University of Southampton
Faculty of Engineering and Applied Science
Department of Electronics and Computer Science
Southampton SO17 1BJ

Content Based Retrieval and Navigation of Music

by

Steven G. Blackburn

A Mini-Thesis submitted for transfer
of registration from MPhil to Ph.D.
10 March 1999

Supervisor: Dr D. DeRoure

Contents

1 Introduction

From talking with creators of multimedia documents, it appears that audio is generally considered as an ‘endpoint’ medium, where there is no way out of the media. Audio and video is used to illustrate points, rather than make them. It would be nice to use the audio as the document rather than as illustrations. There are several problems with this approach, not least the space requirement and that written text is often easier to absorb than the audio equivalent. This author feels that the latter stems from the lack of use of the media; that authors on multimedia documents have not been given the option of using audio as the basis of a document so have never had to look into the problems associated with it.

Current work on audio generally consists of systems which classify short digital audio samples (Foote [12], Wold *et al* [38]), with the aim of helping retrieval. Classification and retrieval of music has also been examined (Pikrakis *et al* [30], Ghias *et al* [15], McNab *et al* [25]).

Little attention has been paid to the use of audio as a linking medium. Temporal linking has been considered in the Soundviewer [16] for Microcosm [13] and is beginning to become more mainstream through the development of standards associated with the world wide web, such as SMIL [34]. There is even less literature on links based on content of audio documents, MAVIS [21][22] being an exception.

This mini-thesis will present techniques which enable the use of multimedia, specifically audio and music, as the basis of a document. A review of the related work is given in Section 2, followed in Section 3 by an explanation of the author’s work so far. Some possible directions for further research are highlighted in Section 4 while Section 5 presents the conclusions, suggested research direction and a plan of work.

2 Literature Review

This section discusses research related to the topic of this thesis. It starts with an explanation of the term ‘navigation’ and what it means to navigate audio. Music representations are then considered before examining content based retrieval and navigation. A discussion of open hypermedia, and other issues relating to audio, follows.

2.1 Navigation

Anyone who uses the Internet is familiar with navigating around a body of documents. Links are created between documents by embedding information, at relevant points in the source document, which indicates related material. The Hyper Text Mark-up Language (HTML) on the world wide web is the, now classic, example of this model.

There are a couple of problems with the use of embedded links. The first being that while it is possible to link *to* any media, the source document is usually text (although image maps are quite common). It is currently not possible to be listening to a piece of audio and link to a web document. This issue is gradually being addressed through standards associated with the web, such as the Synchronised Multimedia Integration Language (SMIL) [34] and associated editors [4].

The second problem is that all instances of conceptual links must be manually implemented. For example, if one wished to link every occurrence of the phrase “University of Southampton” to the home page of the University, links must be implemented for each occurrence of the phrase. Also, if the location of the University page changes, all instances of the links must be found and changed.

2.2 Navigation of audio

The ability to link to specified parts of audio files, and to link out from audio files (for example to the current location in a text transcript of a speech) is not very common but nonetheless desirable. The simplest form of this is temporal navigation.

Temporal navigation is where links are based on the start time and length of a selection in a media document, such as audio of video. A simple database query can then be used identify the appropriate links.

The Microcosm Soundviewer [16] was an audio tool which supported hypermedia links into and out of stored audio via this mechanism. Using the Microcosm open hypermedia system [6][13], links could be authored based on the position of a selection in a document.

An audio only environment, Espace 2, was developed by Sawhney and Murphy [32]. It was a prototype system for navigating hyper-linked audio documents. Continuous audio streams are used to represent the availability of links to other documents. The example content used recordings of conversations and discussions. It is likely that the links were time based, due to the concentration of the project on the user interface rather than content extraction.

Buchanan and Zellweger [3] examined ways of specifying temporal behaviour in hypermedia documents. The Firefly document system provided support for synchronisation of media clips in a presentation similar to that provided by SMIL (allowing clips to occur in sequence or in parallel). Again, links between clips are time based, although the timing is only resolved when the document is displayed (as opposed to being an authoring process).

This approach does have the advantage that it will work with any temporal media, such as video, as the content is not considered. However, it suffers from one of the problems of embedded links: each link must be manually authored. To overcome this, links need to be based on the content of a selection in the document.

2.3 Content representations

How the content is represented is obviously important as it has an effect on the matching algorithms used for linking purposes. Text is a suitable content representation for a recorded speech. Some appropriate representations for music, and their suitability for comparing / matching music, are discussed here. Most of these can be obtained by performing feature extraction on another, usually lower level, representation. As such, feature extraction can be thought of as representation conversion, taking a low level representation and identifying higher level features.

Wiggins *et al* [37] described a framework for evaluating music representation systems. They considered a representation to be useful for one, or more, of three purposes: recording, analysis, or composition. They examined the expressive completeness (the range of musical data which may be represented) and the structural generality (the range of musical structure) of each of the representations. Although scores were never assigned for any of the representations they looked at, these concepts are useful when discussing suitability for a purpose.

The representation used for content based retrieval and navigation systems may depend on where the query comes from. For example, a representation which allows quick, exact, matching might be used when linking from selections in MIDI files, as a precise query can be expected.

2.3.1 The musical score

The musical score is a representation used by a composer to tell a musician how to play a particular piece of music. As such it is the definitive document which conveys all the notational nuances which are lost as soon as the piece is performed. It is also likely to contain some structural information which is also lost in performance. There are many existing file formats which may be used to store musical scores, such as the standard music description language (SMDL) [33]. The expressive completeness and structural generality is high.

2.3.2 Performance data

The Musical Instrument Digital Interface (MIDI) is a standard which allows digital instruments to transmit performance data between each other. This performance data may be stored in MIDI files which consist of a collection of tracks specifying for each instrument what to play and when to play it.

Due to the performance being a musician’s interpretation of a musical score, one could argue that more information is contained in the MIDI. However, MIDI can also be derived from the score and, due to the linear nature of the recording, a lot of structure and markup is lost.

2.3.3 Musical pitch contours

Music is recognisable irrespective of the key it is played in. It is clear that matching on the exact pitch of each note is not desirable. Dowling [11] suggested that the pitch direction, noting whether the pitch went up, down or was repeated, is useful in some situations. This representation is referred to as a musical pitch contour. Lindsay [23] discussed how pitch contours are not a new idea, indeed they were used as a form of musical score for chants.

A musical pitch contour describes a series of relative pitch transitions, an abstraction of a sequence of notes. A note in a piece of music is compared with a previous note and then classified in one of three ways: either the pitch went up (U); the pitch went down (D); or it is a repetition (R). Thus, the piece can be converted into a string with a three letter alphabet (U, D, R). For example, the introductory theme to Beethoven's 5th Symphony would be converted into the sequence: - R R D U R R D. Notice that there is one less symbol than notes as only the transitions between notes are recorded. An example of a musical score and its associated pitch contour is shown in **Figure 1**.

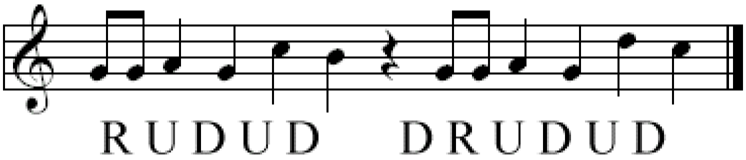


Figure 1 - A musical pitch contour for "Happy Birthday"

Musical pitch contours discard a lot of information which might otherwise reduce the search space, such as rhythm information and pitch intervals. This results in the representation having a certain amount approximate matching built in. If the notes in two pieces of music rise and fall in the same order then they are considered as matching, regardless of the amount of change in pitch.

2.3.4 Alternative representations

Although the pitch contour is suitable for applications where approximate matching must be used, it is not sufficiently specific for other applications. So, other representations must be considered. McNab *et al* [38] used a combination of exact pitch intervals and rhythm, in addition to pitch contours, in their trial. Lemström and Laine [20] used a representation consisting of a 12-tuple for each note, although their application only used relative pitch and rhythm information for matching.

2.3.5 The Fourier transform

The Fourier transform relies on the principle that any waveform can be described as the sum of component sine waves, each at a particular frequency and amplitude. The Fast Fourier Transform (FFT) is an efficient algorithm which approximates the Fourier transform. Given a digital audio file a table, which identifies the amplitude of each component frequency at any moment in time, may be calculated.

This is very close to the digital audio representation so it can be used to express almost anything. However, it is completely lacking in musical structure.

2.4 Content based retrieval

Content based retrieval is important to content based navigation because the issue of content representation has to be addressed. Techniques for indexing and approximate matching are also prevalent in this field. As such, before considering content based navigation, is it useful to investigate content based retrieval systems. Retrieval has an additional use as a means of locating an initial media document before the document space can be navigated.

2.4.1 Retrieval of digital audio samples

The term "digital audio sample" is used here to refer to short, descriptive, sounds such as a bird chirping, a bell ringing or a car starting. Foote [12] and Wold *et al* [38] described systems which could retrieve such samples. The approach is generally to use the Fourier transform of a sample then extract some salient features from it and perform an approximate match against a database of pre-processed samples.

The Fourier transform works well for this application as the samples are short (making comprehensive indexing possible) and distinctive. The matches returned are likely to be very general, returning the sound of all cars starting and not just the sound Ford cars make when starting.

2.4.2 Query by humming

Query by humming is possibly the definitive application of content based retrieval of music. In such a system acoustic input, typically sung or hummed by the user, is transcribed and then music is retrieved for a database. Results are ranked according to how closely the songs match the input. Allowance must be made for inaccurate singing or errors in the tune recall of the user.

McNab *et al* [25] developed such a system which searched a database of folk songs. Transcription was performed by a system called MT (Melody Transcription) [26], which accounted for the user’s gradually changing tuning in an adaptive mode. Searches were carried out using a choice of: an exact match of pitch; pitch and rhythm; pitch contour or contour and rhythm. Approximate matches were possible for pitch and rhythm or contour and rhythm.

A large database of 9600 folk tunes was used to test the system and, using exact matching of rhythm and pitch, identified tunes with very few mismatches. However, the system only matched tunes from the beginning. The task of searching was further simplified due to the folk songs being single track and monophonic (only one note playing at any given time).

A complex system for transcribing acoustic input was employed. This appears to result in an unnecessary processing overhead when not matching using exact pitch intervals. The exact note sung is not important when using the contour, yet time was taken processing it. However, no indication of the speed of the system was given. They noted that a major issue will be the development of approximate string matching algorithms which avoid computational explosion as the size of the database increases.

Ghias *et al* [15] also utilised pitch contours, but with ‘S’ representing repetition (or Same). The test database comprised a collection of all the parts (melody and otherwise) from 183 songs. They noted that 90% of these songs were retrieved with 10-12 pitch transitions. If the list of results was too large, the user could perform a new query on a restricted search list consisting of those songs just retrieved.

While the system allowed searching on any part of a song, and MIDI files are both polyphonic and multi track, it did have some problems. Again the pitch tracking algorithm, using Matlab, was very complex and was the slowest part of the system. Searching the database would slow considerably as more songs are added and the sample database was too small to base any statistical analysis on. There was also a lot of noise in the database, as it contained a lot of information unrelated to the melody.

2.4.3 Lemström and Laine

Lemström and Laine [20] ignored the pitch tracking to focus the retrieval process. They presented a representation for musical data, the inner representation. A two dimensional relative code, derived from the inner representation, was used for the matching as it was independent of the original key. An efficient indexing structure, the suffix-trie, was used in the matching phase.

The music is indexed using relative pitch changes and duration between the starts of notes (the inter onset interval). The inner representation (IRP) file appears very similar to MIDI, having tracks and events. It consists of 17 tracks, the first containing the lyrics while the rest contain instrumental events. Each event is a 12-tuple, giving information such as: the start time and duration of the note; pitch; symbolic representation; interval from previous note and chord information.

Strictly defining the number of tracks to be 17 was a strange decision. This author presumes this to be related to MIDI only supporting 16 channels, but a MIDI channel is a very different concept to a MIDI track. Storing both the exact pitch of the notes as well as the pitch intervals is also a little perplexing as this information is easily derived at run time with little performance hit (probably no more than reading the 12-tuple itself). The most confusing thing is that the IRP was not referred to again in the rest of the paper, preferring to use their two-dimensional relative encoding.

Two-dimensional relative encoding is a representation which consists of a pair of components, relative pitch and duration, for each note. The method for matching the melody was dependent on where the query came from. Musical pitch contours were only used when large errors were expected in the intervals. When a more reliable query was given, quantized partially overlapping intervals (QPI) were used. QPI builds on the idea of defining pitch directions as being small, medium and large. The categories overlap in an attempt to make the retrieval process more tolerant of errors but appears to simply result in an eleven character alphabet.

The indexing structure used was a suffix-trie, a structure developed for string pattern matching. This is a tree like structure which stores each suffix of a word, e.g. “Hello” would be stored as “Hello”, “ello”, “llo”, “lo” and “o”. It is very fast for exact matching ($O(m)$ where m is the length of the query) but the space requirement is not good; it grows quadratically with the size of the database. To reduce the space requirement, only the top part of the tree, up to a certain depth, was stored. They did not specify the depth used.

Approximate matching was limited to errors in relative pitch, something which is not an issue with gross (musical) pitch contours. However, it is possible to implement an improved matching algorithm which would allow for insertion and deletion of symbols. No timing information was given, so it is difficult to determine the suitability of this approach for an interactive system. However, due to the comprehensive indexing, one would imagine it to be superior to the systems developed by Ghias *et al* and McNab *et al*.

The requirements for the lengths of the query keys for each representation were considered using a database of 154 pieces of music. The source of these is not clear, referring to them as "automatic compositions of Pauli Laine, folk-songs, etc." They showed that the QPI representation returns optimal results, where optimal is the response given by exact intervals, with shorter keys than gross pitch contours (7 characters instead of 10 characters).

2.4.4 Image matching techniques applied to audio

Some aspects of image matching are being applied to audio classification / retrieval. This is useful for analysing digital input, such as from a CD, as there is currently no method of converting multi-timbral, polyphonic, music into a musical score. This prevents the previous methods being applied, although we would still like to be able to index and search the music in a similar manner. See Martin [24] for an example of the ongoing research into polyphonic pitch tracking.

Image recognition would allow a user to sketch a picture which would then be analysed and compared with images in the database. The images are indexed by identifying features in them, such as lines and circles and their relationship with each other. If a picture could be created which described a piece of music then features of that picture could be extracted and used to index the music.

This can be achieved by using a Fast Fourier Transform. The frequency / amplitude table can be plotted on a 2-dimensional graph, time against frequency, where the amplitude is represented by use of a colour scale. An example of the reuslting picture is shown in **Figure 2**. By applying image matching techniques, this representation can then be used for indexing. This is a useful technique as it can index any form of digital audio although it tends to be used for classifying music. Pikrakis *et al* [30] reported on one such classification project tries to identify forms of Greek traditional music. Fushikada *et al* [14] used an FFT plot in conjunction with video to improve content based video scene retrieval.

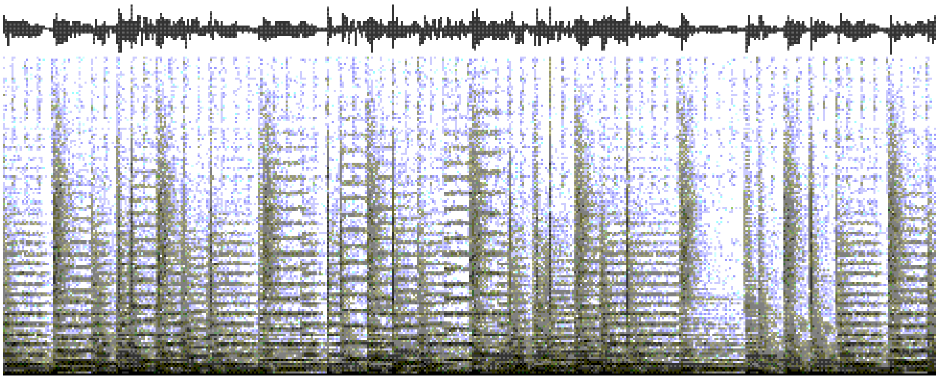


Figure 2 - An FFT plotted on a 2-dimensional graph

2.5 Open hypermedia

Many hypermedia models have been developed. Some of these, such as the Amsterdam hypermedia model [19], specifically consider audio. One which is relevant to content based navigation, and so considered here, is the open hypermedia model [17], as used by Microcosm [6][13] and the DLS [5][10].

2.5.1 Key aspects of the model

There are two key aspects to the open hypermedia model. Firstly, information about links between documents is stored separately from the documents themselves (in contrast to common practice with HTML documents). This information is maintained in link databases (linkbases) and, by selecting different linkbases, the user obtains different views of the documents. Access to linkbases may be abstracted into a link service.

Secondly, when the user requests available links, the linkbases are queried with either the location in the document or with the content at that location in the document. This means it is always possible to link from somewhere in a document. Linking from content in this way is an example of a *generic* link.

Using an open hypermedia system, it is possible to author a link from a phrase to another document in one step. Most existing systems only allow linking from text documents, very few have support for audio and none allow linking from musical constructs.

2.5.2 A standard communication language

The Open Hypermedia Systems Working Group (OHSWG) are developing the Open

Hypermedia Protocol (OHP) [7]. This should provide a standard data model and socket protocol for the communication of link information between open hypermedia systems. Support for temporal media is being addressed as suitable scenarios are developed, although there are unresolved issues as to where content extraction should occur.

2.6 Content based navigation

The Multimedia Architecture for Video, Image and Sound (MAVIS) project [21][22] was an open hypermedia system which allows links to be based on content from pictures, sounds and video clips. Although primarily concerned with image content, it did have support of audio for proof of concept. The features used to represent audio were FFT's. These could be approximately matched to find related audio files.

The application of FFT's is of limited use, as matches using lower level representations are more prone to return irrelevant documents. This is due to the type of data provided being unsuitable for direct matching. For example, a short piece of music played on two different instruments, perhaps an oboe and a flute, will have quite different frequency spectra yet might be expected to match.

Fourier transforms are better suited for instances where the timbre, the relationships of harmonic frequencies (put simply), is important and enough to distinguish between the different samples.

2.7 Other issues for audio

There are other issues which do not fit under a heading are covered here. For example, Ossenbruggen [29] discusses the use of audio hypermedia in object oriented wrappers to existing software.

2.7.1 Melody extraction

The ability to identify the role of each instrumental line in a polyphonic piece of music is very useful. The accuracy of content based retrieval systems could be greatly improved if only the music used for queries were stored. This is typically the main melody line, so identifying this is of particular interest.

Uitdenbogerd and Zobel [35] tried several simple algorithms for producing a monophonic melody from a polyphonic MIDI file. A small user trial suggested that always using the top note gave acceptable results for many cases. However, this approach is flawed if the melody is not the highest pitch being played.

2.7.2 Retaining content

Much of the information which content based systems have to work hard to extract was probably initially available when the document was authored. Having some method of keeping this information would improve the quality of content based applications. MPEG-7 hopes to provide a standard way of doing this. There are currently no papers on the subject but, according to the official web site [28], it will:

> "be a standardized description of various types of multimedia information. This description will be associated with the content itself, to allow fast and efficient searching for material that is of interest to the user. MPEG-7 is formally called 'Multimedia Content Description Interface'."

It appears to be a language with which to specify the manner in which content may be described, like SGML may be used to specify HTML. While this may resolve the issue of extracting content from new documents, there will always be the case of historical archives.

2.8 Conclusions

It can be seen that content based navigation of audio has not been covered in great detail, with only MAVIS supplying audio as a proof of concept rather than an aim. The next most important research is the similarity matching of music used in content based retrieval, as a CBN system will have to compare content at some point.

Both of the systems developed by McNab *et al* and Ghias *et al* have characteristics which are not well suited to the intended application in content based navigation. The former only indexes the first part of each piece, while all alignments need to be identified, and the scalability of the latter need to be improved to ensure response times which are within bounds for an interactive content based navigation system.

Ghias *et al* suggested that three-way discrimination of pitch might be useful for finding a particular song among a private music collection, but that higher resolutions will probably be necessary for larger databases. Lemström used suffix-tries and effectively resolves music to 11 classes. The indexing engine used suffix-tries which gave a fast response time but the size of the database would not scale at all well.

3 Experimentation

This section describes the experiments which were conducted during the investigation of audio hypermedia. A content based retrieval engine was developed together with a temporal navigation system. The navigation was extended to allow navigation based on the document content using concepts developed for the retrieval engine.

3.1 A tool for content based retrieval

Due to the close relation between content based retrieval (CBR) and content based navigation (CBN), and considering the inadequacies of existing CBR systems, it seemed sensible to try and develop a CBR engine. The engine was to provide the facility to retrieve MIDI files using a musical pitch contour as the query while overcoming the problems of speed and scalability.

The MIDI file format is a useful abstraction of musical performance: it can be closely associated with a digital audio representation via time-stamps and position pointers. Digital audio can be converted to MIDI by pitch tracking, for which there are good monophonic solutions. Pitch contours are then constructed from MIDI.

The pitch contour for a track is long, an average of 310 pitch directions per minute in the test material. To allow an efficient database lookup technique to be employed, each track is stored as a set of overlapping sub-contours; sub-contours are the atomic units of the database structure. The length of a sub-contour is referred to as the atomic length, *L*. To give an idea of a typical value, $L = 12$ in the database initially used to test the system. This is an important constant in the database structure as it defines the number of distinct atomic units that may be held in the database (the atomic resolution).

atomic resolution = 3^{L}

Occurrences of a sub-contour in the database are located by converting the contour into a base 3 number (where U=0, D=1 and R=2). This gives the relevant index in an array held at the beginning of the database file. Each element of the array points to byte locations further in the file, identifying the start of the linked list of matches (see **Figure 3**). If the element contains the value zero then there were no matches for that sub-contour.

Each node in the linked list contains: the byte offset in the file to the next node; the id of the file the node relates to; and the number of times this contour occurs in the file. The file id is a unique number for which the corresponding file name can be found by using an associated name database. There is only one node stored per sub-contour per MIDI file which, incidentally, results in a compression effect as this usually refers to more than one 'hit'. It should be noted that the sub-contour is not actually stored in the database, but rather implied by the position of the related data in the lookup table at the start of the file.

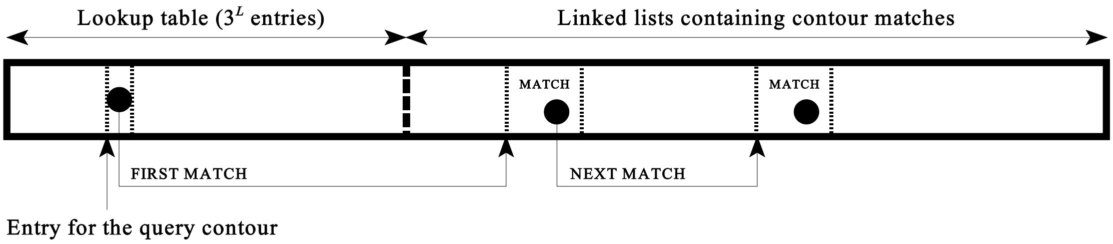

Lookup table (3^L entries)
Linked lists containing contour matches
MATCH
MATCH
FIRST MATCH
NEXT MATCH
Entry for the query contour

Figure 3 - A diagram showing the operation of the database

Using this database structure allows one to determine if there are any matches for a given query in constant time. Each subsequent access retrieves one match. However, this is a *lookup* technique and only works with exact matches. Performing an approximate match, using the database structure alone, would require the entire database to be searched. Obviously this would not scale to large databases. The notion of approximate queries, as opposed to approximate matches, is used to solve this. Possible errors in the query are emulated and the database searched for the resultant list of near matches.

To match a query, the set of contours, which are within a user defined distance of the query, is produced. The distance is given by a string metric which quantifies the minimum cost of transforming one string into another. Cost weights may be assigned to the individual editing operations involved in such transformations, namely symbol substitution, insertion, and deletion.

The metric used may depend on the source of the query, which might be extracted from a MIDI file or from a digital audio file with monophonic or polyphonic content, and it may come from original source material or via the tune recall skills of the user. This system employs the Levenshtein distance, where each transformation has a cost weighting of 1, but in each case certain types of error may be more common so this weighting may not be the most appropriate. For query by humming, more research into tune recall is needed to identify suitable weights. It may be possible to automatically establish appropriate weights for the metric by learning from a reference set of data [31].

A brute force approach to collating the near match set would iterate sequentially through all contours. It is possible to implement a similar algorithm using a tree structure to visit each contour.

Consider a tertiary tree which has a depth equal to the atomic length, *L*. Each node contains a contour. The contour at the root node is empty. The tree branches three ways at each node, appending a pitch direction to the contour at each level. The set of leaf nodes contain the set of all possible contours (see **Figure 4**).

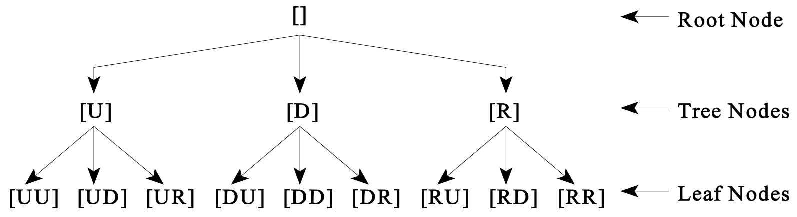


Figure 4 - The complete near match contour tree, where *L* = 2

A simple tree search would traverse the entire tree, comparing the contour at each leaf node with the query. The algorithm can be refined by comparing the contour at each node with a prefix of the query contour. If the distance limit is exceeded then there is no point in traversing the rest of the tree. This allows large sub-trees to be removed, reducing the number of leaf nodes.

There is a further improvement which can be made. Currently each node contour is compared. There are two situations where the distance value, *d*, is irrelevant:

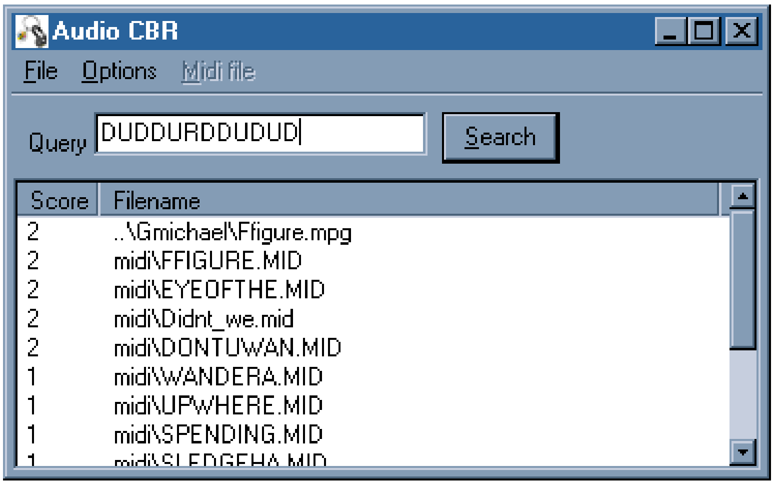
1. If the length of the contour at a tree node is less than the number of errors allowed, then it must be within tolerance;
2. If the height of a sub-tree is less than the remaining number of errors allowed then all leaf nodes of the sub-tree will be within tolerance and no node comparisons need be made.

All contours stored in the database are *L* pitch directions long. Queries longer than this atomic length are split into the set of overlapping sub-contours. Approximation is performed on each sub-contour and the near match sets pooled. This larger set is then used to search the database.

Tune retrieval can be compared with text retrieval; for example, the number of occurrences of a contour in a MIDI file is significant. However, it is perhaps more unusual in text retrieval system for the query to be so prone to error; as noted above, contour matching is sometimes more akin to spell-checking. This means that the policy for ranking results is different, giving priority to a large number of hits over a single precise hit.

A score is calculated for each file by summing the number of times a contour in the near match set occurs in the file. Each hit is inversely weighted by the distance between the query and the contour matched. The inverse weighting ensures that less importance is given to contours matched which are increasingly different to the query. The search results are sorted in order of file score, highest first.

This has the side affect that a longer query, while returning a more relevant top choice, will usually return more matches. This is similar to the way some web search engines work, of which “AltaVista” is an example, work. The more information a search engine is given, the more relevant the top suggestion is likely to be, although the many millions of documents may only partially match. This is due to the fact that no checking is done to ensure that the whole query is actually contained in the matched files, or indeed that they occur next to each other (that they are temporally aligned).



Query DUDDURDDUDUD

Score	Filename
2	..\Gmichael\Ffigure.mpg
2	midi\FFIGURE.MID
2	midi\EYEOFTHE.MID
2	midi\Didnt_we.mid
2	midi\DONTUWAN.MID
1	midi\WANDERA.MID
1	midi\UPWHERE.MID
1	midi\SPENDING.MID
1	midi\SLEDGEHA.MID

Figure 5 - A screen shot of the retrieval program

The content based retrieval program, as running on Windows NT, can be seen in **Figure 5**. It shows a pitch contour from “Father Figure” by George Michael. The top match is an example of the ability to associate a MIDI file with some other media. In this case it is a movie file for which the pitch tracking, normally needed to obtain this functionality, has been imitated.

An improvement was made to the system which performed some alignment checking on the top matches. This required having access to the source files as the version of the database does not store any processed data from the creation of the database (although it is theoretically possible). Searching took significantly longer but the quality of the matches was noticeably increased [1].

3.2 Tools for time based navigation

Temporal navigation can be implemented with relative ease. At any place in a piece of music, perhaps while listening to the music, a user may ask for available links. Their position in the music, or the position of a selection they have made by whatever means, is the source endpoint of those links. This is compared with the link information in the link database to determine all the possible destination endpoints. The information in the link database, which essentially consists of stream and position information in this case, is generated by the authors of links.

The idea also works with other temporal media, such as a news broadcast. News broadcasts typically announce the headlines at the beginning which is followed by the full stories. On hearing a headline of interest, a user may ask to jump to the full story for that headline without the need to listen to the other stories.

Temporal based navigation is similar to embedded hypertext (as found on the world wide web) in that links are not based on content but on the position in the document. This is useful for media files, such as digital audio files, as software to extract content from such files is not reliable enough for general use. Like embedded links, there is a considerable overhead in maintaining the linkbase.

3.2.1 System architecture

A system was developed which implemented temporal navigation of media

[1] Five files were found, instead of seven, for a query consisting of 12 pitch directions.

documents on the Windows 32-bit platform (Microsoft Windows NT4). The architecture for the system is shown in **Figure 6**.

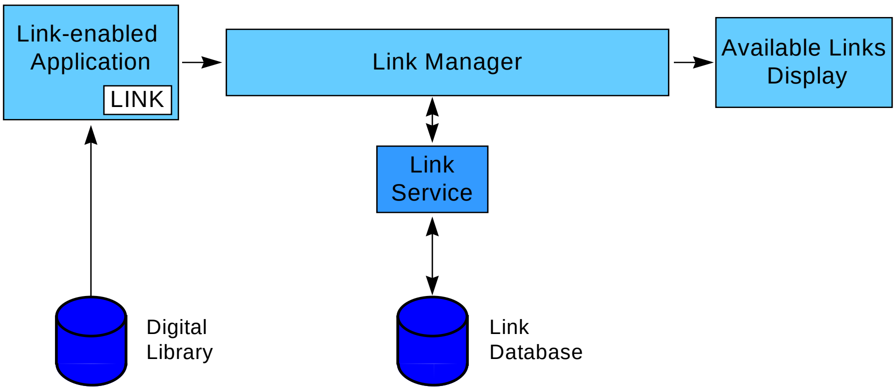
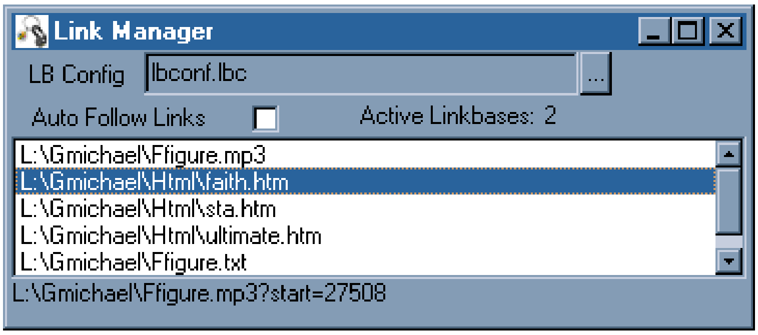


Figure 6 - The architecture used for temporal navigation

An endpoint identifies the document and a position or duration within the document. Endpoints are communicated between the components using a very simple message which is compliant with URL syntax. Links consist of multiple source and destination endpoints; endpoints in links could carry alternative representations (e.g. text, thumbnails) so that it would be possible to navigate through the hyperstructure without accessing the temporal media at all. These formats were adopted for simplicity during development: in due course they will be adapted for interoperability with SMIL, XLink and XPointer.

3.2.2 The Link Manager



LB Config lbconf.lbc ...

Auto Follow Links □ Active Linkbases: 2

```
L:\Gmichael\Ffigure.mp3
L:\Gmichael\Html\faith.htm
L:\Gmichael\Html\sta.htm
L:\Gmichael\Html\ultimate.htm
L:\Gmichael\Ffigure.txt
```

L:\Gmichael\Ffigure.mp3?start=27508

Figure 7 - A screen shot of the Link Manger

The Link Manager component, shown in **Figure 7**, receives source endpoints from the other tools and resolves them using a link service. This link service could be a local linkbase or possibly a remote OHP service. It also functions as the available links display. Following a link is achieved by double clicking on the entry in the list or, if the ’Auto

Follow Links' option is enabled, it will be followed as soon as it is resolved. It should be noted that the source endpoint (at the bottom of the window) is displayed for demonstration purposes only, it is not envisaged that the user would need this information.

3.2.3 The Link Player

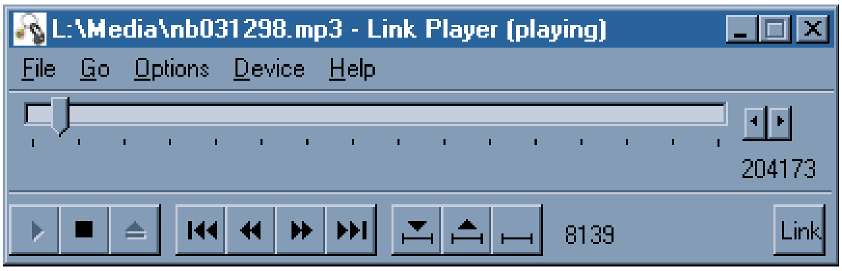


Figure 8 - A screen shot of the Link Player playing a file

The Link Player, shown in **Figure 8**, is a general purpose media player (essentially a wrapper for the operating system multimedia capabilities) which can send source endpoint information to the Link Manager, automatically or as a result of user interaction. It is designed to resemble the standard Media Player, a tool many users are already comfortable with.

The player has several options which control when an endpoint is sent to the Link Manager for resolving. 'Link on Play' sends an endpoint whenever the play button is pressed. 'Auto Link' sends an endpoint more frequently, approximately every second. The size of the selection sent to the Link Manager may be changed using a context menu on the link button.

3.3 Extending the tools for content based navigation

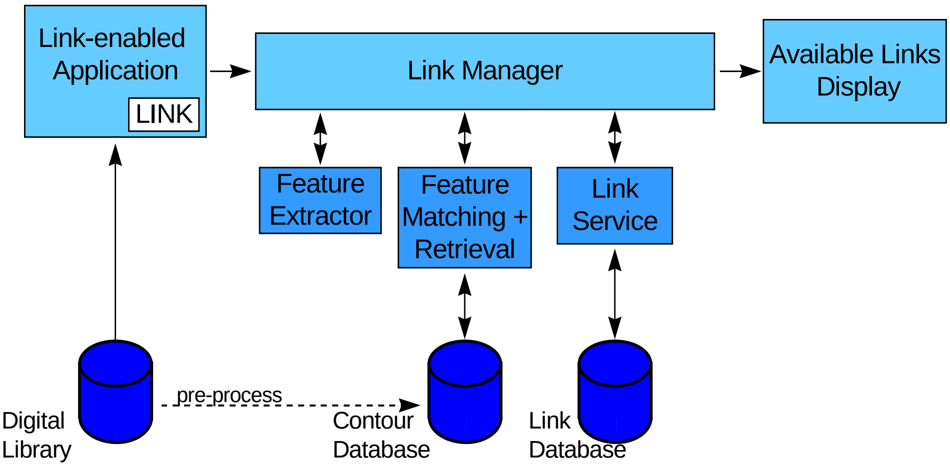


Figure 9 - The architecture extended for content based navigation

When linking on content, the position information is used to identify the content and extract a feature from it. This approach has been adopted because the link-enabled application might not deal with the multimedia data itself, as is the case with the Link Player. There may be scenarios where it is useful to deal directly with the content, such as with streams, but this may sacrifice the document context.

The contour database can be applied to content based navigation (CBN). With CBN the user may make a selection in a piece of music and the content of this selection used to identify relevant documents, or part of documents.

Early versions of the tools formed part of a technical demonstration given at ACM Multimedia ‘97 in Seattle. Since then, further requirements of a useful CBN system have been discovered as a side product of developing the prototype tools. For example, the media browser needs a file history, as is now common among web browsers.

3.4 Online music database

To increase the exposure of the system, the query by humming software was modified for use as a Common Gateway Interface (CGI) program on a web server. This was then installed on a freeware web server on a Pentium class machine running Windows NT. Most of the functionality was made visible, except for the more processor intensive functionality, such as checking contour alignment in searches. A small database of 160 files was used, due to space restrictions on the server. A tutorial of how to convert a tune into a pitch contour was also written and made available on the web site.

Although response to the web page has not been overwhelming, it has given an idea of what people expect of such a system. Several people tried searching using increasingly long queries, presumably expecting fewer matches for longer queries. As has been explained in Section 3, the system does not work this way due to the length of time it takes to check query alignment. This is a point in favour of modifying the system to handle this.

Another reason for the lack of queries is the fact that transcribing pitch contours by hand is not easy. The online system developed by McNab *et al*, MELDEX [27], allows a digital audio file to be sent to the server which then extract pitch contours using the Melody Transcription software they also developed. Commercial software is available for the IBM PC, Auto Score, which can convert singing into MIDI, which could then be used as a query. However, due to the realtime nature of the architecture employed by Auto Score, implementing similar functionality in the author’s software is not feasible.

3.5 Metrics for determining suitability for purpose

The papers on query by humming present figures detailing how long a query contour needs to be in order to distinguish itself from the majority of the songs in the database. For example, Ghias *et al* indicate that 12 notes were enough to uniquely identify 86% of 183 songs. This metric is useful for comparison with query by humming systems.

There is a major difference between the application of a representation to CBR compared to CBN. Retrieval must index all documents in the system which results in a large database where the possibility of false hits is greatly increased. In contrast, the linkbase used for navigation is very much smaller and of higher quality.

For CBN, identifying a file is not that useful, it is more interesting to identify a fragment of music in the whole database. If the metric is to be valid for all music representations (ie: not just contours) then giving the result in the form “a 10 second query is enough to uniquely identify 85% of a database containing 400 minutes of music”. What percentage is required before a system is considered useable is probably dependent on the representation being used. There will probably be a response curve of query length versus percentage of the database. It makes sense to pick the optimum point on the curve for each representation. This metric is probably related to expressiveness of music representation in some way, although the nature of this link is not clear.

This metric can not be accurately calculated for the existing contour database as all timing information is lost. Averages must be used instead by knowing the number of contours in the database and the number of minutes of music held it indexes. The metric has yet to be calculated.

3.6 Suitability of the database for content based navigation

The nature of the database, a reverse index of the entire set of MIDI files, made it possible to determine how useful the data can be. This has some reflection on the validity of using pitch contours as a representation of music. The effect varying the atomic length, *L*, of the database was also investigated.

Databases were compiled with atomic lengths of 12, 13 and 14. Once this was done, a complete pass of the databases was performed. A pass consists of looking at each possible sub-contour and noting the number of files which contain it. The result is a list of the number of sub-contours (queries) and the number of files in which they occur. More queries matching in fewer files suggests that there will be fewer false matches.

Test data was obtained from an FTP site on the Internet[2]. At the time the files were copied, the site contained over ten thousand files. Filenames which only differed in capitalisation were not downloaded. The checksum of each file was compared and exact duplicates removed, although this does not remove multiple versions of a song. This further reduced the number files by 500. Removing files which the database system could not use, either because they were single track MIDI files or because they were corrupt, left just over 7500 files for the experiment.

To give the reader an idea of the nature of the files, the average length was three and a half minutes. There were over 26.5 million pitch directions, averaging 310 pitch directions per minute of each MIDI track.

The results are plotted in **Figure 11** and **Figure 10**. The first graph shows that increasing the atomic length *L* does improve the resolution of the database. That is, more sub-contours are contained in only one file. To get an idea of how the efficiency of the database is affected, in **Figure 10** the same results are plotted against the percentage coverage of the database. This shows that over 25% of the possible contours with 13 pitch directions match just one file. Observe that the efficiency is decreased for greater *L*.

[2] Trantor FTP site: ftp://cam031313.student.utwente.nl/pub/midi [10 February 1999]

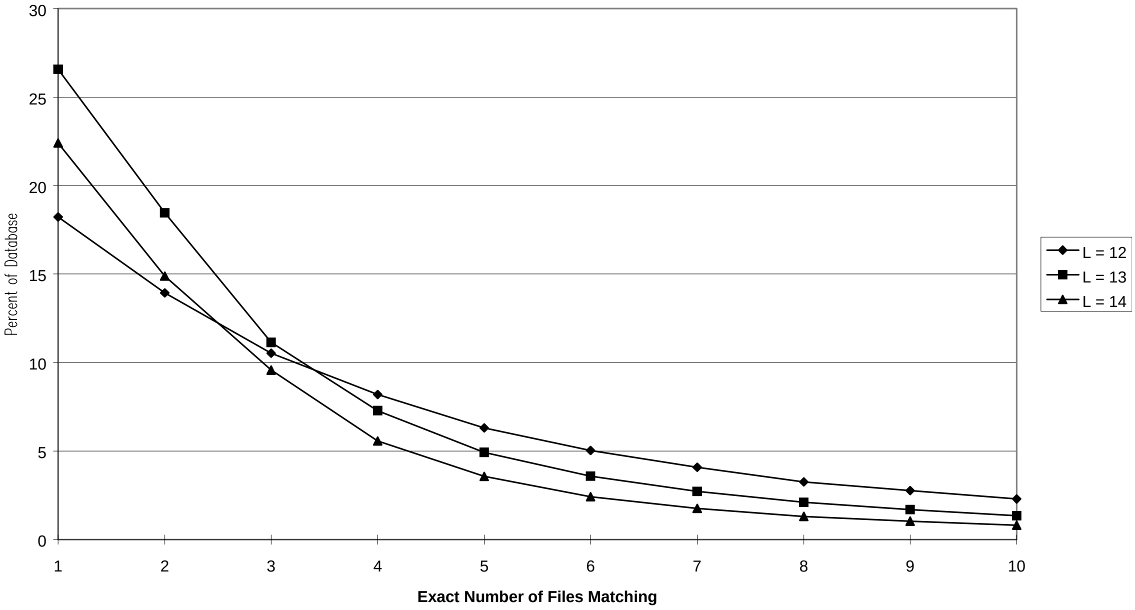


Figure 10 - Percentage coverage *vs.* Exact number of matching files.

The average number of files returned for each database is shown in **Table 1**. The decreasing average, together with the first graph, suggests that increasing *L* does improve performance, although efficiency is subsequently decreased.

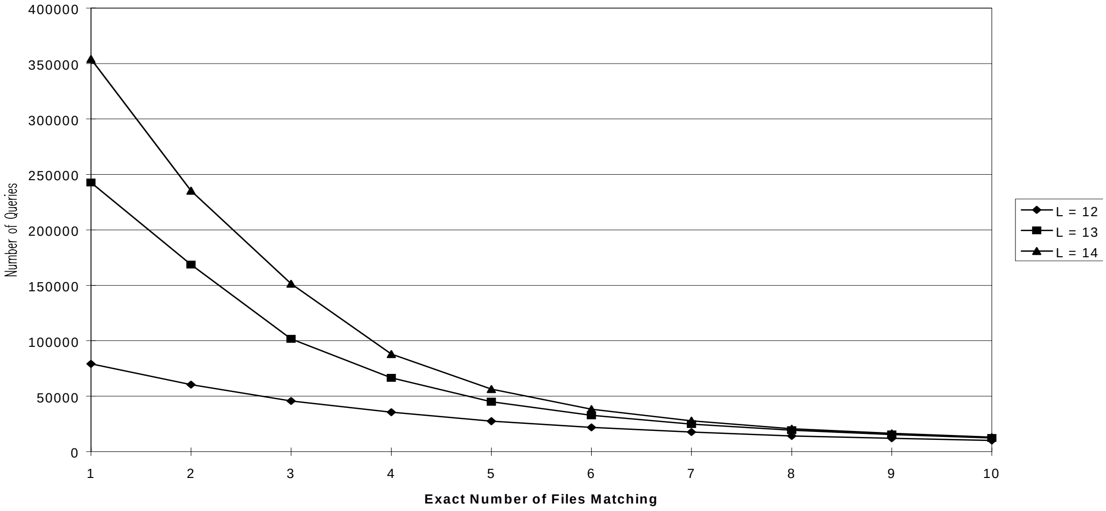


Figure 11 - The number of contours *vs.* exact number of matching files

Table 1 Average number of files returned for different atomic lengths.

L	**average**
12	16.97
13	9.71
14	5.99

3.7 Matching using secondary contours

Pitch contours are designed to be an abstraction of the notes to allow for errors in the input. This is useful for both hummed queries and the result of feature extraction engines. While pitch errors are likely to accumulate over time, more information can be deduced from notes which are close together. Using the example musical score shown in **Figure 12**, the pitch contour for both bars is identical, "UDU", although they are clearly very different to listen to.

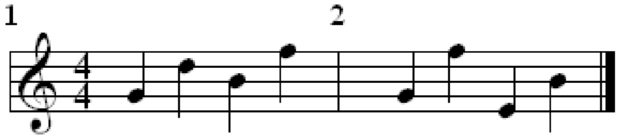


Figure 12 - Two different bars with the same primary contour

One improvement would be to classify intervals out of five possible types: up, up a lot, repeat, down and down a lot. The classification for up, down and repeat is the same as the one discussed previously. The distinction between "up" and "up a lot" could depend on a threshold on interval size, but a more reliable approach is to compare a note with a pitch previously established in the contour; e.g. if the current note is of higher pitch than the note before the last one, then it is classified as being "up a lot". This only applies when the pitch direction changes, otherwise all but the first pitch direction would be "up a lot". This could be encoded as a single character representation of the five pitch direction types based of: u, U, r, d and D for up, up a lot, repeat, down and down a lot respectively. The first bar of the example would be represented as "udU", while the second bar would be "uDu", showing a difference.

An alternate approach has been adopted which has a similar effect but retains the existing representation. A secondary contour is created where each symbol represents the relationship between the current note and the "note before last"; e.g. in the example above, this secondary pitch contour would be "UU" for the first bar and "DD" for the second bar. Although the possible secondary contours are constrained by a given primary contour, preliminary experiments suggest this approach to be very effective in reducing the search space.

Using the existing representation allows the same database structure to be used. A second file which contains the secondary contours is stored alongside the primary contour file. The lookup procedure is carried out for each contour file and the intersection of both sets of results are displayed to the user. This noticeably reduces the list of matches while maintaining a quick response time.

This initial query only identifies file which contain both the primary and secondary contours. Further improvements are made by checking that the contours are aligned: they happen at the same time in the file. This currently involves reading the short listed MIDI files and extracting the contours again. If the pre-processed content were stored then this additional checking would not have such a great effect on the time taken to perform a query.

4 Thoughts and Ideas

This section highlights some of the problems associated with content based retrieval and navigation. It identifies some of the areas which could benefit from further research. The author also gives his insights discovered while conducting the research.

4.1 Increasing the relevance of files matched

There may contours which are more popular than others. Knowing contours which are the musical equivalent of small words in English, such as ‘a’, ‘the’ and ‘it’, could be used to reduce the weight they carry when calculating a score for a match. It may even be worth removing them from the query completely.

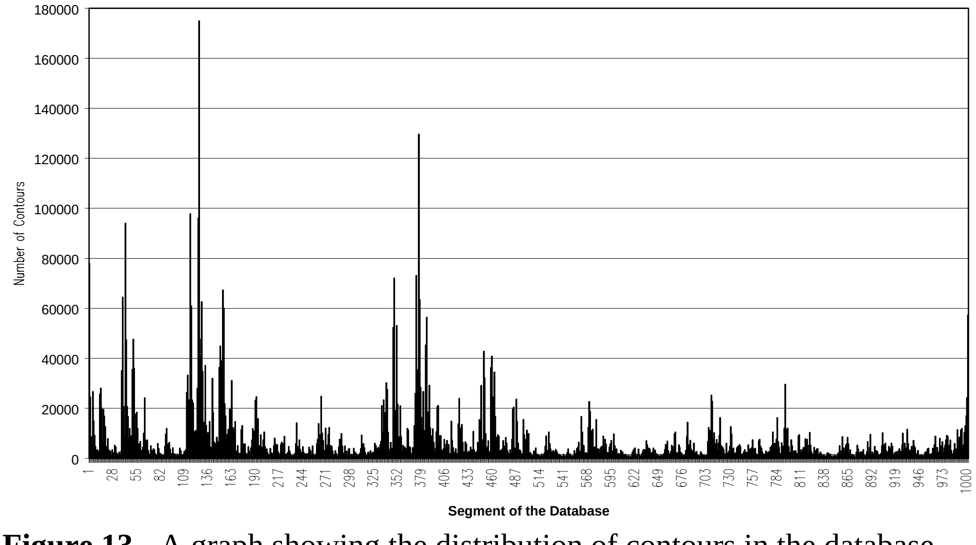


Figure 13 - A graph showing the distribution of contours in the database

Figure 13 shows the distribution of contours in the database of 7500 MIDI files where the atomic length is 12. It is not entirely clear what the graph shows but it suggests that all contours may not be equal, due to the large peaks. If all contours were the same then one would expect a more even distribution.

4.2 Content based retrieval vs. content based navigation

It is important to make the distinction between CBR and CBN, as they can appear similar. Content based retrieval allows a user to locate all documents containing some given content. This facility can be integrated with a media browser, so that the user makes a selection in a document and can locate all documents containing that selection.

Content based navigation has a style of interaction similar to that of retrieval: the user makes a selection and the system returns a list of files. Because of this, it is easily confused with retrieval. The distinction is that navigation returns a list of documents which are *related* to the selection, not necessarily *containing* it.

Another important difference is the resolution required from a representation. As an example, take a set of 100 files which are, on average, 50 seconds long giving 5,000 seconds of music. Content based retrieval of a file requires a query which is long enough to differentiate between the 100 files. Navigation on the same set of files, assuming a average endpoint is five seconds long and that there are no repeats, requires a query which differentiates between 1000 potential endpoints. While this is a very extreme example, a more realistic example might still require a more powerful query.

This is in contradiction to the point made in Section 3.5 which says that a content based navigation database contains less irrelevant data than a database for retrieval. This would mean that the navigation query would not need to be as powerful as retrieval one. It is unclear which of these points of view makes more sense, although it is usually beneficial to have a more accurate representation, assuming a similar computational overhead.

4.3 Expressiveness of representations

Although some experiments have been conducted towards deducing the suitability of contours for music, a more thorough approach would be to determine this mathematically. It is important to note that the number of possible combinations may be different to the number of valid combinations. Taking English as an example, using an alphabet of 26 characters it is possible to create a very wide variety of strings of which very few actually make sense. The author refers to the number of valid combinations for a given representation as the expressiveness. This may be distinct from the number of *useful* combinations.

There is probably a value or formula which represents the expressiveness for each representation. For example, the expressiveness of the primary musical pitch contour (see Section 2.3.3) is the same as the number of possible combinations. This is given by the formulae $e = 3^n$, where n is the length of the contour.

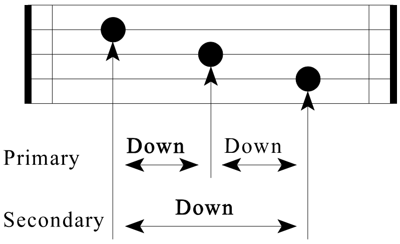


Figure 14 - A secondary contour

It can be seen that there the secondary contour must Down for a primary of Down Down (see **Figure 14**). If the secondary were Repeat on Up then the third note would have to be the same or higher than the first, making the primary contour Down Up.

The expressiveness for a secondary pitch contour is not as straightforward as there are constraints on what makes a valid secondary contour. This is further complicated as the secondary contour is only useful when the primary contour changes direction (UD or DU).

$e = 3^n + 2\ (2/3)\ 3^{n-1}$

There are many research issues which stem from this. Is using a secondary contour just as powerful as using a five segment classification for pitch directions? What about using a tertiary contour? What length of which type of contour is suitable for a given application?

How about other representations, such as chord sequences?

4.4 Using multiple contour representations

It has already been noted that pitch contours may not be powerful enough for some navigation applications, an improved representation is required. Given that an effective tool for working with contours based on an alphabet of three symbols has been developed, and that multiple contours already form part of the matching process, the initial experimental approach has been to work with multiple simple contours rather than developing a single, compound, contour; these approaches may be equivalent.

Another representation which needs to be considered is the time contour. Similar to the way a pitch contour describes a series of relative pitch transitions, a time contour describes a series of relative note lengths (or the length of time between each note). Time in a piece of music is classified in one of three ways: it is either a repetition of the previous time (R), within a tolerance; longer than the previous time (L); or shorter than the previous time (S). Thus the piece can be converted into a string with a three letter alphabet (L, S, R). The duration over which a contour lasts may be used to further reduce the search space; for example, this should help avoid one bar of a melodic part matching another part that changes just once par bar, such as a bass line.

4.5 Classification by music style / part

Another way of reducing the search space would be to limit the matching to a particular style of song (eg: jazz, classical, pop) or to match on a part of a song (bass, melody, chorus, introduction).

The biggest problem with using the style of a song is agreeing on which category a piece of music belongs to. This is due to classification of music being dependent on personal opinion and the continuous nature of music classification (that is, the existence of borderline material).

Identifying the melody in the piece would result in better matching, as the user is usually interested in the melody and only relevant information need be stored. The research of Uitdenbogerd and Zobel is working towards this.

4.6 Putting content based components in their place

It might be interesting to try and determine a common architecture for a hypermedia system. One would envisage this consisting of a set of high level components, their relationship to each other, and the requirements for each component. It would not be a physical architecture but rather a logical one which all hypermedia systems adhere to due to the requirements of a hypermedia system. This would involve examining the architecture of existing hypermedia systems (MAVIS, OHP, etc.) and the reasons behind each approach.

4.7 The Open Hypermedia Protocol (OHP)

One of the groups which is heavily involved with OHP is based in the University of Southampton. They are particularly interested in the content based navigation work as it illustrates many problems which are not evident in simpler, text based, systems.

Several issues for the protocol between the link manager and link services have been raised by DeRoure and Blackburn [8]. The matching algorithm used to find links will depend on the type of media while OHP does not yet provide a way of specifying either the algorithm or its parameters. The matching algorithms will return scores which will be used to rank results but there is no standard method for returning this value. Document relationships are also lacking, such as knowing that different quality versions of the same file can use the same links. The most interesting issue for content based navigation is where feature extraction occurs in the OHP architecture (client side, server side or pre-processed).

While OHP provides mechanisms for supporting temporal media and content based navigation, there are some areas which require clarification. Addressing the issues raised will improve interoperability throughout the OHP components and increase the chances of achieving wide acceptance.

4.8 IBM Data Explorer

The current method for determining the relationships between the size of an atomic unit and the performance and efficiency of the database has been to build an example database and analyse the results. This takes time and is limited to primary contours, although support for secondary contours could be implemented.

Using this strategy to compare alternative representations would require an implementation of a content based retrieval engine which supports each representation and a lot of time spent building and analysing the databases.

A more flexible approach may be to use an analysis tool like IBM’s Data Explorer, a data visualisation tool. This should allow more ‘what if’ type questions to be answered.

4.9 Notational differences for matching

Due to MIDI being a performance oriented standard, key presses rather than notes are communicated. This means that $C^{\#}$ and D^{b} are transmitted as the same key (which they are). The difference is not important to the listener but it is to musicologists. Although similarity matching in music concentrates on pitch, comparison of the notation might allow better use of the existing data. As MIDI files are the most widely available source of music, some method of determining the correct pitch notation will be required. This issue has already been investigated by Blombach [2] who has developed an algorithm for achieving this.

4.10 Texture of music

It would be interesting to see if determining the ‘texture’ of music, through the use of co-occurrence matrices (Haralick [18]) normally used to match texture in images, makes any sense. Co-occurrence matrices for music would give the probability of the next note being *x*, given that the current note is *y*. So if a tune consisted of two alternating notes then the musical equivalent of stripes would be represented. It is unlikely that texture alone would be enough to identify content, as it is very abstract concept, but would be useful in reducing the search space.

Applying the technique in reverse would be intriguing. That is, given a texture matrix, compose a piece of music using these probabilities. However, the value of any results are of questionable value to the field of computer science.

4.11 Contextual linking

Content based linking is usually reserved for generic links (to use a Microcosm term). In fact it is not possible to base a link on content alone without it at least applying to the rest of the file it is in.

Contextual linking allows an endpoint to consist entirely of content. The idea is that an endpoint of a link, which may be a paragraph of text, will remain valid even though the document around it is being edited. The challenge is to determine how much content either side of the endpoint must be stored in order to uniquely identify it. HyperTed, by Vanzyl [36], is a system implementing this for text documents.

Is contextual linking possible for music documents and does it make sense?

5 Conclusions

This mini-thesis has presented a review of the literature and given two methods by which temporal media may be used as the basis of a hypermedia document. Results obtained using the prototype tools, which have been presented at conferences [1][8][9], have been reported.

5.1 Research direction

Future research should investigate other music representations and matching techniques which may be used in a content based navigation system. Looking at indexing using chords sequences would be useful to compare with pitch contours.

The author believes the contribution will be fast and novel algorithms to index and match several music representations. It is likely that a method for determining the usefulness of a representation for content based navigation, and the results of the method applied to these representations, will also be present.

5.2 Work plan

Although there will inevitably be background tasks and unplanned activities, such as collaboration on OHP projects, the proposed work plan is given here. In the short term, working towards using IBM Data Explorer is a priority as this could greatly increase the number of ideas which can be tested.

March - May 1999

Develop the software needed to use Data Explorer.

June 1999

See what the effect of accounting ‘small words’ in queries has.

July - October 1999

Investigate the use of texture in the context of music.

November 1999 - January 2000

Examine the use of chord structure and, time permitting, notational differences for matching.

February - March 2000

Evaluate the representations, determining the limits - hopefully mathematically.

April - July 2000

The last four months will be spent writing up the thesis, with the intention of submitting by the end of July.

Bibliography

1. Blackburn, S. and DeRoure, D., A tool for content based navigation of music, in Proc. ACM Multimedia '98, 1998, pp. 361-368.

2. Blombach, A.K., Determining keys and correct pitch notation in tonal melodies, Computers in Music Research, Volume V, pp. 67-102, Spring 1995.

3. Buchanan, M.C. and Zellweger, P.T., Specifying Temporal Behaviour in Hypermedia Documents, in Proc. ACM ECHT Conference, November 1992, pp. 262-271.

4. Bulterman, D.C.A., Hardman, L., Jansen J., Mullender, K.S. and Rutledge, L., GRiNS: A GRaphical INterface for creating and playing SMIL documents, in Proc. 7th International World Wide Web Conference, pp. 519-529, Brisbane, Australia, April 1998. Computer Networks and ISDN Systems.

5. Carr, L., DeRoure, D., Hall, W. and Hill, G., The distributed link service: A tool for publishers, authors and readers, in Proc. Fourth International World Wide Web Conference: The Web Revolution, pp. 647-656, Boston, Massachusetts, USA, December 1995.

6. Davis, H., Hall, W., Heath, I., Hill, G., and Wilkins, R., Towards an integrated information environment with open hypermedia systems, in Proc. Fourth ACM Conference on Hypertext, Models for Open Systems, pp. 181-190, 1992.

7. Davis, H., Lewis, A. and Rizk, A., OHP: A Draft Proposal for a Standard Open Hypermedia Protocol, in Proc. 2nd Workshop on Open Hypermedia Systems. UCI-ICS Technical Report 96-10, University of California, Irvine, pp. 27-53.

8. DeRoure, D. and Blackburn, S., Amphion: Open Hypermedia Applied to Temporal Media, in Proc. 4th Open Hypermedia Workshop, pp. 27-32, 1998.

9. DeRoure, D., Blackburn, S., Oades, L., Read, J., Ridgway, N., Applying Open Hypermedia to Audio, in Proc. Hypertext 98. 1998, pp. 285-286.

10. DeRoure, D., Carr, L., Hall, W., and Hill, G., A distributed hypermedia link service, in Third International Workshop on Services in Distributed and Networked Environments, pp. 156-161, Macau, June 1996. IEEE.

11. Dowling, W. J., Scale and contour: Two components of a theory of memory for melodies, Psychological Review 85, pp. 341-354, 1978.

12. Foote, J.T., Content-based retrieval of music and audio, in Proc. SPIE ‘97, pp. 138-147, 1997.

13. Fountain, A.M., Hall, W., Heath, I., and Davis, H.C., MICROCOSM: An open model for hypermedia with dynamic linking, in Proc. ECHT ‘90 European Conference on Hypertext, Building Hypertext Applications, pp. 298-311, 1990.

14. Fushikida, K. Hiwatari and Y. Waki, H., A Content-Based Video Retrieval Method Using a Visualized Sound Pattern, in Proc. Visual Database Systems, 1998, pp. 208-213.

15. Ghias, A., Logan, J., Chamberlin, D. and Smith, B. C., Query by humming - musical information retrieval in an audio database, in Proc. Multimedia ‘95 (San Francisco, California, November 1995), pp. 231-236.

16. Goose, S. and Hall, W., The Development of a Sound Viewer for an Open Hypermedia System, in The New Review of Hypermedia and Multimedia, vol. 1, pp. 213-231, 1995.

17. Hall, W., Ending the tyranny of the button. IEEE Multimedia, 1(1): pp. 60-68, Spring 1994.

18. Haralick, R.M., Shanmugam, K., and Dinstein, I., Textural features for image classification. IEEE Transactions on Systems, Man and Cybernetics 3, pp. 610-621, 1973.

19. Hardman, L., Bulterman, D.C.A. and Rossum, G.v., The Amsterdam Hypertext Model: Adding Time and Context to the Dexter Model, Comm. ACM 37(2): pp. 50-62, Feb 1994.

20. Lemström, K. and Laine, P., Musical Information Retrieval Using Musical Parameters, in Proc. 1998 International Computer Music Conference (ICMC ‘98), October 1-6, 1998, pp. 341-348.

21. Lewis, P.H., Davis, H.C., Griffiths, S.R., Hall, W., and Wilkins, R.J., Media-based navigation with generic links, in Proc. 7th ACM Conference on Hypertext, pp. 215-223, New York, 16-20 March 1996. ACM Press.

22. Lewis, P.H., Davis H., Dobie, M., Hall, W., Kuan, J., and Perry, S., Content based navigation in multimedia information systems, in Proc. ACM Multimedia 96, pp. 415-416, New York, NY, USA, November 1996. ACM Press.

23. Lindsay, A., Using contour as a mid-level representation of melody, Masters Thesis, Massachusetts Institute of Technology, 1994.

24. Martin, K.D., Automatic Transcription of Simple Polyphonic Music: Robust Front End Processing, Technical Report 399, Massachusetts Institute of Technology, The Media Laboratory, December 1996.

25. McNab, R.J., Smith, L. A., Witten, I.H., Henderson, C.L. and Cunningham, S.J., Towards the digital music library: Tune retrieval from acoustic input, in Proc. DL'96, 1996., pp. 11-18.

26. McNab, R.J., Smith, L.A. and Witten, I., Signal Processing for Melody Transcription. Working Paper 95/22, August 1995.

27. McNab , R.J., Smith L.A., Bainbridge, D. and Witten I.H., The New Zealand Digital Library MELody inDEX, Technical Report, D-Lib Magazine, May 15, 1997.

28. MPEG-7 http://www.darmstadt.gmd.de/mobile/MPEG7/index.html [5 March 1999].

29. Ossenbruggen, J.v. and Eliëns A., Music in time-based hypermedia, in Proc. ECHT'94 European Conference on Hypermedia Technologies, Technical Briefings, pp. 224-227, 1994.

30. Pikrakis, A., Theodoridis, S. and Kamarotos, D., Recognition Of Isolated Musical Patterns In The Context Of Greek Traditional Music, in Proc. IEEE ICECS '96.

31. Ristad, E.S. and Yianilos, P.N., Learning String Edit Distance, Technical Report CS-TR-532-96, Department of Computer Science, Princeton University, 1996.

32. Sawhney, N. and Murphy, A., ESPACE 2: An Experimental HyperAudio Environment, in Proc. CHI '96, pp. 105-106.

33. SMDL (Standard Music Description Language), ISO/IEC DIS 10743.

34. SMIL (Synchronized Multimedia Integration Language) 1.0 Specification, W3C Technical Report REC-smil-19980615, June 1998. Available as http://www.w3.org/TR/REC-smil/ [5 March 1999].

35. Uitdenbogerd, A.L. and Zobel J., Manipulation of Music for Melody Matching, in Proc. ACM Multimedia '98, 1998, pp. 235-240.

36. Vanzyl, A., HyperScape: The Hypertext and Information Management

Environment for the Macintosh, in Proc. ECHT'94 European Conference on Hypermedia Technologies, Demonstrations, 1994.

37. Wiggins, G., Miranda, E., Smail, A., Harris, M., A Framework for the Evaluation of Music Representation Systems, Computer Music Journal, 17:3, Fall 1993, pp. 31-42.

38. Wold, E., Blum, T., Keislar, D. and Wheaton, J., Content-Based Classification, Search, and Retrieval of Audio, IEEE Multimedia, Fall 1996, pp. 27-36.

Index