 450MHz running Windows NT 4.0. Thus, a complete re-engineering of the package is necessary and we can not exclude the possibility of migrating to other software platforms.

8. ACKNOWLEDGMENTS

The authors wish to thank Fabrizio Trotta who performed most of the preparatory and programming work for this paper. Special thanks to Giulio Agostini, Andrea D'Onofrio and Alessandro Meroni for their precious help and good advice.

This project has been partially supported by the Italian National Research Council in the frame of the Finalized Project "Cultural Heritage" (Subproject 3, Topic 3.2, Subtopic 3.2.2, Target 3.2.1).

9. REFERENCES

[1] Deller, J. R., Porakis, J. G., Hansen, J. H. L. Discrete-Time Processing of Speech Signals. Macmillan Publishing Company, New York, 1993

[2] Digital Ear®, Epinoisis Software, www.digital-ear.com

[3] Francu, C. and Nevill-Manning, C.G. Distance metrics and indexing strategies for a digital library of popular music. Proc. IEEE International Conf. on Multimedia and Expo, 2000.

[4] Ghias, A., Logan, D., Chamberlin, D., Smith, S.C. Query by humming – musical information retrieval in an audio database. in Proc. of ACM Multimedia'95, San Francisco, Ca., Nov. 1995.

[5] Haus, G. and Pollastri, E. A multimodal framework for music inputs. In Proc. of ACM Multimedia 2000, Los Angeles, CA, Nov. 2000.

[6] Kim, Y. Structured encoding of the singing voice using prior knowledge of the musical score. In Proc. of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, New Paltz, New York, Oct. 1999.

[7] Kosugi, N. et al. A practical query-by-humming system for a large music database. ACM Multimedia 2000, Los Angeles, CA, Nov. 2000.

[8] Lemstrom, K. Laine, P., Perttu, S. Using relative slope in music information retrieval. In Proc. of Int. Computer Music Conference (ICMC'99), pp. 317-320, Beijing, China, Oct. 1999

[9] Lindsay, A. Using contour as a mid-level representation of melody. M.I.T. Media Lab, M.S. Thesis, 1997.

[10] Loscos, A. Cano, P. Bonada, J. de Boer, M. Serra, X. Voice Morphing System for Impersonating in Karaoke Applications. In Proc. of Int. Computer Music Conf. 2000, Berlin, Germany, 2000.

[11] Macon, M., Link, J., Oliverio, L., Clements, J., George, E. A singing voice synthesis system based on sinusoidal modeling. In Proc. ICASSP 97, Munich, Germany, Apr. 1997.

[12] McNab, R.J., Smith, L.A., Witten, C.L., Henderson, C.L., Cunningham, S.J. Towards the digital music libraries: tune retrieval from acoustic input. in Proc. of Digital Libraries Conference, 1996.

[13] Melucci, M. and Orio, N. Musical information retrieval using melodic surface. in Proc. of ACM SIGIR'99, Berkeley, August 1999.

[14] Meron, Y. and Hirose, K. Synthesis of vibrato singing. In Proc. of ICASSP 2000, Istanbul, Turkey, June 2000.

[15] Pollastri, E. Melody retrieval based on approximate string-matching and pitch-tracking methods. In Proc. of XIIth Colloquium on Musical Informatics, AIMI/University of Udine, Gorizia, Oct. 1998.

[16] Prechelt, L. and Typke, R. An interface for melody input. ACM Trans. On Computer Human Interaction, Vol.8 (forthcoming issue), 2001.

[17] Profita, J. and Bidder, T.G. Perfect pitch. American Journal of Medical Genetics, 29, 763-771, 1988.

[18] Rabiner, L.R. and Schafer, R.W. Digital signal processing of speech signals. Prentice-Hall, 1978.

[19] Rolland, P., Raskinis, G., Ganascia, J. Musical content-based retrieval: an overview of the Melodiscov approach and system. In Proc. of ACM Multimedia'99, Orlando, Fl., Nov. 1999.

[20] Rossignol, S., Depalle, P., Soumagne, J., Rodet, X., Collette, J.L. Vibrato: detection, estimation, extraction, modification. In Proc. of DAFX99, Trondheim, Norway, Dec. 1999.

[21] Sundberg, J. The science of the singing voice. Northern Illinois University Press, Dekalb, IL, 1987.

[22] Uitdenbogerd, A. and Zobel, J. Melodic matching techniques for large music databases. In Proc. of ACM Multimedia'99, Orlando, Fl., Nov. 1999.

MUSART: Music Retrieval Via Aural Queries

William P. Birmingham, Roger B. Dannenberg, Gregory H. Wakefield, Mark Bartsch,
David Bykowski, Dominic Mazzoni, Colin Meek, Maureen Mellody, William Rand

University of Michigan
128 ATL Building
1101 Beal Avenue
Ann Arbor, MI 48109-2110
(734) 936-1590
wpb@eecs.umich.edu

Carnegie Mellon University
School of Computer Science
Pittsburgh, PA 15213
(412) 268-3827
rbd@cs.cmu.edu

ABSTRACT
MUSART is a research project developing and studying new techniques for music information retrieval. The MUSART architecture uses a variety of representations to support multiple search modes. Progress is reported on the use of Markov modeling, melodic contour, and phonetic streams for music retrieval. To enable large-scale databases and more advanced searches, musical abstraction is studied. The MME subsystem performs theme extraction, and two other analysis systems are described that discover structure in audio representations of music. Theme extraction and structure analysis promise to improve search quality and support better browsing and “audio thumbnailing.” Integration of these components within a single architecture will enable scientific comparison of different techniques and, ultimately, their use in combination for improved performance and functionality.

1. INTRODUCTION
We are integrating components for music search and retrieval into a comprehensive architecture called MUSART, an acronym for *MUSic Analysis and Retrieval Technology*. Like several other music-retrieval systems, MUSART takes as input an aural query, which is typically a theme, hook, or riff, of the piece for which the user is searching. Unlike other systems, however, MUSART automatically builds a thematic index of the pieces in its database. Since users generally remember the theme of a piece of music, and the theme can occur anywhere in a piece, indexing by theme can greatly improve both the precision and recall of the retrieval system.

Moreover, MUSART uses a variety of representations to support multiple search modes. These representations run from a Markov model, to phonetic streams, to strings. This allows us, for example, to easily compute approximate matches and to search based on stylistic similarity or the lyrics in a popular song. Our representation can capture harmony and rhythm, should the user decide to query based on harmonic progression or rhythmic pattern, or both, in addition to or in place of melody.

The current version of the system contains hundreds of pieces from different Western genres (all are tonal pieces). From these pieces, we have automatically induced about 2000 themes. MUSART is able to effectively retrieve pieces from either the theme or full piece databases. The system is relatively robust against queries that contain some classes of errors (e.g., rhythmic changes). We measure performance by rank (the *target* piece is in the top ten pieces retrieved), yielding measures from 100%[1] for queries without errors and degrading from there based on the type of error in the query.

In this paper, we describe the MUSART system architecture[2]. The discussion concentrates mostly on searching themes, as we believe this will be the primary method most users will employ. We will mention, however, extensions to full pieces and other music-analysis strategies. We begin by placing our work in the context of current systems.

2. RELATED RESEARCH
There are varieties of approaches described in the database and information-retrieval (IR) literature on retrieval of music. Some of these approaches, such as Variations [1], are primarily based on retrieving either musical scores or sound recordings using traditional categorization schemes, where the musical items are treated in much the same way as text-based media.

A number of other systems [2-8] have focused on sound and MIDI [9] input. These systems generally take as input a melody that is “hummed” or played on some type of musical input device, such as a MIDI keyboard. The hummed melodies are converted to text strings, usually with a representation of intervallic distance or simply relative pitch contour [10]. For the melody “Happy Birthday,” a user may hum the pitches “G G A G C B,” which may then be more simply categorized as ascending (U), descending (D), or the same (S) to yield the pitch contour “S U D U D.” A commonly used variation to the SUD approach is to divide jumps into large (U or D) and small (u or d), where large is, for example, a jump greater than a minor 3rd. This SUuDd alphabet provides more information than a string composed from the SUD alphabet, but does not substantially change the retrieval process.

[1] That is, in 100% of the cases the target was in the top-ten rank.

[2] See musen.engin.umich.edu for more information on the MUSART project and related projects.

While hummed input is a natural approach, there are some problems with using it as the sole input. Pitch contour and related representations (such using actual or modified pitch normalized to some key) are not necessarily unique: the hummed input is sometimes fraught with a number of distortions inherent in the way that humans remember and produce (sing) melodies [6]. Such distortions include raising or lowering the pitch of various notes, “going flat” over time, losing the beat, or humming “off key.” To account for these distortions, researchers treat the string as imprecise. In other words, regions of uncertainty are added. The retrieval process on imprecise strings employs a string-matching [11], N-gram [12], or other algorithm where the input string is matched against abstractions stored in the database.

While this approach has clearly led to some impressive results, we argue that the “string approach” is fundamentally limited for the following reasons:

- The assumption that a melodic fragment exists and forms a suitable search key is not always true for some types of music such as rap, electronic dance music, and even some modern orchestral music. Users may want to search for non-melodic attributes.

- Given a large, but not unrealistic, corpus of music, pitch contour or intervallic representations will not uniquely identify pieces. In other words, there will be a great many pieces with similar string representations; this problem is exacerbated if the strings are modeled as imprecise.

- In some genres, harmony, homophony or polyphony may be predominant musical features. Representing any of these as simple strings is fraught with significant problems. For example, would a two-voice piece be represented as two “concurrent” strings?

We are not arguing against melodic search *per se*; we only want to recognize that current practice must be augmented with new techniques to achieve better performance and richer functionality. For example, current research suggests using genre to narrow searches, but in many cases users may want to search for orchestration, themes, or rhythms that span many genres. Using genre to narrow searches is similar to MARC-record searching in that pre-established categories are selected using traditional database and query techniques. We look to implement more general and more powerful searching techniques.

Several researchers have described systems based on various non-string methods to represent and classify music [8, 13, 14]. These projects have not yet demonstrated how to integrate general query mechanisms and methods for returning results. Nor do these systems integrate abstraction with a search mechanism.

Retrieval by melody depends on the “hook” of a piece of music. The hook is usually what a person uses as a query when humming. The problem is that although the hook may be contained in the opening melodic line of a song, often it is not. For example, it is common in popular songs for a familiar chorus (such as “Take me out to the ballgame…” by Jack Norworth & Albert Von Tilzer) to follow an unfamiliar verse (such as “Nelly Kelly loved baseball games…,” the actual opening line from the 1927 version of “Take Me Out to the Ballgame”). In classical music, the main theme (or melody) may not be stated for several measures following the start of the piece, or it may be disguised in a variation, and the variation may be better known than the main theme.

Thus, there is a significant problem in determining what part of a melody, or even which melody, should be indexed. Some abstraction of a piece is clearly needed, as many pieces of music are too long to be reasonably searched. Faced with these problems, music librarians have developed thematic indexes which highlight significant themese in staff notation, thus preserving major musical features (harmony, rhythm, etc.) [13].

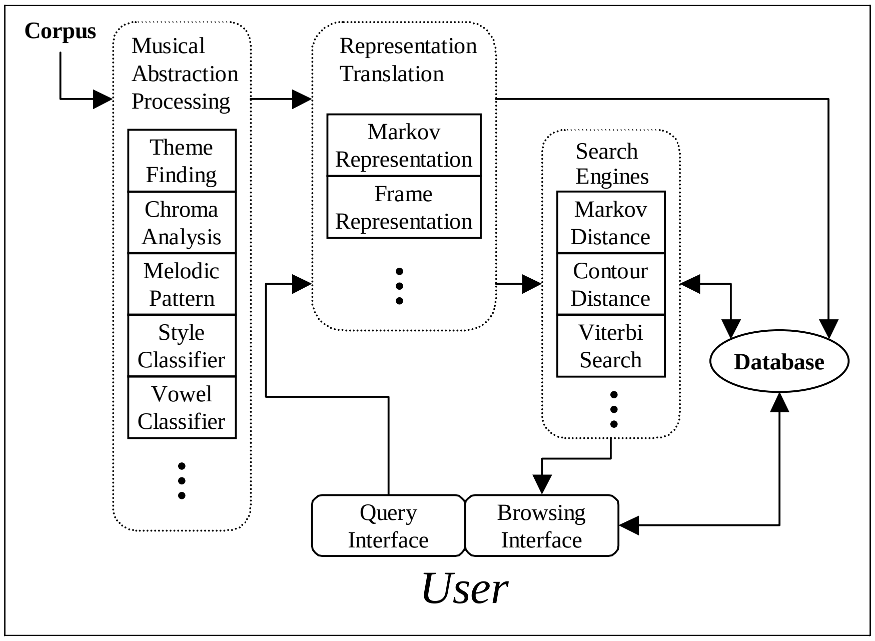


Figure 1. MUSART Architecture.

We know of no other researchers who are addressing the problem of abstraction using automated mechanisms.

3. MUSART ARCHITECTURE

This section illustrates the conceptual organization of our system (see Figure 1). The corpus is preprocessed using a collection of tools to build abstract representations of the music (i.e., our data). The extracted information is then translated to multiple representations. Queries are also translated, and a search engine is selected to search the database. The important idea of this framework is that various modules and representations can work together in parallel or in sequence to achieve more refined searches than any single search technique can offer. We can also compare different techniques on the same set of data.

With this architecture, the user interface can offer a rich set of options to the user. Because music is abstracted in various ways, users can explore musical spaces along many dimensions. In addition, abstract representations lend themselves to music generation and synthesis. We have just begun to explore this possibility, but this promises to be a powerful tool for users to refine queries and to explore different musical dimensions.

So far, we have implemented several new techniques for searching, and we have investigated several techniques for musical abstraction. These are described in the following sections. We are in the process of integrating these subsystems to form a more integrated whole.

4. MARKOV REPRESENTATION AND THE CONCURRENCY

One way we model music is as a stochastic process, where we cast a musical performance as a Markov model. Recently, we have experimented with Markov models where we use simple state features of pitch (or concurrent set of pitches) and duration [14]. The transition table is the likelihood of going from one state to another, where the priors are determined by counting transitions occurring in some set of music (e.g., the piano concerti of Mozart).

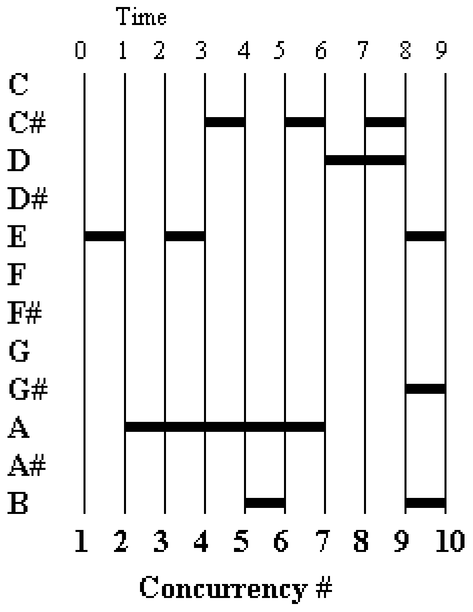


Figure 2: Piano roll for *Wilson's Wilde*, mm. 1 – 4.

State #	Notes in State
1	E
2	A
3	E, A
4	C#, A
5	A, B
6	C#, A
7	D
8	C#, D
9	E, G#, B
10	REST

Table 1: Concurrencies for Wilson's Wilde, mm. 1-4.

We define the states of the Markov model as *concurrencies*. A concurrency consists of all the notes that are sounding at the same time. These notes need not all have the same duration, onset, or offset. Hence, a different concurrency exists at each point a new note begins or terminates in a piece. For the purposes of simplification, the notes in a concurrency are modeled by pitch class, thereby ignoring octave information. By representing concurrencies in this manner, we are able to store and use them as 12-tuple bit vectors. It is easy to extend the concurrency to include octave information, simply by increasing the size of the bit vector. For example, a two-octave range is represented by a 24-tuple and corresponding bit vector.

The concurrency can represent up to twelve simultaneous pitches (every pitch class); elements of the harmony or polyphony are naturally represented. Moreover, this represent is well suited for automated harmonic analysis [15].

We use Figure 2 and Table 1 as examples of a set of concurrencies and the corresponding state and transition tables for the Markov model representing the piece. Figure 2 shows the piano-roll notation for *Wilson's Wilde*, an anonymous 16th-century work. Table 1 shows the corresponding concurrencies (given in the table as Markov states) based solely on this excerpt.

Although relatively simple, the Markov representation has several interesting properties that allow us to use it as one of the primary representations in our system. First, our results from inducing Markov models from the pieces in our corpus indicate that composers occupy (mostly) unique regions in the state space implied by the Markov model. Secondly, we found that the transition matrices are relatively sparse (more so for some works than others). Thirdly, we can impose an order on the states implied by the Markov model.

The ordering is formed as follows: we quantize note duration to some minimum (e.g., 16th-note) and use a pitch-class vector where, for example, <100000000000> represents the pitch class C as the single pitch sounding, <110000000000> represents the pitch classes C and C#/D-flat sounding concurrently, and so forth. A state is represented by a duple (duration, pitch vector). We can simply order the state space as follows: (16th-note, <100000000000>), (8th-note, <100000000000>), … (whole note,

<111111111111>), where durations are ordering by increasing powers of 2 of a 16th-note, and the maximum duration is a whole note.[3]

Because of these three properties, we can assess similarity among pieces of music, or even composers using a variety of techniques. Initial results indicate a strong correspondence between similarity as computed by the MUSART system and educated musical opinion. Consider that once the query is converted to a Markov chain, it can be easily correlated with pieces in the database (See Section 6.) While at worst case this is a linear-time operation, the correlation operation is fast and, with clever indexing, we can significantly reduce this time. Finally, the induction of the Markov model and correlation computation can be done off line.

5. Thematic Abstraction

We are interested in extracting the major themes from a musical piece: recognizing patterns and motives in the music that a human listener would most likely retain. Extracting themes is an important problem to solve. In addition to aiding music librarians and archivists, exploiting musical themes is key to developing efficient music retrieval systems. The reasons for this are twofold. First, it appears that themes are a highly attractive way to query a music-retrieval system. Second, because themes are much smaller and less redundant than the full piece, by searching a database of themes rather than full pieces, we simultaneously get faster retrieval (by searching a smaller space) and get increased relevancy. Relevancy is increased as only crucial elements, variously named motives, themes, melodies or hooks are searched, thereby reducing the chance that less important, but frequently occurring, elements are assigned undue relevancy.

Our theme abstraction subsystem, MME [16], exploits redundancy that is found in music. Thus, by breaking up a piece into note sequences and seeing how often these sequences repeat, we identify the themes. Breaking up a piece involves examining all note sequence lengths of one to some constant. Moreover, because of the problems listed earlier, we must examine the entire piece and all voices. This leads to very large numbers of sequences, thus we must use a very efficient algorithm to compare these sequences.

Once repeating sequences have been identified, we must further characterize them with respect to various perceptually important features in order to evaluate their thematic value. It has been noted, for instance, that the frequency of a pattern is a stronger indication of thematic importance than pattern register. We implement hill-climbing techniques to learn weights across features, i.e., MME learns relative to a training set the relative importance of the features it uses. The resulting evaluation function is then used to rate the sequence patterns we uncover in a piece. A greedy algorithm is used to identify a subset of the piece, consisting of some pre-determined number of note events, containing top-ranked patterns.

Our formal evaluation of MME on over 30 training trials, with 30-piece training and test sets randomly selected across 60 pieces for each trial, MME returns Barlow's *A Dictionary of Musical Themes* [17] "1st theme" in 98.7% of cases (see Figure 4). Figure 3 shows sample output from MME, two slightly different versions of the passage Barlow identifies as the "1st theme".

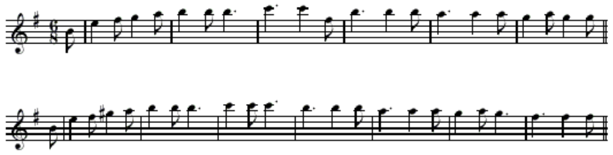


Figure 3: Sample MME output, Smetana's *Moldau*.

Because of the large number of patterns that MME may find in a complex piece of music, it must be computationally efficient. The system's overall complexity is $\Theta(m^3n^2)$ time, where m is the maximum pattern length under consideration, and n is the number of note events in the input piece. In practice, however, we observe sub-linear performance, and reasonable running times on even the largest input pieces.

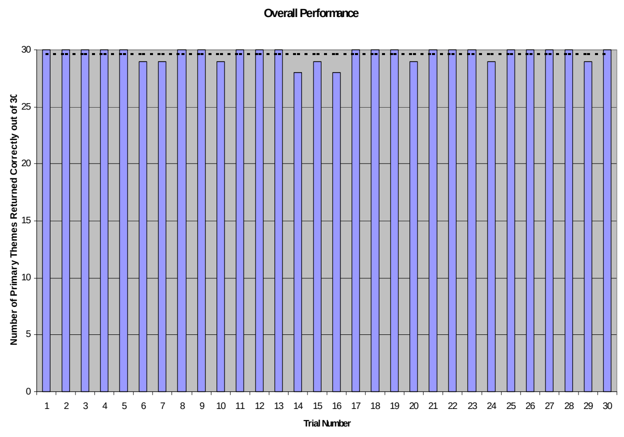


Figure 4: MME test results.

6. Retrieval Methods

The pieces in the MusArts' database are converted by MME into a set of themes, where up to a certain number of themes are identified for each piece. These themes are then converted to the Markov models described in Section 4. Thus, the "Markov Distance" retrieval engine takes a query, converts it to a Markov model, and then computes a variety of correlations measures between the query model and all "theme" models [18].

Currently, we use two correlation methods to examine how the query relates to a theme. Both are based on a standard correlation coefficient. One technique, however, uses the first-order Markov model, while the other simply uses the frequency count of each state observed, which is a zero-order Markov model. Once a comparison has been made to each database entry the results are sorted and presented as a score out of 1000 to the user.

The results so far are promising. We have taken some of the themes extracted automatically and manipulated them to simulate typical errors we expect to see in queries. The three classes of errors that we are investigating are duration change, note drop, and pitch change. In our experiments, we set an error rate for each type of error that indicates what percentage of the MIDI events we will manipulate when comparing the query to the database entries.

[3] We can choose any range of durations, and any method to order the durations.

We define a successful trial as one in which the piece that was being sought is presented as one of the top ten themes returned (a *rank* measure). The system is very robust to duration-change errors, having as high as a 95% success rate even when the error rate is 100%. When it comes to note-drop errors, the system performs at a 75% success rate with error-rates approaching 50%. However the system is not robust to pitch-change errors. It appears that a pitch-change error rate of 10% the system does not return the sought piece in the top rank. We are investigating several solutions to this problem.

We have recently implemented a Viterbi algorithm for retrieval. With this algorithm, we can find which Markov model is most likely to cover the input query. Rather than calculate correlation measures, this approach calculates true posterior probabilities (given the input query as evidence). To account for errors, we insert "error states" in the Markov model for each theme. The distributions for these error states are currently based on our intuition; however, we are conducting an extensive set of experiments to get better error models.

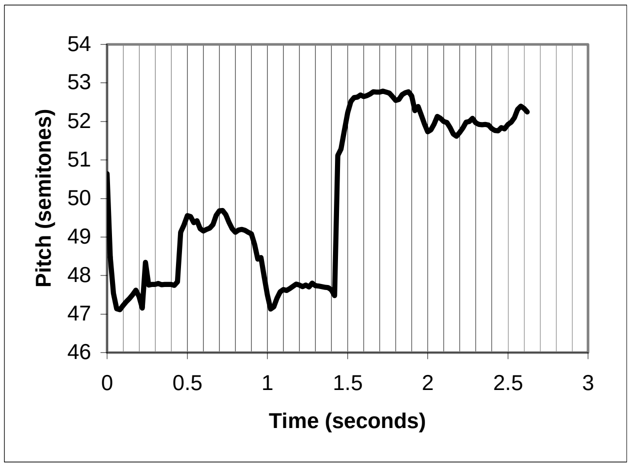
Our initial results with the Viterbi approach are encouraging. Using the same experimental setup as we used for the correlation experiments, we have recorded better results for all error classes. In particular, the Viterbi approach appears to be more robust to pitch-change errors.

The description of both the correlation and Viterbi approaches given in this paper has relied on searching a database of monophonic themes (which do include rhythmic features). Given that both approaches are based on concurrencies, it is very simple to apply both approaches to homophonic or polyphonic music. In fact, we are experimenting with searching for harmonic progressions using the Viterbi approach.

7. FRAME REPRESENTATION AND MELODIC CONTOUR

We are exploring another representation and search strategy to address some of the problems of conventional "event-based" searching, where events are typically musical notes and searching is performed on the basis of note sequences. Event-based searches suffer from at least two problems. First, it is difficult for music transcription systems to segment audio correctly into discrete notes. This is a problematic even when skilled musicians sing queries. Secondly, efficient string-matching algorithms, when given enough leeway to ignore music transcription and performance errors, can often time-align two perceptually dissimilar melodic strings, leading to false positives.

An alternative is to ignore the concept of note and perform a direct comparison of musical contours, representing melody as a function of pitch versus time. This function is discretized by segmenting the melody into time frames. Thus, we call this a "frame-based" approach. Figure 5 illustrates a pitch contour from an audio query and the resulting query string. In this approach, there is no need to quantize pitch, nor is there a problem if pitch varies during a "note" because notes are not represented explicitly. Furthermore, this approach can be extended to incorporate transcription uncertainty by representing pitch as a probability distribution.



Query string: 47.1, 47.4, 47.8, 47.8, 47.8, 49.4, 49.2, 49.6, 49.2, 48.4, 47.4, 47.7, 47.7, 47.7, 51.5, 52.6, …

Figure 5. An audio query is segmented into frames representing a melodic contour. Each frame corresponds to the pitch of a 100 ms timeslice.

To deal with pitch transposition, we transpose queries by many different pitch offsets; to deal with tempo variation, we "stretch" the target melodies in the database by different scale factors. In addition, we use a constrained form of time warping to align queries with the database entries. Constraints prohibit large deviations that would distort the overall contour shape.

This approach is obviously very slow and impractical for searching a large database. However, we believe it would be foolish to limit our research to the existing body of fast search algorithms. Instead, we hope to characterize some limitations of fast search algorithms and then try to overcome them. For example, a slow precise search might be used to refine the results of a fast, imprecise search. To evaluate the frame-based approach, we are replicating various event-based search engines from the literature so that we can compare different approaches using identical queries and databases. Preliminary results show that the frame-based approach is giving substantially better precision. For example, the frame-based approach ranked the correct match as the closest match in 13 out of 20 queries on a collection of 77 big band arrangements, and it ranked the correct match in the top 3 on 16 out of 20 queries. In contrast, an event-based search found the correct match in 5 out of 20 queries, with only 6 out of 20 queries ranked in the top 3 results [19].

8. PHONETIC-STREAM ANALYSIS

In addition to pitch and rhythm, singing as a natural form of query possesses time-varying acoustic-phonetic information, e.g., one sings the *words* of a song according to the pitch and duration of each note. While all three streams may provide useful information for searching the musical database, only the pitch stream has been studied in any detail, and almost no work has been done on the acoustic-phonetic stream. In the ideal case of errorless queries, the acoustic-phonetic stream is likely to be highly redundant with the rhythm and pitch streams, and, therefore, is expected to provide little additional information. In the practical case, where the rhythm and pitch streams may contain a significant number of errors, the acoustic-phonetic stream may be the only reliable source of information.

In its most general form, extracting the stream of phonetic information from the query is a problem in speaker-independent continuous speech recognition, for which a sizable body of research literature exists, all of which suggests that we should expect little success in the case of sung passages without substantial effort. Besides independence across speakers, the problem of speech recognition for singing is further exacerbated by the fact that non-vocalic segments of the stream are generally poorly represented, e.g., one cannot "sing" the fricative /f/. Furthermore, singing extends the pitch range upwards from the normal range of speaking to fundamental frequencies that generally cause problems for many standard recognition systems.

Our work [20] focuses on a reduced version of phonetic-stream analysis. Rather than attempting to transcribe the word that is sung into standard phonetic units, we have studied coarser quantizations of the phonetic stream, which trade robustness to production variations within and across singer against the information-bearing capacity of the stream. The algorithm we have developed extracts a symbol stream consisting of the Cartesian product of a "phonetic" alphabet of four vowel types (front, neutral, back, non-vocalic) and a duration alphabet of long and short.

Among the several approaches we have studied for segmenting the phonetic stream into the 8-element symbol stream, the most promising appears to be based on a self-referential model, as opposed to an absolute-referential one. In an absolute-referential model, queries are segmented based on templates constructed for the four vowel types over the entire population of singers. A self-referential model segments each query individually and then maps these segments to the most likely "universal" alphabet of front, neutral, back, and non-vocalic. Of the two approaches, the self-referential model appears in our preliminary studies to be more robust to such sources of variation in production as gender, training, song, and register.

The self-referential model utilizes nearest-mean reclassification (NMRA) to segment the query into four categories based on properties of the query's short-time Fourier transform. NMRA is performed on the entire set of short-time Fourier transforms to assign one of four raw categories to each time slice. Aggregation is performed across time slices to yield short and long classification of each vowel type. Finally, the raw categories are mapped into front, neutral, back, and non-vocalic labels based on features of the spectral distributions within and across the raw categories. The string for each query is then compared with the database to find the best matches.

Results from pilot studies suggest that the approach outlined above may be useful in music retrieval. A database of sung queries was constructed by having subjects sing one verse from seven familiar songs: "Happy Birthday", "Yankee Doodle", "America the Beautiful", the Beatles' "Yesterday", "Row, Row, Row Your Boat", "Somewhere Over the Rainbow", and "My Bonnie Lies Over the Ocean". Ten subjects were recruited from among staff, faculty, and graduate students at the Advanced Technologies Laboratory at the University of Michigan. Each subject sang four instances of each song. They were allowed to pace themselves and to choose whatever pitch range and tempo they felt most comfortable for each of the songs. Informal classification of the quality of singing ranged from very poor to excellent across the ten subjects.

Error! Reference source not found. shows the percent correct in the nearest neighbor matches as a function of query song. For each query, the nearest neighbor is found as the vowel stream that requires the fewest number of (weighted) edits to be transformed into the desired query. If that selected vowel stream is generated from the same song as the query, then the selection is deemed correct. There are a total of 40 queries for each song, for a total of 279 possible neighbors from which to select, as we do not count the query itself as a possible neighbor. Therefore, a 14% chance exists of a correct answer occurring at random.

Across all songs, the nearest neighbor selection is correctly chosen from the same song 60% of the time, and varies by song between 42% (for "Yankee Doodle") and 80% (for "America the Beautiful"). We interpret these results as supporting the general hypothesis that some representation of vowel stream is useful for music information retrieval.

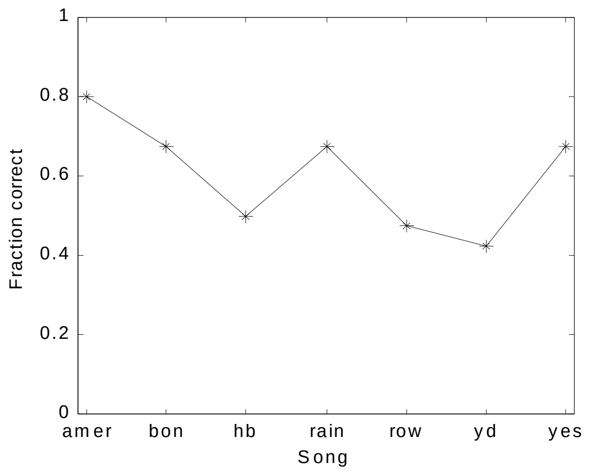


Figure 6. Fraction of correct nearest-neighbor choices as a function of the query song. A correct choice means that the nearest neighbor to the vowel stream is a vowel stream from the same song. There are 40 queries for each song.

9. STRUCTURAL ANALYSIS AND ABSTRACTION

In parallel with our work on melodic search, we are investigating methods by which we can deduce musical structure and genre. Information about structure has many uses. Since repetition is common in music, identifying repetition can aid in music processing and transcription as well as eliminate unnecessary searching. Listeners often remember the most-often repeated parts of a song, so search engines can give extra weight to these, and audio browsers can begin playback at these memorable moments. While a great deal of work has focused on the analysis of symbolic representations of music, we also wish to consider applications to databases of audio waveforms. Both of the approaches described below represent early efforts to derive structure from music audio.

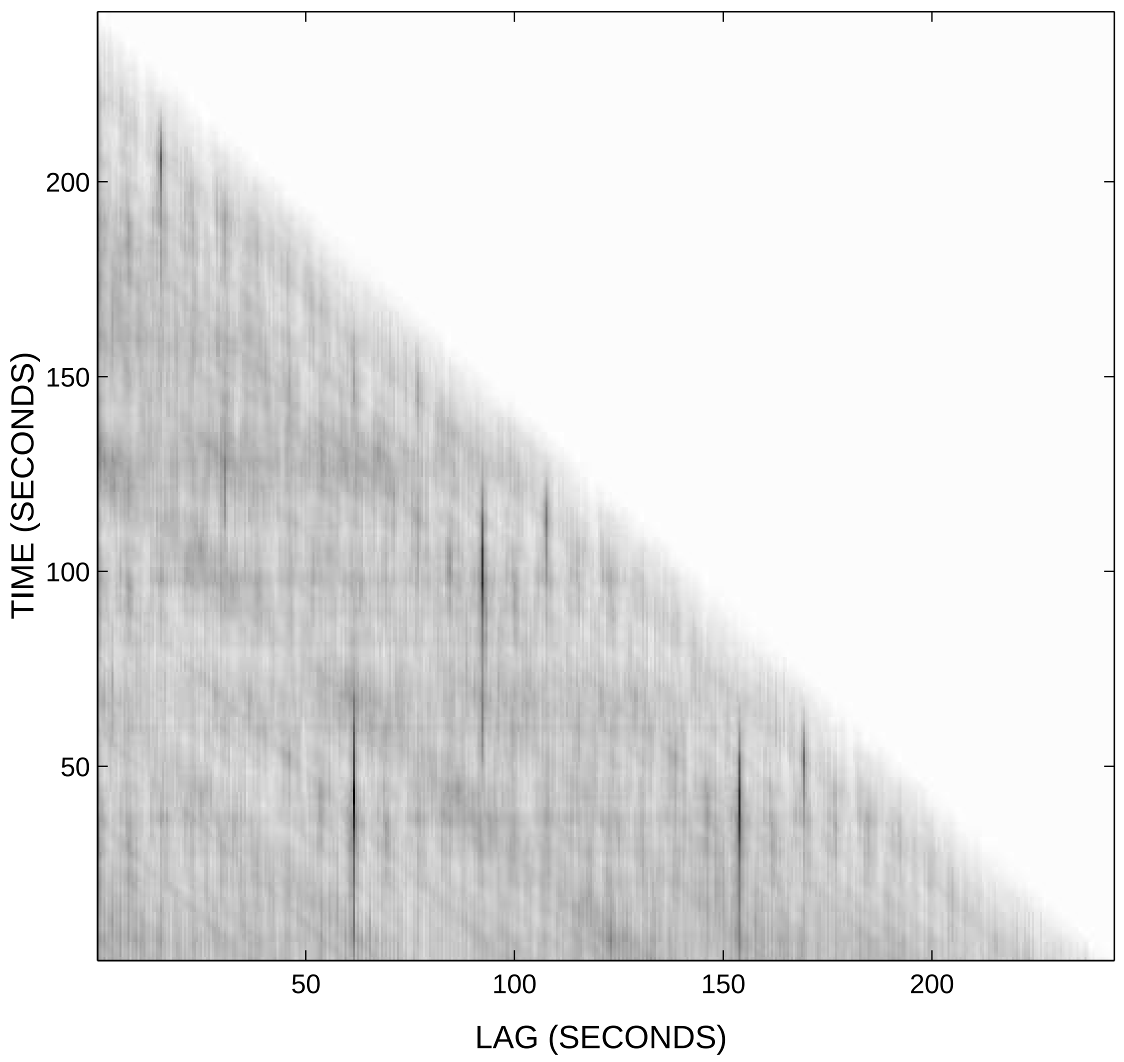


Figure 5. High correlation of chroma at a given lag indicates similarity. Therefore, vertical line segments represent repetition of musical material.

9.1 Chroma-Based Search for Structure

One approach in this direction is our "audio thumbnailing" algorithm [21]. Prior work in this area includes Logan and Chu, [22] who developed algorithms for finding key phrases in selections of popular music. Their work focused on the use of Hidden Markov Models and clustering techniques for mel-frequency cepstral coefficient (MFCC) representations of the acoustic waveform. Their system was subjectively evaluated on a relatively small selection of Beatles songs. In another work, Foote [23, 24] talks about audio "gisting" as an application of his proposed measure of audio novelty. This audio novelty score is based on a similarity matrix, which compares frames of audio based on features extracted from the audio. Foote leaves details such as the similarity metric and feature class as design decisions; however, he does recommends the use of MFCCs as a feature class for computing audio novelty.

Our approach to "audio thumbnailing" draws upon two key concepts: the chromagram and recurrent state. The chromagram is an abstraction of the time-varying spectrum of the audio signal which is based on the perceptual organization of pitch [25]. For each time frame, the chromagram maps the linear-frequency spectrum onto pitch-chroma spectrum, which ranges in semitones over the octave [26]. Among the mathematical properties of the chromagram is that it discounts octave relationships among the components of the frequency spectrum, which are highly redundant in harmonic sources, and places greater emphasis on pitch-class relationships, which bear information about harmony.

Recurrent state abstracts the concept of structural organization in a piece of music. The refrain in a song, for example, is distinguished from the rest of the piece only by virtue of the fact that it, alone, recurs throughout the piece. Thus, in searching for a refrain, or other such structures that repeat often in a piece of music, relatively little can be assumed about the structure save some basic unit of time, which establishes the scale of the structure.

Our system performs audio thumbnailing by examining the audio signal for recurrent states over time scales from 5 to 60 seconds in duration. To reduce redundancies in the representation of harmonic sources, each time-slice of the chromagram is quantized into twelve equal semitone bins. The 12-dimensional feature vectors are correlated across time and the correlation values are aggregated over windows of time to identify recurrent structures at a particular time scale.

Foote's similarity matrix is a convenient way to represent the feature-correlation space. Windowing, in this case, transforms the similarity matrix into a time-lag surface Figure 7 presents a time-lag surface for Jimmy Buffet's *Margaritaville*. A thumbnail for the piece of music is selected by locating the maximum element of the time-lag matrix subject to two constraints. To prevent the selection of quick repetitions and fading repeats, we require that the location have a lag greater than one-tenth the length of the song and occur less than three-fourths of the way into the song. The thumbnail is then defined by the time position of this maximum, which corresponds to the time that the first of the pair of sections begins, and the length of the window used for aggregating the data.

Our primary quantitative studies of the thumbnailing algorithm have been applied to a database of 93 selections of popular music, with styles including rock, folk, dance, country-western, and others [21]. Each song was hand-scored to identify the refrain or chorus segments of the song. The thumbnailing algorithm was then used to identify highly similar segments of music. Our general observations include the following. With respect to frame-level recall and precision rates, the thumbnail algorithm performs as high as 0.9, for proper choice of window. When it fails, it is often the case that the chorus or refrain is repeated, but there is some change, either in instrumentation or in the musical structure of the repeat. These cases violate our initial assumption of high correlation between instances of the chorus and indicate that this assumption would need to be relaxed under such circumstances. It is also interesting to note that thumbnailing based on the chromagram reduction of the audio stream clearly outperforms a comparable system based on MFCC's. We interpret this outcome to reflect the fact that MFCC's provide a low-order representation of the wideband spectrum, whereas the chromagram provides a low-order representation of the "wideband" harmonic content of the signal, by folding harmonically redundant regions of the spectrum into each other.

9.2 Melodic Pattern Analysis

Another effort in the direction of structural analysis also starts with audio but uses conventional autocorrelation-based pitch estimation to extract melodic contour. The top of Figure 8 shows audio taken directly from a commercial recording of a ballad, "Naima," by John Coltrane and performed by his jazz quartet [27]. Below the audio is a piano-roll display of a pitch transcription, accomplished using a straightforward autocorrelation algorithm for pitch estimation. At the bottom of the figure is the analysis, discussed below.

Taking inspiration from the frame-based melodic contour comparison described in Section 7 and the chroma-based correlation analysis of Section 9.1, the analysis procedure computes the length of similar melodic contours starting at all pairs of locations *i* and *j*, which index note position. The matrix $M(i, j)$ is defined as the duration of similar contours starting at locations *i* and *j*. ($M(i, j)$ is mostly zero.) For example, there is a repeated 4-bar phrase at the beginning of the piece, starting at the first and seventh notes. Note that where $M(i, j)$ is non-zero, there will tend to be a slightly shorter duration at $M(i+1, j+1)$. For example, there are similar phrases starting at notes 2 and 8. These "implied" entries are "removed" by setting them to zero.

After eliminating these implied entries, clusters of similar melodic contours are formed. For example, similar lengths at *i*,*j*, *j*,*k*, and *k*,*i* imply that three similar contours are located at *i*, *j*, and *k*. In the saxophone solo, there is a third repetition of the opening four bars near end of the excerpt shown in Figure 8.

After simplifying the matrix *M* and forming clusters from the remaining melodic fragments, a greedy algorithm attempts to "explain" all notes of the transcription in terms of these clusters. The shaded bars at the bottom of Figure 8 locate similar fragments. It can be seen from this that the melody consists of a repeated phrase followed by a shorter repeated phrase and a third phrase. This is followed by a return to the opening phrase. These phrases return after a piano solo (not shown).

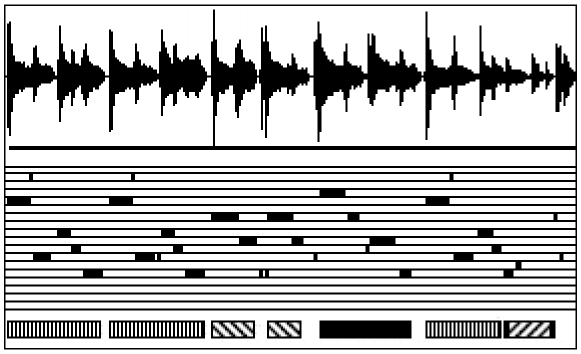


Figure 8. Audio from jazz quartet (top), automatic transcription (middle), and analysis (bottom). Similar shading indicates similar melodic fragments.

This work is at an early stage. It has produced an excellent analysis of a performance of "Naima." This example was chosen because of the clear, simple lines and structure and the dominance of the saxophone in the jazz quartet recording (and also because it is a wonderful ballad). Work is underway to adapt these methods to more challenging examples.

9.3 Genre Classification

Although we have argued that not all searches should be conducted within a given genre, there are certainly many cases where users can narrow their search by specifying music categories. We have investigated the use of machine learning techniques to perform music classification. We trained a neural network classifier on audio power spectra measured within windows of several seconds of duration of different genres. To classify a piece, we divide the piece into many window-sized chunks and run the classifier on each chunk. The overall class is the one reported for the greatest number of chunks. This approach correctly classified all 80 pieces in a database of digital audio as rock, jazz, country, or classical. Much finer classification is desirable, and we hope to incorporate new methods into our architecture as they become available.

10. SUMMARY

We have presented an architecture for music retrieval. The MUSART architecture is motivated by the need to explore and ultimately rely on multiple mechanisms for representation, abstraction and search. We have made progress toward more robust and flexible music databases. Our work on Markov models provides a new an approach to musical abstraction and retrieval. Our frame-based melodic search and phonetic-stream search deal specifically with problems of audio-based queries. Additional work addresses the problems of audio analysis, the identification of musical structure, and music classification.

In the future, we will refine all of these techniques and integrate them into the architecture we have presented. We also need to explore many user interface issues. The benefits will include the ability to combine multiple search strategies and a more formal approach to the evaluation and comparison of music retrieval techniques.

11. ACKNOWLEDGMENT

This material is supported by NSF Award #0085945 and by an NSF Fellowship awarded to Mark Bartsch. The views expressed here are those of the authors. We thank Jonah Shifrin for implementing tphe Viterbi algorithm and generating the data reported in this paper. We also thank Bryan Pardo and Erica Schattle for comments on previous versions of this paper.

12. REFERENCES

[1] Dunn, J.W., C.A. Mayer. *VARIATIONS: A digital music library system at Indiana University*. in Digital Libraries. 1999: ACM.

[2] Kornstadt, A., *Themefinder: A Web-based Melodic Search Tool*, in *Melodic Similarity Concepts, Procedures, and Applications*, W. Hewlett and E. Selfridge-Field, Editors. 1998, MIT Press: Cambridge.

[3] McNab R.J., L.A.Smith, D. Bainbridge and I. H. Witten, *The New Zealand Digital Library MELody inDEX. D-Lib Magazine*, 1997. May.

[4] Bainbridge, D. *The role of Music IR in the New Zealand Digital Music Library project.* in *Proceedings of the International Symposium on Music Information Retrieval, 2000*.

[5] Tseng, Y.H. *Content-based retrieval for music collections*. in *SIGIR*. 1999: ACM.

[6] McNab R.J., L.A.S., Ian H. Witten, C.L. Henderson, S.J. Cunningham. *Towards the digital music library: tune retrieval from acoustic input*. in *Digital Libraries*. 1996: ACM.

[7] Blum, T., D. Keislar, J. Wheaton, E. Wold, *Audio databases with content-based retrieval*, in *Intelligent multimedia information retrieval*, M.T. Mayberry, Editor. 1997, AAAI Press: Menlo Park.

[8] Rolland, P.Y., G. Raskinis, J.G. Ganascia. *Musical content-based retrieval: an overview of the Melodiscov approach and system*. in *Multimedia*. 1999. Orlando, FL: ACM.

[9] Rothstein, J., *MIDI: A Comprehensive Introduction*. 1992, Madison, WI: A-R Editions.

[10] Hewlett, W. and E. Selfridge-Field, eds. *Melodic Similarity Concepts, Procedures, and Applications*. Computing in Musicology. Vol. 11. 1998, MIT Press: Cambridge.

[11] Sankoff, D. and J.B. Kruskal, eds. *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*. 1983, Addison-Wesley: Reading, MA.

[12] Downie, J. S. *Informetrics and music information retrieval: an informetric examination of a folksong database*. in *Proceedings of the Canadian Association for Information Science, 1999 Annual Conference, 1999. Ottawa, Ontario.*

[13] Zhang, T., C.C.J. Kuo, *Hierarchical system for content-based audio classification and retrieval,* University of Southern California: Los Angeles.

[14] Alamkan, C., W. Birmingham, and M. Simoni, Stochastic Generation of Music, 1999, University of Michigan.

[15] Pardo, B. and W. Birmingham. *Automated Partitioning of Tonal Music*. *FLAIRS, 2000.* Orlando, FL.

[16] Meek, C. and W. Birmingham. *Thematic Extractor. International Symposium on Music Information Retrieval* in Second *International Symposium on Music Information Retrieval.* 2001.

[17] Barlow, H. *A Dictionary of Musical Themes.* Crown Publishers. 1983.

[18] Rand, W. and W. Birmingham. *Statistical Analysis in Music Information Retrieva*l in Second *International Symposium on Music Information Retrieval.* 2001.

[19] Mazzoni, D., and R. B. Dannenberg, *Melody Matching Directly from Audio* in *Second International Symposium on Music Information Retrieval.* 2001.

[20] Mellody, M., Bartsch, M., and G. H. Wakefield. *Analysis of Vowels in Sung Queries for a Music Information Retrieval System* submitted for publication in *Journal of Intelligen tInformation Systems*. 2001.

[21] Bartsch, M., and G. H. Wakefield. *To Catch a Chorus: Using Chroma-Based Representations For Audio Thumbnailing* in *Proceedings of the Workshop on Applications of Signal Processing to Audio and Acoustics*, 2001.

[22] Logan, B., and S. Chu. Music summarization using key phrases in *International Conference on Acoustics, Speech, and Signal Processing*, 2000.

[23] Foote, J. *Automatic audio segmentation using a measure of audio novelty* in *Proceedings of IEEE International Conference on Multimedia and Expo*, 1999.

[24] Foote, J. *Viusalizing music and audio using self-similarity* in *Proceedings of ACM Multimedia*, 1999.

[25] Shepard, R. *Circularity in judgements of relative pitch* in *Journal of the Acoustical Society of America, 36, 2346-2353,* 1964.

[26] Wakefield, G.H. *Mathematical Representation of Joint Time-Chroma Distributions*. in *Intl. Symp. on Opt. Sci., Eng., and Instr., SPIE'99*. 1999. Denver.

[27] Coltrane, J. Naima on the album *Giant Steps.* Atlantic Records. 1960.

ISMIR 2001 Keynote Address

Computer Analysis of Musical Allusions

David Cope

Music Department, Division of Arts
University of California, Santa Cruz
244 Music Center, UCSC
Santa Cruz, CA 95064

howell@cats.ucsc.edu

ABSTRACT

I will describe here a computer program called Sorcerer. Sorcerer uses what I call *referential* analysis, a semiotic approach roughly situated between hermeneutic and Rétian analyses, which associates patterns found in a target work—music under study—with several potential source works—music assumed to either influence or be influenced by the target work. Sorcerer then presents these patterns as possible references called allusions. The program lists its findings without regard for whether the composer of the target work consciously or subconsciously referenced the source work, only that the found allusions exist. I will further describe the possible relevance and importance of this type of analysis as a complementary approach to more standard harmonic, melodic, and formal types of analysis, as a method for performers to better interpret the music they play, and as one possible approach to the deeper understanding of meaning in music.

1. INTRODUCTION

Allusions have occurred throughout music history, though very few systematic studies of them have taken place. With the exception of Deryck Cooke's landmark *The Language of Music* [3], for example, few articles or books are devoted exclusively to allusions. Certain musical forms like organum, motets, cantatas, and other *cantus firmus*-based styles often depend on allusions. As well, certain composers' styles, like those of Ives and Berio use allusions extensively, deliberately, and obviously. Late Romantic Russian composers used allusions as a benchmark of their nationalism, considering them not only honorable but requisite to their stylistic heritage.

2. DISCUSSION

I developed Sorcerer in 1995 as analytical software designed to discover the possible sources of a target work in various works chosen as source music. Sorcerer may incorporate pitch and/or rhythm in contiguous or non-contiguous sequences in its searches. Source music may consist of works composed prior to or after the target work depending on whether the desired information is intended to portray what might have influenced the target work or what works the target work might have influenced. In fact, source works may consist of both music composed previously and after the target work as long as one clearly distinguishes their chronological relationships.

I classify allusions into five basic categories—moving from nearly exact quotation to the use of more common musical conventions. I do not use negative words such as *cliché* here, trying to avoid the stigma of what some feel as weakness in art and which I feel can be a strength. My taxonomy for referential analysis includes:

(1) **Quotations**—as in citations, excerpts, or renditions;

(2) **Paraphrases**—as in variations, caricatures, or transcriptions;

(3) **Likenesses**—as in approximations, translations, or similarities;

(4) **Frameworks**—as in outlines, vestiges, or redactions;

(5) **Commonalitie**s—as in conventions, genera, or simplicities.

As can be seen, potential for listener recognition proceeds from strong to weak through these categories and the potential for stylistic integration proceeds inversely. A more detailed description of each of these categories will be presented through visual and aural examples.

Using Sorcerer involves three primary considerations: choosing the proper music, pattern matching, and interpreting the program's output. Choosing appropriate source music for a particular target work is critical to producing useful results. Documentation of the relationship of the target music composer to the composer(s) of the source work can further enhance the logic of selecting appropriate music. Once works have been chosen and checked for errors they are algorithmically encoded for pattern matching.

To pattern match source music with a target work, particularly when matching non-contiguously, requires a general understanding of the basic principles of pattern matching. In order to make a favorable comparison between two patterns, a pattern matcher should:

(a) compare intervals rather than notes;

(b) allow for certain variations in interval size within logical boundaries;

(c) allow the interpolation of a certain number and type of notes between hierarchically more important notes.

In order to expedite this process, I employ an approach which stores source music as a series of patterns rather than as complete works. In other words, the target and source music is segmented into patterns before the actual pattern matching takes place. The advantage of this approach is that re-matching with different settings can take place almost immediately after a matching session. The disadvantage of this approach is that any variations in pattern size requires a complete overhaul of the source music storage, a problem that will discourage most users from even attempting it. The following detail of the storage process and the resultant pattern matching will, I hope, make these advantages and disadvantages clearer.

Patterns must fit within a certain size limit governed by the user. This size should not be so large that it unnecessarily stores music in pattern sizes which will never match other patterns. At the same time, this size should not be so small that it restricts the possibility that slightly larger patterns will match. Setting this maximum size then requires some experience both with pattern matching in general and with the music being used as target and as potential sources.

Once collected, patterns are stored (along with information about their original locations) in special lexicons which are named according to their initiating intervals. Thus all patterns beginning with a second (major or minor) are stored in the same lexicon, and so on. Within each interval lexicon, patterns are further stored in specialized lexicons named according to their second intervals. Within each of these lexicons, patterns are further stored according to their third intervals, and so on. Thus, a pattern consisting of the intervals of a major third, minor second, unison, perfect fourth, and perfect fifth will be stored in the fifth lexicon, within the fourth lexicon, within the unison lexicon, within the second lexicon, within the third lexicon.

To compare a pattern from a target work to a pattern from a source work, then, requires only that the pattern matcher find the appropriate lexicon and then the most appropriate pattern within that lexicon, if there is one. In other words, the process ensures that patterns are compared only to patterns that the approach guarantees will match and avoids the extensive matching of patterns which have no chance of matching (required of more standard pattern matchers). Pattern comparisons survive only as long as an appropriate lexicon exists. This process makes comparing patterns very efficient.

Once pattern matching is complete, users are presented with a mosaic of overlapping potential allusions. Awareness of these allusions can have significant impact in the making and interpretation of music. For example, knowledge of allusions can indicate to performers what to emphasize in various melodic lines and what performance practice to use for certain passages. Knowledge of the location and substance of allusions informs listeners about how to listen to a work, allowing them to link their current experience with other experiences they've had previously with the referenced music. A knowledge of allusions will also provide hints as to how other types of analysis should be used to reveal more analytical information about a work. Allusions can also reference music to traditions of the past or indicate a composer's familiarity with an individual theme or with a class of themes.

3. Conclusions

Ultimately, users must take meaning from Sorcerer's output for themselves. However, I here assert a few general observations:

(a) that all music consists, at least in part, of allusions to other music, providing listeners with a sense of familiarity beyond that of style recognition;

(b) that understanding the presence and origins of allusions gives us a better understanding of music's deeper context;

(c) that awareness of certain allusions can provide interesting and insightful information about the composer of a target work in terms of what that composer finds important in music and to some degree how that composer listens to music;

(d) that tracing lineages of allusions can help define a genealogy of musical styles and influences.

Understanding traditional analytical techniques such as tonal harmonic function as well as formalisms such as canons and fugues, and so on, can have immense value. However, such analyses do not provide a full understanding of music. Such an understanding only comes with the addition of more semantic analysis such as those that referential analysis provides.

4. References and Suggested Readings

[1] Agawu, V. Kofi. *Playing with Signs*. Princeton: Princeton University Press, 1991.

[2] Burkholder, J. Peter. The uses of sxisting music: Musical borrowing as a field. *Notes* 50 (March 1994), 851-70.

[3] Cooke, Deryck. *The Language of Music*. New York: Oxford University Press, 1959.

[4] Cope, David. *Computers and Musical Style*. Madison, WI: A-R Editions, 1991.

[5] ———. *Experiments in Musical Intelligence*. Madison, WI: A-R Editions, 1996.

[6] ———. *The Algorithmic Composer*. Madison, WI: A-R Editions, 2000.

[7] ———. *Virtual Music*. Cambridge, MA: The MIT Press, 2001.

[8] Gherdingen, Robert. *A Classic Turn of Phrase*. Philadelphia: University of Pennsylvania Press, 1988.

[9] LaRue, Jan. Significant and coincidental resemblance between classical themes. *Journal of the American Musicological Society* 14 (Summer 1961), 224-34.

[10] Meyer, Leonard. *Style and Music*. Philadelphia: University of Pennsylvania Press, 1989.

[11] Scott, Hugh Arthur. Indebtedness in music. *The Musical Quarterly* 13/4 (1927), 497-509.

Musical Works as Information Retrieval Entities: Epistemological Perspectives

Richard P. Smiraglia
Professor,
Palmer School of Library and Information Science
Long Island University
720 Northern Blvd.
Brookville, NY 11548
(516) 299-2174
Richard.Smiraglia@liu.edu

ABSTRACT

Musical works form a key entity for music information retrieval. Explicit linkage of relationships among entities is critical for document-based information retrieval. Works contain representations of recorded knowledge. Core bodies of work—canons—function to preserve and disseminate the parameters of a culture. A *musical work* is an intellectual sonic conception. Musical works take documentary form in a variety of *instantiations*. Epistemology for documentary analysis provides key perceptual information about the objects of knowledge organization. Works are carriers of knowledge, representing deliberately-constructed packages of both rational and empirical evidence of human knowledge. Smiraglia (2001) suggests the parameters of a theory of the work, incorporating the tools of epistemology to comprehend works by expressing theoretical parameters in the context of a taxonomic definition. A work is a signifying, concrete set of ideational conceptions that finds realization through semantic or symbolic expression. Semiotic analysis suggests a variety of cultural and social roles for works. Musical works, defined as entities for information retrieval, are seen to constitute sets of varying instantiations of abstract creations. Variability over time, demonstrated empirically, is an innate aspect of the set of all instantiations of a musical work, leading to complexity in the information retrieval domain.

1. INTRODUCTION

Musical works (as opposed to musical documents, such as scores or recordings of musical works) form a key entity for music information retrieval. Ultimately, searches for a given musical work rely on the hope of subsequent selection of instantiation in one of several documentary formats. Musical works have been variously and industriously described by musicologists and music bibliographers. However, in the information retrieval domain, the work as opposed to the document, has only recently received focused attention (Smiraglia 2001). Efforts to define works as information retrieval entities and to document their occurrence empirically are quite recent. In fact, systems for bibliographic information retrieval, and more recently for information storage and retrieval, have been designed with the document as the key entity, and works have been dismissed as too abstract or difficult to define empirically to take a role in information retrieval. Recent work, summarized in Smiraglia (2001), points to the primacy of works for bibliographic information retrieval, and to the importance of works as concepts for all text-based information storage and retrieval systems. In this paper, definitions of works as entities (from the information retrieval perspective) and of musical works (from the musicological perspective) are examined. A taxonomic definition is presented. An epistemological perspective, including empirical evidence, aids in understanding the components of the taxonomic definition. Musical works, thus defined as entities for information retrieval, are seen to constitute sets of varying instantiations of abstract creations.

2. Documentary Entities

A documentary entity is a unique instance of knowledge (e.g., a thesis, a sculpture, a research report, etc.). Each documentary entity has physical and intellectual properties. A containing relationship exists between these two properties. That is, the physical property is the package for the intellectual. The explicit linkage of relationships among documentary entities is critical for document-based information retrieval. Empirical research techniques have illuminated the technical problems of bringing the objective of collocating works, as opposed to documents, into primary position. Tillett (1987) sought to classify and quantify the entire range of bibliographic relationships--relationships that exist among documentary entities. Smiraglia (1992) investigated the derivative relationship, which holds among all versions of a work, refining its definition to include several different categories of derivation. These categories are:

·simultaneous derivations

·successive derivations

·translations

·amplifications

·extractions

·adaptations, and

·performances.

Leazer (1993 and 1994) described a conceptual schema for the explicit control of works in catalogs, taking into account both

Tillet and Smiraglia's taxonomies of relationship types. Leazer and Smiraglia studied the presence of derivative relationships in the OCLC WorldCat (Smiraglia and Leazer 1995 and 1999, Leazer and Smiraglia 1996 and 1999) affirming the taxonomy of derivative relationship types. Yee examined problems of relationships among moving image materials, including the substantial problems of associating bibliographic records for varying instantiations of films. Vellucci (1997) examined musical works and found that the categories Tillett and Smiraglia had suggested were present, and in large numbers; 85.4% of the works in her sample drawn from the catalog of the Sibley Music Library demonstrated derivative relationships. Vellucci also postulated two new categories of derivation applicable only to musical works: musical presentation, and notational transcription.

A 1998 report by a study group of The International Federation of Library Associations (IFLA) was devoted to outlining functional requirements for bibliographic records. Representing the products of intellectual or artistic endeavor, the report suggested a group of documentary entities *works, expressions, manifestations,* and *items*. A work was described as a distinct intellectual or artistic creation, an expression as the intellectual or artistic realization of a work. The entities work and expression reflected intellectual or artistic content. A manifestation embodied an expression of a work, which was in turn embodied by an item. The entities manifestation and item, then, reflected physical form. The report noted that a work might be realized through one or more expressions, which might be embodied in one or more manifestations, which in turn might be exemplified in one or more items (IFLA 1998, 12-13).

3. WORKS AS VEHICLES FOR COMMUNICATION

Works contain representations of recorded knowledge. Works are created deliberately to represent the thoughts, data, syntheses, knowledge, art and artifice of their creators. Works, then, serve as vehicles to communicate one or more of these aspects of new knowledge to potential consumers (readers, scholars, etc.). Consumers of works may and often do use them to inform their own new works, which likewise serve as vehicles to communicate knowledge across time and space to new consumers. In this manner, we can observe the social role of works. Therein we see works as vehicles that transport ideas along a human continuum, contributing to the advancement of human knowledge in specific ways and to the advancement of the human social condition in more general ways.

Saussure described a system for the study of the life of signs in a society, which he named *semiology* (1959, 16). Smiraglia (2001) has used Saussure's system to demonstrate the cultural role of works. Works function in a manner analogous to signs, uniting the conceptual with the semantic, and demonstrating the two properties *immutability* and *mutability*. Peirce and his school of semiotics also shed light on the mutability of signs and the probability of their varying perception across chronological and cultural barriers. Peirce ([1894] 1998, 5) asserted a triad of types of signs: a) likenesses, which convey ideas of the things they represent through imitation; b) indications, which show something about things by being physically connected with them; and c) symbols, or general signs, which have become associated with their meanings by usage. The meaning of a symbol is not fixed, but rather is a function of its perception. Barthes also described reception mutability, suggesting that consumers of works were not concerned so much with the integrity of a text as with their own experience of it (1975, 11). For example, an individual work might be consulted for information, it might be used for recreation, or it might form the basis of a scholar's discourse. Barthes suggests that in essence a text is as though it were *tissue* (1975, 64). Poster (1990) suggested that cultural history was demarcated by variations in the structure of symbolic exchange. In literate society, works are the vehicles that facilitate the propagation of culture through formal symbolic exchange.

Works can be seen as analogous to signs that are mutable over time. The texts of works act as signifiers, seemingly immutable when first fixed, but with other properties (such as cultural identity) that are themselves very mutable indeed. Works are vehicles of culture, entities that arise from a particular cultural perspective. As such they are vehicles with certain cultural obligations--among them dissemination and propagation of the culture from which they spring. This analogy has been demonstrated graphically by Smiraglia (2001) and is reproduced in Figure 1.

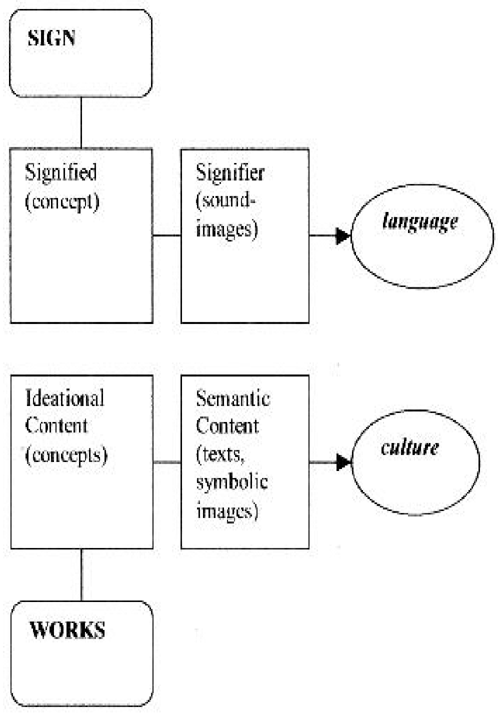

SIGN
Signified (concept)
Signifier (sound-images)
language
Ideational Content (concepts)
Semantic Content (texts, symbolic images)
culture
WORKS

Figure 1. Works are Analogous to Signs

4. WORKS AS ELEMENTS OF CANON

Each work is in some way a part of a larger body of related work. These bodies of work derive meaning from their function in culture as well as from their relations with other works and other bodies of work. Individual works derive meaning from their relations to their human receptors. These core bodies of work, sometimes referred to as canons, function to preserve and disseminate the parameters of a culture by inculcating cultural values through the information conveyed as a whole and in each of the works that comprise them. Smiraglia and Leazer (1999) reported that the size of a family of instantiations of a work seems to be related to its popularity, or ... its *canonicity*. Most families are formed and reach full size soon after publication of the progenitor. On the other hand, older progenitors are the locus for larger families.

Relations that are observed among works in a canon are thought to be conventional rather than natural. That is, they are functions of their roles in the culture from which they spring rather than determined by any inherent characteristics. Eggert (1994) described a phenomenological view of works of art, seeing works as ongoing entities that incorporate across their chronological existence all of the reactions of those who encounter them. Through the vehicle of works, culture is continually communicated. Works have no unchanging existential anchor, no single perfect exemplar. Rather they derive much of their meaning from their reception and continuous reinterpretation in evolving cultures. Works follow the same pattern as Saussure's linguistic signs, mutating across time through the collaboration of the cultures that embrace them. Works are shaped by their audiences, and they reflect the functional requirements of those who will use them. Therefore, works are artifacts of the cultures from which they arise.

5. MUSICAL WORKS

A *musical work* is an intellectual sonic conception. Musical works take documentary form in a variety of *instantiations* (i.e., a sounding of it as in performance, or its representation in printing as in score). The primary purpose of any physical instantiation of a work is to convey the intellectual conception from one person to others. Because musical works fundamentally are meant to be heard, physical instantiations are not of primary importance in the exchange between creator and consumer. Rather, they are media through which musical ideas captured at one end of the continuum may be reproduced so that they may be absorbed at the other. Defining a musical work as a sonic conception allows us to bridge the difficulty that arises between works that are *composed* (such as those in the supposed canon of Western Art Music) and those that are improvised or otherwise realized primarily through performance. In information retrieval, it is critical to make a distinction between the physical artifactual document, on the one hand, and its musical content, on the other.

Because a musical work must first exist in time to be apprehended by an audience, the more accurate instantiation of a musical work truly is likely its performance. Krummel (1988) argues that music is an entity that occurs in time, not on paper. Each performance is a "re-creation" of the work. A performance of a musical work, and by extension a recording thereof, delineates the time factor of a musical work for the receiving audience. For Dahlhaus (1983), the musical work actually inheres in the receiving audience.

Krummel (1970) summarized the historical use of musical documents, which serve as evidence of musical works that have existed and perhaps been performed in the past. He wrote (16): "Behind both [score and performance], apart from but governing both, as something of a Platonic ideal, is the abstract concept of the work of music itself." That is, a musical work (like any creation) is existentially viewed as an abstract concept in time rather than a particular physical entity in space. Scores, performances (and recordings) represent instances of the work, none of which can be equated fully with the work itself. Nattiez (1990) described a semiology of music that comprehends musical *works* as multi-dimensional because their realization is in sound. Goehr (1992) pointed to the human's natural tendency to take musical works for granted, enjoying their reception but without any clear understanding of the complexity of their origin or existence. Goehr posited an imaginary museum of works--imaginary to those who cannot see beyond the objectification of works of sonic art. With Nattiez and Goehr we approach the concept of mutability of works one step further. That is, we can clearly comprehend works that might have no concrete tokens--as literary works have words on paper--but which find their realization in sonic performances, each of which is uniquely created and uniquely perceived. Ingarden (1986) approached the central problem of the nature of a musical work by considering that the work represents a congruence between the composer and the listener. Talbot (2000) includes eleven papers on the musical work-concept, demonstrating little consensus on the historical meaning of the concept or its time of origin. There is however, convergence that a musical work must be discrete, reproducible and attributable (Talbot 2000, 3). The volume is filled with criticisms of the concepts of the musical work and the attendant canons. Curiously, just as scholars of information storage and retrieval and of knowledge organization have turned their attention to the concept of the work as an entity for information retrieval, musical scholars seem to be less sanguine about the concept.

Thomas and Smiraglia (1998) reflect on more than a century of formal rules for the cataloging of musical documents, speaking to the cataloging community at the point at which video-recordings of musical performances have become entities for documentary retrieval. They described the nature of the musical work as an entity for information retrieval, suggesting the concept functions in the manner of a surname for a family, around which cluster all instantiations known by that concept-name in horizontal, but explicitly described, relations.

Ligabue (1998) attempts to discover a real process of semiosis in music, beginning with the understanding that every sign is essentially inherently empty--a signifier without signification. Thus a sign finds its meaning revealed only within a relational context (p. 35). In music, single, isolated sounds can offer only pure information about themselves; only when contextualized does a sound acquire specificity. Therefore, sounds "become meaningful only and exclusively in relation to a context (p. 37)" In other words, sounds alone are no more musical signs than are letters or words linguistic signs. Rather, the semiosis is context-dependent. Signs are cultural constructs, and musical signs, like linguistic signs, depend on specific cultural contexts for their meaning. Liguabue demonstrates that "within organized sound systems, the perceptive act undergoes a mental rationalizing process [which is] culturally determined." Therefore, he writes, music as organized sonic events demonstrates a signification process analogous to other semiotic systems (p. 43):

> Meaning is not to be found *among* notes, but *in* them, *even if it manifests itself only among them.* Therefore if a sign only exists in virtue of another sign, which, though different, shares its nature but not its essence, the same thing occurs in music where each note has its precise meaning, which expresses itself in its specificity but can manifest itself only in the wholeness of the system. This manifestation takes place in a musical context according to existing modes which cannot be the same as those of the verbal context.

He concludes, that what is heard or listened to (in other words, what is signified) is in essence different from the acoustic physical phenomenon, and is interpreted within conventional cultural behaviors symbolically interpreted.

Hamman (1999) wrote about the role of computers in music composition, asserting that computers generate semiotic rather than symbolic frameworks. Hamman suggests that a composer is not only producer of musical artifacts, which he defines as “pieces,” “sounds,” etc. (in other words, works). Rather, the composer (102): “makes traces of processes by which abstract ideas are concretized according to particular performances and interactions *vis* a task environment.” Turino (1999), like Ligabue, asserts a Peircian semiotic theory of music in which components of musical units (that is, works) such as pitch, scale, tempo, etc. function as components of signs. The present paper relies on applied semiotics to demonstrate the effect of the social role of works on their complexity as entities for information retrieval. Turino’s semiotic analysis demonstrates the complex functioning of music and its components as signs at a meta-level. Echoing the comments of Eggert and Poster, van Leeuwen (1998) suggests a systemic-functional semiotics of music in which music is seen as an abstract representation of social organization, concerned with meta-level cultural interactions that find their expression in music functioning as signs.

6. EPISTEMOLOGY, KNOWLEDGE ORGANIZATION, INFORMATION RETRIEVAL

Epistemology is the division of philosophy that investigates the nature and origin of knowledge. Poli (1996) contrasted the tools of ontology and epistemology for knowledge organization, suggesting that where ontology represents the "objective" side of reality, epistemology represents the "subjective" side. Ontology ("being") provides a general objective framework within which knowledge may be organized, but epistemology ("knowing") allows for the perception of the knowledge and its subjective role. Olson (1996) used an epistemic approach to comprehend Dewey’s classification, asserting a single knowable reality reflected in the topography of recorded knowledge. Dick (1999) described epistemological positions in library and information science. He suggested that experience (empiricism) provides the material of knowledge, and reason (rationalism) adds the principles for its ordering. Rationalism and empiricism supply the basic platform for epistemological positions.

Hjørland (1998) asserts a basic epistemological approach to base problems of information retrieval, particularly to the analysis of the contents of documentary entities. He begins from a basic metaphysical stance, stating that ontology and metaphysics describe what exists (basic kinds, properties, etc.), whereas epistemology is about knowledge and ways in which we come to know. Hjørland lists four basic epistemological stances:

·Empiricism: derived from observation and experience;

·Rationalism: derived from the employment of reason;

·Historicism: derived from cultural hermeneutics; and,

·Pragmatism: derived from the consideration of goals and their consequences.

Hjørland describes a domain-analytic approach to subject analysis, recognizing that any given document may have different meanings and potential uses to different groups of users. Hjørland and Albrechtsen (1999) delineate recent trends in classification research, demonstrating the utility of Hjørland’s epistemological framework for deriving categories.

Marco and Navarro (1993) described contributions of the cognitive sciences and epistemology to a theory of classification. They suggest that (p. 128):

> The study of epistemology is, therefore, essential for the design and implementation of better cognitive strategies for guiding the process of documentary analysis, particularly for indexing and abstracting scientific documents. The ordering and classifying of information contained in documents will be improved, thus allowing their effective retrieval only, if it is possible to discover the conceptual framework (terms, concepts, categories, propositions, hypotheses, theories, patterns, and paradigms) or their authors from the discursive elements of texts (words, sentences and paragraphs).

Epistemology, then, is concerned with the theory of the nature of knowledge. The potential uses of epistemology for documentary analysis are many; a few have been attempted. Whereas ontology may be relied upon to frame the organization of knowledge, epistemology provides us with key perceptual information about the objects of knowledge organization. Empiricism can lead us to taxonomies of knowledge entities. Rationalism can demonstrate the cultural role of, and impact on, knowledge entities.

Works are key carriers of knowledge, representing not simply raw data or facts, but deliberately-constructed packages of both rational and empirical evidence of human knowledge. The organization of works for information retrieval along topical and disciplinary lines has been the key task of knowledge organization, specifically of classification. But works, too--especially those with canonical importance, have been organized using inadequate alphabetico-classified orders.

For instance, we can take the example of a well-known musical work, Beethoven’s *Moonlight sonata*. An important part of Beethoven’s oeuvre, this popular work has become a cultural icon. Quite aside from its formal performance, the lilting arpeggios are associated in the public imagination with concepts of nighttime and sleep. The work has demonstrated Eggert’s concept of canonical mutation by becoming part of our cultural consciousness. As Ligabue and Turino suggest, the signifying role of the *Moonlight sonata* is grounded in the personal experience of listeners over time and across cultures. In the summer of 2000, it was used as background for a television commercial for a new sleep-inducing medication.

In Figure 2 we see an array of descriptions of physical instantiations of this work in a typical online bibliographic retrieval system. As is often the case, this array consists of traditional name-title citations, qualified by publisher and date. Note there is no differentiation among the citations that can indicate any sort of variation among the sonic instantiations they represent.

Beethoven, Ludwig v	Moonlight.	E.F. Kalmus, 1970
Beethoven, Ludwig v	Moonlight Sonata.	presso Gio. Ca 1802
Beethoven, Ludwig v	Moonlight sonata.	G.D. Russell & 1863
Beethoven, Ludwig v	Moonlight sonata.	F. A. North & 1872
Beethoven, Ludwig v	Moonlight sonata.	Schirmer, 1894
Beethoven, Ludwig v	Moonlight sonata.	T. Presser, 1900
Beethoven, Ludwig v	Moonlight sonata	Carl Fischer, 1906
Beethoven, Ludwig v	Moonlight sonata.	Century Music, 1906

Beethoven, Ludwig v Moonlight sonata.	Fischer, 1906
Beethoven, Ludwig v Moonlight sonata.	Carl Fischer, 1916
Beethoven, Ludwig v Moonlight sonata	H.W. Gray, 1918
Beethoven, Ludwig v Moonlight sonata.	Angel Publicat 1961
Beethoven, Ludwig v Moonlight sonata.	Shattinger-Int 1971
Beethoven, Ludwig v Moonlight sonata.	Lyra Music Co. 1975
Beethoven, Ludwig v Moonlight sonata	The Hornists' 1978
Beethoven, Ludwig v Moonlight sonata.	G. Schirmer ; 1980
Beethoven, Ludwig v Moonlight sonata.	Alfred Pub. Co 1986
Beethoven, Ludwig v Moonlight sonata	Alfred Pub. Co 1991
Beethoven, Ludwig v Moonlight sonata	Beam Me Up Mus 1992

Figure 2. *Moonlight sonata*

To solve this problem, music librarians have traditionally superimposed an ordering device called a uniform title. Inserted in square brackets between the composer's name and the transcription of the title from the physical instantiation, the uniform title consists of a bibliographically significant title for the work, based on its original as given by the composer. To this are added musical identifiers (such as opus number and key), to assist with both differentiation and order in a file consisting of all of the composer's works. Excerpts are identified by movement or section title, and to all of this might be added terms that indicate variation in the sonic instantiation of the work. Taken altogether the name-uniform title citation provides the means for an alphabetico-classified ordering of a composer's works in an information retrieval venue.

In Figure 2 the last citation carries the curious publisher name "Beam Me Up Music." This citation actually identifies an arrangement of the adagio movement of *Moonlight* arranged for guitar. The uniform title for this work is as follows:

Beethoven, Ludwig van, 1770-1827.

[Sonatas, piano, no. 14, op. 27, no. 2, C# minor. Adagio sostenuto; arr.]

The purpose of this example is to demonstrate the centrality of the identity of musical works for music information retrieval. The uniform title not only identifies the present physical instantiation, but it also places it well amidst other physical instantiations, themselves representative of a variety of sonic instantiations. From the uniform title we learn the form, medium, number and key of the original work, the title of the specific movement, and the fact that this edition represents an arrangement. Seen in array, as in Figure 3, the alphabetical identifiers serve a classificatory role, arranging and displaying for differentiation the total available instantiations (physical and sonic) of the work.

Beethoven, Ludwig van, 1770-1827.

[SONATAS, PIANO, NO. 14, OP. 27, NO. 2, C# MINOR.]

[SONATAS, PIANO, NO. 14, OP. 27, NO. 2, C# MINOR; ARR.]

[SONATAS, PIANO, NO. 14, OP. 27, NO. 2, C# MINOR. ADAGIO SOSTENUTO]

[SONATAS, PIANO, NO. 14, OP. 27, NO. 2, C# MINOR. ADAGIO SOSTENUTO; ARR.]

[SONATAS, PIANO, NO. 14, OP. 27, NO. 2, C# MINOR. ALLEGRETTO]

[SONATAS, PIANO, NO. 14, OP. 27, NO. 2, C# MINOR. ALLEGRETTO; ARR.]

[SONATAS, PIANO, NO. 14, OP. 27, NO. 2, C# MINOR. PRESTO AGITATO]

[SONATAS, PIANO, NO. 14, OP. 27, NO. 2, C# MINOR. PRESTO AGITATO; ARR.]

Figure 3. Instantiations Arranged by Uniform Title

We also see in this example a simple representation of the need for a complex definition of the musical work as an entity for information retrieval. Musical works constitute complex sets of varying sonic and physical instantiations, all derived from a common progenitor. Information retrieval systems need to go well beyond the simple identification of the progenitor work. As we see demonstrated in this example, a useful information retrieval system needs to have the capability to differentiate among the varying instantiations, in order to allow searches to make the best possible choice among alternatives.

7. A TAXONOMIC DEFINITION OF THE WORK

Smiraglia (2001) suggests the parameters of a theory of the work. Smiraglia (2000) incorporated the tools of epistemology to comprehend works by incorporating those theoretical parameters in the context of a taxonomic definition, which is repeated here.

A work is a signifying, concrete set of ideational conceptions that finds realization through semantic or symbolic expression. That is, a work embraces a set of ideas that constitute both the conceptual (signified) and image (signifier) components of a sign. A work functions in society in the same manner that a sign functions in language. Works, like signs, demonstrate the characteristics of arbitrariness (the absence of a natural link between the signified and the signifier) and linearity (signifiers unfold sequentially over time). Therefore, works are subject to the natural ambiguity of signs, having both the properties of immutability (the fixed nature of a signifier in a given community) and mutability (change over time in their perception and use).

Further, a work has the characteristics of a Peircean symbol, reflecting both the physical connections of indications and the imitative ideational likenesses. Like works, Peircean symbols incorporate words or phrases that have become associated with their meanings by usage.

If a work enters a canon then its signifying texts may derive and mutate. Derivations may take one or more forms: 1) simultaneous editions; 2) successive editions; 3) amplifications; or, 4) extractions. Musical works, according to Vellucci (1997), may also derive in two additional ways through musical presentation or notational transcription. In these categories the work derives culturally over time, but ideational and semantic content do not change.

Mutations may take one or more forms as well: 1) translations; 2) adaptations; or 3) performances. In these categories the ideational and semantic content have mutated to some degree. The relations among the exemplars of a work constitute a network of related entities that has been described variously as a bibliographic family (Smiraglia 1992) or a textual identity network (Leazer and Furner 1999).

Using Hjørland's epistemological framework we can comprehend the origins of the components of this taxonomic definition. Empirically derived components are those that have been demonstrated quantitatively in the research by Smiraglia, Smiraglia and Leazer, and Vellucci. Through these studies we have quantitative evidence that works are signifying sets of ideational conceptions that take realization through semantic or symbolic expression. The characteristics of arbitrariness and linearity are clearly demonstrated by the quantification of derivations and mutations of works. Evidence of canonicity is demonstrated by the increased rate of derivation and mutation observed among works that have become part of the academic canon.

Rationalism allows us to perceive the cultural function of works, which function in society in the same manner that signs function in language. We also see through the application of rationalism that works have the characteristics of Peircean symbols, reflecting both the physical connections of indications and the imitative ideational likenesses. Pragmatism gives us the perspective that the array of instantiations of works for information retrieval must incorporate mechanisms to differentiate among the demonstrated derivations and mutations of a given work. Works, particularly musical works, that gain popularity take on the perspective of cultural icons, and from that point the rate of derivation and mutation and thus of the creation of varying physical and sonic instantiations increases. Finally, historicism provides the nominal anchor for a set of instantiations of a work. That is, the citation for the original work (such as the very useful uniform title), derived through bibliographical research, stands as the central point for linkage of instantiations in an information retrieval system.

Thus our epistemological perspective yields a logic for the construction of music information retrieval mechanisms. The nominal anchor for the accumulated artifacts or their representations is the historically-derived citation for the original ideational set, occasionally altered as a result of the natural evolutionary action over time. Rationalism provides the principles for apprehending and ordering the entire construct. Entities are derived empirically; their cultural role is described pragmatically. Derivation, mutation, and the rate thereof are empirically verifiable, pragmatic, collaborative socio-cultural constructs.

8. CONCLUSION

Musical works form a key entity for music information retrieval. Semiotic analysis suggests a variety of cultural and social roles for works, and for music in particular. Musical works, defined as entities for information retrieval, are seen to constitute sets of varying instantiations of abstract creations. Variability over time, demonstrated empirically, is an innate aspect of the set of all instantiations of a musical work, leading to complexity in the information retrieval domain.

Musical works have been well comprehended as documentary entities. Understanding the social roles of musical works expands the boundaries of their definition. Epistemological frameworks can help us understand the socio-cultural origins of concepts of the musical works. Taxonomic definition contributes to the epistemological perception of works as specific entities of recorded knowledge. An historically-generated nominal anchor for a musical work can be used to collect the entire array of instantiations.

More importantly, for music information retrieval, it is critical to comprehend the cultural role of musical works because it is at the heart of their dissemination and reception. In a digital era of music information retrieval, the question of the degree to which differing sonic instantiations represent the same work have epistemological bases. In the nineteenth century one bought a musical work by buying its score, and creating one's own sonic conception. In the twentieth century one bought a musical work by buying a recording of a performance of it--LP or CD. In both cases all copies were identical. But in the digital age, the opportunities for mutation are rampant. This must raise constantly then, the question of just what constitutes a given musical work. The answer is to be found in the epistemological understanding of the reception of musical works, and in the semiotic explanation of the role of musical works as cultural icons.

In any event, an expanded perception of musical works helps us understand the variety of ways in which mechanisms for their control and retrieval might better be shaped in future.

9. REFERENCES

Barthes, Roland. 1975. *The pleasure of the text*; trans. by Richard Miller with a note on the text by Richard Howard. New York: Noonday Press.

Eggert, Paul. 1994. Editing paintings/conserving literature: The nature of the 'work.' In *Studies in Bibliography* v.47 ed. by David L. Vander Meulen, pp. 65-78. Charlottesville, Pub. for The Bibliographical Society of the University of Virginia by The University Press of Virginia.

Dahlhaus, Carl. 1983. *Foundations of music history*, trans. J.B. Robinson. Cambridge; New York: Cambridge University Press.

Dick, Archie L.. 1999. Epistemological positions and library and information science. *Library quarterly* 69: 305-23.

Goehr, Lydia. *The Imaginary museum of musical works: an essay in the philosophy of music*. Oxford: Clarendon, 1992.

Hamman, Michael. 1999. From symbol to semiotic: Representation, signification, and the composition of music interaction. *Journal of new music research* 28: 90-104.

Hjørland, Birger. 1998. Theory and metatheory of information science: a new interpretation. *Journal of documentation* 54: 606-21.

Hjørland, Birger and Hanne Albrechtsen. 1999. An analysis of some trends in classification research. *Knowledge organization* 26: 131-9.]

Ingarden, Roman. 1986. *The work of music and the problem of its identity*, trans. Adam Czerniawski. Berkeley, Calif.: Univ. of Calif. Pr.

International Federation of Library Associations, Study Group on the Functional Requirements for Bibliographic Records. 1998. *Functional requirements for bibliographic records*. München: K.G. Saur..

Krummel, D.W. 1988. *The memory of sound: Observations on the history of music on paper*. Washington, D.C.: Library of Congress.

Krummel, D.W. 1970. "Music as Vibrations and as Flyspecks," *Wisconsin Academy of Sciences, Arts & Letters* 58.

Leazer, Gregory Hart. 1993. A conceptual plan for the description and control of bibliographic works. DLS diss., Columbia Univ.

Leazer, Gregory H. 1994. A conceptual schema for the control of bibliographic works. In *Navigating the networks: Proceedings of the ASIS mid-year meeting 1994*, ed. by D. L. Andersen, T. J. Galvin and M. D. Giguere. Medford, NJ: Learned Information, Inc., pp. 115-35.

Leazer, Gregory H. and Jonathan Furner. 1999. Topological indices of textual identity networks. *Proceedings of the 62nd annual meeting of the American Society for Information Science*, ed. Larry Woods. Medford, NJ: Information Today, pp. 345-58.

Leazer, Gregory H. and Richard P. Smiraglia. 1996. Toward the bibliographic control of works: Derivative bibliographic relationships in an online union catalog. In *Digital libraries '96: 1st ACM International Conference on Digital Libraries, March 20-23, 1996, Bethesda, Maryland*. Assn. for Computing Machinery.

Leazer, Gregory H. and Richard P. Smiraglia. 1999. Bibliographic families in the library catalog: A qualitative analysis and grounded theory. *Library resources & technical services* 43:191-212.

Ligabue, Marco. 1998. Sound and sign: some notes and reflections on the modalities of signification in music. *Contemporary music review* 17n2: 35-46.

Marco, Francisco Javier Garcia and Miguel Angel Esteban Navarro. 1993. On some contributions of the cognitive sciences and epistemology to a theory of classification. *Knowledge organization* 20: 126-32.

Nattiez, Jean-Jacques. 1990. *Music and discourse: Toward a semiology of music*, trans. by Carolyn Abbate. Princeton, N.J.: Princeton Univ. Pr.

Olson, Hope A.. (1996). Dewey thinks therefore he is: The epistemic stance of Dewey and DDC. In Green, Rebecca (Ed.), *Knowledge organization and change: proceedings of the Fourth International ISKO Conference, 15-18 July 1996, Washington, DC, USA* (pp. 302-3). Advances in knowledge organization, 5. Frankfurt/Main: Indeks Verlag.

Peirce, Charles Sanders. [1894] 1998. *The essential Peirce: selected philosophical writings*. Vol. 2 (1893-1913), ed. by the Peirce Edition Project, Nathan Houser [et al.], p. 4-10, "What is a sign?" Bloomington: Indiana Univ. Pr.

Poli, Roberto. 1996. Ontology for knowledge organization In Green, Rebecca (Ed.), *Knowledge organization and change: proceedings of the Fourth International ISKO Conference, 15-18 July 1996, Washington, DC, USA* (pp. 313-19). Advances in knowledge organization, 5. Frankfurt/Main: Indeks Verlag.

Poster, Mark. 1990. *The mode of information: Poststructuralism and social context*. Univ. of Chicago Pr.

Saussure, Ferdinand de. 1959. *Course in general linguistics*. Ed. by Charles Bally and Albert Sechehaye; in collaboration with Albert Riedlinger; trans., with an introd. and notes by Wade Baskin. New York: McGraw-Hill.

Smiraglia, Richard P. 1992. Authority control and the extent of derivative bibliographic relationships. Ph.D. diss., University of Chicago.

Smiraglia, Richard P. 2000. "Words and works; Signs, symbols and canons: The epistemology of the work." In *Dynamisn and stability in knowledge organization: Proceedings of the Sixth International ISKO Conference, 10-13 July 2000, Toronto, Canada*, ed. Clare Beghtol, Lynn C. Howarth, Nancy J. Williamson. Advances in knowledge organization v. 7. Würzburg: Ergon Verlag, pp. 295-300.

Smiraglia, Richard P. 2001. *The nature of "The work."* Lanham, Md.: Scarecrow Press.

Smiraglia, Richard P. and Gregory H. Leazer. 1995. Toward the bibliographic control of works: Derivative bibliographic relationships in the online union catalog. *OCLC annual review of research 1995* Dublin, OH: OCLC Online Computer Library Center, Inc..

Smiraglia, Richard P. and Gregory H. Leazer. 1999. Derivative bibliographic relationships: The work relationship in a global bibliographic database. *Journal of the American Society for Information Science* 50:493-504.

Talbot, Michael, ed. 2000. *The musical work: Reality or invention?* Liverpool Music Symposium 1. Liverpool: Liverpool University Press.

Thomas, David H. and Richard P. Smiraglia. 1998. Beyond the score. *Notes: The quarterly journal of the Music Library Association* 54:649-66.

Tillett, Barbara Ann Barnett. 1987. Bibliographic relationships: Toward a conceptual structure of bibliographic information used in cataloging. Ph.D. dissertation, University of California, Los Angeles.

Turino, Thomas. 1999. Signs of imagination, identity, and experience: A Peircian semiotic theory for music. *Ethnomusicology* 43: 221-55.

van Leeuwen, Theo. 1998. Music and ideology: Notes toward a sociosemiotics of mass media music. *Popular music and society* 22n4: 25-54.

Vellucci, Sherry L. 1997. *Bibliographic relationships in music catalogs*. Lanham, Md.: Scarecrow Press.

The *JRing* System for Computer-Assisted Musicological Analysis

Andreas Kornstädt
Arbeitsbereich Softwaretechnik (SWT), Fachbereich Informatik, Universität Hamburg
Vogt-Kölln-Straße 30, 22527 Hamburg, Germany
++49 40 69424-200
kornstae@informatik.uni-hamburg.de

ABSTRACT

Among other factors, high complexity and mandatory expert computer knowledge make many music IR and music analysis systems unsuitable for the majority of largely computer-illiterate musicologists. The *JRing* system offers highly flexible yet intuitively usable search and comparison operations. to aid musicologists during score analysis. This paper discusses the requirement analysis that led to *JRing's* inception, its IR tools and graphical user interface plus the kind of musical material it works on and the *Humdrum*-based technical realization of IR operations.

1 USER NEEDS

JRing was conceived as set of tools to assist musicologists during that kind of score analysis which aims at:

- a score in which all musicologically relevant elements are completely marked up
- a catalogue which contains all occurrences of all of these elements in complete form plus an arbitrary set of annotations

Depending on the type of work and / or analysis, the elements could be themes, leitmotivs or sets. At the beginning of the *JRing* development process, musicologists at the University of Hamburg and Stanford University were asked to specify the kind of computer assistance they would like to have during analysis. The five results that directly or indirectly pertain to IR were:

(1) Print-like rendition of all musical materials (score, excerpts, search results) as the basis for identifying and comparing elements optically.

(2) Search capabilities for finding elements by arbitrary combinations of musical features (pitch, harmony, and rhythm in various forms).

(3) Tools that help to create catalogue entries, comprising (a) making excepts and (b) filling in information about the elements that can be automatically derived from the excerpt such as set class or position within the score.

(4) Catalogue management capabilities to sort and filter catalogue entries according to certain criteria.

(5) Customization of the structure of catalogue entries and consequently the search and comparison operations based on them.

Other – non IR-related – requirements included the ability to switch back and forth between different kinds of analyses plus a maximum degree of platform independence.

It becomes evident from the composition of the set of requirements what at least the musicologists that took part in the development of *JRing* do aim for. It is not a “big automaton” that can be fed with a score and some kind of theory description and that churns out a results that has to be interpreted by the musicologist [1, 7]. Instead, what is asked for is a set of tools that leaves the analyst permanently in charge of analysis decisions and that merely assist him in making choices faster and with less effort. The basic ways of traditional, manual analyses should not be changed.

2 SOLUTION COMPONENTS

A comprehensive solution that meets all the above-mentioned requirements can hardly be furnished single-handedly. Although there is no adequate reusable and platform independent graphical user interface, many results in the area of data storage and retrieval can be incorporated and made accessible through a new user interface.

2.1 Data

The foremost problem is a lack of data to be analyzed that is available in an appropriate format. Although MIDI and score notation data is widely available, these formats are ill-suited for analysis and musical IR.

- MIDI data focuses on pitch while durations are often not quantized. As MIDI is mainly intended for controlling electronic instruments, enharmonic spelling, articulation, ornamentation, and lyrics cannot be represented among other things. Although there are several extensions of the MIDI file format that try to overcome some of these limitations, none has captured a sizable market share and is thus a good source for widely usable analytical data [9].
- Data from notation programs such as SCORE, Finale or NP-DARMS tends to be layout-oriented and often subordinates logical musical information (C##) to purely spatial descriptions (“black circle between two horizontal lines”). The true pitch can only be determined by scanning backwards for a clef, key signatures, ottava marks, etc. Although, these formats are mostly page-oriented so that musical features that cross page boundaries are very difficult to recognize. Therefore, analytic applications that work on this kind of data often restrict themselves to dealing with works that fit on a single page [8]. Also, they need to implement complex algorithms to extract the logical musical information from the spatial information.

In contrast to highly specialized MIDI and notation formats, analytical formats like *Humdrum* [4] and *MuseData* [3] are better suited for analytic applications. Both have been specifically designed with music analysis and IR in mind. They do not exclusively focus on one single aspect of the score (audible pitch or visual rendition) but offer flexible, extensible encoding schemes that can be used to represent arbitrary musical abstractions in different (parts of) files. Common abstractions besides pitch and duration are melodic contour, rhythmic weight, scale degree, pitch class, frequency, MIDI events or lyrics (full text, syllables and phonetic equivalents). All in all, over predefined 40 *Humdrum* formats exist. As *Humdrum* makes only minimal structural requirements, new formats can be added to encode almost any kind of new musical abstraction that musicologist can conceive.

The separation of different musical aspects within the data representation forms the basis for that kind of search and comparison features that users require according to the results of section 1. Therefore, developers of music analysis and IR programs can make use of these existing analytical formats instead of coming up with completely new encoding schemes.

Because MuseData has been less widely publicized than *Humdrum* and does not split different musical aspects into different files, *Humdrum* is becoming the main target format for conversion programs. If copyright problems can be solved, the vast majority of high quality scores can be converted into *Humdrum* using *FinalSCORE* (by Leland Smith), *scr2hmd* [6] (by this author) and *muse2kern* (by Bret Aarden and this author).

2.2 Information Retrieval

As on the data side, *Humdrum* offers an ideal platform for information retrieval programs. It comes with over 40 specialized UNIX-programs and tools, many of which can work with all file formats. As all formats have the same structure, they can be manipulated and analyzed with a few common tools that – among other things – can assemble individual specialized files into one combined file or extract certain section either by data type (e.g. pitch) or by position (e.g. measures 60 to 70). Analytic questions that pertain to a certain representation can be answered by running chains ("pipes" in UNIX-lingo) of small programs, each of which performs a very limited task. Plugged together in a useful way, these pipes can find answers to quite complex questions such as "Do melodic passages in folk songs tend to exhibit an arch shape?". Because *Humdrum* (1) runs in the UNIX environment, (2) stores its data in ASCII format, and (3) provides a wide range of reference records for encoding non-musical information, the standard UNIX tools for data management (**`find`**, **`grep`**, **`sort`**, **`sed`**, etc.) can be used.

For example, in order to mark instances of a certain pattern in a score by its semitone contour, the following UNIX pipe of commands is necessary:

```
extract –i'**kern' score.krn | semits -x |
xdelta –s = | patt –t MotivX –s = -f MotivX.dat |
extract –i'**patt' | assemble score.krn
```

The complexity of the patterns to be matched depends on the program used:

(1) When the **`patt`** command is used, search patterns are limited to quite simple sequences of tokens. To search for a melodic contour of "same, up 2, down 2", that pattern looks like this:

```
0
+2
-2
```

(2) The **`pattern`** command allows for highly complex search patterns. The following sequence of tokens matches one or more G naturals followed optionally by a single G-sharp followed by one or more records containing one or more pitches from an A major triad the last of which must end a phrase.

```
[Gg]+[^#-] +
[Gg]+#[^#-] ?
([Aa]+|([Cc]+#)|[Ee]+)[^#-] *
(}.*([Aa]+|([Cc]+#)|[Ee]+)[^#-]))|(
([Aa]+|([Cc]+#)|[Ee]+)[^#-].*})
```

(3) A different level of flexibility can be achieved by using the **`simil`** command. It does not only match precise instances of a given pattern but measures the editing distance between the given pattern and the material in the score [5]. The result is a list of numbers between 1 (perfect match) and 0 (no similarity at all) that quantifies the degree of similarity for every position in the score.

Humdrum data and tools form the ideal technical infrastructure for analytic and IR applications. They permit a wide range of analytical and IR operations while being open for custom-made extensions. Especially, *Humdrum* is suitable to realize all those search and comparison features described in the requirements section. The basic modus operandi can be described as follows:

1. Any element (theme / leitmotiv / set) as well as the score is converted into those specific *Humdrum* formats that can be used for conducting searches and comparisons. E.g. semitone intervals, pitch classes, durations and lyrics in syllables.

2. According to the specifications of the user, one or more of these representations are separately processed with the appropriate program (**`patt`**, **`pattern`**, **`simil`**). E.g. the semitone representation is searched for the sequence "same, up 2, down 2" while the duration representation is searched for "`Hap-py birth-day`"

3. The results are merged to form the combined result. For example, only those positions in the score are returned that feature the semitone pattern AND the lyrics.

Despite its benefits, for the vast majority of musicologists, *Humdrum* can only serve as technical infrastructure and not as IR or analytic system itself because it does not provide a graphical user interface that allows analysts to manipulate the score, its elements and catalogues in a way that they are accustomed to. Although there are sometimes GUIs that help constructing the command pipes [2, 10], working with musical materials the traditional way is still not possible.

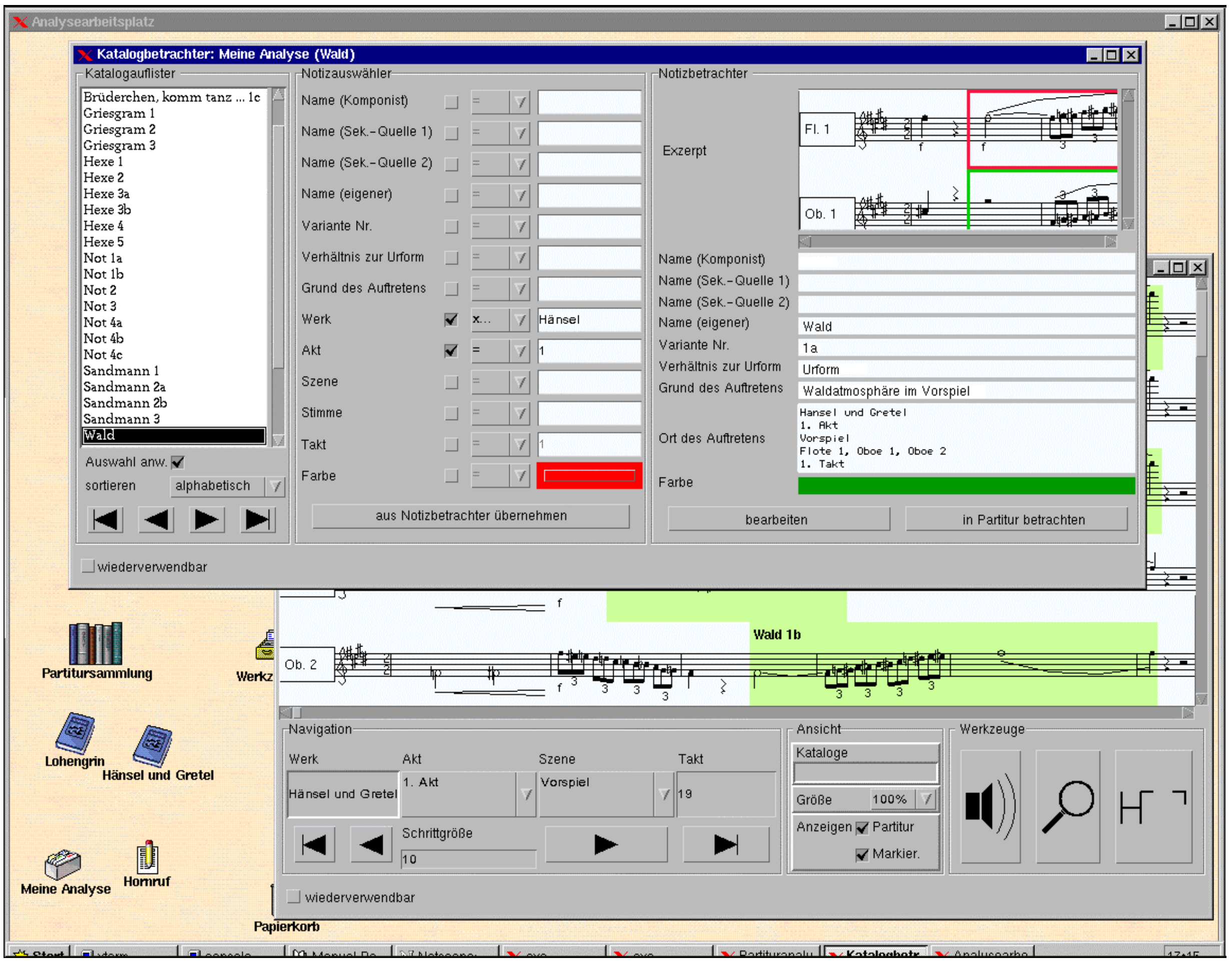
3 *JRING*

JRing aims at providing musicologists with an electronic equivalent of their physical desktop complete with fully graphical scores, thematic / leitmotivic / set notes and catalogues of these elements. Scores, notes and catalogues can be perused in the same way as their physical equivalents, i.e. always with a view of the print-like view of the material. It also offers IR-related functionality that goes beyond what can be done with physical scores and catalogues.

In order to better understand the way in which IR functionality is embedded into *JRing*, its non-IR related features are described first.

3.1 Non-IR-Related Features

JRing works on scores, notes and catalogues.


Analysearbeitsplatz

Katalogbetrachter: Meine Analyse (Wald)

Katalogauflister

Brüderchen, komm tanz ... 1c
Griesgram 1
Griesgram 2
Griesgram 3
Hexe 1
Hexe 2
Hexe 3a
Hexe 3b
Hexe 4
Hexe 5
Not 1a
Not 1b
Not 2
Not 3
Not 4a
Not 4b
Not 4c
Sandmann 1
Sandmann 2a
Sandmann 2b
Sandmann 3
Wald

Auswahl anw. ✓

sortieren: alphabetisch

Notizauswähler

Name (Komponist)		=	
Name (Sek.-Quelle 1)		=	
Name (Sek.-Quelle 2)		=	
Name (eigener)		=	
Variante Nr.		=	
Verhältnis zur Urform		=	
Grund des Auftretens		=	
Werk	✓	x...	Hänsel
Akt	✓	=	1
Szene		=	
Stimme		=	
Takt		=	1
Farbe		=	🟥

aus Notizbetrachter übernehmen

Notizbetrachter

Exzerpt

Name (Komponist)	
Name (Sek.-Quelle 1)	
Name (Sek.-Quelle 2)	
Name (eigener)	Wald
Variante Nr.	1a
Verhältnis zur Urform	Urform
Grund des Auftretens	Waldatmosphäre im Vorspiel
Ort des Auftretens	Hansel und Gretel 1. Akt Vorspiel Flote 1, Oboe 1, Oboe 2 1. Takt
Farbe	🟩

bearbeiten | in Partitur betrachten

wiederverwendbar

Navigation

Werk	Akt	Szene	Takt
Hänsel und Gretel	1. Akt	Vorspiel	19

Schrittgröße: 10

wiederverwendbar

Ansicht

Kataloge

Größe: 100%

Anzeigen: ✓ Partitur, ✓ Markier.

Werkzeuge

Figure 1. The main components of the *JRing* system: desktop, score analyzer (partly covered), and catalogue browser.

3.1.1 Scores

Scores are displayed in a high quality, print-like rendition. Voices are basically arranged the same way as in the printed score but vertical positions are fixed for every voice. Therefore, the topmost row is always reserved for e.g. the piccolo while the lowest voice always holds the display for e.g. the double base. Therefore, the vertical size of the score is always the same even if some voices are pausing. This grid-like organization of the score makes it easy (1) to browse the score without the need to search for a specific instrument and (2) to mark an occurrence of a specific element in one piece even if it covers one or more system / page breaks in the printed score.

The tool that works on scores is the score analyzer. It displays the score in the above-mentioned way. It has several features:

- The position within the score can be changed by either making use of the scroll bars or by jumping to any logical location such as measure 23 of the 2nd scene of the 3rd act.
- Already marked up elements can be shown or hidden. The ability to see intermediate results gives valuable information on where to look for new elements. In combination with zooming out to, say, a 10% magnification this feature to get an overview over large-scale patterns of elements.
- The score can be searched for arbitrary combinations of musical features (see section 3.2.2).
- Newly found elements can be graphically marked up with a marker tool (see figure 2). Although marks can consist of a single contiguous block, they can be made up of any number of blocks. Marks form the basis for notes.

3.1.2 Notes

Notes describe one occurrence of a musically relevant element of a work. They consist of a graphical rendition of the marked element (see above) plus an arbitrary number of additional fields. As notes used in manual analysis, they contain fields for non- or meta-musical information such as the position within the score, the formal relation of the element to other elements, the reason for

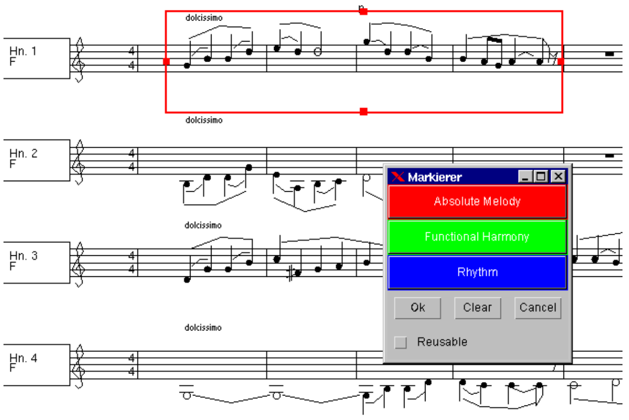


Figure 2. The marker tool.

the element's occurrence or – for leitmotivic analysis – the name of the leitmotiv. Note fields form the basis for comparisons with other elements which are penned down on other notes.

Two tools directly work on notes:

- Note editors pop up after an element has been marked up using the marker tool. Its graphical excerpt and its position within the score are automatically filled in. Information that depends on human judgement (formal relation, reason for occurrence, etc.) needs to be filled in manually.
- Note viewers (see rightmost sub tool in catalogue browser in figure 1) display notes.

A third kind of note-related tool are note selectors. They do not work on individual notes but on catalogues.

3.1.3 Catalogues

Catalogues contain all the notes an analyst furnishes. As a musicologist can find a huge number of elements in a sizable score, tool support for managing a catalogue is need. This support is provided by catalogue browsers.

Catalogue browsers consist of three sub tools: A catalogue lister, a note selector and a note viewer. Basically, the user can select a note in the note lister which is then shown by the note viewer. The note selector can be used to make the lister show only a specific subset of notes from the catalogue.

Note selectors (see central sub tool in catalogue browser in figure 1) are structurally similar to note viewers and note editors: Like these, they list every field of the note and like note editors every filed is editable. Different from these, the values entered into the fields of a note selector do not describe a specific note but a pattern that is matched against all notes contained in a catalogue. If for example "1" is entered into the field named "Act", only notes that pertain to elements from the first act are listed in the note lister.

Although valuable, matching a single criterion does not meet the user requirements stated in section 1 (cf.). Therefore, note selectors offer two additional features for describing note selections more precisely:

1. A combo box to select the matching condition. For textual fields the options are "equals", "contains", "starts with" and "ends with". For numerical fields these are "equals", "less than", "less than or equals", etc.
2. A checkbox which users can employ to indicate whether a field should be taken into account when finding matches or not. In figure 1 this feature is used to show only those notes in the lister that matches certain work names AND occur in the first act.

3.2 IR-Related Features

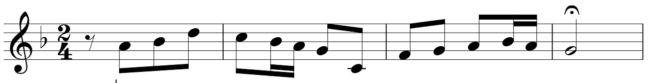
The features described so far replicate the traditional way of dealing with scores, notes and catalogues. Although they facilitate the analytic process considerably by relieving musicologists of strenuous tasks by means of automatically doing excerpts, filling out large parts of notes, and dealing with catalogues but they do not provide any IR capabilities.

JRing provides these capabilities not through completely new tools but integrates them into the tools already described in section 3.1.

3.2.1 Catalogue browser

In manual analysis, the musicologist just needs to write down the excerpt and can then make any kind of comparison based solely on that excerpt. Because he can make arbitrary abstractions of that excerpt, he can simply choose to concentrate on certain abstract features and then browse the other excerpts in the catalogue to find exact, partial or fuzzy matches. As a computer IR system cannot know in which respect two notes should be compared, this choice of the analyst has to made explicit in a computer assisted system. Therefore, in addition to the text fields discussed in section 3.1.2, each note in *JRing* carries a list of musical abstractions that are automatically derived from the excerpt. For example, the pitch information of the marked up element can be used to derive absolute pitch, pitch class, and any melodic contour in semitone steps (see table 1).

Table 1. Some melodic abstractions of a Beethoven theme



pitch	a1	b1-	d2	c2	b1-	a1	g1	c1	f1	g1	a1	b1-	a1	g1
pitch class	9	A	2	0	A	9	7	0	5	7	9	A	9	7
semitone interval	*	+1	+4	-2	-2	-1	-2	-7	+5	+2	+2	+1	-1	-2
refined contour	*	^	/	v	v	v	v	\	/	^	^	^	v	v
gross contour	*	/	/	\	\	\	\	\	/	/	/	/	\	\

In the note selector, these fields with musical abstractions can be used in the same way as text fields in order to determine the notes that are shown in the note lister: The checkbox next to them indicates whether or not a certain field is to be included in the comparison, and the combo box can be used to determine the matching condition (from a 100% perfect match down to 5% similarity).

To make filling in the fields of the note selector easier, the contents of the note displayed in the note viewer can be copied into the note selector and then modified.

3.2.2 Score Searcher

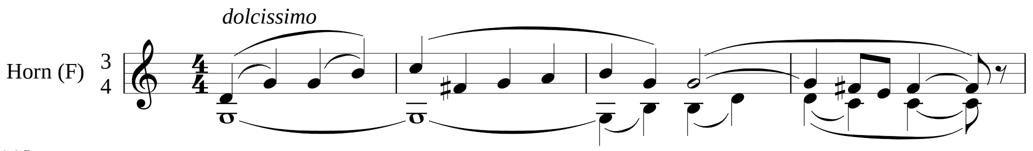
The score searcher is a sub tool of the score analyzer described in section 3.1.1. It basically has the same layout as the note selector, but has only fields for musical abstractions and no text fields except for specifying a search range within the score. When the user chooses to search the score for a certain combination of features, the results are temporarily marked up in the score and a result list browser pops up. Clicking on a result entry in the lister takes the user to the score position of the match. He can decide to discard the result list or can invoke the score marker to make individual results the basis for new notes in the catalogue.

JRing's tools are adequate to fulfill user requirements (1) to (4) stated in section 1. The remaining fifth requirement - customization of the structure of catalogue entries – is the topic of section 4.2.

4 TECHNICAL REALIZATION

4.1 IR-Related Features

As motivated in section 2, *JRing* uses *Humdrum* as its technical infrastructure and does not implement music IR functionality itself. While presenting *Humdrum* data in a way that is tailored to



```
**kern       **layout
*Icor        *SCORE
*Itrd4c7     *
=1           =1
*clefG2      *
*M4/4        *
4G           1 14 13.3602 2 10 0 1|16 14 13.6785 17 1 1 0 0 0 0 0 13 13.2094 dolcissimo|5 14 14.4385 10.5 ...
4c           1 14 19.5083 5 10 0 1
4c           1 14 25.654 5 10 0 1|5 14 27.0518 8 10 32.0118 1.348 -1
4e           1 14 31.7996 7 10 0 1|14 14 37.9542 1
=2           =2
...          ...
```

the needs of the majority of musicologists, *JRing* can be seen as an – albeit very extensive – GUI front-end for *Humdrum*. It transforms user input into *Humdrum* commands and parses the results in such a way that it can be displayed in a form that is understandable for musicologists that are used to carrying out analyses in a traditional form.

The core of the transformation process are the structurally identical files the contain the analytical score information and the graphical layout information. Because they are synchronized, user input that pertains to the graphic representation on the screen can be mapped to a precise portion of analytical information and search results can be mapped back to the graphical score rendition (see figure 3). A search in the score is thus carried out the following way:

1. For every field that the user marked by ticking the checkbox next to it, its contents are interpreted and converted into the appropriate *Humdrum* representation.
2. If the user has not included one of the selected representations, the required abstraction file is generated with *Humdrum*. E.g. `**semits` for semitone intervals.
3. For every representation, a separate *Humdrum* pattern matching tool is invoked. If perfect matches are desired, **`patt`** is used, **`simil`** otherwise.
4. The individual results are merged. Only those hits become part of the comprehensive result that occur in *every* individual result.
5. *JRing* compiles the result list from the comprehensive result. By going from analytical spines to synchronized layout spines, the its precise position within the graphically rendered score can be determined.

Because melodic matching often requires finding roving or hidden themes [9], *JRing* generates an extra file that contains the highest consonant pitches of the whole score. This “melody” file is automatically included into any melodic search.

The technical realization of note comparisons using the note selector is similar to using the score searcher. The only difference is that notes are matched against each other and that results are listed in the note lister instead and not in a separate result lister.

4.2 Customization

Although the list of musical abstractions discussed so far does cover some of the more frequently used features, it is not exhaustive by far. Taking into account that *Humdrum* comes with over 40 data formats and that new formats can easily be added, no tool that comes with a fixed and thus limited set of abstractions / note elements can be very useful. If all possible abstractions are automatically generated, notes become huge and unwieldy. On the other hand, if the “right” abstractions are not offered, the tool is in risk of becoming useless.

Similar cases can be made for the capability to display scores in different formats (mensural, CMN, chromatic) or to work with different *Humdrum* implementations or without any *Humdrum*-subsystem.

To deal with the demand for flexibility, *JRing* can be customized in three ways:

(1) **The structure of notes.** Depending on the type of analysis, notes can be made up of different fields. In a leitmotivic analysis, there might be several name fields for leitmotiv names according to different sources, and musical abstractions from the melodic, harmonic and rhythmic domain. For an analysis based on Forte’s set theory, the name of the element can be automatically derived from the excerpt (by determining the prime form and its set class) while there is no need for melodic, harmonic and rhythmic abstractions.

(2) **The type of score rendition.** To represent a score on screen, there needs to be a component that takes a graphical score presentation and displays it. Depending on the type of score notation (mensural, CMN, chromatic, or something completely new), different display components can be selected.

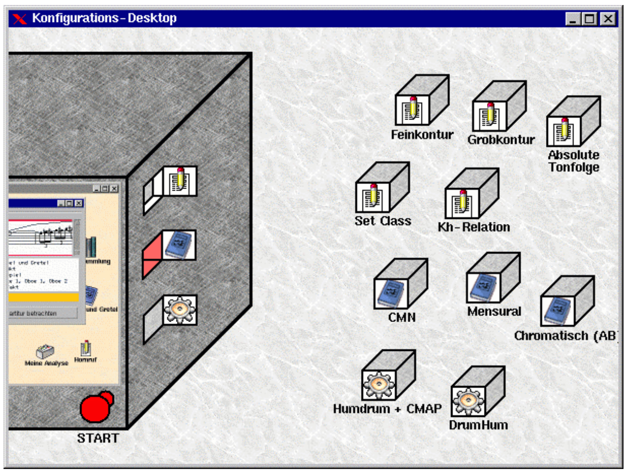
(3) **Type of musical subsystem.** As *Humdrum* consists of UNIX commands, it requires a UNIX shell for operation. This might not be available on non-UNIX systems because the UNIX shell emulators available on these platforms cannot be used free of charge. Therefore, the component that connects to the musical subsystem can be totally absent in some cases or a substitute might be available. If totally absent, *JRing* functions normally except that searches and comparisons based on musical features are disabled.

The source of *JRing’s* flexibility is the slide-in approach. Components implementing (1) a single musical abstraction, (2) score renderer, or (3) musical subsystem can be put into matching “holes” of the *JRing* core system at startup time. In contrast to plug-ins or other kinds of dynamic libraries, slid-ins have the following advantages:

1. Slide-ins implement exactly one flexible feature of the system whose functionality actually means something to its users.
2. Slide-ins only fit into the matching slide-in frame (“hole”) of the core system, thus making misconfigurations impossible.

3. Individual slide-in frames have specific capacities. While the slide-in frame for musical abstractions can hold any number of slide-ins, the slide-in frame for musical subsystems can have at most one slide-in (either *Humdrum*, a substitute or nothing at all), and the slide-in frame for score renders must have exactly one slide-in.

Because slide-ins are easy to visualize, a configuration desktop is provided that can even be used by those musicologists that would not normally be able to configure technical systems.



As does *Humdrum*, *JRing* just offers a core system that can be easily extended in compliance with its interfaces. In the case of *Humdrum*, these interfaces are the *Humdrum* file format structure and the POSIX standard. In the case of *JRing*, these interfaces are the three slide-in frames (for (1) new abstractions, (2) score renderers, and (2) musical subsystems) plus the Java programming language. As with *Humdrum*, users of *JRing* can profit from its high reuse potential, i.e. when starting a new project, chances are that the required slide-ins already have been written by some one else. If not, just the missing slide-ins need to be implemented and can later be passed on into the pool of available slide-ins so that they can be used in future projects by others.

5 DISCUSSION

JRing meets all requirements listed in section 1 by providing a graphical user interface that can easily be used by the vast majority of musicologists. While this GUI is merely a *Humdrum* GUI from a technical point of view, it adds the user-oriented features of notes and catalogues that are not present in the *Humdrum* world. The system allows for complex searches and comparisons by combining arbitrary musical abstractions in precise or fuzzy searches using combinations of *Humdrum*'s **`patt`** and **`simil`** commands. Due to its compliance with the slide-in approach, it can easily be extended in a fashion similar to *Humdrum*.

The limitation of *JRing* stem from the way in which searches and comparisons can be formulated. While *Humdrum* allows extremely complex pattern definitions (see section 2.2), *JRing* pares down searches to fixed length patterns that can merely be combined with Boolean conjunctions (ANDs). Although patterns can be matched using similarity, not even Boolean subjunctions (ORs) are possible.

Still, this limitation appears to be acceptable as most musicological queries done by traditional musicologists do not require the maximum degree of flexibility offered by *Humdrum*. To make this limitation acceptable to more demanding musicologists, *JRing* maintains all notes as separate *Humdrum* files (with text information as reference records). These files can be used independently from *JRing* in *Humdrum* pipes to make full use of *Humdrum*'s powerful yet difficult pattern matching syntax.

6 REFERENCES

[1] Bo Alphonce, The Invariance Matrix, Dissertation, New Haven, CT, Yale University, 1974.

[2] Peter Castine, Set Theory Objects: Abstractions for Computer-Aided Analysis and Composition of Serial and Atonal Music, Frankfurt: Peter Lang, 1994.

[3] Walter Hewlett, „*MuseData*: Multipurpose Representation“, Eleanor Selfridge-Field (ed.), Beyond MIDI: The Handbook of Musical Codes, Cambridge, MA: The MIT Press, 1997, pp. 402-447.

[4] David Huron, The Humdrum Toolkit Reference Manual, Menlo Park, CA, CCARH, 1994.

[5] David Huron, Keith Orpen, „Measurement of Similarity in Music: A Quantitative Approach for Non-parametric Re presentations“, Computers in Music Research, Vol. 4, 1992, pp. 1-44.

[6] Andreas Kornstädt, „SCORE-to-Humdrum: A Graphical Environment for Musicological Analysis“, Computing in Musicology, Vol. 10, 1996, pp. 105-122.

[7] Guerino Mazzola, Thomas Noll, and Oliver Zahorka, „The RUBATO Platform“, Computing in Musicology, Vol. 10, 1996, pp. 143-149.

[8] Nigel Nettheim, „Melodic Pattern-Detection Using MuSearch in Schubert's Die schöne Müllerin“, Computing in Musicology, Vol. 11, 1998, pp. 159-168.

[9] Eleanor Selfridge-Field, „Introduction“, Beyond MIDI: The Handbook of Musical Codes, Cambridge, MA, The MIT Press, 1997, pp. 3-38

[10] Michael Taylor, Humdrum Graphical User Interface, MA Thesis, Belfast, Queen's University, 1996.

Automated Rhythm Transcription

Christopher Raphael[*]
Department of Mathematics and Statistics
University of Massachusetts, Amherst
raphael@math.umass.edu

May 21, 2001

Abstract

We present a technique that, given a sequence of musical note onset times, performs simultaneous identification of the notated rhythm and the variable tempo associated with the times. Our formulation is probabilistic: We develop a stochastic model for the interconnected evolution of a rhythm process, a tempo process, and an observable process. This model allows the globally optimal identification of the most likely rhythm and tempo sequence, given the observed onset times. We demonstrate applications to a sequence of times derived from a sampled audio file and to MIDI data.

1 Introduction

A central challenge of music IR is the generation of music databases in formats suitable for automated search and analysis [1], [2], [3], [4], [5], [6]. While a certain amount of information can always be compiled by hand, the thought of "typing in," for example, the complete works of Mozart seems daunting, to say the least. Given the enormity of such tasks we expect that automatic music transcription will play an important role in the construction of music databases.

We address here a component of this automatic transcription task: Given a sequence of times, we wish to identify the corresponding musical rhythm. We refer to this problem as "Rhythmic Parsing." The sequences of times that form the input to our system could come from a MIDI file or be estimated from (sampled) audio data. On output, the rhythmic parse assigns a score position, a (measure number, measure position) pair, to each time.

A trained musician's rhythmic understanding results from simultaneous identification of rhythm, tempo, pitch, voicing, instrumentation, dynamics, and other aspects of music. The advantage of posing the music recognition problem as one of simultaneous estimation is that each aspect of the music can inform the recognition of any other. For instance, the estimation of rhythm is greatly enhanced by dynamic information since, for example, strong beats are often points of dynamic emphasis. While we acknowledge that in restricting our attention to timing information we exclude many useful clues, we feel that the basic approach we present is extendible to more complex inputs.

We are aware of several applications of rhythmic parsing. Virtually every commercial score-writing program now offers the option of creating scores by directly entering MIDI data from a keyboard. Such programs must infer the rhythmic content from the time-tagged data and, hence, must address the rhythmic parsing problem. When the input data is played with anything less than mechanical precision, the transcription degrades rapidly, due to the difficulty in computing the correct rhythmic parse.

[*]This work is supported by NSF grant IIS-9987898.

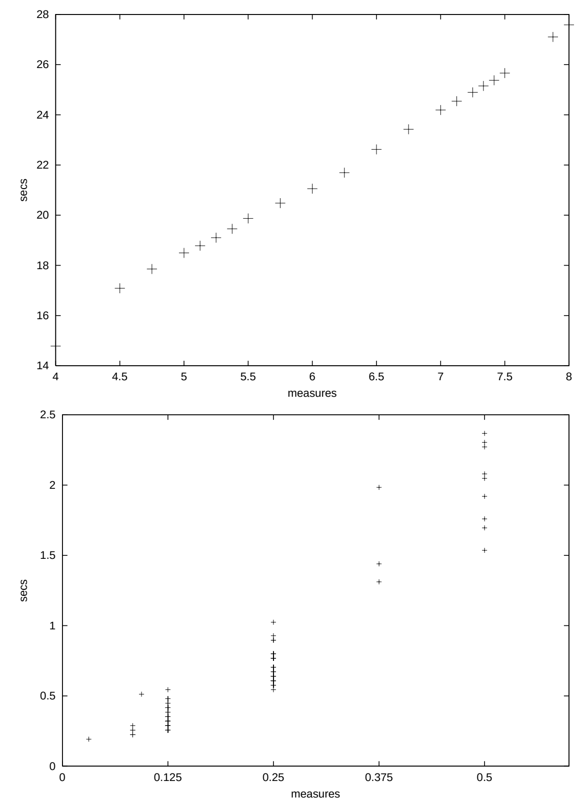


Figure 1: **Top:** Real time (seconds) vs. Musical time (measures) for a musical excerpt. **Bottom:** The actual inter onset intervals (seconds) of notes grouped by the musical duration (measures).

Rhythmic parsing also has applications in musicology where it could be used to separate the inherently intertwined quantities of notated rhythm and expressive timing [7], [8], [9]. Either the rhythmic data or the timing information could be the focal point of further study. Finally, the musical world eagerly awaits the compilation of music databases containing virtually every style and genre of (public domain) music. The construction of such databases will likely involve several transcription efforts including optical music recognition, musical audio signal recognition, and MIDI transcription. Rhythmic parsing is an essential ingredient to the latter two efforts.

Consider the data in the top panel of Figure 1 containing estimated note times from an excerpt of Schumann's 2nd Romance for oboe and piano (oboe part only). The actual audio file can be heard at http://fafner.math.umass.edu/rhythmic_parsing. In this figure we have plotted the score position of each note, in measures, versus the actual onset time, in seconds. The points trace out a curve in which the player's tempo can be seen as the slope of the curve. The example illustrates a very common situation in music: The tempo is not a single fixed number, but rather a time-varying quantity. Clearly such time-varying tempi confound the parsing problem leading to a "chicken and egg" problem: To estimate the rhythm, one needs to know the tempo process and vice-versa.

Most commercially available programs accomplish the rhythmic parsing task by *quantizing* the observed note lengths, or more precisely inter-onset intervals (IOIs), to their closest note values (eighth note, quarter note, etc.), given a known tempo, or quantizing the observed note onset times to the closest points in a rigid grid [10]. While such quantization schemes can work reasonably well when the music is played with robotic precision (often a metronome is used), they perform poorly when faced with the more expressive and less accurate playing typically encountered. Consider the bottom panel of Figure 1 in which we have plotted the written note lengths in measures versus the actual note lengths (IOIs) in seconds from our musical excerpt. The large degree of overlap between the empirical distributions of each note length class demonstrates the futility of assigning note lengths through note-by-note quantization in this example.

We are aware of several research efforts in this direction. Some of this research addresses the problem of *beat induction*, or *tempo tracking* in which one tries to estimate a sequence of times corresponding to evenly spaced musical intervals (e.g. beats) for a given sequence of observed note onset times [11], [12]. The main issue here is trying to follow the tempo rather than transcribing the rhythm. Another direction addresses the problem of rhythmic transcription by assigning simple integer ratios to observed note lengths without any corresponding estimation of tempo [13], [14], [15]. The latter two of these approaches assume that beat induction has already been performed, whereas the former assumes that tempo variations are not significant enough to obscure the ratios of neighboring note lengths.

In many kinds of music we believe it will be exceedingly difficult to *independently* estimate tempo and rhythm, as in the cited research, since the observed data is formed from a complex interplay between the two, as illustrated by the example of Figure 1. Thus, in this work we address the problem of *simultaneous* estimation of tempo and rhythm; in the following we refer to such a simultaneous estimate as a *rhythmic parse*. From a problem domain point of view, our focus on simultaneous estimation is the most significant contrast between our work and other efforts.

2 The Model

We construct a generative model that describes the simultaneous evolution of three processes: a rhythm process, a tempo process, and an observable process. The rhythm process takes on values in a finite set of possible measure positions whereas the tempo process is *continuous-valued.* In our model, these two interconnected processes are not directly observable. What we observe is the sequence of inter-onset intervals (IOIs) which depend on both unobservable quantities.

To be more specific, suppose we are given a sequence of times $o_0, o_1, \ldots, o_N$, in seconds, at which note onsets occur. These times could be estimated from audio data, as in the example in Figure 1, or could be times associated with MIDI "note-ons." Suppose we also have a finite set, $\mathcal{S}$, composed of the possible *measure positions* a note can occupy. For instance, if the music is in 6/8 time and we believe that no subdivision occurs beyond the eighth note, then

$$\mathcal{S} = \{\frac{0}{6}, \frac{1}{6}, \frac{2}{6}, \frac{3}{6}, \frac{4}{6}, \frac{5}{6}\}$$

More complicated subdivision rules could lead to sets, $\mathcal{S}$, which are not evenly spaced multiples of some common denominator, as shown in the experiments of Section 4. We assume only that the possible onset positions of $\mathcal{S}$ are rational numbers in $[0,1)$, decided upon in advance. Our goal, in part, is to associate each note onset o_n with a score position — a pair consisting of a measure number and an element of $\mathcal{S}$. For the sake of simplicity, assume that no two of the $\{o_n\}$ can be associated with the exact same score position as would be the case for data from a single monophonic instrument. We will drop this assumption in the second example we treat.

We model this situation as follows. Let $S_0, S_1, \ldots, S_N$ be the discrete measure position process, $S_n \in \mathcal{S}, n = 0, \ldots, N$. In interpreting these positions we assume that each consecutive pair of positions corresponds to a note length of at most one measure. For instance, in the 6/8 example given above $S_n = 0/6, S_{n+1} = 1/6$ would mean the nth note begins at the start of the measure and lasts for one eighth note, while $S_n = 1/6, S_{n+1} = 0/6$ would mean the nth note begins at the second eighth note of the measure and lasts until the "downbeat" of the next measure. We can then use $l(s, s')$,

$$l(s,s') = \begin{cases} s' - s & \text{if } s' > s \\ 1 + s' - s & \text{otherwise} \end{cases} \tag{1}$$

to unambiguously represent the length, in measures, of the transition from s to s'. Note that we can recover the actual score positions from the measure position process. That is, if $S_0 = s_0, S_1 = s_1, \ldots, S_N = s_N$ then score position, in measures, of the nth note is $m_n = s_0 + l(s_0, s_1) + \ldots, l(s_{n-1}, s_n)$. Extending this model to allow for notes longer than a measure complicates our notation slightly, but requires no change of our basic approach. We model the S process as a time-homogeneous Markov chain with initial distribution $p(s_0)$ and transition probability matrix

$$R(s_{n-1}, s_n) = p(s_n | s_{n-1})$$

With a suitable choice of the matrix R, the Markov model captures important information for rhythmic parsing. For instance, R could be chosen to express the notion that, in 4/4 time, the last sixteenth note of the measure will very likely be followed by the downbeat of the next measure: $R(15/16, 0/16) \approx 1$. In practice, R should be learned from actual rhythm data. When R accurately reflects the nature of the data being parsed, it serves the role of a musical expert that guides the recognition toward musically plausible interpretations.

The tempo is the most important link between the printed note lengths, $l(S_n, S_{n+1})$, and the observed note lengths, $o_{n+1} - o_n$. Let $T_1, T_2, \ldots, T_N$ be the continuously-valued tempo process, measured in *seconds per measure*, which we model as follows. We let the initial tempo be modeled by

$$T_1 \sim N(\nu, \phi^2)$$

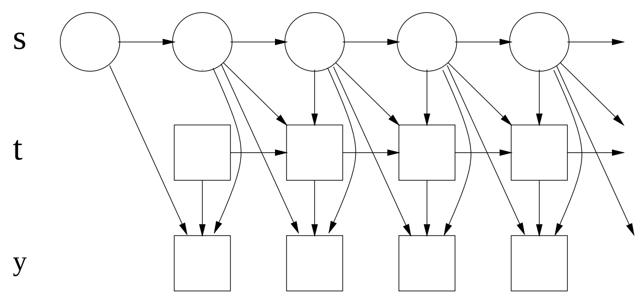


Figure 2: The DAG describing the dependency structure of the variables of our model. Circles represent discrete variables while squares represent continuous variables.

where $N(\nu, \phi^2)$ represents the normal distribution with mean ν and variance ϕ^2. With appropriate choice of ν and ϕ^2 we express both what we "expect" the starting tempo to be (ν) and how confident we are in this expectation ($1/\phi^2$). Having established the initial tempo, the tempo evolves according to

$$T_n = T_{n-1} + \delta_n$$

for $n = 2, 3, \ldots, N$ where $\delta_n \sim N(0, \tau^2(S_{n-1}, S_n))$. When τ^2 takes on relatively small values, this "random walk" model captures the property that the tempo tends to vary smoothly. Note that our model assumes that the variance of $T_n - T_{n-1}$ depends on the transition S_{n-1}, S_n. In particular, longer notes will be associated with greater variability of tempo change.

Finally we assume that the observed note lengths $y_n = o_n - o_{n-1}$ for $n = 1, 2, \ldots, N$ are approximated by the product of the length of the note, $l(S_{n-1}, S_n)$, (measures) and local tempo, T_n, (secs. per measure). Specifically

$$Y_n = l(S_{n-1}, S_n)T_n + \epsilon_n$$

where

$$\epsilon_n \sim N(0, \rho^2(S_{n-1}, S_n)) \qquad (2)$$

Our model indicates that the observation variance depends on the note transition. In particular, longer notes should be associated with greater variance.

These modeling assumptions lead to a graphical model whose directed acyclic graph is given in Figure 2. In the figure each of the variables $S_0, \ldots, S_N$, $T_1, \ldots, T_n$, and $Y_1, \ldots, Y_N$ is associated with a node in the graph. The connectivity of the graph describes the dependency structure of the variables and can be interpreted as follows. The conditional distribution of a variable given all ancestors ("upstream" variables in the graph) depends only on the immediate parents of the variable. Thus the model is a particular example of a Bayesian network [16], [17], [18], [19]. Exploiting the connectivity structure of the graph is the key to successful computing in such models. Our particular model is composed of both discrete and Gaussian variables with the property that, for every configuration of discrete variables, the continuous variables have multivariate Gaussian distribution. Thus, the $S_0, \ldots, S_N$, $T_1, \ldots, T_N, Y_1, \ldots, Y_N$ collectively have a conditional Gaussian (CG) distribution [20], [21], [22], [23].

3 Finding the Optimal Rhythmic Parse

Recall that by "rhythmic parse" we mean a simultaneous estimate of the unobserved rhythm and tempo variables $S_0, \ldots, S_N$ and $T_1, \ldots, T_N$ given observed IOI data $Y_1 = y_1, \ldots, Y_n = y_N$. In view of our probabilistic formulation of the interaction between rhythm, tempo and observables, it seems natural to seek the *most likely* configuration of rhythm and tempo variables given the observed data, i.e. the *maximum a posteriori* (MAP) estimate. Thus, using the notation $a_i^j = (a_i, \ldots, a_j)$ where a is any vector, we let $f(s_0^N, t_1^N, y_1^N)$ be the joint probability density of the rhythm, tempo and observable variables. This joint density can be computed directly from the modeling assumptions of Section 2 as

$$\begin{aligned} f(s_0^N, t_1^N, y_1^N) &= p(s_0) \prod_{n=1}^{N} p(s_n | s_{n-1}) \\ &\times \; p(t_1) \prod_{n=2}^{N} p(t_n | s_{n-1}, s_n, t_{n-1}) \\ &\times \; \prod_{n=1}^{N} p(y_n | s_{n-1}, s_n, t_n) \end{aligned}$$

where $p(s_0)$ is the initial distribution for the rhythm process, $p(s_n|s_{n-1}) = R(s_{n-1}, s_n)$ is probability of moving from measure position s_{n-1} to s_n, $p(t_1)$ is the univariate normal density for the initial distribution of the tempo process, $p(t_n|s_{n-1}, s_n, t_{n-1})$ is the conditional distribution of t_n given t_{n-1} whose parameters depend on s_{n-1}, s_n, and $p(y_n|s_{n-1}, s_n, t_n)$ is the the conditional distribution of y_n given t_n whose parameters also depend s_{n-1}, s_n. The rhythmic parse we seek is then defined by

$$\hat{s}_0^N, \hat{t}_1^N = \arg\max_{s_0^N, t_1^N} f(s_0^N, t_1^N, y_1^N)$$

where the observed IOI sequence, y_1^N, is fixed in the above maximization.

This maximization problem is ideally suited to dynamic programming due to the linear nature of the graph of Figure 2 describing the joint distribution of the model variables. Let $f_n(s_0^n, t_1^n, y_1^n)$ be the joint probability density of the variables S_0^n, T_1^n, Y_1^n (i.e. *up to* observation n) for $n = 1, 2, \ldots, N$. If we define $H_n(s_n, t_n)$ to be the density of the optimal configuration of unobservable variables ending in s_n, t_n:

$$H_n(s_n, t_n) \overset{\text{def}}{=} \max_{s_0^{n-1}, t_1^{n-1}} f_n(s_0^n, t_1^n, y_1^n)$$

then $H_n(s_n, t_n)$ can be computed through the recursion

$$\begin{aligned} H_1(s_1, t_1) &= \max_{s_0} p(s_0)p(s_1|s_0)p(t_1)p(y_1|s_0, s_1, t_1) \\ H_n(s_n, t_n) &= \max_{s_{n-1}, t_{n-1}} H_{n-1}(s_{n-1}, t_{n-1}) \\ &\times\ p(s_n|s_{n-1}) \\ &\times\ p(t_n|t_{n-1}, s_{n-1}, s_n) \\ &\times\ p(y_n|s_{n-1}, s_n, t_n) \end{aligned}$$

for $n = 2, \ldots, N$. Having computed H_n for $n = 1, \ldots, N$ we see that

$$\max_{s_N, t_N} H_N(s_N, t_N) = \max_{s_0^N, t_1^N} f(s_0^N, t_1^N, y_1^N)$$

is the most likely value we seek.

When all variables involved are discrete, it is a simple matter to perform this dynamic programming recursion and to traceback the optimal value value to recover the *globally* optimal sequence $\hat{s}_0^N, \hat{t}_1^N$. However, the situation is complicated in our case due to the fact that the tempo variables are continuous. We have developed methodology specifically to handle this important case, however a presentation of this methodology takes us too far afield. A general description of a strategy for computing the global MAP estimate of unobserved variables, given observed variables, in conditional Gaussian distributions (such as our rhythmic parsing example), can be found in [24].

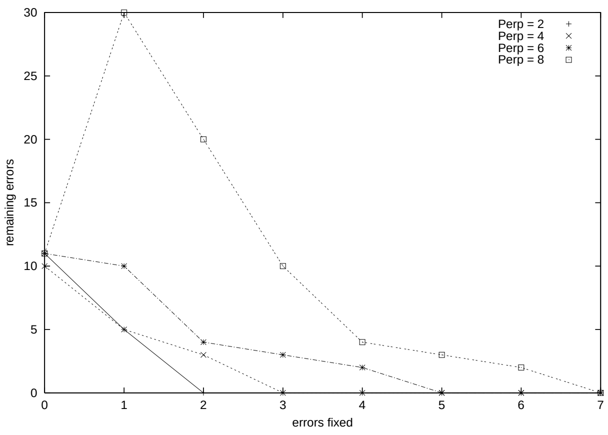


Figure 3: The number of errors produced by our system at different perplexities and with different numbers of errors already corrected.

4 Experiments

We performed several experiments using two different data sets. The first data set is a performance of the first section of Schumann's *2nd Romance for Oboe and Piano* (oboe part only), an excerpt of which is depicted in Figure 1. The original data, which can be heard at http://fafner.math.umass.edu/rhythmic_parsing, is a sampled audio signal, hence inappropriate for our experiments. Instead, we extracted a sequence of 129 note onset times from the data using the HMM methodology described in [25]. These data are also available at the above web page. In the performance of this excerpt, the tempo changes quite freely, thereby necessitating simultaneous estimation of rhythm and tempo.

Since the musical score for this excerpt was available, we extracted the complete set of possible measure positions,

$$\mathcal{S} = \left\{ \frac{0}{1}, \frac{1}{8}, \frac{1}{4}, \frac{1}{3}, \frac{3}{8}, \frac{5}{12}, \frac{15}{32}, \frac{1}{2}, \frac{5}{8}, \frac{3}{4}, \frac{7}{8} \right\}$$

(The position 15/32 corresponds to a grace note which we have modeled as a 32nd note coming before the 3rd beat in 4/4 time). The most crucial parameters in our model are those that compose the transition probability matrix R. The two most extreme choices for R are the uniform transition probability matrix

$$R^{\text{unif}}(s_i, s_j) = 1/|\mathcal{S}|$$

and the matrix ideally suited to our particular recognition experiment

$$R^{\text{ideal}}(s_i, s_j) = \frac{|\{n : S_n = s_i, S_{n+1} = s_j\}|}{|\{n : S_n = s_i\}|}$$

R^{ideal} is unrealistically favorable to our experiments since this choice of R is optimal for recognition purposes and incorporates information normally unavailable; R^{unif} is unrealistically pessimistic in employing no prior information whatsoever. The actual transition probability matrices used in our experiments were convex combinations of these two extremes

$$R = \alpha R^{\text{ideal}} + (1 - \alpha) R^{\text{unif}}$$

for various constants $0 < \alpha < 1$. A more intuitive description of the effect of a particular α value is the *perplexity* of the matrix it produces: $\text{Perp}(R) = 2^{H(R)}$ where $H(R)$ is the $\log_2$ entropy of the corresponding Markov chain. Roughly speaking, if a transition probability matrix has perplexity M, the corresponding Markov chain has the same amount of "indeterminacy" as one that chooses randomly from M equally likely possible successors for each state. The extreme transition probability matrices have

$$\begin{aligned} \text{Perp}(R^{\text{ideal}}) &= 1.92 \\ \text{Perp}(R^{\text{unif}}) &= 11 = |\mathcal{S}| \end{aligned}$$

In all experiments we chose our initial distribution, $p(s_0)$, to be uniform, thereby assuming that all starting measure positions are equally likely. The remaining constants, $\nu, \phi^2, \tau^2, \rho^2$ were chosen to be values that seemed "reasonable."

The rhythmic parsing problem we pose here is based solely on timing information. Even with the aid of pitch and interpretive nuance, trained musicians occasionally have difficulty parsing rhythms. For this reason, it is not terribly surprising that our parses contained errors. However, a virtue of our approach is that the parses can be incrementally improved by allowing the user to correct individual errors. These corrections are treated as constrained variables in subsequent passes through the recognition algorithm. Due to the global nature of our recognition strategy, correcting a single error often fixes others parse errors automatically. Such a technique may well be useful in a more sophisticated music recognition system in which it is unrealistic to hope to achieve the necessary degree of accuracy without the aid of a human guide. In Figure 3 we show the number of errors produced under various experimental conditions. The four traces in the plot correspond to perplexities $2, 4, 6, 8$, while each individual trace gives the number of errors produced by the recognition after correcting $0, \ldots, 7$ errors. In each pass the first error found from the previous pass was corrected. In each case we were able to achieve a perfect parse after correcting 7 or fewer errors. Figure 3 also demonstrates that recognition accuracy improves with decreasing perplexity, thus showing that significant benefit results from using a transition probability matrix well-suited to the actual test data.

In our next, and considerably more ambitious, example we parsed a MIDI performance of the Chopin Mazurka Op. 6, no. 3. for solo piano. Unlike the monophonic instrument of the previous example, the piano can play several notes at a single score position. This situation can be handled with a very simple modification of the approach we have described above. Recall from Section 2 that $l(s, s')$ describes the note length associated with the transition from state s to state s'. We modify the definition of Eqn. 1 to be

$$l(s, s') = \begin{cases} s' - s & \text{if } s' \geq s \\ 1 + s' - s & \text{otherwise} \end{cases}$$

where we have simply replaced the $>$ in Eqn. 1 by $\geq$. The effect is that a "self-transition" (from state s to state s) is interpreted having 0 length, i.e. corresponding to two notes having the same score position.

For this example, in 3/4 time, we took the possible measure positions from the actual score, giving

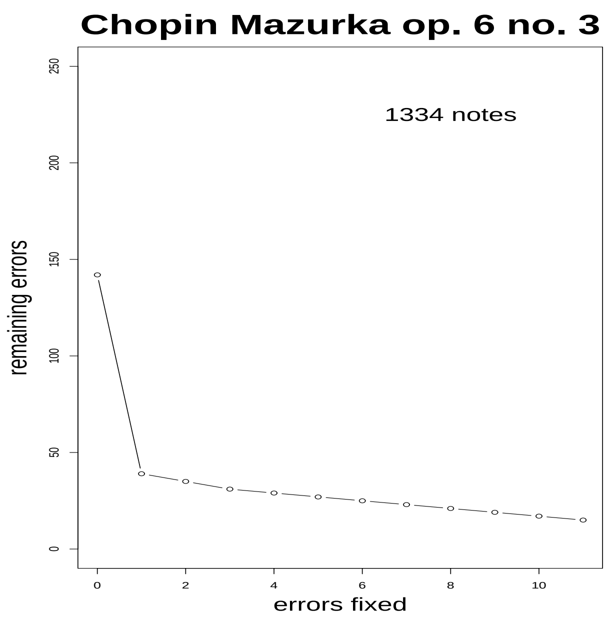


Figure 4: Results of rhythmic parses of Chopin Mazurka Op. 6, No. 3.

the set

$$\mathcal{S} = \left\{ \frac{0}{1}, \frac{1}{3}, \frac{2}{3}, \frac{1}{6}, \frac{11}{12}, \frac{23}{24}, \frac{1}{4}, \frac{1}{9}, \frac{2}{9}, \frac{1}{2}, \frac{5}{6}, \frac{1}{12}, \frac{13}{24}, \frac{7}{12}, \frac{1}{24} \right\}$$

Again, several of the measure positions correspond to grace notes. Rather than fixing the parameters of our model by hand, we instead estimated them from actual data. The transition probability matrix, R, was estimated from scores of several different Chopin Mazurka extracted from MIDI files. The result was a transition probability matrix having $\text{Perp}(R) = 2.02$, thereby providing a model that has enormously improved predictive power over the uniform transition model having perplexity $\text{Perp}(R) = |\mathcal{S}| = 15$. We also learned the variances of our model, $\tau^2(S_{n-1}, S_n)$ and $\rho^2(S_{n-1}, S_n)$ by applying the EM algorithm to a MIDI Mazurka using a known score.

We then iterated the procedure of parsing the data and then fixing the error beginning the longest run of consecutive errors. The results of our experiments with this data set are shown in Figure 4. The example contained 1334 notes. The MIDI file can be heard at http://fafner.math.umass.edu/rhythmic_parsing.

5 Discussion

We have presented a method for simultaneous estimation of rhythm and tempo, given a sequence of note onset times. Our method assumes that the collection of possible measure positions is given in advance. We believe this assumption is a relatively simple way of limiting the complexity of the recognized rhythm produced by the algorithm. When arbitrary rhythmic complexity is allowed without penalty, one can always find a rhythm with an arbitrarily accurate match to the observed time sequence. Thus, we expect that any approach to rhythm recognition will need some form of information that limits or penalizes this complexity. Other than this assumption, all parameters of our model can, and should, be learned from actual data, as in our second example. Such estimation requires a set of training data that "matches" the test data to be recognized in terms of rhythmic content and rhythmic interpretation. For example, we would not expect successful results if we trained our model on Igor Stravinsky's *Le Sacre du Printemps* and recognized on Hank Williams' *Your Cheatin' Heart.* In our experiments with the Chopin Mazurka in Section 4, we used different Chopin Mazurkas for training; however, it is likely that a less precise match between training and test would still prove workable.

We believe that the basic ideas we have presented can be extended significantly beyond what we have described. We are currently experimenting with a model that represents simultaneous evolution of rhythm *and pitch.* Since these quantities are intimately intertwined, one would expect better recognition of rhythm when pitch is given, as in MIDI data. For instance, consider the commonly encountered situation in which downbeats are often marked by low notes as in the Chopin example.

The experiments presented here deal with estimating the *composite* rhythm obtained by superimposing the various parts on one another. A disadvantage of this approach is that composite rhythms can be quite complicated even when the individual voices have simple repetitive rhythmic structure. For instance, consider a case in which one voice uses triple subdivisions while another use duple subdivisions. A more sophisticated project we are exploring is the simultaneous estimation of rhythm, tempo and voicing. Our hope is that rhythmic structure becomes simpler and easier to recognize when one models and recognizes rhythm as the superposition of several rhythmic sources. Rhythm and voicing collective constitute the "lion's share" of what one needs for for automatic transcription of MIDI data.

While the Schumann example was much simpler than the Chopin example, it illustrates another direction we will pursue. Rhythmic parsing can play an important roll in interpreting the results of a preliminary analysis of audio data that converts a sampled acoustic signal into a "piano roll" type of representation. As discussed, we favor simultaneous estimation over "staged" estimation whenever possible, but we feel that an effort to simultaneously recover all parameters of interest from an acoustic signal is extremely ambitious, to say the least. We feel that the two problems of "signal-to-piano-roll" and rhythmic parsing together constitute a reasonable partition of the problem into manageable pieces. We intend to consider the transcription of audio data for considerably more complex data than those discussed here.

References

[1] Hewlett W., (1992), "A Base-40 Number-Line Representation of Musical Pitch Notation," *Musikometrika* Vol. 4, 1–14, 1992.

[2] Hewlett W., (1987), "The Representation of Musical Information in Machine-Readable Format," *Directory of Computer Assisted Research in Musicology*, Vol. 3, 1–22 1987.

[3] Selfridge-Field E., (1994), "The MuseData Universe: A System of Musical Information," *Computing in Musicology*, Vol. 9, 9–30, 1994.

[4] McNab R., Smith L., Bainbridge D., Witten I., (1997) "The New Zealand Digital Library MELody inDEX," D-Lib Magazine, http://www.dlib.org/dlib/may97/meldex/05witten.html May 1997.

[5] Bainbridge D. (1998), "MELDEX: A Web-based Melodic Index Search Service," *Computing in Musicology* Vol. 11 223–230, 1998.

[6] Schaffrath, H., (1992), "The EsAC Databases and MAPPET Software," *Computing and Musicology* vol. 8, 1992, 66.

[7] Desain P, Honing H., (1991) "Towards a calculus for expressive timing in music," *Computers in Music Research*, Vol. 3,43–120, 1991.

[8] Repp B., (1990), "Patterns of Expressive Timing In Performances of a Beethoven Minuet by Nineteen Famous Pianists," *Journal of the Acoustical Society of America* Vol. 88, 622–641, 1990.

[9] Bilmes J., (1993), "Timing is of the essence: Perceptual and computational techniques for representing, learning, and reproducing expressive timing in percussive music," S.M. thesis, Massachusetts Institute of Technology Media Lab, Cambridge, 1993.

[10] Trilsbeek P., van Thienen H., (1999), "Quantization for Notation: Methods used in Commercial Music Software," handout at *106th Audio Engineering Society conference*, May 1999, Munich.

[11] Cemgil A. T., Kappen B., Desain P., Honing, H. (2000), "On Tempo Tracking: Tempogram Representation and Kalman Filtering" *Proceedings of the International Computer Music Conference*, Berlin, 2000.

[12] Desain P., Honing H. (1994), "A Brief Introduction to Beat Induction," *Proceedings of the International Computer Music Conference*, San Francisco, 1994.

[13] Desain P., Honing H. (1989), "The Quantization of Musical Time: A Connectionist Approach," *Computer Music Journal*, Vol 13, no. 3.

[14] Desain P., Aarts R., Cemgil A. T., Kappen B., van Thienen H, Trilsbeek P. (1999), "Robust Time-Quantization for Music from Performance to Score," *Proceedings of 106th Audio Engineering Society conference*, May 1999, Munich.

[15] Cemgil A. T., Desain P., Kappen B. (1999), "Rhythm Quantization for Transcription," *Computer Music Journal*, 60-76.

[16] Lauritzen S. L., (1996), "Graphical Models," Oxford University Press, New York.

[17] Spiegelhalter D., Dawid A. P., Lauritzen S., Cowell R. (1993), "Bayesian Analysis in Expert Systems," *Statistical Science,* Vol. 8, No. 3, pp. 219–283.

[18] Jensen F., (1996), "An Introduction to Bayesian Networks," Springer-Verlag, New York.

[19] Cowell R., Dawid A. P., Lauritzen S., Spiegelhalter D. (1999), "Probabilistic Networks and Expert Systems," Springer, New York.

[20] Lauritzen S. L. and Wermuth N (1984), "Mixed Interaction Models," *Technical Report R-84-8*, Institute for Electronic Systems, Aalborg University.

[21] Lauritzen S. L. and Wermuth N (1989), "Graphical Models for Associations Between Variables, some of which are Qualitative and some Quantitative," *Annals of Statistics*, 17, 31-57.

[22] Lauritzen S. (1992), "Propagation of Probabilities, Means, and Variances in Mixed Graphical Association Models," *Journal of the American Statistical Association*, Vol. 87, No. 420, (Theory and Methods), pp. 1098–1108.

[23] Lauritzen S. L., Jensen F. (1999), "Stable Local Computation with Conditional Gaussian Distributions," *Technical Report R-99-2014*, Department of Mathematic Sciences, Aalborg University.

[24] Raphael C., (2001), "A Mixed Graphical Model for Rhythmic Parsing," *Proceedings of 17th Conference on Uncertainty in Artificial Intelligence*, Seattle, 2001

[25] Raphael C., (1999), "Automatic Segmentation of Acoustic Musical Signals Using Hidden Markov Models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no 4, 360 – 370, 1999.

Melody Spotting Using Hidden Markov Models

Adriane Swalm Durey
Center for Signal and Image Processing
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332
404-894-8361
gte401k@ece.gatech.edu

Mark A. Clements
Center for Signal and Image Processing
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332
404-894-4584
clements@ece.gatech.edu

ABSTRACT
As we acquire more digitized copies of musical recordings, it becomes increasingly necessary to have the assistance of a computer in sorting through the information that it stores. In this paper, we propose a new system for melody-based retrieval of a song from a musical database which adapts wordspotting techniques from automatic speech recognition to create a melody spotting system in the musical domain. This system is tested using a variety of feature sets derived from raw audio data. It results in a successful proof of the melody spotting concept which offers great potential for development into a musical database capable of being queried by melody.

1. INTRODUCTION
As the digital age progresses, we have at our disposal an increasing amount of digitized information ranging from plain text to audio, video, and multimedia content. In order for a person to sift through this vast quantity of information, it is necessary to enlist the aid of the computer which stores it. For text, rapid matching methods have become the familiar search engines of the World Wide Web. Work on video (image) recognition and audio (speech) recognition has a long history. Only recently have we recognized the need to develop recognition systems for other types of audio, particularly music.

The problem we will address in this paper is that of musical database query by melody. In such a system, the user provides the query melody (by humming, whistling, keyboard, etc.) and the musical database system returns the most likely matches to it. Thus, we are interested in recognizing the melodic content of a piece of music, that part which is most recognizable to the casual listener, and, therefore, most easy for him to recall.

There is a growing body of research addressing the problem of music information retrieval (MIR) on the melody level. In the following section, we will address some of the methods used to approach this process in previous research. Then in Section 1.2, we will explore one of the new directions in which this research directs us, adapting the use of hidden Markov models from speech recognition to their use for melody recognition. In Section 2 we will look at the details of the system we propose. Section 3 will discuss the results of preliminary testing of this system. Future directions for research and conclusions will be presented in the final section.

1.1 Melody-Level MIR Research
The seminal works in melodic MIR, those by Ghias, et al. [8] and McNab, et al. [11], form the basis for much of the melody-level MIR research which followed them. Following their format, which is generalized in Figure 1, a database of musical recordings is developed by hand in MIDI (or some similar discrete format) and stored in monophonic tracks. These tracks are then further abstracted into melodic and/or rhythmic contours whose content varies from general (up, down, or the same) to exact (plus or minus a specific number of semi-tones.) When the user queries the system, he sings or hums the desired melody. The system then extracts the pitches (and rhythm) in that query and encodes them as the database has been encoded. The database contents and the query are then compared, generally using some variation of dynamic programming (DP), especially that developed by Mongeau and Sankoff [12]. This process produces a ranked list of the songs in the database which are most likely to contain the melody. While this does not apply to all melodic MIR systems which have been proposed, many of them adopt at least some portion of this framework.

There are a number of problems identifiable in this general structure. The first of these is the use of highly structured audio formats, such as MIDI, to represent the database contents. An accurate automatic process for generating MIDI from polyphonic raw audio has yet to be developed; this means that MIDI encoding must often be done by hand and ignores a vast wealth of data already digitized in less structured formats (like MPEG Layer 3). This format also requires that the contents of the database be firmly (even if erroneously) defined before processing begins. Likewise, if the database utilizes only melodies in processing, they too must be pre-extracted by an expert for greatest accuracy

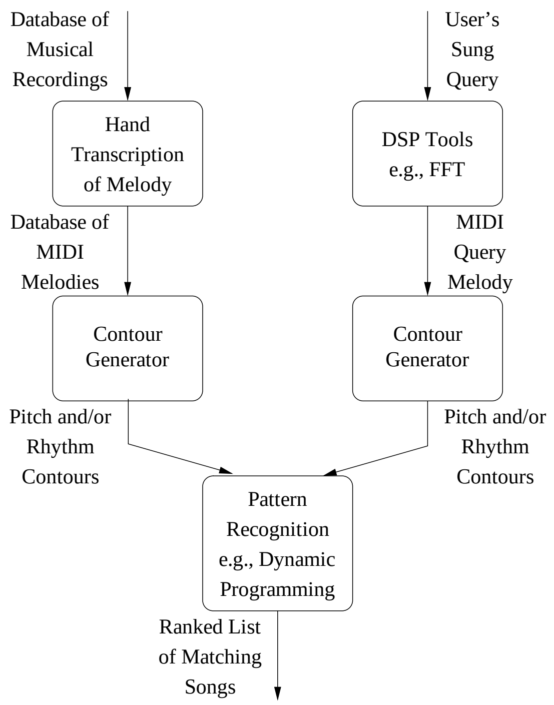

Database of Musical Recordings
User's Sung Query
Hand Transcription of Melody
DSP Tools e.g., FFT
Database of MIDI Melodies
MIDI Query Melody
Contour Generator
Contour Generator
Pitch and/or Rhythm Contours
Pitch and/or Rhythm Contours
Pattern Recognition e.g., Dynamic Programming
Ranked List of Matching Songs

Figure 1: Generalized architecture used for melody-based music information retrieval

(though reasonably good melody-from-MIDI extractors are being developed [17]) and a hard decision made about their content. Polyphonic audio is often not addressed in melody level MIR systems, though sometimes it is represented as several monophonic melodies each representing one voice of the larger polyphonic piece. The encoding of melodies in pitch and rhythm contours is a good way to resolve errors in the database and made by the user. However, we must be careful not to discard too much query information when we can have a reasonable level of confidence in it.

Though other methods applied to melodic MIR have included techniques such as distance measure/template matching [7], n-gram matching [16], tree indexing [3], and text retrieval [4] [6], the primary methodology of most melodic MIR systems is as described above. Such exclusive use of dynamic programming for melodic MIR ignores other comparison techniques which have been successfully applied to other forms of audio signal understanding, particularly those tools used for speech recognition. In the next section, we will explore the new research directions suggested by the problems described above and a particular tool used for automatic speech recognition, the hidden Markov model as it is applied to keyword spotting.

1.2 Hidden Markov Model Research

The analysis of previous work on melody-based MIR suggests a number of potential next steps. The goal of the average user of a MIR system such as we propose is to locate a recording of a piece of music which is much more easily found in an unstructured format (for example, on a CD or in the ubiquitous `mp3` format.) Thus, we would like our system to operate on raw audio formats, such as `wav`, `aiff`, and `mp3`. To avoid the additional problem of making a hard decision about content when generating a structured encoding from raw audio, we would prefer a system that maintains a number of guesses about the content of a recording and qualifies each guess with a likelihood of correctness. This is analogous to the treatment of data in speech recognition databases.

Breaking away from discretized data formats also suggests breaking away to explore new forms of data comparators. Many of the ideas experimented with thus far in MIR research share a common background with speech recognition processing. Tools such as the fast Fourier transform [13] and dynamic programming (DP) [11] have been used for melodic matching with varying levels of success. Other tools such as mel-frequency cepstral coefficients (MFCCs) [7] and hidden Markov models (HMMs) [21] have been used for high-level content matching, classifying audio by type, but are less frequently applied to melody recognition.

There are few references in the literature to the use of HMMs for analysis of musical audio. Logan and Chu [10] used HMMs to analyze the structure of rock and pop songs which had been represented as MFCCs. This structural analysis was then used as a basis for extracting "key phrases" (those of interest structurally, e.g., a chorus) from the audio. This system could be used to supply musical thumbnails of larger recordings to a database access system, but the authors did not specifically address incorporation of their software into such a database. Batlle and Cano [1] likewise used musical audio converted to MFCCs and energy to train competitive HMMs to perform a blind (unsupervised) segmentation task.

HMMs have not yet been applied specifically to melodic content recognition. The application of HMMs closest to MIR appears in Raphael [15] where HMMs are used as a precursor to automatic accompaniment by performing segmentation of raw audio when the score is known. HMMs are constructed based on the pitches and duration noted in a score using features based on amplitude, activity, and frequency content of the signal. After training, these HMMs are used to align audio data with the score that represents it. The problem which we will address allows data to be less well defined in advance and applies a less constrained approach to analysis of the contents of a recording.

HMMs and related techniques have also been applied to the pitch extraction task. Goto [9] used estimation maximization (EM) techniques common to the training of continuous distribution HMMs to evaluate the contributions of various tone models to polyphonic audio sampled from a CD. He then used these weights to make a determination of which frequencies were the most probable melody and bass lines with 88% accuracy and 80% accuracy respectively. Such a system could eventually provide a valuable front end to a musical database system.

Many of the tasks performed in speech processing have direct analogues in music processing. Automatic speech recognition (ASR) can be likened to transcription from raw audio to common music notation. Keyword wordspotting (WS) can be likened to spotting a query melody occurring in raw audio. The successful use of HMMs in automatic speech recognition and wordspotting applications suggests that, like DP, such tools might make a successful transition to melody recognition. These correlations drive the system developed in this paper.

In speech processing, wordspotting is the task of searching for keywords occurring somewhere in a larger body of less constrained audio data (noises, non-keywords, etc.) [18] In music, we would like to search for specific melodies occurring somewhere in a larger body (the database) of less constrained audio data (other music, harmony, noises, speech, etc.) Some of the most successful methods for wordspotting have involved HMMs: one set trained to recognize a key word or words and one set trained to account for the "garbage" making up the rest of the audio. We propose to define a set of HMMs to recognize a query melody and a set of HMMs trained to account for the auditory "garbage" not associated with that query. Thus, we hope to perform the musical equivalent of wordspotting. The next section will develop the specific configurations of HMMs used in our implementation of melody spotting.

2. EXPERIMENTAL SET-UP

The following two sections will describe the composition of the melody spotting system we test in this paper. First, the methods of data collection used in our experiments are described. We elaborate on the construction of our HMM melody spotting system in Section 2.2.

2.1 Database and Training Data

The data for this series of experiments consists of a collection of simple monophonic melodies. This allows us to begin with a relatively easy task, since the melody is the only musical data in each recording. The melodies were transposed such that no accidentals occurred in play to ensure adequate coverage of each note. All the notes fell into the range from C_4 to G_5. Each melody was played ten times on a keyboard by an amateur musician. The melodies were played each time in the same key and register (the same exact notes) but using five different instrumental voices (two recordings per voice.) The different recordings were produced for each melody to allow for natural variations in play. The changes in instruments and duration add some difficulty to the task; no two renditions of a melody are exactly alike. As the acoustic data was collected, so was the corresponding MIDI data. The use of each type of data will be illustrated in the next section. The selection of tunes recorded is given in Table 1 and the instrument voices in Table 2. (A transcription of the songs as recorded is provided in Appendix A.)

The data were collected from a Yamaha W7 keyboard at a sampling rate of 22050 Hertz using mono audio input and saved in `wav` format. This introduced additional complexity to the task due to the varying levels of noise (sometimes negligible, sometimes nearly overpowering) in the data set generated by the recording process. This resulted in approximately half an hour of recorded data requiring about 100 Mbytes of computer storage.

Table 1: List of Songs Used in Testing

	Song Title
1	*Auld Lang Syne*
2	*Barbara Allen*
3	*Frere Jacques*
4	*Happy Birthday to You*
5	*I'm a Little Teapot*
6	*Mary Had a Little Lamb*
7	*Scarborough Fair*
8	*This Old Man*
9	*Three Blind Mice*
10	*Twinkle, Twinkle, Little Star*

Table 2: List of Instrument Voices Used in Testing

Instrument Name
Clarinet
Flute
Piano
Soprano Saxophone
Violin

2.2 HMM Melody Recognition System

The construction of our melody spotting system is very similar to that of a wordspotting system for speech recognition. We will describe the system from the lowest levels, where the collected data intersect it, to the highest, where we see the melody spotting results, that is, the occurrences in the database of the desired query.

A hidden Markov model (HMM) describes a doubly stochastic process in which only the current state affects the choice of the next state. [14] HMMs model two levels of activity, a visible layer represented by the observation data it produces, and an underlying hidden layer representing the states through which the model passes. The transitions between states and the observations are governed by probabilities learned from the training data using a collection of well-known techniques. Our implementation is based on the HMM Tool Kit (HTK). [20] This system was selected because it has been used successfully in automatic speech recognition research (as in Woodland, et al. [19],) but allows the user enough design freedom to adapt it to a musical task such as our own.

The HMM is the basic building block for our melody spotter. A simple, left-to-right, five state HMM is trained to represent each note for which data is available (C_4–G_5), plus a rest (silence/noise). Such an HMM is shown in Figure 2. The first and final states are non-emitting, serving only to designate the beginning and end of the HMM. Probabilities a_{ij} govern the transitions from state i to state j. Probability distributions $b_i(O_t)$ represent the probability of seeing a given observation O at time t while in state i. The a_{ij} and $b_i(O_t)$ are the parameters that we must train using the available data. In our implementation, the probability density functions $b_i(O_t)$ for each state are modeled as Gaussian distributions with a non-uniform diagonal variance.

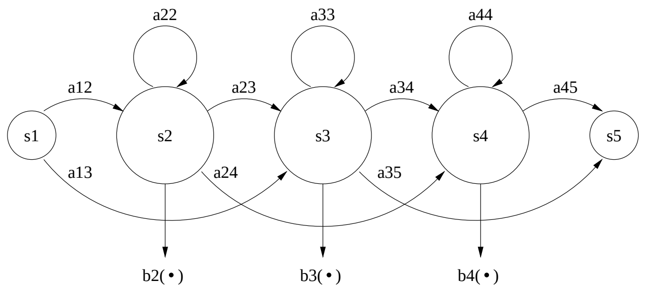


Figure 2: Hidden Markov model representing a basic musical unit: a note (for example, C_4) or rest

The raw audio data described in the previous section are stored for database purposes, but are also processed for use in training and evaluating the HMMs. For comparison purposes, we extract two different sets of features. These features were chosen as simple starting points and will be tested against a selection of other representations in later versions of this system. In both cases, each raw audio file is normalized. The first features are produced using a fast Fourier transform (FFT) with a Hamming window length of 2048 and a skip of 1024 (each window covers approximately 100 msec = .1 sec). The full results of the FFT are then trimmed (band-limited) to use only those bins corresponding to a reasonable range of frequencies for music and for the task at hand (from C_4 to C_6, or approximately 261–1044 Hz.) For the second set of features, a single pitch estimate is selected using an autocorrelation method. [5] The result in both cases is a series of observation vectors ready to be processed by the HMMs, one of length 75, the other of length one. The corresponding MIDI data is aligned with the recording by matching the onset of the first note in the `wav` recording with the first note in the MIDI file. It is then transformed into a label file in the style used by HTK, which will play a role in training and testing of the HMM melody spotting system. Figure 3 outlines this process. In order to save processing time, this preprocessing can be performed once, and the resulting data stored for later use.

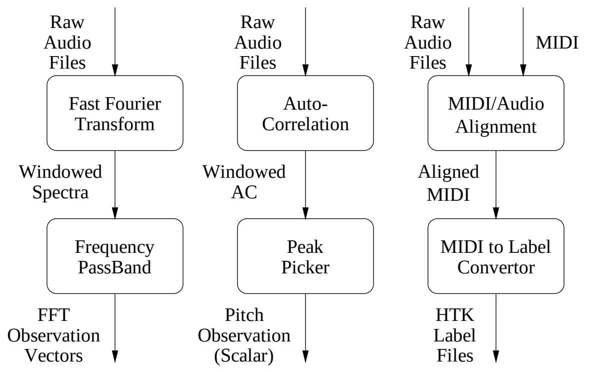


Figure 3: Data processing from raw audio to HMM observation vectors and from MIDI to HTK label data

With the HMM prototype and the preprocessed data, training can now proceed. In this preliminary system, because of the small size of the melodic database, both training and testing utilized all of the available data. We hope that future work will show that it is possible to adequately model the contents of the database without requiring the system to be trained using all of the music contained by the database. First, the label data is used to segment the input data by note as shown in Figure 4. Each segment of data is used to initialize the HMM representing that note using Viterbi alignment. The same feature segments are then used to refine each HMM using Baum-Welch reestimation. Finally, concatenations of the note level HMMs appropriate to each training melody are created and embedded Baum-Welch reestimation is performed on each one using the complete sequence of feature vectors for that recording. [20] (Such a concatenation is illustrated at the top of Figure 5.) Once this training is complete, the system is ready to accept a query.

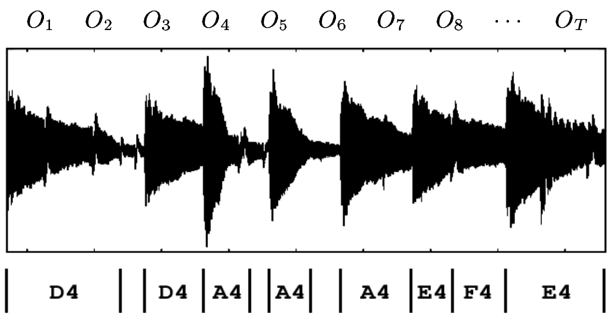


Figure 4: A series of feature vectors, O_t, extracted from raw audio and segmented using the MIDI label data. This is the first eight notes of the song *Scarborough Fair.*

When a sequence of notes defining a query is provided by the user, we are ready to create the remainder of our recognition system. For this beginning system, the query is simply an exact sequence of notes entered as text by the user, e.g, "D_4, D_4, ... , E_4." First, copies of the note-level HMMs making up the query are concatenated together to form an HMM representing the query as a whole. (See Figure 5.) Because of the current selection of models and features, this will seek an exact match to the pitches in the query. Then, all of the available notes and a rest/noise HMM are placed in parallel with the desired melody to act as fillers or garbage collectors. [18] To prevent the fillers from overwhelming the more complex (and therefore less probable) melody HMM, entry into those states is penalized, especially for those notes occurring in the melody. Looping from the final to initial states allows the HMM system, now a musical language model, to process a much larger piece of data, a complete song stored in the database.

The features representing each song in the database are then run through the constructed HMM using Viterbi scoring; the song is sectioned and each section is labelled as to its probability of belonging to the melody or to one of the fillers. The locations labelled as the melody are then sorted according to their probability of occurrence, producing a ranked list of most likely occurrences of the melody in the database songs. In testing, these results can be compared to the MIDI reference data to determine their accuracy. (Once initialization is complete, the MIDI data is not used except for judging the accuracy of the results returned by the HMM system

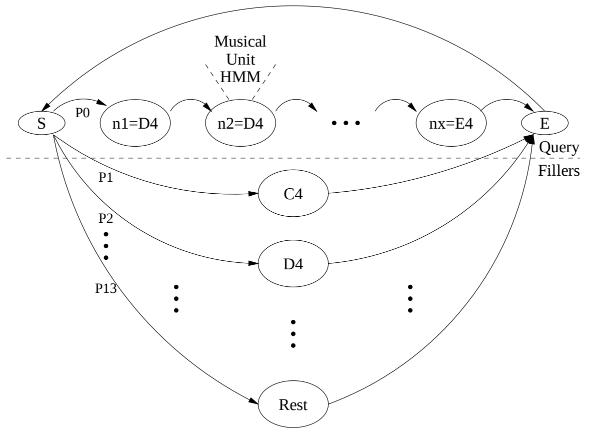


Figure 5: Construction of a musical language model representing a melodic query from the musical unit HMMs shown in Figure 2. The musical unit HMMs are used both to model the query melody and the filler composing the rest of a database entry.

in testing.) If it is an actual query, the results can be presented to the user for audition. The sectioning and scoring also allows us to audition the piece where the melody likely occurs, rather than from the beginning.

It is important to note that the MIDI data is not a requirement for this system, but rather a convenience. Initialization could be accomplished without the exact MIDI data using only a note list (no onset and offset data) in a bootstrap method. [20] It should also be possible to initialize the system with a subset of MIDI-labelled recordings sufficient to train the musical models in the system. The eventual goal would be to add to the database other musical pieces for which such label data is unavailable. Further testing would be needed to determine the efficacy of these plans compared to our current training process.

3. EXPERIMENTAL RESULTS

Melody recognizers based on both sets of observation vectors were trained using the techniques described in the previous section. These systems were then evaluated by searching for a selection of melodies of various lengths, from three to ten notes each. As mentioned previously, the queries are posed as a string of textual note names, e.g. "G_4, F_4, E_4, D_4, C_4." For these initial tests of the system, no errors were introduced into the queries. The test queries were selected based on their frequency of occurrence in the database songs so that several copies would exist to be searched for by the system. The selected query melodies and the songs in which they occur are given in Table 3. For some lengths, multiple queries were used to allow the search to operate over several different songs. Though this is an admittedly small selection of queries from a small collection of melodies, further testing on an expanded database is currently underway.

Once each query is selected, a detection network similar to that in Figure 5 is constructed to represent it. Initial penalties for the melody and filler arcs are assigned as:

Table 3: List of Query Melodies and Their Locations

Query	*Note String*	*Found in Songs*
3a.	E_4 D_4 C_4	2, 5, 7, 9
3b.	E_5 D_5 C_5	3, 8, 10
4.	F_4 E_4 D_4 C_4	2, 5, 7, 9
5.	G_4 F_4 E_4 D_4 C_4	2, 7, 9
6.	D_5 C_5 C_5 B_4 B_4 A_4	10
7a.	C_5 C_5 B_4 A_4 B_4 C_5 G_4	9
7b.	G_4 D_5 D_5 E_5 E_5 D_5 C_5	10
8a.	C_5 C_5 B_4 A_4 B_4 C_5 G_4 G_4	9
8b.	C_4 D_4 E_4 F_4 G_4 C_5 A_4 C_5	5
8c.	E_5 E_5 D_5 C_5 C_5 B_4 B_4 A_4	10
9a.	C_5 C_5 B_4 A_4 B_4 C_5 G_4 G_4 G_4	9
9b.	C_4 D_4 E_4 F_4 G_4 C_5 A_4 C_5 G_4	5
9c.	G_4 G_4 D_5 D_5 E_5 E_5 D_5 C_5 C_5	10
10a.	G_4 C_5 C_5 B_4 A_4 B_4 C_5 G_4 G_4 G_4	9
10b.	G_4 G_4 D_5 D_5 E_5 E_5 D_5 C_5 C_5 B_4	10

$$P_n = \begin{cases} 2/M, & \text{arc into melody block} \\ 1/M, & \text{arc into filler note not in melody} \\ 1/1000, & \text{arc into filler note in melody} \end{cases}$$

where M is the number the filler blocks and melody blocks in that network (for example, $M = 14$ for C_4–G_5 plus a rest and a query melody block.) The selection of the penalties for fillers occurring in the melody will be addressed again in the following sections. Once this is done, each song in the database is scored using the query network, resulting in a segmentation of each song into melody and fillers, each with a start and stop time and an associated probability score. Now, the segments labelled as melodies can be ranked by score and provided to a database user as results. Since the data is preprocessed, the scoring requires less than real-time to compute. Real-time here is defined by 100 songs times an average of 20 seconds a song or ≈ 2000 sec = 33.33 min. On a 650 MHz Pentium III with 256 Kb of RAM, receiving results from the scoring algorithm requires only about a minute, a speed up of approximately 3300%.

For testing, the results of the scoring are compared to a label file which marks the query locations in the database (the second use of the MIDI reference data.) This test ensures that those database elements labelled as containing the query do so and in the locations given by the recognizer. Two results are provided by this comparison. The first is a numerical figure-of-merit (FOM) commonly used in evaluating speech-based wordspotters. The FOM places an upper-bound estimate on word spotting accuracy (the percentage of true hits) averaged over 1 to 10 false alarms per hour. [20] (The FOM as defined by HTK is further described in Appendix B.) In the case of a melody spotter, however, it is possible for the false alarms to be interpreted differently. They must be examined to determine if they sound similar to the query—an inexact, but still valid, match.

The following two sections address the systems built using each of the observation vectors defined in Section 2. Section 3.3 describes a third system tested on a musically based subset of the FFT data. Discussion of results common to all of these systems is presented in Section 3.4. Problems and future work suggested by this testing are addressed in the final section of the paper.

3.1 Pitch-Based Recognizer

Our first melody recognition system is constructed on a length one observation vector, the fundamental frequency of the audio signal as estimated using autocorrelation (AC). [5] This estimator was selected because, of those tested, it provided the best pitch estimates using our data set. The preprocessing time necessary to calculate these features for all of the music in the database is approximately 40 minutes on a 650 MHz Pentium III when computed using Matlab.

For each query melody, we began testing the recognition with a penalty of 1/1000 on fillers occurring in the melody, then tested several lower penalties as well.[1] Generally, new penalties were tried as long as they continued to reduce false alarms without causing any decrease in the number of accurate hits. The penalties were never less than 1/25, a limit slightly greater than the penalties (1/14) applied to the other fillers. The hits, false alarms, actual occurrences, and figures of merit achieved for each query and the penalties applied to the fillers for that value are given in Table 4.

Table 4: Query Results Using Pitch-Based HMM Recognizer

Query	*# Hits*	*# FAs*	*# Actual*	*FOM*	P_n
3a.	70	2	70	95.63	1/500
3b.	19	69	50	0.00	
4.	40	0	40	100.00	1/500
5.	30	10	30	78.80	1/500
6.	40	32	40	0.00	1/50
7a.	30	19	30	42.80	1/250
7b.	8	5	20	28.00	
8a.	30	10	30	78.60	1/500
8b.	20	0	20	100.00	
8c.	18	24	20	27.50	
9a.	30	7	30	93.40	1/500
9b.	20	0	20	100.00	
9c.	8	7	20	23.20	
10a.	20	30	20	23.80	1/500
10b.	8	8	20	20.80	

Overall, the performance of the pitch-based recognizer was poorer than that of the FFT-based recognizer, especially for queries in the range of C_5–G_5. It was observed that the pitch estimator was less accurate for these frequencies than for those in the lower register, so it was expected that the recognizer trained using those pitch estimates would be as well. It was also more difficult to reduce false alarms using the penalties without lowering the hits as well. As might be expected, the quality of the false alarms was not good in all cases, especially when large numbers of them were returned by the system.

[1]Penalty will henceforth refer to the penalty on fillers occurring within the query melody. For our purposes, a lower penalty implies a fraction with a smaller denominator; hence, lowering a penalty increases the probability that a filler will be found in the data set.

3.2 FFT-Based Recognizer

The second melody recognition system is based on the length 75, band-limited, fast Fourier transform (FFT) of the audio data. The preprocessing time required to compute in Matlab the FFTs of all the music in the data set is approximately five minutes.

Again, each query melody was tested under a variety of penalties ranging from 1/1000 to 1/25. The results of testing this recognizer are summarized in Table 5. Generally, the FFT-based recognizer was able to operate reasonably well under the same penalty, 1/1000, locating all but one of the actual occurrences, and, generally, only false alarms sounding similar to the query. This recognizer did not seem to share the upper register difficulties of the pitch-based mechanism, most probably because no hard decision about the content of the data (i.e., the pitch of the melody) was required before processing the observations. In this system, we allowed the HMM to learn statistical representations of the content of the signal, rather than defining the melodic content (the pitch) before applying statistical tools to it. In most cases, it was possible to achieve as much reduction in false alarms as desired without sacrificing successful hits.

Table 5: Query Results Using FFT-Based HMM Recognizer

Query	*# Hits*	*# FAs*	*# Actual*	*FOM*	P_n
3a.	70	1	70	100.00	1/1000
3b.	49	7	50	83.24	
4.	40	3	40	96.10	1/1000
5.	30	0	30	100.00	1/50
6.	40	10	40	99.25	1/1000
7a.	30	0	30	100.00	1/1000
7b.	20	13	20	100.00	
8a.	30	0	30	100.00	1/1000
8b.	20	0	20	100.00	
8c.	20	20	20	100.00	
9a.	30	0	30	100.00	1/1000
9b.	20	0	20	100.00	
9c.	20	10	20	100.00	
10a.	20	10	20	79.60	1/1000
10b.	20	12	20	100.00	

3.3 Scale-Based Recognizer

A third system based on a subset of the FFT observation vector data was implemented and tested as well. First, those FFT bins containing the fundamental frequencies and harmonics of the notes in the range C_4–G_5 (of the equally tempered scale) were identified. They were then used to select a subset of 25 of the 75 bins composing the previously calculated FFT feature vector. The system was designed to offer a compromise between forcing a hard pitch decision before processing (as in the AC pitch feature) and extracting a feature set using no specifically musical knowledge (as in the FFT feature vector). The preprocessing time for this feature set is negligible, since it simply masks the pre-computed FFT data.

The query data are tested as before and the results are summarized in Table 6. These results suggest that the scale-based recognizer strikes a good balance between discarding as much unnecessary information as possible while retaining as much information as it can that is relevant to this musical task. It successfully finds all of the occurrences of each query, generally without producing false alarms which sound too dissimilar from the query. It seems to operate as well as or better than the FFT-based recognizer while reducing the size of the feature vectors by a factor of three (and thus the storage requirements, the number of free parameters to train, and the overall processing time.)

Table 6: Query Results Using Scale-Based HMM Recognizer

Query	*# Hits*	*# FAs*	*# Actual*	*FOM*	P_n
3a.	70	0	70	100.00	1/100
3b.	50	0	50	100.00	
4.	40	0	40	100.00	1/25
5.	30	0	30	100.00	1/25
6.	40	10	40	100.00	1/100
7a.	30	3	30	82.00	1/100
7b.	20	11	20	100.00	
8a.	30	0	30	100.00	1/100
8b.	20	0	20	100.00	
8c.	20	11	20	100.00	
9a.	30	0	30	100.00	1/100
9b.	20	0	20	100.00	
9c.	20	10	20	100.00	
10a.	20	10	20	76.90	1/100
10b.	20	10	20	100.00	

3.4 Common Results

There are several common threads running through the results presented above. The first of these is that, in general, the greater the penalties, the more false alarms and hits allowed by the system, and the lower the penalties, the more false alarms are reduced (though sometimes at the expense of the hits.) In the case of the shorter queries, it is more reasonable to reduce the penalties because they tend to produce a large number of accurate hits. However, as queries become longer, it becomes more desirable to allow more non-exact ("inaccurate") matches to be presented as results, so increasing the penalties to allow more false alarms makes sense. It is worth noting that under many different levels of penalties, nearly all of the desired results were accurately returned by the system. This implies that penalties could be used to dynamically control the level of inexact responses provided by the system.

In all test cases, there is also something the FOM does not tell us: the quality of the segments it labels as false alarms. In the case of wordspotting, distinguishing close but wildly different words like "goat" from "goatee" makes sense, but in music, such a close match often sounds similar enough to count as a good match. One example of this phenomenon from the test data is shown in Figure 6. Generally, those inexact results found by the system resulted from consolidation of repeated notes in the query into one longer note in the response or from deletion of a note from the query. Thus the FOM is both useful and deceptive, since many of the false alarms would be worth presenting to a database user as possible matches to a melodic query.

G_4	G_4	D_5	D_5	E_5	E_5	D_5	C_5	C_5	B_4
Twin-	kle,	Twin-	kle,	lit-	tle	star.	How	I	won-

versus

G_4	D_5	E_5	D_5	C_5	B_4
ing.	All	the	bells	are	ring-

Figure 6: A query melody from *Twinkle, Twinkle, Little Star* and a shorter inexact, but similar sounding, match from *Frere Jacques*

4. CONCLUSIONS AND FUTURE WORK

It is apparent from this work that the melody recognition systems described here may stand only as a proof-of-concept. While we feel it is a successful first foray, there is a great deal of future work that should be done to improve them. This work can be grouped into three areas: that relating to the database and its contents, to the HMM recognizers, and to testing the system. All of these suggestions share the eventual aim of providing a raw audio-based musical database which is friendly to novice and expert user alike.

There are a number of improvements that should be made to the database and the data which it contains. We would like to test this system on a collection of recordings from acoustic instruments, rather than from synthesized ones, since the goal of the system is to operate on such data. Likewise, we would like to test the system with polyphonic data as well as monophonic, thus representing the vast majority of recorded music. We also want to increase the size of any database on which we operate to be more realistic, preferably while retaining an implementation that does not require hand-marking the complete contents of the database. Finally, we would like to improve the interfacing of the database for the benefit of both the researcher and the eventual end-user.

There are many aspects of research on HMMs for automatic speech recognition and wordspotting that we would like to explore in conjunction with our melody recognition system. It would be valuable to experiment with training that does not require full labelling of the data for all database entries and with training on only a subset of the data. We would like to handle different abstractions of query and database data (as other systems have done using melodic contours) based on user confidence in their query. This will likely require us to research a different selection of feature vectors than those presented here. We will also want to examine how different feature vectors affect the quality of our results; for example, a better pitch extractor (as in Goto [9]) might allow us to use smaller observation vectors without sacrificing accuracy or a constant-Q transform [2] might allow us to use only that frequency information pertaining to the musical scale. Such changes might allow us to maintain or increase accuracy while reducing the number of free parameters estimated by the system and the overall processing time required. Feature selection will also impact the types of input accepted by the system; a different selection of features might make it easier for the system to accept sung queries, for example.

Techniques like incorporating bi- and tri-grams, as is done in large vocabulary continuous speech recognizers, might also improve accuracy, as might exploring filler models other than actual notes. We should also further explore penalty selection, hopefully finding a consistent method by which to apply them. We need to look at ways we can ensure that the system accounts for allowable melodic differences (errors users might make in a query) like insertions, substitutions, deletions, etc. We must also explore ways to incorporate duration information in the query (perhaps similarly to [15].) Currently, this information is normalized out of the note level HMMs during training. We need also to address questions of complexity and scalability, perhaps exploring the common ground between our system and those developed for large scale speech recognition tasks. Lastly, we need to collect training data over a broader range of notes so that note level HMMs can be trained for them as well; we could also explore ways of filling in the notes for which adequate data is missing, perhaps with parameter tying. Techniques like parameter tying might also allow a valuable reduction of the number of free parameters to be trained.

Finally, the system needs vastly expanded testing, since the results of such testing would be invaluable in the development of later generations of this melody recognition software. It needs to be tested on a larger collection of exact melodies, as well as on melodies which encompass query errors commonly made by a user. We need to develop a way to convert a user query in an audio format (sung, hummed, whistled, etc.) into a form appropriate to querying the underlying HMM system. (Currently, the most viable method of doing so would be to perform pitch detection on the input, then use the detected pitches in place of our textual query string.) It would also be desirable to compare this system to other comparable systems. However, there are no truly comparable systems in the literature at this time (i.e., those which perform melody recognition on database entries that have not been discretely encoded) nor does there exist a standard body of music on which to perform such a comparison.

In this paper, we have presented a prototype system for spotting a query melody in a larger body of raw musical audio data by adapting HMM wordspotting techniques from the speech recognition domain to the musical domain. While the system as presented here represents a proof-of-concept for this work, it offers great potential for future development. It also offers by-products to other music related applications, such as transcription from raw audio to musical notation and automatic accompaniment. We plan to continue developing this system into a more fully capable melody-based musical database query system, following the suggestions set forth above. To address this desire, the authors are currently working on testing an expanded melody-only and polyphonic database using a collection of feature vectors and incorporating new elements, such as errors in query formulation, into the testing.

Acknowledgments

The authors would like to acknowledge Lincoln Durey for his helpful suggestions, Emperor Linux for generous computer donations, the Interactive Media Technology Center for loan of the Yamaha keyboard, and the Clare Booth Luce Fellowship program for their financial support of this project.

5. REFERENCES

[1] E. Batlle and P. Cano. Automatic segmentation for music classification using competitive hidden Markov models. In *Proc. of ISMIR*, Plymouth, MA, Oct. 2000. Available on the Internet: http://ciir.cs.umass.edu/music2000/.

[2] J. C. Brown. Calculation of a constant Q spectral transform. *J. of the Acoustical Society of America*, 89(1):425–434, Jan. 1991.

[3] J. C. C. Chen and A. L. P. Chen. Query by rhythm: An approach for song retrieval in music databases. In *Proc. of 8th International Workshop on Research Issues in Data Engineering*, pages 139–146, Orlando, FL, Feb. 1998.

[4] M. Clausen, R. Engelbrecht, D. Meyer, and J. Schmitz. PROMS: A web-based tool for searching in polyphonic music. In *Proc. of ISMIR*, Plymouth, MA, Oct. 2000. Available on the Internet: http://ciir.cs.umass.edu/music2000/.

[5] J. R. Deller, J. G. Proakis, and J. H. L. Hansen. *Discrete-Time Processing of Speech Signals.* Prentice Hall, Upper Saddle River, NJ, 1987.

[6] J. S. Downie. Music retrieval as text retrieval: Simple yet effective. In *Proc. of SIGIR*, pages 297–298, Berkeley, CA, Aug. 1999.

[7] J. Foote. An overview of audio information retrieval. *ACM Multimedia Systems J.*, 7(1):2–10, Jan. 1999.

[8] A. Ghias, J. Logan, D. Chamberlin, and B. C. Smith. Query by humming: Musical information retrieval in an audio database. In *Proc. of ACM MM '95*, pages 231–236, San Francisco, CA, Nov. 1995.

[9] M. Goto. A predominant-F0 extimation method for CD recordings: MAP estimation using EM algorithm for adaptive tone models. In *Proc. of ICASSP*, Salt Lake City, UT, May 2001. Electronic Proc.

[10] B. Logan and S. Chu. Music summarization using key phrases. In *Proc. of ICASSP*, volume 2, pages 749–752, Istanbul, Turkey, June 2000.

[11] R. J. McNab, L. A. Smith, D. Bainbridge, and I. H. Witten. The New Zealand Digital Library MELody inDEX. *D-Lib Magazine*, May 1997. Available on the Internet: http://www.dlib.org/dlib/may97/meldex/05witten.html.

[12] M. Mongeau and D. Sankoff. Comparison of musical sequences. *Computers and the Humanities*, 24(3):161–175, June 1990.

[13] S. Pfeiffer, S. Fischer, and W. Effelsberg. Automatic audio content analysis. In *Proc. of ACM MM '96*, pages 21–30, Boston, MA, Nov. 1996.

[14] L. R. Rabiner and B. J. Juang. An introduction to hidden Markov models. *IEEE Audio, Speech, and Signal Processing Magazine*, pages 4–16, Jan. 1986.

[15] C. Raphael. Automatic segmentation of acoustic musical signals using hidden Markov models. *IEEE Trans. on PAMI*, 21(4):360–370, April 1999.

[16] A. Uitdenbogerd and J. Zobel. Melodic matching techniques for large music databases. In *Proc. of ACM MM '99*, pages 57–66, Orlando, FL, Oct.–Nov. 1999.

[17] A. L. Uitdenbogerd and J. Zobel. Manipulation of music for melody matching. In *Proc. of ACM MM '98*, pages 235–240, Bristol, UK, Sept. 1998.

[18] J. G. Wilpon, L. R. Rabiner, C.-H. Lee, and E. R. Goldman. Automatic recognition of keywords in unconstrained speech using hidden Markov models. *IEEE Trans. on ASSP*, 38(11):1870–1878, Nov. 1990.

[19] P. C. Woodland, J. J. Odell, V. Valtchev, and S. J. Young. Large vocabulary continuous speech recognition using HTK. In *Proc. of ICASSP*, volume 2, pages II–125–128, Adelaide, Australia, April 1994.

[20] S. Young, D. Kershaw, J. Odell, D. Ollason, V. Valtchev, and P. Woodland. *The HTK Book (for HTK Version 3.0)*. The Microsoft Corporation, 2000. Available on the internet: http://htk.eng.cam.ac.uk/.

[21] T. Zhang and C.-C. Jay Kuo. Hierarchical system for content-based audio classification and retrieval. In *Proc. of the SPIE Conference on Multimedia Storage and Archiving Systems III*, pages 398–409, Boston, MA, Nov. 1998.

APPENDIX

A. SONG TRANSCRIPTIONS

Note name transcriptions are provided here for each of the pieces of music used in testing our melody spotting system. Durations are excluded for space considerations and because the current implementation effectively normalizes them out of consideration. Examples of each piece may be heard at: `http://www.ece.gatech.edu/~gte401k/melodyspotting/` Though all pitches are played as notated, not all durations are played equally due to the natural variation that was desired in the training data.

Auld Lang Syne

C_4 F_4 E_4 F_4 A_4 G_4 F_4 G_4 A_4 G_4 F_4 F_4 A_4 C_5 D_5
D_5 C_5 A_4 A_4 F_4 G_4 F_4 G_4 A_4 G_4 F_4 D_4 D_4 C_4 F_4

Barbara Allen

C_4 E_4 F_4 G_4 F_4 E_4 D_4 C_4 D_4 E_4 G_4 C_5 C_5 B_4 G_4
C_5 C_5 A_4 G_4 F_4 A_4 G_4 E_4 D_4 C_4 D_4 E_4 F_4 G_4 F_4 E_4 D_4

Frere Jacques

G_4 A_4 B_4 G_4 G_4 A_4 B_4 G_4 B_4 C_5 D_5 B_4 C_5 D_5
D_5 E_5 D_5 C_5 B_4 G_4 D_5 E_5 D_5 C_5 B_4 G_4
G_4 D_4 G_4 G_4 D_4 G_4

Happy Birthday

G_4 G_4 A_4 G_4 C_5 B_4 G_4 G_4 A_4 G_4 D_5 C_5
G_4 G_4 G_5 E_5 C_5 B_4 A_4 F_5 F_5 E_5 C_5 D_5 C_5

I'm a Little Teapot

C_4 D_4 E_4 F_4 G_4 C_5 A_4 C_5 G_4
F_4 F_4 G_4 E_4 E_4 D_4 D_4 E_4 C_4
C_4 D_4 E_4 F_4 G_4 C_5 A_4 C_5 G_4
C_5 C_4 D_4 E_4 F_4 E_4 D_4 C_4

Mary Had a Little Lamb

B_4 A_4 G_4 A_4 B_4 B_4 B_4 A_4 A_4 A_4 B_4 D_5 D_5
B_4 A_4 G_4 A_4 B_4 B_4 B_4 B_4 A_4 A_4 B_4 A_4 G_4

Scarborough Fair

D_4 D_4 A_4 A_4 A_4 E_4 F_4 E_4 D_4
A_4 C_5 D_5 C_5 A_4 B_4 G_4 A_4
D_5 D_5 D_5 C_5 A_4 A_4 G_4 F_4 E_4 C_4
D_4 A_4 G_4 F_4 E_4 D_4 C_4 D_4

This Old Man

D_5 B_4 D_5 D_5 B_4 D_5 E_5 D_5 C_5 B_4 A_4 B_4 C_5
B_4 C_5 D_5 G_4 G_4 G_4 G_4 G_4 A_4 B_4 C_5 D_5
D_5 A_4 A_4 C_5 B_4 A_4 G_4

Three Blind Mice

E_4 D_4 C_4 E_4 D_4 C_4 G_4 F_4 F_4 E_4 G_4 F_4 F_4 E_4
G_4 C_5 C_5 B_4 A_4 B_4 C_5 G_4 G_4
G_4 C_5 C_5 C_5 B_4 A_4 B_4 C_5 G_4 G_4
G_4 G_4 C_5 C_5 B_4 A_4 B_4 C_5 G_4 G_4 G_4 F_4 E_4 D_4 C_4

Twinkle, Twinkle, Little Star

G_4 G_4 D_5 D_5 E_5 E_5 D_5 C_5 C_5 B_4 B_4 A_4 A_4 G_4
D_5 D_5 C_5 C_5 B_4 B_4 A_4 D_5 D_5 C_5 C_5 B_4 B_4 A_4
G_4 G_4 D_5 D_5 E_5 E_5 D_5 C_5 C_5 B_4 B_4 A_4 A_4 G_4

B. FIGURE-OF-MERIT

HTK [20] uses the National Institute of Standards and Technology (NIST) wordspotting Figure-of-Merit defined as "an upper-bound estimate on word spotting accuracy averaged over 1 to 10 false alarms per hour." It is calculated using the following equation:

$$\text{FOM} = \frac{1}{10T}(p_1 + p_2 + ... + p_N + ap_{N+1})$$

where $a = 10T - N$ is a factor that interpolates to 10 false alarms per hour and T is the total hours of duration of the test data. To perform this calculation, first all of the spotted words (melodies) are ranked in score order. Then, the percentage of true hits p_i found before the i'th false alarm is calculated for $i = 1, ..., N+1$ where N is the first integer greater than or equal to $10T - 0.5$. In the case of the experiments described here, $T = 0.5$ hours of recorded musical data, so:

$$\text{FOM} = \frac{1}{5}(p_1 + p_2 + p_3 + p_4 + p_5)$$

Thematic Extractor

Colin Meek
University of Michigan
1101 Beal Avenue
Ann Arbor MI 48104
1-734-763-1561
meek@umich.edu

William P. Birmingham
University of Michigan
1101 Beal Avenue
Ann Arbor MI 48104
1-734-936-1590
wpb@umich.edu

ABSTRACT

We have created a system that identifies musical *keywords* or themes. The system searches for all patterns composed of melodic (intervallic for our purposes) repetition in a piece. This process generally uncovers a large number of patterns, many of which are either uninteresting or only superficially important. Filters reduce the number or prevalence, or both, of such patterns. Patterns are then rated according to perceptually significant characteristics. The top-ranked patterns correspond to important thematic or motivic musical content, as has been verified by comparisons with published musical thematic catalogs. The system operates robustly across a broad range of styles, and relies on no meta-data on its input, allowing it to independently and efficiently catalog multimedia data.

1. INTRODUCTION

We are interested in extracting the major themes from a musical piece: recognizing patterns and motives in the music that a human listener would most likely retain. *Thematic extraction*, as we term it, has interested musician and AI researchers for years. Music librarians and music theorists create thematic indices (e.g., Köchel catalog [1]) to catalog the works of a composer or performer. Moreover, musicians often use thematic indices (e.g., Barlow's *A Dictionary of Musical Themes* [2]) when searching for pieces (e.g., a musician may remember the major theme, and then use the index to find the name or composer of that work). These indices are constructed from themes that are manually extracted by trained music theorists. Construction of these indices is time consuming and requires specialized expertise. Figure 1 shows a simple example.

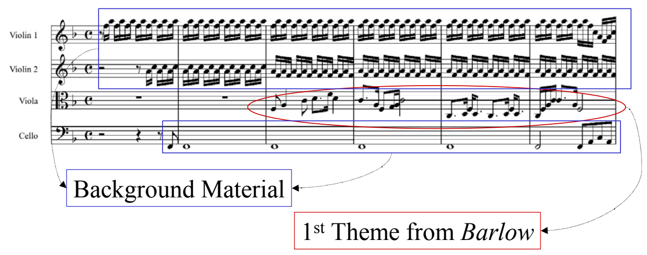


Figure 1: Sample Thematic Extraction from opening of Dvorak's *American Quartet*

Theme extraction using computers has proven very difficult. The best known methods require some 'hand tweaking' [3] to at least provide clues about what a theme may be, or generate thematic listings based solely on repetition and string length [4]. Yet, automatically extracting major themes is an extremely important problem to solve. In addition to aiding music librarians and archivists, exploiting musical themes is key to developing efficient music-retrieval systems. The reasons for this are twofold. First, it appears that themes are a highly attractive way to query a music-retrieval system. Second, because themes are much smaller and less redundant than full pieces, by searching a database of themes, we simultaneously get faster retrieval (by searching a smaller space) and get increased relevancy. Relevancy is increased as only crucial elements, variously named motives, themes, melodies or hooks, are searched, thus reducing the chance that less important, but commonly occurring, elements will fool the system.

There are many aspects to music, such as melody, structure and harmony, each of which may affect what we perceive as major thematic material. Extracting themes is a difficult problem for many reasons. Among these are the following:

- The major themes may occur anywhere in a piece. Thus, one cannot simply scan a specific section of piece (e.g., the beginning).
- The major themes may be carried by any voice. For example, in Figure 2, the viola, the third lowest voice, carries the principal theme. Thus, one cannot simply "listen" to the upper voices.
- There are highly redundant elements that may appear as themes, but should be filtered out. For example, scales are ubiquitous, but rarely constitute a theme. Thus, the relative frequency of a series of notes is not sufficient to make it a theme.

In this paper, we introduce an algorithm, Melodic Motive Extractor (MME), that automatically extracts themes from a piece of music, where music is in a note representation. Pitch and duration information are given; metrical and key information is not required.

MME exploits redundancy that is found in music: composers will repeat important thematic material. Thus, by breaking a piece into note sequences and seeing how often sequences repeat, we identify the themes. Breaking up involves examining all note sequence lengths of two to some constant. Moreover, because of the problems listed earlier, we must examine the entire piece and all voices. This leads to very large numbers of sequences (roughly 7000 sequences on average, after filtering), thus we must use a very efficient algorithm to compare these sequences.

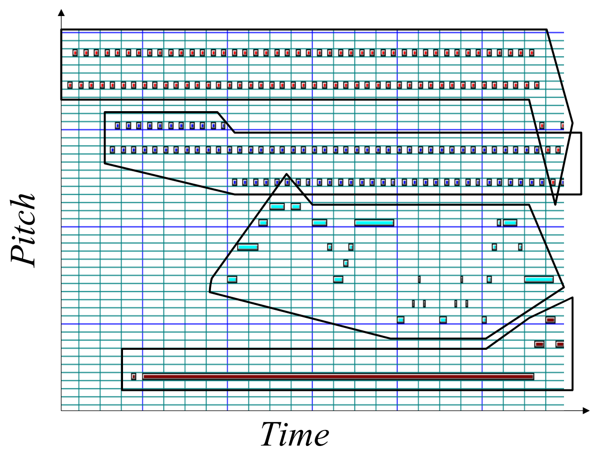


Figure 2: Opening Phrase of Dvorak's "*American*" Quartet

Once repeating sequences have been identified, we must further characterize them with respect to various perceptually important features in order to evaluate if the sequence is a theme. Learning how best to weight these features for the thematic value function is an important part of our work. For example, we have found that the frequency of a pattern is a stronger indication of thematic importance than is the register in which the pattern occurs (a counterintuitive finding). We implement hill-climbing techniques to learn weights across features. The resulting evaluation function then rates the sequences.

Across a corpus of 60 works, drawn from the Baroque, classical, romantic and contemporary periods, MME extracts sections identified by Barlow as "1st themes" over 98% of the time.

1.1 Problem Formulation

Input to MME is a set of note events making up a musical composition $N = \{n_1, n_2 ... n_3\}$. A note event is a triple consisting of an onset time, an offset time and a pitch (in MIDI note numbers, where 60 = 'Middle C' and the resolution is the semi-tone): n_i = <*onset*, *offset*, *pitch*>. We note that several other valid representations of a musical composition exist, taking into account amplitude, timbre, meter and expression markings among others [6]. We limit the domain because pitch is reliably and consistently stored in MIDI files--the most easily accessible electronic representation for music--and because we are interested primarily in voice contour as a measure of redundancy.

The goal of MME is to identify patterns and rank them according to their perceptual importance as a theme. We readily acknowledge that there may, in some cases, be disagreement among listener about what constitutes a theme in a piece of music; however, , we note that t published thematic catalogs represent common convention. These catalogs thereby provide a concrete measure by which the system can be evaluated..

2. Algorithm

In this section, we describe the operation of MME. This includes identifying patterns and computing pattern characteristics, such that "interesting" patterns can be identified. MME's main processing steps are the following:

1. Input
2. Register
3. Stream segregation
4. Filter top voice
5. Calculate event transitions
6. Generate event keys
7. Identify and filter patterns
8. Frequency
9. Compute other pattern features
10. Rate patterns
11. Return results

2.1 Input

MME generally takes as input MIDI files, which are translated into lists of note events in the described format. Information is also maintained about the channel and track of each event, which is used to separate events into streams.

2.2 Register

Register is an important indicator of perceptual prominence [10]: we listen for higher pitched material. For the purposes of MME, we define register in terms of the voicing, so that for a set of *n* concurrent note events, the event with the highest pitch is assigned a register of 1, and the event with the lowest pitch is assigned a register value of *n*. For consistency across a piece, we map register values to the range [0,1] for any set of concurrent events, such that 0 indicates the highest pitch, 1 the lowest.

```
Given the input set of events N[]:
 1.  Sort(N, onset[N])
 2.  ActiveList ← NULL
 3.  index ← 0
 4.  while index < n
 4.     onset ← Onset[N[index]]
 5.     • remove all inactive events
 6.     Remove(ActiveList, Offset[N] • Onset)
 7.     • add all events with the same onset
 8.     while index < n - 1 and Onset[N[index]] = onset
 9.        Register[N[index]] ← 0
10.           add N[index] to ActiveList
11.           increment index
12.        • update Register value of active events
13.        Sort(ActiveList, Pitch[ActiveList])
14.        n ← Size[ActiveList] - 1
15.        for j ← 0 to n
16.           register ← n - j / n
17.           if register > Register[ActiveList[j]]
18.              Register[ActiveList[j]] ← register
```

Algorithm 1: Calculating Register

Table 1: Register values at each iteration of register algorithm

Adding	e_0	e_1	e_2	e_3	e_4	e_5	e_6	e_7	ActiveList
e_0	0								$\{e_0\}$
e_1	1	0							$\{e_0, e_1\}$
e_2	1	0	1/2						$\{e_0, e_1, e_2\}$
e_3	1	0	1	0					$\{e_2, e_3\}$
e_4, e_5	1	0	1	2/3	1/3	0			$\{e_2, e_3, e_4, e_5\}$
e_6, e_7	1	0	1	2/3	1/3	0	1/2	1	$\{e_4, e_6, e_7\}$

We need to define the notion of concurrency more precisely. Two events with intervals $I_1 = [s_1, e_1]$ and $I_2 = [s_2, e_2]$ are considered concurrent if there exists an common interval $I_c = [s_c, e_c]$ such that $s_c < e_c$ and $I_c \subseteq I_1 \wedge I_c \subseteq I_2$. The simplest way of computing these values is to walk through the event set ordered on onset time, maintaining a list of (notes that are on) events, or events sharing a common interval (see Algorithm 1).

Consider the example piece in Figure 3. The register value assigned to each event $\{e_0 \ldots e_7\}$ at each iteration is shown in Table 1.

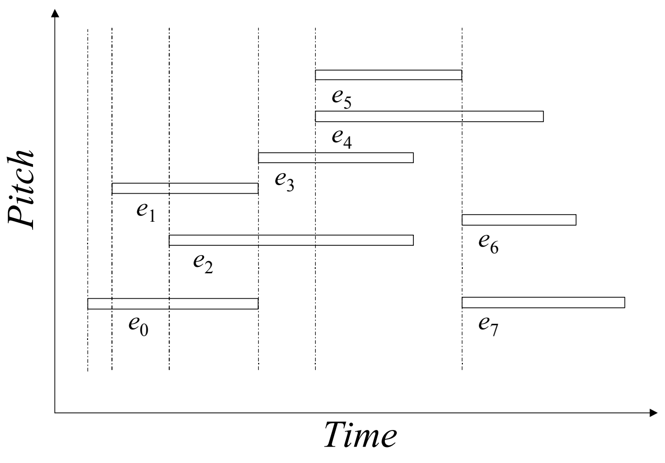


Figure 3: Register, Example Piece

2.3 Stream Segregation and Filtering Top Voice

Generally, the individual channels of a MIDI file correspond to the different instruments or voices of a piece. Figure 2 shows a relatively straightforward example of segmentation, from the opening of Dvorak's "*American*" *Quartet*, where four voices are present. In cases where several concurrent voices are present in one instrument, for example in piano music, we deal with only the top sounding voice. This is clearly a restriction, albeit a reasonable one, as certain events are disregarded. This restriction is necessary . Although existing analysis tools, such as MELISMA [7], perform stream segregation on abstracted music, i.e., note-event representation, they have trouble with overlapping voices [8], as seen between the middle voices in Figure 2.

Identifying the top sounding voice is not as straightforward as it may appear. Some MIDI scores contain overlapping consecutive events within a single voice. To avoid filtering out such notes, we employ an algorithm similar to the register algorithm (see Algorithm 1), wherein events are removed from the active list for their particular channel some ratio (0.5) of their duration from their onset, and as such avoid being falsely labeled as "lower-sounding" notes. For instance, an event in the time interval [30, 50] will be removed from the active list when the sweep reaches time 40.

Additionally, when long pauses (greater than some time constant) are found in a stream, the stream is broken at that point. In this manner, we exclude sequences enclosing large stretches of silence from gaining arbitrary advantage from the duration feature.

For the purposes of this paper, we will indicate events using the notation $e_{stream,\ index}$, such that $e_{0,1}$ indicates the second note of the first stream.

2.4 Calculating Transitions

We are primarily concerned with melodic contour as an indicator of redundancy. For our purposes, contour is defined as the sequence of pitch intervals across a sequence of note events in a stream. For instance, the stream consisting of the following event sequence: $e_s = \{<0, 1, 60>, <1, 2, 62>, <2, 3, 64>, <3, 4, 62>, <4, 5, 60>\}$ has contour $c_s = \{+2, +2, -2, -2\}$.

MME considers contour in terms of simple interval, which means that although the sign of an interval (+/-) is considered, octave is not. As such, an interval of +2 is equivalent to an interval of +14 = (+2 + octave = +2 + 12). We normalize each interval corresponding to an event, i.e., the interval between that event and its successor, to the range [-12, 12]:

$$real\_interval_{s,i} = Pitch[e_{s,i+1}] - Pitch[e_{s,i}]$$

$$c_{s,i} = \begin{cases} real\_interval_{s,i}, & \text{if } -12 \le real\_interval_{s,i} \le +12 \\ -\operatorname{mod}_{12} - real\_interval_{s,i}, & \text{if } real\_interval_{s,i} < -12 \\ \operatorname{mod}_{12} real\_interval_{s,i} & \text{o.w.} \end{cases}$$

Another transition measure we employ is known as the Inter-Onset Interval (IOI), used to describe the rhythmic content of a sequence, and the rhythmic consistency of a pattern. This measure ignores the rhythmic articulation of events, but maintains the basic rhythmic information. In the above example, the IOI values are simply {1, 1, 1, 1}:

$$IOI[e_{s,i}] = Onset[e_{s,i+1}] - Onset[e_{s,i}]$$

2.5 Calculating Keys

To efficiently uncover patterns, or repeating sequences, we assign a key k to each event in the piece that uniquely identifies a sequence of m intervals, where m is the maximum pattern length under consideration. Length refers to the number of intervals in a pattern, one less than the number of events. The keys must exhibit the following property:

$$k_{s1,i1}(m) = k_{s2,i2}(m) \leftrightarrow \{c_{s1,i1}, c_{s1,i1+1}, \ldots, c_{s1,i1+m-1}\} = \{c_{s2,i2}, c_{s2,i2+1}, \ldots, c_{s1,i2+m-1}\}$$

Since only 25 distinct simple intervals exist, we can refer to sequences of intervals in radix-26 notation, reserving a digit (0) for the ends of streams. An m-digit radix-26 number, where each digit corresponds to an interval in sequence, thus uniquely identifies that sequence of intervals, and our key values can then be calculated as follows, re-mapping intervals to the range [1, 25]:

$$k_{s,i}(m) = \sum_{j=0}^{m-1} (c_{s,i+j} + 13) * 26^{m-j-1}$$

The following derivations allow us to more efficiently calculate the value of $k_{s,i}$:

Equation 1

$$k_{s,i}(1) = c_{s,i} + 13$$

Equation 2

$$k_{s,i}(n) = \begin{cases} 26 * k_{p,i}(n-1) + k_{p,i+n-1}(1) & \text{if } n \le |c_s| - i \\ k_{s,i}(|c_s| - i) * 26^{n-|c_s|+i} & o.w. \end{cases}$$

The second case of this last equation deals with the situation where no additional information is gained by increasing n, since there are no additional intervals to consider beyond the end of the stream. It is derived from the observation that when $i \ge |c_s|$, $k_{s,i}(1) = 0$, the end of stream zero padding.

By removing the most significant digit of a key $k_{s,i}(n)$, we get the key for the subsequent event $k_{s,i+1}(n-1)$:

Equation 3

$$k_{s,i+1}(n-1) = k_{s,i}(n) - k_{s,i}(1) * 26^{n-1}$$

This in turn allows us to calculate the subsequent key value in constant time, using Equation 2.

Using Equation 1 and Equation 2, we can calculate the key if the first event in a stream in linear time with respect to the maximum pattern length, or the stream length, whichever is smaller (this is essentially an application of *Horner's Rule* [9]). Equation 3 allows us to calculate the key of each subsequent event in constant time (as with the *Rabin-Karp* algorithm [9]). As such, the overall complexity for calculating keys is $\Theta(n)$ with respect to the number of events.

Consider the following simple example for $m = 4$, a single phrase from Mozart's *Symphony no. 40*: $c_0 = \{-1, 0, +1, -1, 0, +1, -1, 0 +8\}$.

First we calculate the key value for the first event ($k_{0,0}(4)$), using Equation 1 and Equation 2 recursively:

$$\begin{aligned} k_{0,0}(4) &= 26 * k_{0,0}(3) + k_{0,3}(1) \\ &= 26 * (26 * k_{0,0}(2) + k_{0,2}(1)) + 12 \\ &= 26 * (26 * (26 * k_{0,0}(1) + k_{0,1}(1)) + 14) + 12 \\ &= 26 * (26 * (26 * 12 + 13) + 14) + 12 \\ &= 220076 \end{aligned}$$

Then we calculate the remaining key values:

$$k_{0,1}(3) = k_{0,0}(4) - k_{0,0}(1) * 26^3 = 9164 \text{ (Equation 3)}$$

$$k_{0,1}(4) = 26 * k_{0,1}(3) + k_{0,4}(1) = 238277 \text{ (Equation 2)}$$

Using the same procedure, we generate the remaining key values:

$k_{0,2}(4) = 254528$ $k_{0,3}(4) = 220076$ $k_{0,4}(4) = 238277$ $k_{0,5}(4) = 254535$
$k_{0,6}(4) = 220246$ $k_{0,7}(4) = 242684$ $k_{0,8}(4) = 369096$ $k_{0,9}(4) = 0$

2.6 Identifying and Filtering Patterns

We employ one final derivation on k for the pattern identification:

Equation 4

$$\forall n, 0 < n \le m : k_{s,i}(n) = \left\lfloor \frac{k_{s,i}(m)}{26^{m-n}} \right\rfloor$$

Events are then sorted on key so that pattern occurrences are adjacent in the ordering. We make a pass through the list for pattern lengths from $n = [m \ldots 2]$, resulting in a set of patterns, ordered from longest to shortest. This procedure is straightforward: during each pass through the list, we group together keys for which the value of $k(n)$ - calculated using Equation 4 – is the same. Such groups are consecutive in the sorted list. Occurrences of a given pattern are then ordered according to their onset time, a property necessary for later operations.

Continuing with the Mozart example, sorting the keys we get: $\{k_{0,9}, k_{0,0}, k_{0,3}, k_{0,6}, k_{0,1}, k_{0,4}, k_{0,7}, k_{0,2}, k_{0,5}, k_{0,8}\}$.

On our first pass through the list, for $n = 4$, we identify patterns $\{k_{0,0}, k_{0,3}\}$ and $\{k_{0,1}, k_{0,4}\}$, since there keys are identical. During the second pass, for $n = 3$, we identify patterns $\{k_{0,0}, k_{0,3}\}$, $\{k_{0,1}, k_{0,4}\}$ and $\{k_{0,2}, k_{0,5}\}$, noting that $k_{0,2}/26^{4-3} = k_{0,5}/26^{4-3}$ (which by Equation 4 indicates that a pattern of length three exists.) Similarly, we identify the following patterns for $n = 2$: $\{k_{0,0}, k_{0,3}, k_{0,6}\}$, $\{k_{0,1}, k_{0,4}\}$ and $\{k_{0,2}, k_{0,5}\}$. The patterns are shown in Table 2.

Table 2: Patterns in opening phrase of Mozart's *Symphony no. 40*

Pattern	Occurrences at	Characteristic interval sequence
P_0	$e_{0,0}, e_{0,3}$	{-1, 0, +1, -1}
P_1	$e_{0,1}, e_{0,4}$	{0, +1, -1, 0}
P_2	$e_{0,0}, e_{0,3}$	{-1, 0, +1}
P_3	$e_{0,1}, e_{0,4}$	{0, +1, -1}
P_4	$e_{0,2}, e_{0,5}$	{+1, -1, 0}
P_5	$e_{0,0}, e_{0,3}, e_{0,6}$	{-1, 0}
P_6	$e_{0,1}, e_{0,4}$	{0, +1}
P_7	$e_{0,2}, e_{0,5}$	{+1, -1}

We associate a vector of parameter values $V_i = < v_1, v_2, \ldots, v_n >$ and a set of occurrences to each pattern. Length, v_{length}, is one such parameter. The assumption was made that longer patterns are more significant, simply because they are less likely to occur by chance.

As patterns are identified, they are filtered according to several criteria. Since zero padding is used at the ends of streams, it must be verified that a sequence does not overrun the end of a stream, which frequently happens since all streams end with the same zero-padding. Two other filtering criteria are considered as well: intervallic variety, and doublings.

2.6.1 Intervallic Variety

Early experiments with this system indicated that sequences of repetitive, simple pitch-interval patterns dominate given the parameters outlined thus far. For instance, in the Dvorak example (see Figure 2) the melody is contained in the second voice from the bottom, but highly consistent, redundant figurations exist in the upper two voices. Intervallic variety provides a means of distinguishing these two types of line, and tends to favor important thematic material since that material is often more varied in terms of contour.

Given that intervallic variety is a useful indicator of how interesting a particular passage appears, we count the number of distinct intervals observed within a pattern, not including 0. We calculate two interval counts: one in which intervals of $+n$ or $-n$ are considered equivalent, the other taking into account interval direction. Considering the entire Mozart example, which is indeed a pattern within the context of the whole piece, there are three distinct directed intervals, -1, +1 and 8, and two distinct undirected intervals, 1 and 8.

At this stage, we filter out all patterns whose characteristic interval sequence has below certain minimum values for these interval counts. In addition, interval counts are maintained for each pattern.

2.6.2 Doublings

Doublings are a special case in MME. A doubled passage occurs where two or more voices simultaneously play the same line. In such instances, only one of the simultaneous occurrences is retained for a particular pattern, the highest sounding to maintain the accuracy of the register measure.

We must provide a definition of simultaneity to clearly describe this parameter. To provide for inexact performance, we allow for a looser definition: two occurrences o_a and o_b, with initial events $e_{s1,i1}$ and $e_{s2,i2}$ respectively, and length n, are considered simultaneous if and only if $\forall j, 0 \le j \le n : e_{s1,i1+j}$ overlaps $e_{s2,i2+j}$.

Two events are in turn considered overlapping if they strictly intersect. It is easier to check for the non-intersecting relations -- using the conventions and notations of Beek [11] -- $e_{s1,i1}$ before (*b*) $e_{s2,i2}$ or the inverse (*bi*) (see Algorithm 2):

We check each occurrence of a pattern against every other occurrence. Note that since occurrences are sorted on onset, we know that if o_i and o_j are not doublings, where $j > i$, o_i cannot double o_k for all $k > j$. This provides a way of curtailing searches for doublings in our algorithm, and provides significant performance gains (experimentally, a tenfold improvement). This is because partial doublings rarely occur, where only some subset of corresponding intervals is simultaneous.

```
Given a pattern P with n occurrences in O[] and length l
 1.  for i ← 0 to n - 2
 2.     for j ← i + 1 to n - 1
 3.        if ~Remove[O[i]] and ~Remove[O[j]]
 4.           Simultaneous = true
 5.           for k ← 0 to l
 6.              if ~Intersects(e_{Stream[O[i]],Index[O[i]]}, e_{Stream[O[j]],Index[O[j]]}) then
 7.                 Simultaneous ← false
 8.                 k ← l + 1
 9.           if Simultaneous then
10.              if Pitch(e_{Stream[O[i]],Index[O[i]]}) > Pitch(e_{Stream[O[j]],Index[O[j]]})
11.                 Remove[j] ← true
12.                 Doubled[i] ← true
13.              else
14.                 Remove[i] ← true
15.                 Doubled[j] ← true
16.           else
17.              j ← n
18.  Remove(O, Remove[O])
```

Algorithm 2: Filter Doublings

This doubling filtering occurs before other computations, and thus influences frequency. We, however, retain the doubling information (Lines 12 and 15, Algorithm 2), as it is a musical emphasis technique.

If after filtering doublings less than two occurrences remain, the pattern is no longer considered a pattern, and removed from consideration. Doublings serve to reinforce a voice, and as such do not constitute repetition.

2.7 Frequency

Frequency of occurrence is one of the principal parameters considered by MME in establishing pattern importance. All other things being equal, higher occurrence frequency is considered an indicator of higher importance. Our definition of frequency is complicated by the inclusion of partial pattern occurrences. For a particular pattern, characterized by the interval sequence $\{C_0, C_1, ..., C_{v_{length}-1}\}$, the frequency of occurrences is defined as follows:

$$\frac{\displaystyle\sum_{l=v_{length}}^{2} \sum_{j=0}^{v_{length}-1} \text{non-redundant and un-filtered occurrences of } \{C_j, C_{j+1}, \ldots, C_{j+l-1}\} * l}{v_{length}}$$

An occurrence is considered non-redundant if it has not already been counted, or partially counted (i.e., it contains part of another sub-sequence that is longer or precedes it.) Consider the piece consisting of the following interval sequence, in the stream e_0: $c_0 = \{-2,+2,-2,+2,-5,+5,-2,+2,-2,+2,-5,+5,-2,+2,-2,+2\}$, and the pattern $\{-2,+2,-2,+2,-5\}$. Clearly, there are two complete occurrences at $e_{0,0}$ and $e_{0,6}$, but also a partial occurrence of length four at $e_{0,12}$. The frequency is then 2.8 for this pattern.

To efficiently calculate frequency, we first construct a set of pattern occurrence lattices, on the following binary occurrence relation $\prec$:

Given occurrences o_1 and o_2 characterized by event sequences E_1 and E_2, $o_1 \prec o_2 \Leftrightarrow E_1 \subset E_2$. In other words, each occurrence in the lattice covers all patterns occurrences containing a subsequence of that occurrence.

As such, in establishing frequency, we need consider only those patterns covered by occurrences of P in the lattices. Two properties of our data facilitate this construction:

1. The pattern identification procedure adds patterns in reverse order of pattern length.

2. For any pattern occurrence of length $n > 2$, there are at most two occurrences of length $n - 1$, one sharing the same initial event, one sharing the same final event. If one of these two child occurrences does not exist, it is due to the filtering described above. Because of the nature of the filtering, no patterns of length less than $n - 1$ will be covered by the occurrence in these instances, so we need only generate links to occurrences of length $n - 1$ in the lattices. The branching factor is thus limited to two.

The lattice is described as follows: given a node representing an occurrence of a pattern o with length l, the left child is an occurrence of length $l - 1$ beginning at the same event. The right child is an occurrence of length $l - 1$ beginning at the following event. The left parent is an occurrence of length $l + 1$ beginning at the previous event, and the right parent is an occurrence of length $l + 1$ beginning at the same event. Consider the patterns the Mozart excerpt (see Table 2): P_0's first occurrence, with length 4 and at $e_{0,0}$, directly covers two other occurrences of length 3: P_2's first occurrence at $e_{0,0}$ (left child) and P_3's first occurrence at $e_{0,1}$ (right child). The full lattice is shown in Figure 4, where each occurrence in the lattice is labeled with its respective pattern.

Lattices are constructed from the top down, since patterns are added in reverse order of length. Each note event in the piece contains a pointer to an occurrence, such that as occurrences are added, lattice links can be built in constant time (see Algorithm 3).

Consider the patterns identified in the Mozart example (Table 2), from which we build the lattice in Figure 1. When the first occurrence of pattern P_4 is inserted, *o_left* = the first occurrence of P_3, and *o_right* = null. Since P_3 has the same length as P_4, we check the right parent of the *o_right*, and update the link between those occurrences of P_1 and P_4. Other links are updated in a more straightforward manner.

```
Given a series of n patterns P[]
 1.  for i ← 0 to n - 1
 2.     O ← Occurrences[P[i]]
 3.     for j ← 0 to Size[O]
 4.        • occurrence pointed to by the first event of O
 5.        o_right ← Occurrence[$e_{Stream[O[j]]}$, $e_{Index[O[j]]}$]
 6.        • occurrence pointed to by the preceding event
 7.        if Index[O[j]] = 0
 8.           o_left ← null
 9.        else
10.           o_left ← Occurrence[$e_{Stream[O[j]]}$, $e_{Index[O[j]]-1}$]
11.        • we consider three cases for the value of o_left
12.        if o_left = null
              • we learn nothing about the lattice
13.        else if Length[o_left] > Length[O[j]]
14.           Right_Child[o_left] ← O[j]
15.        else
16.           Right_Child[Right_Parent[o_left]] ← O[j]
17.        • we consider two cases for the value of o_right
18.        if o_right = null
              • we learn nothing about the lattice
19.        else
20.           Right_Parent[O[j]] ← o_right • used in line 16
21.           Left_Child[o_right] ← O[j]
22.           Occurrence[$e_{Stream[O[j]]}$, $e_{Index[O[j]]}$] ← O[j]
```

Algorithm 3: Lattice Construction

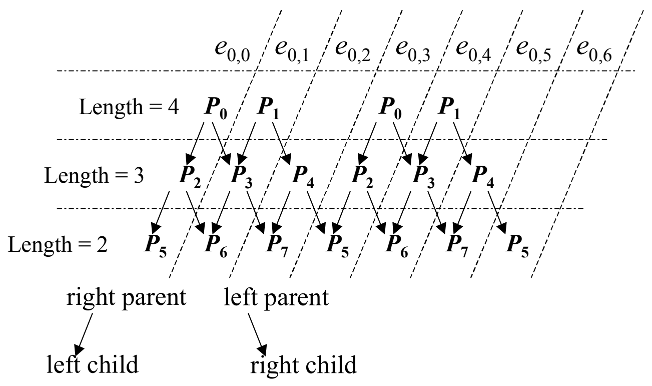


Figure 4: Lattice for the First Phrase of Mozart's *Symphony no. 40*

From this lattice, we easily identify non-redundant partial occurrences of patterns. For each pattern, we perform a breadth-first traversal from its occurrences in the lattice, marking patterns and events as they are counted so that none are included twice. Simultaneously, the number of doubled occurrences is counted. In this manner, we calculate the value of the $v_{doublings}$ and $v_{frequency}$ features for each pattern (see Algorithm 4).

Take for instance pattern P_2 in the Mozart example. By breadth-first traversal, starting from either occurrence of P_2, the following elements are added to Q: P_2, P_5 and P_6. First, we add the two occurrences of P_2, tagging events $e_{0,0}$, $e_{0,1}$, … , $e_{0,5}$, and setting $v_{frequency} \leftarrow 6$. The first two occurrences of P_5 contain tagged events, so we reject them, but the third occurrence at $e_{0,6}$ is un-tagged, so we tag $e_{0,6}$, $e_{0,7}$, $e_{0,8}$ and set $v_{frequency} \leftarrow 6 + 2$. All occurrences of P_6 are tagged, so the frequency of P_2 is equal to 8 / 3.

```
Given a pattern P:
 1.  id ← unique identifier for pattern
 2.  Tag[P] ← id
 3.  push(Q, P)
 4.  while ~empty(Q)
 5.     • add chbildren to Queue (DFS)
 6.     pop(Q, p)
 7.     o_left ← Left_Child[Occurrences[p][0]]
 8.     o_right ← Right_Child[Occurrences[p][0]]
 9.     if o_left ~= null and Tag[Pattern][o_left] ~= id
10.        Tag[Pattern[o_left]] ← id
11.        push(Q, Pattern[o_left]])
12.     if o_right ~= null and Tag[Pattern][o_right] ~= id
13.        Tag[Pattern[o_right]] ← id
14.        push(Q, Pattern[o_left]])
15.     • count non-redundant occurrences of p
16.     for i ← 0 to Size[Occurrences[p]] - 1
17.        if events in Occurrences[p][i] have Tag ~= id [1]
18.           set Tag ←
                 id for all events in Occurrences[p][i]
19.           $v_{frequency}$[P] ← $v_{frequency}$[P] + Length[Occurrences[p][i]]
20.           $v_{doublings}$[P] ← $v_{doublings}$[P] + Length[Occurrences[p][i]]
21.  $v_{frequency}$[P] ← $v_{frequency}$[P] / $v_{length}$[P]
22.  $v_{doublings}$[P] ← $v_{doublings}$[P] / $v_{length}$[P]
```

Algorithm 4: Calculating Frequency

2.8 Other Pattern Features

Several pattern features have been described thus far: $v_{interval\_count}$, $v_{absolute\_interval\_count}$, v_{length}, $v_{frequency}$ and $v_{doublings}$. In addition, we consider pattern duration ($v_{duration}$), rhythmic consistency (v_{rhythm}), position in the piece ($v_{position}$), and register (calculated from event register, $v_{register}$).

2.8.1 Duration

The duration parameter is an indicator of the temporal interval over which occurrences of a pattern exist. For a given occurrence o, with initial event $e_{s1,i1}$ and final event $e_{s2,i2}$, the duration $D(o) = Offset[e_{s2,i2}] - Onset[e_{s1,i1}]$. For a pattern P, with occurrences o_0, o_1, ... , o_{n-1}, the distance parameter is calculated to be the average duration of all occurrences:

$$v_{duration} = \frac{\sum_{i=0}^{n-1} D(o_i)}{n}$$

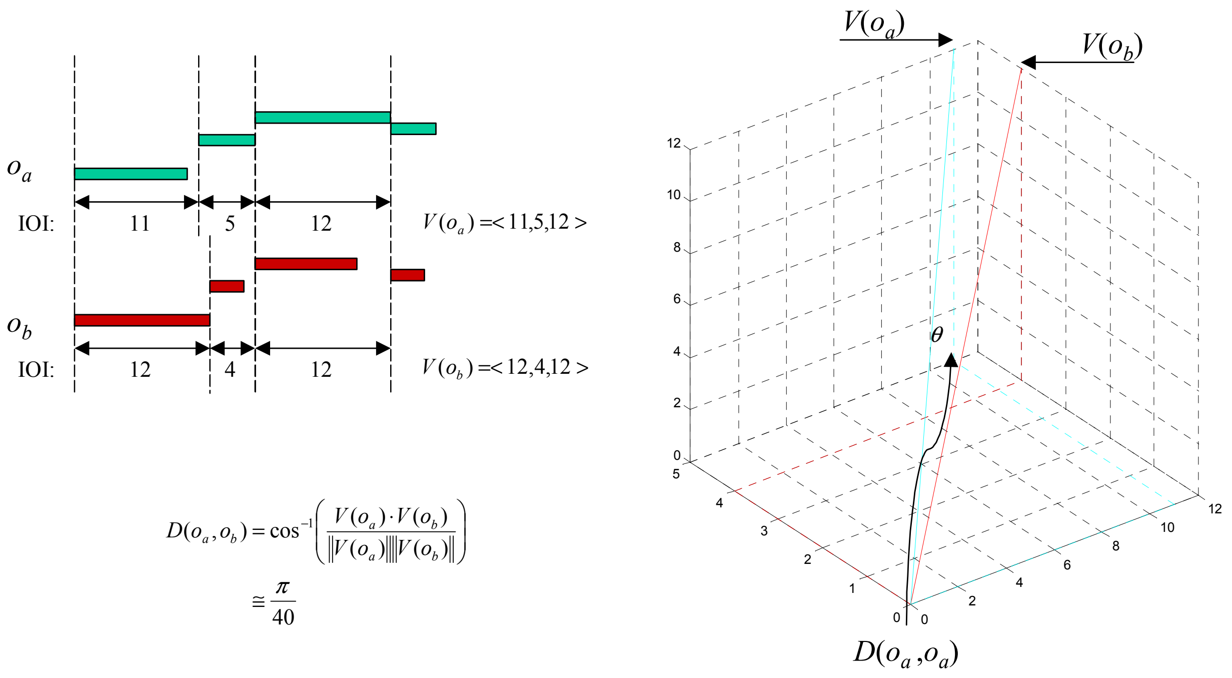
2.8.2 Rhythmic Consistency

We calculate the rhythmic distance between a pair of occurrences as the angle difference between the vectors built from the IOI values of each occurrence. For occurrence o, with events E_0, E_1, … , $E_{length\ -\ 1}$,[2] the IOI vector is $V(o) = < IOI[E_0], IOI[E_1], ..., IOI[E_{length-1}] >$. The rhythmic distance between a pair of occurrences o_a and o_b is then the angle distance between the vectors $V(o_a)$ and $V(o_b)$:

$$D(o_a, o_b) = \cos^{-1}\left(\frac{V(o_a) \cdot V(o_b)}{\|V(o_a)\| \|V(o_b)\|}\right)$$

[1] If the first and last events of *Occurrences*[*p*][*i*] are un-tagged, then we can assume the occurrence has not been counted even in part, since previously considered occurrences are necessarily of greater or equal length. As such, only the first and last events are examined here.

[2] We use the notation E_j to refer to an arbitrary event $e_{s,i}$. Note that E_j and E_{j+1} refer to consecutive events $e_{s,i}$ and $e_{s,i+1}$.



$$D(o_a, o_b) = \cos^{-1}\left(\frac{V(o_a)\cdot V(o_b)}{\|V(o_a)\|\|V(o_b)\|}\right) \cong \frac{\pi}{40}$$

Figure 5: Rhythmic Distance Measure

A 3-dimensional example of the rhythmic distance calculation between two occurrences o_a and o_b is shown in

Figure 5.

We take the average of the distances between all occurrence (o_0, o_1, ... , o_{n-1}) pairs for a pattern P to calculate its rhythmic consistency:

$$v_{rhythm} = \frac{\sum_{i=0}^{n-2}\sum_{j=i+1}^{n-1} D(V(o_i), V(o_j))}{\frac{n(n-1)}{2}}$$

This value is a measure of how similar different occurrences are with respect to rhythm. Notice that two occurrences with the same notated rhythm presented at different tempi have a distance of 0. Consider the case where o_a has k times the tempo of o_b. In this case, $V(o_b) = kV(o_a)$, and $D(V(o_a), V(o_b)) = D(V(o_a), kV(o_a)) = 0$.

Occurrences with similar rhythmic profiles have low distance, so this approach is robust with respect to performance and compositional variation. For instance, in the *Well-Tempered Clavier* Bach often repeats fugue subjects at half speed. The rhythm vectors for the main subject statement and the subsequent stretched statement will thus have the same angle, and a distance of zero. Similarly, if two presentations of a theme have slightly different rhythmic inflections, their IOI vectors will nonetheless be quite similar.

2.8.3 Position

Noting that significant themes are sometimes introduced near the start of a piece, we also characterize patterns according to the onset time of their first occurrence (o). Note that occurrences are sorted according to *Onset* as patterns are identified, so the first occurrence is also the earliest occurrence:

$$v_{position} = Onset[e_{Stream[o], Index[o]}]$$

2.8.4 Register

Given the register values calculated for note events, the register value for a pattern P with occurrences o_0, o_1, ... , o_{n-1}, is equal to the average register of all events contained in those occurrences:

$$v_{register} = \frac{\sum_{i=0}^{n-1}\sum_{j=0}^{v_{length}} Register[e_{Phrase[o_i], Index[o_i]+j}]}{n*(v_{length}+1)}$$

2.9 Rating Patterns

For each pattern P, we have calculated several feature values. We are interested in comparing the importance of these patterns, and a convenient means of doing this is to calculate percentile values for each parameter in each pattern, corresponding to the percentage of patterns over which a given pattern is considered stronger for a particular feature. These percentile values are stored in a feature vector:

$$F[P] =< p_{Length}, p_{interval\_count}, \ldots, p_{register} >$$

We define stronger as either less than or greater than depending on the feature. Higher values are considered desirable for length, duration, interval counts, doublings and frequency; lower values are desirable for rhythmic consistency, pattern position and register.

The rating of a pattern P, given some weighting of features W, is:

$$Rating[P] \leftarrow W \cdot F[P]$$

2.10 Returning Results

Patterns are then sorted according to their *Rating* field. This sorted list is scanned from the highest to the lowest rated pattern until some pre-specified number (k) of note events has been returned. Often, MME will rate a sub-sequence of an important theme highly, but not the actual theme, owing to the fact that parts of a theme are more faithfully repeated than are others. As such, MME will return an occurrence of a pattern with an added margin on either end, corresponding to some ratio *g* of the occurrences duration, and some ratio of the number of note events *h*, whichever ratio yields the tightest bound.

In order to return a high number of patterns within *k* events, we use a greedy algorithm to choose occurrences of patterns when they are added: whichever occurrence adds the least number of events is used.

Output from MME is a MIDI file consisting of a single channel of monophonic (single voice) note events, corresponding to important thematic material in the input piece.

3. Results

A set of 60 pieces from the Baroque, Classical, Romantic, Impressionistic and 20th Century were used to train and test the software. Bach, Mozart, Beethoven, Brahms, Schubert, Mendelssohn, Dvorak, Smetana, Debussy, Bartok and Stravinsky are represented, in chamber, orchestral and solo piano works.

A few details of MME's configuration should be mentioned: the intervallic variety filter required a minimum of at least zero distinct intervals, and two distinct absolute intervals. Maximum pattern length is set to 12 transitions, and streams are broken with silences longer than one and a half seconds. For the sake of result output and training, there is a margin of 0.5 on both ends for both events and duration. Up to 240 note events are returned for each piece, as compared with an average of over 8500 notes per piece originally. We employ a hill-climbing algorithm to discover good values for *W*.

3.1 Preliminary Results

Given even feature weighting, the primary theme was returned in 51 of the 60 pieces. Learning weights *W across* this entire set, and testing across the same set, the primary theme was returned on 60 of the 60 pieces. These results are presented only to provide context for later results, and to provide some indication of the importance of learning appropriate weights.

3.2 Training Trials

We performed 30 trials, randomly selecting a 30-piece training set for each trial. During each trial, the hill-climbing algorithm was permitted 50 random restarts. These weights were then evaluated against the test set, consisting of the remaining 30 pieces. In two trials, MME identified 28 of the 30 primary themes, in seven trials 29 out of 30, and in 21 trials 30 out of 30, or on average roughly 29.6 out of 30, as compared with an expected average of 25.5 out of 30 using even weights (see Figure 6.)

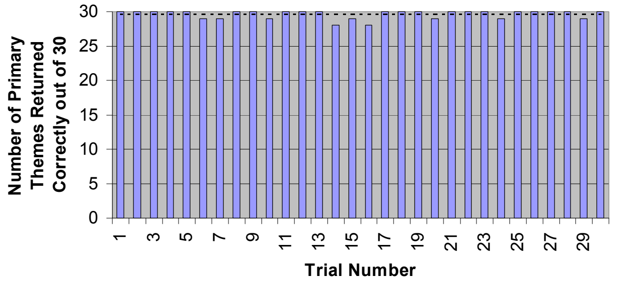


Figure 6: Trial Results

3.2.1 Weights

Examining the weights learned during the trials, we get some idea of the relative importance of the different pattern features examined. The average and median weights across the 30 trials are listed in Table 3.

Of particular interest is the negative weight for absolute interval count. Although our early experiments indicated that filtering patterns with low intervallic variety improves algorithm performance, it appears this parameter does not usefully distinguish the remaining patterns. The weight given to the register feature is perhaps most surprising., as we normally associate important melodies with the highest-sounding voice in a passage. Position is clearly the dominant feature, perhaps owing to our focus on primary themes, which tend to occur near the opening of pieces.

Table 3: Feature Weights

Feature	**Average Weight**	**Median Weight**
absolute interval count	-0.016249988	-0.021642894
register	0.051727027	0.041694308
doublings	0.085212347	0.055842776
interval count	0.121993193	0.110731687
frequency	0.119746216	0.125918866
rhythmic consistency	0.176786867	0.181440092
duration	0.233749767	0.237064805
length	0.344768215	0.274449283
position	0.819313306	0.872008477

3.2.2 Errors

Three pieces were responsible for all errors in MME's output: the first movement of Mozart's *Symphony no. 40*, the second movement of Brahms' *Cello Sonata in E minor*, and Brahms' *Academic Festival Overture*. In the first two cases, the proper theme was only partly returned in some trials, and in the last case, another theme sometimes dominated, albeit one that might be considered subjectively more prevalent than that listed first in Barlow.

Examining the Mozart example (see Figure 7), the opening few notes exhibit a low absolute interval count (only minor seconds, +/- 1), which explains why MME returned only the subsequent portion of the theme in some trials. This piece was included in 20 of the 30 test sets, and in three of those cases, the output was offset as described. In the remaining 17 cases, the proper theme was returned in full.

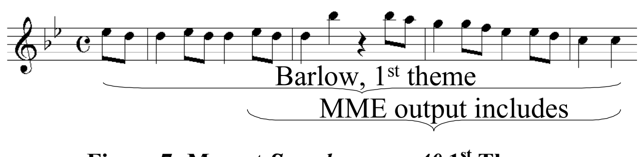


Figure 7: Mozart *Symphony no. 40* 1st Theme

In the case of the cello sonata, MME again selected only a portion of the 1st theme, in four of the 14 trials in which it appeared in the test set. This movement contains a great deal of repetition and variation, on the one hand offering a wealth of potentially important targets, and on the other, confusing the system due to its reliance on exact repetition.

The *Academic Festival Overture* contains a large number of themes, and in every trial, MME returned a fair number of them. The first theme listed in Barlow, however, was returned only six of the 10 times the piece appeared in the test set. In all cases, MME returned another theme (see Figure 8).

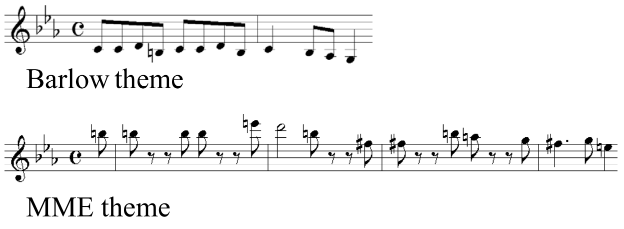


Figure 8: Themes from Brahms' *Academic Festival Orchestra*

3.2.3 Sample of Output

MME's output from Smetana's *The Moldau* (a movement of *My Country*) is shown in Figure 10. The first section *A* contains the 1st theme as indicated by Barlow. Section *F* contains a slight rhythmic variation on the same material, and section *H* presents the subsequent phrase. In addition, section *B* and *D* contain tonal variations of the same material (presented here in the major, whereas the main presentation is in the minor.) To many listeners, these sections sound similar. This highlights a potential weakness of the algorithm: although the correct material is returned, there is redundancy in the output.

3.2.4 Popular Music

MME has been tested on several pieces of popular music, though we present no formal results in the absence of an accepted benchmark for system performance in this genre. Across 20 songs, ranging from the Beatles to Nirvana, an untrained version of MME returned the chorus where applicable, and what we considered to be significant "hooks" in all cases.

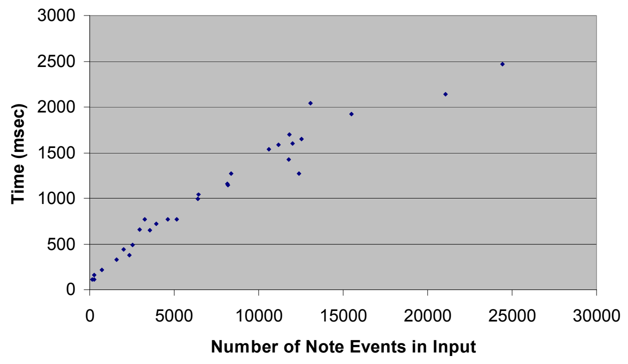


Figure 9

4. Summary

Identifying the major themes in a sophisticated musical work is a difficult task. The results show that MME correctly identifies the major themes in 100% of the test cases (when learning is employed), and identifies 85% of the major when learning is not used.

It is interesting to note that MME contains no deep musical knowledge, such as theory of melody, harmony, or rhythm. Rather, it works entirely from surface features, such as pitch contour, register, and relatively duration. We found, surprisingly, that register is not a good indicator of the thematic importance.

MME is computationally efficient. The system's overall complexity is dominated by the frequency calculation, which in the worst-case operates in $\Theta(m^3n^2)$ time, where m is the maximum pattern length under consideration, and n is the number of note events in the input piece. In practice, however, we observe sub-linear performance (see Figure 9), and reasonable running times on even the largest input pieces.

5. Acknowledgements

We gratefully acknowledge the support of the National Science Foundation under grant IIS-0085945, and The University of Michigan College of Engineering seed grant to the MusEn project. The opinions in this paper are solely those of the authors and do not necessarily reflect the opinions of the funding agencies.

We also thank member of the MusEn research group for their comments. This group includes Greg Wakefield, Roger Dannenberg, Mary Simoni, Mark Bartsch and Bryan Pardo.

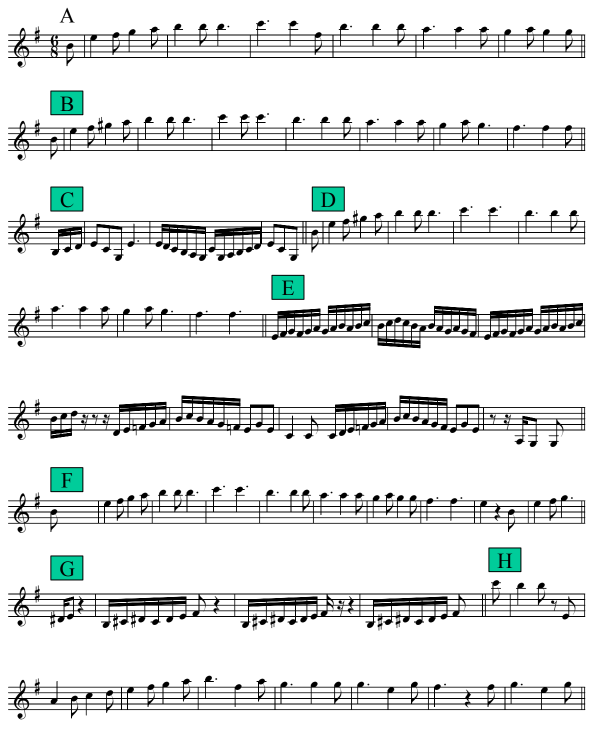


Figure 10: Output from Smetana's *Moldau*

6. References

[1] Ludwig Ritter von Köchel. *Chronologisch-thematisches Verzeichnis sämtlicher Tonwerke Wolfgang Amadé Mozarts; nebst Angabe der verlorengegangenen, ange l'angene, von fremder Hand bearbelteten, zwelfelhaften und unterschobe-nen Kompositionen.* Wiesbaden, Breitkopf & Härtel, 6th edition, 1964.

[2] H. Barlow. *A dictionary of Musical Themes*. Crown Publish-ers, New York, 1975.

[3] David Cope. *Experiments in Musical Intelligence*. A-R Editions, 1996.

[4] Alexandra and Uitdenbogerd. Manipulation of music for melody matching. *ACM Multimedia, Electronic Proceedings*, 1998.

[5] Y.-H. Tseng. Content-based retrieval for music collections. *SIGIR*, 1999.

[6] M. Simoni, C. Rozell, C. Meek, and G. Wakefield. A theoretical framework for electro-acoustic music. *ICMC*, 2000.

[7] David Temperley. Modeling meter and harmony: A preference-rule approach. *Computer Music Journal*, 23.

[8] David Temperley. A model for contrapontal analysis. *unpublished*.

[9] R. L. Rivest T. H. Cormen, C. E. Leiserson. *Introduction to Algorithms*. The MIT Press, Cambridge, Mass., 1999.

[10] A.S. Bregman. *Auditory Scene Analysis: The Perceptual Organization of Sound*. MIT Press.

[11] Peter van Beek. The design and experimental analysis of algorithms for temporal reasoning. *Journal of Artificial Intelligence Research*, 1996.

[12] Joseph G. D'Ambrosio, William O, Birmingham. Preference-Directed Design. *AI EDAM*, 1994. R. L. Rivest T. H. Cormen, C. E. Leiserson. *Introduction to Algorithms*. The MIT Press, Cambridge, Mass., 1999.

Figured Bass and Tonality Recognition

Jerome Barthélemy
Ircam
1 Place Igor Stravinsky
75004 Paris France
33 01 44 78 48 43
jerome.barthelemy@ircam.fr

Alain Bonardi
Ircam
1 Place Igor Stravinsky
75004 Paris France
33 01 44 78 48 43
alain.bonardi@ircam.fr

ABSTRACT

In the course of the WedelMusic project [15], we are currently implementing retrieval engines based on musical content automatically extracted from a musical score. By musical content, we mean not only main melodic motives, but also harmony, or tonality.

In this paper, we first review previous research in the domain of harmonic analysis of tonal music.

We then present a method for automated harmonic analysis of a music score based on the extraction of a figured bass. The figured bass is determined by means of a template-matching algorithm, where templates for chords can be entirely and easily redefined by the end-user. We also address the problem of tonality recognition with a simple algorithm based on the figured bass.

Limitations of the method are discussed. Results are shown and compared to previous research.

Finally, potential uses for Music Information Retrieval are discussed.

KEYWORDS

Music analysis, automatic extraction of musical features, figured bass and tonality recognition.

1. INTRODUCTION

As stated by Ian Bent in his article ”Analysis” of the New Grove’s Dictionary, musical analysis is “the resolution of a musical structure into relatively simpler constituent elements, and the investigation of the functions of these elements within that structure”.

Harmonic analysis is one of the principal means to achieve this goal through the production of a figured bass and the analysis of the function of chords based on the relationship of their root to the main tonality. In this paper, we describe a technique for the automated extraction of the figured bass.

The figured bass is a very old principle, described in several treatises, starting from “Del sonare sopra il basso” by Agazzari (1607).

The aim of the figured bass was, in principle, oriented towards interpretation. Rameau turned it into a genuine theory of tonality with the introduction of the fundamental concept of root. Successive refinements of the theory have been introduced in the 18th, 19th (e.g., by Reicha and Fetis) and 20th (e.g., Schoenberg [10, 11]) centuries. For a general history of the theory of harmony, one can refer to Ian Bent [1] or Jacques Chailley [2]

Several processes can be build on the top of a harmonic reduction

- detection of tonality,
- recognition of cadence,
- detection of similar structures

Following a brief review of systems addressing the problem of tonal and harmonic analysis, we first point out the problems raised by harmonic reduction. We then describe our algorithm, and show its use in some examples. In the subsequent section, we show the application of a simple process of tonality detection on top of harmonic reduction.

The analysis tools that are described here are part of the WedelMusic project, which is funded by the European Commission [15]. Its aim is the development of a system of distribution of music scores over the Internet while preserving the owner’s rights. This project includes a cataloguing system. Indexes are built from such metadata as name of composer, date of composition and so on. Indexes are also built on the basis of musical content, as extracted from the score by analysis tools developed at Ircam. They include such elements as main motives, descriptions of tonalities and their relation with the main tonality, etc.

These elements can be used in a more general strategy of Music Information Retrieval, which would be based not only just on motives, but also on tonal style, harmony and so on.

2. A BRIEF TOUR OF MUSIC ANALYSIS SYSTEMS

In the past, a number of systems have been developed to address the problem of automatic tonal harmonic analysis. Only a few tackle the difficult problem of chord generation - that is, generation of root and encoding of the nature of the chord - directly from the score.

Maxwell’s expert system for harmonic analysis of tonal music [6] is a rule-based system, consisting of more than 50 rules. The first phase performs a reduction of the vertical sonorities of the piece into a chord sequence, by recognizing dissonances and consonances. Maxwell’s complex set of decision rules for consonance and dissonance is difficult to adapt to situations where the notion of dissonance is slightly different, such as music of the 19th century. In addition, as noticed by David Temperley [12], Maxwell's algorithm appears not to be capable of correctly handling situations where notes of the chord are stated in sequence.

Temperley's approach to harmonic analysis [12] consists of a set of preference rules, as described in Lerdahl's and Jackendoff's generative theory of tonal music [5]. As in Maxwell's system, the first phase of Temperley's algorithm leads to the production of the roots of chords. Despite the strongly encouraging results he achieved, the author himself pointed out several problems with the algorithm, especially in the analysis of the Gavotte from the French Suite n° 5 by J.-S. Bach.

Pardo and Birmingham [8] developed HarmAn, a system that partitions tonal music into harmonically significant segments corresponding to single chords. It also tags these segments with the proper chord label. A strength of the system is that it is independent of rhythm. New templates for chords can be introduced, but this requires a rethinking of both the preferences rules and the scoring method for a single template, as stated by the authors. A numerical method is used for scoring elements, with known drawbacks: as stated by Francois Pachet [7], "numerical values are difficult to justify, difficult to maintain, and have poor explanatory capacity". The system works with a MIDI-like representation of notes, and no enharmonic spelling algorithm is implemented. The system thus suffers from a number of drawbacks by not recognizing the difference between, for example, F# and Gb. This will certainly lead to a number of problems in passages belonging to tonalities with several accidentals. In addition, some aggregations used in the late 18th century and in the 19th century, such as the augmented sixth (C – E – G – A#) cannot be distinguished from other chords (in this case, from a seventh on the fifth degree).

Other systems have been developed, which don't address the first difficulty of chord recognition and segmentation of the score.

Winograd [14], in a pioneering work, addressed the analysis of musical scores by using systemic grammars. His method needs a preliminary hand-made conversion of the original score into a score expressed as a sequence of four-part perfect chords. During this operation, ornamental notes, like suspensions, passing notes and the like, are eliminated.

Ulrich [13] developed a process of functional analysis, this term referring to the identification of the function of each chord in a song, and the grouping together of measures that move the tune from one key center to another one. Similarly to Winograd, the input to the program consists of a sequence of chords, each of them consisting of a set of musical notes. An interesting part of the system is an algorithm for detection of keys, described as an "island-growing" mechanism.

François Pachet's approach to computer analysis of jazz chord sequences [7] can be seen as an extension of Ulrich's island growing mechanism, as stated by the author himself. The input of the system is a chord sequence, already explicitly mentioned on the score. The most important improvement to Ulrich's mechanism is that the system outputs a hierarchical description of modulations.

Hoffmann and Birmingham [4] use a constraint satisfaction approach in order to solve the problem of tonal analysis. Similarly to Winograd's method, a preliminary hand-made conversion of the score is necessary.

3. PROBLEM STATEMENT

The issues raised by harmonic reduction are the following:

- ornamental notes and incomplete harmony,
- ambiguities,
- non-regularity of harmony,
- non-universality of harmony rules.

We shall now address each of these points.

3.1 Ornamental notes and incomplete harmonies

In the process of harmonic reduction, some notes are extraneous to the harmony - these are ornamental notes, like appoggiaturas, suspensions, passing notes and so on. On the other hand, harmony is frequently incomplete – i.e., some notes may be missing.

This is illustrated in the following example:

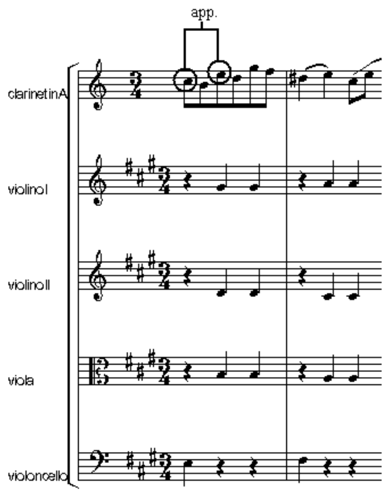


Figure 1. Ornamental notes and incomplete harmonies (Trio of the Clarinet Quintet by Mozart, KV 581)

The circled A and C# (written transposed C and E) in the clarinet part in the first measure are not part of the harmony, thus they are to be considered as ornamental notes.

In the second measure, the harmony – a fifth chord on F# - is never complete anywhere in the measure.

To cope with these problems, we must apply a fundamental rule of analysis, as described by Cook [3] in his treatise: "Essentially there are two analytical acts: the act of omission and the act of relation". In order to decide if a note is an ornamental, we use the rule handling the resolution: in general, resolution of an ornamental note such as a suspension, a passing note, an appoggiatura is performed with a conjunct degree.

In some cases, however, the resolution of an ornamental will be done through disjoint motion: for example, a suspension can be resolved by first playing a harmonic note before playing the resolution. For now, we only apply “natural” resolution, and we will extend our rule to handle more cases.

Another rule for deciding if a note is an ornamental is based on the relative duration (weight) of the note as compared to the other notes of the chord.

3.2 Ambiguities

Some ambiguities have to be resolved, since certain vertical aggregations are not “true harmony”, as shown in the following example:

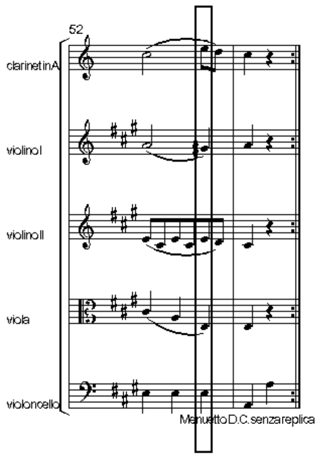


Figure 2. Ambiguous harmony.
(Trio of the Clarinet Quintet by Mozart, KV 581)

The harmony found on the third beat, surrounded here by a rectangle, looks like a sixth. If it is so analysed, its root would then be C#, the third degree of A Major. But this is a nonsense in this context.

3.3 Non-regularity of harmony

In traditional harmony, one cannot assume that the harmonic rhythm is regular. In other words, a harmonic reduction process cannot be based on the assumption that harmony is the same for a beat and for a measure.

3.4 Non-universality of harmony rules

The “theory of harmony” is not to be considered as a genuine, universal and well-defined set of rules. As François Pachet states [7], it is “a theory without theorems”. Rules of harmony have evolved through history. As noticed by Hoffmann [4], “the rules for tonal harmony are not specifically stated, but are conventions drawn from centuries of musical experience”.

To cope with this problem, we must let the user define his own sets of “harmonic rules”, and choose which set of “right” rules to apply.

4. DESCRIPTION OF HARMONIC REDUCTION

The harmonic reduction process includes two main phases:

- a *clusterisation*, which is composed of a first phase of *quantization* of each measure, followed by a vertical aggregation of notes belonging to the same quantization,
- an iterative horizontal aggregation, in which unnecessary notes are eliminated, and successive clusters are merged into chords.

The *quantization* of the measure is simply the computation of the duration of the shortest note in the measure.

For each quantization, we store the result of a vertical aggregation as a *cluster*. We use this term here to designate an aggregate of notes which has not yet reached the status of a chord; it is represented as a list of notes, each of them stored with its diatonic value (pitch, accidental, octave) and its melodic function (interval to the following note in the same voice). The information about the melodic function is used to decide whether the note is an ornamental note or a harmonic note: a note with an interval of a second to the following note is considered to be a possible ornamental note. A note with an interval greater than a second (a third, a fourth and so on) is considered to be a harmonic note.

The iterative horizontal aggregation uses a set of user-defined chord templates, i.e., a list of chords together with their figures. For the analysis presented in this paper, we have used a set of 33 chords, including some seventh and ninth chords, which can be considered as representative of classical harmony as used by composers at the end of the 18th century.

For other styles, the user can choose another pre-defined set of chords, or to redefine entirely his own set of chords, and store it in the database. The definition of the set of chords is easily input through the Wedel score editor, which is also used for displaying the score being analysed. The process of horizontal aggregation extensively uses the set of chords that the user has selected.

We begin the process of aggregation by comparing two consecutive clusters. They are considered the same if the sounds composing the two clusters are the same, regardless of their octave, i.e., each sound of the first cluster belongs to the second, and each sound of the second belongs to the first. In this case, the two clusters are merged in one.

If they have not been merged, the process performs a union of both clusters and compares the result against the each chord in the set of chords:

- If the union, except for the possible ornamental notes, can be exactly mapped to a chord, the two clusters are merged into one.
- If the union, including the possible ornamental notes, can be exactly mapped to a chord, the two clusters are merged into one, and the ornamental notes are now considered to be harmonic notes.

The merge is first applied beat by beat, and then measure by measure, and is iteratively repeated until no more merge can be achieved.

When no further merge can be accomplished, an attempt is made to turn each cluster into a chord, by mapping it to the nearest chord possible.

First, we try to find a chord containing all the harmonic notes of the cluster and conversely. If this attempt fails, we then search for a chord containing all the notes of the cluster (this assumes that the cluster can be an incomplete chord). If this fails, we try to find a chord such that the duration of those cluster's notes which cannot be mapped to any note of this chord, is significantly shorter (actually by a factor of 6) than the total duration of the notes of the cluster (this assumes that these notes are really ornamental notes, but were not previously detected as being so).

5. EVALUATION

5.1 Limitations

Some very special cases are not taken into account in our algorithm, notably pedals. Another limitation is due to the oversimplicity of our rule for detection of ornamental notes: some ornamentals can be followed by a disjoint interval, and these can only be detected by the last attempt of turning a cluster into a chord, as described above.

A further limitation is due to the fact that our algorithm doesn't take sufficiently into account the context. Some problems of context dependencies are handled, as shown below in fig. 5, but the resolution of ambiguities is not sufficiently strong. Let us examine this example extracted from the Gavotte from the French Suite n° 5 by J.-S. Bach:

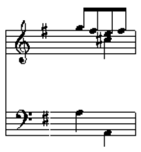


Figure 3. Gavotte from French Suite n°5 by J.-S. Bach, measure 8

In this Gavotte, whose figured bass is given below (see Figure 11), the harmony is a seventh chord on the dominant of D (A - C# - E - G). But in some other contexts, it can be a sixth chord on the root of F# (this analysis being the one produced by our algorithm).

More generally, we must limit the scope of our harmonic analysis to accompanied melody, even if in some limit cases of monophonic voice, a good result can be obtained (as shown below with Mozart's example). We think also that these results can be applied to some music of the 20th century, for example Bartok's works, by redefining the set of chords, but we are aware that this method cannot be applied to contrapuntal work.

5.2 Examples

These examples show the process of harmonic reduction applied to the Trio of Mozart's Clarinet Quintet.

The first example[1] shows elimination of ornamental notes and reconstruction of incomplete chords

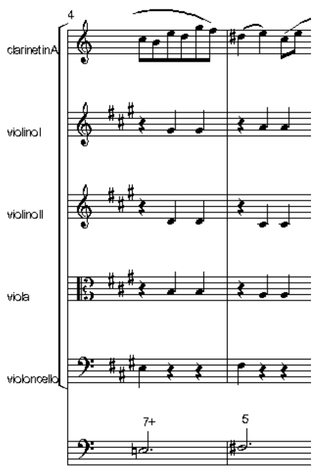


Figure 4. Elimination of ornamental notes and reconstruction of incomplete harmony. (Trio of the Clarinet Quintet by Mozart, KV 581)

The figure 5 shows the resolution of ambiguities:

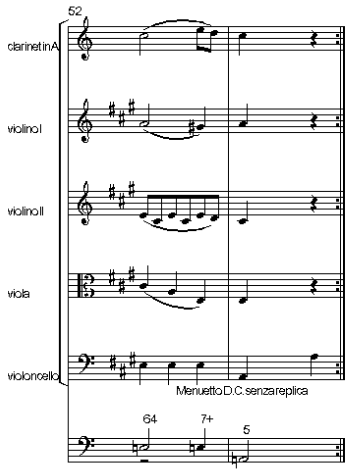


Figure 5. Resolution of ambiguities (Trio of the Clarinet Quintet by Mozart, KV 581)

The harmony on the third beat is not analysed as being a 6th chord, as the C# in the clarinet part is determined as a potential ornamental note (an appoggiatura), and thus, the harmony is merged with the following one, giving as a result a correct analysis of a 7th chord on the fifth degree.

The following example shows that the algorithm can produce correct results even in the case of a simple monophonic voice:

[1] The notation of figures follows the conventions of figured bass as stated in the treatise, with the following exceptions: figures are written from left to right and not from top to bottom, and a slash following a figure indicates that this figure is diminished.

7+ is for $\overset{7}{+}$, 65/ is for $\overset{6}{\not{5}}$.

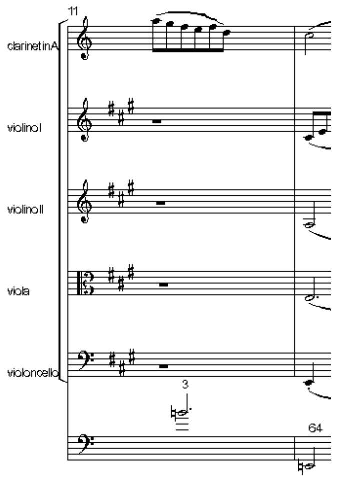


Figure 6. Detection of the root for a monophonic voice (Trio of the Clarinet Quintet by Mozart, KV 581)

The root is correctly detected as being a B.

This last example shows that detection of figured bass is not constrained by rhythm:

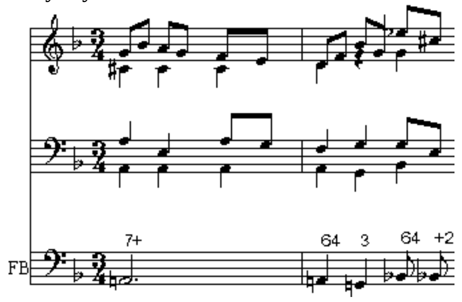


Figure 7. Measures 6 – 7 , Sarabande in D minor by J.-S. Bach

6. Application to tonality detection

On top of this harmonic reduction, we have developed a simple algorithm of tonality detection. This algorithm is based on the fact that each chord can belong to a limited number of tonalities.

The possible tonalities are derived from the figured bass as previously obtained, and a process of elimination is then applied by successively merging regions where there is at least one tonality in common, eliminating tonalities not common to the regions being merged. Where there is no common tonality, a change of tonality is therefore detected.

This algorithm, proceeding as an "island-growing" mechanism, is very near to the system implemented by Ulrich.

The result of this operation for the Trio of the Clarinet Quintet by Mozart is shown here, together with the complete figured bass generated by the system:

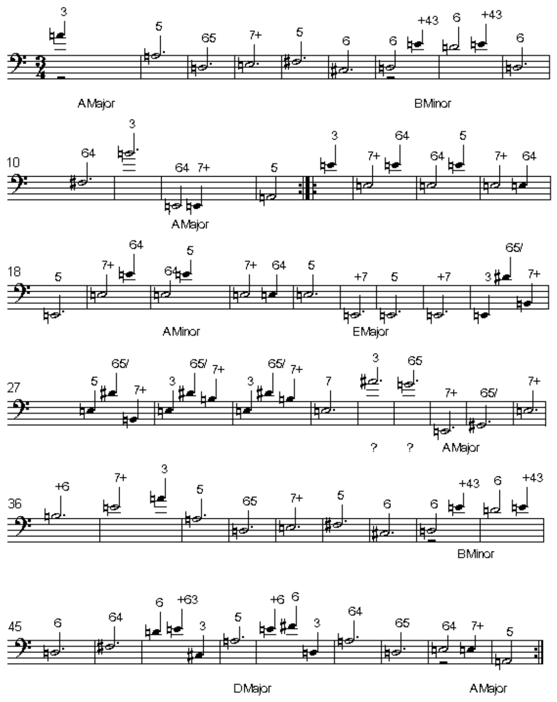


Figure 8: Figured Bass and Tonalities detected for the Trio of the Clarinet Quintet by Mozart, KV 581

The figured bass presented here is totally consistent with an analysis done by a human analyst, with a small exception (in measures 31 and 32).

The detected tonalities are written below the figured bass. When a change of tonality is detected, it is written on the score, the tonality is determined to be the same until the next change of tonality. If a tonality is not recognized, it is denoted bys "?".

The tonalities are correctly detected as being A Major, B Minor, A Minor, E Major and D Major, with the exception of measures 31 and 32 where the tonality is unrecognised.

The advantage of this approach is that, due to the harmonic reduction process, a number of problems related to tonality recognition are easily solved.

In particular, certain notes "out of the tonality", that is, notes which are not really part of the tonality, are eliminated from the process. One can notice, using the original score, that a B# in measure 5 or a E# in measure 51 are completely ignored and do not interfere with the process of tonality recognition.

However, some problems are raised by this simplification.

In the following example from "Eine Kleine Nachtmusik" by Mozart, measures 24-28, a main tonality is simply ignored:

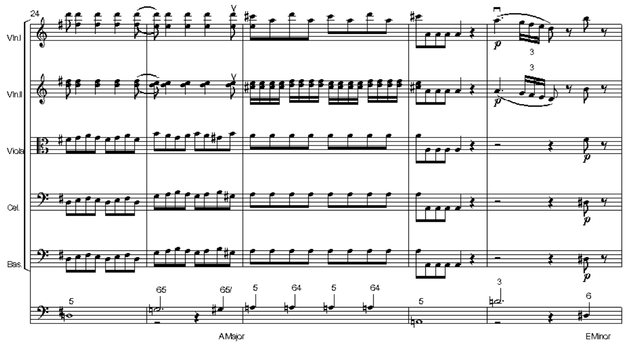


Figure 9. Mozart's "Eine Kleine Nachtmusik"

The musicologist easily recognizes in measure 28 the main entry of the second theme, in D Major.

Unfortunately, the G natural is ignored by the process of harmonic reduction, being a passing note, even if the root harmony is correctly recognized as D. So, between the (short) modulation in A found at the end of measure 25, and the (short) modulation in E minor correctly recognized at the end of measure 28, the main tonality of D Major is not recognized.

A possible solution to this problem can be a refinement of the model of tonality recognition by adding a rule recognizing some modulations as being embedded modulations (in some French treatises of Harmony, such modulations are called "emprunts", i.e., "loans"). To this end, a derivation of the model of François Pachet can be applied.

7. COMPARISON

For the purpose of comparing our models with other work, we show here the result of the production of figured bass applied to a fragment of a Sarabande in D minor by J.-S. Bach, whose analysis can be found in the papers of Maxwell [6] and Pardo [8]:

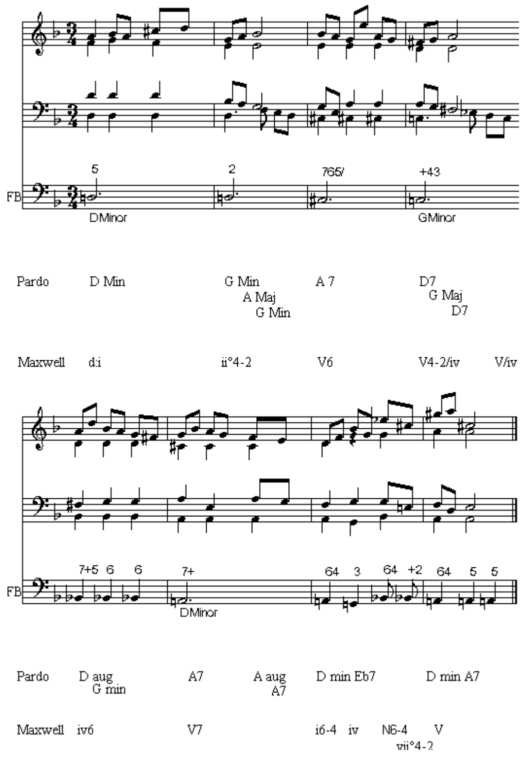


Figure 10. Sarabande in D minor by J.-S. Bach

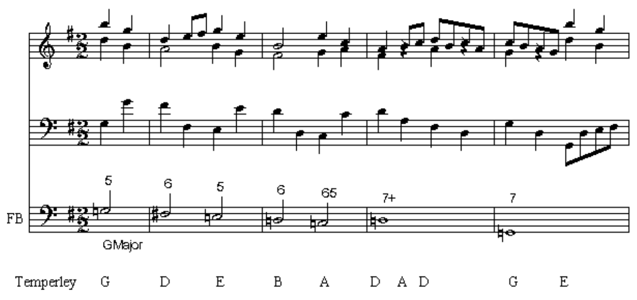
The result of the production of figured bass is shown here on the third staff, marked "FB", together with the recognized tonalities.

The results of Pardo and Maxwell are shown on the following lines.

The results of Maxwell are identical to ours, with a (very little) exception at the beginning of measure 5: the reason is that chord Bb – D - F# - A is part of our templates. In measure 8, Maxwell's system doesn't recognize the sixth-fourth chord on the root of D.

Pardo's result suffers from several drawbacks: the system produces an A Major chord on the second eighth note of measure 2, and a G Major chord on the second eighth note of measure 4, this last one being quite annoying since the correct tonality in this context is G Minor. Incorrect analysis of augmented chords on the first beat of measure 5 and on the third beat of measure 6 are certainly due to the MIDI-like representation of notes. In addition, one cannot understand the analysis of the last chord (A7), the seventh - G - being not in the chord.

We have also applied the Figured Bass to the Gavotte already analysed by Temperley [12].



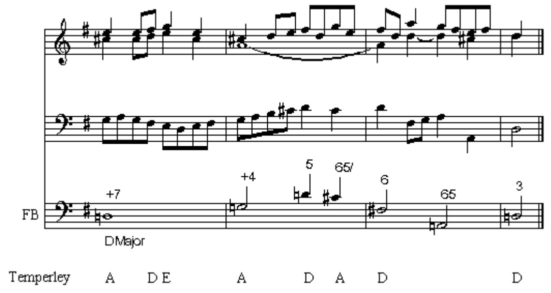


Figure 11. Gavotte from French Suite n°5 by J.-S. Bach

There are several drawbacks in this Figured Bass:

- measure 6 is incorrectly analysed as a seventh of dominant on the tonic, this chord being part of our templates,
- the last chord of measure 8 is incorrectly analysed as a sixth and fifth chord,
- the root of measure 5 is correctly detected, but incorrectly figured as a seventh chord.

Temperley's analysis of the same Gavotte also suffers from several drawbacks:

- measure 8 is incorrectly analysed as entirely based on the root of D,
- a incorrect root of A is detected for the second beat of measure 4,
- the second half of measure 5 is incorrectly detected as being based on the root of E,
- incorrect roots of D and E are detected in measure 6.

Temperley's analysis of measure 6 can be considered better than our Figured Bass, but our analysis of measure 4 can be considered better. The definite mistake made in both cases in measure 8 is due to the same fact: our models are not able to analyse correctly the last F# as an ornamental.

8. Conclusion and perspectives

In this paper, we described an algorithm for production of a figured bass.

This algorithm allows the musicologist to redefine "harmony rules" entirely, merely by redefining the chord templates. It is thus much more general than algorithms found in the literature. We have also shown that our results can be considered at least as good as the best results previously found. We are currently trying to make improvements to the algorithm.

We have shown that higher-level processes, for example tonality recognition, can be build on the top of the figured bass. As stated in the introduction, several processes can be build on the top of a figured bass: detection of cadence, of tonality, of similar structures, and so on. Results of these processes can be stored and indexed in database for the purpose of Music Information Retrieval.

One can notice that the Figured Bass, as the result of a standardized process, can be used as a retrieval criterion. It is a useful criterion for a teacher, for example, in the retrieval of scores using of the same fragment of Figured Bass (the Figured Bass of Sarabande in D minor by J.-S. Bach is an interesting one). To this end, a transposition independent encoding of the Figured Bass must be developed, and we are currently working on it.

Other applications for Music Information Retrieval are possible, such as classification of style based upon the frequency of chords, or upon the relationship between the recognized tonalities and the main tonality, assuming that the complexity of tonal relations is characteristic of a given style. Some techniques actually used to classify melody, such as the Hidden Markov Model (Pollastri, [9]), or techniques issued from Graph Theory can be also applied on the description of the score generated by the harmonic analysis.

9. REFERENCES

[1] Bent, I. (1987), Analysis, Macmillan Press, London, 1987.

[2] Chailley, J., (1977), Traité historique d'analyse musicale, Alphonse Leduc, Paris, 1977.

[3] Cook, N., (1987), A Guide to Musical Analysis, Oxford University Press, 1987.

[4] Hoffmann, T., Birmingham, W. P.,(2000), A Constraint Satisfaction Approach To Tonal Harmonic Analysis, Technical report, Electrical Engineering and Computer Science Department, University of Michigan.

[5] Lerdahl, F., Jackendoff, R., (1983) A Generative Theory of Tonal Music, MIT press, London, 1985

[6] Maxwell, H.J. (1992) "An Expert System for Harmonic Analysis of Tonal Music", Understanding Music with AI: Perspectives on Music Cognition, M. Balaban, K. Ebcioglu, and O. Laske, eds., AAAI Press, 335-353.

[7] Pachet, F., (1997) "Computer Analysis of Jazz Chord Sequences: Is Solar a Blues?", in Readings in Music and Artificial Intelligence, Miranda, E. Ed, Harwood Academic Publishers, February 2000.

[8] Pardo, B, Birmingham, W. P., (1999) Automated Partitioning of Tonal Music, Technical report, Electrical Engineering and Computer Science Department, University of Michigan.

[9] Pollastri, E., Simoncelli, G., (2001), Classification of Melodies by Composer with Hidden Markov Models, to be published in Proceedings of the WedelMusic conference, Florence, 2001

[10] Schoenberg, A., Structural Functions of Harmony, London, Faber and Faber, 1969.

[11] Schoenberg, A., Theory of Harmony, London, Faber and Faber, 1978.

[12] Temperley, D., (1996) “An Algorithm for Harmonic Analysis”, Music Perception 15/1 (1997), 31-68.

[13] Ulrich, W., (1977) “The Analysis and Synthesis of Jazz by Computer”, Fifth International Joint Conference on Artificial Intelligence, MIT, Cambridge, MA, 865-872.

[14] Winograd, T., (1968) “Linguistic and Computer Analysis of Tonal Harmony”, Journal of Music Theory;, 12, 2-49.

[15] Wedelmusic, http://www.wedelmusic.org

ISMIR 2001

Wednesday, October 17, 2001

7:00am-5:00pm	E-mail room: IMU088 *
7:30-8:15am	Registration and Continental Breakfast: Alumni Hall
8:15-8:20am	Opening Remarks
8:20-8:45am	*Indiana University Digital Music Library Project: An Update* Jon W. Dunn, Mary Wallace Davidson, Eric J. Isaacson

8:45-10:20am, Music Retrieval 1

8:45-9:30am	Invited Speaker, Roger B. Dannenberg: *Music Information Retrieval as Music Understanding*
9:30-9:55am	*Towards a Cognitive Model of Melodic Similarity* Ludger Hofmann-Engl
9:55-10:20am	*Building a Platform for Performance Study of Various Music Information Retrieval Approaches* Jia-Lien Hsu, Arbee L. P. Chen

10:20-10:45am	Break/Refreshments

10:45-12:25pm, Music Retrieval 2

10:45-11:10am	*Efficient Multidimensional Searching Routines for Music Information Retrieval* Josh Reiss, Jean-Julien Aucouturier, Mark Sandler
11:10-11:35am	*Expressive and Efficient Retrieval of Symbolic Musical Data* Michael Droettboom, Ichiro Fujinaga, Karl MacMillan, Mark Patton, James Warner, G. Sayeed Choudhury, Tim DiLauro
11:35-12:00n	*A Technique for "Regular Expression" Style Searching in Polyphonic Music* Matthew J. Dovey
12:00-12:25	*An Approach Towards a Polyphonic Music Retrieval System* Shyamala Doraisamy, Stefan M. Rüger

12:30-1:45pm	Lunch

1:45-3:45pm, Audio and MPEG

1:45-2:30pm	Invited Speakers, Adam Lindsay and Youngmoo Kim: *Adventures in Standardization, or How We Learned to Stop Worrying and Love MPEG-7*
2:30-2:55pm	*Content-based Identification of Audio Material Using MPEG-7 Low Level Description* Eric Allamanche, Jürgen Herre, Oliver Hellmuth, Bernhard Fröba, Thorsten Kastner, Markus Cremer
2:55-3:20pm	*Automatic Musical Genre Classification Of Audio Signals* George Tzanetakis, Georg Essl, Perry Cook
3:20-3:45pm	*Music Signal Spotting Retrieval by a Humming Query Using Start Frame Feature Dependent Continuous Dynamic Programming* Takuichi Nishimura, Hiroki Hashiguchi, Junko Takita, J. Xin Zhang, Masataka Goto, Ryuichi Oka

3:45-4:10pm	Break/Refreshments
4:10-5:00pm	Closing/Discussion: *Whither Music Information Retrieval: Ten Suggestions to Strengthen the MIR Research Community* J. Stephen Downie

* E-mail room computers will be configured with Telnet and Web browsers. Check with your network and/or security personnel on any special arrangements you may need to connect to your home or institutional resources.

Indiana University Digital Music Library Project: An Update

Jon W. Dunn
Digital Library Program
Indiana University
1320 E. 10th St.
Bloomington IN 47405, USA
jwd@indiana.edu

Mary Wallace Davidson
Cook Music Library
Indiana University
1201 E. 3rd St.
Bloomington IN 47405, USA
mdavidso@indiana.edu

Eric J. Isaacson
School of Music
Indiana University
1201 E. 3rd St.
Bloomington IN 47405, USA
isaacso@indiana.edu

ABSTRACT
This talk will present a progress report on the Indiana University Digital Music Library project as it enters its second of four years.

1. INTRODUCTION
The Indiana University Digital Music Library project [1] aims to establish a digital music library testbed system containing music in a variety of formats, involving research and development in the areas of system architecture, metadata standards, component-based application architecture, network services, and intellectual property rights. This system will be used as a foundation for digital library research in the areas of instruction, usability, and human-computer interaction.

Key to the projec t is the interdisciplinary team of investigators, who represent the academic disciplines of information science, computer science, law, music theory and music education, as well as the professional disciplines of academic research libraries and information technology services.

The project builds in part upon experiences with a previous operational digital music library system at Indiana University known as VARIATIONS [2].

The Digital Music Library (DML) testbed system will provide users with access to a collection of music in several formats from a range of musical styles and types. Users will be able to listen to sound recordings of musical performances; display and browse images of published scores; view and manipulate encoded score notation files; have notation translated into MIDI format for audio playback; and make use of active links that connect a musical work in one format to a representation in a different format.

The DML system will provide navigation, search, and retrieval functions for this large and diverse information space. This will include search based on descriptive metadata; retrieval and synchronized playback of recorded music, MIDI files and encoded music notation files; access to structural metadata for manipulation of and navigation within individual recordings or other music representations; access control and authentication services; and administrative metadata for rights management.

The DML system will provide a software framework to make digital music objects (music sound recordings, score notation files, text files, etc.) accessible to music instructors and application developers, using a component-based programming architecture. This framework will serve as the foundation for developing and delivering software applications that integrate the collections of the DML into teaching and research in the field of music.

In addition, as the DML collection of audio and notation content grows, we hope to work with other research groups to integrate content-based audio and notation music IR technologies as part of the testbed system.

2. PROJECT PROGRESS
Current plans are to have an initial version of the DML testbed system, developed as a client-server Java application and providing basic bibliographic search capabilities and access to sound recordings and score images, available in January 2002. The system will be tested at IU and at seven "satellite sites" in the US, the UK, and Japan. New versions of the system will follow roughly every six months thereafter, adding new functionality and new user interfaces.

Over the course of the first year of the project, which began in October 2000, much progress has been made, including:

- Usability testing of the existing VARIATIONS digital library system
- Documentation of requirements for the first version of the system
- Design and development of a data model and metadata specification for the system
- Design and prototyping of user interfaces for the first version of the system
- Gathering of user requirements and development of specifications for the Multimedia Music Theory Teaching (MMTT) tool [3], to be included in later versions of the DML testbed
- Research into copyright issues surrounding the creation of digital library systems for music

Our presentation at ISMIR 2001 will provide an overview of the project and an update on project progress from the perspectives of three of the project co-PI's representing diverse backgrounds: technologist, music librarian, and music theory faculty member.

2. ACKNOWLEDGMENTS

This work is supported in part by the Digital Libraries Initiative – Phase 2 (DLI2) program, with funding from the National Science Foundation and National Endowment for the Humanities, NSF award 9909068.

REFERENCES

[1] Digital Music Library project web site. http://dml.indiana.edu/

[2] Dunn, J. and Mayer, C. VARIATIONS: A Digital Music Library System at Indiana University. In *Proceedings of the Fourth ACM Conference on Digital Libraries*, Berkeley, California, 1999.

[3] Multimedia Music Theory Teaching project web site. http://theory.music.indiana.edu/mmtt/

ISMIR 2001 Invited Address

Music Information Retrieval as Music Understanding

Roger B. Dannenberg
Carnegie Mellon University
School of Computer Science
Pittsburgh, PA 15213 USA
1-412-268-3827
rbd@cs.cmu.edu

ABSTRACT

Much of the difficulty in Music Information Retrieval can be traced to problems of good music representations, understanding music structure, and adequate models of music perception. In short, the central problem of Music Information Retrieval is Music Understanding, a topic that also forms the basis for much of the work in the fields of Computer Music and Music Perception. It is important for all of these fields to communicate and share results. With this goal in mind, the author's work on Music Understanding in interactive systems, including computer accompaniment and style recognition, are discussed.

1. INTRODUCTION

One of the most interesting aspects of Music Information Retrieval (MIR) research is that it challenges researchers to form a deep understanding of music at many levels. While early efforts in MIR were able to make impressive first steps even with simple models of music, it is becoming clear that further progress depends upon better representations, better understanding of music structure, and better models of music perception.

As MIR research progresses, the community will undoubtedly find more and closer ties to other music research communities, including "Computer Music," probably best represented by the International Computer Music Association and its annual conference [22], and "Music Perception" as represented by the Society for Music Perception and Cognition [25]. While MIR is not the main focus of either of these communities, there is considerable overlap in terms of music processing, understanding, perception, and representation.

The goal of this presentation is to survey some work (mostly my own) in Music Understanding and to describe work that is particularly relevant to MIR. Most of my work has focused on interactive music systems. Included in this work is extensive research on computer accompaniment systems, in which melodic search and comparison are essential components. Other efforts include beat-tracking, listening to and accompanying traditional jazz performances, and style classification of free improvisations. Along with the preparation of this presentation, I am placing many of the cited papers on-line so they will be more accessible to the MIR community.

My thesis is that a key problem in many fields is the *understanding and application of human musical thought and processing*; this drives much of the research in all fields related to music, science, and technology. This is not to say that these fields are equivalent, but it is important to understand how and why they are related. The work that I describe here shares many underlying problems with MIR. I hope this overview and the citations will be of some benefit to the MIR community.

2. COMPUTER ACCOMPANIMENT

The general task of computer accompaniment is to synchronize a machine performance of music to that of a human. I introduced the term *computer accompaniment* in 1984, but others terms have been used including *synthetic performer* [26], *artificially intelligent performer* [2] and *intelligent accompanist* [5]. In computer accompaniment, it is assumed that the human performer follows a composed score of notes and that both human and computer follow a fully notated score. In any performance, there will be mistakes and tempo variation, so the computer must listen to and follow the live performance, matching it to the score.

Computer accompaniment involves the coordination of signal processing, score matching and following, and accompaniment generation. Because of the obvious similarity of score matching to music search, I will focus on just this aspect of computer accompaniment. See the references for more detail.[9, 12]

2.1 Monophonic Score Following

My first computer accompaniment systems worked with acoustic input from monophonic instruments. The system is note-based: the sequence of performed pitches is compared to the sequence of pitches in the score. Times and durations are ignored for the purposes of matching and comparison, although timestamps must be retained for tempo estimation and synchronization.

Originally, I tried to apply the algorithm from the Unix *diff* command, which, viewed from the outside, seems to be perfect for comparing note sequences. Unfortunately, *diff* does not work here because it assumes that lines of text are mostly unique. This led to the exploration and application of dynamic programming, inspired by longest common substring (LCS) and dynamic timewarp algorithms [24]. To my knowledge, this is the first use dynamic programming for melodic comparison.

Recall that LCS computes a matrix of size $m{\cdot}n$ for strings of length m and n. An important refinement for real-time music recognition is the introduction of a sliding window centered around the current score position. This reduces the computation cost per note to a constant.

This windowing idea could be used in music search applications, especially to compare long query strings to stored strings. The window only affects the result in cases where the match is poor, but presumably these cases are not of interest anyway.

Dynamic programming algorithms typically look at the final $m \cdot n$ matrix to determine the result, but this is not possible in real time. As a heuristic, my score follower reports a match when a "match score" is computed that is higher than any previous value computed so far. The "match score" is essentially the number of notes matched so far minus the number of notes skipped in the score. This formulation compensates for the tendency to skip over notes in order to find a match.

It is perhaps worth noting that matching the performance (a prefix of the score) to the score is a bit like matching a query, which may be a melodic fragment, to a complete melody. Since a fragment may start and end anywhere in a complete melody, we want to compute the least distance from *any* contiguous fragment of the melody, ignoring a certain (but unknown) prefix and suffix of the melody. Dynamic programming, as formulated for score following, can find this best match with a cost of $m \cdot n$, where m and n are the lengths of the melody and query. (Unfortunately, the windowing idea does not seem to apply here because we do not know where the best match will start in the melody.)

Monophonic score following works very well. The original implementation ran on a 1MHz 8-bit processor that performed pitch estimation, score following, accompaniment, and synthesizer control in real time, and fit under an airline seat in 1984! As a historical note, I suggested in a talk given in 1985 [7] that these matching algorithms could be used to quickly search a database of songs. Unfortunately, I missed the opportunity to mention this in my patent [8] or create the first such implementation.

2.2 Polyphonic Score Following

The logical next step in this research was to consider accompanying keyboard performances, and two algorithms were developed for polyphonic score following. [3] Rather than repeat their full descriptions here, I will simply try to give the main ideas and properties of the algorithms. One approach, developed by Josh Bloch, generalizes the idea of "note" to "compound event." A *compound event* is set of simultaneous note onsets, i.e. a chord. The score and performance are regarded as sequences of compound events, and we are essentially looking for the best match. The quality of the match is determined by the number of pitches that match within corresponding compound events minus the number of pitches that are skipped. This is easily solved using dynamic programming, where rows and columns correspond to compound events.

One problem with the preceding algorithm is that it relies upon some process to group events into compound events. We form compound events by grouping notes whose onsets are separated by less than 50 to 100ms. Another algorithm for polyphonic score following was created that forms compound events dynamically. In this algorithm, score events are initially grouped into compound events, but performed events are processed one-at-a-time. What amounts to a greedy algorithm is used to associate performed notes with compound events. Unfortunately, this algorithm does not always find the optimal match because of the heuristic nature of its grouping.

In practice, both algorithms work very well, failing only in (different) contrived pathological cases. Both use the same windowing technique introduced in the monophonic matcher and therefore run in constant time per performed note. The precision of a MIDI keyboard compared to acoustic input, combined with the additional information content of a polyphonic score, makes computer accompaniment of keyboard performances very robust.

These algorithms could be used for music search, but they rely on matching notes as opposed to something more abstract such as harmony. If an improviser plays correct harmonies but in different rhythms or voicings, the match rating might be low. On the other hand, the algorithm can be used as a sort of *diff* on MIDI files, for example to compare different performances [21, 23] or editions. Another interesting application of this technology is in intelligent piano tutoring systems. [4, 6, 13]

2.3 Ensemble Accompaniment

With keyboard performance, the right and left hands are generally synchronized, but this is not so true of ensembles. Following and accompanying an ensemble can be accomplished by following each musician separately and then integrating the results.[10, 15, 16] One of the interesting problems encountered here is that different performers may have more or less relevance at any given time. Usually, performers that have performed a note more recently and that are synchronized with other performers are better sources of timing information. The situation changes constantly in a performance as one part assumes prominence and another plays a background role or rests.

In MIR research, it is common to assume music is a totally ordered sequence of notes or features. It might be useful to consider that, in performance, individuals are not always synchronized. Instead, each performer has a separate notion of time and has a strong goal to produce coherent musical gestures. The synchronization of all these independent lines and gestures is a quasi-independent task performed as each performer listens to the others.

2.4 Vocal Accompaniment

In spite of the success of monophonic and polyphonic matchers for score following, these techniques do not work well for vocal soloists. The main problem is that vocal melodies are difficult to segment into discrete notes, so the data seen by the matcher has a high error rate. Similar problems occur in MIR systems, and a more detailed analysis can be found in Lorin Grubb's thesis [19].

Given that discrete string matching methods cannot be applied to vocal music, Grubb's solution [18, 20] is based on the idea of using probability theory to form a consistent view based on a large number of observations that, taken individually, are unreliable. The probabilistic framework allows the system to be trained on actual performance data; thus, typical performance errors and signal processing errors are all integrated into the framework and accounted for.

The system effectively matches pitch as a function of time to the score, but rather than use dynamic time warping, Grubb's system represents score position as a probability density function. This density function is updated using a model of tempo variation, accounting for natural variations in performed tempo, and a model of pitch observations, accounting for the natural distribution of pitch around the one notated in the score. In addition, phonetic information and note onset information can be integrated within the probabilistic framework.[17] This work forms an interesting basis for MIR using vocal queries.

3. Listening to Jazz

It would be wrong to assume every MIR query can be formulated as a melodic fragment. Similarly, it is restrictive to assume accompanists can only follow fully notated music. What about jazz, where soloists may follow chord progressions, but have no predetermined melody? Working with Bernard Mont-Reynaud, I developed a real-time blues accompaniment system that analyzed a 12-bar blues solo using supervised learning to characterize typical pitch distributions and a simple correlation strategy to identify location.[11] This work also included some early beat induction techniques.[1] It seems unlikely that these techniques will be directly applicable to MIR systems, but the general idea that improvised solos (or even stylized interpretations of melodies) can be understood in terms of harmonic and rhythmic structure is important for future MIR research.

4. Style Classification

An underlying structure of beats, measures, harmony and choruses supports traditional jazz solos. I am interested in interactive improvisations with computers where this structure is absent. Instead, I want the computer to recognize different improvisational styles, such as “lyrical,” “syncopated,” and “frantic” so that the improviser can communicate expressive intentions to the computer directly through the music, much as human musicians communicate in collective improvisations. This goal led to work in style classification using supervised machine learning.[14] This work has obvious applications to music search where the object is to retrieve music of a certain genre or style. We were able to obtain good classification rates on personal styles using quite generic features obtained from a real-time pitch analyzer. Recognition was based on only 5 seconds of music to minimize latency in a real-time performance.

5. Conclusions

Music Understanding is a critical part of Music Information Retrieval research as well as a central topic of Computer Music and Music Perception. The similarities between score following and style classification to problems in MIR are striking. I hope that this paper will introduce some pioneering work in Music Understanding to a broader audience including especially MIR researchers.

REFERENCES

[1] Allen, P. and Dannenberg, R.B., Tracking Musical Beats in Real Time. in *1990 International Computer Music Conference*, (1990), International Computer Music Association, 140-143. http://www.cs.cmu.edu/~rbd/bib-beattrack.html#icmc90 (note that an extended version of the paper is available online).

[2] Baird, B., Blevins, D. and Zahler, N. Artificial Intelligence and Music: Implementing an Interactive Computer Performer. *Computer Music Journal*, *17* (2). 73-79.

[3] Bloch, J. and Dannenberg, R.B., Real-Time Accompaniment of Polyphonic Keyboard Performance. in *Proceedings of the 1985 International Computer Music Conference*, (1985), International Computer Music Association, 279-290. http://www.cs.cmu.edu/~rbd/bib-accomp.html#icmc85.

[4] Capell, P. and Dannenberg, R.B. Instructional Design and Intelligent Tutoring. *Journal of Artificial Intelligence in Education*, *4* (1). 95-121.

[5] Coda. SmartMusic Studio 6.0, Coda Music Technology, Inc., Eden Prarie, MN, 2000. http://www.codamusic.com/coda/sm.asp.

[6] Dannenberg, R., Sanchez, M., Joseph, A., Saul, R., Joseph, R. and Capell, P., An Expert System for Teaching Piano to Novices. in *1990 International Computer Music Conference*, (1990), International Computer Music Association, 20-23. http://www.cs.cmu.edu/~rbd/bib-ptutor.html#icmc90.

[7] Dannenberg, R.B. Computer Accompaniment (oral presentation), STEIM Symposium on Interactive Music, Amsterdam, 1985.

[8] Dannenberg, R.B. Method and Apparatus for Providing Coordinated Accompaniment for a Performance, US Patent #4745836, 1988.

[9] Dannenberg, R.B. and Bookstein, K., Practical Aspects of a Midi Conducting Program. in *Proceedings of the 1991 International Computer Music Conference*, (1991), International Computer Music Association, 537-540. http://www.cs.cmu.edu/~rbd/subjbib2.html#icmc91.

[10] Dannenberg, R.B. and Grubb, L.V. Automated Musical Accompaniment With Multiple Input Sensors, US Patent #5521324, 1994.

[11] Dannenberg, R.B. and Mont-Reynaud, B., Following an Improvisation in Real Time. in *Proceedings of the International Computer Music Conference*, (1987), International Computer Music Association, 241-248. http://www.cs.cmu.edu/~rbd/bib-accomp.html#icmc87.

[12] Dannenberg, R.B. and Mukaino, H., New Techniques for Enhanced Quality of Computer Accompaniment. in *Proceedings of the International Computer Music Conference*, (1988), International Computer Music Association, 243-249. http://www.cs.cmu.edu/~rbd/bib-accomp.html#icmc88.

[13] Dannenberg, R.B., Sanchez, M., Joseph, A., Capell, P., Joseph, R. and Saul, R. A Computer-Based Multimedia Tutor for Beginning Piano Students. *Interface - Journal of New Music Research*, *19* (2-3). 155-173.

[14] Dannenberg, R.B., Thom, B. and Watson, D., A Machine Learning Approach to Style Recognition. in *1997 International Computer Music Conference*, (1997), International Computer Music Association. http://www.cs.cmu.edu/~rbd/bib-styleclass.html#icmc97.

[15] Grubb, L. and Dannenberg, R.B., Automated Accompaniment of Musical Ensembles. in *Proceedings of the Twelfth National Conference on Artificial Intelligence*, (1994), AAAI, 94-99.

[16] Grubb, L. and Dannenberg, R.B., Automating Ensemble Performance. in *Proceedings of the 1994 International Computer Music Conference*, (1994), International Computer Music Association, 63-69. http://www.cs.cmu.edu/~rbd/bib-accomp.html#icmc94.

[17] Grubb, L. and Dannenberg, R.B., Enhanced Vocal Performance Tracking Using Multiple Information Sources. in *Proceedings of hte International Computer Music Conference*, (1998), International Computer Music Association, 37-44. http://www.cs.cmu.edu/~rbd/bib-accomp.html#icmc88.

[18] Grubb, L. and Dannenberg, R.B., A Stochastic Method of Tracking a Vocal Performer. in *1997 International Computer Music Conference*, (1997), International Computer Music Association. http://www.cs.cmu.edu/~rbd/bib-accomp.html#icmc97.

[19] Grubb, L.V. *A Probabilistic Method for Tracking a Vocalist*. Carnegie Mellon University, Pittsburgh, PA, 1998. http://reports-archive.adm.cs.cmu.edu/anon/1998/abstracts/98-166.html.

[20] Grubb, L.V. and Dannenberg, R.B. System and Method for Stochastic Score Following, US Patent #5913259, 1997.

[21] Hoshishiba, T., Horiguchi, S. and Fujinaga, I., Study of Expression and Individuality in Music Performance Using Normative Data Derived from MIDI Recordings of Piano Music. in *International Conference on Music Perception and Cognition*, (1996), 465-470. http://www.jaist.ac.jp/~hoshisi/public/papers/icmpc96.pdf.

[22] ICMA. http://www.computermusic.org/, International Computer Music Association, 2001.

[23] Large, E.W. Dynamic programming for the analysis of serial behaviors. *Behavior Research Methods, Instruments, and Computers*, *25* (2). 238-241.

[24] Sankoff, D. and Kruskal, J.B. *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*. Addison-Wesley, Reading, MA, 1983.

[25] SMPC. http://psyc.queensu.ca/~smpc, Society for Music Perception and Cognition, 2001. http://psyc.queensu.ca/~smpc.

[26] Vercoe, B., The Synthetic Performer in the Context of Live Performance. in *Proceedings of the International Computer Music Conference 1984*, (1984), International Computer Music Association, 199-200.

Towards a cognitive model of melodic similarity

Ludger Hofmann-Engl
Keele University UK
+44 (0) 20 87710639
ludger.hofmann-engl@virgin.net

ABSTRACT

In recent years the interest in melodic similarity has mushroomed mainly due to the increased importance of music information retrieval (MIR). A great number of similarity models and algorithms have been developed, but little or no attention has been paid to cognitive or perceptual aspects to the issue at hand. Questions, about the relevant parameters and the appropriate implementation are under-researched as are experimental data. This paper focuses on the pitch aspect of melodic similarity, scrutinising the term pitch replacing it by a less ambivalent and overused term, which we will call meloton. Based on the term meloton the paper suggests to approach the issue of 'melotonic' similarity from a transformational angle, where transformations are executed as reflections and translations. 'Melotonic' similarity then is seen as related to the transformation process in form of a transpositional and interval vector. Finally, melotonic similarity as portrait in a psychological context emerges as a multi-facet phenomenon requiring the development of flexible models.

1. INTRODUCTION

Unarguably, melodic similarity has been of great interest to composers (e.g., Schoenberg, 1967), ethnomusicologists (e.g., Adams, 1976) and music analysts (e.g., Reti, 1951). However, the issue has received new interest due to the development of the internet and the need to administrate and retrieve musical information. Early works on MIR date back to the 60's with Kassler (1966) as one of the pioneers. Not much research was done on the topic for some time, but by now the growing interest is reflected for instance in the fact that in 2000 the first international symposium on MIR was organised and attended by researchers from a great variety of fields. The interest of MIR in the issue of similarity does not necessarily add importance to the issue but certainly urgency to develop reliable and relevant similarity models. In fact, by now several models and algorithms have been proposed (Anagnostopoulou, Hörnel & Höthker, 1999; Cambouropoulos, 2000; Crawford, Iliopoulos & Raman, 1998; Dovey &Crawford, 1999; Downie, 1999; Kluge, 1996; Maidin & Fernström, 2000; Smith & McNab & Witten, 1998), but none of these models take cognitive or perceptual issues sufficiently into account, nor do they pay a closer look at as to what parameters to select and how to implement them.

Admittedly, the psychological and more specific the music-psychological research in this field leaves possibly more questions unanswered than answered (compare Goldstone, 1998), but this seems not to justify the dismissal of existing research. Notably, none of the researchers takes dynamic aspects into account and rhythmic aspects play no or little role by regarding melodic similarity exclusively as a pitch phenomenon, without considering the limitations of their models.

Interestingly, the question of what the term pitch, a term which from a psychological angle is more than problematic, is not being asked, although there exists some awareness to the related issue of musical representation; that is whether we are dealing with the representation of music in form of a score, with recorded music or digital sources such as MIDI files (e.g. Wiggins & Harris & Smaill, 1989), but the central issue of pitch perception is hardly ever touched.

Most strikingly, the works of Egmond & Povel & Maris (1996) have not been referred to in a single instance to the knowledge of the author, although their findings are in agreement with previous research (Francès, 1988) and also in agreement with more recent research by Hofmann-Engl & Parncutt (1998). Their experiments indicate that transposition is a significant factor for melodic similarity judgements whereby melodic similarity decreases with increasing size of the transpositional interval. However, all the models known to the author are transpositional invariant. This demonstrates the strong tendency of researchers to borrow their tools from music theoretical teachings where transpositions of a motive are regarded as being equivalent. This is not to say that search tools within MIR should under no circumstances consider transpositions as equivalent (for instance when analysing the structure of a specific composition), but where perceptual issues are of importance such attempt will have to be seen as a shortcoming.

While some models are seemingly unaware of the evidence as produced by researchers (e.g. Dowling & Harwood, 1986; Dowling 1994; Edworthy, 1982, 1985; White, 1960) that contour is somewhat a factor determining cognitive similarity (for instance models based on dynamic programming) there are models which take contour into account (for instance Maidin's model), but still reference to psychological research is not given.

Experiments by Hofmann-Engl and Parncutt (1998) indicate that contour is in fact an imbedded factor of what they called interval difference. This is, a reference-melody raised by an interval *I* between two consecutive tones (let us say tone 1 and tone 2) produces the interval difference $D = I - I'$, with I' being the interval between the two corresponding tones (tone 1 and tone 2) of the comparison-melody. Melodies which show contour differences also produce interval differences and hence contour appears to be a factor. However, multiple regression shows that interval difference is the sole factor. Up to this date no model has accounted for these findings.

Finally, none of the models takes emotional aspects into account. True, that at this point it seems an almost unattainable task, but research by Tekman (1998) shows, that emotional aspects can be at least partly sensibly measured. Clearly, there exists a level of unawareness amongst melodic similarity researchers of psychological issues which seems hardly acceptable.

It is the intention of the present paper to contribute to the bridging of exactly this gap. Although as mentioned before dynamic, rhythmic and emotional aspects will have to be seen as factors alongside pitch, we will focus on pitch exclusively. This is, pitch is the most discussed aspect of melodic similarity, and treating all parameters would exceed the framework of this paper. However, the author is in process of developing a similarity model which takes dynamics and rhythmic features into account alongside with pitch. In the first instance we will scrutinise the term pitch arguing for its substitution by the new term, which will be called meloton, we then will consider the transformation of melodies and finally we will develop a similarity model based on the composition of two specific transformations.

2. PITCH VERSUS MELOTON

The term pitch is intriguing and perplexing at the same time, intriguing, because it is probably the most discussed musical and music psychological term, and perplexing, because it has been employed in so many different contexts that it frequently requires specification to what actually is meant by it. A situation which led Rohwer (1970) to question the usage of the term pitch altogether. However, pitch is commonly understood to be the correlate to the fundamental frequency as Rasch & Plomp (1982) explain: "Pitch is the most characteristic property of tones ... . Pitch is related to the frequency of a simple tone and to the fundamental frequency of a complex tone." Pitch is seen here exclusively as being related to a physical quantity. This is as widespread an approach as is insufficient because subtleties of pitch perception are not captured by referring to physical dimensions only. It seems this is the issue Sundberg (1991) is addressing when he suggests to see pitch as locating musical sound in terms of musical intervals and to call other aspects of pitch perception tone height (the high or lowness of sound). However, the categorical differentiation between pitch and tone height seems frail as even sounds with low pitch salience (e.g., musical chimes) can produce a distinctive pitch sensation (Askill 1997). Confronted with the phenomenon that sounds without fundamental frequency can produce pitch sensations, Schouten (1938) introduced the term residual pitch. However, from a phenomenological angle there is no difference between pitch and residual pitch — both appear to a listener in the same way. Concepts of virtual pitch (Houtsma & Goldstein 1972, Terhardt 1982, Hofmann-Engl 1999) added further complexity to the issue by demonstrating that sounds do not have one single pitch but a multiplicity of pitches with varying degrees of probabilities. Employing this concept the term pitch has to be replaced by the term most probable pitch. It seems the introduction of a new term which will have some specific psychological meaning is more than appropriate. We argue to use the term meloton as suggested by Hofmann-Engl (1989).

Without going into detail, we will consider some features of what this term is to deliver. We wish for this term to be purely of psychological meaning. This is we endeavour to understand melodic similarity from a cognitive angle. Thus, concepts of fundamental frequency and other physicalistic approaches are inadequate. Hereby, the term meloton will represent the psychological concept whereby a listener listens to a sound directing her/his attention to the sound with the intention to decide whether the sound is high or low. True, this does not deliver a quantity we could input into a similarity model, and hence we will have to define the value of a meloton somehow without using a physicalistic concept. In this context it seems most appropriate to consider an experimental setting as employed by Schouten (1938). A selected group of listeners is asked to tune in a (sinusoidal) comparison tone with variable frequency to match according to the listener's perception a test tone (for which we want to obtain a melotonic value). The logarithm of the comparison tone then will be called m-response of this listener. Assuming that the group of listeners consist of n listeners, we will obtain *n* m-responses. We will call the mean of this distribution the m-center. The mode of the distribution will be called m-peak. The relative density of the m-peak will be called melograde, and can be defined as:

$$M_g = \frac{D(p_m)}{\sum_{i=1}^{n} D(l_m)_i}$$

where M_g is the melograde, $D(p_m)$ the density of the peak of the m-distribution, $D(l_m)_i$ the density of the location lm at the place *i* and *n* the number of locations. The range of M_g is]0, 1]. Note, that models of the pitch salience (Terhardt, Stoll & Seewann 1982) are predictors for the peak of the response distribution.

We finally define the value *M* of a meloton as given by the value of the m-centre. However, if the melograde of a m-distribution is larger than 0.75 and the peak and the centre coincide with maximum deviation of 25 cents, the value of the meloton is given by the peak of the m-distribution. In this case we speak of *strong meloton* M_s. In all other cases we speak of *weak meloton* M_w.

There are several advantages to this approach as there are limitations. Firstly, the classification of melota into weak and strong melota guaranties that tones such as produced by a drum instrument will also fetch a melotonic value. This allows for the inclusion of 'drum-melodies' into a melodic similarity model. Secondly, we replaced the dogmatic attitude towards pitch perception by an understanding which is sensitive towards individual, cultural, educational and social differences; what might appear to one group as a sound with a certain melotonic value might appear to another group as a sound with a different melotonic value. This is certainly of importance when considering melodic similarity. Right at the basis melodic similarity judgement might differ due to lower level perceptual differences. Objections might be raised that this approach is impracticable in many ways, as the measurement of melotonic values is time consuming and expensive. Still, we might expect that data bases containing melotonic measurements might be established and made available in future. Another objection might be that even if measurements of single tones are available, there is no guaranty that meloton will retain their values when put into a melodic context. Although this might be true, this will have to be an issue to be investigated. However, considering the success of aural training (where listeners are required to identify tones in any context), we expect that this approach is more than promising.

Before we will base a transformation theory on melota, we will abandon the term melody due its many ambivalent connotations and replace it by the term chain. We will as mentioned only consider the melotonic component of a chain (excluding properties such as timbre, duration and loudness). A melotonic chain will be written as m-chain and as *M*(*ch*). Now, we will consider transformations of melotonic chains.

2. MELOTONIC TRANSFORMATIONS

The motivation for developing a transformation theory is driven by the proposal as put forward by Palmer (1983), where similarity is understood to be related to the transformation process involved in mapping two objects onto each other. There have been several attempts within the camp of pitch class theorists to introduce similarity measures and transformations of pitch class sets (e.g., Isaacson (1996), Lewin (1977), Morris (1979)). However, none of these approaches are of interest in this context, as a pitch class set is fundamentally a different entity than is a melotonic chain. One attempt to describe melodic transformations has been put forward by Mazzola (1987) and has been further developed by Hammel (1999) in form of matrices. Without going into detail, the main deficiency of their transformation matrices is the combination of time and pitch in one matrix leading to time and pitch appearing as mixed terms. Maybe even more important, the matrices as they stand, do not allow for general transformations. Thus, a theory of similarity based on their concept would not allow for the comparison of any melody with any melody.

Instead we will take inversions and transpositions as a starting point and generalise these two transformations. As we are dealing with melotonic transformations, we will require that two chains will have the same rhythm and the same dynamic structure. This is a restriction to the model, but this deficiency can only be overcome at a later stage including rhythm and dynamics and possibly emotional aspects in a final model. However, as mentioned, this would exceed the framework of this paper.

2.1. Inversion

It is a well known geometrical fact, that inversion can be illustrated as a reflection along a straight line. Taking a melotonic chain (melody) *M*(*ch*) to consist of an initial tone with meloton m_1 and then the subsequent intervals 100, 100, -100, -100 (where 100 might be taken to mean 100 cents), we write: *M*(*ch*) = (m_1)[100, 100, -100, -100] to be reflected onto the chain *M*(*ch*') = (m_1')[-100, -100, 100, 100] will require a straight line through the point $p = 50$ (see Figure 1)

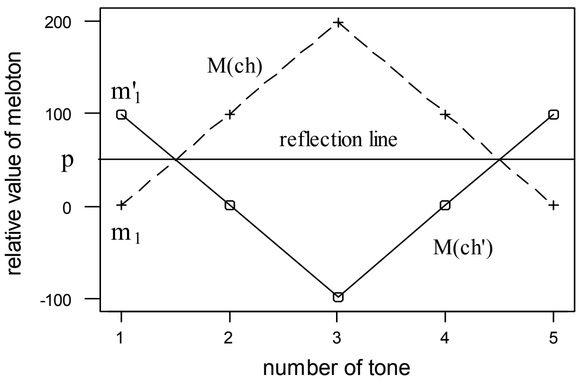


Figure 1: The m-chain M(ch) = (m_1)[100, 100, -100, -100] is mapped onto the chain *M*(*ch*') = (m_1')[-100,-100, 100, 100] via the reflection line through *p*, with $m_1 = 0$ and $p = 50$

2.2. Transposition

Executing two reflections along two different reflection lines results in transposition. Taking the example from above, where *M*(*ch*) = (m_1)[100, 100, -100, -100], we obtain the inversion *M*(*ch*') = (m_1')[-100, -100, 100, 100], when reflecting *M*(*ch*) through $p_1 = 50$ and the transposition *M*(*ch*'') = (m_1'')[100, 100, -100, -100] when reflecting *M*(*ch*') through $p_2 = 150$. The difference between p_2 and p_1 is $p_2 - p_1 = 100$, and the transposition interval between *M*(*ch*) and *M*(*ch*'') is $2(p_2 - p_1) = 200$ (Figure 2). The transposition interval is generally $2(p_2 - p_1)$ regardless where p_2 and p_1 are located. Allowing for the reflecting of single melota rather than the reflection of an entire m-chain, we can illustrate transposition as a reflection along a reflection chain, which we will call *M*(χ), we obtain figure 3.

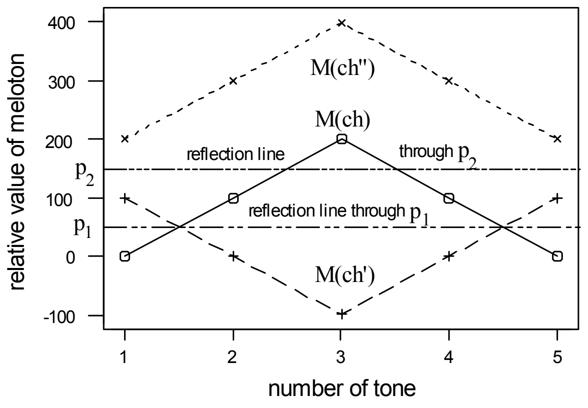


Figure 2: The m-chain $M(ch)$ = (m_1)[100, 100, -100, -100] is mapped onto the chain $M(ch'')$ = (m_1')[-100,-100, 100, 100] via the reflection line through p_1 = 50 and onto $M(ch'')$ = (m_1'')[100, 100, -100, -100] via the reflection line through p_2 = 150. The graph illustrates, that reflection along two lines results in transposition.

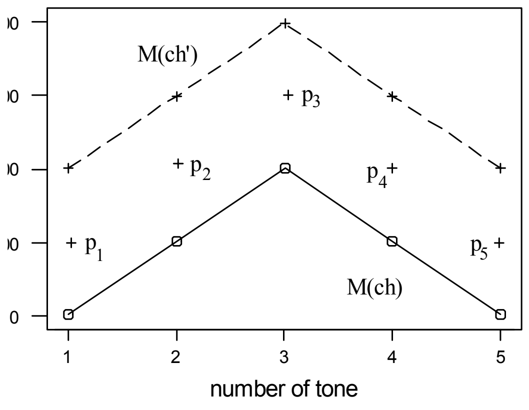


Figure 3: The m-chain $M(ch)$ = (m_1)[100, 100, -100, -100] is mapped onto the chain $M(ch')$ = (m_1')[-100,-100, 100, 100] via the sequence of reflection points p_1, p_2, ... p_5, thus effecting the transposition of $M(ch)$.

2.3. General melotonic transformations

Given a m-chain $M(ch)$ of the length n and a reflection chain $M(\chi)$ of the length n, we find that $M(ch)$ will be mapped onto $M(ch')$, where for all $m_i \in M(ch)$ and $_{all\ pi} \in M(\chi)$, that $m'_i = 2p_i - m_i$, for all $m'_i \in M(ch')$. This is important, when defining reflections and translations (we will use the mathematical term instead for transposition) within the vector-space R^{n+1}. As we will see later, the composition of two specific reflections will enable us to produce similarity measures. However, we define reflections and translations on a more formal level first. For this purpose we define the m-vector $\vec{m}$:

Definition:

Given a m-chain of length n, with $M(ch) = [m_1, m_2, ..., m_n]$, we define the m-vector $\vec{m}$ of length n+1 as:

$$\vec{m} = \begin{pmatrix} m_1 \\ m_2 \\ . \\ . \\ m_n \\ 1 \end{pmatrix}$$

This enables us to define the reflection matrix R as:

Definition

$$R = \begin{pmatrix} -1 & 0 & . & . & 0 & 2p_1 \\ 0 & -1 & & & 0 & 2p_2 \\ . & & & . & & . \\ . & & & & . & . \\ 0 & 0 & & & -1 & 2p_n \\ 0 & 0 & & & 0 & 1 \end{pmatrix}$$

Multiplying the reflection matrix R by a m-vector $\vec{m}$, we obtain:

$$R \cdot \vec{m} = \begin{pmatrix} -1 & 0 & . & . & 0 & 2p_1 \\ 0 & -1 & & & 0 & 2p_2 \\ . & & & . & & . \\ . & & & & . & . \\ 0 & 0 & & & -1 & 2p_n \\ 0 & 0 & & & 0 & 1 \end{pmatrix} \begin{pmatrix} m_1 \\ m_2 \\ . \\ . \\ m_n \\ 1 \end{pmatrix} = \begin{pmatrix} m_1 - 2p_1 \\ m_2 - 2p_2 \\ . \\ . \\ m_n - 2p_n \\ 1 \end{pmatrix}$$

Clearly, multiplying the reflection matrix by a m-vector, results in the reflection of this m-vector. As two reflections result in translation, we will define the translation matrix, where each component $m_i \in \vec{m}$ will be translated by the translation interval

$I_i = 2(p_{2i} - p_{1i})$. We define:

Definition:

$$T = \begin{pmatrix} 1 & 0 & & & 0 & I_1 \\ 0 & 1 & & & 0 & I_2 \\ . & & . & & & . \\ . & & & . & & . \\ 0 & 0 & & & 1 & I_n \\ 0 & 0 & . & . & 0 & 1 \end{pmatrix}$$

Multiplying the translation matrix T with a m-vector $\vec{m}$, we get:

$$T \cdot \vec{m} = \begin{pmatrix} 1 & 0 & . & . & 0 & I_1 \\ 0 & 1 & & & 0 & I_2 \\ . & . & & & & . \\ . & . & & & & . \\ 0 & 0 & & & 1 & I_n \\ 0 & 0 & . & . & 0 & 1 \end{pmatrix} \begin{pmatrix} m_1 \\ m_2 \\ . \\ . \\ m_n \\ 1 \end{pmatrix} = \begin{pmatrix} m_1 + I_1 \\ m_2 + I_2 \\ . \\ . \\ m_n + I_n \\ 1 \end{pmatrix}$$

There exists a complex algebraic structure between reflections and translations. However, the framework of the paper exceeds a discussion of this issue. Still, it might be worth mentioning that the composition of two reflection matrices results in a translation matrix. This is the reason, why we had to introduce translations, although translations are of no significance in context of melotonic similarity. We are now equipped to consider melotonic similarity.

3. MELOTONIC SIMILARITY

It seems the best way of approaching melotonic similarity is, when we consider two m-vectors $\vec{m}_1$ and $\vec{m}_2$, such as:

$$\vec{m}_1 = \begin{pmatrix} m_{11} \\ m_{12} \\ . \\ . \\ m_{1n} \\ 1 \end{pmatrix} \text{ and } \vec{m}_2 = \begin{pmatrix} m_{21} \\ m_{22} \\ . \\ . \\ m_{2n} \\ 1 \end{pmatrix} = \begin{pmatrix} m_{11} + a \\ m_{12} + a \\ . \\ . \\ m_{1n} + a \\ 1 \end{pmatrix}$$

with *a* as a constant

We will now reflect the m-vector $\vec{m}_1$ through the 0-point R^{n+1} via the reflection matrix R_0. We obtain the m-vector $\vec{m}'_1$, with:

$$\vec{m}'_1 = \begin{pmatrix} -1 & 0 & . & . & 0 & 0 \\ 0 & -1 & & & 0 & 0 \\ . & & . & & & . \\ . & & & . & & . \\ 0 & 0 & & & -1 & 0 \\ 0 & 0 & . & . & 0 & 1 \end{pmatrix} \begin{pmatrix} m_1 \\ m_2 \\ . \\ . \\ m_n \\ 1 \end{pmatrix} = \begin{pmatrix} -m_1 \\ -m_2 \\ . \\ . \\ -m_n \\ 1 \end{pmatrix}$$

Reflecting the m-vector $\vec{m}'_1$ onto the m-vector $\vec{m}_2$ requires the reflection matrix R^s given as:

$$R^S = \begin{pmatrix} -1 & 0 & . & . & 0 & 2\left(\frac{-m_{11} + m_{21}}{2}\right) \\ 0 & -1 & & & 0 & 2\left(\frac{-m_{12} + m_{22}}{2}\right) \\ . & & . & & & . \\ . & & & . & & . \\ 0 & 0 & & & -1 & 2\left(\frac{-m_{1n} + m_{2n}}{2}\right) \\ 0 & 0 & . & . & 0 & 1 \end{pmatrix}$$

As we find:

$$R^S \cdot \vec{m}'_1 = \vec{m}_2$$

Isolating the last column in the subspace R^n of R^{n+1}, we can define the similarity vector $\vec{V}_s$:

Definition:

$$\vec{V}_s = 2 \begin{pmatrix} \frac{-m_{11} + m_{21}}{2} \\ \frac{-m_{12} + m_{22}}{2} \\ . \\ . \\ \frac{-m_{1n} + m_{2n}}{2} \end{pmatrix} = \begin{pmatrix} -m_{11} + m_{22} \\ -m_{12} + m_{22} \\ . \\ . \\ -m_{1n} + m_{2n} \end{pmatrix}$$

With $m_{2i} = m_{1i} + a$, we obtain the similarity vector:

$$\vec{V}_s = \begin{pmatrix} a \\ a \\ . \\ . \\ a \end{pmatrix}$$

Geometrically, this means that the similarity vector comes to coincide with the diagonal of the space R^n. We further find for the length of this vector:

$$\left\| \vec{V}_s \right\| = a\sqrt{n}$$

Clearly, the larger the transposition interval a is, the larger will be the length of the similarity vector. According the van Egmond, Povel & Maris (1996) melodic similarity decreases with increasing transposition interval. Thus, we will expect that the length of the similarity vector will be correlated to the transpositional component of melotonic similarity.

The intervalic component of melotonic similarity, according to Hofmann-Engl & Parncutt (1998) is a significant predictor. This is, when two m-vectors $\vec{m}_1$ and $\vec{m}_2$ are not simply transpositions of each other but deviate in shape. The similarity vector $\vec{V}_s$ will then deviate from the diagonal in R^n. Without going into a lengthily discussion, the angle between the similarity vector and the diagonal of R^n is not a suitable measure of the intervalic similarity component, as small intervalic changes can lead to a sudden increase of the angle. However, the differences between the components of the similarity vector are. Given the similarity vector $\vec{V}_s$ as:

$$\vec{V}_s = \begin{pmatrix} s_1 \\ s_2 \\ . \\ . \\ s_n \end{pmatrix}$$

We define the interval vector $\vec{V}_i$ with $I_i = s_{i+1} - s_i$ as a vector of the dimension R^{n-1}

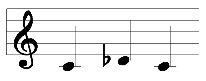
Definition:

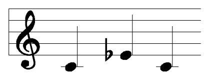
$$\vec{V}_i = \begin{pmatrix} I_1 \\ I_2 \\ . \\ . \\ I_{n-1} \end{pmatrix}$$

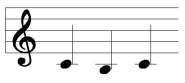
We will give an example referring to the four m-chains $M(ch_1) = (m_1)[1, -1]$, $M(ch_2) = (m_1)[3, -3]$, $M(ch_3) = (m_1)[-1, 1]$ and $M(ch_4) = (m_1)[1, 1]$ (where 1 unit might be one semitone). Setting m_1 to be $m_1 = 0$, we obtain the four m-vectors:

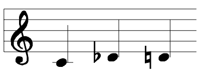
$$\vec{m}_1 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}, \quad \vec{m}_2 = \begin{pmatrix} 0 \\ 3 \\ 0 \\ 1 \end{pmatrix}, \quad \vec{m}_3 = \begin{pmatrix} 0 \\ -1 \\ 0 \\ 1 \end{pmatrix} \text{ and } \vec{m}_4 = \begin{pmatrix} 0 \\ 1 \\ 2 \\ 1 \end{pmatrix}$$

In musical notation we obtain (setting the first tone to be c):

$M(ch_1) =$ 

$M(ch_2) =$ 

$M(ch_3) =$ 

$M(ch_4) =$ 

We find:

$$\vec{V}_s(\vec{m}_1,\vec{m}_2) = \begin{pmatrix} 0 \\ 2 \\ 0 \end{pmatrix} \text{ and } \vec{V}_i(\vec{m}_1,\vec{m}_2) = \begin{pmatrix} 2 \\ -2 \end{pmatrix}$$

$$\vec{V}_s(\vec{m}_1,\vec{m}_3) = \begin{pmatrix} 0 \\ 2 \\ 0 \end{pmatrix} \text{ and } \vec{V}_i(\vec{m}_1,\vec{m}_3) = \begin{pmatrix} 2 \\ -2 \end{pmatrix}$$

$$\vec{V}_s(\vec{m}_1,\vec{m}_4) = \begin{pmatrix} 0 \\ 0 \\ 2 \end{pmatrix} \text{ and } \vec{V}_i(\vec{m}_1,\vec{m}_4) = \begin{pmatrix} 0 \\ 2 \end{pmatrix}$$

The lengths of the similarity vectors $\vec{V}_s(\vec{m}_1,\vec{m}_2)$, $\vec{V}_s(\vec{m}_1,\vec{m}_3)$ and $\vec{V}_s(\vec{m}_1,\vec{m}_4)$ are identical (=2). Moreover, the lengths of the interval vectors $\vec{V}_i(\vec{m}_1,\vec{m}_2)$ and $\vec{V}_i(\vec{m}_1,\vec{m}_3)$ are identical (= $\sqrt{8}$), although $M(ch_1)$ and $M(ch_2)$ have the same contour, while $M(ch_1)$ and $M(ch_3)$ have different contour. However, both chains show the same interval difference. As mentioned above contour differences are imbedded within interval differences. Thus, similarity and interval vector are in agreement with experimental findings. Further, the comparison of $M(ch_1)$ and $M(ch_4)$ demonstrates, that the length of the similarity vector (=2) does not necessarily produce a length of $\sqrt{8}$. Thus, similarity and interval vector are independent similarity predictors. Because of the smaller length of the interval vector, we expect $M(ch_1)$ to be more similar when compared with $M(ch_4)$ than when compared with $M(ch_2)$.

We will expect that melotonic similarity will be correlated to the length or a derivative of the similarity vector (the longer the vector the smaller the similarity) and the deviation of the similarity vector from the diagonal measured by the length or a derivative of the interval vector (the longer the interval vector the smaller the similarity).

While the similarity vector takes the differences of two corresponding melota m_i and m'_i into account without considering any higher order (it does not matter where a pair of melota is placed within a chain), the interval vector considers higher order relationships in as much as pair-wise groupings are covered (the difference between $m_i - m_{i+1}$ and $m'_i - m'_{i+1}$). We could take even higher order relationships into account by forming the differences of the components of the interval vector in the fashion we formed the differences of the similarity vector obtaining the interval vector. We then could from the differences of these differences and so on. We then would obtain a series of vectors with decreasing dimensions starting with the similarity vector of dimension n, followed by the interval vector with the dimension n-1, followed by the differences of the interval vector producing a vector of dimension n-2 and so on till we obtain a vector of dimension 1. Thus, higher order relationships would be covered and a predictor of melotonic similarity could be modelled around something comparable to a Taylor series. However, we expect that the similarity vector and interval vector will be sufficient to produce useful approximations.

Basing a similarity model exclusively on the lengths of the similarity and interval vector will still produce several complications. Without going into much detail, we will consider some experimental findings. Hofmann-Engl & Parncutt (1998) showed that keeping the transposition interval constant and varying the length of two melodic fragments (one to five tones), that similarity judgements increase with increasing length of the fragments. This seems to call for enveloping the components of the similarity vector by a exponential function giving more weight to earlier tones than later tones. Further, comparing two given m-chains of the length n, which are identical except one interval, Hofmann-Engl & Parncutt (1998) found that by varying the length n, that similarity judgements increase with increasing length n. Thus, a model will also have to be length sensitive. According to these researchers, tempo is not a factor in melotonic similarity, but appears as a rhythmic factor. We also might expect that aspects concerning the shape of two m-chains as covered by the interval vector will affected by the primacy/recency effect, where earlier and later tones are weighted more than are tones in the middle. This might call for enveloping the components of the interval vector by a Gauss distribution. Finally, a suitable model will require some empirical constants which will have to be determined through experimentation. However, at this point we might suggest a simple melotonic model of the form:

:

$$\left\|\vec{F}_1\right\| = \left\| \begin{pmatrix} \dfrac{e^{-\frac{k_1}{n^{c_1}}s_1^2}}{\sqrt{n}} \\ \dfrac{e^{-\frac{k_1}{n^{c_1}}s_2^2}}{\sqrt{n}} \\ \cdot \\ \cdot \\ \dfrac{e^{-\frac{k_1}{n^{c_1}}s_n^2}}{\sqrt{n}} \end{pmatrix} \right\| = \sqrt{\frac{\sum_{i=1}^{n}\left(e^{-\frac{k_1}{n^{c_1}}s_i^2}\right)^2}{n}} \quad \text{and} \quad \left\|\vec{F}_2\right\| = \left\| \begin{pmatrix} \dfrac{e^{-\frac{k_2}{(n-1)^{c_2}}I_1^2}}{\sqrt{n-1}} \\ \dfrac{e^{-\frac{k_2}{(n-1)^{c_2}}I_2^2}}{\sqrt{n-1}} \\ \cdot \\ \cdot \\ \dfrac{e^{-\frac{k_2}{(n-1)^{c_2}}I_n^2}}{\sqrt{n-1}} \end{pmatrix} \right\| = \sqrt{\frac{\sum_{i=1}^{n-1}\left(e^{-\frac{k_2}{(n-1)^{c_2}}I_i^2}\right)^2}{n-1}}$$

where $\left\|\vec{F}_1\right\|$ is the transpositional similarity predictor and $\left\|\vec{F}_2\right\|$ is the interval similarity predictor, k_1 and k_2 are empirical constants determining the strength of each interval component, c_1 and c_2 are empirical constants determining ho much the length of a chain affects similarity, s_1, s_2, ... s_n are the components of the similarity vector, I_1, I_2, ..., I_{n-1} are the components of the interval vector and n is the length of the compared m-chains.

An overall similarity could then be defined as:

$$S = \left\|\vec{F}_1\right\| \cdot \left\|\vec{F}_2\right\|$$

In fact, setting $c_1 = 1$ and $c_2 = -2$, we obtain a correlation of 87% with the data as produced by the two experiments as conducted by Hofmann-Engl & Parncutt (1998).

An overall similarity model taking rhythmic, dynamic and pitch features into account, might be of the form:

$$S = \alpha S_m + \beta S_d + \gamma S_r.$$

where α, β, γ are empirical constants, S_m as the pitch similarity, S_d as the dynamic similarity and S_r as the rhythmic similarity

This is not to say that this the most adequate model, but it is fashioned based on some available data, some theoretical concepts and is similar to Shepard's (1987) model. However, the model as it stands does not take into account any rhythmic or dynamic aspects nor does it pay tribute to harmonic features or emotional aspects. It also allows for the comparison of m-chains only which have equal length. Further, we might find that tones which are longer will bear more weight than shorter tones. Thus, as the model stands it might be only suitable as a predictor for short isochronous m-chains.

4. CONCLUSION

This paper set out to investigate an aspect of melodic similarity from a cognitive angle. We found that the term pitch is little satisfying and we argued for replacing it by the term meloton which was defined as a cognitive quality of sound. We further proposed that melotonic similarity is best approached by defining a set of transformations (reflections and translations). Based on the composition of two specific reflections we were able to define a similarity and interval vector which we propose to be somewhat sufficient to form the basis for a melotonic predictor. Specifically, we presented a simple similarity model which admittedly shows limitations but might demonstrate that more complex and comprehensive models can be developed. However, before a more comprehensive model will become available, many more experiments on melodic similarity will have to be conducted. Considering that we covered melotonic similarity only, we can by now conclude that the construction of sufficient models is a far more complex task than generally acknowledged, but at the same time it appears to be an achievable task.

5. REFERENCES

Adams, C. Melodic Contour Typology. Ethnomusicology, vol 20.2 (1976)

Anagnostopoulou, C., Hörnel, D. & Höthker, K. Investigating the Influence of Representations and Algorithms in Music Classification. Proceedings of the AISB '99 Symposium on Musical Creativity. (1999, Edinburgh)

Askill, J. The Subjective Pitch of Musical Chimes. (1997, Online publication).

http://faculty.millikin.edu/~jaskill.nsm.faculty.mu/chimes.html

Cambouropoulos, E. Melodic cue abstraction, similiarty and category formation:A computational appraoch. In Proceedings of ICMPC 2000. (2000, Keele University)

Crawford, T., Iliopoulos, C. & Raman, R. String-Matching Techniques for Musical Similarity and Melodic Recognition. In: Melodic Similarity - Concepts, Procedures, and Applications (ed. Hewlett, W. & Selfridge-Field, E.), Computing in Musicology 11. MIT Press, CA, 1998

Edworthy, J. Pitch and contour in music processing. Psychomusicology, 2, (1982), 44-46

Edworthy, J. Interval and contour in melody processing. Music Perception, 2, (1985), 375-388

Dovey, M. & Crawford, T. Heuristic Models of Relevance Ranking in Searching Polyphonic Music. In: Proceedings Diderot Forum on Mathematics and Music, (1999, Vienna, Austria), pp 111-123.

Dowling, W. J. Recognition of inversion of melodies and melodic contour. Perception & psychophysics, 12.5, (1971), 417-421

Dowling, W. J. & Harwood, D., (1986). Music Cognition, New York Academic Press, 1986

Downie, S., (1999). Evaluating a Simple Approach to Musical Information Retrieval: Conceiving Melodic N-Grams as Text. Ph.D. dissertation, University of Western Ontario

Egmond van, R. & Povel, D-J.. & Maris, E. The influence of height and key on the perceptual similarity of transposed melodies. Perception & Psychophysics, vol. 58, 1996, 1252-1259

Francés, R. The perception of music. (trans. Dowling), Hillsdale, New York, 1988

Goldstone, R. L. The Role of Similarity in Categorization: Providing a Groundwork. (1998, Online publication) http://cognitrn.psych.indiana.edu/rgoldsto/sim&cat.html, Indiana University

Hammel, B. Motivic Transformations & Lie Algebras. (1999, Online publication) http://graham.main.nc.us/~bhammel/Music/Liemotiv.html

Hofmann-Engl, L. J. Beiträge zur theoretischen Musikwissenschaft. M 65+1, vol. 2, 1989, Technical University Berlin

Hofmann-Engl, L. J. Virtual Pitch and pitch salience in contemporary composing.. In Preceedings the VI Brazilian Symposium on Computer Music at (1999, PUC Rio de Janeiro)

Hofmann-Engl, L. & Parncutt, R.. Computational modeling of melodic similarity judgments - two experimetns on isochronous melodic fragments. (1998, Online publication) http://www.chameleongroup.org.uk/research/sim.html

Hofmann-Engl, L. Review : Review of W.B. Hewlett & E. Selfridge-Field, eds., Melodic Similarity: Concepts, Procedures, and Applications.(Cambridge, Massachusetts: MIT Press, 1999). MTO 5.4, 1999, MIT CA

Houtsma, A J. M. & Goldstein, J. L. The central origin of the pitch of complex tones: Evidence from musical interval recognition. Journal of the Acoustical Society of America, vol. 51, 1972, 520-529

Isaacson, E. Issues in the Study of Similarity in Atonal Music. MTO 2.7, 1996, MIT CA

Kassler, M.. Toward Musical Information Retrieval. Perspectives of Music 4.2, 1966, 59-67

Kluge, R. Ähnlichkeitskriterien für Melodieanalyse. Systematische Musikwissenschaft, 4.1-2, 1996, 91-99

Lewin, D. Forte's Interval Vector, My Interval Function, and Regner's Common-note Function. Journal of Music Theory 21, 1977, 194-237

Maidin O, D. & Fernström, M. The Best of two Worlds: Retrieving and Browsing. Proceedings of COST G-6 on Digital Audio Effects, (2000, Verona)

Mazzola, G. Geometrie der Töne. Birkhäuser, Basel, 1990

Morris, R. A Similarity Index of Pitch-class Sets. Prespectives of New Music 18, 1979

Palmer, S. The psychology of perceptual organization. A transformational approach. Human and Machine Vision, New York Academic Press, 1983, 269-339.

Rasch, R. A. & Plomp, R.The Perception of Musical Tones. In: The Psychology of Music (e.g.. Deutsch), Academic Press, New York, 1982

Reti, R. The Thematic Process in Music. New York, Macmillan Company, 1951

Rohwer, J. Die harmonischen Grundlagen der Musik. Bärenreiter, Kassel, 1970

Schönberg, A. Fundamentals of Musical Composition. St. Martin's Press, New York, 1967

Schouten, J. F. The perception of subjective tones. Proceedings of the Koninklijke Nederlandse Akademie van Wetenschappen, 41, 1938, 1418-1424

Smith, L., McNab, J. & Witten, I. Sequence-Based Melodic Comparison: A Dynamic-Programming Approach. In: Melodic Similarity - Concepts, Procedures, and Applications (ed. Hewlett, W. & Selfridge-Field, E.), Computing in Musicology 11, 1998, MIT Press, CA

Sundberg, J.The Science of Musical Sound. Academic Press, New York, 1991

Tekman, H. G. A Multidimensional Study of Preference Judgments for Pieces of Music. Psychological Report, 82, 1998, 851-860

Terhardt, E Die Psychoakustischen Grundlagen der musikalischen Akkordgrundtöne und deren algorithmische Bestimmung. In: Tiefenstruktur der Musik, Technische Universität Berlin, Berlin, 1979

Terhardt, E. & Stoll, G. & Seewann, M. Algorithm for extraction of pitch and pitch salience from complex tonal signals. Journal of the Acoustical Society of America, vol. 71, 1982, 679-688

White, B. Recognition of distorted melodies. American Journal of Psychology, 73, 1960, 100-107

Building a Platform for Performance Study of Various Music Information Retrieval Approaches

Jia-Lien Hsu and Arbee L.P. Chen

Department of Computer Science
National Tsing Hua University
Hsinchu, Taiwan 300, R.O.C.

alpchen@cs.nthu.edu.tw

ABSTRACT
In this paper, we describe the Ultima project which aims to construct a platform for evaluating various approaches of music information retrieval. Three approaches with the corresponding tree-based, list-based, and (*n*-gram+tree)-based index structures are implemented. A series of experiments has been carried out. With the support of the experiment results, we compare the performance of index construction and query processing of the three approaches and give a summary for efficient content-based music information retrieval.

1. Introduction
With the growth of music objects available, it is getting more attention on the research of constructing music information retrieval systems. To provide an efficient and effective content-based retrieval of music objects, various approaches have been proposed in which the music representations, index structures, query processing methods, and similarity measurement are key issues.

Regarding the issue of music representation, several approaches are introduced to model various features of music content, such as pitch, rhythm, interval, chord, and contour. To efficiently resolving user queries, different kinds of techniques are proposed, including string matching methods, dynamic programming methods, *n*-gram indexing methods, and list-based and tree-based indexing structures with the corresponding traversal procedures. Most researchers present their solutions to the key issues separately. However, the research work focusing on the quantitative and qualitative comparison of various techniques used in music information retrieval is still limited.

On the contrary, in the traditional information retrieval, the problems and techniques involved in the evaluation of retrieval systems and procedures have been investigated. The most common evaluation criteria have also been identified, such as precision and recall, response time, user effort, form of presentation, and collection coverage [18].

Due to the multi-faceted properties of music, there exist intrinsic difficulties for content-based music information retrieval (MIR). The framework of a formal MIR evaluation mechanism becomes necessary. The point is emphasized in [7], which states "a formalized set of MIR evaluation standards must become part of the MIR researcher toolkit" and "a set of music test databases of substantial size and varied content must be formed so that all MIR researchers can properly compare and contrast techniques under a variety of scenarios."

From the database point of view, we initiate the project of building a platform for the evaluation of music information retrieval systems. Considering the retrieval efficiency and effectiveness, we focus on the performance study of music representations, indexing and query processing which involve a wide range of techniques used in content-based music information retrieval.

The rest of this paper is organized as follows. In Section 2, we describe our project for evaluating music information retrieval approaches. The issues of system design, data set, query set generation, and efficiency and effectiveness study are also introduced in this section. The three approaches implemented in our platform are described in Section 3. We perform a series of experiments and illustrate the experiment results and performance study in Section 4. Section 5 concludes this paper and points out our future directions.

1.1 Related Work
Selfridge-Field [19] provides a survey of clarifying and resolving conceptual and representational issues in melodic comparison. Research work on MIR systems are introduced as follows. Ghias, *et al.* [10] propose an approach for modeling the content of music objects. A music object is transformed into a string which consists of three kinds of symbols, 'U', 'D', and 'S' which represent a note is higher than, lower than, or the same as its previous note, respectively. The problem of music data retrieval is then transformed into that of approximate string matching.

In [1][6], a system supporting the content-based navigation of music data is presented. A sliding window is applied to cut a music contour into sub-contours. All sub-contours are organized as an index structure for the navigation. Tseng [20] proposes a content-based retrieval model for music collections. The system uses a pitch profile encoding for music objects and an *n*-gram indexing for approximate matching. A framework is also proposed in which the music objects are organized as an *n*-gram structure for efficient searching [22]. Different techniques of local alignment and local common subsequences have also been applied for comparison. Similar techniques of *n*-gram indexing have also been employed in [8][24][25]. Furthermore, Downie and Nelson [8] provide an effectiveness evaluation of an *n*-gram based MIR system by using statistical analysis.

The work [17] focuses on music retrieval from a digital library in which dynamic programming is used to match melodic phrases. The issues of melody transcription and matching parameters are discussed and the trade-off between the matching criteria and retrieval effectiveness is shown. Also using dynamic programming, Lemstrom and Perttu [14] present a bit-parallel algorithm for efficiently searching melodic excerpts. In the bit-parallel processing, the whole table for dynamic programming need not be created, and thus it leads to a better performance. Clausen, *et al.* [5] design a web-based tool for searching polyphonic music objects. The applied algorithm is a variant of the classic inverted file index for text retrieval. A prototype is implemented and its performance is investigated.

To develop a content-based MIR system, we have implemented a system called *Muse* [3][4][13]. In this system, various methods are applied for content-based music data retrieval. The rhythm, melody, and chords of a music object are treated as music feature strings and a data structure called *1D-List* is developed to efficiently perform approximate string matching [13]. Moreover, we consider music objects and music queries as sequences of chords [4] and *mubol* strings [3]. A tree-based index structure is developed for each approach to provide efficient matching capability. In [3], we propose an approach for retrieving music objects by rhythm. Instead of using only melody [1][4][6][10][13] or rhythm of music data, we consider both pitch and duration information plus the music contour, coded as *music segment*, to represent music objects [2]. Two index structures, called *one-dimensional augmented suffix tree* and *two-dimensional augmented suffix tree*, are proposed to speed up the query processing. By specifying the similarity thresholds, we provide the capability of approximate music information retrieval. When considering more than one feature of music objects for query processing, we propose multi-feature index structures [12]. With the multi-feature index, both exact and approximate search functions on various music features are provided.

2. The Ultima Project

The Ultima project is established with the goal to make a comprehensive and comparative assessment of various MIR approaches. Under the same environment and real data sets, a series of experiments can be performed to evaluate the efficiency and effectiveness of the MIR systems. Issues such as the threshold setting and the identification of most influential factors which dominates the system performance can be explored. Furthermore, heuristics for choosing appropriate representation schemes, indexing structures, and query processing methods when building an MIR system can be provided based on the performance study. The Ultima platform will be continuously maintained and served as the testbed whenever new approaches of content-based music information retrieval are proposed.

2.1 System Design and Implementation

The system is implemented as a web server, which runs on the machine of Intel Pentium III/800 with 1GB RAM on MS Windows 2000 by JDK 1.3. For posing queries at the client end, we provide the ways of humming songs, playing the piano keyword, uploading MIDI files, and using the computer keyboard and mouse. The server end consists of a mediator, four modules, and a data store, as shown in Figure 1. The mediator receives user queries and coordinates with other modules. The music objects and the corresponding information, such as title, composer, and genre, are organized as standard MIDI files and relational tables, respectively. The summarization module aims to resemble and visualize query results. The query generation module aims to generate parameterized user queries for performance evaluation, as discussed in Section 2.3. The implementations of the two modules are not finished yet. The report module aims to monitor and assess the performance of the system, such as the elapsed time of query processing, space of indices, and precision and recall of the retrieved results. The query processing module aims to resolve queries from the client end or the query generation module. The query processing module is designed as a "container" to which each query processing methods can be "plugged-in". Whenever a new method is proposed, it can be easily plugged into the module for performing experiments under the same environment. Currently, three methods are considered, *i.e.*, 1D-List [13], APS [2] and APM [3] which will be further discussed in Section 3.

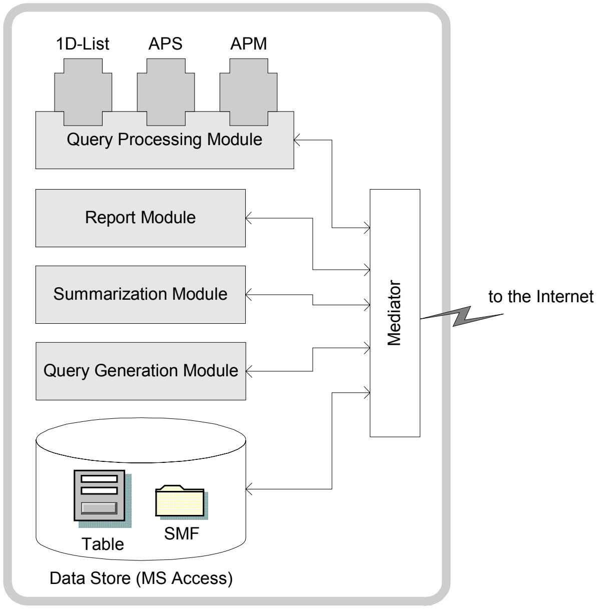


Figure 1: The function blocks of the server in the Ultima project.

2.2 Data Set

The testing data of music objects, from CWEB Technology, Inc., is a collection of 3500 single track and monophonic MIDI files. Most of them are pop music of Chinese and English songs in various genres.

The average object size is 328.05 notes. When coding these objects in the mubol and music segment representations, the average object size is 78.34 (mubol) and 272 (segment), respectively. Based on the statistics of the CWEB dataset, we estimate that one mubol corresponds to 4.19 notes, and one music segment corresponds to 1.21 notes. The *note count* is defined as the number of distinct notes appearing in a music object. According to the MIDI standard, the alphabet size is 128. Therefore, the note count of every melody string is between 1 and 128. For the CWEB data set, the average note count is 13.46. Due to the space limitation, the histograms of the object size and note count of the CWEB data set are skipped.

Moreover, when coding music objects by music segment, the distribution of segment pitch is shown in Figure 2. For the segment duration, most of its value is between 0 and 20 beats without any obvious clustering.

2.3 Query Set Generation

In the traditional information retrieval, there exist standard testing data, queries and the associated answers [9][23].

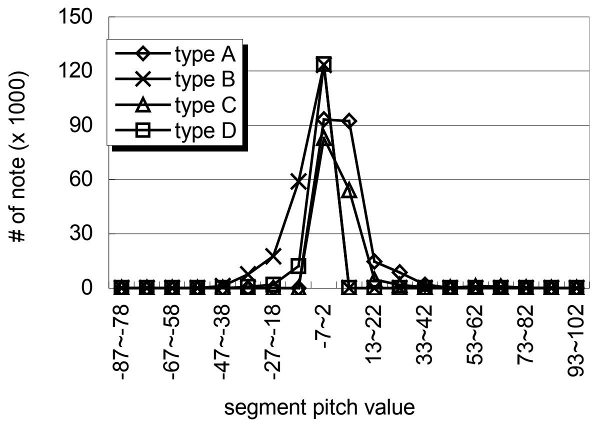


Figure 2: The distribution of segment pitch values of CWEB data set.

Therefore, a fair performance evaluation can be performed. However, there is no such kind of benchmarks dedicated to the MIR systems. In this project, we will also investigate a standard procedure for generating parameterized queries and the associated answers from a data set. With the variety of queries, the performance study will be more accurate.

2.4 Efficiency and Effectiveness Study

We design a series of experiments to evaluate the methods of indexing and query processing. Factors influencing the system performance are identified, such as query length, database size, and query approximation degree. The measurement of performance is based on memory usage, retrieved candidates and elapsed time for efficiency, and precision and recall for effectiveness.

3. Description of the Three Approaches

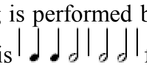
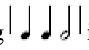
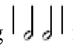
Instead of detailed procedures and algorithms, each approach is illustrated by an example to show the basic idea. The three approaches cover various methods of music representation, indexing, and query processing, as summarized in Table 1. Note that all the approaches support the functionality of exact, partial, and approximate matching. For simplicity, we only show an example of exact query processing.

Table 1: The representations and indexing structures of the three approaches.

Approach	Representation	Index structure
APM	Mubol (rhythm)	(*N*-gram+tree)-based
1D-List	Melody (pitch)	List-based
APS	Music segment (pitch+duration)	Suffix tree-based

3.1 The APM Approach

The rhythm information of music objects is coded as mubol strings. A mubol is a rhythmic pattern of a measure in a music object. A mubol string of a music object is the string of mubols which are determined by each measure of the music objects. For example, as shown in Figure 3(a), R1 is a mubol string of eight measures. The *n*-grams of R1, where *n* = 1, 2, 3, with the associated positions are listed in Figure 3(b). For example, the position "R1: 1,4,7" of the mubol in the first row of Figure 3(a) indicates the mubol appears in the first, forth, and seventh measure of R1. All the prefixes of an *n*-gram can be found in the (*n*−1)-grams, (*n*−2)-grams, …, and 1-grams. To efficiently process queries, the *n*-grams of mubol strings are organized as a tree structure, called L-tree. The tree height *h* of L-tree is the maximal *n*, i.e., *h* = 3 in our example. As shown in Figure 3(c), the nodes in level 1 of the tree indicate the first mubols of 1-grams, the nodes in level 2 indicate the second mubols of 2-grams, etc. Note that there are two kinds of links in L-tree, namely, solid link and dotted link. The internal nodes are connected with solid links, while the leaf nodes with the associated information are indicated by dotted links.

The exact query processing is performed by a tree traversal of the L-tree. Suppose the query is  for exact searching. The query will be processed by traversing the L-tree in level-wise manner. When processing the first mubol of the query at level 1 of the L-tree, the node containing  is matched and its children will be reached for processing the next mubol. When processing the second mubol at level 2, those children (only one node in this example) will be compared to the second mubol. Since the node containing  is matched to the second mubol of the query string, the two children of this node will be reached for further processing. Moreover, all the mubols of the query string have been processed and only (R1:2,5) is the answer.

The L-tree is a (*n*-gram+tree)-based index structure. In the approach of *n*-gram indexing, if the length of the query string is larger than *n*, the false match may happen. For the L-tree of tree height *h*, if the query length is larger than *h*, the query will be divided into subqueries and the intermediate answers with respective to each subquery will be merged and confirmed by the join processing.

3.2 The 1D-List Approach

In this approach, music objects are coded as melody strings. For example, there are two music objects M1 and M2 in the database. The melody strings of M1 and M2 are "so-mi-mi-fa-re-re-do-re-mi-fa-so-so-so" and "do-mi-so-so-re-mi-fa-fa-do-re-re-mi", respectively.

To support efficient string matching, melody strings are organized as linked lists, as shown in Figure 4(a). For the notes of the same pitch in the melody strings, they are linked in an increasing order. Each node in the linked lists is of the form (*x*:*y*) which denotes the *y*-th note of the melody string of the *x*-th music object in the database.

When a query Q = "do-re-mi" is specified, the lists involved in Q are retrieved with the two dummy nodes *start* and *end* as shown in Figure 4(b). Then, the exact query processing goes as follows. Let A_x be the first element of node *A*, and A_y be the second element of node *A*. For each pair of nodes (*A*, *B*) taken from two adjacent linked lists, if $A_x = B_x$ and $A_y+1 = B_y$, we build an exact link from *A* to *B*, as shown in Figure 4(c). Also, we build an exact link from *start* to node *F* of the first list if *F* has an outgoing link, and from node *L* of the last list to *end* if *L* has an incoming link. By traversing the exact links from *start* to *end*, each path indicates a substring appearing in the melody string of the database and will be considered as a result. In our example shown in Figure 4(c), there exists only one path, which is denoted in bold-faced links, *i.e.*, "*start*-(1:7)-(1:8)-(1:9)-*end*".

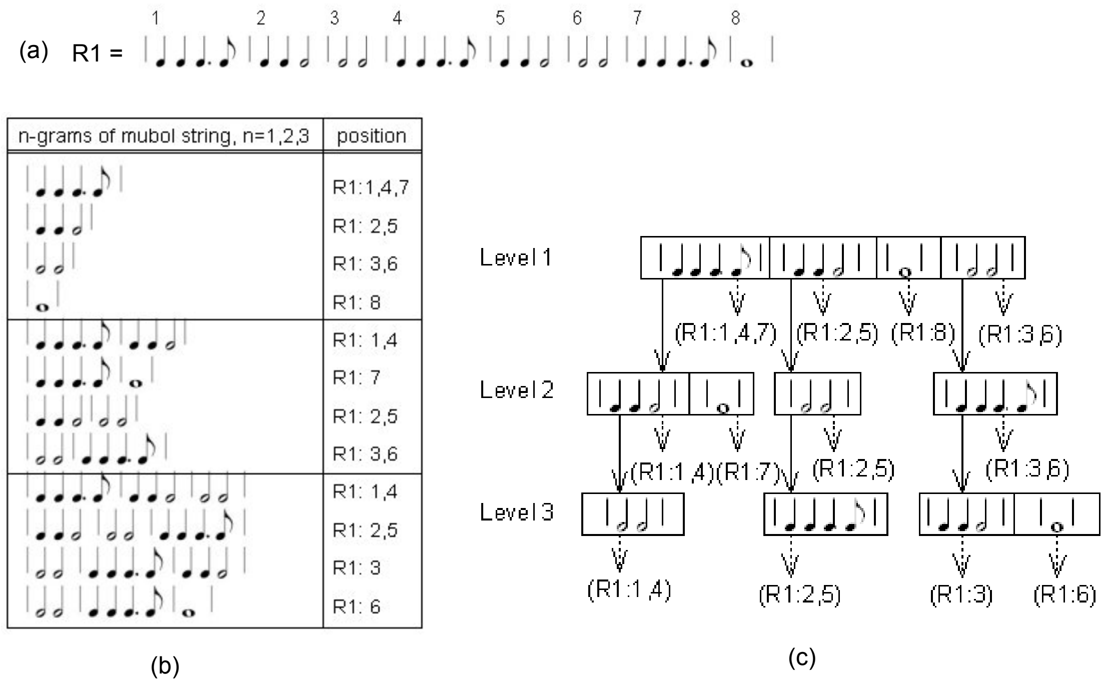
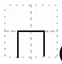
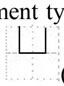
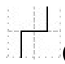
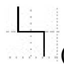


Figure 3: (a) A sample mubol string R1. (b) The table of *n*-grams associated with the corresponding positions. (c) The L-tree of the mubol string R1.

3.3 The APS Approach

For better readability, the representation, indexing, and query processing are separately described as follows.

3.3.1 Representation of Music Objects

Taking into account of music contour with note duration and pitch, the APS approach represents music objects by sequences of music segments. A *music segment* is a triplet which consists of the *segment type* and the associated duration and pitch information. There are four segment types defined to model the music contour, *i.e.*,  (type A),  (type B),  (type C), and  (type D). Define the *segment base* as the horizontal part of a music segment. The beat information of a music segment is represented by the *segment duration* which is the number of beats in the corresponding segment base. The pitch information of a music segment is represented by the *segment pitch* which is the *note number* in the MIDI standard of the corresponding segment base minus the note number of the segment base of the previous segment base. For example, for the piece of music shown in Figure 5, the corresponding representation as a sequence of music segments is shown in Figure 6. The music segment (A, 1, +1) indicates that it is a type A segment with the segment duration and segment pitch being 1 and +1, respectively. When coding by music segments, the first music segment and the last music segment are ignored due to lack of information to assign the segment type. Therefore, the music object of Figure 5 is represented by the sequence of (B,3,-3) (A,1,+1) (D,3,-3) (B,1,-2) (C,1,+2) (C,1,+2) (C,1,+1). In priori to describing the dedicated indexing structures for APS, we introduce a data structure named *suffix tree*. A suffix tree is originally developed for substring matching [11][15].

For example, Figure 7 shows the suffix tree of the string S = ”ABCAB”. Each leaf node (denoted by a box) corresponds to a substring starting at the position indicated in the node in S, and each link is labeled with a symbol α, where $\alpha \in \Sigma \cup \{\$\}$, Σ is the alphabet of S and ‘$’ is a special symbol denoting end-of-string. As a result, all the suffixes, *i.e.*, “ABCAB”, “BCAB”, “CAB”, “AB”, and “B”, are organized in the tree. For a query string, the matching processing is a typical tree traversal. For example, suppose that the query string is “AB”. We follow the leftmost path to the black node, and all leaf nodes descending from the black node are the results, *i.e.*, the first and the forth position.

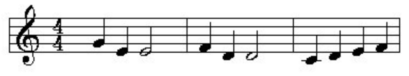


Figure 5: A piece of music.

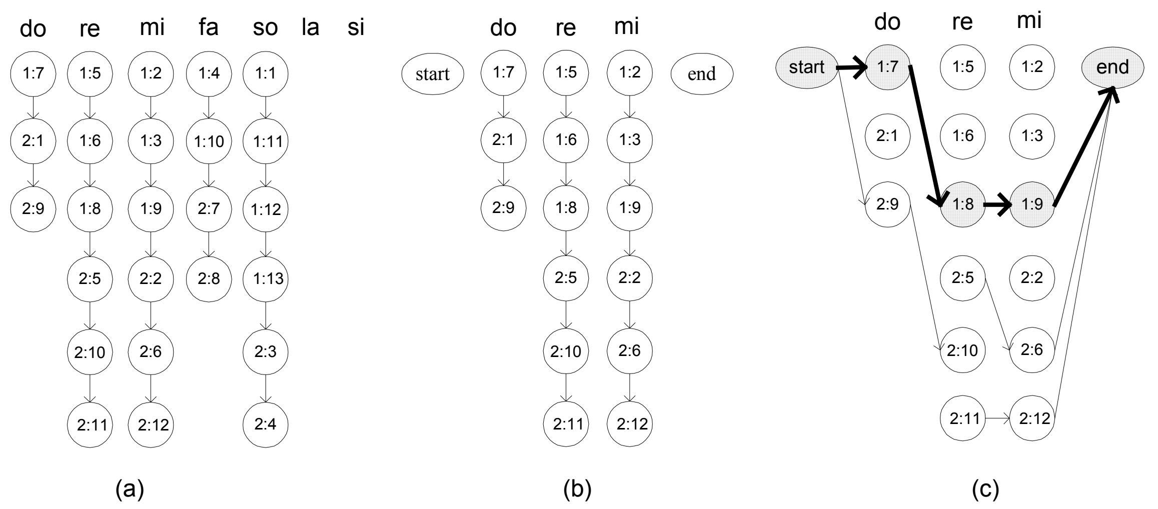


Figure 4: (a) The index structure of M1 and M2 for the 1D-List approach. (b) An example of exact query Q = "do-re-mi". (c) The exact link and result of the query Q.

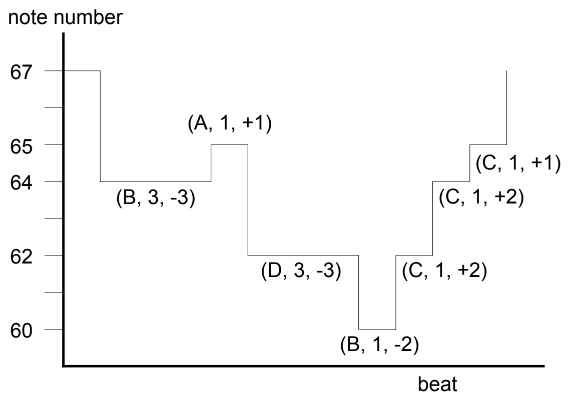


Figure 6: The corresponding sequence of music segments of the music object in Figure 5.

3.3.2 Index Structures for Sequences of Music Segments

In the following, we introduce two index structures for efficiently processing queries of music segments, *i.e.*, the one-dimensional and two-dimensional augmented suffix trees.

A one-dimensional augmented suffix tree (1-D AST, in short) is a suffix tree with the segment duration information being added to the edges. First, a suffix tree based on the sequences of the segment types is constructed. Each edge of the suffix tree refers to a symbol appearing in one or more positions in the sequence. For example, let the sequence of music segments be (A,2,+1) (B,5,-1) (C,1,+1) (A,3,+1) (B,3,-2). Using only the segment types, the suffix tree can be constructed as shown in Figure 7. The bold-faced edge in Figure 7 refers to the 'B' in the second and fifth position. Since the corresponding segment durations are 5 and 3, we attach the range of segment duration <*min*, *max*> = <3, 5> to the edge. This range can be used to filter out some results which cannot be answers during query processing. Figure 8 shows an example of a 1-D AST.

To exploit the filtering effect, the range <*min, max*> should be as compact as possible. For a given population of segment durations, such as {1, 2, 2, 3, 7, 8, 8}, two ranges <1, 3> and <7, 8> are better than one range <1, 8>. Thus, the edge should be split into two edges labeled with <1, 3> and <7, 8>. This method is called *dynamic splitting*. In some cases, however, if it is hard to find compact ranges from a given population, we may apply *static splitting* method by splitting a range into some predefined smaller ranges which can be obtained from the statistics of data set.

The 2-D AST is an extension of the 1-D AST by attaching both segment duration and segment pitch information to the edge.

3.3.3 Query Processing

The query processing for the augmented suffix tree is called the *thresholding-based matching*, which is able to deal with both exact and approximate queries [2]. The approximation degree of the query is specified by means of thresholds. The exact queries can be considered as a special case with the thresholds being set to zero. For ease of illustration, we only show the processing of exact queries in the following.

Based on the 1-D AST in Figure 8(b), given the query Q = (A,1,–) (C,2,–) (A,5,–), we find the music objects whose sequences of music segments contain Q. When processing queries against a 1-D AST, the segment pitch in the queries is not needed and denoted by '–'.

The tree traversal starts from the root node and goes as follows. When processing the first music segment (A,1,–), the edge A<1, 1> is matched such that we reach the node N_1. Then, when processing the second music segment (C,2,–), the edge C<1, 3> is satisfied because the duration of the music segment is covered by the range of the edge. For the last music segment (A,5,–), although the segment type of the two edges from node N_2 is matched, the two edges are filtered out because the duration of the music segment is not covered by any ranges of the edges. Therefore, the processing terminates without any answer. Note that the results derived from this tree traversal are not necessary the answers to the query. Further verification of the results is required.

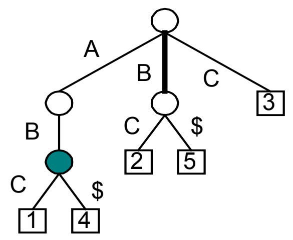


Figure 7: The suffix tree of the string S = "ABCAB".

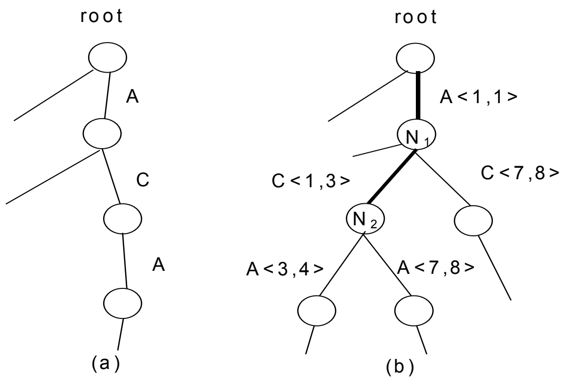


Figure 8: (a) An example of suffix tree. (b) The 1-D augmented suffix tree.

4. The Efficiency Study

In this section, we show the experiment results on the efficiency of the three approaches described in Section 3. For the APS approach, both 1-D AST and 2-D AST are implemented. For comparison, we also construct a suffix tree, denoted as ST, based on the segment types of music segment sequences.

4.1 Index Construction

The elapsed time and memory usage for constructing indices of the three approaches are illustrated as follows. For the APM approach, the tree height of L-tree is set to 6 in our experiments. As shown in Figure 9 and Figure 10, both the elapsed time and the memory usage of 1D-List are less than those of L-tree. This is because the construction of 1D-List is a simple process of transforming the melody strings to the linked lists, and the number of nodes in the 1D-List is linear to the database size.

The suffix tree-like data structures including the augmented suffix trees in the APS approach suffer from the space consumption. It is not reasonable to construct a full and complete augmented suffix tree just for handling the rare cases of extremely long-length queries. On the contrary, an augmented suffix tree with longer tree heights is beneficial to the efficiency of query processing. In our experiments, the tree height of augmented suffix tree is set to 4, 6, 8, 10, and 12. We construct three indices of APS, *i.e.*, ST, 1-D AST, and 2-D AST. As described in Section 3.3, we apply the static splitting method to divide the domain of duration into three ranges and the domain of pitch into two ranges. Obviously, the elapsed time and memory usage of the three indices ST, 1-D AST, and 2-D AST are increasing. We only show the construction of the 2-D AST in Figure 11 and Figure 12, where 'h_4' indicates the tree height of four, 'h_6' indicates tree height of six, and so on. The elapsed time and memory usage in the cases of smaller tree heights are much less than the cases of larger tree heights.

4.2 Exact Query Processing

In the following, we discuss the efficiency of processing exact queries for the APM, 1D-Lst, and APS approaches. Factors of query length, number of objects, and tree height of indices of APS will be investigated.

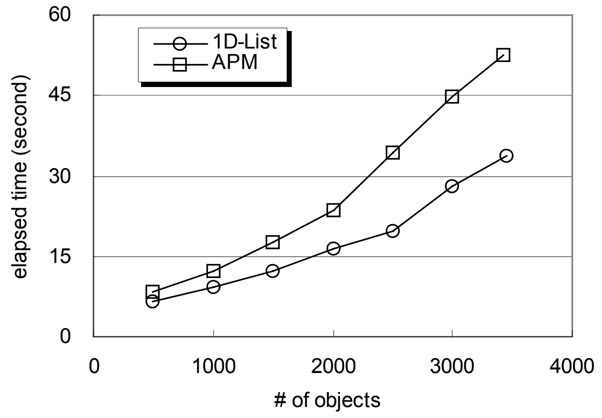


Figure 9: Elapsed time vs. # of objects for index construction of APM (L-tree, h=6) and 1D-List.

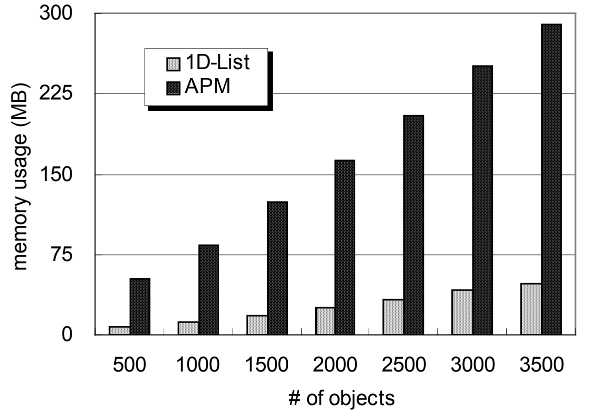


Figure 10: Memory usage vs. # of objects for index construction of APM (L-tree, h=6) and 1D-List.

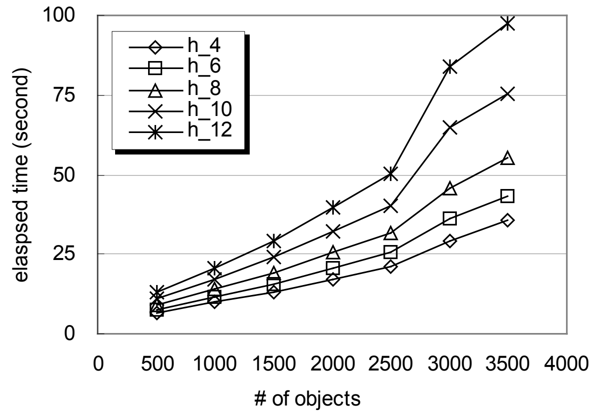


Figure 11: Elapsed time vs. # of objects for index construction of APS (2-D AST).

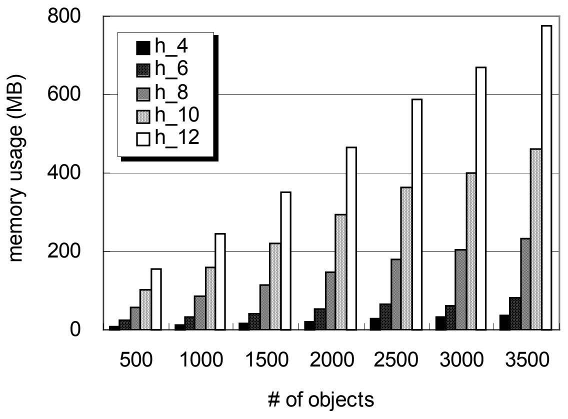


Figure 12: Memory usage vs. # of objects for index construction of APS (2-D AST).

For the APM and 1D-List approaches, the factors of query length and number of objects are explored, as shown in Figure 13, Figure 14, and Figure 15. Figure 13 shows the scalability of two approaches. The query length for 1D-List, denoted by |Qn|, is of twelve notes. Accordingly, the query length for APM, denoted by |Qm|, is of three mubols. Compared to the APM approach, the 1D-List approach scales well as the number of music objects.

Figure 14 and Figure 15 show the elapsed time versus query length of APM and 1D-List, where 'obj_0.5K' indicates five hundred music objects in the experiment, 'obj_1.0K' indicates one thousand objects, and so on. For the APM approach, as shown in Figure 14, the elapsed time decreases rapidly when processing queries of length from one to six. As processing queries of length seven, the elapsed time rises up substantially. In the experiment setting, the tree height of L-tree is six. As in Section 3.1, if the query is of length seven, it will be divided into two subqueries. As a result, two times of the L-tree traversal are required. In addition, the join processing also contributes extra elapsed time. For the queries of length from seven to thirteen, similar behavior can be observed. When processing queries of length from seven to twelve, the elapsed time decreases. As processing the queries of length thirteen, the elapsed time rises up again, and so on. For the 1D-Lsit approach, Figure 15 shows the elapsed time versus the query length. The elapsed time increases slightly for query lengths ranging from 1 to 10, and remains almost the same for longer queries. Since only the lists involved in the query are retrieved for building exact links, the elapsed time is linear to the query length.

The elapsed time consists of the time for building links and traversing links. When dealing with shorter queries, the number of lists to be processed is small and the elapsed time increases slightly. When dealing with longer queries, although the number of lists to be processed increases, the number of answers to the query dramatically reduces such that the elapsed time remains almost the same. In our experiment, the number of answers is less than 2 for the query of lengths ranging from 16 to 64.

For APS, factors of query length, number of objects, and tree height of the three indices are explored as follows.

Figure 16 shows the scalability of APS with 1-D AST and 2-D AST of tree height of eight. The APS with ST is not included because of a much larger elapsed time under the same condition. Compared to the 1-D AST, the 2-D AST performs well as the number of objects increases.

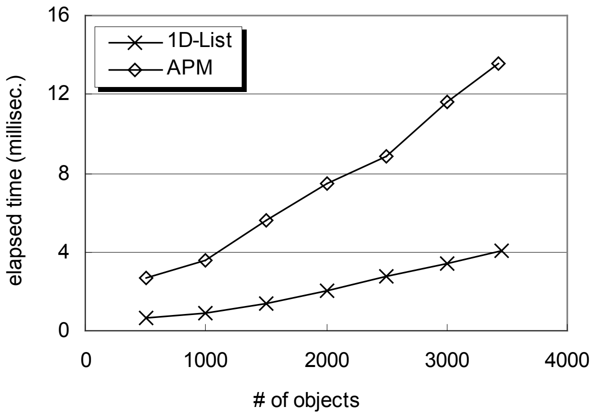


Figure 13: Elapsed time vs. # of objects for query processing of APM (|Qm|=3, L-tree, h=6) and 1D-List (|Qn|=12).

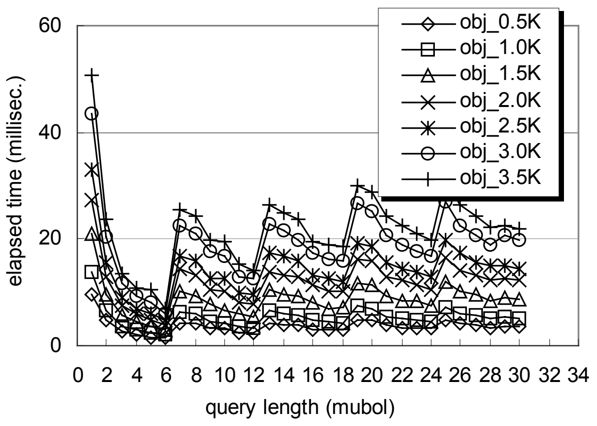


Figure 14: Elapsed time vs. query length for query processing of APM (L-tree, h=6).

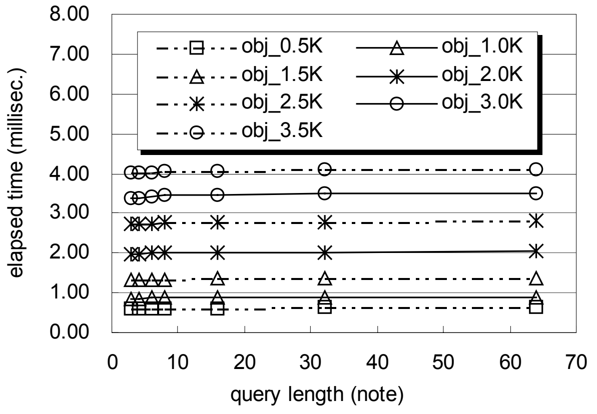


Figure 15: Elapsed time vs. query length for query processing of 1D-List.

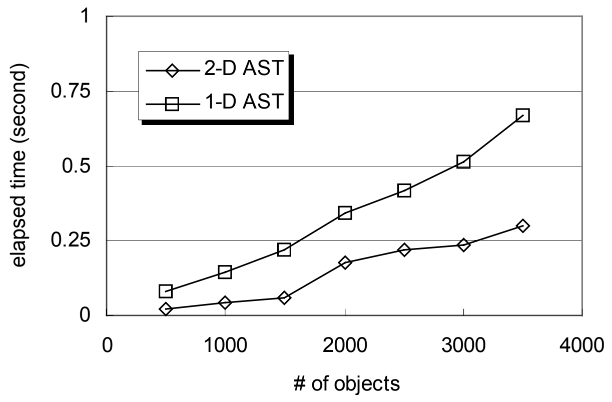


Figure 16: Elapsed time vs. # of objects for query processing of APS (1-D AST and 2-D AST, |Qs|=8, h=8).

Figure 17, Figure 18, and Figure 19 show the elapsed time versus query length for ST, 1-D AST, and 2-D AST, respectively. The curves in Figure 19 have a similar trend to the curves in Figure 14. However, for shorter queries ranging from one to eight music segments, such kind of trend is not obvious in APS. Two reasons are given as follows. In APM, leaf nodes are regarded as results, while leaf nodes of APS are just candidates for further confirmation. In addition, the number of leaf nodes retrieved in APS is much more than the one in APM. For example, after the tree traversal, there are four leaf nodes for four-mubol queries in APM, while there are 16968, 5429, 105 nodes for four-segment queries in APS with the index of ST, 1-D AST, and 2-D AST of tree height twelve, respectively. Post processing of a large number of candidates results in extra computation which smoothes the curves.

The total elapsed time of query processing in APS consists of three parts, *i.e.*, tree traversal, joining processing (if the query length is longer than the tree height), and post processing (for similarity computation). Among the three parts, the post processing consumes most of the elapsed time. For example, with the 2-D AST of tree height ten, the total elapsed time of processing a ten-segment query is 811 milliseconds, where 10 milliseconds for tree traversal and 801 milliseconds for computing similarity. When processing queries whose length is longer than the tree height, the query will be divided into subqueries. The number of candidates will be reduced after the joining processing. However, our database of 3500 music objects is only of moderate size. No matter what the tree height is, the number of candidates does not change much. Therefore, the difference of the performance with various tree heights is not obvious in our experiments, as shown in Figure 17, Figure 18, and Figure 19. We believe that, when dealing with much more music objects in databases, the influence of tree height will be revealed.

For comparison, we show the elapsed time for different indices in Figure 20. The performance gain of 2-D AST is obvious because of substantial edge pruning and candidate reduction.

In the following, we show the filtering effect of APS by applying 1-D AST and 2-D AST. The number of candidates is the number of leaf nodes retrieved after tree traversal. The filtering effect is measured by the *candidate reduction rate* (*CRR*), which is defined as the ratio of the number of reduced candidates using 1-D AST or 2-D AST to the number of candidates using ST.

$$CRR = \frac{N_{ST} - N_{AST}}{N_{ST}}$$

where N_{ST} denotes the number of candidates by applying ST and N_{AST} denotes the one by 1-D AST or 2-D AST.

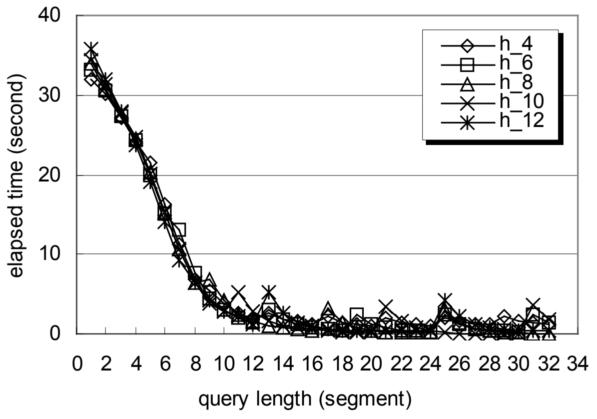


Figure 17: Elapsed time vs. query length for query processing of APS (ST, 3.5K objects).

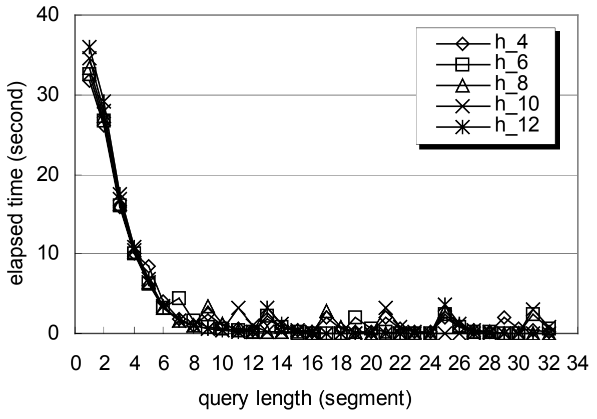


Figure 18: Elapsed time vs. query length for query processing of APS (1-D AST, 3.5K objects).

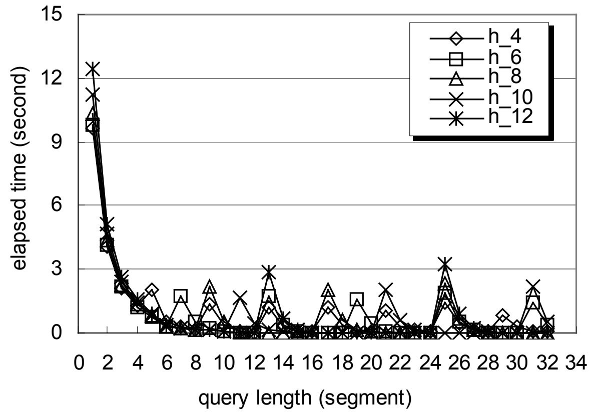


Figure 19: Elapsed time vs. query length for query processing of APS (2-D AST, 3.5K objects).

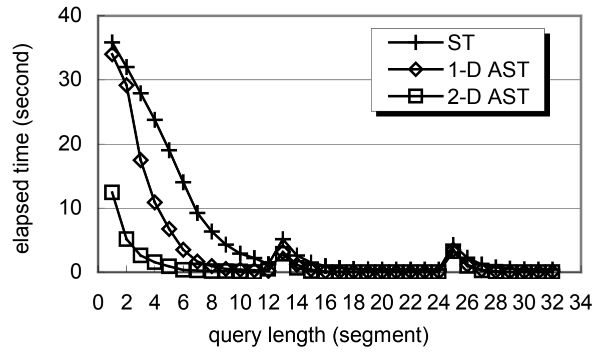


Figure 20: Elapsed time vs. query length for comparison of query processing of APS using various indices (h=12, 3.5K objects).

Higher reduction rates suggest better filtering effects. As shown in Figure 21, there are two kinds of curves with respect to the corresponding y-axis. The 'ST', '1-D AST', and '2-D AST' indicate the number of candidates applying the corresponding indices. The 'R1D' and 'R2D' indicate the CRR of the corresponding indices.

For the 1-D AST, the CRR increases as the query length is less than 14, while the ratio decreases as the query length ranges from 15 to 32. For the 2-D AST, since there are much fewer candidates, the CRR for the query lengths ranging from 1 to 24 is at least 80%. For the longer queries, the CRR is decreased to 67%.

For shorter queries, the APS approaches with 1-D AST and 2-D AST get benefits through attaching the beat and pitch information. However, for longer queries, all the methods have fewer candidates such that the filtering effect decreases slightly. For example, as the query lengths range from 24 to 32, the number of candidates for ST, 1-D AST, and 2-D AST is 3, 1, and 1, respectively. In general, the filtering effect of 2-D AST is better than that of 1-D AST. Moreover, a significant reduction of the candidates can be achieved using our approaches as the query length is less than 14.

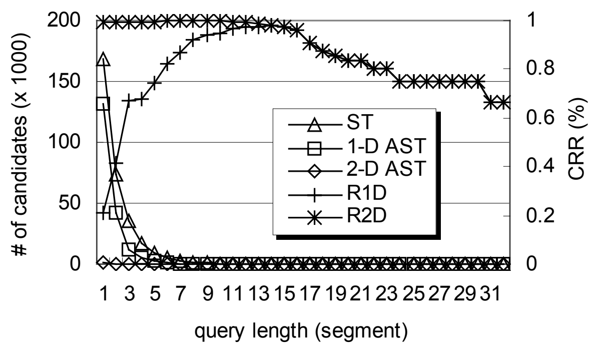


Figure 21: Reduction rate vs. query length for comparison of query processing of APS using various indices (h=12, 3.5K objects).

4.3 Summary of the Experiment Results

Following the comprehensive illustrations of the performance with respect to each approach, we summarize the experiment results for a comparison in Table 2. Four sets of query lengths for query processing are selected, *i.e.*, 1, 2, 3, and 4 mubols for APM, 4, 7, 10, and 12 notes for 1D-List, and 4, 8, 12, and 14 music segments for APS.

For reference, we also implement the string matching methods, namely, STR_MAT_n, and STR_MAT_ms. STR_MAT_n is a standard string matching method using the `indexOF` function in Java, which can be used to compare melody strings. On the other hand, STR_MAT_ms is for comparing sequences of music segments, which match segment types, followed by a checking for segment duration and segment pitch.

We summarize the experiment results as follows.

First, the 1D-List approach is superior in terms of indexing and query processing. However, the melody string of 1D-List approach is coded as the string of pitch values (*i.e.*, the note number in MIDI standard). If the MIR system is designed for end users and the query approximation is one of major concerns, 1D-List may not result in good effectiveness. If it is the case of exact searching from the bibliographic catalog, the 1D-List approach is suggested.

Second, the APM outperforms the APS family. Two reasons are given as follows. The APS family needs an extra cost for post processing. In addition, the average number of branches of a tree node in L-tree is much more than that of AST. It results in fewer candidates of APM. Therefore, the elapsed time of APS family is more than that of APM.

Third, constructing indices for the APS family is not always beneficial to query processing, especially when the query length is less than four music segments. For longer query lengths, the performance of 2-D AST is impressive, as shown in Figure 20 and Table 2. In addition, the performance difference between the 2-D AST with various tree heights is limited, as shown in Figure 17, Figure 18, and Figure 19. Therefore, for the APS family, we suggest using the 2-D AST of smaller tree heights. This is because the index size of 2-D AST substantially reduces when the tree height is smaller. For example, as shown in Figure 12, the index size of 2-D AST of tree height 12 is 774.46 MB, while that of tree height 10 is 461.57 MB.

5. Conclusion

In this paper, we describe the Ultima project which aims at building a platform for evaluating the performance of various approaches for music information retrieval. The issues of system design, query set generation, and performance study are discussed. The list-based, tree-based, (*n*-gram+tree)-based approaches are considered. Concerning the efficiency study, a series of experiments are conducted. The factors of database size, query length, tree height are investigated. We also provide a comparative study and summarization of the three approaches.

Future work include a performance evaluation of retrieval effectiveness among these approaches. Also, various input methods, the summarization module, and the query generation module will be implemented. The dynamic programming-based approaches, which are not covered in this project yet, will be considered in the next stage. While more and more polyphonic music retrieval methods are proposed, we also plan to extend our project to build a database of polyphonic music objects for evaluating these methods.

Table 2: The comparison of various approaches.

Approach (|DB| = 3500)	Index		Exact query processing[(1)(2)] (millisec.)				
	Size (MB)	Time (sec.)	|Qm| = 1 mubol |Qs| = 4 notes |Qn| = 4 segments	|Qm| = 2 |Qs| = 7 |Qn| = 8	|Qm| = 3 |Qs| = 10 |Qn| = 12	|Qm| = 4 |Qs| = 12 |Qn| = 14	In average
APM (L-tree, h=6)	289.0	52.5	50.6	23.8	13.6	10.1	24.5
1D-List	48.3	33.7	4.0	4.0	4.1	4.0	4.0
STR_MAT_n	N/A	N/A	861.0	852.0	852.0	851.0	854.0
APS (ST, h=12)	48.3	39.9	23767.0	9239.0	2899.0	1271.0	9294.0
APS (1-D AST, h=12)	290.7	54.0	10882.0	1630.0	416.0	195.0	3280.1
APS (2-D AST, h=12)	774.5	90.0	1570.0	244.0	96.0	9.0	479.8
STR_MAT_ms	N/A	N/A	2974.0	2814.0	2794.0	2814.0	2849.0

Note:
(1) Qn, Qm, Qs indicate that queries are coded as melody strings for the 1D-List approach, mubol strings for the APM approach, and sequences of music segments for the APS approach, respectively.
(2) |Qn|, |Qm|, and |Qs| indicate the length of queries in note, mubol, and music segment, respectively.

Acknowledgment

We would like to thank the CWEB Technology, Inc., for sharing us the data set used in our experiments.

References:

[1] Blackburn, S. & DeRoure, D. (1998). A tool for content-based navigation of music. In Proc. of ACM Multimedia.

[2] Chen, A. L. P., Chang, M., Chen, J., Hsu, J. L., Hsu, C. H. & Hua, S. Y. S. (2000). Query by music segments: An efficient approach for song retrieval. In Proc. of IEEE Intl. Conf. on Multimedia and Expo (ICME). New York.

[3] Chen, J. C. C. & Chen, A. L. P. (1998). Query by rhythm: An approach for song retrieval in music databases. In Proc. of the 8th Intl. Workshop on Research Issues in Data Engineering, (pp. 139-146).

[4] Chou, T. C., Chen, A. L. P., & Liu, C. C. (1996). Music databases: Indexing techniques and implementation. In Proc. of IEEE Intl. Workshop on Multimedia Data Base Management System.

[5] Clausen, M., Engelbrecht, R., Mayer, D. & Smith, J. (2000). PROMS: A web-based tool for searching in polyphonic music.

[6] DeRoure, D. & Blackburn, S. (2000). Content-based navigation of music using melodic pitch contours. Multimedia Systems, 8(3), Springer. (pp. 190-200).

[7] Downie, S. (2000). Thinking about formal MIR system evaluation: Some prompting thoughts. Available on http://www.lis.uiuc.edu/~jdownie/mir_papers/downie_mir_eval.html.

[8] Downie, S. & Nelson, M. (2000). Evaluation of a simple and effective music information retrieval method. In Proc. of ACM SIGIR, (pp. 73-80).

[9] Frakes, W. B. & Baeza-Yates, R. (1992). Information retrieval: Data structures and algorithms, Prentice-Hall.

[10] Ghias, A., Logan, H., Chamberlin, D., & Smith, B. C. (1995). Query by humming: Musical information retrieval in an audio database. In Proc. of ACM Multimedia, (pp. 231-236).

[11] Gusfield, D. (1997). Algorithms on strings, trees, and sequences. Cambridge University Press.

[12] Lee, W. & Chen, A. L. P. (2000). Efficient multi-feature index structures for music data retrieval. In Proc. of SPIE Conference on Storage and Retrieval for Image and Video Databases.

[13] Liu, C. C., Hsu, J. L., & Chen, A. L. P. (1999). An approximate string matching algorithm for content-based music data retrieval. In Proc. of Intl. Conference on Multimedia Computing and Systems (ICMCS'99).

[14] Lemstrom, K. & Perttu, S. (2000). SEMEX: An efficient music retrieval prototype. In Proc. of Intl. Symposium on Music Information Retrieval.

[15] McCreight, E. M. (1976). A space economical suffix tree construction algorithm. Journal of Assoc. Comput. Mach., 23, 262-272.

[16] MIDI Manufactures Association (MMA), MIDI 1.0 Specification, http://www.midi.org/.

[17] McNab, R. J., Smith, L. S., Witten, I. H., & Henderson, C. L. (2000). Tune retrieval in the multimedia library. Multimedia Tools and Applications, 10(2/3), Kluwer Academic Publishers.

[18] Salton, G. & McGill, M. (1983). Introduction to modern information retrieval. MaGraw-Hill Book Company.

[19] Selfridge-Field, E. (1998). Conceptual and representational issues in melodic comparison. In Hewlett, W. B. & Selfridge-Field E. (Ed.), Melodic similarity: Concepts, procedures, and applications (Computing in Musicology: 11), The MIT Press.

[20] Tseng, Y. H. (1999). Content-based retrieval for music collections. In Proc. of ACM SIGIR.

[21] Uitdenbogerd, A. & Zobel, J. (1998). Manipulation of music for melody matching. In Proc. of the 6th ACM Intl. Multimedia Conference, (pp. 235-240).

[22] Uitdenbogerd, A. & Zobel, J. (1999). Melodic matching techniques for large music databases. In Proc. of the 7th ACM Intl. Multimedia Conference, (pp. 57-66).

[23] Witten, I. H., Moffat, A., & Bell, T. C. (1994). Managing gigabytes: compressing and indexing documents and images, International Thomson Publishing company.

[24] Yanase, T. & Takasu, A. (1999). Phrase based feature extraction for musical information retrieval. In Proc. of IEEE Pacific Rim Conf. on Communications, Computers, and Signal Processing.

[25] Yip, C. L. & Kao, B. (2000). A study on *n*-gram indexing of musical features. In Proc. of IEEE ICME.

Efficient Multidimensional Searching Routines for Music Information Retrieval

Josh Reiss
Department of Electronic Engineering,
Queen Mary, University of London
Mile End Road, London E14NS
UK
+44 207-882-7986
josh.reiss@elec.qmw.ac.uk

Jean-Julien Aucouturier
Sony Computer Science Laboratory
6, rue Amyot,
Paris 75005,
France
+33 1-44-08-05-01
jjaucouturier@caramail.com

Mark Sandler
Department of Electronic Engineering,
Queen Mary, University of London
Mile End Road, London E14NS
UK
+44 207-882-7680
mark.sandler@elec.qmw.ac.uk

ABSTRACT

The problem of Music Information Retrieval can often be formalized as "searching for multidimensional trajectories". It is well known that string-matching techniques provide robust and effective theoretic solutions to this problem. However, for low dimensional searches, especially queries concerning a single vector as opposed to a series of vectors, there are a wide variety of other methods available. In this work we examine and benchmark those methods and attempt to determine if they may be useful in the field of information retrieval. Notably, we propose the use of *KD-Trees* for multidimensional near-neighbor searching. We show that a KD-Tree is optimized for multidimensional data, and is preferred over other methods that have been suggested, such as the *K-Tree*, the *box-assisted sort* and the *multidimensional quick-sort*.

1. MULTIDIMENSIONAL SEARCHING IN MUSIC IR

The generic task in Music IR is to search for a query pattern, either a few seconds of raw acoustic data, or some type of symbolic file (such as MIDI), in a database of the same format.

To perform this task, we have to encode the files in a convenient way. If the files are raw acoustic data, we often resort to a *feature extraction* (fig. 1). The files are cut into M time frames and for each frame, we apply a signal-processing transform that outputs a vector of *n* features (e.g. psychoacoustics parameters such as pitch, loudness, brightness, etc…). If the data is symbolic, we similarly encode each symbol (e.g. each note, suppose there are M of them) with an *n*-dimensional vector (e.g. pitch, duration). In both cases, the files in the database are turned into a trajectory of M vectors of dimension *n*.

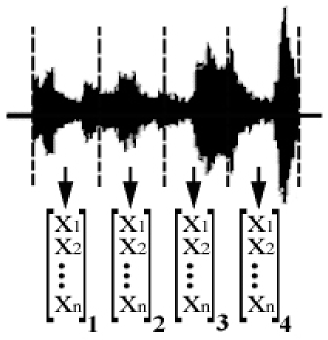


Figure 1- Feature extraction

Within this framework, two search strategies can be considered:

- String-matching techniques try to align two vector sequences of length $m \ll \mathrm{M}$, $(\vec{x}(1),\vec{x}(2),...\vec{x}(m))$ and $(\vec{y}(1),\vec{y}(2),...\vec{y}(m))$ using a set of elementary operations (substitutions, insertions…). They have received much coverage in the Music IR community (see for example [1]) since they allow a context-dependent measure of similarity and thus can account for many of the high-level specificities of a musical query (i.e., replacing a note by its octave shouldn't be a mismatch). They are robust and relatively fast.

- Another approach would be to "fold" the trajectories of *m* vectors of dimension *n* into embedded vectors of higher dimension $\mathrm{N} = m \cdot n$. For example, with *m*=3 and *n*=2:

$$\left(\vec{x}(1),\vec{x}(2),..\vec{x}(m)\right) = \left(x_1(1), x_2(1), x_1(2), x_2(2), x_1(3), x_2(3)\right)$$

The search problem now consists of identifying the nearest vector in a multidimensional data set (i.e., the database) to some specified vector (i.e., the query). This approach may seem awkward, because

- We lose structure in the data that could be used to help the search routines (e.g., knowledge that $x_1(1)$ and $x_1(2)$ are coordinates of the same "kind").
- We increase the dimensionality of the search.

However, there has been a considerable amount of work in devising very efficient searching and sorting routines for such multidimensional data. A complete review of the multidimensional data structures that might be required is described by Samet, et al. [2,3]. Non-hierarchical methods, such as the use of grid files [4] and extendable hashing [5], have been applied to multidimensional searching and analyzed extensively. In many areas of research, the KD-Tree has become accepted as one of the most efficient and versatile methods of searching. This and other techniques have been studied in great detail throughout the field of computational geometry [6,7].

Therefore, we feel that Music IR should capitalize on these well-established techniques. It is our hope that we can shed some light on the beneficial uses of *KD-Trees* in this field, and how the multi-dimensional framework can be adapted to the peculiarities of music data.

The paper is organized as follows. In the next four sections, we review four multidimensional searching routines: The KD-Tree, the K-Tree, the Multidimensional Quick-sort, which is an original algorithm proposed by the authors, and the Box-Assisted Method. Discussion of each of these methods assumes that the data consists of M N-dimensional vectors, regardless of what each dimension represents or how the vectors were created or extracted. We then benchmark and compare these routines, with an emphasis on the very efficient KD-Tree algorithm. Finally, we examine some properties of these algorithms as regards a multidimensional approach to Music IR.

2. THE KD-TREE

2.1 Description

The K-dimensional binary search tree (or KD-Tree) is a highly adaptable, well-researched method for searching multidimensional data. This tree was first introduced and implemented in Bentley, et al. [8], studied extensively in [9] and a highly efficient and versatile implementation was described in [10]. It is this second implementation, and variations upon it, that we will be dealing with here.

There are two types of nodes in a KD-Tree, the terminal nodes and the internal nodes. The internal nodes have two children, a left and a right son. These children represent a partition along a given dimension of the N-dimensional hyperplane. Records on one side of the partition are stored in the left sub-tree, and on the other side are records stored in the right sub-tree. The terminal nodes are buckets which contain up to a set amount of points. A one- dimensional KD-Tree would in effect be a simple quick-sort.

2.2 Method

The building of the KD-Tree works by first determining which dimension of the data has the largest spread, i.e. difference between the maximum and the minimum. The sorting at the first node is then performed along that dimension. A quickselect algorithm, which runs in order M time for M vectors, finds the midpoint of this data. The data is then sorted along a branch depending on whether it is larger or smaller than the midpoint. This succeeds in dividing the data set into two smaller data sets of equal size. The same procedure is used at each node to determine the branching of the smaller data sets residing at each node. When the number of data points contained at a node is smaller than or equal to a specified size, then that node becomes a bucket and the data contained within is no longer sorted.

Consider the following data:

A (7,-3); B (4,2); C (-6,7); D (2,-1); E (8,0);
F (1,-8); G (5,-6); H (-8,9); I (9,8); J (-3,-4);

Fig. 2 depicts the partition in 2 dimensions for this data set. At each node the cut dimension (X or Y) and the cut value (the median of the corresponding data) are stored. The bucket size has been chosen to be one.

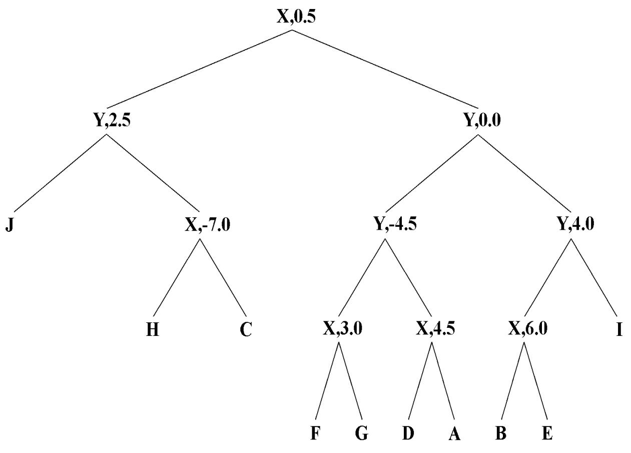


Figure 2- The KD-Tree created using the sample data.

The corresponding partitioning of the plane is given in Fig. 3. We note that this example comes from a larger data set and thus does not appear properly balanced. This data set will be used as an example in the discussion of other methods.

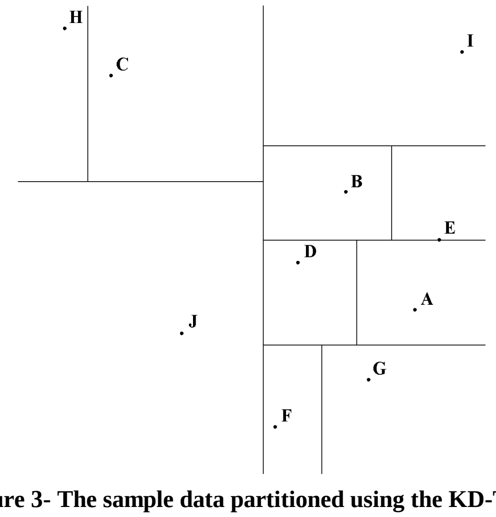


Figure 3- The sample data partitioned using the KD-Tree.

A nearest neighbor search may then be performed as a top-down recursive traversal of a portion of the tree. At each node, the query point is compared with the cut value along the specified cut dimension. If along the cut dimension the query point is less than the cut value, then the left branch is descended. Otherwise, the right branch is descended. When a bucket is reached, all points in the bucket are compared to see if any of them is closer than the distance to the nearest neighbor found so far. After the descent is completed, at any node encountered, if the distance to the closest neighbor found is greater than the distance to the cut value, then the other branch at that node needs to be descended as well. Searching stops when no more branches need to be descended.

Bentley recommends the use of parent nodes for each node in a tree structure. A search may then be performed using a bottom-up approach, starting with the bucket containing the search point and searching through a small number of buckets until the appropriate neighbors have been found. For nearest neighbor searches this reduces computational time from O(log M) to O(1). This however, does not immediately improve on search time for finding near neighbors of points *not* in the database. Timed trials indicated that the increased speed due to bottom-up (as opposed to top-down) searches was negligible. This is because most of the computational time is spent in distance calculations, and the reduced number of comparisons is negligible.

3. THE K-TREE

3.1 Description

K-trees are a generalization of the single-dimensional M-ary search tree. As a data comparative search tree, a K-tree stores data objects in both internal and leaf nodes. A hierarchical recursive subdivision of the N-dimensional search space is induced with the space partitions following the locality of the data. Each node in a K-tree contains $K=2^N$ child pointers. The root node of the tree represents the entire search space and each child of the root represents a K-*ant* of the parent space.

One of the disadvantages of the K-tree is its storage space requirements. In a standard implementation, as described here, a tree of M N-dimensional vectors requires a minimum of $(2^N+N)\cdot M$ fields. Only M-1 of the 2^N branches actually point to a node. The rest point to NULL data. For large N, this waste becomes prohibitive.

3.2 Method

Consider the case of two-dimensional data (N=2, K=4). This K-tree is known as a quad-tree, and is a 4-ary tree with each node possessing 4 child pointers. The search space is a plane and the partitioning induced by the structure is a hierarchical subdivision of the plane into disjoint quadrants. If the data consists of the 10 vectors described in Section 2.2, then the corresponding tree is depicted in Fig. 4 and the partitioning of the plane in Fig. 5.

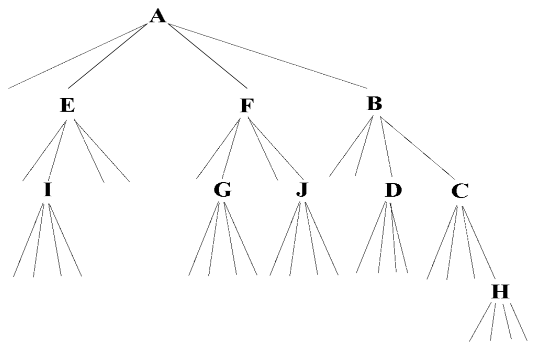


Figure 4- The KDTree created using the sample data.

Note that much of the tree is consumed by null pointers.

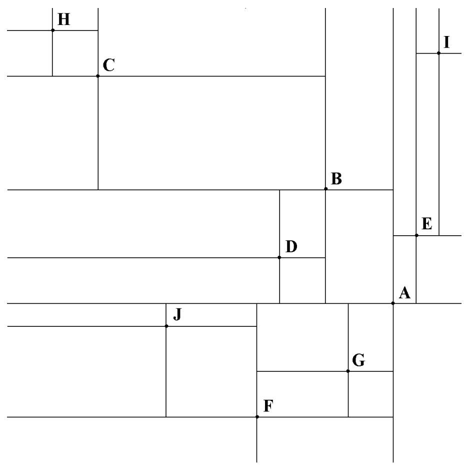


Figure 5- Sample data partitioned using the KTree method.

Searching the tree is a recursive two-step process. A cube that corresponds to the bounding extent of the search sphere is intersected with the tree at each node encountered. The bounds of this cube are maintained in an N-dimensional range array. This array is initialized based on the search vector. At each node, the direction of search is determined based on this intersection. A search on a child is discontinued if the region represented by the child does not intersect the search cube. This same general method may be applied to weighted, radial, and nearest neighbor searches. For radial searches, the radius of the search sphere is fixed. For nearest neighbor searches it is doubled if the nearest neighbor has not been found, and for weighted searches it is doubled if enough neighbors have not been found.

4. MULTIDIMENSIONAL QUICKSORT

4.1 Description

For many analyses, one wishes to search only select dimensions of the data. A problem frequently encountered is that a different sort would need to be performed for each search based on a different dimension or subset of all the dimensions. We propose here a multidimensional generalization of the quick-sort routine.

4.2 Method

A quick-sort is performed on each of the N dimensions. The original array is not modified. Instead, two new arrays are created for each quick-sort. The first is the quick-sort array, an integer array where the value at position *k* in this array is the position in the data array of the *k*th smallest value in this dimension. The second array is the inverted quick-sort. It is an integer array where the value at position *k* in the array is the position in the quick-sort array of the value *k.* Keeping both arrays allows one to identify both the location of a sorted value in the original array, and the location of a value in the sorted array. Thus, if $\vec{x}(1)$ has the second smallest value in the third dimension, then it may be represented as $\vec{x}^3(2)$. The value stored at the second index in the quick-sort array for the third dimension will be 1, and the value stored at the first index in the inverted quick-sort array for the third dimension will be 2. Note that the additional memory overhead need not be large. For each floating-point value in the original data, two additional integer values are stored-, one from the quick-sort array and one from the inverted quick-sort array.

We begin by looking at a simple case and showing how the method can easily be generalized. We consider the case of two-dimensional data, with coordinates x and y, where we make no assumptions about delay coordinate embeddings or uniformity of data.

Suppose we wish to find the nearest neighbor of the 2-dimensional vector $\vec{x} = (x_1, x_2)$. If this vector's position on the first axis quick-sort is *i* and its position on the second axis quick-sort is *j* (*i* and *j* are found using the inverted quick-sorts), then it may also be represented as

$$\vec{x} = \vec{x}^1(i) = (x_1^1(i), x_2^1(i)) = \vec{x}^2(j) = (x_1^2(j), x_2^2(j)).$$

Using the quick-sorts, we search outward from the search vector, eliminating search directions as we go. Reasonable candidates for nearest neighbor are the nearest neighbors on either side on the first axis quick-sort, and the nearest neighbors on either side on the second axis quick-sort. The vector $\vec{x}^1(i-1) = (x_1^1(i-1), x_2^1(i-1))$ corresponding to position *i*-1 on the first axis quick-sort is the vector with the closest coordinate on the first dimension such that $x_1^1(i-1) < x_1$. Similarly, the vector $\vec{x}^1(i+1)$ corresponding to *i*+1 on the first axis quick-sort is the vector with the closest coordinate on the first dimension such that $x_1^1(i+1) < x_1$. And from the y-axis quick-sort, we have the vectors $\vec{x}^2(j-1)$ and $\vec{x}^2(j+1)$. These are the four vectors adjacent to the search vector in the two quick-sorts. Each vector's distance to the search vector is calculated and we store the minimal distance and the corresponding minimal vector. If $|x^1(i-1) - x|$ is greater than the minimal distance, then we know that all vectors $\vec{x}^1(i-1), \vec{x}^1(i-2), \ldots \vec{x}^1(1)$ must also be further away than the minimal vector. In that case, we will no longer search in decreasing values on the first axis quick-sort. We would also no longer search in decreasing values on the first axis quick-sort if $\vec{x}^1(1)$ has been reached. Likewise, if $|x^1(i+1) - x|$ is greater than the minimal distance, then we know that all vectors $\vec{x}^1(i+1), \vec{x}^1(i+2), \ldots \vec{x}^1(M)$ must also be further away than the minimal vector. If either that is the case or $\vec{x}^1(M)$ has been reached then we would no longer search in increasing values on the x-axis quick-sort. The same rule applies to $|x^2(j-1) - x|$ and $|x^2(j+1) - x|$.

We then look at the four vectors, $\vec{x}^1(i-2)$, $\vec{x}^1(i+2)$, $\vec{x}^2(j-2)$ and $\vec{x}^2(j+2)$. If any of these is closer than the minimal vector, then we replace the minimal vector with this one, and the minimal distance with this distance. If $|x^1(i-2) - x|$ is greater than the minimal distance, then we no longer need to continue searching in this direction. A similar comparison is made for $|x^1(i+2) - x|, |x^2(j-2) - x|$ and $|x^2(j+2) - x|$.

This procedure is repeated for $\vec{x}^1(i-3)$, $\vec{x}^1(i+3)$, $\vec{x}^2(j-3)$ and $\vec{x}^2(j+3)$, and so on, until all search directions have been eliminated. We find the distance of the four points from our point of interest and, if possible, replace the minimal distance. We then proceed to the next four points and proceed this way until all directions of search have been eliminated.

The minimal vector must be the nearest neighbor, since all other neighbor distances have either been calculated and found to be greater than the minimal distance, or have been shown that they must be greater than the minimal distance.

Extension of this algorithm to higher dimensions is straightforward. In N dimensions there are 2N possible directions. Thus 2N immediate neighbors are checked. A minimal distance is found, and then the next 2N neighbors are checked. This is continued until it can be shown that none of the 2N directions can contain a nearer neighbor.

It is easy to construct data sets for which this is a very inefficient search. For instance, if one is looking for the closest point to (0,0) and one were to find a large quantity of points residing outside the circle of radius 1 but inside the square of side length 1 then all these points would need to be measured before the closer point at (1,0) is considered. However, similar situations can be constructed for most multidimensional sort and search methods, and preventative measures can be taken.

5. THE BOX-ASSISTED METHOD

The box-assisted search method was described by Schreiber, et al.[11] as a simple multidimensional search method for nonlinear time series analysis. A grid is created and all the vectors are sorted into boxes in the grid. Fig. 6 demonstrates a two-dimensional grid that would be created for the sample data. Searching then involves finding the box that a point is in, then searching that box and all adjacent boxes. If the nearest neighbor has not been found, then the search is expanded to the next adjacent boxes. The search is continued until all required neighbors have been found.

One of the difficulties with this method is the determination of the appropriate box size. The sort is frequently tailored to the type of search that is required, since a box size is required and the preferred box size is dependent on the type of search to be done. However, one usually has only limited a priori knowledge of the searches that may be performed. Thus the appropriate box size for one search may not be appropriate for another. If the box size is too small, then many boxes are left unfilled and many boxes will need to be searched. This results in both excessive use of memory and excessive computation.

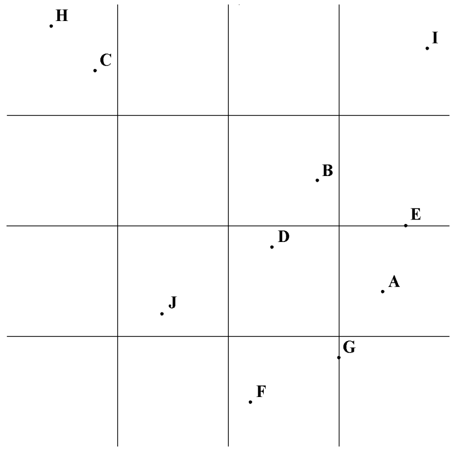


Figure 6- The sample data set as gridded into 16 boxes in two dimensions, using the box-assisted method.

The choice of box dimensionality may also be problematic. Schreiber, et al.[11] suggest 2 dimensional boxes. However, this may lead to inefficient searches for high dimensional data. Higher dimensional data may still be searched although many more boxes are often needed in order to find a nearest neighbor. On the other hand, using higher dimensional boxes will exacerbate the memory inefficiency. In the benchmarking section, we will consider both two and three-dimensional boxes.

6. BENCHMARKING AND COMPARISON OF METHODS

In this section we compare the suggested sorting and searching methods, namely the box assisted method, the KD-Tree, the K-tree, and the multidimensional quick-sort. All of these methods are preferable to a brute force search (where no sorting is done, and all data vectors are examined each time we do the searching). However, computational speed is not the only relevant factor. Complexity, memory use, and versatility of each method will also be discussed. The versatility of the method comes in two flavors- how well the method works on unusual data and how adaptable the method is to unusual searches. The multidimensional binary representation and the uniform K-Tree, described in the previous two sections, are not compared with the others because they are specialized sorts used only for exceptional circumstances.

6.1 Benchmarking of the KDTree

One benefit of the KD-Tree is its rough independence of search time on data set size. Figure 7 compares the average search time to find a nearest neighbor with the data set size. For large data set size, the search time has a roughly logarithmic dependence on the number of data points. This is due to the time it takes to determine the search point's location in the tree. If the search point were already in the tree, then the nearest neighbor search time is reduced from O(log *n*) to O(1). This can be accomplished with the implementation of Bentley's suggested use of parent pointers for each node in the tree structure.[10] This is true even for higher dimensional data, although the convergence is much slower.

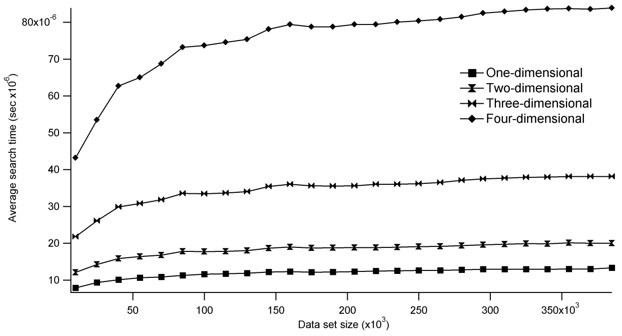


Figure 7- The dependence of average search time on data set size.

In Figure 8, the KD-Tree is shown to have an exponential dependence on the dimensionality of the data. This is an important result, not mentioned in other work providing diagnostic tests of the KD-Tree.[10, 12] It implies that KD-Trees become inefficient for high dimensional data. It is not yet known what search method is most preferable for neighbor searching in a high dimension (greater than 8), although Liu, et. al. have proposed a method similar to the multidimensional quicksort for use in multimedia data retrieval.[13]

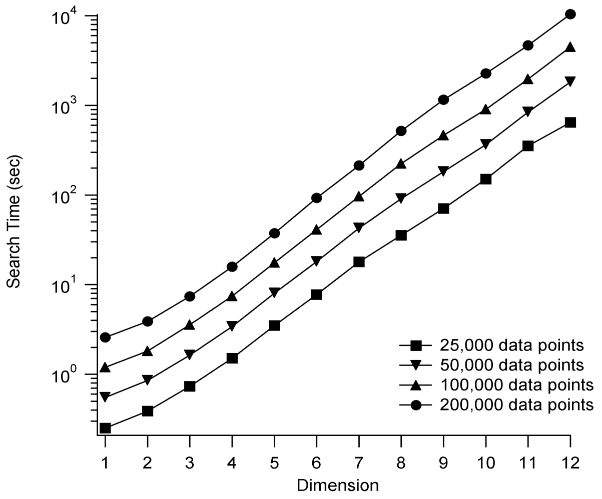


Figure 8- A log plot of search time vs dimension.

Figure 9 shows the relationship between the average search time to find *n* neighbors of a data point and the value *n*. In this plot, 10 data sets were generated with different seed values and search times were computed for each data set. The figure shows that the average search time is almost nearly linearly dependent on the number of neighbors *n*. Thus a variety of searches (weighted, radial, with or without exclusion) may be performed with only a linear loss in speed.

The drawbacks of the KD-Tree, while few, are transparent. First, if searching is to be done in many different dimensions, either a highly inefficient search is used or additional search trees must be built. Also the method is somewhat memory intensive. In even the simplest KD-Tree, a number indicating the cutting value is required at each node, as well as an ordered array of data (similar to the quick-sort). If pointers to the parent node or principal cuts are used then the tree must contain even more information at each node. Although this increase may at first seem unimportant, one should note that a music information retrieval system may consist of a vast number of files, or alternately, a vast number of samples within each file. Thus memory may prove unmanageable for many workstation computers.

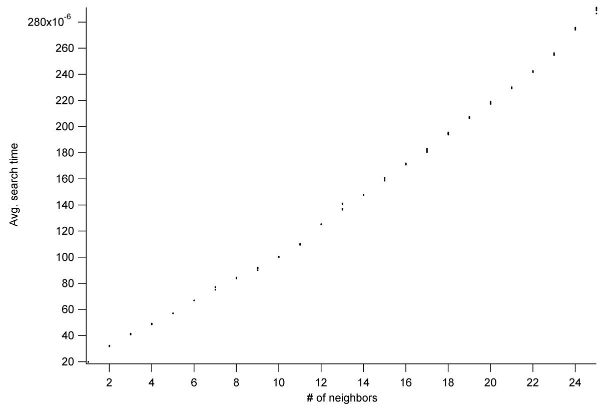


Figure 9- A plot of the average search time to find *n* neighbors of a data point, as a function of *n*.

We have implemented the KD-Tree as a multiplatform dynamic linked library consisting of a set of fully functional object oriented routines. The advantage of such an implementation is that the existing code can be easily ported into existing MIR systems. In short, the core code consists of the following functions

```
Create(*phTree, nCoords, nDims, nBucketSize,*aPoints);
FindNearestNeighbor(Tree,*pSearchPoint, *pFoundPoint);
FindMultipleNeighbors(Tree, *pSearchPoint,
*pnNeighbors, *aPoints);
FindRadialNeighbors(Tree, *pSearchPoint, radius,
**paPoints, *pnNeighbors);
ReleaseRadialNeighborList(*aPoints);
Release(Tree);
```

6.2 Comparison of methods

The KD-Tree implementation was tested in timed trials against the multidimensional quick-sort and the box-assisted method. In Figure 10 through Figure 13, we depict the dependence of search time on data set size for one through four dimensional data, respectively.

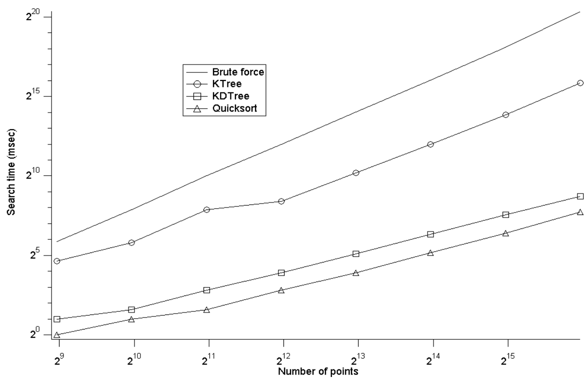


Figure 10- Comparison of search times for different methods using 1 dimensional random data.

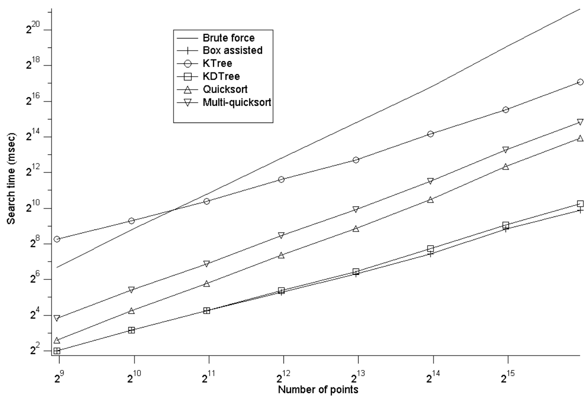


Figure 11- Comparison of search times for different methods using 2 dimensional random data.

In Figure 10, the multidimensional quick-sort reduces to a one-dimensional sort and the box assisted method as described by [11] is not feasible since it requires that the data be at least two-dimensional. We note from the slopes of these plots that the box-assisted method, the KDTree and the KTree all have an O(n log n) dependence on data set size, whereas the quick-sort based methods have approximately O($n^{1.5}$) dependence on data set size for 2 dimensional data and O($n^{1.8}$) dependence on data set size for 3 or 4 dimensional data. As expected, the brute force method has O(n^2) dependence.

Despite its theoretical O(n log n) performance, the KTree still performs far worse than the box-assisted and KDTree methods. This is because of a large constant factor worse performance that is still significant for large data sets (64,000 points). This constant worse performance relates to the poor balancing of the KTree. Whereas for the KDTree, the data may be permuted so that cut values are always chosen at medians in the data, the KTree does not offer this option because there is no clear multidimensional median. In addition, many more branches in the tree may need to be searched in the KTree because at each cut, there are 2^k instead of 2 branches.

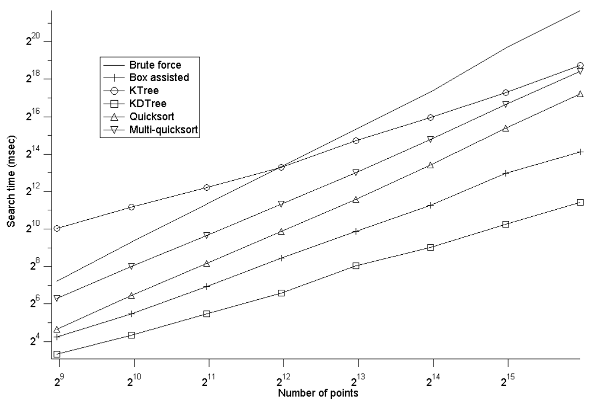


Figure 12- Comparison of search times for different methods using 3 dimensional random data.

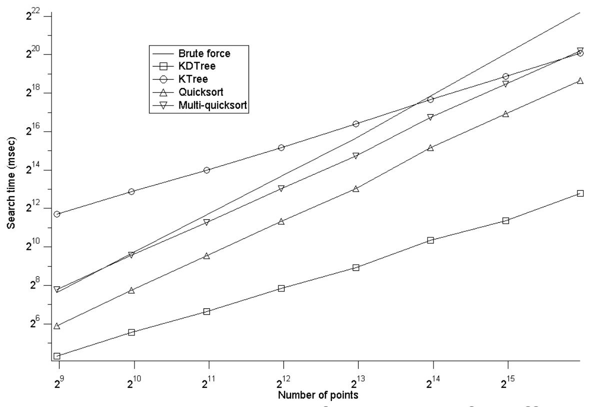


Figure13- Comparison of search times for different methods using 4 dimensional random data.

However, all of the above trials were performed using uniform random noise. They say nothing of how these methods perform with other types of data. In order to compare the sorting and searching methods performance on other types of data, we compared their times for nearest neighbor searches on a variety of data sets. Table 1 depicts the estimated time in milliseconds to find all nearest neighbors in different 10,000 point data sets for each of the benchmarked search methods. The uniform noise data was similar to that discussed in the previous section.

Each Gaussian noise data set had a mean of 0 and standard deviation of 1 in each dimension. The identical dimensions and one valid dimension data sets were designed to test performance under unusual circumstances.

For the identical dimensions data, uniform random data was used and each coordinate of a vector was set equal, e.g.,

$$\vec{x}(i) = (x_1(i), x_2(i), x_3(i)) = (x_1(i), x_1(i), x_1(i))$$

For the data with only one valid dimension, uniform random data was used in only the first dimension, e.g.,

$$\vec{x}(i) = (x_1(i), x_2(i), x_3(i)) = (x_1(i), 0, 0)$$

In all cases the KD-Tree proved an effective method of sorting and searching the data. Only for the last two data sets did the multidimensional quick-sort method prove faster, and these data sets were constructed so that they were, in effect, one-dimensional. In addition, the box method proved particularly ineffective for high dimensional Gaussian data where the dimensionality guaranteed that an excessive number of boxes needed to be searched, and for the Lorenz data, where the highly non-uniform distribution ensured that many boxes went unfilled. The K-tree also performed poorly for high dimensional data (four and five dimensional), due to the exponential increase in the number of searched boxes with respect to dimension.

A summary of the comparison of the four routines can be found in Table 2. The "adaptive" and "flexible" criteria refer to the next section.

Table 2- Comparison of some features of the four routines. Rating from 1=best to 4=worst.

Algorithm	Memory	Build	Search	Adapt.	Flexible
KDTree	2	3/4	1	yes	yes
KTree	3	3/4	3	no	no
Quick-sort	1	2	4	yes	yes
BoxAssisted	4	1	2	no	yes

7. INTERESTING PROPERTIES FOR MUSIC IR

The multi-dimensional search approach to Music IR, and the corresponding algorithms presented above have a number of interesting properties and conceptual advantages.

7.1 Adaptive to the distribution

A truly multi-dimensional approach enables an adaptation to the distribution of the data set. For example, the KD-Tree algorithm focuses its discriminating power in a non-uniform way. The search tree it creates represents a best fit to the density of the data. This could be efficient for, say, search tasks in a database where part of the features remain quasi constant, e.g. a database of samples which are all pure tones of a given instrument, with quasi constant pitch, and a varying brightness. It is interesting to compare this adaptive behavior with a string-matching algorithm that would have to compare sequences that all begin with "aaa…". The latter can't adapt and systematically tests the first three digits, which is an obvious waste of time.

7.2 Independent of the metric and of the alphabet

All the methods presented here are blind to the metric that is used. This is especially useful if the set of features is composite, and requires a different metric for each coordinate, e.g. pitches can be measured modulo 12. The routines are also independent of the alphabet, and work for integers as well as for

Table 1- Nearest neighbor search times for data sets consisting of 10000 points. The brute force method, multidim. quick-sort, the box assisted method in 2 and 3 dimensions, the KDTree and the KTree were compared. An X indicates that it wasn't possible to use this search method on this type of data. The fastest method is given in bold and the second fastest method is given in italics.

Data set	Dimension	Brute	Quicksort	Box (2) method	Box (3) method	KDTree	KTree
Uniform noise	**3**	32567	2128	344	*210*	**129**	845
Gaussian	**2**	16795	*280*	623	X	**56**	581
Gaussian	**4**	44388	8114	54626	195401	**408**	*3047*
Identical dimensions	**3**	33010	**19**	1080	5405	*42*	405
One valid dimension	**3**	30261	**31**	1201	7033	*37*	453

floating-points. This makes them very general, as they can deal with a variety of queries on mixed low-level features and high-level meta-data such as:

Nearest neighbor $(pitch1, pitch2, pitch3, "BACH")$

7.3 Flexibility

There are a variety of searches that are often performed on multidimensional data.[14] Perhaps the most common type of search, and one of the simplest, is the nearest neighbor search. This search involves the identification of the nearest vector in the data set to some specified vector, known as the search vector. The search vector may or may not also be in the data set. Expansions on this type of search include the radial search, where one wishes to find all vectors within a given distance of the search vector, and the weighted search, where one wishes to find the nearest *A* vectors to the search vector, for some positive integer *A*.

Each of these searches (weighted, radial and nearest neighbor) may come with further restrictions. For instance, points or collections of points may be excluded from the search. Additional functionality may also be required. The returned data may be ordered from closest to furthest from the search vector, and the sorting and searching may be required to handle the insertion and deletion of points. That is, if points are deleted from or added to the data, these additional points should be added or deleted to the sort so that they can be removed or included in the search. Such a feature is essential if searching is to performed with real-time analysis.

Most sorting and searching routine presented above are able to perform all the common types of searches, and are adaptable enough so that they may be made to perform any search.

7.4 A note on dimensionality

One of the restrictions shared by the multidimensional search routines presented on this paper is their dependence on the dimensionality of the data-set (not its size). This is detrimental to the sheer "folding" of the trajectory search as presented in the introduction, especially when it involves long *m*-sequences of high-*n*-dimension features (dimension $N = m \cdot n$ may be too high). However, as we mentioned in the course of this paper, there are still a variety of searches that can fit into the multidimensional framework. We notably wish to suggest:

- Searches for combinations of high-level metadata (m=1)
- It is possible to reduce N with classic dimensionality reduction techniques, such as Principal Component Analysis or Vector Quantization.
- It is possible to reduce M by computing only 1 vector of features per audio piece. It is the approach taken in the Muscle Fish™ technology [15], where the mean, variance and correlation of the features are included in the feature vector.
- It is possible to reduce M by computing the features not on a frame-to-frame basis, but only when a significant change occurs ("event-based feature extraction", see for example [16]).
- For finite alphabets, it is always possible to reduce the dimension of a search by increasing the size of the alphabet. For example, searching for a set of 9 notes out of a 12 semi-tone alphabet can be reduced to a 3D search over an alphabet of 12^3 symbols.

8. CONCLUSION

We've presented and discussed four algorithms for a multidimensional approach to Music IR. The KD search tree is a highly adaptable, well-researched method for searching multidimensional data. As such it is very fast, but also can be memory intensive, and requires care in building the binary search tree. The k tree is a similar method, less versatile, more memory intensive, but easier to implement. The box-assisted method on the other hand, is used in a form designed for nonlinear time series analysis. It falls into many of the same traps that the other methods do. Finally the multidimensional quick-sort is an original method designed so that only one search tree is used regardless of how many dimensions are used.

These routines share a number of conceptual advantages over the approaches taken so far in the Music IR community, which -we believe- can be useful for a variety of musical searches. The aim of the paper is to be only a review, and the starting point of a reflection about search algorithms for music.

In particular, we still have to implement specific music retrieval systems that use the results presented here.

9. REFERENCES

[1] K. Lemstrom, String Matching Techniques for Music Retrieval. Report A-2000-4, University of Helsinki Press.

[2] H. Samet, *Applications of Spatial Data Structures*: Addison-Wesley, 1989.

[3] H. Samet, *The design and analysis of spatial data structures*: Addison-Wesley, 1989.

[4] H. Hinterberger, K. C. Sevcik, and J. Nievergelt, *ACM Trans. On Database Systems*, vol. 9, pp. 38, 1984.

[5] N. Pippenger, R. Fagin, J. Nievergelt, and H. R. Strong, *ACM Trans. On Database Systems*, vol. 4, pp. 315, 1979.

[6] K. Mehlhorn, *Data Structures and Algorithms 3: Multidimensional Searching and Computational Geometry*: Springer-Verlag, 1984.

[7] F. P. Preparata and M. I. Shamos, *Computational geometry: An introduction*. New York: Springer-Verlag, 1985.

[8] J. H. Bentley, "Multidimensional Binary Search Trees Used for Associative Searching," *Communications of the ACM*, vol. 18, pp. 509-517, 1975.

[9] J. H. Friedman, J. L. Bentley, and R. A. Finkel, "An algorithm for finding best matches in logarithmic expected time," *ACM Trans. Math. Software*, vol. 3, pp. 209, 1977.

[10] J. L. Bentley, "K-d trees for semidynamic point sets," in *Sixth Annual ACM Symposium on Computational Geometry*, vol. 91. San Francisco, 1990.

[11] T. Schreiber, "Efficient neighbor searching in nonlinear time series analysis," *Int. J. of Bifurcation and Chaos*, vol. 5, pp. 349-358, 1995.

[12] R. F. Sproull, "Refinement to nearest-neighbour searching in k-d trees," *Algorithmica*, vol. 6, p. 579-589, 1991.

[13] C.-C. Liu, J.-L. Hsu, A. L. P. Chen, "Efficient Near Neighbor Searching Using Multi-Indexes for Content-Based Multimedia Data Retrieval," *Multimedia Tools and Applications*, Vol 13, No. 3, 2001, p.235-254.

[14] J. Orenstein, *Information Processing Letters*, vol. 14, pp. 150, 1982.

[15] E. Wold, T. Blum et al., "Content Based Classification, Search and Retrieval of Audio", in IEEE Multimedia, Vol.3, No. 3, Fall 1996, p.27-36.

[16] F. Kurth, M. Clausen, "Full Text Indexing of Very Large Audio Databases", in Proc. 110th AES Convention, Amsterdam, May 2001.

Expressive and efficient retrieval of symbolic musical data

Michael Droettboom, Ichiro Fujinaga, Karl MacMillan
The Peabody Institute
The Johns Hopkins University
{mdboom,ich,karlmac}@peabody.jhu.edu

Mark Patton, James Warner, G. Sayeed Choudhury, Tim DiLauro
Digital Knowledge Center
Milton S. Eisenhower Library
The Johns Hopkins University
{sayeed,timmo,mpatton,jwarner}@jhu.edu

ABSTRACT

The ideal content-based musical search engine for large corpora must be both expressive enough to meet the needs of a diverse user base and efficient enough to perform queries in a reasonable amount of time. In this paper, we present such a system, based on an existing advanced natural language search engine. In our design, musically meaningful searching is simply a special case of more general search techniques. This approach has allowed us to create an extremely powerful and fast search engine with minimal effort.

1. INTRODUCTION

This paper describes a system for music searching that is expressive enough to perform both simple and sophisticated searches that meet a broad range of user needs. It is also efficient enough to search through a large corpus in a reasonable amount of time. The music search system was created by extending an existing advanced natural language search engine with simple filters and user-interface elements.

This paper will describe the search engine in the context of our larger sheet music digitization project, and relate it to other musical search engines already available for use on the web. Then, the capabilities of the non-music-specific core of the search engine will be described, followed by the extensions necessary to adapt it to music.

2. BACKGROUND

The Lester S. Levy Collection of Sheet Music represents one of the largest collections of sheet music available online. The Collection, part of the Special Collections of the Milton S. Eisenhower Library at The Johns Hopkins University, comprises nearly 30,000 pieces of music (Choudhury et al. 2000). It provides a rich, multi-facetted view of life in late 19th and early 20th century America. Scholars from various disciplines have used the Collection for both research and teaching. All works in the public domain are currently available online as JPEG images. The user can browse the collection by category or search based on metadata, such as author, title, publisher, and date. Musical searches, such as finding a particular melodic or rhythmic pattern, will soon be possible once the collection has been converted to symbolic musical data.

To convert this data, an optical music recognition (OMR) system is being developed (Choudhury et al. 2001). We chose GUIDO as the target representation language due to its simplicity and extensibility (Hoos and Hamel 1997). Having music in a symbolic format opens the collection to sound generation, musicological analysis and, the topic of the present paper, musical searching.

3. PRIOR ART

None of the available musical search engines we evaluated met the needs of the diverse user base of the collection, or could handle the large quantity of data in the complete Levy collection. In particular, we evaluated two projects in detail: Themefinder (Huron et al. 2001) and MELDEX (McNab et al. 1997).

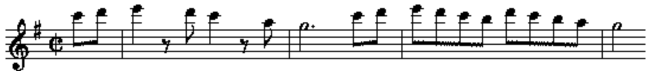
3.1 Themefinder

Themefinder's goal is to retrieve works by their important themes. These themes are manually determined ahead of time and placed in an incipit database.

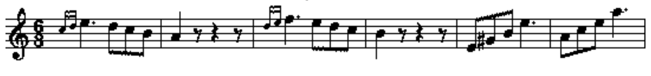
One can query the database using five different kinds of search queries: pitch, interval, scale degree, gross contour, and refined contour. These five categories served as the inspiration for a subset of our basic query types. The user can query within an arbitrary subset of these categories and then intersect the results. However, Themefinder does not allow the user to combine these query types within a single query in arbitrary ways. For instance, a user may know the beginning of a melodic phrase, while the ending is more uncertain. Therefore, the user may want to specify exact intervals at the beginning and use gross contours or wild-cards at the end. Unfortunately, in Themefinder, the user must have the same level of certainty about all of the notes in the query. Unfortunately, this is not consistent with how one remembers melodies (McNab et al. 2000).

In addition, Themefinder does not have a notion of rhythmic searching. While its invariance to rhythm can be an asset, it can also be cumbersome when it provides too many irrelevant matches. Figure 1 shows the results of a query where one result is more relevant than the other. Such queries may return fewer false matches if they could include rhythmic information.

The searches themselves are executed in Themefinder using a brute-force method. The entire database is linearly searched for the given search query string. While this is acceptable for the 18,000 incipits in Themefinder's largest database, it may not scale well for searching across a full-text database such as the Levy collection.



Beethoven, Ludwig Van. Quartet in E Minor, Op. 59, No. 2 "Rasoumowsky", 4th Movement.



Beethoven, Ludwig Van. Sonata No. 4, in A Minor, Op. 23, Violin and Pianoforte, 1st Movement.

Figure 1: **These two incipits start with the identical set of pitches,** $[c\ d\ e\ d]$**, but with different rhythmic content. With better rhythmic specificity, irrelevant results could be eliminated.** (`http://www.themefinder.org/`)

3.2 MELDEX

The simple text-based query strings in Themefinder are easy to learn and use by those with moderate musical training. MELDEX, however, has a more natural interface for non-musicians. The user sings a melody using a syllable with a strong attack such as "tah." The pitches of the melody are determined using pitch-tracking, and the rhythm is quantized. The results are used as the search query. The query is approximately matched to melodies in the database using a fast-matching algorithm related to dynamic programming. While this approach is highly effective for non-musicians and simple queries, it is limiting to those wanting more fine-grained control.

4. CAPABILITIES

Our musical search engine supports both melodic and rhythmic searches. Search queries can also include the notion of simultanaeity. That is, events can be constrained to occur at the same time as other events. The search engine, as described here, is limited to standard-practice Western music, though modifications could be made to support other musical traditions.

4.1 Extensibility

Other types of musical searching beyond these core capabilities require additional layers of musical knowledge to be built on top of the search engine. The general design of the search engine encourages such extensibility. Any analytical data that can be derived from the score data can be generated offline (ahead of time) and later used as search criteria. This data can be generated by new custom tools or existing analysis tools such as the Humdrum toolkit (Huron 1999).

For example, the search engine could be extended to support harmonic searches with respect to harmonic function. Western tonal harmonic theory is ambiguous, making it difficult to objectively interpret and label harmonies. This is a largely unsolved problem that is not the subject of our present research. However, assuming an acceptable solution to these issues could be found, labeling of harmonic function could be implemented as an input filter.

Also, the core search engine does not include any notion of melodic similarity. This is an open problem strongly tied to subjective matters of human perception (Hewlett and Selfridge-Field 1998). It is possible for a specialized front-end to include notions of melodic similarity by generating specialized search queries. The search query language of the core search engine is expressive enough that these advanced features could be added without modifying the core itself.

4.2 Meeting diverse user requirements

We define the users of our musical search engine as anyone who wants to access the collection in a musical way. Of course, the needs of different users are greatly varied. A non-musician may want to hum into a microphone to retrieve a particular melody. A copyright lawyer may want to track the origins of a particular melody, even melodies that are merely similar. A musicologist may want to determine the frequency of particular melodic or rhythmic events. To meet these diverse needs, it is necessary to provide different interfaces for different users. The set of interfaces is arbitrary and can be extended as new types of users are identified. It may include graphical applications, web-based forms and applets, or text-based query languages. Audio interfaces, with pitch- and rhythm-tracking may also be included. The purpose of these interfaces is to translate a set of user-friendly commands or interactions into a query string accepted by the search engine. The details of that query can be hidden from the end-user and therefore can be arbitrarily complex.

At present, we have focused our attention on the core search engine itself. In the second phase of the search engine project, the user interfaces will be developed in collaboration with a usability specialist.

5. THE CORE SEARCH ENGINE

The core search engine in our system was originally developed for text-based retrieval of scores based on their meta-data and full-text lyrics. Its overall design was inspired by recent developments in the field of natural-language searching (DiLauro et al. 2001). These features allow the user to perform search queries using the embedded context in natural languages, such as parts of speech, rhyming scheme, and scansion. While not originally intended for musical searching, it was soon discovered that the core was very well suited for searching across symbolic musical data.

The core itself did not need to be modified to support music searching. Instead, specialized filters and front-ends were added to adapt it to the music domain. In the ingestion stage, the data is filtered to store it in the appropriate indices and partitions (see Section 6). When searching, special user interfaces handle the details of generating search query strings and filtering and displaying the resulting data. Figure 2 shows how the individual parts of the system fit together to ingest and query the data.

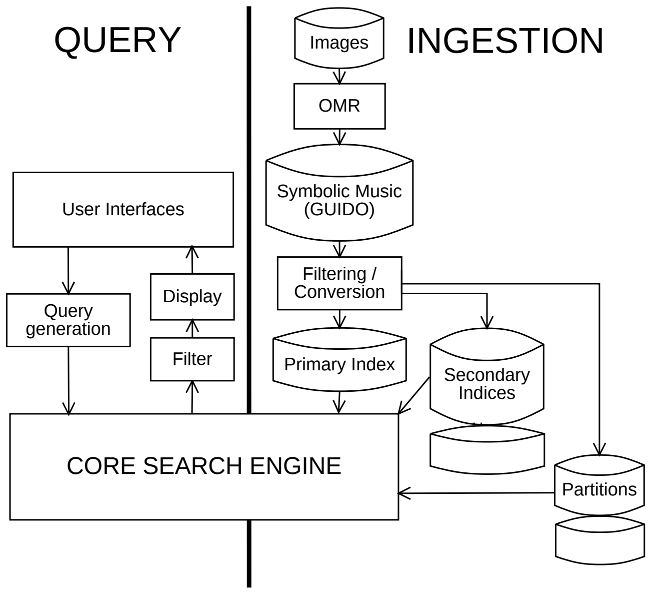


Figure 2: **Workflow diagram of the musical search engine.**

5.1 Inverted lists

Many search engines, including ours, are built on the concept of an inverted list. For a complete discussion of inverted list search engines, see Witten et al. (1999).

Sequential data, such as English prose or melodic data, is stored on disk as a sequence of atoms. In the case of English, the atom is the word and the sequence is simply the ordered words as they appear in sentences and paragraphs. Take for example the following sentence:

```
To be , or not to be , that is
the question .
```

Note that both words and punctuation are treated as indivisible atoms. To search for a particular atom in this string, a computer program would need to examine all thirteen atoms and compare it with a query atom. To increase searching efficiency, an inverted list search engine would store this string internally as:

`,` $\longrightarrow \{3, 8\}$
`.` $\longrightarrow \{13\}$
`be` $\longrightarrow \{2, 7\}$
`is` $\longrightarrow \{10\}$
`not` $\longrightarrow \{5\}$
`or` $\longrightarrow \{4\}$
`question` $\longrightarrow \{12\}$
`that` $\longrightarrow \{9\}$
`the` $\longrightarrow \{11\}$
`to` $\longrightarrow \{1, 6\}$

Here, each atom in the string is stored with a list of numbers indicating the atom's ordinal location within the string. The set of words in the index is called the vocabulary of the index. To search for a particular atom using the index, the program needs only to find that word in the vocabulary and it can easily obtain a list of indices (or pointers) to where that atom is located within the string. Since the vocabulary can be sorted, the lookup can be made faster using hashing or a binary search.

Inverted lists perform extremely well when the size of the vocabulary is small relative to the size of the corpus. In the case of English, of course, the vocabulary is much smaller relative to the size of all the works written in that language. This property also allows us to improve the efficiency of musical searching as we will see below.

6. THE MUSICAL SEARCH ENGINE

The musical search capabilities are supported by three main features of the core search engine:

1. **Secondary indices** allow the amount of specificity to vary with each token.
2. **Partitions** allow search queries to be performed upon specific discontiguous parts of the corpus.
3. **Regular expressions** allow advanced pattern matching.

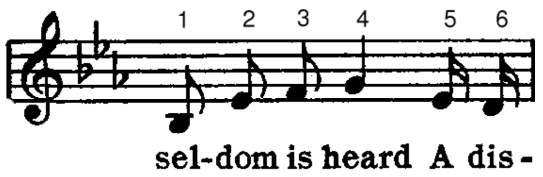


Figure 3: **A measure of music, from the Levy collection, used as an example throughout this section. (Guion, D. W., arr. 1930. "Home on the range." New York: G. Schirmer.)**

6.1 Secondary indices

In the case of music, the searchable atom is not the word, but the musical event. Events include anything that occurs in the score, such as notes, rests, clefs, and barlines. Each of these events, of course, can have many properties associated with it. For instance, the note $b^\flat$ at the beginning of the fragment in Figure 3 has the following properties:

- **Pitch name:** *b*
- **Accidental:** $\flat$
- **Octave:** -1 (first octave below middle-*c*)
- **Twelve-tone pitch:** 10 (10th semitone above *c*)
- **Base-40 pitch:** -3 (see Hewlett 1992)
- **Duration:** eighth note
- **Interval to next note:** perfect 4th
- **Contour (direction) to next note:** up
- **Scale degree:** *so* (5th scale degree in $E^\flat$ major)
- **Lyric syllable:** "sel–"
- **Metric position:** Beat 1 in a $\frac{6}{8}$ measure

All of these properties are self-evident, with the exception of base-40 pitch, which is a numeric pitch representation where the intervals are invariant under transposition while maintaining enharmonic spelling (Hewlett 1992). Note also, we use GUIDO-style octave numbers, where the octave containing middle-*c* is zero, as opposed to ISO-standard octave numbers.

The concept of secondary indices allows the individual properties of each atom to be indexed independently of any other properties. This allows search queries to have arbitrary levels of specificity in each event. The set of properties can be extended to include any kinds of data that can be extracted from the musical source. For example, if the harmonic function of chords could be determined unambiguously, a secondary index containing chord names in Roman numeral notation could be added. In our design, we use secondary indices to handle the properties of events that change from note to note. Continuous properties of events, that are carried from one event to the next, such as clefs, time signatures, and key signatures, are handled using partitions, explained below (see Section 6.2).

6.1.1 Ingestion of secondary indices

During the ingestion phase, the source GUIDO data is first converted to an interim format where all of each event's properties are fully specified. For example, the GUIDO representation of Figure 3 is as follows:

```
[ \clef<"treble"> \key<-3> \time<6/8>
    \lyric<"sel-"> b&-1*/8.
    \lyric<"dom"> e&*0
    \lyric<"is"> f
    \lyric<"heard"> g/4
    \lyric<"A"> e&/16
    \lyric<"dis-"> d
]
```

Each event is then extended so it is fully specified. In this format, each note event is a tuple of properties:

> *pitch-name, accidental, octave, twelve-tone-pitch, base-40-pitch, duration, interval, contour, scale-degree, lyric-syllable, metric-position*

Figure 4 shows the example in fully-specified symbolic representation.

Each one of these fields is used to index the database in a particular secondary corpus. For example, if the notes in the example were labeled 1 through 6, the data in the secondary indices may look something like:

- **Pitch name**
  - `a` $\longrightarrow \emptyset$
  - `b` $\longrightarrow \{1\}$
  - `c` $\longrightarrow \emptyset$
  - `d` $\longrightarrow \{6\}$
  - `e` $\longrightarrow \{2, 5\}$
  - `f` $\longrightarrow \{3\}$
  - `g` $\longrightarrow \{4\}$
- **Accidentals**
  - `n` ($\natural$) $\longrightarrow \{3, 4, 6\}$
  - `&` ($\flat$) $\longrightarrow \{1, 2, 5\}$
- **Octave**
  - `-1` (octave below middle-*c*) $\longrightarrow \{1\}$
  - `0` (octave above middle-*c*) $\longrightarrow \{2, 3, 4, 5, 6\}$
- **Duration**
  - `1/4` (quarter note) $\longrightarrow \{4\}$
  - `1/8` (eighth note) $\longrightarrow \{1, 2, 3\}$
  - `1/16` (sixteenth note) $\longrightarrow \{5, 6\}$

6.1.2 Searching using secondary indices

The search query itself is simply a series of events. Each event can be indicated as specifically or as generally as the end user (as represented by a user interface) desires. For example, the following query would match any melodic fragment that begins on a $b^\flat$ eighth note, has a sequence of 3 ascending notes, ending on a g:

```
b,&,1/8 / / / g
```

To execute a search query using secondary indices, the search engine looks up each "parameter" in their corresponding secondary indices, and retrieves tokens in the secondary index. These tokens are then looked up in the primary index, returning a list of positions. These lists are intersected to find the common elements. This list of locations is then filtered to include only those events that are sequenced according to the search query.

6.1.3 Supported user interfaces

This design supports a broad range of user interfaces. A text-based user interface may allow a user to be very specific in

```
[ \clef<"treble"> \key<-3> \time<6/8>
    b, &, -1, 10, -3,  1/8, P4, /, so, "sel-",      0
    e, &,  0,  3, 14,  1/8, M2, /, do, "dom",     1/8
    f, n,  0,  5, 20,  1/8, M2, /, re, "is",      1/4
    g, n,  0,  7, 25,  1/4, M3, \, mi, "heard",   3/8
    e, &,  0,  3, 15, 1/16, m2, \, do, "A",       5/8
    d, n,  0,  2,  9, 1/16, M2, \, ti, "dis-",  11/16
]
```

Figure 4: **Fully specified symbolic representation of the example in Figure 3**.

the query, and then incrementally remove layers of specificity until the desired match is retrieved. An audio-based user interface could be more or less specific depending on the pitch tracker's confidence in each event.

6.1.4 *Efficiency of secondary indices*

One of the efficiency problems with this approach is that the vocabularies of the individual secondary indices tend to be quite small, and thus the index lists for each atom are very large. For instance, the "pitch name" secondary index has only seven atoms in its vocabulary (*a* - *g*). "Accidentals" is even smaller: $\{\flat\flat, \flat, \natural, \sharp, \times\}$. Therefore, a search for a $b^\flat$ must intersect two very large lists: the list of all *b*'s and the list of all flats. However, the search engine can combine these secondary indices in any desired combination off-line. For example, given the "pitch name" and "accidental" indices, the search engine can automatically generate a hybrid index in which the vocabulary is all possible combinations of pitch names and accidentals. The secondary indices can be automatically combined in all possible combinations, to an arbitrary order.

6.2 Partitions

Partitioning can be used to restrict a search query to a particular part of the corpus. Each partition is a description of how to divide the corpus into discontiguous, non-overlapping regions. More specifically, each partition is a file containing a list of regions. Each region within a partition is named and has a list of its start and stop positions.

In our music search engine, the metadata is used to partition the corpus into regions. For example, all works by a given composer would make up a discontiguous region in the "composer" partition. Partitions exist for all types of metadata in the collection, including date, publisher, geographical location, etc.

In addition, we have extended partitioning to include musical elements derived directly from the GUIDO data. Regions are generated from key signatures, clefs, time signatures, measures, movements, repeats, etc. This allows for searching for a particular melody in a particular key and clef, for example.

6.2.1 *Ingestion of partition data*

When a new work is added to the corpus, the data is partitioned automatically. First, the metadata regions, such as title, composer, and date, are set to include the entire piece. As the piece is scanned, continuous musical elements, such as clef, key signature, and time signature, are regioned on the fly. Therefore, when the ingestion filter sees a "treble clef" token, all further events are added to the "treble clef" region until another clef token is encountered. Lastly, events are added to the moment regions on an event-by-event basis.

For the example in Figure 3, again assuming the notes are numbered 1 through 6, the partitions may look something like:

- **Title partition**
  "Home on the range" $\longrightarrow$ [1, 6]
- **Clef partition**
  Treble clef $\longrightarrow$ [1, 6]
- **Time signature partition**
  $\frac{6}{8} \longrightarrow [1, 6]$

6.2.2 *Searching using partitions*

Extending the example in Section 6.1.2, the user may wish to limit the search to the key signature of $E^\flat$*-major*:

```
( b,&,1/8 / / / g ) @ key:"E& major"
```

Here the non-partitioned search query is performed as described above, and then the results are intersected with the results of the partition lookup. Since in our case, the entire range of notes [1, 6] is in the key signature of $E^\flat$*-major*, the query will retrieve the example in Figure 3.

6.2.3 *Searching with simultanaeity using partitions*

Scores are also partitioned at the most atomic level by "moments." A moment is defined as a point in time when any event begins or ends in any part. Moments almost always contain multiple events, and events can belong to multiple moments (e.g. when a half note is carried over two quarter notes in another part). Each moment within a score is given a unique numeric identifier, and all events active at a given point are included in a moment region. In this way, one can search for simultaneous polyphonic events very efficiently.

To explain this further, Figure 5 shows the example measure with its assigned moment numbers. Each event is assigned to one or more moments so that it can be determined which, if any, of the events are active at the same time. These moment numbers are used to create regions. For example, the dotted half note in the left hand of the piano part would be assigned to all seven moment regions.

To perform searches involving simultanaeity, the query for each part is performed separately, and then the results are intersected based on their moments. Only the query results that occur at the same time (existing in the same moment regions) will be presented to the user.

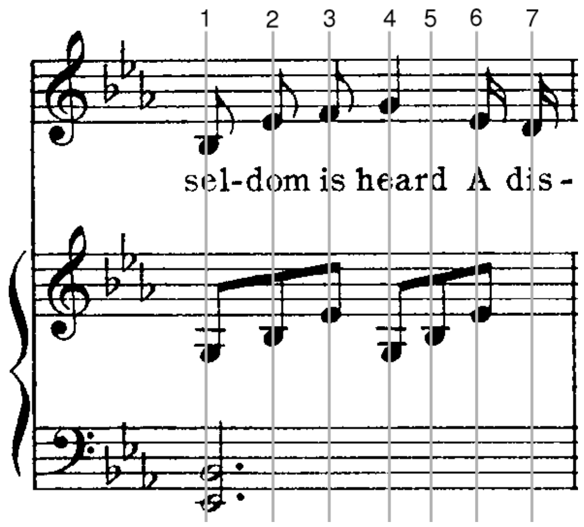


Figure 5: **The example measure of music showing moment numbers.**

6.3 Regular expressions

The core search engine supports a full complement of POSIX-compliant regular expressions. Regular expressions, a large topic beyond the scope of this paper, are primarily used for pattern-matching within a search string (Friedl 1997).

Many users find regular expressions difficult and cumbersome ways to express searches. However, it is our intent that most of these details will be hidden from the user by appropriate interfaces. For example, regular expressions would be very useful for an interface that allowed searching by melodic similarity. What is important to our present research is that regular expressions are supported in the core search engine, leaving such possibilities open.

7. CONCLUSION

Based on existing advanced natural-language search techniques, we have developed an expressive and efficient musical search engine. Its special capabilities include: secondary indices for gradiated specificity, partitions for selective scope and simultanaeity, and regular expressions for expressive pattern matching. This allows users with different search needs to access the database in powerful and efficient ways.

8. ACKNOWLEDGMENTS

The second phase of the Levy Project is funded through the NSF's DLI-2 initiative (Award #9817430), an IMLS National Leadership Grant, and support from the Levy Family. We would like to thank David Yarowsky of the Department of Computer Science for motivation and discussion.

9. REFERENCES

Choudhury, S., T. DiLauro, M. Droettboom, I. Fujinaga, B. Harrington, and K. MacMillan. 2000. Optical music recognition within a large-scale digitization project. *ISMIR 2000 Conference*.

Choudhury, G. S., T. DiLauro, M. Droettboom, I. Fujinaga, and K. MacMillan. 2001. Strike up the score: Deriving searchable and playable digital formats from sheet music. *D-Lib Magazine*: 7(2).

DiLauro, T., G. S. Choudhury, M. Patton, J. W. Warner, and E. W. Brown. 2001. Automated name authority control and enhanced searching in the Levy collection. *D-Lib Magazine*: 7(4).

Friedl, J. E. F. 1997. *Mastering regular expressions.* Sebastopol, CA: O'Reilly.

Hewlett, W. B. 1992. A base-40 number-line representation of musical pitch notation. *Musikometrika* 4: 1–14.

Hewlett, W. B., and E. Selfridge-Field. 1998. *Melodic similarity: Concepts, procedures and applications.* Cambridge, MA: MIT Press.

Hoos, H. H., and K. Hamel. 1997. GUIDO music notation: Specification Part I, Basic GUIDO. Technical Report TI 20/97, Technische Universität Darmstadt.

Huron, D., W. Hewlett, E. Selfridge-Field, et al. 2001. How Themefinder works. `http://www.themefinder.org`

Huron, D. 1999. *Music research using Humdrum: A user's guide.* Menlo Park, CA: Center for Computer Assisted Research in the Humanities.

McNab, R. J., L. A. Smith, D. Bainbridge, and I. H. Witten. 1997. The New Zealand Digital Library MELody inDEX. *D-Lib Magazine*: 3(5).

McNab, R. J., L. A. Smith, I. H. Witten, and C. L. Henderson. 2000. Tune retrieval in the multimedia library. *Multimedia Tools and Applications* 10(2/3): 113–32.

Witten, I., A. Moffat, and T. Bell. 1999. Managing gigabytes. 2nd Ed. San Francisco: Morgan Kaufmann.

A technique for “regular expression” style searching in polyphonic music

Matthew J. Dovey
Visiting Research Fellow
Dept. of Computer Science
Kings College, London
+44 1865 278272
matthew.dovey@las.ox.ac.uk

ABSTRACT
This paper discussed some of the ongoing investigative work on integrating these two systems conducted as part of the NSF/JISC funded OMRAS (Online Music Retrieval and Searching) project into polyphonic searching of music. It describes a simple and efficient “piano-roll” based algorithm for locating a polyphonic query within a large polyphonic text. It then describes ways in which this algorithm can be modified without affecting the performance to allow more freedom in the how a match is made, allowing queries which involve something akin to polyphonic regular expressions to be located in the text.

1. INTRODUCTION
The OMRAS (Online Music Retrieval And Searching) project is a three year collaborative project between Kings College London and the Center for Intelligent Information Retrieval, University of Massachusetts. Its primary aim is to look at various issues surrounding content based searching of polyphonic music; current research in content based music searching has primarily concentrated on monophonic music, that is to say music consisting of only a single melodic line and ignoring the complexities find in a more complex music texture for example as found in an say an orchestral symphony.

Different computer representations of music roughly fall into two basic categories: those representing the audio of the music such as a typical wave or aiff file (or more topically MP3), and those representing the symbolic structure of the music as indicated in a typically written musical score. The audio file formats typically represent an actual performance of a piece, whilst the symbolic formats represent the composer’s instructions and guidelines to the performer. In practice, as described in Byrd and Crawford (2001) [1], these are two extremes of a spectrum with various formats falling in between which contain elements of both such as MPEG7 and MIDI. MIDI, for example, was originally designed for representing music performances but is closer to a symbolic notation than an audio representation (and is used to indicate instructions for a performance rather than record a performance). (Byrd and Crawford actually talk of three distinct types – the two referred to above plus MIDI representing the middle ground)

One aspect of OMRAS is to look at conversion of formats moving along this spectrum. Moving from symbolic notations to audio is fairly straightforward but this is to be expected since the role of the symbolic notation is to describe how to enact a performance of the music. Going from the audio to a symbolic description of how to perform that audio is far more difficult. A good comparison would be between a performance of a play, and the script and stage-directions for performing that play.

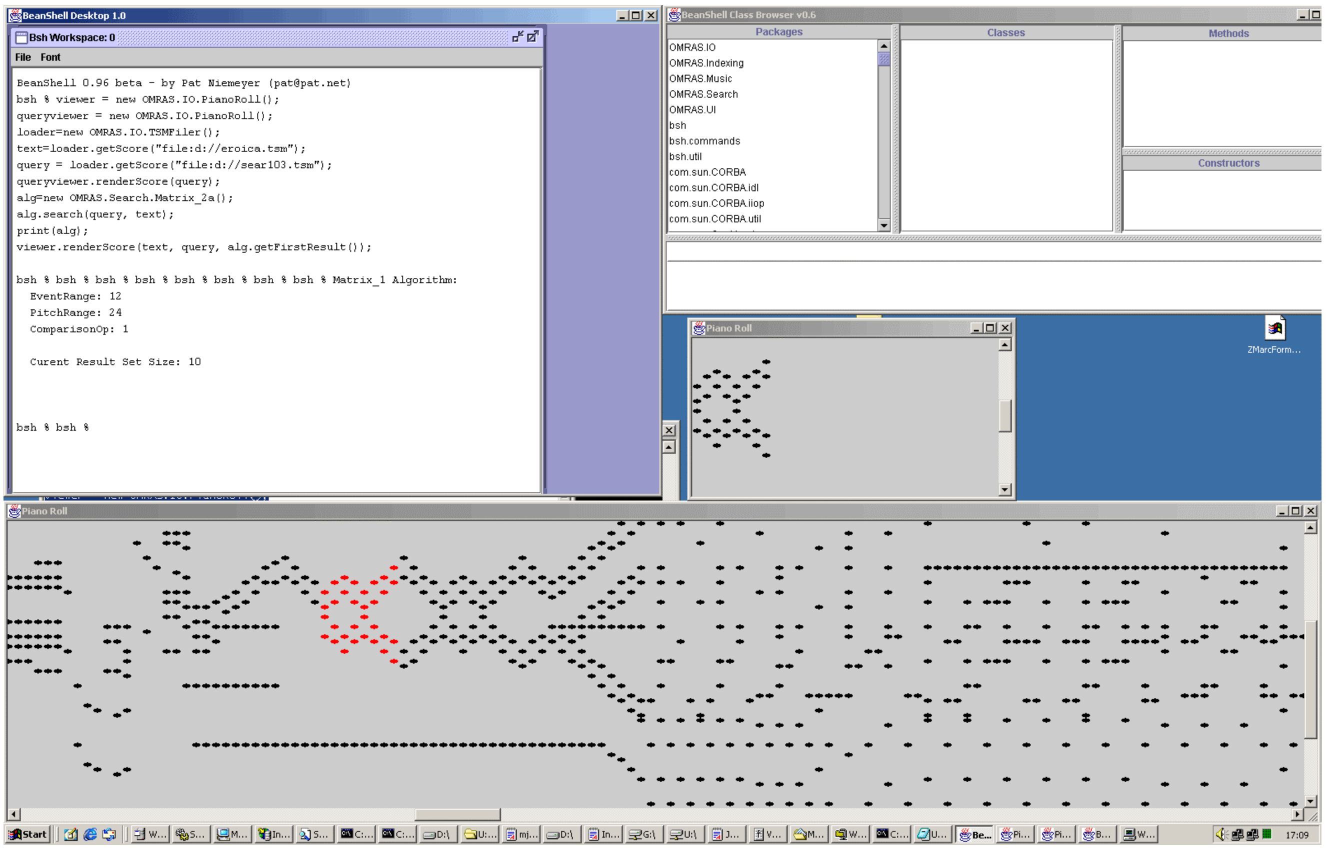
A second aspect on OMRAS is to consider the searching of music by content. As indicated above we are concentrating on polyphonic music since this is currently neglected in the existing work. One of the groups in OMRAS has been looking at audio work, but this paper concentrates on searching symbolic formats which represent Common Music Notation (CMN) of the Western tradition of music. Most work on music searching has concentrated on searching within monophonic, single voice music, often applying techniques derived from the text-retrieval world (e.g. Downie (1999) [5]). There are been some approaches at polyphonic searching (e.g. Uitdenbogerd and Zobel (1998) [9], Holub, Iliopoulos, Melichar and Mochard (1999) [6] and Lemström and Perttu (2000) [7]). This paper will outline some of the algorithms we have developed for this purpose, which we believe are more versatile for regular expression style searching than previous work in this area.

A key mission statement of the joint JISC/NSF International Digital Library Initiative, which is funding the OMRAS work is that the projects should make existing digital collections more accessible. We feel that the work of OMRAS makes digital collections of music more accessible by providing content based searching possible, in addition to standard metadata searched such as by composer or title. OMRAS is collaborating with the JISC funded JAFER project at Oxford University to provide integration with existing music library catalogues[1].

2. THE OMRAS TEST FRAMEWORK
Within the OMRAS project we have written a test framework using the Java programming language for testing the performance of various search algorithms and techniques. The framework is command line driven (and not really designed for novice users). From the command line we can load a music file into the system), can load in different algorithms and can load in different user interface components for displaying and editing queries and results. The framework allows us to experiment with a number of different algorithms and a number of different representations of music. A screenshot of the command line framework in use is given in figure 1.

The framework has been designed to take advantage of Java’s object-oriented nature: all the components such as file format modules, user interface widgets and search algorithms are

[1] http://www.lib.ox.ac.uk/jafer and http://www.jafer.org



```
BeanShell 0.96 beta - by Pat Niemeyer (pat@pat.net)
bsh % viewer = new OMRAS.IO.PianoRoll();
queryviewer = new OMRAS.IO.PianoRoll();
loader=new OMRAS.IO.TSMFiler();
text=loader.getScore("file:d://eroica.tsm");
query = loader.getScore("file:d://sear103.tsm");
queryviewer.renderScore(query);
alg=new OMRAS.Search.Matrix_2a();
alg.search(query, text);
print(alg);
viewer.renderScore(text, query, alg.getFirstResult());

bsh % bsh % bsh % bsh % bsh % bsh % bsh % bsh % Matrix_1 Algorithm:
  EventRange: 12
  PitchRange: 24
  ComparisonOp: 1

  Curent Result Set Size: 10



bsh % bsh %
```

Figure 1

implemented as independent java objects, so that the entire framework is modular. These java objects can be manipulated in the framework using a scripting interface for java such as BeanShell[2], but selected components can be used in other software. In particular, some engineering undergraduates have been working with OMRAS to build a GUI interface to the framework in addition to the current command line interface. Collaborating with the JISC funded JAFER Project at Oxford University[3] we have reused the components to build a prototype demonstrating how our algorithms for content based searching can be integrated into existing bibliographic oriented music library catalogue systems (Dovey, M (2001) [4]).

At the moment we only have modules for handling a small number of file formats (including MIDI) but are working on others most notably GUIDO and Kern[4]. The user interface components are based on piano roll type displays, but we are working on incorporating better CMN software for displaying scores into the framework in the near future. We also have objects for rendering scores into audio, using the Java Media Framework.

[2] http://www.beanshell.org

[3] http://www.jafer.org and http://www.lib.ox.ac.uk/jafer

[4] A good reference of various music file formats can be found in Selfridge-Field (1997) [8].

3. BASE ALGORITHM FOR SEARCHING POLYPHONIC MUSIC

3.1 "Piano Roll" Model of CMN

Common Music Notation (CMN) is a very complex pictorial language for describing music with a number of cues as to how it should be performed. Its pictorial nature proved very difficult for in the history of the printing press; in many cases the most efficient means to produce a printed score was to create an engraving rather than attempt to produce a generic type-set for music. It is not surprising, therefore, that CMN produces complex problems for computer based representations. We are working with a very simple model for representing music, but one, which can provide a skeleton for re-introducing some of the complexities of CMN.

The representational model of music we are currently using can be compared to that of a piano roll, where punched holes indicate whether a note should be playing. The horizontal position in the roll indicates the time of the note, whilst the vertical position indicates the pitch of the note. In our base model we only concern ourselves with the beginnings of notes (in MIDI terms with the Note On events). We consider a musical "event" to be a list of notes which begin to play at the same time (in some ways this is similar to a chord but is more generalized). Whilst not an ideal terminology, we will use the term "event" in this manner, for the purposes of this paper. We only introduce a new musical event where needed, i.e. because a new note has begun to play. In this model we can regard a piece of music to be a sequence of musical events. For example the extract in Figure 2,

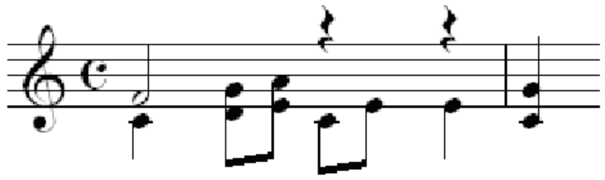


Figure 2

could be represented as the following list of musical events:

1. F, C
2. G, D
3. A, E
4. C
5. E
6. E
7. G, C

This leads an obvious representation as a matrix of zeros and ones indicating whether a note begins to play at that time and pitch and this is used as an effective way to display queries and results. For example the more complex extract in Figure 3

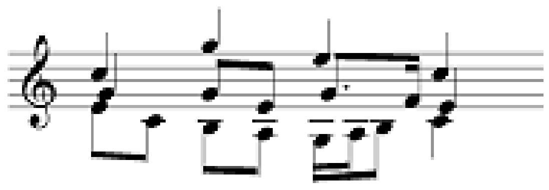


Figure 3

can be represented in this piano roll format as in Figure 4

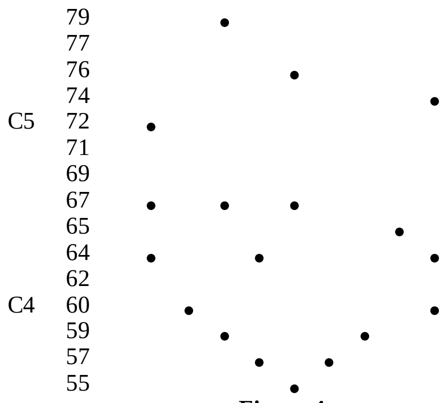


Figure 4

In the OMRAS framework, we represent as an XML structure such as:

```
<score>
  <event>
    <note pitch="72"/>
    <note pitch="67"/>
    <note pitch="64"/>
  </event>
    <note pitch="60"/>
  <event>
  </event>
```
etc…

By representing this structure as an XML Schema document we can generate the Java object model for this format from the XML Schema description. Representing CMN in this manner allows us to add additional music information. For example onset time (in milliseconds as for MIDI, or metrical information such as bar number of beat within the bar) can be added as additional attributes for the event element; duration of notes, voicing, instrumentation etc. can be added as additional attributes to the note element.

3.2 Searching in a "piano roll" model

In the model, described above the typical search problem can be expressed as follows: given a query as a sequence of musical events and a text to perform the query on as a sequence of musical events, we are trying to find a sequence of musical events in the text such that each musical event in the query is a subset of the musical event in the text. This again is best illustrated by an example. Consider the musical extract in figure 5.

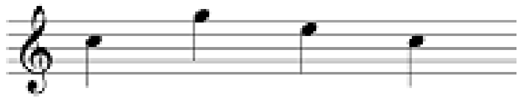


Figure 5

As a piano roll, this would be represented as in Figure 6.

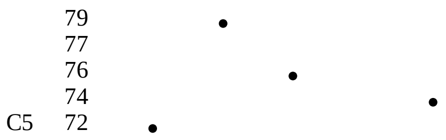
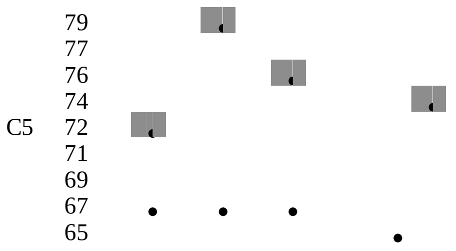


Figure 6

A potential match is illustrated in Figure 7. As can be seen, there is some allowance in that intervening musical events are allowed between matched events in the text. The freedom of this can be limited to avoid too many spurious matches.



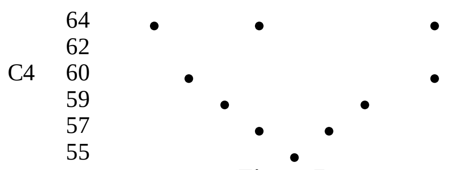


Figure 7

In musical terms, it is also necessary to allow transpositions of pitch. Figure 8 gives a second match at a different transposition.

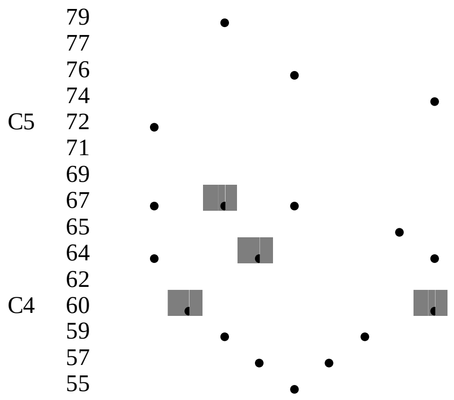


Figure 8

There are other degrees of freedom that need to be allowed in searching, in order to accommodate inaccurate recall of the user, and also inaccuracies in the data, for example if the musical information has been created by automatic transcription of either audio data or via optical recognition of the printed score. A fuller description of these areas of fuzziness is given in Crawford and Dovey (1999) [2].

3.3 A "piano roll" based algorithm

In Dovey (1999) [3], we presented a matrix-based algorithm for searching the music using the piano roll abstract model described above. That algorithm has underpinned much of our subsequent work, however there were some serious performance issues with the algorithm described there in pathological cases. The algorithm described here is a more refined version of that algorithm which performs in effectively linear time for a given text.

Let the text be represented by the sequence $<T_m>$ and the query by the sequence $<S_n>$ and consider the case when we can allow k intervening musical events in the text between the matches for consecutive events in the query.

In essence, we are performing a pass over the text for each musical event in the query. In the first case we are looking for the matches for the first event in the query to occur in the text. For all subsequent passes we are looking for an occurrence of the n^{th} term of the query occurring within k musical events of an occurrence of the $n\text{-}1^{th}$ term found in the previous pass.

Mathematically speaking, we construct the matrix M, where

$M_{i\,j} =$ $k+1$ iff $S_j \subseteq T_i$ and ($M_{i-1\,j-1} \neq 0$ or $i = 0$)

$M_{i\,j-1}-1$ iff (not $S_j \subseteq T_i$) and ($M_{i\,j-1} \neq 0$)

0 otherwise

A result can be found by traversing the last row of the matrix constructed for the value k+1, this indicates by it horizontal position the match in the text for the last event in the query, reading to the left from this position in the row above for the value k gives the match for the penultimate event in the query and so on. We then repeat the process for each pitch transposition of the query sequence (i.e. where the value pitch of note in the sequence $<S_n>$ is transposed up or down by the same amount). We clearly need not consider any transposition such that the transposed sequence of $<S_n>$ and the sequence $<T_m>$ have not pitches in common.

In the worse case this algorithm can never take more that m x n steps to locate each pitch transposition, and the number of transpositions will never exceed the range of pitches which occur in $<T_m>$.

The following worked example would make this clearer. Let use consider the text in figure 4. Then our text is

T_0 = 72, 76, 64

T_1 = 60

T_2 = 79, 67, 59

etc.

Let us consider a simple query

S_0 = 67, 59

S_1 = 67, 55

S_2 = 59

For this case we allow one intervening gap, i.e. k=1.

We build the first row of the matrix by writing k+1 (i.e. 2) where S_0 contains all the pitches in T_i, otherwise if the value of the preceding square is non-zero we write that value less one, otherwise zero. So the first line becomes

	T_0	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8
S_0	0	0	2	1	0	0	0	0	0

For the second line we perform a similar operation for S_1, however we only write k+1 if all the pitches of S_1 are in T_i and the value of the preceding square in the row above is non-zero. So we now have

	T_0	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8
S_0	0	0	2	1	0	0	0	0	0
S_1	0	0	0	0	2	1	0	0	0

We then repeat for S_2 giving

	T_0	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8
S_0	0	0	2	1	0	0	0	0	0
S_1	0	0	0	0	2	1	0	0	0
S_2	0	0	0	0	0	0	2	1	0

Note that although S_2 occurs within T_2 the preceding square above in the second row is zero, so we do not write k+1 (i.e. 2) here. The results can now be found in linear time by reading the matrix backwards looking for the occurrence of the value 2 i.e.

here $T_2\, T_4\, T_6$ is our match. We can then repeat for the other 15 possible pitch transpositions of the query.

There is an optimization to avoid unnecessary comparisons of musical events. For the row i of the matrix there exist a maximal s such that $M_{i\,j} = 0$ for all $j < s$. Similarly there exists a minimal t such that $M_{i\,j} = 0$ for all $j > t$. From this we can deduce that $M_{i+1\,j} = 0$ for all $j < s$ and $j > t+k$, i.e. we can limit our attention when constructing the next row of the matrix to the subsequence T_s, $T_{s+1} \ldots T_{t+k}$ of the text. Of course when $t < s$ there can be no matches for the query in the text. This optimization can cut processing time dramatically.

The above method forms the basis for more complex searching techniques where we have a query which in many ways resembles a regular expression for a musical phrase.

4. EXTENSIONS OF BASE ALGORITHM TO HANDLE REGULAR EXPRESSION TYPE QUERIES

4.1 Comparisons of Music Events

In the algorithm described in section 3, we considered only one way in which a musical event in the query can match a musical event in the text, namely that of inclusion. i.e. we say that a musical event S_j matches T_i if all the notes in S_j are in T_i (i.e $S_j \subseteq T_i$). Dovey (1999) [3] considers four such comparison operators:

- $S_j \subseteq T_i$ – as described here.
- $S_j = T_i$ – for exact matching
- $S_j \cap T_i \neq \emptyset$ – S_j here represents a disjunctive lists of notes we wish to match
- $T_i \subseteq S_j$ – for symmetry of the operators

This is generalized in Crawford and Dovey (1999) [2], so that these become just four of the most typical comparison operators in a lattice of all possible comparison operators with equals being the Top of this lattice and always matches the Bottom. In general we have some relationship **R** which defines whether two events "match" or not. Other typical relationships may include:

- $\mathbf{R_h}$: S_j and T_i are harmonically similar.
- $\mathbf{R_r}$: For each note in S_j there is a note in T_i whose pitch is within a given range of the pitch of the note in S_j
- $\mathbf{R_{r*}}$: For all but x notes in S_j there is a note in T_i whose pitch is within a given range of the pitch of the note in S_j

Our base algorithm described in section 3.3 can easily be extended to accommodate any such relationship **R** without any lose of performance. In this case given our relationship **R** we construct the matrix M where

$M_{i\,j} =$ $k+1$ iff $S_j\, \mathbf{R}\, T_i$ and $(M_{i-1\,j} \neq 0$ or $i = 0)$

$M_{i\,j-1}-1$ iff (not $S_j\, \mathbf{R}\, T_i$) and $(M_{i\,j-1} \neq 0)$

0 otherwise

We can further generalize this. Given that we perform a pass over $\langle T_m\rangle$ for each S_j in $\langle S_n\rangle$, we can use a different relationship for each pass again without any loss in performance. i.e. we now have a sequence $\langle \mathbf{R_n}\rangle$ where $\mathbf{R_j}$ describes the comparison operation for locating matches of S_j in $\langle T_m\rangle$. In this case we construct the matrix M where

$M_{i\,j} =$ $k+1$ iff $S_j\, \mathbf{R_j}\, T_i$ and $(M_{i-1\,j-1} \neq 0$ or $i = 0)$

$M_{i\,j-1}-1$ iff (not $S_j\, \mathbf{R_j}\, T_i$) and $(M_{i\,j-1} \neq 0)$

0 otherwise

This allows use for each event in the query to specify how the match will be found in the text using a polyphonic comparison range from exact match to more fuzzy matching comparisons.

4.2 Adding additional dimensions to notes

So far we have considered a note only to have a single value indicating its pitch. Clearly the algorithm described in section 3.3 and the enhancements described in section 4.1 would apply to any property. In the case of duration we would clearly be more interested in a comparison operator of the type $\mathbf{R_r}$ or $\mathbf{R_{r*}}$. For instrumentation or voicing we would be more interested in strict equality. If we define each note to be an n-tuple of the properties we are interested in we can perform a match against all these properties by defining a suitable composite comparison operator **R** without affecting the efficiency of our algorithm.

For example, in the case of three properties pitch, duration and instrument then **R** might behave as follows

$S_j\, \mathbf{R}\, T_i$ if for almost all notes in S_j there is a note in T_j with a similar pitch, a similar duration and in the same voice (with some suitable parameters defining the ranges for similar and almost all).

4.3 Varying gaps between event matches

In section 4.1, we showed that we could have a different comparison operator for each term in the query. However, we still only have a single value of k, determining the allowed "gap" between matched events in the text, for the entire query. This also need not be the case. We can instead consider the sequence $\langle k_n\rangle$ where k_j indicates the number of allowed gaps that can occur in the text after a match of S_j before a match of S_{j+1} occurs. k_n clearly has no meaning but must be non-zero for the algorithm to work properly. Modifying the generic form of the algorithm from section 3.3 gives

$M_{i\,j} =$ k_j+1 iff $S_j \subseteq T_i$ and $(M_{i-1\,j-1} \neq 0$ or $i = 0)$

$M_{i\,j-1}-1$ iff (not $S_j \subseteq T_i$) and $(M_{i\,j-1} \neq 0)$

0 otherwise

When reading the matrix for results we now look for the value k_j+1 to indicate matches where j is the current row of the matrix.

Modifying the form of the algorithm given at the end of section 4.1 gives

$M_{i\,j} =$ k_j+1 iff $S_j\, \mathbf{R_j}\, T_i$ and $(M_{i-1\,j-1} \neq 0$ or $i = 0)$

$M_{i\,j-1}-1$ iff (not $S_j\, \mathbf{R_j}\, T_i$) and $(M_{i\,j-1} \neq 0)$

0 otherwise

So far we have considered the "gap" to be measured in terms of the number of events that can occur. Clearly in a piece of music the time between two consecutive events can vary. We can incorporate this into the algorithm by allowing the "gap" to be specified in terms of the time between matched events rather than the number of intervening events. We define an monotonic increasing function O on $\langle T_m\rangle$ where

$O(T_i)$ is the time at which T_i occurs.

The units could be milliseconds as in MIDI or could be based on a musical metric such as number of quarter notes.

In this case we set k to be the maximum allowed duration between matched events in the text. The base algorithm from section 3.3 now becomes

$M_{i\,j} =$ k iff $S_j \subseteq T_i$ and $((M_{i-1\,j} > 0$ and $M_{i-1\,j} < k)$ or $i = 0))$

$M_{i\,j-1} - (O(T_i) - O(T_{i-1}))$

iff (not $S_j \subseteq T_i$) and $(M_{i\,j-1} - (O(T_i) - O(T_{i-1})) > 0)$

0 otherwise

Reading the results matrix now involves looking for the occurrences of the value of k. The modifications described in sections 4.1 and 4.2 can be applied to this form.

4.4 Omission of events in the query

The final modification to the base algorithm is to allow a match to be found even if some of the events in the query are not present in the matched subsequence of the text. Essentially when parsing the matrix we consider non-zero values in not only the line immediately above but also previous lines. To avoid excessive parsing of the matrix which would degrade the performance of the algorithm we can build a matrix of ordered pairs. For notational purposes, given an ordered pair p, we will denote the first value as p[0] and the second as p[1].

Working with the base algorithm from section 3.3 and allowing a sequences of up to l events from the query to be omitted from the match we build the matrix

$M_{i\,j} =$ $(k+1, l)$ iff $S_j \subseteq T_i$ and $(M_{i-1\,j-1} \neq (0, 0)$ or $i = 0)$

$(M_{i\,j-1}[0] - 1, M_{i\,j-1}[1])$ iff (not $S_j \subseteq T_i$) and $(M_{i\,j-1}[0] \neq 0)$

$(M_{i-1\,j}[0], M_{i-1\,j}[1] - 1)$ iff (not $S_j \subseteq T_i$) and $(M_{i-1\,j}[1] \neq 0)$

(0, 0) otherwise

Parsing the matrix for matches is a matter of looking for the value k+1 as the first member of any ordered pairs. Limiting the number of sequences omitted can be performed when parsing the matrix for results. Again the modifications described in sections 4.1, 4.2 and 4.3 can also be applied in conjunction with this modification.

4.5 An XML query structure

Combined these modifications allow us to search a musical text given a polyphonic query and a number of degrees of freedom. Considering just note pitches and durations, we can use the following XML structures such as the following to write a query allowing some of these degrees of freedom

```
<query omissions="0">
  <event
    following-gap=2
    following-gap.units="events"
    content-omissions.max="0"
    content-omissions.min="0">
     <note
       pitch.min="60"
       pitch.max="65"
       duration.min="1"
       duration.max="1"/>
  </event>
```
etc…

Here the **omissions** attribute of the **query** element tells us that we do not allow any events of the query to be omitted in the match. The **following-gap** attribute of the **event** element tells us the "gap" that can occur after this event in the text before the next event must occur; the **following-gap.units** whether this is measure in events or durations. The **content-omissions.min** and **content-omissions.max** tell us how many notes can be omitted from the match in order for it still to be classified as a match. The **pitch.min**, **pitch.max**, **duration.min** and **duration.max** attributes of the note element define ranges for a note in the text to match.

Whilst the algorithms described here can efficiently cope with this sort of query, there are other queries which can be handled which cannot be articulated in this XML structure.

5. FURTHER WORK

At present the algorithms described here merely locate matches given a number of degrees of freedom. There is no attempt to rank these matches. The calculation of a rank could be made as the matrix is parsed for matches and some pre-parsing could also be performed as the matrix is built. Crawford and Dovey (1999) [2] outline a number of similarity measures which could be used in creating a ranking algorithm such as completeness, compactness, musical salience, harmonic progression, rhythmic similarity, metrical congruity. This ranking process is essential before we can fully evaluate the effectiveness of this type of approach to music information retrieval.

Given the amount of freedom these algorithms allow in the specification of a query there is a need for query languages and GUI query interfaces to allow users to easily express such queries. Some work has already been undertaken in this area. The XML structure above is very much a working structure and not intended for general use.

6. ACKNOWLEDGEMENTS

I would like to thank Tim Crawford of Kings College London and Don Byrd of the University of Massachusetts at Amherst for their assistance in working on this topic and their patience in the amount of time it has taken to put together a description of these algorithms. I would also like to acknowledge the NSF/JISC IDLI and JISC DNER programmes which are funding the OMRAS and JAFER projects upon which this work is based.

7. REFERENCES

[1] Byrd, D. and Crawford, T. (2001) Problems of Music Information Retrieval in the Real World. To appear in Information Processing and Management.

[2] Crawford, T and Dovey, M. "Heuristic Models of Relevance Ranking in Musical Searching", *Proceedings of the Fourth Diderot Mathematical Forum.*, Vienna, 1999.

[3] Dovey, M, 'An algorithm for locating polyphonic phrases within a polyphonic piece', *Proceedings of the AISB'99 Symposium on Musical Creativity*, Edinburgh, April, 1999. Pages 48-53.

[4] Dovey, M, 'Adding content-based searching to a traditional music library catalogue server', *Proceedings of the Joint Conference on Digital Libraries*, Roanoake, VA, 2001.

[5] Downie, J. S., 'Music retrieval as text retrieval: simple yet effective', *Proceedings of the 22nd International Conference on Research and Development of Information Retrieval*, Berkeley, CA, 1999. Pages 297-298.

[6] Holub, J., Iliopoulos, C. S., Melichar, B. and Mouchard, L. 'Distributed String matching using finite automata', *Proceedings of the 10th Australiasian Workshop on Combinatorial Algorithms'*, Perth, 1999. Pages 114-128.

[7] Lemström, K. and Perttu, S., 'SEMEX – an efficient music retrieval protoype', *First International Symposium on Music Information Retrieval*, University of Massachusetts, Plymouth 2000.

[8] Selfridge-Field, E. (editor), *Beyond MIDI*, CCARH 1997. ISBN 0262193949.

[9] Uidebogerg, A. L. and Zobel, J. 'Manipulation of music for melody matching', *ACM Multimedia 98 Proceedings*, Bristol 1998. Pages 235-240.

An Approach Towards A Polyphonic Music Retrieval System

Shyamala Doraisamy
Dept. of Computing
Imperial College,
London SW7 2BZ
+44-(0)20-75948230
sd3@doc.ic.ac.uk

Stefan M Rüger
Dept. of Computing
Imperial College
London SW7 2BZ
+44-(0)20-75948355
srueger@doc.ic.ac.uk

ABSTRACT

Most research on music retrieval systems is based on monophonic musical sequences. In this paper, we investigate techniques for a full polyphonic music retrieval system. A method for indexing polyphonic music data files using the pitch and rhythm dimensions of music information is introduced. Our strategy is to use all combinations of monophonic musical sequences from polyphonic music data. 'Musical words' are then obtained using the n-gram approach enabling text retrieval methods to be used for polyphonic music retrieval. Here we extend the n-gram technique to encode rhythmic as well as interval information, using the ratios of onset time differences between two adjacent pairs of pitch events. In studying the precision in which intervals are to be represented, a mapping function is formulated in dividing intervals into smaller classes. To overcome the quantisation problems that arise with using rhythmic information from performance data, an encoding mechanism using ratio bins is also adopted. We present results from retrieval experiments with a database of 3096 polyphonic pieces.

1. INTRODUCTION

Music documents encoded in digital formats have rapidly been increasing in number with the advances in computer and network technologies. Managing large collections of these documents can be difficult and this has consequently motivated research towards computer-based music information retrieval (IR) systems. Music documents encompass documents that contain any music-related information such as music recordings, musical scores, manuscripts or sketches and so on [1]. Many studies have been carried out in using the music-related information contained in these documents for the development of content-based music IR systems. Such systems retrieve music documents based on information such as the incipits, themes and instrument families. However, most of these content-based IR systems are still research prototypes. Music IR systems that are currently in wide-spread use are systems that have been developed using meta-data such as file-names, titles and catalogue references.

One common approach in developing content-based music IR systems is with the use of pitch information. Examples of such systems are Themefinder [2] and Meldex [3]. However, these systems were developed using monophonic musical sequences, where a single musical note is sounded at one time, as opposed to polyphonic music where more than one note is sounded simultaneously at any one point in time. With vast collections of polyphonic music data available, research on polyphonic music IR is on the rise [4].

Our aim is the development of a polyphonic music IR system for retrieving a title and performance of a musical composition given an excerpt from a musical performance as a query. For content-based indexing, we use the pitch and rhythm dimensions of music information and propose an approach for indexing full polyphonic music data. In this paper we present our approach and evaluate it using a database of polyphonic pieces.

The paper is structured as follows: Section 2 highlights some of the issues and challenges in content-based indexing. Section 3 presents the approach taken in using the pitch and duration information for indexing. The steps in constructing n-grams from polyphonic music data and the mechanism of extending the representation to include rhythm information are outlined. The empirical analysis performed and approach for encoding patterns derived from n-gramming is presented. Section 4 reports the retrieval experiments using ranked retrieval and evaluation results using the mean reciprocal rank measure of our polyphonic music IR system.

2. ISSUES IN CONTENT-BASED INDEXING AND RETRIEVAL OF MUSICAL DATA

The problem of varying user requirements is common to most IR systems. Music IR systems are no exception. Music librarians, musicologists, audio engineers, choreographers and disc-jockeys are among the wide variety of music IR users with a wide range of requirements [1]. For example, with a musical query where the user plays a recording or hums a tune, one user could possibly require all musical documents with the same key to be retrieved while another user's requirement might be to obtain all documents of the same tempo. Looking at another example where a musical composition's title is queried, one user could require the composer's full-name and another user might need to know the number of times the violin had a solo part in the composition. Knowledge of user requirements is an important aspect in developing useful indexes, and with music IR systems this challenge is compounded with others such as the multiple dimensions of music data and digital music data formats.

Music data are multi-dimensional; musical sounds are commonly described by their pitch, duration, dynamics and timbre. Most music IR systems use one or two dimensions and these vary based on types of users and queries. Selecting the appropriate dimension for indexing is an important aspect in developing a useful music IR system. Indexing a system based on its genre class would be useful for a system that retrieves music based on mood but not for a system where a user needs to identify the title of a music piece queried by its theme.

The multiple formats in which music data can be digitally encoded present a further challenge. These formats are generally categorised into a) highly structured formats such as Humdrum [5] where every piece of musical information on a piece of musical score is encoded, b) semi-structured formats such as MIDI in which sound event information is encoded and c) highly un-structured raw audio which encodes only the sound energy level over time. Most current music IR systems adopt a particular format and therefore queries and indexing techniques are based upon the dimensions of music information that can be extracted or inferred from that particular encoding method.

There are many approaches for the development of music IR systems. Some of these include the use of approximate matching techniques in dealing with challenges such as recognising melodic similarity [6], the use of standard principles of text information retrieval and exact matching techniques that demand less retrieval processing time [7,8].

3. A TECHNIQUE FOR INDEXING POLYPHONIC MUSICAL DATA

3.1 Pattern extraction

The approach we take for indexing is full-music indexing, similar to full-text indexing in text IR systems. This approach was studied by Downie [8], where a database of folksongs was converted to an interval-only representation of monophonic 'melodic strings'. Using a gliding window, these strings were fragmented into length-n subsections or windows called n-grams for music indexing.

With polyphonic music data, a different approach to obtaining n-grams would be required since more than one note can be sounded at one point in time (known as the onset time in this context). In sorting polyphonic music data with ascending onset times and dividing it into windows of n different adjacent onset times, one or more possible monophonic 'melodic string(s)' can be obtained within a window. The term melodic string used in this context may not be a melodic line in the musical sense. It is simply a monophonic sequence extracted from a sequence of polyphonic music data.

Various approaches in deriving patterns from unstructured polyphonic music for computer-based music analysis have been investigated in a study by Crawford et. al. [9]. The approach taken for our study would be a musically unstructured but an exhaustive mechanism in obtaining all possible combinations of monophonic sequences from a window for the n-gram construction. Each n-gram on its own is unlikely to be a musical pattern or motif but a pattern amenable for digital string-matching. The n-grams encoded as musical words using text representations would be used in indexing, searching and retrieving a set of sequences from a polyphonic music data collection.

The summary of steps taken in obtaining these monophonic musical sequences is as follows:

Given a polyphonic piece in terms of ordered pairs of onset time and pitch sorted by onset time,

1. Divide the piece using a gliding window approach into overlapping windows of n different adjacent onset times
2. Obtain all possible combinations of melodic strings from each window

N-grams are constructed from the interval sequence(s) of one or more monophonic sequence(s) within a window. Intervals (the distance and direction between adjacent pitch values) are a common mechanism for deriving patterns from melodic strings, being invariant to transpositions [10]. For a sequence of *n* pitches, an interval sequence is derived with *n-1* intervals by Equation (1).

$$Interval_i = Pitch_{i+1} - Pitch_i \qquad (1)$$

To illustrate the pattern extraction mechanism for polyphonic music data, the first few bars of Mozart's Alla Turca, as shown in Figure 1, is used. The performance data of the first two bars of the piece was extracted from a MIDI file and converted into a text format, as shown in Figure 2(a). The left column contains the onset times sorted in ascending order, and the corresponding notes (MIDI semitone numbers) are on the right column. The performance visualised on a time-line is shown in Figure 2 (b).

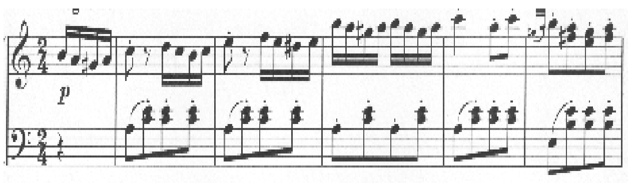
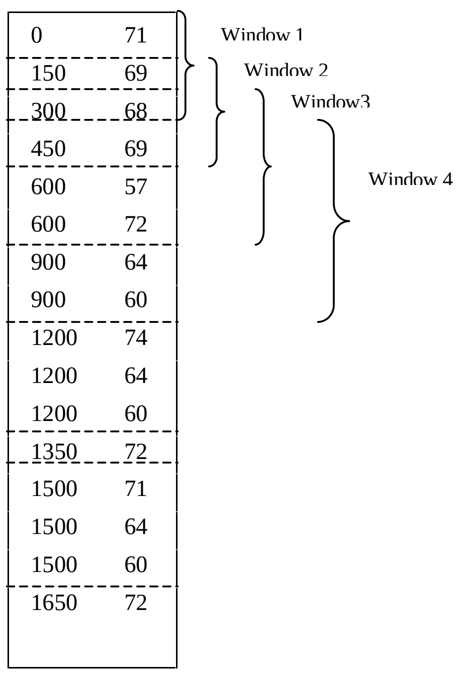
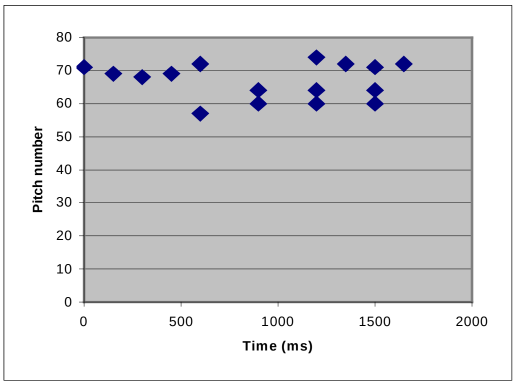


Figure 1. Excerpt from Mozart's Alla Turca



0	71
150	69
300	68
450	69
600	57
600	72
900	64
900	60
1200	74
1200	64
1200	60
1350	72
1500	71
1500	64
1500	60
1650	72

2(a)



2(b)

Figure 2. (a) Onset times and pitch events for Mozart's All Turca (b) performance visualized on a time-line

Following the steps outlined in obtaining the n-grams and applying Equation (1) in pattern derivation, the interval sequences from the first 3 windows of length-3 onset times of the performance data in Figure 2 are:

Window 1: [-2 -1]

Window 2: [-1 1]

Window 3: [1 –12] and [1 3]

To add to the information content of the n-grams constructed using interval sequences, the duration dimension of music information is used. Numerous studies have been carried out with the use of patterns generated from various combinations of the pitch and duration dimensions. These studies either used pitch information [8, 11], rhythm information [12] or both pitch and rhythm information simultaneously [4, 13]. In using the duration dimension for pattern derivation, a common mechanism is to use the relative duration of a note to a designated base duration such as the quarter or the sixteenth note. Relative durations are widely used as they are invariant to changes of tempo [10]. However, the choice of base durations such as the quarter or the sixteenth note could pose quantisation problems with performance data compared to data obtained from score encodings. With performance data, one option for the selection of a base duration could be the time difference between the first two notes of a given performance. However, with errors such as timing deviations of these two notes or recordings being slightly trimmed off at the beginning, this error would be duplicated in obtaining rhythmic information of the whole performance data.

In our approach, we look at the onset times pattern based on the timeline - the times at which pitch events occur. The approach in using time between consecutive note onsets has been studied by I. Shmulevich et. al. [14]. For pattern derivation using rhythm information, the ratios of time difference between adjacent pairs of onset times form a rhythmic ratio sequence. With this approach, it is not necessary to quantise on a predetermined base duration, to use the duration length of a note (which can be difficult to determine from audio performances) and we do not assume any knowledge of beat and measure information. For a sequence of *n* onset times, a rhythmic ratio sequence is derived with *n-2* ratios obtained by Equation (2).

$$Ratio_i = \left(\frac{Onset_{i+2} - Onset_{i+1}}{Onset_{i+1} - Onset_i}\right) \qquad (2)$$

In obtaining n-grams that incorporate interval and rhythmic ratio sequences using *n* onset times and pitches, the n-gram would be constructed in the pattern form of:

[Interval_1 Ratio_1 … Interval_{n-2} Ratio_{n-2} Interval_{n-1}]

Using the example of Figure 2, the combined interval and ratio sequences from the first 3 windows of length 3-onset are:

Window 1: [-2 1 -1]

Window 2: [-1 1 1]

Window 3: [1 1 -12] and [1 1 3]

Note that the first and last number of each tuple are intervals while the middle number is a ratio.

3.2 Pattern encoding

In order to be able to use text search engines we need to encode our n-gram patterns with text characters. One challenge that arises is to find an encoding mechanism that reflects the pattern we find in musical data. With large numbers of possible interval values and ratios to be encoded, and a limited number of possible text representations, classes of intervals and ratios that clearly represent a particular range of intervals and ratios without ambiguity had to be identified. For this, the frequency distribution for the directions and distances of pitch intervals and ratios of onset time differences that occur within the data set were obtained. A data collection of 3096 MIDI files of a classical music collection was used in obtaining these frequencies. These were mostly classical music performances obtained from the Internet. [http://www.classicalarchives.com]

For the pitch encoding, firstly the data set was analysed for the range and interval distances that occur within the data set and the frequency at which these occur. The frequency distribution versus interval (in units of semitones) graph obtained is shown in Figure 3.

According to Figure 3, the vast bulk of pitch changes occurs within one octave (i.e., -12 to +12 semitones). A good encoding should be more sensitive in this area than outside of it. We chose the code to be the integral part of a differentiable continuously changing mapping function (3), the derivative of which approximately matches the empirical distribution of intervals in Figure 3.

$$Code = \text{int}\left(X \tanh\left(\frac{Interval_{n-1}}{Y} \right) \right) \qquad (3)$$

In Equation (3), *X* is a constant set to 27 for our experiments as a mechanism to limit the codes range to the 26 text letters. *Y* is set to 24 to obtain a 1-1 mapping of semitone differences in the range [-13, 13]. In accordance with the empirical frequency distribution of Figure 3, less frequent semitone differences (which are bigger in size) are squashed and have to share codes. Based on the property of the tanh curve, Y determines the rate at which class sizes increase as interval sizes increase. This is a trade-off between classes of small (and frequent) versus large (and rare) intervals. The codes obtained are then mapped to the ASCII character values for letters. In encoding the interval direction, positive intervals are encoded as uppercase A-Z and negative differences are encoded with lower case a-z and in the centre code 0 being represented by the numeric character 0.

In using duration ratios, most studies have assumed *quantised rhythms, i.e., rhythm as notated in the score [14]* owing to simplicity and timing deviations that could occur with performance data. To deal with performance data, we adopt ratio bins for our study.

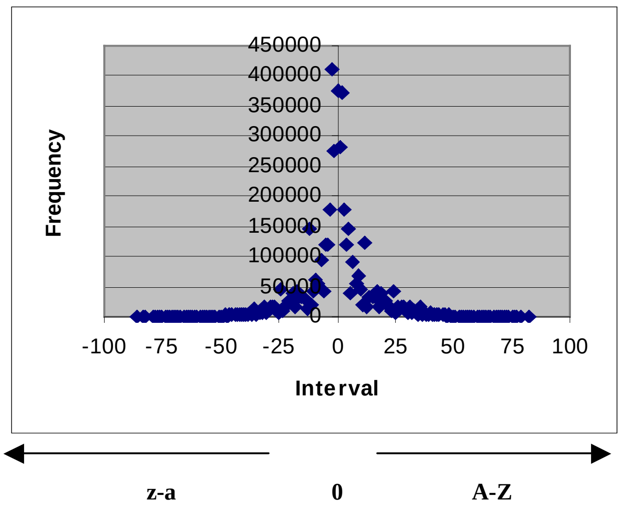


Figure 3. Interval Histogram

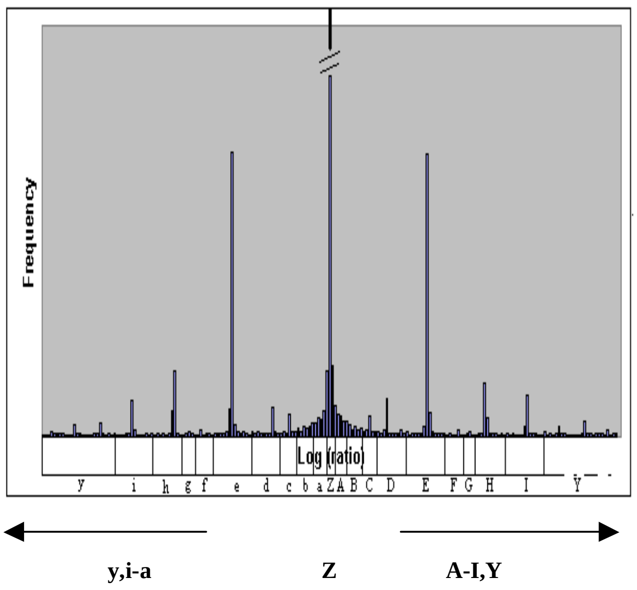


Figure 4. Ratio Histograms and Ratio Bins

Figure 4 shows the frequency versus the log of the ratios (onset times were obtained in units of milliseconds). We analysed the frequency distribution of ratio values of the data collection in order to provide quantisation ranges for the bins that reflect the data set. The peaks clearly discriminate ratios that are frequent and bins for ratio values for encoding can be established. Mid-points between these peak ratios were then used to construct bins which provided appropriate quantisation ranges in encoding the ratios. Ratio 1 has the highest peak as expected and other peaks occur in a symmetrical fashion where for every *peak ratio* identified, there is a symmetrical peak value of 1/*peak ratio*. From our data analysis, the peaks identified as ratios greater than 1 are 6/5, 5/4, 4/3, 3/2, 5/3, 2, 5/2, 3, 4 and 5.

The ratio 1 is encoded as Z. The bins for ratios above 1 as listed above are encoded with uppercase alphabets A-I and any ratio above 4.5 is encoded as Y. The various bins for ratios smaller than 1 as listed above are encoded with lowercase alphabets a-i and y respectively. The ranges identified with this symmetry and corresponding codes assigned are visualised in Figure 4.

4 IMPLEMENTATION

4.1 Database development

One of the main aims of this study is to examine the retrieval effectiveness of the musical words obtained from n-grams based on the pitch and duration information. The experimental factors investigated for this initial study were a) the size of interval classes and bin ranges for ratios, b) the query length and c) the window size used for the n-gram construction. We use the same data collection of 3096 classical MIDI performances for the database development as in Section 3.

6 databases P4, R4, PR3, PR4, PR4CA and PR4CB were developed. The minimum window size is 3, as at least 3 unique onset times would be required in obtaining one onset time difference ratio. A description of each database and its experimental factors follows:

P4: Only the pitch dimension is used for the n-gram construction with the window size of 4 onset times. Each n-gram is encoded as a string of 3 characters corresponding to 3 intervals. *Y* is set to 24 to enable a 1-1 mapping of codes to most of the intervals within a distance of 20. The theoretical maximum of possible index terms is 148,877 = $(26*2+1)^3$.

R4: Only the rhythm dimension is used for the n-gram construction with the window size of 4 onset times. All bin ranges identified as significant ratio ranges were used in encoding. The theoretical maximum of possible index terms is 441 = $(10*2+1)^3$.

PR3: The pitch and rhythm dimensions are used for the n-gram construction in the combined pattern form stated in Section 3 with the window size of 3 onset times. *Y* is assigned 24 to enable similar interval class encoding as P4. All bin ranges identified as significant ratio ranges are used in encoding. The theoretical maximum of possible index terms is 58,989 = (53*21*53).

PR4: The pitch and rhythm dimensions are used for the n-gram construction as above but with the window size of 4 onset times. All bin ranges identified as significant ratio ranges are used in encoding. The theoretical maximum of possible index terms is 65,654,757 = (53^3*21^2).

PR4CA: The pitch and rhythm dimensions are used for the n-gram construction as above. To study the effects of the interval class sizes within the range of 2 octaves for a 2-1 mapping for most intervals for most intervals smaller than 20 semitones, *Y* is set to 48. Although one character now covers at least 2 semitones (as opposed to 1 semitone above), still all alphabets are used with this encoding, i.e. 26 uppercase and 26 lowercase letters and 0 for no change. The encoding for the ratios was made coarser as well : where we previously used the codes A-I,Y and a-I,y we now use the codes A-D, Y and a-d,y respectively, now A covers what used to be represented by A and B, B covers what used to be C and D , C covers what used to be E and F etc. The theoretical maximum of possible index terms is 18,014,177 = (53^3*11^2).

PR4CB: The pitch and rhythm dimensions are used for the n-gram construction as above. To study the effects of the interval class sizes within the range of 2 octaves for a 3-1 mapping for most intervals up to around 20 semitones, *Y* is set to 72. Coarse ratio encoding with bins used as in PR4CA.

The summary of databases and experimental factors are shown in Table 1.

Table 1. Databases and experimental factors

Database	Pitch	Rhythm	n	Y	# R.Bins	#Terms
P4	Y		4	24		148,877
R4		Y	4		21	441
PR3	Y	Y	3	24	21	58,989
PR4	Y	Y	4	24	21	65,654,757
PR4CA	Y	Y	4	48	11	18,014,117
PR4CB	Y	Y	4	72	11	18,014,117

4.2 Retrieval Experiments

In examining the retrieval effectiveness of the various formats of musical words and to evaluate the various experimental factors, an initial run, R1, was performed on the 6 databases. For query simulation, polyphonic excerpts are extracted from randomly selected musical documents of the data collection. Query locations were set to be the beginning of the file. In simulating a variety of query lengths, lengths of the excerpts extracted from the randomly selected files were of 10, 30 and 50 onset times. These excerpts were then pre-processed and encoded to generate musical words with similar formats to the corresponding 6 databases: P4, R4, PR3, PR4, PR4CA and PR4CB. The ranked retrieval method was used for run R1 averaged over 30 queries. In ranking the documents retrieved, the cosine rule used by the MG system was adopted [15] and in evaluating our retrieval using the known item search of our query excerpt, the Mean Reciprocal Rank (MRR) measure was used. The reciprocal rank is equal to 1/r where r is the rank of the music piece the query was taken from. In using the known item search, the rank position of the document that the query was extracted from was used in obtaining the reciprocal rank measure. These were averaged over the 30 queries. This MRR measure is between 0 and 1 where 1 indicates perfect retrieval. The retrieval results are shown in Table 2.

Table 2. MRR measures for run R1

	10	**30**	**50**
P4	0.60	0.77	0.81
R4	0.03	0.11	0.15
PR3	0.46	0.74	0.81
PR4	0.74	0.90	0.95
PR4CA	0.71	0.83	0.71
PR4CB	0.47	0.68	0.73

The results clearly indicate that using n-grams with polyphonic music retrieval is a promising approach with the best retrieval measure 0.95 being obtained by musical words of the PR4 format and a query length of 50 onset times. Comparing the retrieval measures of P4 and PR4 for all 3 query lengths, it can be said that the addition of rhythm information to the n-gram is a definite improvement to widening the scope of n-gram usage in music information retrieval.

The length of a window for n-gram construction would require further study, as there are clear improvements of measures between PR3 and PR4 for all query lengths. Further experiments will be needed to obtain the optimal length. In looking at the class size of the intervals and bin range of ratios, measures clearly deteriorate from smaller class sizes of PR4 to larger sizes of PR4CA and PR4CB. The class sizes require further investigation to determine its usefulness in providing allowances for more fault-tolerant retrieval.

In general, and as expected, the measure improves with the length of the query for all databases although retrieval using only ratio information with R4 is almost insignificant. Clearly, the 441 possible different index terms are insufficient to discriminate music pieces.

4.3 Error Simulation

A second run, R2, was performed by simulating errors in the queries to study the retrieval behaviour under error conditions. Error models used in monophonic music described in [3, 8] were not adopted for this study as the range of intervals was significantly different. As there were no error models available with polyphonic music, we adopted the Gaussian error model for intervals as shown in Equation (4) and for ratios as shown in Equation (5). ε is the Gaussian standard random variable and D_i is the mean deviation for an interval error and D_r is the mean deviation for an error in the ratio.

$$NewInterval_k = Interval_k + (D_i * \varepsilon) \qquad (4)$$

$$NewRatio_k = Ratio_k * \exp.(D_r * \varepsilon) \qquad (5)$$

As an initial attempt to investigate retrieval with error conditions, we arbitrarily selected two sets of error deviation values D1 and D2. With D1, D_i was assigned 3 and D_r assigned as 0.3. For the second set of mean error deviation values, D_i was assigned 2 and D_r was retained as 0.3. D_r was left unchanged, as the ratio bin range was not varied between PR4CA and PR4CB. All musical words generated for the similar queries used in R1 and with length 30 were modified by incorporating the error deviation for the pitch and duration dimensions correspondingly for the 3 databases PR4, PR4CA and PR4CB. The MRR measures are shown in Table 3.

Table 3. MRR measures for run R2

	D1	**D2**
PR4	0.24	0.50
PR4CA	0.30	0.65
PR4CB	0.27	0.50

The results clearly indicate that musical words encoded with a wider interval class size perform better with error conditions. A compromise between musical words encoded using larger interval class sizes and wider ratio bin ranges and smaller ones is clearly required. This can be seen from the improvement in measures obtained with run R2 and deviation set D2 of Table 3 where the measure of PR4CA is 0.65 and PR4 only 0.50. For the counterpart run, R1, with no query errors, it indicates deterioration in measure with the wider encoding (where a measure of 0.90 was obtained with PR4 and only 0.83 for PR4CA with query length 30).

This initial experiment under error conditions clearly identifies the need for a detailed analysis in obtaining optimal values for interval class size and effective retrieval in using n-grams in polyphonic music retrieval.

5 FUTURE WORK

Based on the experimental results and initial experimental factors investigated, this study will be continued with an in-depth study of the following experimental factors: a) query length b) window length c) ration bin range d) Y value for interval classification e) error model. Further issues for investigation are a) the development of error models with polyphonic music, b) a relevance judgment investigation in assessing the documents and finer retrieval measures, c) suitability of the ranking mechanism for musical words, d) an analysis of the search complexity of the algorithm in extracting all possible patterns

6 CONCLUSIONS

This study has proven the usefulness of using n-grams in polyphonic music data retrieval. An interval mapping function was utilised and proved useful in mapping interval classes over the text alphabetical codes. Onset time ratios have proven useful for incorporating rhythm information. With the use of bins for ranges of significant ratios, the rhythm quantisation problem in music performance data has been overcome. The results presented so far for polyphonic retrieval are qualitatively comparable to published successful monophonic retrieval experiments [8] and, hence, very promising.

7 ACKNOWLEDGEMENTS

This work is partially supported by the EPSRC, UK.

8 REFERENCES

[1] David Huron, *Perceptual and Cognitive Applications in Music Information Retrieval*, International Symposium on Music Information Retrieval, Music IR 2000, Oct 23rd - 25th, 2000, Plymouth, Massachussetts.

[2] Andreas Kornstadt, *Themefinder: A Web-Based Melodic Search Too*l, Computing in Musicology 11, 1998, MIT Press

[3] Rodger J. MacNab, Lloyd A.Smith, David Bainbridge and Ian H. Witten, *The New Zealand Digital Library MELody inDEX*, D-Lib Magazine, May 1997

[4] M. Clausen, R. Engelbrecht, D. Meyer, J. Schmitz, *PROMS: A Web-based Tool for Searching Polyphonic Music*, International Symposium on Music Information Retrieval, Music IR 2000, Oct 23rd - 25th, 2000, Plymouth, Massachussetts.

[5] David Huron, *Humdrum and Kern: Selective Feature Encoding*, Beyond MIDI: The Handbook of Musical Codes, pp 375-40.

[6] Eleanor Selfridge-Field, *Conceptual and Representational Issues In Melodic Comparison*, Computing in Musicology 11, 1998, pp 1-64

[7] Massimo Melucci and Nicola Orio, *Music Information Retrieval using Melodic Surface*, The Fourth ACM Conference on Digital Libraries '99, Berkeley, USA,pp 152-160

[8] Stephen Downie and Michael Nelson, *Evaluation of A Simple and Effective Music Information Retrieval Method*, SIGIR 2000, Athens, Greece, pp 73-80

[9] Tim Crawford, Costas S. Iliopoulus and Rajeev Raman, *String-Matching Techniques for Musical Similarity and Melodic Recognition*, Computing in Musicology 11, 1998, MIT Press, pp 73-100

[10] Kjell Lemström, Atso Haapaniemi, Esko Ukkonen, *Retrieving Music - To Index or not to Index*, ACM Multimedia '98, -Art Demos, Technical Demos - Poster Papers, September 1998, Bristol, UK

[11] Steven Blackburn and David DeRoure, *A Tool for Content-Based Navigation of Music*, ACM Multimedia '98, Bristol, UK, pp 361 – 368

[12] Chen, J.C.C. and A.L.P. Chen, *Query by Rhythm: An Approach for Song Retrieval in Music Databases*, In proc. Of IEEE Intl. Workshop on Research issues in Data Engineering, pp 139-146, 1998

[13] Shyamala Doraisamy, *Locating Recurring Themes in Musical Sequences*, M. Info. Tech. Thesis, 1995, University Malaysia Sarawak.

[14] I. Shmulevich, O. Yli-Harja, E. Coyle, D.-J. Povel, and K. Lemström, *Perceptual Issues in Music Pattern Recognition – Complexity of Rhythm and Key Finding*, In Proceedings of the AISB '99 Symposium on Musical Creativity, pages 64-69, Edinburgh, 1999

[15] Ian H. Witten, Alistair Moffat and Timothy C. Bell, *Managing Gigabytes: Compressing and Indexing Documents and Images*, 2nd edition, 1999, Morgan Kaufmann Publishers

ISMIR2001 Invited Address

Adventures in Standardization, or, how we learned to stop worrying and love MPEG-7

Computing Department
Lancaster University
Lancaster, LA1 4YR, UK
+44 1524 594 537
atl@comp.lancs.ac.uk

Youngmoo Kim
MIT Media Lab,
E15-401, 15 Ames St.
Cambridge, MA, USA 02138
+1-617-253-0619
moo@media.mit.edu

ABSTRACT

The authors give a brief account of their combined 7+ years in multimedia standardization, namely in the MPEG arena. They discuss specifics on musical content description in MPEG-7 Audio and other items relevant to Music Information Retrieval among the MPEG-7 Multimedia Description Schemes. In the presentation, they will give a historical overview of the MPEG-7 standard, its motivations, and what led to its current state.

1. INTRODUCTION

MPEG-7, officially known as the Multimedia Content Description Interface, is an ISO/IEC standard whose first version will be finalized at the end of 2001. Its goal is to provide a unified interface for describing multimedia content in all forms. Although an obvious application for this is in multimedia information retrieval, it by no means limits itself to that domain, also encompassing broadcast-style scenarios, real-time monitoring, and potentially semi-automated editing. The Audio part of the standard includes descriptors for musical timbre and for melodic similarity.

2. MPEG-7 AUDIO

2.1 Melody in MPEG-7

Of chief interest to the authors in MPEG-7 is the Melody description scheme. The MPEG-7 Audio standard will include a unified *Description Scheme* (DS) for melodic information, which contains two variants at different levels of detail. Within the Description Scheme, there are a series of features common, but auxiliary, to either variant. These features include meter, key, and scale used, and their use is optional. Of the two options for representing the melody itself, the first, called MelodyContour, was designed to facilitate the type of imprecise musical matching required by the query-by-humming application with as little overhead as possible. The second representation, MelodySequence, is considerably more verbose as a precise description of melody to facilitate search and retrieval using a wide variety of queries [1].

Defining exactly what is or is not a melody can be somewhat arbitrary. Melodies can be monophonic, homophonic, or contrapuntal. Sometimes what one person perceives to be the melody is not what another perceives. A melody can be pitched or purely rhythmic, such as a percussion riff. The MPEG-7 Melody DS does not attempt to address all of these cases and is limited in scope to pitched, monophonic melodies.

Importantly for recognition purposes, people can still uniquely identify melodies after they have undergone transposition (we still recognize a familiar tune in a different key as being the same tune). For this reason, absolute pitch is not the best descriptor for melodic pitch information. What is important are the relative intervals between successive notes in a melody, since interval relations are also invariant to key transposition. This is a key fact exploited by both MelodyContour and MelodySequence.

With MelodyContour, however, we do not assume that a query will contain precise and accurate intervals. A more robust feature is the melody *contour*, which is derived from interval information and is also invariant to transposition [4]. Based on research and experimental evidence, a five-level contour was chosen for MelodyContour in MPEG-7, dividing the contour into small and large ascending and descending intervals with one level indicating no pitch change. Separation of the five contour levels is defined as in table 1.

Table 1: The five levels of contour information in MelodyContour

Contour value	*Change in interval*
-2	Descent of a minor-third or greater
-1	Descent of a half-step or whole-step
0	No change
1	Ascent of a half-step or whole-step
2	Ascent of a minor-third or greater

In contrast, the MelodySequence makes no *a priori* assumptions about the errors made in the query, and does not try to eliminate them through quantization. Rather, it takes the approach that the data set itself provides the best recovery for user error [5]. For a query on a melody with *n* notes, the representation transforms the query into *n-1* dimensional *interval space*, to enable a comparison between two melodies using an L2 norm. This approach has the advantages of not eliminating any pitch or rhythm information, and therefore is able to reconstruct melodies and queries after the fact. MelodyContour and MelodySequence are compared in Tables 2 and 3.

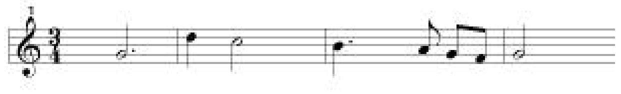


Figure 1: The first measures of "Moon River"

Table 2: MelodyContour code for "Moon River"

```
  <Contour>
  <!–- MelodyContour description: "Moon River"-->
  <!-- (7 intervals = 8 notes total) -->
    <ContourData>2 –1 –1 –1 –1 –1 1</ContourData>
  </Contour>
  <Meter>
    <Numerator>3</Numerator>
    <Denominator>4</Denominator>
  </Meter>
  <Beat>
    <BeatData>1 4 5 7 8 9 9 10</BeatData>
  </Beat>
```

Table 3: MelodySequence code for "Moon River"

```
<MelodySequence>
  <!-- [+7 -2 -1 -2 -2 -2 +2]               -->
  <!-- [2.3219 -1.5850 1 -0.4150 -1.5650 0 0]-->
    <Note>
      <Interval>7</Interval>
      <NoteRelDuration>2.3219</NoteRelDuration>
      <Lyric>Moon</Lyric>
      <PhoneNGram>m u: n</PhoneNGram>
    </Note>
    <Note>
      <Interval>-2</Interval>
      <NoteRelDuration>-1.5850</NoteRelDuration>
      <Lyric>Ri-</Lyric>
    </Note>
    <Note>
      <Interval>-1</Interval>
      <NoteRelDuration>1</NoteRelDuration>
      <Lyric>ver</Lyric>
    </Note>
    <!-- Other notes elided                  -->
</MelodySequence>
```

2.2 Other features in MPEG-7 Audio

The other Description Scheme relevant to music is the musical instrument timbre DS. This description scheme groups up to five different features to estimate the perceptual similarity between segmented musical tones. There are two different "timbre spaces" possible, with harmonic, sustained, coherent sounds, and with non-sustained, percussive sounds [6]. The different spaces use the following features as shown in table 4.

Table 4: Timbre features

Harmonic	*Percussive*
Harmonic Spectral Centroid	Log Attack Time
Harmonic Spectral Deviation	Temporal Centroid
Harmonic Spectral Spread	Spectral Centroid
Harmonic Spectral Variation	
Log Attack Time	

The application-oriented Description Schemes within MPEG-7 audio also include a representation of spoken content (e.g. as an output of a speech recognition engine) [3], robust audio identification, and generalized sound recognition tools that use spectral basis functions [2].

To complement the application-oriented Description Schemes, there are general audio features that may apply to any signal. These low-level audio descriptors form a basic compatibility framework so that different applications have a baseline agreement on such aspects as standard definitions of features, standard sampling rates for regularly sampled descriptors, or how to segment sounds hierarchically [7].

3. GENERAL MPEG-7 FEATURES

Typical descriptors for traditional information retrieval, such as title, composer, and year of recording, are covered in detail in the Multimedia Description Schemes (MDS) part of the standard, particularly in the section on creation and production information. Similarly, one may try to describe musical genre as a hierarchical ontology, or describe musical instrument from a list of controlled terms. The mechanisms by which one can create these ontologies and dictionaries are in the MDS as the Controlled Term datatype and the Classification Scheme DS. One may also describe aspects of the medium itself, such as the encoding format or sound quality, with the media description tools in the MDS [7].

Other important technologies in MPEG-7, such as the Description Definition Language (DDL)—the XML Schema-based language for giving the syntax of Description Schemes—and various systems technologies for compressing MPEG-7 data, are sadly out of the scope of this paper.

4. ACKNOWLEDGMENTS

The authors gratefully acknowledge the work of the hundreds of people actively developing MPEG-7 (especially the dozen or so cool people in the Audio subgroup), and the organizers of ISMIR2001 for giving us the chance to present a small cross-section of that work.

5. REFERENCES

[1] *Information Technology—Multimedia Content Description Interface—Part 4: Audio*, ISO/IEC FDIS 15938-4, 2001.

[2] Casey, M., "MPEG-7 Sound-Recognition Tools," IEEE Trans. on Circuits and Systems for Video Technology, Vol. 11, No. 6 (June 2001), pp. 737–747.

[3] Charlesworth, J.P.A., and P.N. Garner, "Spoken Content Representation in MPEG-7," IEEE Trans. on Circuits and Systems for Video Technology, Vol. 11, No. 6 (June 2001), pp. 730–736.

[4] Ghias, A., J. Logan, D. Chamberlin, and B. C. Smith. "Query by Humming: musical information retrieval in an audio database." *Proc. ACM Multimedia*, San Francisco, 1995.

[5] Lindsay, A.T. , "Using Contour as a Mid-level Represenation of Melody," September 1996, unpublished S.M. Thesis at MIT Media Laboratory. Also at http://sound.media.mit.edu/~alindsay/thesis/

[6] Peeters, G., S. McAdams, and P. Herrera, "Instrument sound description in the context of MPEG-7," *Proc. ICMC 2000*, International Computer Music Conference, Berlin, 2000.

[7] Quackenbush, S., and A. Lindsay. "Overview of MPEG-7 Audio," IEEE Trans. on Circuits and Systems for Video Technology, Vol. 11, No. 6 (June 2001), pp. 725–729.

Content-based Identification of Audio Material Using MPEG-7 Low Level Description

Eric Allamanche
Fraunhofer IIS-A
Am Weichselgarten 3
D - 91058 Erlangen, Germany
+49 9131 776 322
alm@iis.fhg.de

Jürgen Herre
Fraunhofer IIS-A
Am Weichselgarten 3
D - 91058 Erlangen, Germany
+49 9131 776 353
hrr@iis.fhg.de

Oliver Hellmuth
Fraunhofer IIS-A
Am Weichselgarten 3
D - 91058 Erlangen, Germany
+49 9131 776 372
hel@iis.fhg.de

Bernhard Fröba
Fraunhofer IIS-A
Am Weichselgarten 3
D - 91058 Erlangen, Germany
+49 9131 776 535
bdf@iis.fhg.de

Thorsten Kastner
Fraunhofer IIS-A
Am Weichselgarten 3
D - 91058 Erlangen, Germany
+49 9131 776 348
ksr@iis.fhg.de

Markus Cremer
Fraunhofer IIS-A / AEMT
Am Ehrenberg 8
D - 98693 Ilmenau, Germany
+49 3677 69 4344
cre@emt.iis.fhg.de

ABSTRACT

Along with investigating similarity metrics between audio material, the topic of robust matching of pairs of audio content has gained wide interest recently. In particular, if this matching process is carried out using a compact representation of the audio content ("audio fingerprint"), it is possible to identify unknown audio material by means of matching it to a database with the fingerprints of registered works. This paper presents a system for reliable, fast and robust identification of audio material which can be run on the resources provided by today's standard computing platforms. The system is based on a general pattern recognition paradigm and exploits low level signal features standardized within the MPEG-7 framework, thus enabling interoperability on a world-wide scale.

Compared to similar systems, particular attention is given to issues of robustness with respect to common signal distortions, i.e. recognition performance for processed/modified audio signals. The system's current performance figures are benchmarked for a range of real-world signal distortions, including low bitrate coding and transmission over an acoustic channel. A number of interesting applications are discussed.

1. INTRODUCTION

Stimulated by the ever-growing availability of musical material to the user via new media and ways of distribution (e.g. the Internet, efficient audio compression schemes) an increasing need to identify and classify audio data has emerged. Given the enormous amount of available audio material it has become more and more difficult for the consumer to locate music that fits his or her personal tastes.

Descriptive information about audio data which is delivered together with the actual content would be one way to facilitate this search immensely. This so-called metadata ("data about data") could e.g. describe the performing artist, composer or title of the song and album, producer, date of release, etc.. Examples of de-facto and formal standards for metadata are the widely used ID3 tags attached to MP3 bitstreams [1] and the forthcoming MPEG-7 standard [2].

Another way of retrieving these information resides in the characteristics of the medium on which the audio data is comprised. This kind of services are provided by e.g. *Gracenote,* formerly *CDDB*, [3] where the *Table Of Content* (TOC) of an audio CD is compared against a vast database. Obviously, this kind of mechanism fails when the CD is a self made compilation, or when commercially not available.

A lot of different approaches have addressed the automatic analysis of audio content, be it speech/music classification [4, 5, 6], retrieval of similar sounds ("sounds like" data base search) [7, 8, 9], or music genre classification [10].

The main topic of this paper, however, is to present a system which performs an automated identification of audio signals rather than assigning them to predefined categories. The essential property of the introduced system lies in the fact that it does not rely on the availability of metadata information that is attached to the audio signal itself. It will, however, identify all incoming audio signals by means of a database of works that are known to the system. This functionality can be considered the algorithmic equivalent of human recognition of a song from the memory of the recognizing person.

This observation yields the key criteria for the performance requirements of an audio identification system. It should be able to identify the song as long as a human being is able to do so. To come as close as possible to this aim, the system should be robust against alteration commonly applied to musical material, like filtering, dynamic range processing, audio coding, and so on. Additionally, arbitrary excerpts of the music signal should be sufficient for the recognition.

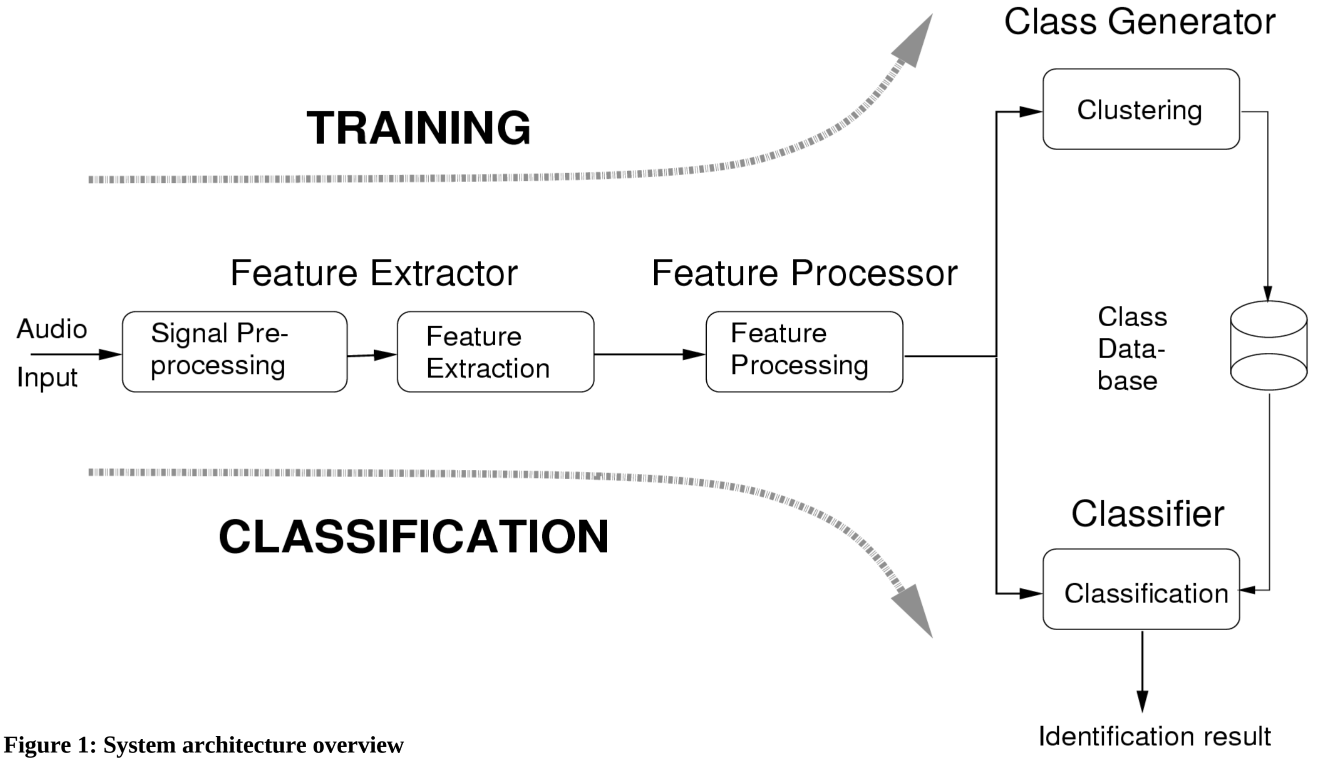


Figure 1: System architecture overview

The segment size needed for recognition by an ideal system should not be longer than a few seconds, with other words as long as it would take a human listener to identify a piece of music correctly.

On top of that the system should be able to operate with large databases of registered works while reliably discriminate between the items, and computational complexity should stay within acceptable limits ("System Scalability").

While the task of music recognition may appear easy to human listeners, lately introduced technologies definitely fall short of reaching these high goals, e.g. in terms of robustness of recognition [11] or computational complexity [12].

The system presented in this paper has been designed to meet many of the requirements mentioned above. The system's complexity is low enough to allow operation on today's personal computers and other cost-effective computing platforms and the described algorithm is based on well-known feature extraction/pattern recognition concepts [13]. It includes extraction of a set of robust features with a psychoacoustic background. The extraction process itself is based on so called *Low Level Descriptors* that will be part of the upcoming MPEG-7 standard.

In the following chapters an overview of the presented system is provided first. The architecture of the system as well as the basic underlying concepts are explained. Subsequently, the system requirements for robust recognition are discussed by identifying a suite of typical alterations of the original audio material. The influence of the audio feature selection on the recognition performance is addressed thereafter based on test results using different sets of test data. In the following two chapters potential applications of the proposed system are identified and the compliance to the upcoming MPEG-7 standard is accounted for. Finally, a conclusion section will present promising future improvements and directions for further enhancement of the overall system performance.

2. SYSTEM OVERVIEW

The audio identification system presented here follows a general pattern recognition paradigm as described in [13]. From the block diagram shown in Figure 1, two distinct operating modes can be identified, namely the *training* mode and the *classification* (recognition) mode. During training a condensed "fingerprint" of each item from the training sample is created which is used in the recognition phase to identify the item under test. In a preprocessing step a signal preprocessor converts the audio input signal into a fixed target format with predefined settings. In the present configuration, the signal is converted to a mono signal using common downmix techniques and then, if necessary, resampled to a sampling frequency of 44.1 kHz.

2.1 Feature Extraction

Feature extraction is a central processing step which has a high influence on the overall system performance. The chosen feature set should be robust under a wide class of distortions (see Section 3.2) and the computational burden should be low enough to allow for real-time calculation. In the present configuration the audio time signal is segmented by a windowing function and each window is mapped to a spectral representation by means of a DFT (Discrete Fourier Transform). A set of psychoacoustic features is extracted from the spectrum of each analysis window to form a feature vector. This vector is regarded as an elementary feature at a discrete time instant *t* and undergoes further processing.

The elementary features are then normalized to have component-wise unit variance. Note that no removal of the mean is necessary prior to normalization, as suggested in [14], since only the difference between the feature vectors to be classified and the reference vectors from the "fingerprint" are considered. Through this normalization step, a balanced feature vector is generated which can be filtered optionally.

Normalized features from subsequent time steps are then grouped together to form a composite feature vector of higher dimension. In addition, the feature statistics of the single vectors are estimated.

2.2 Vector Quantization for Pattern Recognition

The identification system uses a linear classifier which is based on a compact representation of the training vectors, the above mentioned fingerprint. The classification is performed using a standard *NN* (Nearest Neighbor) rule. To obtain a compact class representation a *VQ* (Vector Quantization) algorithm is applied for training. This method approximates the training data for each class with a so-called vector codebook by minimizing a *RMSE* (Root Mean Square Error) criterion. The codebook consists of a certain number of code vectors depending on the maximum permitted RMSE. An upper limit of the number of code vectors may be specified. The VQ clustering algorithm is an iterative algorithm which approximates a set of vectors by a much lower number of representative code vectors, forming a codebook. Such a codebook is needed for each class (audio item). In Figure 2 an example of the representation of a set of 2-D feature vectors by 6 code vectors is shown.

The code vectors are obtained using a simple *k-means* clustering rule. The code vectors computed during training phase are stored in a database together with other associated descriptive information of the music items, such as title and composer of the item.

In Figure 3 the approximation error of a feature vector set is shown, depending on the number of code vectors used for the codebook. The training set can be approximated ideally if the number of code vectors reaches the number of training vectors. For distorted versions of the training vectors, on the other hand, the approximation error does not converge toward zero.

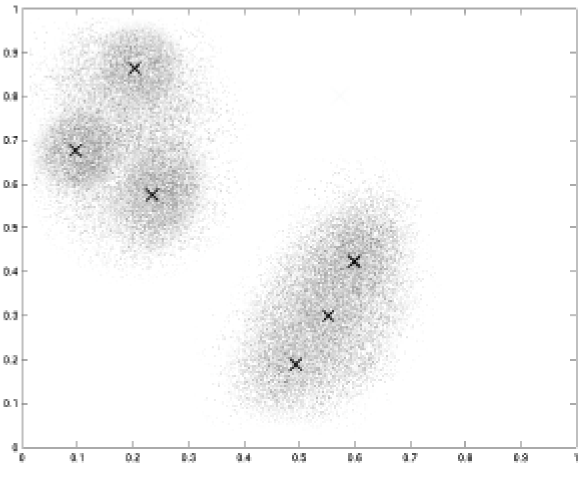


Figure 2. Example of 2-D feature set and it's approximation using 6 code vectors.

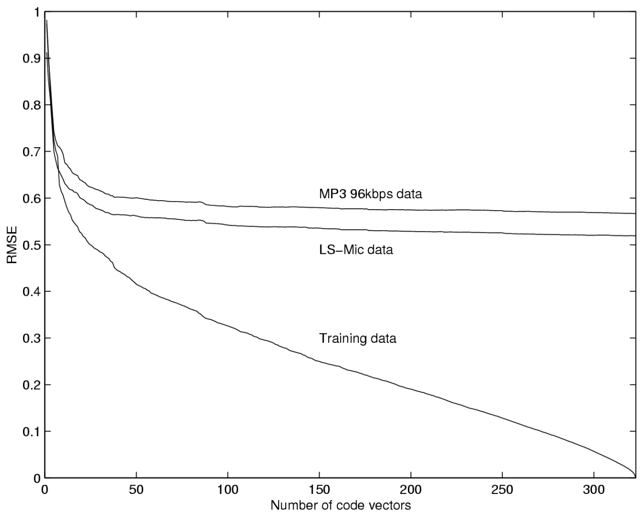


Figure 3. RMS error as a function of the number of code vectors.

2.3 Classification

The music identification task here is an *N-class* classification problem. For each of the music items in the database one class, i.e. the associated codebook, is generated. To identify an unknown music item which is included in the reference database, a sequence of feature vectors is generated from the unknown item and these features are compared to the codebooks stored in the database.

In more detail, during the identification process each vector is subsequently approximated by all stored codebooks using some standard distance metric. For each of the classes the approximation errors are accumulated and, as a result, the music item is assigned to the class which yields the smallest accumulated approximation error.

In a more recent version of the system, the statistics of the features is used for the classification task instead of the features themselves. The extracted features are collected over a certain period of time and short-time statistics are calculated. Furthermore, the temporal dependencies between the features are taken into account. This results in both higher recognition performance and lower processing time.

3. SYSTEM REQUIREMENTS

3.1 Robustness Requirements

For a human listener, just a few seconds, even in noisy environments, may be sufficient to identify a song. In order to design a prototype system which approximates this behavior, special attention has to be paid to the alterations an audio signal can be subjected to and to measure the impact of these degradations on the recognition performance. It is therefore of great importance for an audio identification system to handle "real world" audio signals and distortions. Some of these types of distortions are discussed subsequently, forming the basis of the development process of a robust identification system.

A basic type of signal "degradation" which exists in real world are time shifted signals. If a feature turns out to be very sensitive towards this kind of signal modification, it is likely that this feature will also yield a poor recognition performance when faced with "real world" signals.

Another essential aspect is the sensitivity of the identification system against level changes. This is particularly important when the level of the input signal is unknown, or even worse, may slowly vary over time. Such situations arise when, for example, a song is recorded via a microphone. When considering this kind of distortion, the selected features should be invariant to scaling. This is, for instance, the case for energy envelopes and loudness. However, appropriate post processing of such features can avoid this dependency. A simple example could be the calculation of the difference of two consecutive feature vectors (these are the so-called *delta features*). Other transforms may be applicable as well to overcome this deficiency.

The following list enumerates a selection of signal distortions which were used during the development process of the identifications system to form a test suite of typical "reference distortions", each representing a different aspect of robustness.

- Time shift: Tests the system’s robustness against arbitrary time shifts of the input signal. This can be performed very easily by accessing the original audio signal randomly. Care should be taken that the entry points do not correspond to a block boundary used during training.
- Cropping: It is desirable that an audio identification system may be able to identify a small excerpt from a musical item with sufficient accuracy. In this way, identification of an entire song would be possible when only parts (such as the introduction or chorus) are used for recognition. As a consequence, the duration of a song to be entered in the base class database cannot be used as a feature.
- Volume change: By scaling the input signal by a constant or slightly time varying factor, the signal amplitude (volume) may be varied within a reasonable range. In order to counter level dependency, all features/post processing chosen for the identification system were designed to be level independent. Thus, no separate test results will be listed for this type of robustness test.
- Perceptual audio coding: An ever-increasing amount of audio is available in various compressed audio formats (e.g. MP3). It is therefore important for an identification system to maintain high recognition performance when faced with this kind of signals. The bitrate should be selected within a reasonable range, so that the degradation of subjective audio quality is not excessive. A bitrate of 96kbps for an MPEG-1/2 Layer-3 coded stereo signal is considered to be appropriate for general testing.
- Equalization: Linear distortion may e.g. result from applying equalization which is widely used to adapt the frequency characteristics to the users personal taste. For robustness testing of the audio identification system, octave band equalization has been used with adjacent band attenuations set to -6dB and +6dB in an alternating fashion.
- Bandlimiting: Bandlimited signals occur when the signal was represented at a low sample rate or, simply, if a low pass filter has been applied. This can be regarded as a special case of equalization.
- Dynamic range compression: Dynamic range compression is frequently used in broadcast stations. In order to identify audio signals from these stations, robustness against this *time-variant* type of processing must be considered.
- Noise addition: *White noise* or *pink noise* with a reasonable SNR (like e.g. 20-25 dB) was added to the item with a constant level in order to simulate effects such as analog background noise.
- Loudspeaker-microphone transmission (Ls-Mic): This kind of distortion appears when a musical item is played back over a loudspeaker and the emitted sound is recorded via a microphone. The resulting analog signal is then digitized by means of an A/D converter and presented to the input of the system. Such a setup provides a realistic combination of both severe linear and non-linear distortions and has been found to be one of the most challenging types of distortions with respect to automatic audio recognition. A system exhibiting robustness with respect to such a scenario is perfectly suitable for a wide range of applications. The test setup used in the presented work consists of a pair of small multimedia PC speakers and a standard PC microphone, which is directed towards the speakers at a distance of around 10cm.

While this list is by far not exhaustive, it should be sufficient for a general assessment of a system’s robustness qualities. In particular, the robustness of each feature with respect to these distortion types can be quantified effectively by such a test suite and then taken into account for the final feature selection process.

3.2 Computational Requirements

When investigating the necessary computational resources of all the software components involved in the identification process, it becomes apparent that that there exists a clear asymmetry between the feature extractor and the classifier in terms of processing power and memory space (both RAM and disk space). More precisely, the extraction process ("fingerprint generation") can be performed several times faster than real-time, since it only consists of a signal analysis followed by a feature calculation. This processing step is independent from the used classification scheme and from the database size, and thus only requires a small amount of CPU processing power and RAM storage.

In contrast, the required resources for the classification task are directly related to the underlying matching algorithm, the size of the database (i.e. the number of trained reference items) and the size and type of the fingerprint information.

While there is a trade-off between the degree of tolerable distortions, the fingerprint size and the computational complexity of the matching algorithm, it was the goal of the work described in this paper to find efficient configurations which would allow for both reliable recognition of real-world audio signals and real-time operation on today’s standard PC computing platforms.

4. RECOGNITION PERFORMANCE

This section discusses the recognition performance achieved by the prototype system depending on the choice of features. More specifically, the performance of the system is investigated when faced with distorted audio signals like the ones listed in the previous section. Figures are provided for different configurations of the system, including three features and different sizes of the test database.

4.1 Features

A decisive factor in the performance of the identification system is the selection of features. An extensive review of potentially interesting features led to the selection of the following candidate features which have been used for further experimentation.

- An important part in the perception of sound is represented by the so-called Loudness. Loudness belongs to the category of intensity sensations [15]. It seems intuitive that this basic aspect of an audio signal could serve as a robust feature for audio identification. Simple computational models of loudness are known, including both the calculation of the signal's total loudness and partial loudness in different frequency bands. This provides plenty of flexibility for defining a loudness-based feature set. For the following investigations a multi-band loudness feature was used.
- Besides the loudness sensation, another important characteristics of the audio signal relates to the distinction between more tone-like and more noise-like signal quality. The so-called SFM (Spectral Flatness Measure) [16] is a function which is related to the tonality aspect of the audio signal and can therefore be used as a discriminating criterion between different audio signals. Similar to loudness, the SFM can be used to describe the signal in different frequency bands. Such a multi-band version of the SFM features was used for the following evaluations.
- Similar to SFM, a so-called SCF ("Spectral Crest Factor") feature was investigated which is related to the tonality aspect of the audio signal as well. Instead of calculating the mean value for the numerator the maximum is computed, i.e. the ratio between the maximum spectral power within a frequency band and its mean power is determined. In the same way as for SFM, a multi-band version is used.

The next sections present classification results based on different setups. Each setup consists of a data base holding an increasing number of music items.

For each setup, a few tables are provided which reflect the recognition performance of the identification system. The performance is characterized by a pair of numbers, where the first stands for the percentage of items correctly identified (top 1), while the second expresses the percentage for which the item was placed within the first ten best matches (top 10).

4.2 1,000 Items Setup

An experimental setup of 1,000 musical items was chosen first, each item stored in the compressed MPEG-1/2 Layer 3 format (at a data rate of 192 kbit/s for a stereo signal). The items were chosen from the combined genre rock/pop, to make a distinction between the items more demanding than if material with a wider diversity of characteristics would have been used. To achieve a fast classification of the test items the processed length was set to 20 seconds while training was limited to 30 seconds, i.e. the data had to be recognized based on an excerpt of the sequence only. The feature extractor uses a block size of 1,024 samples. Both the Loudness and the SFM feature were using 4 frequency bands. After feature extraction, temporal grouping and subsequent transformation techniques were applied prior further processing. The generation of the base classes was conducted as described above (VQ clustering algorithm). The setup described here allowed a classification time of 1 second per item (measured on a Pentium III 500 MHz class PC). A selection of the recognition performance for this setup of the system is reported in Table 1.

Table 1. Recognition performance of Loudness and SFM features (1,000 item setup, top 1/ top 10)

Feature	Loudness	SFM
No distortion	100.0% / 100.0%	100.0% / 100.0%
Cropping 15s	51.0% / 75.5%	92.3% / 99.6%
Equalization	99.6% / 100.0%	14.1% / 29.8%
Dynamic Range Compression	89.5% / 94.9%	99.0% / 99.3%
MPEG-1/2 Layer 3 @ 96 kbit/s	19.0% / 33.3%	90.0% / 98.6
Loudspeaker / Microphone Chain	38.3% / 61.7%	27.2% / 59.7%

As can be seen from these figures, the Loudness feature provides a rather low recognition performance for the case of cropping effects (further restriction to 15s length) or MPEG-1/2 Layer-3 robustness. In contrast to this, SFM shows very good performance concerning these robustness tests. Both features do not perform very well in this configuration for the loudspeaker/microphone chain experiment.

4.3 15,000 Items Setup

This setup represents one significant step on the way to a "real world" scenario. A set of 15,000 items was chosen as a database for the classification system, representing a clearly more demanding task. Again the chosen test items belong mostly to the rock/pop genre. To cope with the two main points of interest (namely speed and discrimination) while handling this amount of data, some improvements were made compared to the previous setup. To realize an even faster classification speed with a larger number of items, the statistical analysis of the features was exploited and used for classification instead of the raw features themselves. Furthermore, the number of frequency bands was increased from 4 to 16 bands in order to achieve a more precise description of the audio signal.

A further difference compared to the previous setup is the fact that the features were implemented in accordance with the time/frequency resolution as specified for the extraction of Low Level Descriptors (LLDs) by the MPEG-7 audio standard [2] (i.e. same window/DFT and shift length).

Tables 2 and 3 show the recognition performance achieved for this experimental setup, now investigating the behavior of the promising features which are related to the signal's spectral flatness properties (and thus "tone-likeness"). Table 2 reports the classification results of a standard Vector Quantization approach, whereas Table 3 shows the results for a more advanced matching algorithm including aspects of temporal relationship between subsequent feature vectors. As can be seen from the figures, both features (SFM and SCF) perform extremely well even under severe distortion conditions, such as the loudspeaker/microphone chain. It can be observed that the SFM feature performs very good while using a standard VQ classifier. This is further increased to recognition rates above 97% with the more sophisticated matching algorithm. In both cases, SCF shows an even better recognition performance. Being at some kind of "saturation level" further tests with an increased amount of items and additional robustness requirements are mandatory for a better discrimination of the two features. Classification time is 7.5 seconds for standard and 2.5 seconds for advanced matching (per item).

Table 2. Recognition performance of *SFM* and *SCF* features using *standard* matching (15,000 item setup)

Feature	SFM	SCF
No distortion	100.0% / 100.0%	100.0% / 100.0%
Cropping	100.0% / 100.0%	100.0% / 100.0%
MPEG-1/2 Layer 3 @ 96 kbit/s	96.1% / 97.2%	99.4% / 99.6%
MPEG-1/2 Layer 3 @ 96 kbit/s & Cropping	92.2% / 94.3%	98.8% / 99.3%

Table 3. Recognition performance of *SFM* and *SCF* features using *advanced* matching (15,000 item setup)

Feature	SFM	SCF
No distortion	100.0% / 100.0%	100.0% / 100.0%
Cropping	100.0% / 100.0%	100.0% / 100.0%
MPEG-1/2 Layer 3 @ 96 kbit/s	99.6% / 99.8%	100.0% / 100.0%
MPEG-1/2 Layer 3 @ 96 kbit/s & Cropping	97.9% / 99.9%	99.7% / 100.0%
Loudspeaker / Microphone Chain & Cropping	98.0% / 99.0%	98.8% / 99.5%

5. APPLICATIONS

The identification of audio content based on matching to a database of known works has many attractive applications, some of which are presented in the following:

- **Audio Fingerprinting:** Matching of audio signals as described in this paper is closely related to the much-discussed topic of "Audio Fingerprinting". A compact representation of the signal features for matching (e.g. the VQ codebooks) resembles the condensed "essence" of the audio item and is thus usable as a fingerprint of the corresponding item.
- **Identification of music and linking to metadata:** Automated identification of audio signals is a universal mechanism for finding associated descriptive data (metadata) for a given piece of audio content. This is especially useful when the format the content has been delivered in is irrelevant for the identification process and when furthermore this format does not support the transport of associated metadata or reference thereto. Under these premises recognition of the song will also serve to provide links to the corresponding metadata. Since the metadata is not necessarily embedded in the audio content, access to a remote database could carry updated information on the artist, concerts, new releases and so on.
- **Broadcast monitoring:** A system for automatic audio recognition can identify and protocol transmitted audio program material on broadcasting stations. With a system like the one introduced in this paper this can be achieved without the need for special processing of the transmitted audio material, as would otherwise be required when using branding methods like watermarking. Applications that require monitoring of radio programs would include verification of scheduled transmission of advertisement spots, securing the composer’s royalties for broadcast material or statistical analysis of program material (charts analysis).
- **Music Sales:** Automatic audio identification can also be used to retrieve ordering and pricing information of the identified material and additionally offer similar material. The recording of sound/music and storage of the signature on small handheld devices (such as Personal Digital Assistants) will enable the customer to find the recorded music item in the music store or by connecting to the Internet.

6. MPEG-7 AND ROBUST IDENTIFICATION OF AUDIO

Due to the ever-increasing amount of multimedia material which is available to users, efficient management of such material by means of so-called content-related techniques is of growing importance. This goal can be achieved by using pre-computed descriptive data ("metadata") which is associated with the content. One example of a number of upcoming metadata standards for audiovisual data is the MPEG-7 [2] process which is planned to be finalized in a first version in late 2001.

MPEG-7 defines a wide framework for the description of audio, visual and generic properties of multimedia content, covering both high level semantic concepts as well as low level features (the latter can be extracted directly from the signal itself) [17].

The basic descriptive entities in MPEG-7 are called *Descriptors* (D) and represent specific content properties or attributes by means of a defined syntax and semantics. *Description Schemes* (DS) are intended to combine components with view towards application and may comprise both Descriptors and other Description Schemes. Both Descriptors and Description Schemes are syntactically defined by a so-called *Description Definition Language* (DDL) which also provides the ability for future extension/modification of existing elements. The MPEG-7 DDL is based on XML Schema as the language of choice for the textual representation of content description and for allowing extensibility of description tools.

In the area of audio signal description, MPEG-7 provides a set of *Low Level Descriptors* (LLDs) which are defined in terms of both syntactic format and semantics of the extraction process. While these descriptors can be considered to form a universal toolbox for many future applications, a number of concrete functionalities have already been envisaged during the development process of the standard [2]. These include "Query by humming"-type search for music, sound effects recognition, musical instrument timbre description, annotation of spoken content and robust matching of audio signals.

Specifically, the functionality of content-related identification of audio signals is supported within MPEG-7 audio by means of the `AudioSpectrumFlatness` low level descriptor which is designed to support robust matching of a pair of audio signals, namely the unknown signal and the known reference signal. The `AudioSpectrumFlatness` descriptor specifies the flatness property of the signal's power spectrum within a certain number of frequency bands, i.e. the underlying feature of the recognition system, as described previously. Using the *Scalable Series* concept, this data can be delivered with varying temporal granularity to achieve different tradeoffs between descriptive accuracy and compactness.

This standardized descriptor design forms the basis for achieving an open, interoperable platform for automatic audio identification:

- Identification relies on a published, open feature format rather than proprietary solutions. This allows all potential users to easily produce descriptive data for the audio works of interest (e.g. descriptions of newly released songs).
- Due to the exact standardized specification of the descriptor, interoperability is guaranteed on a worldwide basis, i.e. every search engine relying on the MPEG-7 specification will be able to use compliant descriptions, wherever they may have been produced.

In this sense, MPEG-7 provides a point of interoperability for these applications at the feature level. Since textual descriptions based on an XML representation are not designed to provide extremely compact representations, applications may choose to transcode the MPEG-7 compliant description into a smaller, compressed representation for storage in an internal database ("fingerprint", "signature"). Still, the "un-packed" representation will remain to be available as a point of interoperability with other schemes.

7. CONCLUSIONS AND OUTLOOK

This paper discussed methods for achieving automatic content-based identification of audio material by means of robust matching to a set of known reference items. Particular attention was paid to aspects of robustness with respect to common types of signal alterations, including both linear and non-linear distortions, audio compression and cropping to a reasonably-sized excerpt. The ability to handle these types of distortions is vital to the usability of systems for content-based processing in many real-world application scenarios.

Relying on a general feature extraction/pattern recognition paradigm, a prototype system for automatic identification of audio material was described in its architecture and background. Clearly, the selection of appropriate robust features can be considered crucial for achieving a good recognition performance under a wide range of possible distortions.

Recognizing the importance of the application, the upcoming MPEG-7 audio standard defines a descriptor designed to provide the functionality of robust matching of pairs of audio signals which relates to the "un-flatness" of the signal's power spectrum and thus the tone-like quality of the signal in a number of frequency bands.

Using this (and related) features, the recognition performance of the identification system was assessed in a number of experiments. The system configuration used showed excellent matching performance for a test set comprising 15,000 songs. A correct identification rate of better than 98% was achieved even for severe distortion types, including an acoustic transmission over a loudspeaker/microphone chain. The system runs about 80 times real-time performance on a Pentium III 500MHz class PC.

Clearly, there is still a long way to go until such an automatic system will be able to match the recognition performance of a human listener. Nonetheless, the current level of performance already opens the door for a number of very interesting applications, including finding associated metadata for a given piece of audio content, broadcast monitoring and music sales.

8. REFERENCES

[1] S. Hacker. *MP3: The Definitive Guide*. O'Reilly, 2000.

[2] ISO-IEC/JTC1 SC29 WG11 Moving Pictures Expert Group. Information technology – multimedia content description interface – part 4: Audio. Comitee Draft 15938-4, ISO/IEC, 2000.

[3] Gracenote homepage: http://www.gracenote.com

[4] E. Scheirer, and M. Slaney. *Construction and evaluation of a robust multifeature speech music discriminator*. In ICASSP, 1997.

[5] R. M. Aarts, and R. T. Dekkers. *A real-time speech-music discriminator*. J. Audio Eng. Soc., 47(9), 1999.

[6] K. El-Maleh, M. Klein, G. Petrucci, and P. Kabal. *Speech music discrimination for multimedia applications*. In ICASSP, vol. IV, pages 2445-2448, 2000.

[7] Cantametrix homepage: http://www.cantametrix.com

[8] Musclefish homepage: http://www.musclefish.com

[9] E. Wold, T. Blum, D. Keislar, and J. Wheaton. *Content-based classification, search, and retrieval of audio*. In IEEE Multimedia, vol. 3, pages 27-36, 1996.

[10] D. Pye. *Content-based methods for the management of digital music*. In ICASSP, vol. IV, pages 2437-2440, 2000.

[11] Relatable homepage: http://www.relatable.com

[12] C. Papaodysseus, G. Roussopoulos, D. Fragoulis, T. Panagopoulos, and C. Alexiou. *A new approach to the*

automatic recognition of musical recordings. J. Audio Eng. Soc., 49(1/2), 2001.

[13] A. K. Jain, R. P. W. Duin, and J. Mao. *Statistical Pattern Recognition: A Review*. IEEE Transaction in Pattern Analysis and Machine Intelligence, 2(1), 2000.

[14] D. Kil, and F. Shin. *Pattern Recognition and Prediction with Applications to Signal Characterization*. American Institute of Physics, 1996.

[15] E. Zwicker, and H. Fastl. *Psychoacoustics*. Springer, Berlin, 2nd edition, 1999.

[16] N. Jayant, and P. Noll. *Digital Coding of Waveforms*. Prentice-Hall, Englewood Cliffs, NJ, 1984.

[17] ISO/IEC JTC1/SC29/WG11 (MPEG): "Introduction to MPEG-7", available from http://www.cselt.it/mpeg.

Automatic Musical Genre Classification Of Audio Signals

George Tzanetakis
Computer Science Department
35 Olden Street
Princeton NJ 08544
+1 609 258 5030
gtzan@cs.princeton.edu

Georg Essl
Computer Science Dep.
35 Olden Street
Princeton NJ 08544
+1 609 258 5030
gessl@cs.princeton.edu

Perry Cook
Computer Science and Music Dep.
35 Olden Street
Princeton NJ 08544
+1 609 258 5030
prc@cs.princeton.edu

ABSTRACT
Musical genres are categorical descriptions that are used to describe music. They are commonly used to structure the increasing amounts of music available in digital form on the Web and are important for music information retrieval. Genre categorization for audio has traditionally been performed manually. A particular musical genre is characterized by statistical properties related to the instrumentation, rhythmic structure and form of its members. In this work, algorithms for the automatic genre categorization of audio signals are described. More specifically, we propose a set of features for representing texture and instrumentation. In addition a novel set of features for representing rhythmic structure and strength is proposed. The performance of those feature sets has been evaluated by training statistical pattern recognition classifiers using real world audio collections. Based on the automatic hierarchical genre classification two graphical user interfaces for browsing and interacting with large audio collections have been developed.

1. INTRODUCTION
Musical genres are categorical descriptions that are used to characterize music in music stores, radio stations and now on the Internet. Although the division of music into genres is somewhat subjective and arbitrary there are perceptual criteria related to the texture, instrumentation and rhythmic structure of music that can be used to characterize a particular genre. Humans are remarkably good at genre classification as investigated in [1] where it is shown that humans can accurately predict a musical genre based on 250 milliseconds of audio. This finding suggests that humans can judge genre using only the musical surface without constructing any higher level theoretic descriptions as has been argued in [2]. Up to now genre classification for digitally available music has been performed manually. Therefore techniques for automatic genre classification would be a valuable addition to the development of audio information retrieval systems for music.

In this work, algorithms for automatic genre classification are explored. A set of features for representing the music surface and rhythmic structure of audio signals is proposed. The performance of this feature set is evaluated by training statistical pattern recognition classifiers using audio collections collected from compact disks, radio and the web. Audio signals can be automatically classified using a hierarchy of genres that can be represented as a tree with 15 nodes. Based on this automatic genre classification and the extracted features two graphical user interfaces for browsing and interacting with large digital music collections have been developed. The feature extraction and graphical update of the user interfaces is performed in real time and has been used to classify live radio signals.

2. RELATED WORK
An early overview of audio information retrieval (AIR) (including speech and symbolic music information retrieval) is given in [3]. Statistical pattern recognition based on the extraction of spectral features has been used to classify Music vs Speech [4], Isolated sounds [5, 6] and Instruments [7]. Features related to timbre recognition have been explored in [8,9]. Extraction of psychoacoustic features related to music surface and their use for similarity judgements and high level semantic descriptions (like slow or loud) is explored in [10]. Content-based similarity retrieval from large collections of music is described in [11]. Automatic beat tracking systems have been proposed in [12, 13] and [14] describes a method for the automatic extraction of time indexes of occurrence of different percussive timbres from an audio signal. Musical genres can be quite subjective making automatic classification difficult. The creation of a more objective genre hierarchy for music information retrieval is discussed in [15]. Although the use of such a designed hierarchy would improve classification results it is our belief that there is enough statistical information to adequately characterize musical genre. Although manually annotated genre information has been used to evaluate content-based similarity retrieval algorithms to the best of our knowledge, there is no prior published work in automatic genre classification.

3. FEATURE EXTRACTION

3.1 Musical Surface Features

In this work the term "musical surface" is used to denote the characteristics of music related to texture, timbre and instrumentation. The statistics of the spectral distribution over time can be used in order to represent the "musical surface" for pattern recognition purposes. The following 9-dimensional feature vector is used in our system for this purpose: (**mean-Centroid, mean-Rolloff, mean-Flux, mean-ZeroCrossings, std-Centroid, std-Rolloff, std-Flux, std-ZeroCrossings, LowEnegry**).

The means and standard deviations of these features are calculated over a "texture" window of 1 second consisting of 40 "analysis" windows of 20 milliseconds (512 samples at 22050 sampling rate). The feature calculation is based on the Short Time Fourier Transform (STFT). that can be efficiently calculated using the Fast Fourier Transform (FFT) algorithm [16].

The following features are calculated for each "analysis" window: (M[f] is the magnitude of the FFT at frequency bin f and N the number of frequency bins):

- **Centroid :** $$C = \frac{\sum_1^N fM[f]}{\sum_1^N M[f]} \quad \textbf{(1)}$$

  The Centroid is a measure of spectral brightness.

- **Rolloff :** is the value R such that :

  $$\sum_1^R M[f] = 0.85 \sum_1^N M[f] \quad \textbf{(2)}$$

  The rolloff is a measure of spectral shape.

- **Flux:** $$F = \left\| M[f] - M_p[f] \right\| \quad \textbf{(3)}$$

  where M_p denotes the FFT magnitude of the previous frame in time. Both magnitude vectors are normalized in energy. Flux is a measure of spectral change.

- **ZeroCrossings:** the number of time domain zerocrossings of the signal. ZeroCrossings are useful to detect the amount of noise in a signal.

- **LowEnergy:** The percentage of "analysis" windows that have energy less than the average energy of the "analysis" windows over the "texture" window.

3.2 Rhythm features

The calculation of features for representing the rhythmic structure of music is based on the Wavelet Transform (WT) which is a technique for analyzing signals that was developed as an alternative to the STFT. More specifically, unlike the STFT that provides uniform time resolution for all frequencies the DWT provides high time resolution for all frequencies, the DWT provides high time resolution and low frequency resolution for high frequencies and high time and low frequency resolution for low frequencies.

The Discrete Wavelet Transform (DWT) is a special case of the WT that provides a compact representation of the signal in time and frequency that can be computed efficiently. The DWT analysis can be performed using a fast, pyramidal algorithm related to multirate filterbanks [17]. An introduction to wavelets can be found in [18].

For the purposes of this work, the DWT can be viewed as a computationally efficient way to calculate an octave decomposition of the signal in frequency. More specifically the DWT can be viewed as a constant Q (bandwidth / center frequency) with octave spacing between the centers of the filters.

In the pyramidal algorithm the signal is analyzed at different frequency bands with different resolutions by decomposing the signal into a coarse approximation and detail information. The coarse approximation is then further decomposed using the same wavelet step. The decomposition is achieved by successive highpass and lowpass filtering of the time domain signal and is defined by the following equations:

$$y_{high}[k] = \sum_n x[n]g[2k-n] \quad \textbf{(4)}$$

$$y_{low}[k] = \sum_n x[n]h[2k-n] \quad \textbf{(5)}$$

where $y_{high}[k], y_{low}[k]$ are the output of the highpass (g) and lowpass (h) filters, respectively after subsampling by two. The DAUB4 filters proposed by Daubechies [19] are used.

The rhythm feature set is based on detecting the most salient periodicities of the signal. Figure I shows the flow diagam of the beat analysis. The signal is first decomposed into a number of octave frequency bands using the DWT. Following this decomposition the time domain amplitude envelope of each band is extracted separately. This is achieved by applying full wave rectification, low pass filtering and downsampling to each band. The envelopes of each band are then summed together and an autocorrelation function is computed. The peaks of the autocorrelation function correspond to the various periodicities of the signal's envelope. These stages are given by the equations:

1. **Full Wave Rectification (FWR):**

   $$y[n] = abs(x[n]) \quad \textbf{(6)}$$

2. **Low Pass Filtering (LPF):** (One Pole filter with an alpha value of 0.99) i.e:

   $$y[n] = (1-\alpha)x[n] - \alpha y[n] \quad \textbf{(7)}$$

3. **Downsampling (↓)** by k (k=16 in our implementation)**:**

   $$y[n] = x[kn] \quad \textbf{(8)}$$

4. **Normalization (NR)** (mean removal)**:**

   $$y[n] = x[n] - E[x[n]] \quad \textbf{(9)}$$

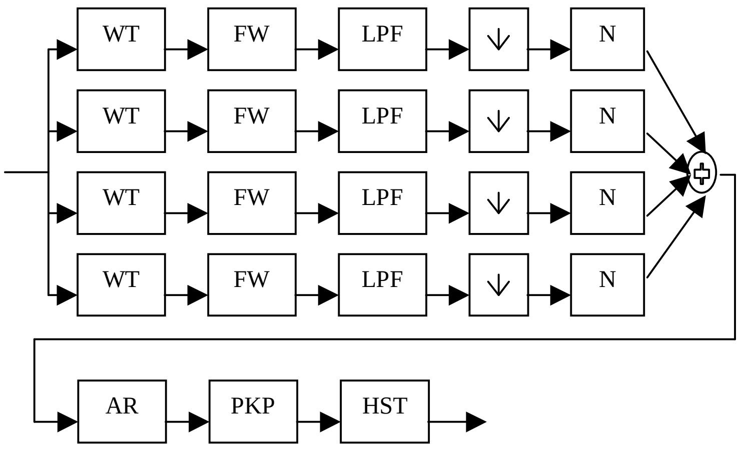


Fig. I Beat analysis flow diagram

5. **Autocorrelation (AR)** (computed using the FFT for efficiency) **:**

$$y[n] = \frac{1}{N}\sum_{n} x[n]x[n+k] \qquad \textbf{(10)}$$

The first five peaks of the autocorrelation function are detected and their corresponding periodicities in beats per minute (bpm) are calculated and added in a "beat" histogram. This process is repeating by iterating over the signal and accumulating the periodicities in the histogram. A window size of 65536 samples at 22050 Hz sampling rate with a hop size of 4096 samples is used. The prominent peaks of the final histogram correspond to the various periodicities of the audio signal and are used as the basis for the rhythm feature calculation.

The following features based on the "beat" histogram are used:

1. **Period0**: Periodicity in bpm of the first peak Period0
2. **Amplitude0**: Relative amplitude (divided by sum of amplitudes) of the first peak.
3. **RatioPeriod1**: Ratio of periodicity of second peak to the periodicity of the first peak
4. **Amplitude1**: Relative amplitude of second peak.
5. **RatioPeriod2, Amplitude2**, **RatioPeriod3, Amplitude3**

These features represent the strength of beat ("beatedness") of the signal and the relations between the prominent periodicities of the signal. This feature vector carries more information than traditional beat tracking systems [11, 12] where a single measure of the beat corresponding to the tempo and its strength are used.

Figure II shows the "beat" histograms of two classical music pieces and two modern pop music pieces. The fewer and stronger peaks of the two pop music histograms indicate the strong presence of a regular beat unlike the distributed weaker peaks of classical music.

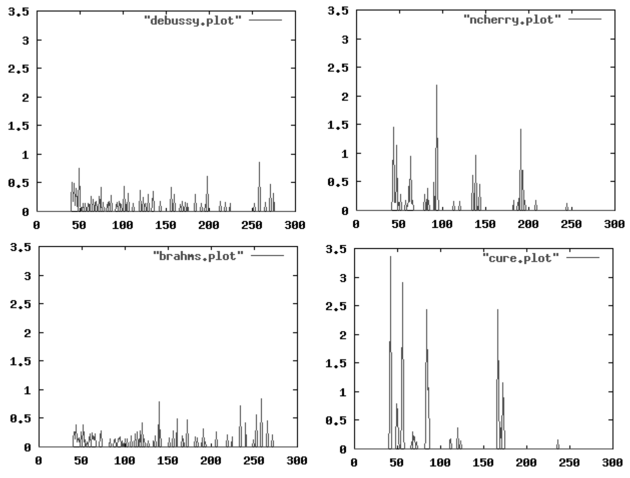


Fig. II Beat Histograms for Classical (left) and Pop (right)

The 8-dimensional feature vector used to represent rhythmic structure and strength is combined with the 9-dimensional musical surface feature vector to form a 17-dimensional feature vector that is used for automatic genre classification.

4. CLASSIFICATION

To evaluate the performance of the proposed feature set, statistical pattern recognition classifiers were trained and evaluated using data sets collected from radio, compact disks and the Web. Figure III shows the classification hierarchy used for the experiments. For each node in the tree of Figure III, a Gaussian classifier was trained using a dataset of 50 samples (each 30 seconds long). Using the Gaussian classifier each class is represented as a single multidimensional Gaussian distribution with parameters estimated from the training dataset [20]. The full digital audio data collection consists of 15 genres * 50 files * 30 seconds = 22500 seconds (i.e 6.25 hours of audio).

For the Musical Genres (Classical, Country…..) the combined feature set described in this paper was used. For the Classical Genres (Orchestra, Piano…) and for the Speech Genres (MaleVoice, FemaleVoice…) mel-frequency cepstral coefficients [21] (MFCC) were used. MFCC are perceptually motivated features commonly used in speech recognition research. In a similar fashion to the Music Surface features, the means and standard deviations of the first five MFCC over a larger texture window (1 second long) were calculated. MFCCs can also be used in place of the STFT-based music surface features with similar classification results. The use of MFCC as features for classifying music vs speech has been explored in [22].

The speech genres were added to the genre classification hierarchy so that the system could be used to classify live radio signals in real time. "Sports announcing" refers to any type of speech over noisy background.

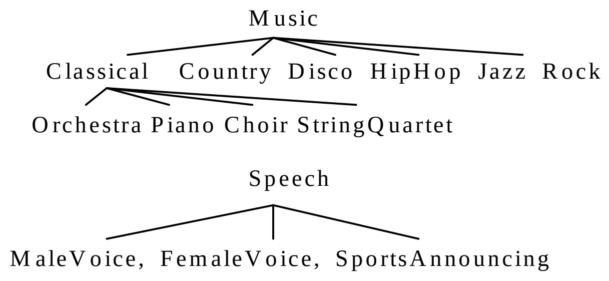


Fig. III Genre Classification Hierarchy

Table 1. Classification accuracy percentage results

	MusicSpeech	**Genres**	**Voices**	**Classical**
Random	50	16	33	25
Gaussian	86	62	74	76

Table 1. summarizes the classification results as pecentages of classification accuracy. In all cases the results are significantly better than random classification. These classification results are calculated using a 10-fold evaluation strategy where the evaluation data set is randomly partitioned so that 10% is used for testing and 90% for training. The process is iterated with different random partitions and the results are averaged (in the evaluation of Table.1 one hundred iterations where used).

Table 2. shows more detailed information about the genre classifier performance in the form of a confusion matrix. The columns correspond to the actual genre and the rows to the predicted genre. For example the cell of row 2, column 1 with value 0.01 means that 1 percent of the Classical music (column 1) was wrongly classified as Country music (row 2). The percentages of correct classifications lie in the diagonal of the confusion matrix. The best predicted genres are classical and hiphop while the worst predicted are jazz and rock. This is due to the fact that the jazz and rock are very broad categories and their boundaries are more fuzzy than classical or hiphop.

Table 3. shows more detailed information about the classical music classifier performance in the form of a confusion matrix..

	classic	*country*	*Disco*	*Hiphop*	*jazz*	*Rock*
classic	86	2	0	4	18	1
country	1	57	5	1	12	13
disco	0	6	55	4	0	5
Hiphop	0	15	28	90	4	18
Jazz	7	1	0	0	.37	12
Rock	6	19	11	0	27	48

Table 2. Genre classification confusion matrix

	choral	*orchestral*	*Piano*	*string 4tet*
choral	99	10	16	12
orchestral	0	53	2	5
piano	1	20	75	3
string 4tet	0	17	7	80

Table 2. Classical music classification confusion matrix

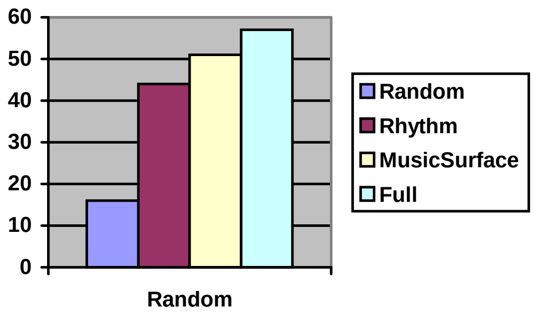


Fig. IV Relative feature set importance

Figure IV shows the relative importance of the “musical surface” and “rhythm” feature sets for the automatic genre classification. As expected both feature sets perform better than random and their combination improves the classification accuracy. The genre labeling was based on the artist or the compact disk that contained the excerpt. In some cases this resulted in outliers that are one of the sources of prediction error. For example the Rock collection contains songs by Sting that are more close to Jazz than Rock even for a human listener. Similarly the Jazz collection contains songs with string accompaniment and no rhythm section that sound like Classical music. It is likely that replacing these outliers with more characteristic pieces would improve the genre classification results.

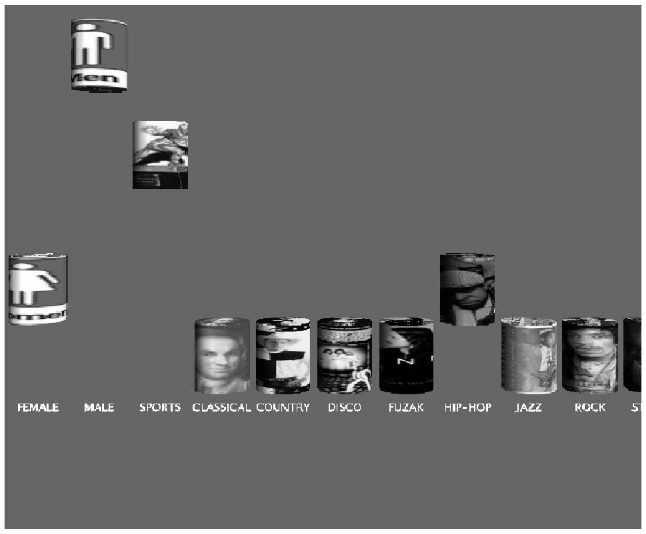


Fig. IV GenreGram

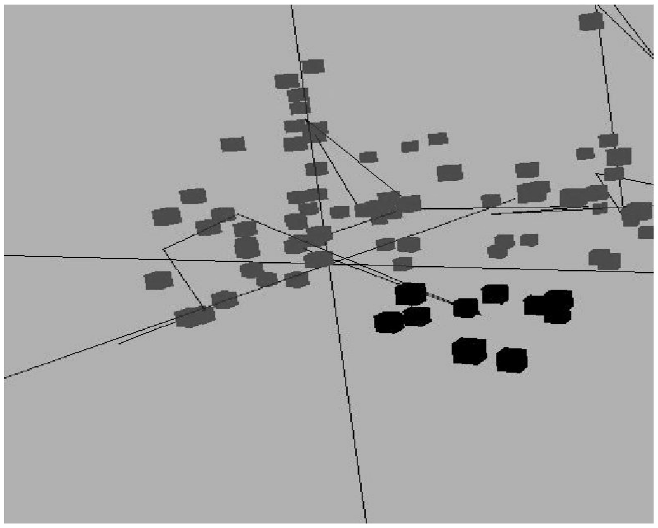


Fig. V GenreSpace

5. USER INTERFACES

Two new graphical user interfaces for browsing and interacting with collections of audio signals have been developed (Figure IV,V) . They are based on the extracted feature vectors and the automatic genre classification results.

- **GenreGram** is a dynamic real-time audio display for showing automatic genre classification results. Each genre is represented as a cylinder that moves up and down in real time based on a classification confidence measure ranging from 0.0 to 1.0. Each cylinder is texture-mapped with a representative image of each category. In addition to being a nice demonstration of automatic real time audio classification, the *GenreGram* gives valuable feedback both to the user and the algorithm designer. Different classification decisions and their relative strengths are combined visually, revealing correlations and classification patterns. Since the boundaries between musical genres are fuzzy, a display like this is more informative than a single classification decision. For example, most of the time a rap song will trigger *Male Voice, Sports Announcing* and *HipHop*. This exact case is shown in Figure IV.

- **GenreSpace** is a tool for visualizing large sound collections for browsing. Each audio file is represented a single point in a 3D space. Principal Component Analysis (PCA) [23] is used to reduce the dimensionality of the feature vector representing the file to the 3-dimensional feature vector corresponding to the point coordinates. Coloring of the points is based on the automatic genre classification. The user can zoom, rotate and scale the space to interact with the data. The *GenreSpace* also represents the relative similarity within genres by the distance between points. A principal curve [24] can be used to move sequentially through the points in a way that preserves the local clustering information.

6. FUTURE WORK

An obvious direction for future research is to expand the genre hierarchy both in width and depth. The combination of segmentation [25] with automatic genre classification could provide a way to browse audio to locate regions of interest. Another interesting direction is the combination of the graphical user interfaces described with automatic similarity retrieval that takes into account the automatic genre classification. In its current form the beat analysis algorithm can not be performed in real time as it needs to collect information from the whole signal. A real time version of beat analysis is planned for the future. It is our belief that more rhythmic information can be extracted from audio signals and we plan to investigate the ability of the beat analysis to detect rhythmic structure in synthetic stimuli.

7. SUMMARY

A feature set for representing music surface and rhythm information was proposed and used to build automatic genre classification algorithms. The performance of the proposed data set was evaluated by training statistical pattern recognition classifiers on real-world data sets. Two new graphical user interfaces based on the extracted feature set and the automatic genre classification were developed.

The software used for this paper is available as part of *MARSYAS* [26] a software framework for rapid development of computer audition application written in C++ and JAVA. It is available as free software under the Gnu Public License (GPL) at:

http://www.cs.princeton.edu/~gtzan/marsyas.html

8. ACKNOWLEDGEMENTS

This work was funded under NSF grant 9984087 and from gifts from Intel and Arial Foundation. Douglas Turnbull helped with the implementation of the GenreGram. The authors would also like to thank the anonymous reviewers for their valuable feedback.

9. REFERENCES

[1] Perrot, D., and Gjerdigen, R.O. Scanning the dial: An exploration of factors in the identification of musical style. In Proceedings of the 1999 Society for Music Perception and Cognition pp.88(abstract)

[2] Martin, K.,D., Scheirer, E.D., Vercoe, B., L. Musical content analysis through models of audition. In Proceedings of the 1998 ACM Multimedia Workshop on Content-Based Processing of Music.

[3] Foote, J. An overview of audio information retrieval. Multimedia Systems 1999. 7(1), 42-51.

[4] Scheirer, E. D. and Slaney, M. Construction and evaluation of a robust multifeature speech/music discriminator. In Proceedings of the 1997 International Conference on Acoustics, Speech, and Signal Processing, 1331-1334.

[5] Wold, E., Blum, T., Keislar, D., and Wheaton, J. Content –based classification, search and retrieval of audio. IEEE Multimedia, 1996 3 (2)

[6] Foote, J., Content-based retrieval of music and audio. In Multimedia Storage and Archiving Systems II, 1997 138-147

[7] Martin, K. Sound-Source Recognition: A theory and computational model. PhD thesis, MIT Media Lab. http://sound.media.mit.edu/~kdm

[8] Rossignol, S et al. Feature extraction and temporal segmentation of acoustic signals. In Proceedings of International Computer Music Conference (ICMC), 1998.

[9] Dubnov, S., Tishby, N., and Cohen, D. Polyspectra as measures of sound and texture. Journal of New Music Research, vol. 26 1997.

[10] Scheirer, E. Music Listening Systems. Phd thesis., MIT Media Lab: http://sound.media.mit.edu/~eds

[11] Welsh, M., Borisov, N., Hill, J., von Behren, R., and Woo, A. Querying large collections of music for similarity. Technical Report UCB/CSD00-1096, U.C Berkeley, Computer Science Division, 1999.

[12] Scheirer, E. Tempo and beat analysis of acoustic musical signals. Journal of the Acoustical Society of America 103(1):588-601.

[13] Goto, M. and Muraoka, Y. Music understanding at the beat level: real time beat tracking for audio signals. In D.F Rosenthal and H. Okuno (ed.), Readings in Computational Auditory Scene Analysis 156-176.

[14] Gouyon, F., Pachet, F. and Delerue, O. On the use of zero-crossing rate for an application of classification of percussive sounds. Proceedings of the COST G-6 conference on Digital Audio Effects (DAFX-00), Verona, Italy, 2000.

[15] Pachet, F., Cazaly, D. “A classification of musical genre”, Content-Based Multimedia Information Access (RIA) Conference, Paris, March 2000.

[16] Oppenheim, A. and Schafer, R. Discrete-Time Signal Processing. Prentice Hall. Edgewood Cliffs, NJ. 1989.

[17] Mallat, S, G. A theory for multiresolution signal decomposition: The Wavelet representation. IEEE Transactions on Pattern Analysis and Machine Intelligence,1989, 11, 674-693.

[18] Mallat, S,G. A wavelet tour of signal processing. Academic Press 1999.

[19] Daubechies, I. Orthonormal bases of compactly supported wavelets. Communications on Pure and Applied Math.1988. vol.41, 909-996.

[20] Duda, R. and Hart, P. Pattern classification and scene analysis. John Willey & Sons. 1973.

[21] Hunt, M., Lennig, M., and Mermelstein, P. Experiments in syllable-based recognition of continuous speech. In Proceedings of International Conference on Acoustics, Speech and Signal Processing, 1996, 880-883.

[22] Logan, B. Mel Frequency Cepstral Coefficients for music modeling. Read at the first International Symposium on Music Information Retrieval.. http://ciir.cs.umass.edu/music2000

[23] Jollife, L. Principal component analysis. Spring Verlag, 1986.

[24] Herman, T, Meinicke, P., and Ritter, H. Principal curve sonification. In Proceedings of International Conference on Auditory Display. 2000.

[25] Tzanetakis, G. and Cook, P. Mutlifeature audio segmentation for browsing and annotation. In Proceedings of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics. 1999.

[26] Tzanetakis, G. and Cook, P. MARSYAS: a framework for audio analysis. Organised Sound 2000. 4(3)

Music Signal Spotting Retrieval by a Humming Query Using Start Frame Feature Dependent Continuous Dynamic Programming

Takuichi Nishimura
Real World Computing Partnership / National Institute of Advanced Industrial Science and Technology
Tsukuba Mitsui Building 13F, 1-6-1 Takezono Tsukuba-shi, Ibaraki
305-0032, Japan
+81-298-53-1686
nishi@rwcp.or.jp

Hiroki Hashiguchi
Real World Computing Partnership
Tsukuba Mitsui Building 13F, 1-6-1 Takezono Tsukuba-shi, Ibaraki
305-0032, Japan
+81-298-53-1668
hiro@rwcp.or.jp

Junko Takita
Mathematical Systems Inc.
2-4-3 Shinjuku, Shinjuku-ku, Tokyo
160-0022 Japan
+81-3-3358-1701
takita@msi.co.jp

J. Xin Zhang
Media Drive Co.
3-195 Tsukuba, Kumagaya-shi, Saitama, 360-0037, Japan
+81-48-524-0501
chou@mediadrive.co.jp

Masataka Goto
National Institute of Advanced Industrial Science and Technology /"Information and Human Activity" PRESTO, JST
1-1-1 Umezono, Tsukuba-shi, Ibaraki
305-8568, Japan
+81-298-61-5898
m.goto@aist.go.jp

Ryuichi Oka
Real World Computing Partnership
Tsukuba Mitsui Building 13F, 1-6-1 Takezono Tsukuba-shi, Ibaraki
305-0032, Japan
+81-298-53-1686
oka@rwcp.or.jp

ABSTRACT
We have developed a music retrieval method that takes a humming query and finds similar audio intervals (segments) in a music audio database. This method can also address a personally recorded video database containing melodies in its audio track. Our previous retrieving method took too much time to retrieve a segment: for example, a 60-minute database required about 10-minute computation on a personal computer. In this paper, we propose a new high-speed retrieving method, called *start frame feature dependent continuous Dynamic Programming*, which assumes that the pitch of the interval start point is accurate. Test results show that the proposed method reduces retrieval time to about 1/40 of present methods.

1. INTRODUCTION
Although a large amount and variety of musical audio signals have been available on the internet and stored in personal hard disks at home, information retrieval methods for those signals are still in infancy. A typical method is a simple text-based search which seeks property tags attached to those signals, such as the song title, artist name, and genre. The purpose of this research is to enable a user to retrieve a segment of a musical audio signal desired just by singing its melody. Such a music retrieval method is also useful for retrieving video clips if those clips contain music in audio tracks. For practical application, we think it important that the method can be applied to an audio database that is not segmented into musical pieces (as is often the case with broadcast recordings).

In this paper, we focus on a music retrieval system that takes a humming query and finds similar audio intervals (segments) in a music audio database. If the database is composed of melody scores such as MIDI, symbols (e.g., relative pitch change or span change) can be extracted robustly. In this case, symbol-based retrieval [1-4] is very efficient. On the other hand, extracted melody from an audio signal usually suffers a lot of error and such symbol-based methods are not applicable. Therefore, we developed the pattern-based retrieval method [5]. Fundamentally, we find a similar pitch sequence of a query (query pattern) in the melody-likeness pattern on the pitch-temporal plane obtained from the database shifting the query pattern along pitch axis and warping temporally as shown in Figure 1. The previous matching method is called *model-driven path Continuous Dynamic Programming (mpCDP)* which compares the reference and all the partial intervals in the database and outputs similar intervals by considering multiple possibilities of transpositions. Because the mpCDP achieves pattern-based matching, the method can also take whistle sound or tempo-varying humming. (In the following, we use two terms, "reference pattern" and "input pattern": we extract a "reference pattern" from a query and an "input pattern" from a database in order to feed those patterns into matching methods.) This previous mpCDP, however, is too slow to be used in practical applications: it needs much computational cost because it accumulates local similarities in 3-D space, which is composed from model-axis, input-axis, and pitch-axis.

In this paper, we propose a new quick retrieval method, called *start frame feature dependent continuous DP (s-CDP),* which searches only in 2-D space (reference-axis and input-axis) assuming that the feature (pitch in this paper) of the start point of the extracted optimal interval is accurate. Our new s-CDP is different from conventional continuous DP in the respect that we do not calculate local similarities beforehand because the start point of the optimal interval is obtained from bottom left to top right successively in the 2-D space.

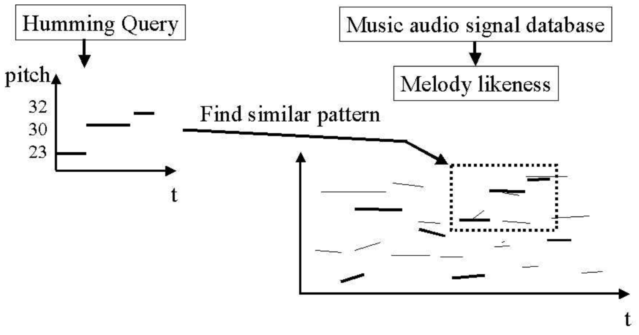


Figure 1. Pattern matching for humming retrieval. Pitch shift and temporal-warp are considered.

We evaluate our new method using 20 popular music selections comparing the conventional method to show that the proposed method can reduce search time to about 1/40 with retrieval rates in excess of 70%.

The following section describes the conventional retrieval method. The s-CDP and the retrieval method using s-CDP is proposed in Section 3. The method is evaluated in Section 4 and newly developed retrieval system is introduced in Section 5.

2. OUR CONVENTIONAL RETRIEVAL METHOD

Continuous DP [7] is improved from temporally monotonous Dynamic Programming (DP) for speech or gesture recognition. Continuous DP achieves inconsistent recognition because it can segment the input pattern automatically but it cannot consider multiple possibilities of transpositions. Therefore, we proposed mpCDP [5] for music retrieval method adding one dimension (pitch) for input pattern and finding similar pitch sequence to the query.

2.1 Continuous DP

As shown in Figure 2 (a), continuous DP calculates accumulated similarities $S(t,\tau)$ between reference R_τ $(1 \le \tau \le T)$ and input I_t $(0 \le t \le \infty)$ by adding local similarities $s(t,\tau)$. Denoting the temporal axis of reference and input as t and τ , respectively, continuous DP calculates $S(t,\tau)$ using the following recursive equations.

Boundary conditions $(1 \le \tau \le T, 0 \le t)$:

$$S(t,0) = S(t,-1) = 0$$

$$S(-1,\tau) = S(0,\tau) = 0$$

Recursive equations $(1 \le t)$:

$$S(t,1) = 3 \cdot s(t,1)$$

$$S(t,\tau) = \max\begin{cases} S(t-2,\tau-1) + 2s(t-1,\tau) + s(t,\tau) \\ S(t-1,\tau-1) + 3s(t,\tau) \\ S(t-1,\tau-2) + 3s(t,\tau-1) + 3s(t,\tau) \end{cases} \quad (1)$$

$$(2 \le \tau \le T)$$

In this equation, local path with maximum similarity is chosen among the three paths as shown in Figure 2(c). Numbers besides each point are weights for local similarities. Notice that the summation of weights is always three for one frame up along the reference axis. Therefore, dividing accumulated similarities by three can normalize them.

Next, we find a similar interval. Figure 2(b) shows accumulated similarities $S(t,T)$; and the continuous DP decide the maximum point with higher similarity than a certain threshold α as the end point of the similar interval.

In brief, continuous DP finds the optimal path and maximum accumulated similarities by choosing the maximum local path successively from bottom left to top right in the reference-input plane. The $S(t,T)$ is the similarity between the whole reference and the input considering temporal warps from 1/2 to 2 times.

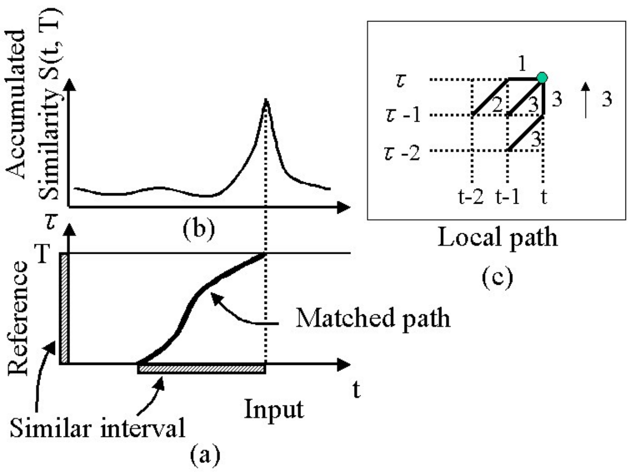


Figure 2. Continuous Dynamic Programming.

2.2 Humming Retrieval by mpCDP

This section explains the conventional humming retrieval method using Figure 3. The method has three steps. First, database audio signals are analyzed every frame, 64ms in this paper, and sequence of melody-likeness of each pitch (SMLP) is obtained. When a query is input to the method, highest melody-likeness pitch is chosen from SMLP every frame and query model is created by the relative pitch change. Third, model-driven path Continuous Dynamic Programming (mpCDP) compares the model and all partial intervals in the database and output similar intervals by considering multiple possibilities of transpositions.

The local path of mpCDP is different from that of continuous DP in that they are shifted along the pitch axis according to the model, which is the relative pitch change of the query. By executing such processes successively, accumulated similarity $S(t,T,x)$ (Here x denotes pitch axis) takes a maximum similarity along the pitch axis at the end of the model as shown in top-right figure of Figure 3. Then, $\max_x S(t,T,x)$ changes similar to Figure 2(b). Finally the mpCDP outputs the maximum point with higher similarity than a certain threshold α as the end point of the similar interval.

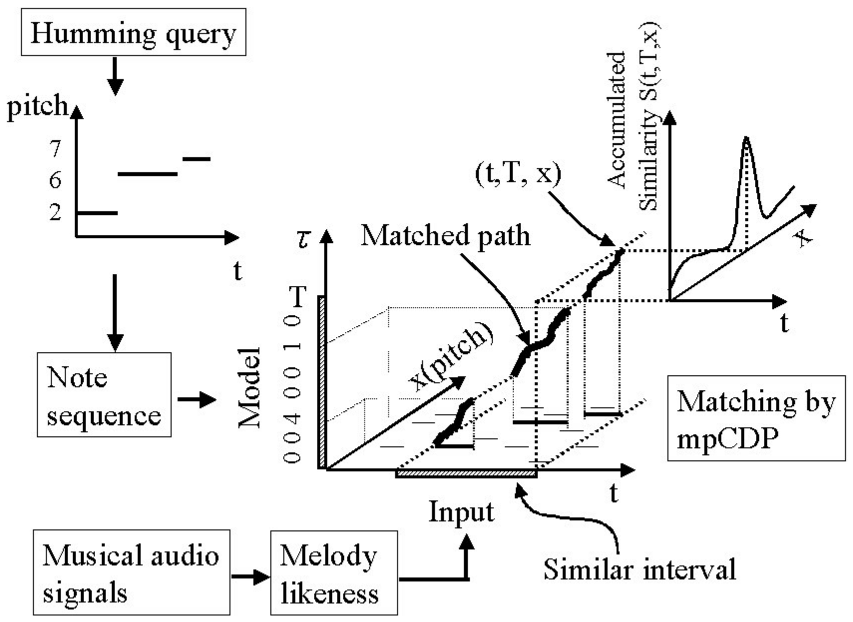


Figure 3. Conventional retrieval method.

3. PROPOSAL OF s-CDP

3.1 s-CDP

The definition of s-CDP is that local similarities $s(t,\tau)$ are dependent on the feature of the start point $(p(t,\tau),1)$ of optimal paths. Therefore local similarities of s-CDP are described as $s(t,\tau)=f(R_\tau,I_t,R_1,I_{p(t,\tau)})$ using a certain similarity function $f()$. (For continuous DP: $s(t,\tau)=f(R_\tau,I_t)$).

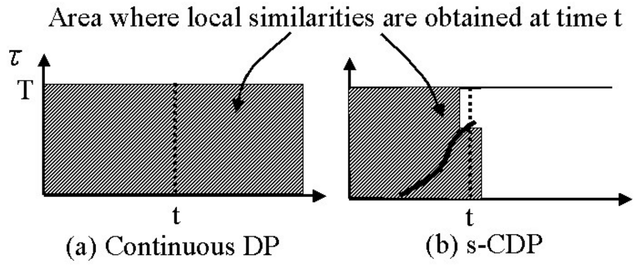


Figure 4. Comparison of matching methods.

The difference between conventional continuous DP and s-CDP is shown in Figure 4. All local similarities can be calculated beforehand for continuous DP (Figure 4(a)), whereas local similarities are obtained incrementally as the optimal paths are fixed for s-CDP as shown in Figure 4(b). There are three local similarities for each point (t,τ) because the start points are different among three local paths as shown in Figure 2 (c). Equation (1) is rewritten for s-CDP as:

Recursive equations ($1 \le t$):

$$S(t,1) = 3 \cdot s_2(t,1)$$

$$S(t,\tau) = \max \begin{cases} S(t-2,\tau-1) + 2s_2(t-1,\tau) + s_1(t,\tau) \\ S(t-1,\tau-1) + 3s_2(t,\tau) \\ S(t-1,\tau-2) + 3s_2(t,\tau-1) + 3s_3(t,\tau) \end{cases} \quad (2)$$

$$(2 \le \tau \le T)$$

Here we call the local paths in Figure 2 (c) path1, path2, path3 from top-left and define $s_1(t,\tau)$, $s_2(t,\tau)$, $s_3(t,\tau)$ as local similarities for path1, path2, and path3, respectively. Those are defined as:

$$s_2(t,\tau) = f(R_\tau, I_t, R_1, I_{p(t,\tau)}) \ (\tau = 1)$$

$$s_1(t,\tau) = f(R_\tau, I_t, R_1, I_{p(t-2,\tau-1)}) \ (2 \le \tau)$$

$$s_2(t,\tau) = f(R_\tau, I_t, R_1, I_{p(t-1,\tau-1)}) \ (2 \le \tau)$$

$$s_3(t,\tau) = f(R_\tau, I_t, R_1, I_{p(t-1,\tau-2)}) \ (2 \le \tau)$$

The $s_2(t,\tau)$ alone takes $\tau = 1$ and requires exception because the start point is the same as the point $(t,1)$. Second terms of path1 and path3 in equation (2) are $s_2(\cdot,\cdot)$ because both points come from points (–1,–1) relatively in the plane.

Outputs of s-CDP are obtained as points with maximum accumulated similarities in the same way as continuous DP, but they have positions of start points and features of start points.

To calculate the start position $p(t,\tau)$ on the input axis, first initialize with time t when $\tau = 0,1$.

$$p(t,0) = p(t,1) = t$$

Next, copy the start point memorized at the local start point to $p(t,\tau)$ as follows:

$$p(t,\tau) = \begin{cases} p(t-2,\tau-1) \ (\textit{if path}1) \\ p(t-1,\tau-1) \ (\textit{if path}2) \\ p(t-1,\tau-2) \ (\textit{if path}3) \end{cases}$$

Start positions $(p(t,\tau),1)$ are obtained incrementally by the recursive equations above.

In case of error in start points, features of start points $R_1, I_{p(t,\tau)}$ are incorrect - leading to miscalculation of accumulated similarities. Therefore, one must choose a high melody-likeness point as the start point in order to segment the reference. As for input, segmenting a similar interval is the method's purpose, so such a measure is impossible. Still, those errors are recovered if one feature near the start point is correct because s-CDP matches, warping the reference temporarily.

3.2 Apply for Humming Retrieval

In this section, s-CDP is modified for the humming retrieval method. Because melody is expressed as the pitch sequence, the feature is pitch and each pitch is subtracted from the start pitch to cope with transpositions, assuming that start pitch is correct. Query is transposed to make the start pitch 0 as shown in Figure 5. On the other hand, database transposition is possible just after the start pitch is fixed by tracing back the optimal path derived from s-CDP. Then the segmented database is similarly transposed to make the start pitch 0 as shown in Figure 5. Finally, s-CDP compares the transposed query and database without any influence of transpositions.

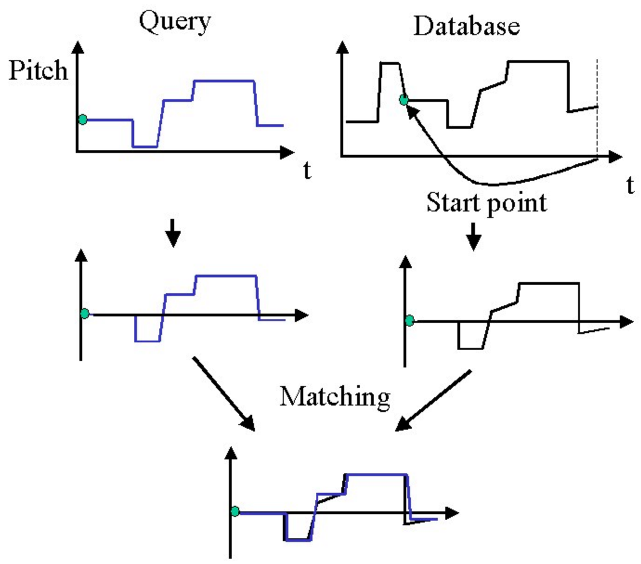


Figure 5. Overview of melody matching considering transpositions.

Figure 6 helps to explain the detail method. First, make relative pitch sequence of query as the reference $R_\tau^{'} = R_\tau - R_1$. Second, obtain N high melody-likeness candidates from database musical signal as input $I_t(k)(k = 1, \cdots, N)$. In this study, local similarities $s(t,\tau)$ are defined:

$$s(t,\tau) = \begin{cases} 1(D_{t,\tau} = 0) \\ 0(D_{t,\tau} \neq 0) \end{cases}$$

Here $D_{t,\tau} = \min_k abs[R_\tau^{'} - \{I_t(k) - I_{p(t,\tau)}(1)\}]$ and local similarities are 1 if one of the relative pitches of N candidates is equal to relative pitch of the query. We set $N = 5$ in experiments in this paper.

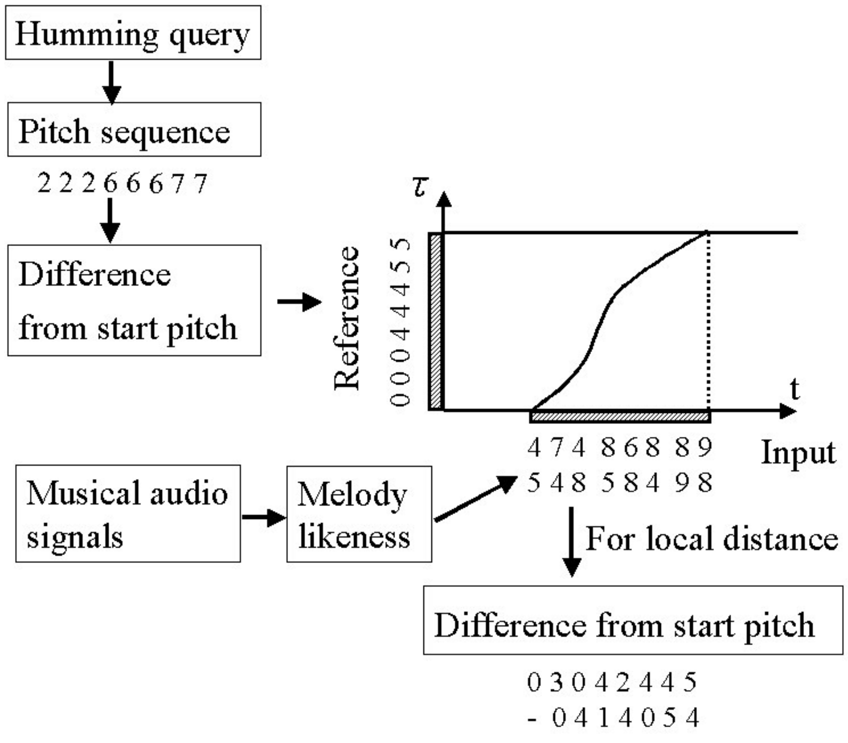


Figure 6. Proposed retrieval method using s-CDP.

4. EXPERIMENTS

4.1 Experimental Method

We prepared 20 WAV files (about 80 minutes in total) in 16 kHz sampling and monaural recording format as a music database to compare s-CDP with mpCDP. This database includes 10 Japanese pop songs, 8 children's songs, an animation song, and a Japanese *enka*. There are 4 male and 16 female vocal artists in the database. Also, 3 males and 2 females sang a portion of each song in the database for about 20 seconds. Hence, the total number of queries was 100.

Let f_b[Hz] (in this experiment: 55[Hz]) denote the lowest frequency of melody. We compute the melody likeness for the frequency $2^{x/12} f_b (x = 1, \cdots, X)$ [Hz] by FFT analysis in consideration of the harmonic structure of acoustic signals. This method is a simple version of [5]. The step of the pitch axis of mpCDP is set at 60 (5 octaves: $X = 60$) considering the male and female voice pitch range.

Two thresholds segmented a reference from a query. To decide the start point, the threshold is set at $0.5\overline{P}$, where $\overline{P}$ is average power (summation of square of signal). For the end point, the threshold is set at $0.1\overline{P}$. The start point threshold is set larger because the start point pitch should be correct.

The search rate, which depends on a threshold $\alpha(0 \le \alpha \le 1)$, is defined as the average of precision rate *NC/ND* and recall rate *NC/NT*, where *NC*, *ND,* and *NT* represent the number of similar terms to the humming query in detected terms, the number of detected terms by mp-CDP, and the number of similar terms to the humming, respectively. The detected term by mp-CDP is correct if the following overlapping rate

$$\text{overlapping rate} = \frac{\text{similar term} \cap \text{detected term}}{\text{similar term} \cup \text{detected term}}$$

is greater than 0.5. This means the overlapping terms between similar and detected ones has 70% intersection when lengths of both terms are mutually identical. Since the search rate depends on threshold α, "the search rate for a humming query" is defined as the maximum value running over all α. The computer for this experiment is OS: Windows2000, CPU: Pentium IV 1.5GHz.

4.2 Results

Table 1 shows average search rates and search times for 5 persons × 20 songs = 100 queries for query duration of 20 seconds. Those results show that s-CDP reduced search time to about 1/40. The mpCDP has 60 times the local path calculation because it has 60 steps in pitch axis. The reduction effect is only about 1/40 because s-CDP calculates the start point and has three times the number of local similarities.

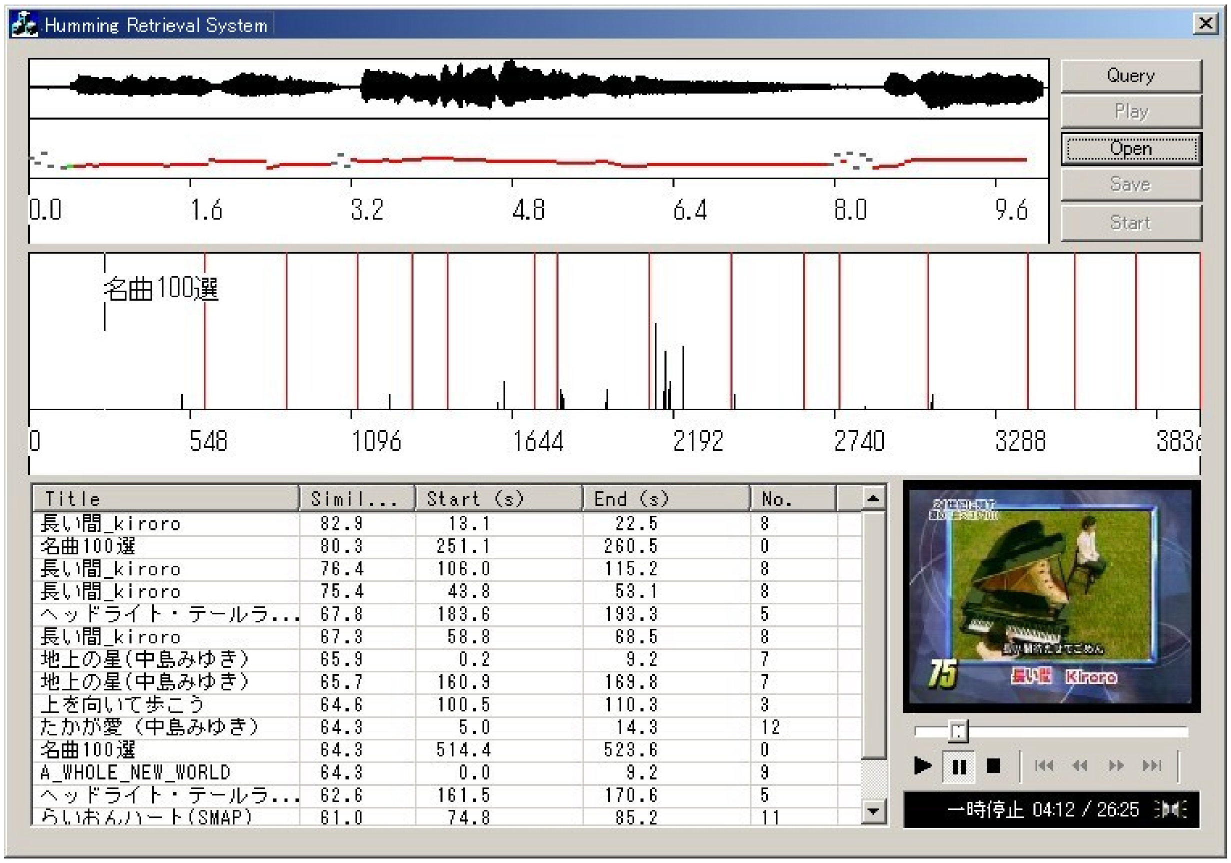
The s-CDP search rate was lower by 16%, though it is more than 70%, showing practical usefulness.

Table 1: Comparison of the two search methods.

Search method	mpCDP	s-CDP
Average search rate	89.0%	73.3%
Search time	532(s)	14.2(s)

5. RETRIEVAL SYSTEM

We made a humming retrieval system as shown in Figure 7. From the top window, the query wave, the query pitch sequence, similarities in the database, and hit results are shown. On clicking the peak on the similarities or the hit results, similar intervals of video or music are played in the right-bottom window. Using a notebook PC with a Pentium III 750MHz CPU, retrieving a 10[s] query from 60[min] database took about 10 seconds. There are several similarity peaks in Figure 7 because the same song was recorded from a TV program and a radio program and also the song had several repeated melodies. Retrieving from an audio video database that captured a scene from karaoke has been successful with this system, which will be demonstrated in presentation.



Title	Simil...	Start (s)	End (s)	No.
長い間_kiroro	82.9	13.1	22.5	8
名曲100選	80.3	251.1	260.5	0
長い間_kiroro	76.4	106.0	115.2	8
長い間_kiroro	75.4	43.8	53.1	8
ヘッドライト・テールラ...	67.8	183.6	193.3	5
長い間_kiroro	67.3	58.8	68.5	8
地上の星(中島みゆき)	65.9	0.2	9.2	7
地上の星(中島みゆき)	65.7	160.9	169.8	7
上を向いて歩こう	64.6	100.5	110.3	3
たかが愛（中島みゆき）	64.3	5.0	14.3	12
名曲100選	64.3	514.4	523.6	0
A_WHOLE_NEW_WORLD	64.3	0.0	9.2	9
ヘッドライト・テールラ...	62.6	161.5	170.6	5
らいおんハート(SMAP)	61.0	74.8	85.2	11

Figure 7. Monitor of the developed retrieval system.

6. SUMMARY

We proposed a start frame feature dependent continuous DP assuming that the start point pitch of the extracted optimal interval is correct. Test results showed that the proposed method reduces computational costs to about 1/40.

One method for further reducing retrieval time is to compress similar intervals in the database because a song usually has a repeated melody.

7. REFERENCES

[1] Kageyama T., Mochizuki K., and Takashima Y.: Melody Retrieval with Humming, ICMC Proc., 1993, 349-351.

[2] Asif Ghias and Logan J.: Query By Humming – Musical Information Retrieval in an Audio Database, ACM Multimedia '95, Electronic Proc., 1995.

[3] Sonoda T., Goto M., and Muraoka Y.: A WWW-based Melody Retrieval System, ICMC'98 Proc., 1998, 349-352.

[4] Kosugi N., Nishihara Y., Sakata T., Yamamuro M., and Kushima K.: A practical Query-by-Humming system for a large music database, ACM Multimedia 2000, 333--342.

[5] Hashiguchi H., Nishimura T., Takita J., Zhang J. X., and Oka R.: Music Signal Spotting Retrieval by Humming Query Using Model Driven Path Continuous Dynamic Programming, SCI2001,

[6] Goto M.: A Predominat-F0 Estimation Method for CD Recordings: MAP Estimation using EM Algorithm for Adaptive Tone Models, Proc. of ICASSP 2001.

[7] Oka R.: Continuous word recognition with Continuous DP (in Japanese), Report of the Acoustic Society of Japan, S78-20, 1978, 145-152.

ISMIR 2001 Chair's Address

Whither Music Information Retrieval: Ten Suggestions to Strengthen the MIR Research Community

J. Stephen Downie

Graduate School of Library and Information Science
University of Illinois at Urbana-Champaign

501 East Daniel St.
Champaign, IL 61820

jdownie@uiuc.edu

ABSTRACT

I intend to use this forum to share with you my personal thoughts and feelings concerning the future of the music information retrieval (MIR) research community. I wish to propose that the MIR community begin, in earnest, to construct more formal and permanent organizational frameworks explicitly designed to maximize the benefits of being a multi-disciplinary and multi-national research community while at the same time minimizing their inherent costs. Throughout this presentation I make suggestions and recommendations that I hope will prompt others to take up the challenge of creating a stronger and more vibrant future for MIR research and development.

1. INTRODUCTION

Music information retrieval research is alive and well. It is a thriving endeavour that is producing interesting, insightful, and intriguing solutions to difficult and non-trivial problems. It is a multi-disciplinary research programme involving researchers from traditional, music and digital libraries, information science, computer science, law, business, engineering, musicology, cognitive psychology and education. It is also a multi-national enterprise. An informal perusal of the MIR literature finds MIR papers and presentations originating from Australia, Austria, Canada, United Kingdom, New Zealand, Ireland, Greece, United States, Japan, Taiwan, Germany, France, Italy, Spain, and Finland.

Multi-disciplinarity both blesses and curses the MIR research community. We are blessed with the wide variety of techniques and technologies that are being brought to bear on MIR problems. We are cursed, however, with its tower-of-babel effect: different researchers speaking different research languages. We are doubly cursed with the scattering of MIR papers across disparate disciplinary literatures making comprehension of the true state-of-the-art difficult.

Multi-nationality similarly has its benefits and its drawbacks. Again, the wealth of international experience in all aspects of MIR research provides us with framework of transnational connections that rival the United Nations. Like the United Nations, however, we must be ever vigilant that the cultural assumptions and practices of one particular group do not overshadow the needs and concerns of the other members. On a more pragmatic level, communications conducted in what is for many a second, or third, language, across several, if not many, time zones only adds to the extra burden of sustaining multinational research and organizational ties.

2. THE FUTURE OF ISMIR GATHERINGS

The success of ISMIR 2000 and ISMIR 2001, along with other recent MIR workshops and panel sessions, clearly illustrates both the desire and need for MIR researchers to come together as a community to discuss matters unique to our research problem. In conversations and correspondence with participants of these events several issues have appeared consistently enough that they require addressing. Find below my suggestions regarding future ISMIR events stemming from these discussions.

Suggestion #1: *Reclassify ISMIR.* Many participants have mentioned that funding bodies and faculty reviewers have clearly defined hierarchies of precedence concerning types of academic gatherings. Many researchers have mentioned that they can more readily obtain funding for a "conference" than for a "symposium". We should, therefore, classify each of our future meetings as a "conference" rather than a "symposium". I believe that we now have the depth and breadth of research to justify this reclassification. The astute reader will have ascertained that the name "ISMIR" will have to be modified to reflect this change. This is certainly is the case, however, I will discuss later a proposal to keep the acronym but still affect the reclassification.

Suggestion #2: *Expand ISMIR.* Our present format of two-and-one-half days is clearly insufficient to meet the communication needs of our participants. Future iterations must be longer (four days?) to allow for meaningful interactions at a formal level (i.e., organizational meetings; specialized panels) and at an informal level (i.e., impromptu discussions; social events). Participants travel great distances to attend and it makes sense to get the maximum "bang" out their travel investments.

Suggestion #3: *Hold a TREC-like panel every conference.* One specialized panel that I would like to see made regular part of the conference would be one structured along the lines of the Text REtrieval Conference (TREC) [12] [6] [7]. Each year a ***different*** predefined set of collections, queries, responses and metrics would be put forward for the purposes of comparing the efficacy of different approaches. Note here my deliberate use of "different". Because there are so many types of tasks that we are proposing for our MIR systems, we should focus our attention on one specific task each year. This will bring coherence to the necessary compare-and-contrast aspects of the evaluations. For example, one year we might focus upon known-item identification; another year we could turn our attention to user interface issues, and so on.

Suggestion #4: *Hold introductory workshops*. As part of an expanded conference setup, it is imperative that we begin to hold discipline-specific workshops as part of the programme of events. Ideally, these would not be held concurrently on some special "workshop day" as is the practice at some conferences, but on consecutive days to allow participants the opportunity to attend as many as desired. These introductory workshops will help us minimize some of problems associated with our multi-disciplinarity by exposing all to the basic principles, methods, and technologies of the various disciplines. For example, I envision workshops with such titles as "Basics of Information Retrieval Evaluation", "Introduction to Digital Signal Processing", "Music Encoding 101", and so on.

Suggestion #5: *Consistently rotate the conference location*. We must formally adopt as a first principle our commitment to fostering and nurturing our multi-national character. To this end, we must establish a governing principle that our conferences shall rotate through the major regions currently conducting MIR research: North America, Europe, Asia and Australasia. As other regions become more active, they too should be added to the rotation list.

Suggestion #6: *Keep the current cost model*. Two things are points of pride with me concerning my involvement with ISMIR 2001. First, we presented a first-rate symposium while keeping the fees to a very reasonable US$150. Even at this modest level, issues of international currency exchange rates and limited home-base resources made the fee problematic for some. We must strive to keep our fee structure as low as humanly possible. We should also set up resources to support colleagues with more limited funding opportunities and/or weaker currencies.

Second, we were able to secure principal funding for student stipends from the National Science Foundation. Additional stipend support was provided by the National Center for Supercomputing Applications. This funding allowed us to waive fees, provide accommodations, and/or pay travel expenses for approximately one dozen student researchers. MIR research and development will wither and die without an ongoing influx of younger researchers. Conference attendance is expensive and many departments, especially those in the humanities, simply do not have the resources to support student travel. We must do everything possible to ensure that we can continue to offer financial support to our future colleagues regardless of their discipline or country of origin.

3. THE MIR BIBLIOGRAPHY PROJECT

The MIR Annotated Bibliography Project outlined in Downie [8] should be seen an integral part of the MIR community's future. Initial work on the project has two principal components:

a) the creation of a MIR-specific bibliography designed to counter the scattering of the MIR literature across disciplinary boundaries; and,

b) the creation of ancillary, discipline-specific, bibliographies designed to provide basic background materials necessary to acquaint MIR researchers with the fundamental principles and methods of all the contributing disciplines.

Together, these two components could go a long way in helping us bridge the disciplinary divides that so hinder our effective communications. The domain name ***music-ir.org*** has been acquired, under which access to the bibliography will be afforded.

Suggestion #7: *Embrace the bibliography project*. Primary funding for the bibliography project has been generously provided by the Andrew W. Mellon Foundation [2]. Notwithstanding the ever-present need for ongoing financial support for the bibliography project, I believe it to be more important that we as a community take it upon ourselves to put a little "sweat equity" into this and similar resources. We have been designing into our project the ability to distribute the data entry and collection tasks. By distributing these tasks, we hope to improve the sustainability of the resource. For example, we are implementing an end-user data entry system that we hope will guide users in the creation of well-structured bibliographic records that will require a minimum amount of editorial intervention. I suggest that we all make it a habit to submit a record of our publications as soon as we have them published. Similarly, as part of embracing the bibliography project as a community resource, I would like encourage all to contribute suggestions and recommendations concerning its development. More specifically, consider this a formal solicitation for your input on the creation of the discipline-specific supplementary and explanatory materials.

Suggestion #8: *Consider* ***music-ir.org*** *as our home-base*. Over the next several years, I would like to see ***music-ir.org*** become the principal starting point for all things related to MIR. I would like us to see the current MIR bibliography project as only one seed in a potentially fruitful garden. We need a central repository for our collective work whether it be in the form of test collections, papers, theses, reports, discussions, presentations, membership contact lists, software, and so on. One thing that will help us to expand our horizons from the bibliography project to a more comprehensive WWW portal is our decision to use the *Greenstone Digital Library Software* (GSDL) [13] as the system's principal retrieval mechanism. The GSDL software will enable us to bridge the gap between bibliographic retrieval to full-fledged digital library/portal with a minimum of bother. Again, a combination of "sweat equity" and financial resources can make this a reality.

4. FORMAL ORGANIZATION

Of all my suggestions and recommendations, I believe the most controversial will be those concerning how we as a research community constitute ourselves. We have several options before us, including the option to ignore the issue altogether. I no longer see ignoring the question as a reasonable option. If we are going to continue to hold ISMIR conferences and create resources like the proposed portal then the time has arrived that we address the issues of what we are and how we are going make things happen.

In discussion with various members of the MIR community two contending options seem the most prevalent:

Option #1: We form an independent organization specifically to foster MIR research and development.

Option #2: We seek membership as a special interest group (SIG) under the auspices of such organizations as:

a) the Association for Computing Machinery [5], perhaps as part of their SIG Information Retrieval (SIGIR) [4] or SIG Multimedia (SIGMM) [3];

b) the Institute of Electrical and Electronic Engineers (IEEE) [9]

c) the American Society for Information Science and Technology (ASIS&T) [1]; or,

d) the International Computer Music Association (ICMA) [11], who are the convenors of the annual International Computer Music Conference (ICMC) [10].

Suggestion #9: *Let us reject Option #2*. There several attractive features to Option #2 including pre-existing infrastructures, name recognition, and access to possible financial and technological resources. These are, however, outweighed by three serious drawbacks. Let me address these in turn:

a) **Loss of identity**: By choosing to affiliate ourselves with one organization over another we are in essence affirming that our work is best conceived as being a subset of that particular organization's rubric. For example, if IEEE is our choice, then we are saying that MIR is an engineering problem. If we choose SIGMM then we are saying that MIR is really only a multimedia retrieval problem (as it is conceived by the current members of SIGMM). In reality, MIR is broader than either and it is in our best interests to ensure that we do not arbitrarily narrow our research focus and agenda.

b) **Weakening of our multi-disciplinary strengths**. We must remember that each of these groups has its own paradigm of accepted practice that privileges some research questions and methods over others. I would hate to see our intellectual toolkit deprived of any of the research approaches that we have seen presented at our recent meetings simply because it was expedient to join a particular organization. I wonder—absent any real data, of course—how many musicologists and music librarians we might attract over time after subjugating ourselves to IEEE or ACM.

c) **Hidden costs to members**. As an organization it might be more financially sound to affiliate ourselves with one of these organizations. I suspect, however, that this would represent a false economy to the MIR community. I recently paid, for example, $US395 in fees to attend the joint ACM/IEEE digital libraries conference. I am similarly concerned that access to our publications and proceedings could be caught up in their respective proprietary, fee-based, digital libraries.

Suggestion #10: *Let us adopt Option #1.* Given the aforementioned drawbacks are rather serious, I suggest that we strike out on our own. I see six reasons to support this idea:

a) **Build upon ISMIR**: ISMIR has begun to have considerable name recognition as being ***the*** forum for MIR research. We can exploit the acronym by forming the ***International Society for Music Information Retrieval*** (i.e., ISMIR). It would be under the auspices of this organization that our expanded conferences would be held. This would also allow us, for example, to refer to our meetings as ISMIR 2002, and so on. The "Society" would be responsible for shaping and upkeep of whatever develops at the ***music-ir.org*** site which is fitting because the site is fundamentally an outcome of ISMIR 2000.

b) **We define the research paradigms**. The recurring theme throughout my discussion has been the important role multi-disciplinarity plays in enriching our endeavours. Under on independent ISMIR we can reinforce this strength both formally through our statement of principles and informally by creating the mechanisms the ensure all disciplinary voices have the opportunity to express their respective world views.

c) **We define the priorities**. We have problems unique to MIR research and development that I believe are best addressed by ourselves. These include, for example, the creation of multi-representational test collections, tasks and MIR-specific evaluation metrics. Effective solicitation of support from the music recording and publication industry is another priority I believe to be unique to MIR.

d) **We control our resources**. Rather than pay substantial membership and conference fees to a larger organization, we will be able to tailor our finances to suit our needs. This includes deliberately holding down conference costs, both in terms of registration fees and in accommodations by taking advantage of the flexibility inherent in a smaller, independent organization. This alone should help us continue to attract international scholars and students. Our bibliography and portal can remain under our control and be shaped as we see fit. Finally, as we mature, we will want to establish our own peer-reviewed e-journal exclusively devoted to MIR issues. Under the auspices of ISMIR, we will be able to set and control the editorial rules governing the e-journal. We will also be able to provide *true* peer-review since the reviewers will be drawn from our MIR peers. Since the envisaged e-journal will exploit resources already under the control of ISMIR, there is added benefit that access to the e-journal could be provided at no cost to readers.

e) **We can affiliate on more equal terms**. Nothing in the suggestion to form ISMIR precludes our affiliation with other related organizations. In fact, it is very important that we do strengthen our ties with outside groups. The advantage we would have under an independence model is that we would be empowered to negotiate the specific levels of interaction that we think would best suit us on a case-by-case basis. This might include a special joint conference or our sponsorship of an MIR panel session at their meeting in exchange for some of their members providing us with special workshops, and so on.

f) **We still have the Option #2**. It would be easier to move from independence to affiliation, than from affiliation to independence. Disentangling ourselves from a larger organization, should we become dissatisfied with our arrangement, could be next to impossible as we would have to negotiate such contentious issues as intellectual property rights and finances. This suggests that the prudent course of action would be to first give independence a try for some reasonable amount of time. If the membership is unhappy with the results, then we can move to become affiliated with one of the larger organizations.

5. CLOSING COMMENTS

MIR research and development is fast approaching a crossroads in its evolution. The time is upon us to make some important decisions as to which direction we would like to take as a research community. I would like to iterate one last time that our true strengths are rooted in our wide variety of multi-disciplinary and multi-national world views. Based upon these strengths, I have offered ten suggestions that I believe will help foster the continued growth and vitality of our endeavours. The MIR community may accept or reject my suggestions as it sees fit, but I hope that the alternatives that it adopts are predicated upon preserving our multi-disciplinary and multi-national character.

6. ACKNOWLEDGMENTS

Dr. Suzanne Lodato is thanked for helping the MIR Annotated Bibliography Website project obtain its principal financial support from the Andrew W. Mellon Foundation. Dr. Stephen Griffin must be thanked for helping us secure the NSF funding that made both ISMR 2000 and 2001 a reality. I would also like to thank Dr. Radha Nandkumar and the National Center for Supercomputing Applications Faculty Fellows Programme for their financial support. Those members of the MIR research community who have volunteered to act as project advisors to the MIR bibliography are also thanked. Finally, I must thank all those on the organizing committees of ISMIR 2000 and 2001 for their selfless efforts in making these important events thrive.

7. REFERENCES

[1] American Society for Information Science and Technology. 2001. *American Society for Information Science and Technology homepage.* Available at: http://www.asis.org .

[2] Andrew. W. Mellon Foundation. *The Andrew W. Mellon Foundation homepage.* Available at: http://www.mellon.org .

[3] Association for Computing Machinery. 2001. *SIG Multimedia.* Available at: http://www.acm.org/sigmm

[4] Association for Computing Machinery. 2001. *SIGIR: Special Interest Group on Information Retrieval*. Available at: http://www.acm.org/sigir/

[5] Association for Computing Machinery. 2001. *The first society in computing.* Available at: http://www.acm.org .

[6] Crawford, T, and Byrd, D. (2000). Background document for ISMIR 2000 on music information retrieval evaluation. In D. Byrd & J. S. Downie (Chairs), *Proceedings of the First Annual International Symposium on Music Information Retrieval (ISMIR 2000).* Available at: http://ciir.cs.umass.edu/music2000/evaluation.html

[7] Downie, J. Stephen. 2000. Thinking about formal MIR system evaluation: Some prompting thoughts. In D. Byrd & J. S. Downie (Chairs), *Proceedings of the First Annual International Symposium on Music Information Retrieval (ISMIR 2000).* Available at: http://alexia.lis.uiuc.edu/~jdownie/mir_papers/downie_mir_eval.html

[8] Downie, J. Stephen. 2001. Music Information Retrieval Annotated Bibliography Website Project, Phase I. In J. S. Downie and D. Bainbridge (Chairs), *Proceedings of the Second Annual International Sysmposium on Music Information Retrieval (ISMIR 2001).*

[9] Institute of Electrical and Electronic Engineers. 2001. *The IEEE.* Available at: http://www.ieee.org .

[10] International Computer Music Association. 2001. *ICMC.* Available at: http://www.computermusic.org/icmc/ .

[11] International Computer Music Association. 2001. *International Computer Music Association.* Available at: http://www.computermusic.org .

[12] National Institute of Standards and Technology. 2001. *Text REtrieval Conference (TREC) homepage.* Available at: http://trec.nist.gov.

[13] New Zealand Digital Library Project. 2001. *About the Greenstone software.* Available at: http://www.nzdl.org/cgi-bin/library?a=p&p=gsdl .