Multimedia Systems Manuscript-Nr.
(will be inserted by hand later)

Content based navigation of music using melodic pitch contours

David C. De Roure and Steven G. Blackburn

Department of Electronics and Computer Science, University of Southampton,
Southampton, SO17 1BJ, UK

Received October 31, 1998

Abstract. We describe a system which supports dynamic user interaction with multimedia information using content based hypermedia navigation techniques, specialising in a technique for navigation of musical content. The model combines the principles of *open hypermedia*, whereby hypermedia link information is maintained by a *link service*, with *content based retrieval* techniques in which a database is queried based on a feature of the multimedia content; our approach could be described as 'content based retrieval of hypermedia links'. The experimental system focuses on temporal media and consists of a set of component-based navigational hypermedia tools. We propose the use of melodic pitch contours in this context and we present techniques for storing and querying contours, together with experimental results. Techniques for integrating the contour database with open hypermedia systems are also discussed.

Key words: content based retrieval, content based navigation, open hypermedia, melodic pitch contours, music

1 Introduction

This paper describes a project to investigate content based navigation (CBN) of music. The project has adopted an open hypermedia approach, with support for temporal media, navigation based on multimedia content, and specifically navigation based on melodic pitch contours in music. The key innovation of the CBN model is that the user can select an arbitrary part of any multimedia document and use this to query a hypermedia link database; the query is based on information extracted from the selection using a technique specific to the content type. This enables the user to find links even when there are no pre-authored links in the specific document (e.g. no buttons). We believe this to be a powerful technique for dynamic user interaction, to support both multimedia authors and users.

Our work extends previous work in open hypermedia by investigating content based navigation of temporal media. Melodic pitch contours are our case study and we extend work in this area through our novel application and a new contour database design which meets the requirements of a content-based navigation system (our preliminary work on this design was reported in [1]). We also address the systems integration issues of combining the content-based techniques with open hypermedia (building on [2]), including an approach based on software agents. With our focus on the CBN 'middleware' and associated contour database design, we do not specifically address user interface issues in this paper.

The motivation for the project is described in Section 2, where we discuss some scenarios which have helped in identifying the requirements for our prototype system. The open hypermedia approach is discussed in Section 3. Recent work in the open hypermedia research community has focussed on link services [3] and protocols [4]; Section 4 describes the prototype tools we have implemented to explore the issues in open hypermedia and temporal media in this context, building on the pioneering work of Hardman et al. [5]. Our investigation concentrates on audio, though many of the techniques are intended to be more general and we believe that the issues raised are relevant to other temporal media.

The extension of the system to support content based navigation is addressed in Section 5, including a discussion of temporal media issues. In principle we can work with a variety of features extracted from representations of musical data; Section 6 discusses the options and Section 7 focuses on one particular feature, the melodic pitch contour. Our use of contours is based on the work of Ghias et al. [6] and McNab et al. [7], who describe systems for querying an audio database by acoustic input, such as humming the tune of a song: the query is pitch-tracked and represented as a sequence of relative pitch changes (a melodic pitch contour), thus factoring out inaccuracies of timing and tuning. We extend this work with an alternative design for the contour database, presented in Section 8 which is designed to support CBN. We have exercised the design with a database of around 7500 songs and the results of this experiment are presented and discussed in Section 9.

In addition to extending the tools to support CBN, we consider two generalisations of this architecture which facilitate integration of the contour database with other open hyper-

Correspondence to: David C. De Roure

media systems. The first is query routing, which permits this system to participate in other link services, and the second is the use of software agents. These approaches are described in Section 10. Finally, Section 11 concludes the paper and discusses future work.

2 Motivation

The ability to follow hypermedia links to audio or video files is a standard feature of hypermedia systems; the facility to link to specified parts of these files, or to link out from them (for example, to the current location in a musical score or textual transcription of a speech) is currently less common but nonetheless desirable. Furthermore, there are situations which benefit from links between points within one temporal media type, such as interactive TV and radio applications.

In traditional audio applications the data is linear and unstructured; in contrast, the hypermedia approach regards audio as a branching, structured medium, like traditional hypertext but with the additional challenge of working with temporal data and streams. The Synchronized Multimedia Integration Language (SMIL) [8] goes some way towards supporting these requirements. With branching audio, listeners interact with the system to influence their route through the structure, and they can experience different routes according to their context and history. Branching material involves additional structural information, some of which is available from existing production processes and might be discarded at present, but could be retained and enhanced. Some of the structural information can be authored manually, a process which provides 'added value' as these authored links might not be able to be derived automatically.

Perhaps the most familiar opportunity for hypermedia linking within audio or video occurs in structured presentations, e.g. lectures, documentaries or meetings. Such presentations typically commence with an overview and, when playing back a recording, links can be made available from within the overview to the corresponding sections of the presentation. On hearing a particular topic of interest the listener can opt to jump immediately to that item; finding an interesting item abbreviated, the user might opt to listen to the full item. We have studied this scenario using radio documentaries, recordings of lectures and recordings of video conferences held using Internet conferencing tools. We have also assisted historical researchers working with recordings of speeches, where the particular requirements were synchronisation between the digital audio and the text transcriptions, linking to and from commentaries and pictures, and searching for material and associated hypermedia links based on fragments of speeches.

Interaction with musical structure is regarded by listeners as a more specialist activity than listening to documentaries, and is especially important to all those involved with music composition, performance, production and indeed education activities. As a case study we have investigated tools to assist in music education. Given a set of multimedia resources for music tuition, the user may wish to navigate around the material, following authored links, identifying occurrences of phrases or sequences in other songs, and linking between different representations; the set of links available might be a function of the stage of a course and the individual progress of a student, i.e. the hypermedia can be *adaptive*. We also envisage application in composition and, exploiting the network, in collaborative musical activities.

The author of the multimedia material in these scenarios needs not only the technology to associate hypermedia links with points in temporal media, but also tools to perform searches based on the multimedia content in order to facilitate the authoring of links. For example, an author may need to find occurrences of a spoken phrase, a melody or a chord sequence that they have specified or selected. Users also benefit from these techniques, such as the composer who wishes to ask "where did others go from here?" or the music student who wishes to find songs with a similar chord sequence to a fragment of music they have selected, to assist in their practice. Indeed users may create links, perhaps as 'bookmarks' or annotations, or as part of multimedia essays: as readers can be authors, so listeners can be composers.

3 Open hypermedia

The hypermedia model we have adopted for this work with branching audio is an example of an open hypermedia system [9]. Information about links between multimedia documents is stored and managed separately from the documents themselves. This information is stored in link databases (linkbases) and by selecting different linkbases, the user obtains different views of the documents. Access to linkbases may be abstracted by a *link service*, and there is ongoing work on the creation of a protocol for link service access [4].

By way of contrast, common usage of the Web exhibits the characteristics of 'closed' hypermedia. In particular, embedding URLs in documents is inflexible and makes maintenance very difficult, leading to a link integrity problem. Note that a 'link' in Web terminology usually consists of one URL, which in open hypermedia terminology would be called the *destination endpoint* (the link's source endpoint is implicit when the URL is embedded in HTML); a link in an open hypermedia system is a distinct entity that represents an association between *two* or more endpoints. Recent evolutions in Web standards provide the infrastructure to use the Web in a more open manner, with 'out of line' links [8].

Treating links as 'first class citizens' has many benefits. For example, it is easier to maintain link integrity (preventing 'dangling links') as it is linkbases, not documents, which need updating when a destination changes; it is also possible to identify links *to* a destination. Recent open hypermedia projects show a number of interesting developments based on this architecture. In the MEMOIR project [10], the trail of multimedia documents visited by users in a collaborative environment is recorded. In our setting this would enable users to ask questions such as "who else has listened to this?" and "what else did they listen to?".

4 Implementing open hypermedia tools for temporal media

One of our objectives is to explore open hypermedia applied to temporal media, in particular streamed audio, using contemporary link services. The pioneering Microcosm [11] system

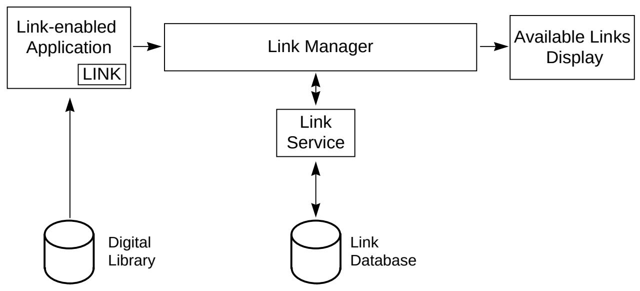


Fig. 1. Initial system architecture.

provides the sound viewer [12], an audio tool supporting hypermedia links into and out of stored audio, which we were able to use for our early studies. Our system resembles the Microcosm architecture but is designed for streamed media and new link services, such as the Distributed Link Service (DLS) [3]. In addition to Microcosm, the models which have influenced this work are the design of the Amsterdam Hypermedia Model [5] and previous work on the incorporation of musical objects into hypermedia systems; e.g. [13]. The MAVIS (Multimedia Architecture for Video, Image and Sound) system [14] also has some support for digital audio. However, since the existing systems do not directly support streamed media or distributed working, we have implemented new tools [1] for our experimental work.

We have adopted a component-based approach to the design and implementation of our experimental tools as this allows individual components to be replaced with alternative implementations. By writing a component which communicates with other open hypermedia systems, we achieve interoperability with the other system; we exploit this feature in the integration experiments discussed in Section 10. As well as the DLS, the system is designed to be compliant with the Open Hypermedia Protocol (OHP) [4] which is under development.

The system architecture is shown in Fig. 1 and the components are described below. The messages communicated between these components typically carry *endpoints*, which identify a (multimedia) document and a position or duration within the document. The messages use a simple format which is compliant with URL syntax, adopted for simplicity during development; this is currently being adapted for interoperability with SMIL, XLink and XPointer[8]. In line with the OHP model, a link consists of multiple source and destination endpoints. Endpoints can also be associated with alternative representations (e.g. text, thumbnails) so that it is practical to navigate through the hyperstructure without accessing the temporal media at all; this is a feature of the open hypermedia model which is particularly valuable in this context as it permits many operations (e.g. editing, browsing) when the user is remote from the audio data, perhaps on a low bandwidth network connection.

Central to our architecture is the Link Manager. This component receives messages from the other tools and resolves them using one or more link services (e.g. a local linkbase, or the DLS), presenting available links to the user via an appropriate interface component. The link presentation and selection mechanism is an interface issue and, while we have adopted standard tools in this project, novel interfaces are possible. In some situations the links might be followed automatically.

The Link Player is an example of a ‘link-enabled application’ and is a general purpose media player, essentially a wrapper for the multimedia capabilities of the operating system. It plays a single multimedia file and can send location and duration information to the Link Manager. A message may be sent as a result of user interaction, perhaps by pressing the link button, or automatically (e.g. at regular intervals or at predefined points).

While the Link Player deals with one multimedia document, another link-enabled application, the Sequence Player plays a linear sequence of media fragments, possibly from different documents, according to a simple description (list of fragments). This tool was created in order to deliver a synopsis of existing presentations, and is an example of a multimedia presentation application controlled by a session description language. We are currently investigating the use of players for SMIL documents in this way, using GRiNS [15].

One of the challenges of working with temporal media is that the data may be served from a dedicated streaming device, in contrast to text and image documents which are typically transported using a store-and-forward model. Furthermore, a user might join a session in mid-stream (analogous to tuning in to a radio or TV station) and leave at any time. For streaming work we have adopted the Realtime Streaming Protocol (RTSP) [16] and have constructed an RTSP server which can also describe and deliver streams of links. Our current interface to the RTSP server is an evolution of the Microcosm sound viewer.

We have built two other tools as part of our experimental work. The first is the Link Hider, which embeds endpoint information directly within digital audio data. Although this appears to contravene the open hypermedia philosophy, there are situations where such embedding is useful; e.g. where transport or storage is restricted to one digital audio stream, and for editing with standard audio software. In fact embedding endpoint or link information in audio can be far less invasive than with text, as audio formats typically support multiple channels; a useful illustration of this is provided by MIDI, where synchronized endpoint and link information could be coded on its own MIDI channel or using ‘system exclusive’ messages.

Finally, to address issues of synchronization between multiple versions of the same document in different media (in this case audio and text) we developed an authoring tool which facilitates the creation of links between speeches and their transcripts. The tool is implemented as a special HTTP server and the interface is via a Web browser.

5 Content based navigation

Many hypermedia and interactive multimedia systems have a notion of *buttons*; pressing the button causes a link to be followed. This is true of links embedded in the text of HTML documents on the Web, and images can have ‘image maps’ which provide an association between regions in the image and

[1] We call our toolset Amphion: according to Greek mythology, Amphion disliked the rougher life of his brother, a hunter and warrior. He preferred to accomplish by music what others did by force, easily moving rocks which his brother could not manage by physical means.

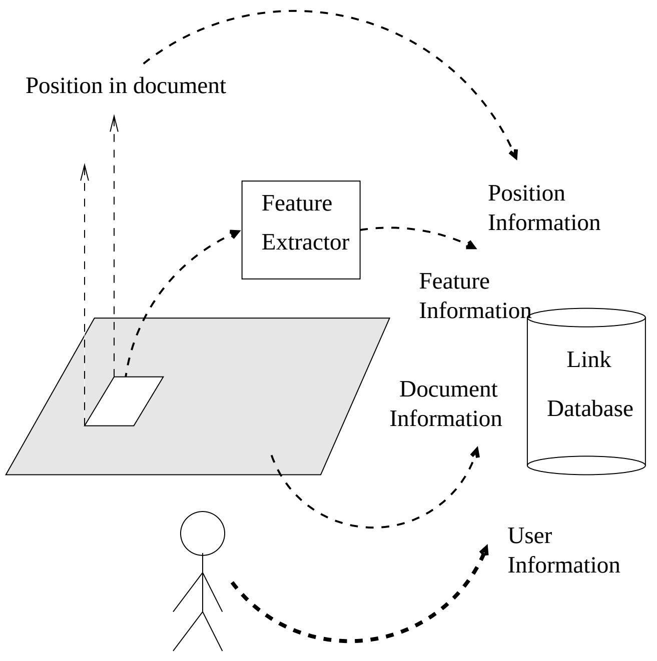


Fig. 2. Sources of information to assist queries.

links (in fact, by analogy our early temporal media solutions could be described as 'audio maps'). In both cases, there is a *position* in the document which has an associated link.

An open hypermedia system too can use position information, looking up the position in the linkbase to obtain the available links; i.e. the source endpoint stored in the link can correspond to a position (or duration) in the document. The key to content based navigation is the use of the *content* at that position to interrogate the linkbase, a concept introduced in the Microcosm system [11]. With text this might simply be a word; with multimedia content we derive features from the content [14]. The query to the linkbase can also carry context information, such as the identity of the current document and of the user. These sources of information are depicted in Fig. 2.

Buttons are a way of indicating that a position in a document has an associated link, and are an important interface component in many applications. One way of inserting buttons into a document is to use the linkbase to identify the links that are relevant in a particular context (e.g. given document, given user), and to insert them automatically; they can be 'precompiled' or inserted on-the-fly as in the Web proxy used by the DLS. However, with content based navigation the user can make an arbitrary selection of content, whether or not there is a button, and find the available links. This means they can 'link from anywhere' and it promotes a more query-oriented style of interaction [17].

The relationship between linking and querying is a theme which has been explored in the hypermedia community [18]. Content based navigation brings together content based retrieval (CBR) techniques with open hypermedia techniques; indeed, the content based navigation model might be thought of as content based retrieval of links. Content based retrieval of text is commonly provided by search engines and there are also many systems for content based retrieval of images, such as IBM's *Query By Image Content* system [19]. In the absence of any other link information, CBR is a useful tool to assist the user in navigating the information space. Although it performs a function which resembles link resolution, in that it generates available links from a given musical fragment, these are strictly links to similar material; this should be contrasted with the different types of available links that might be obtained from a linkbase. In fact, content based retrieval could be seen as a means of obtaining similar source endpoints rather than a mechanism for link resolution *per se*. For an author, creating a linkbase, it is another tool to assist in identifying endpoints.

For CBN, temporal media is distinguished from text and images in having a time axis along which selections are made, so that a selection corresponds to a point in time or a time interval. We also make the assumption that the digital content itself might be stored and managed by an entirely separate system, such as a server dedicated to streaming. At any place in a piece of temporal media (perhaps while listening to music), a user may ask for available links. The time or interval defined by the selection is compared with the source endpoint information in the linkbase, to determine all the available links and hence the possible destination endpoints. Comparison of position information involves testing whether a point in time falls in an interval, or whether an interval contains certain points in time, or whether an interval overlaps with other intervals; temporal proximity can also be taken into account. In many ways these techniques mirror the two dimensional case of images, and would combine to a three dimensional match in image sequences.

When linking on content, rather than position, the position information is passed to a separate component to identify the content and extract features on which to base the query; the multimedia data itself is not communicated from the user interface. We have adopted this approach because the application might not deal with the multimedia data directly, instead controlling the multimedia 'subsystem'. Note that with synchronised media, the content can be extracted at the same position in any stream; e.g. a selection of a sequence of frames in a video stream has associated content in an audio stream from which a query can be derived. In some situations, it can also be useful to deal directly with isolated content fragments (e.g. with streams, clipboards and 'unaware' applications) although this might sacrifice useful information about the document context.

6 Content based navigation of music

To navigate an information space including music we need to match one fragment of music against others, hence we need to extract features for the purpose of comparison. In line with our application scenarios, we have chosen to investigate one pitch-based comparison technique and one based on chord sequences. Both of these would be useful for a student studying jazz improvisation, for example.

Pitch-based comparisons are well understood. Comparison of absolute pitch information is ruled out because it is not independent of transposition. We could instead represent a melody as a series of relative pitch changes, but we go a stage further: the feature we adopt discards interval information and simply records pitch directions, which have been shown to be significant [20]. This *contour* is an established technique, as used for example in the *Melodic Index to the Works of Johann Sebastian Bach* [21]. Melodic comparison based on profiles of pitch direction has the attraction of being effective with human-generated queries, and the pitch-tracking technology to convert a single voice is readily available. There is a danger that the technique will match too broadly, but in our applications there are other techniques for reducing search space; for example, we are interested to see whether such a general technique would suffice in combination with some notion of context.

Chord recognition from a set of note sequences is also known to be a tractable problem, and although accurate polyphonic pitch tracking is problematic in the general case, chord tracking techniques are effective for a broad domain of music. Matching of chord sequences is a subject of current work and follows the similar principles to those discussed here for melodic contours.

In the longer term we seek to explore multiple techniques, separately and in combination, and our architectures are designed with this in mind. The reader is referred to [22] and [23] for a comprehensive description of available techniques. Mapping from digital audio to notes and thence to a feature is a traditional approach which is challenged by work on content analysis through models of audition at the MIT Media Laboratory [24]. With respect to our work, this emphasises the broader range of techniques in music understanding that are candidates for content based navigation.

We have employed the MIDI format as a representation from which to extract features; from MIDI we can construct pitch contours and representations of chord sequences. MIDI files consist of a collection of tracks specifying for each instrument what to play and when to play it. We can convert digital audio to MIDI by available pitch tracking techniques, and the representations can be aligned via time-stamps and position pointers. MIDI versions of material may be available directly from the production process (synchronized with digital audio) or they may be created independently. The latter case, synchronization with a digital audio version, presents an alignment problem which might be straightforward as adjusting tempo or as difficult as adjusting the arrangement.

7 Contours

A 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 as: the pitch went up (U); the pitch went down (D); or it is a repetition (R)[2]. 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 fewer symbol than notes as only the transitions between notes are recorded.

In a CBN system, contours might be derived from a number of source. One sources is a pitch-tracked voice, the so-called 'query by humming' (QBH) approach; the use of contours eliminates input errors which are due to the user singing out of key, out of time or out of tune. Another is from a fragment of MIDI. The sorts of errors that might occur in a contour depend upon the source and the technique by which the contour is derived. There are situations in content based navigation where the query and the contour data are both of very high quality, for example when the query is derived from a fragment of the same MIDI data that is being searched. Linkbase data can be very high quality, having been 'hand-crafted' and checked by the author. However, the query might be error-prone as in QBH, and the song data may be error-prone (perhaps derived by polyphonic pitch tracking). Although the matching in all cases implements the notion of 'sounds similar', the exact parameters of the matching process need to be tuned to the precise task.

Our song data is derived from a database of MIDI files. It is not always possible to know in advance how useful a track will be when deriving contours; for example, it is probably not useful to store drum tracks. In the absence of reliable heuristics for identifying prominent melodic features, we rely on an established standard for MIDI files called General MIDI, which matches instrument numbers with a known set of instruments; for example, this provides the useful information that the rhythm tracks are on channel 10. Converting a MIDI track is a simple matter of identifying the pitch directions when a note is played. However, one channel may carry several notes sounding simultaneously, so it is necessary to adopt heuristics such as taking the most recent note or, when several notes can be identified as commencing together (a chord) taking the highest note [25].

8 Contour database

The systems described by Ghias et al. [6] and McNab et al. [7] both have characteristics which are not well suited to our intended application in content based navigation as described in Section 5. The former only indexes the first part of each piece, while we need to identify all alignments, and we need to improve on the scalability of the latter to ensure response times are within the bounds for an interactive CBN system.

The pitch contour for a track is long, for example the mean value of pitch directions per minute in the test material is 310. 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. Our first example database used sub-contours which were a maximum of 12 pitch directions. The length of a sub-contour is referred to as the *atomic length*, which we shall designate by L. For example, where $L = 12$, the contour DURDRUURDRUDUR is split into the overlapping contour set:

```
DURDRUURDRUD
 URDRUURDRUDU
  RDRUURDRUDUR
```

The atomic length 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). For an atomic length of $L = 3$, the number of possible contours would be 27 (since $atomic\ resolution = 3^L$). This means that the system could only differentiate between 27 queries, which

[2] We use R (for repeat) in the same way as Ghias et al. use S (for same), as we reserve S for another contour representation.

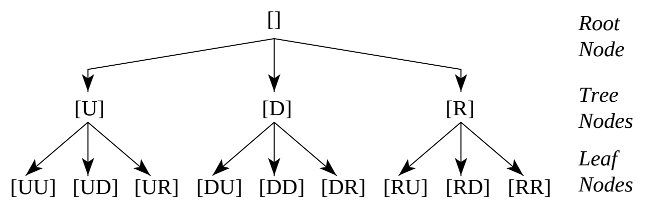


Fig. 3. The contour tree, where $L = 2$.

only need to be 3 pitch directions long. This is clearly not enough to identify a piece of music; L should be big enough to uniquely identify a song. The research by Ghias et al. [6] found that 12 notes (i.e. 11 pitch directions) were enough to identify 90% of their 183 songs. To allow selection from a larger number of songs, the atomic length should be as high as possible.

The database structure allows constant time access to determine if there are any matches for a query. Each subsequent access retrieves one match. However, this improved lookup technique only works with exact matches; performing an approximate match would require the entire database to be searched (which does not scale to large databases). To avoid this problem, we have adopted the notion of fuzzy queries, as opposed to fuzzy matches: possible errors in the query are emulated and the resultant list of near matches used to search the database.

To match a query, we produce a near match set of contours which are within a user-defined distance of the query. The distance is 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 distance limit d and choice of cost parameters depends 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. A number of algorithms are possible, including standard string matching algorithms; the one we have explored is the Levenshtein distance, where by default 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 QBH, more research into tune recall is needed to identify suitable weights; for example, common errors need to be identified, as with spelling checking techniques. It may also be possible to establish an appropriate distance metric automatically by learning from a reference set of data.

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 whose depth is 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 Fig. 3).

A simple tree search would compare the contour at each leaf node with the query, as in the brute force approach; indeed, it has the same time complexity as that approach.

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 traversal of the rest of the tree is unnecessary. This permits large sub-trees to be removed, reducing the number of leaf nodes. 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, for example, there is one error in the contour and there are two levels of the tree still to be recursively traversed, and three errors are allowed overall, then the whole of the sub-tree must be within tolerance.

This means that if $d = L$ then all leaf nodes could be added without comparing the contour at each node. This improves performance because the first d levels would not be compared. This decreases the time taken to match for large values of d but also has a noticeable effect for smaller values (i.e. $d < 3$).

All contours stored in the database are L pitch directions long. If a query is submitted with more pitch directions than this, a more accurate search of the database is expected. This is achieved by building a list of sub-contours from the query, as used when adding a contour to the database (see Section 8). Approximate matching 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 less usual in a 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 means 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.

9 Results

The nature of the database, a reverse index of the entire set of MIDI files, allows us to determine how useful the data can be. This has some reflection on the validity of using pitch contours as a representation of music. We also wanted to investigate the effect of varying the atomic length L of the database.

We compiled databases 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 in which it is contained. 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.

We obtained our test data from the Internet. When the files were copied, the site contained over ten thousand files. File-

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

L	number
12	16.97
13	9.71
14	5.99

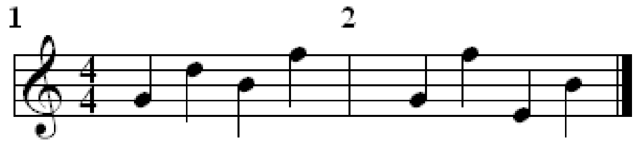


Fig. 6. Fragment of music with contour UDU in each bar.

names which differed only in capitalisation were not downloaded. We compared the checksum of each file and removed exact duplicates, although this does not remove multiple versions of a song. This further reduced the number of 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 Fig. 4 and Fig. 5. The first graph shows that increasing the atomic length L improves 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 Fig. 5 we plot the same results against the percentage coverage. This shows that over 25% of the possible contours with 13 pitch directions match just one file. We observe that the efficiency is decreased for greater L.

The average number of files returned for each database is shown in Table 1. The decreasing average, together with the first graph, suggests that, although the database may not be as efficient, increasing L does improve performance.

The performance of the system is suitable for content based navigation. On a single user Pentium 133 system, exact matches take 0.01 seconds on average for a query with L pitch directions. Matches allowing for one error in the query take 0.12 seconds.

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 occur close together. Using the example musical score in Fig. 6, the pitch contour for both bars is identical, 'UDU', although they are clearly very different to listen to.

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 we observe that 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 is only useful when the pitch direction changes, otherwise all but the first pitch direction would be 'up a lot'. A single character representation of the five pitch direction can be conceived; e.g. u, U, r, d and D for up, up a lot, repeat, down and down a lot respectively. The first bar of the example is then represented as 'udU', while the second bar is 'uDu', showing a difference.

We have investigated an alternate approach 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, our experiments indicate that this approach is effective in reducing the search space.

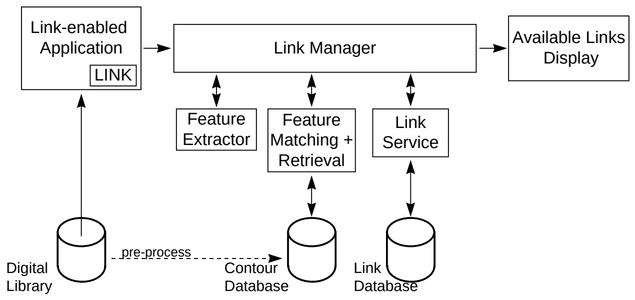


Fig. 7. System architecture for CBN

10 Integration of tools for CBN

We have conducted three experiments to explore systems architectures for integration of the open hypermedia and content based retrieval techniques. The first is an extension of the existing tools to incorporate the contour database as a 'CBR tool', the second integrates the contour database with other link resolution components to form a hybrid link service, and the third implements a 'wrapper' for the contour database so that it can participate in a community of software agents.

Figure 7 shows the architecture of the first experiment. As before, the Link Manager is the central application and is responsible for supervising link resolution; all the other tools communicate with it. When a user selects LINK from a viewer application, information about the current position or selection in the audio stream is sent to the Link Manager, which then resolves the link by interacting with other components and generating a list of possible destinations. The Available Links Display presents these destinations to the user via an appropriate interface.

The feature extractor uses information about a selection in an audio file to obtain the content of that selection and then extracts one or more contours (according to the number of tracks present). These contours are returned to the Link Manager, where they can be resolved independently or in combination. The system uses MIDI but in principle, given a selection in a digital audio file or a movie, the feature extractor could call a pitch tracker to generate the MIDI selection (or indeed, for a chord sequence database, a chord tracker). The Link Manager performs link resolution via a link service as before.

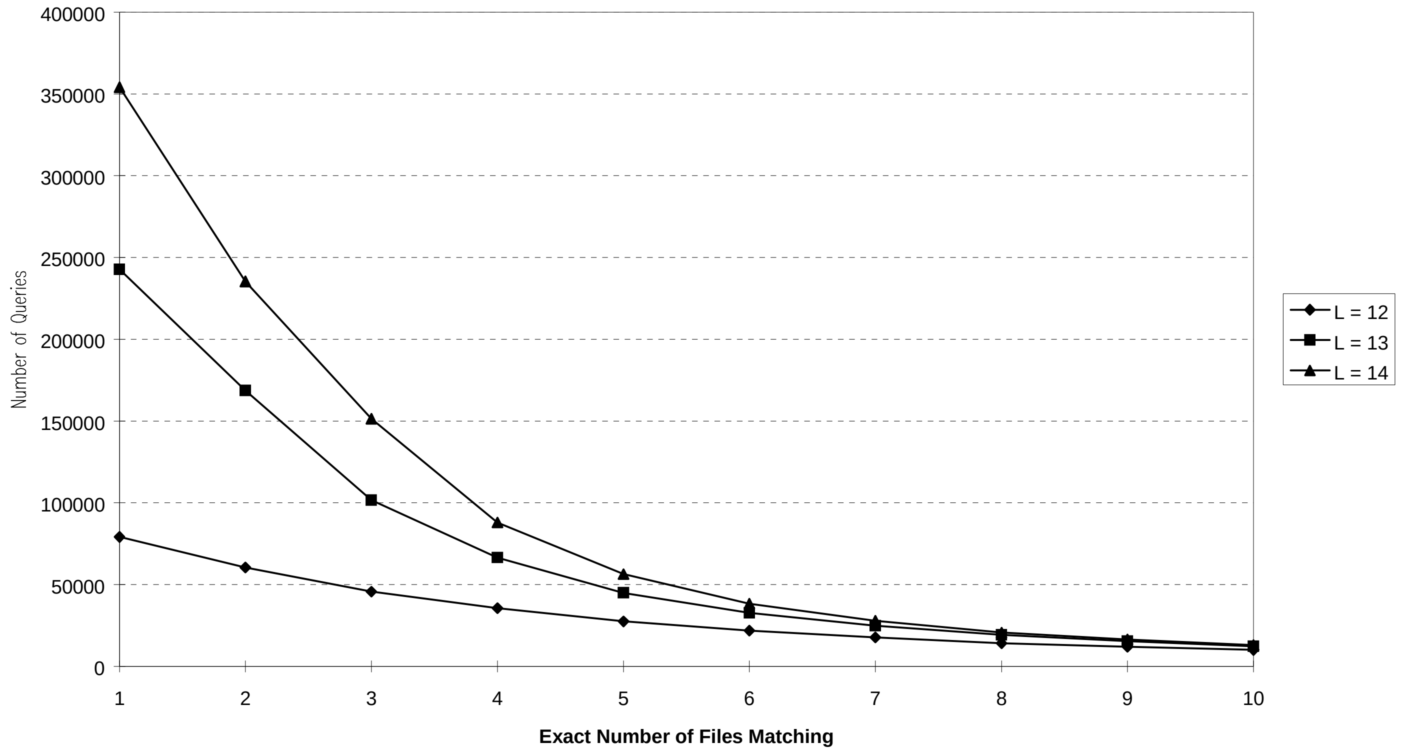
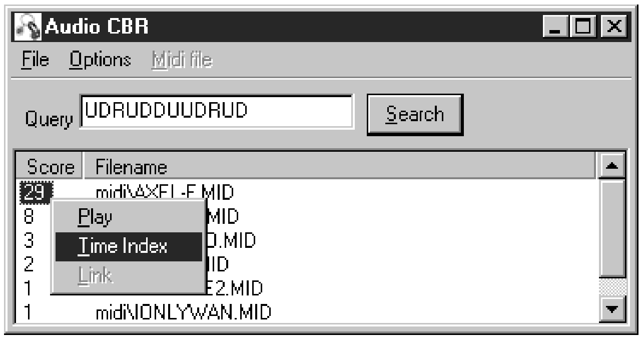


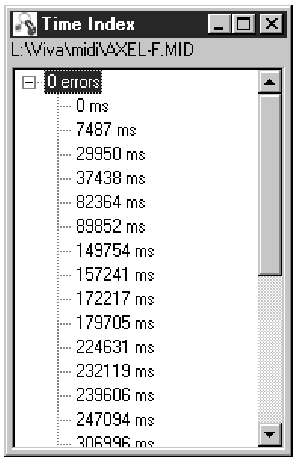
Fig. 4. Number of contours *vs.* exact number of matching files.



UDRUDDUUDRUD

Score	Filename
29	midi\AXEL-F.MID
8	MID
3	D.MID
2	ID
1	E2.MID
1	midi\ONLYWAN.MID

Fig. 8. The Content Based Retrieval Tool



L:\Viva\midi\AXEL-F.MID

```
⊟ 0 errors
    0 ms
    7487 ms
    29950 ms
    37438 ms
    82364 ms
    89852 ms
    149754 ms
    157241 ms
    172217 ms
    179705 ms
    224631 ms
    232119 ms
    239606 ms
    247094 ms
    306996 ms
```

Fig. 9. The time index display

The CBR component implements the database structure described in Section 8, accepting a contour as a query and displaying matches with their scores (see Fig. 8). Having selected one of the matches, the user can opt to play that song or to compute information on alignment of the contour matches; i.e. the positions at which the contour occurs, expressed as a time index suitable for use as an endpoint. Figure 9 shows the time indices for the best scoring match of the 12-symbol contour shown in the previous figure.

This system works well as a self-contained system, and can communicate with independent link databases via the appropriate link service protocol; i.e. DLS or OHP. In our subsequent experiments, we investigate whether the contour matcher can itself participate as a component of a link service for use by other systems.

Figure 10 illustrates a generalisation of the previous architecture, in which the Link Manager has a *query routing* function Q1 and there may be other such query routing components in the system, such as Q2 which routes between two separate contour databases. The role of the Link Manager is much as before but, along with the other query routing components, it now deals with *referrals*: when a query is sent to a component, the component has the option of returning a referral which directs the manager to an alternative component. For query routing to be effective, some components of the system need knowledge of the information contained by others. This is achieved by exchanging summary information, which must be done dynamically for components with databases that can be updated; e.g. when creating new links.

This experiment was conducted using tools which support the Whois++ directory service [26], and for the summary in-

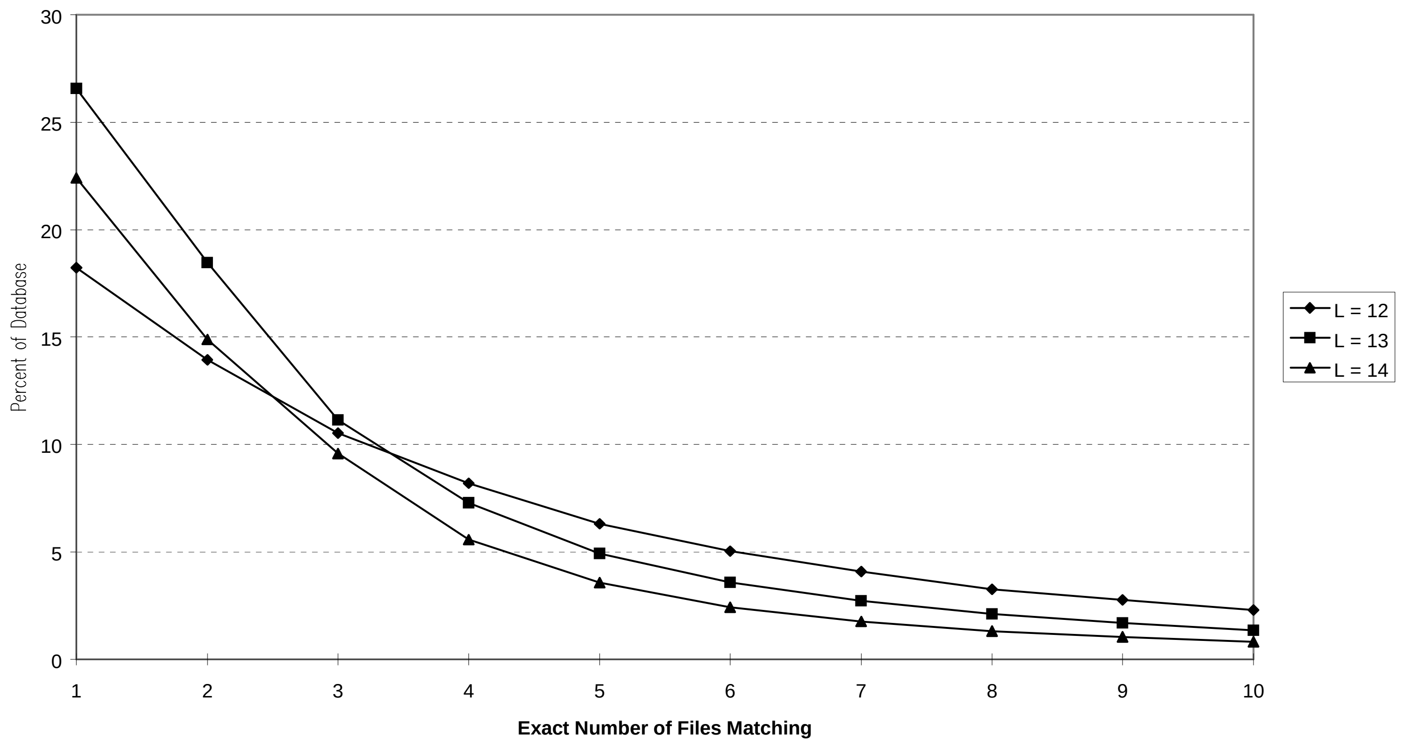


Fig. 5. Percentage of coverage *vs.* exact number of matching files.

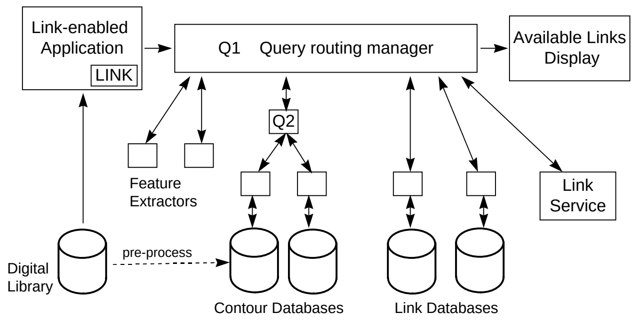


Fig. 10. System architecture for query routing

formation we adopted its attribute-value pairs approach. The query routing approach is effectively testing the idea that the link service can be implemented as a directory service; this works well for routine link resolution. It is straightfoward to integrate further services and for this system to participate in other services, as long as they support the notion of referrals and can provide summary information (dynamically if necessary).

However, the contour database has a very large index and producing summary information simply by listing contours is inefficient, particularly when loading the summary into other components. For the purposes of the experiment, we split the existing database in half, created two and concealed them behind query router Q2 which then provided a 'virtual' contour database. We found that Q2 could route more effectively (and with more concise summaries) if it could apply a simple match to the incoming queries, such as matching the prefix of the contour. This can be viewed as an extension of the contour matching algorithm into a distributed search.

Our final experiment is a further generalisation, in which all system components are regarded as software agents. Each agent implements a 'knowledge base' containing predicates. When an agent is started, it registers with a central registry. To make a query, the appropriate agents are identified from the registry and then they are queried; they return those predicates which match the query. We built an agent 'wrapper' in Java for the contour database, implementing the predicate **hasContour** with these parameters:

```
hasContour(mediaObject, contour, distance, rank)
```

The media object is an object containing the URL of the MIDI file, the contour is an object containing two strings representing the primary and secondary contours, the distance is the distance limit d for the fuzzy queries, and the rank is the position in the ordered list of matches. When the agent is queried with a contour and distance, it interrogates the contour database and returns a number of **hasContour** predicates, one for each match and each containing the appropriate media object and rank.

The contour database becomes a service which, once registered, is available for use anywhere in the system. The agent approach appears to be the most powerful and versatile: it accommodates diverse components in a common framework and can be extended to deal with the aggregation and filtering of link information.

11 Conclusions and future work

Our first experiments with the prototype tools included production of branching presentations by editing linear recordings of lectures, linking a historical speech to its transcript and to external documents, production of alternative views of musical performances, and demonstration of content based retrieval and navigation using a database of some 7500 documents. We have included video in our demonstrations by synchronizing with the audio track. These experiments have provided proof of concept.

The tools have evolved in response to user feedback. For example, we have introduced file history, to allow similar functionality to page history in a Web browser, and a context menu on the link button in the Link Player to facilitate rapid selection of endpoint size for queries based on content.

The requirements for the feature matching algorithm vary according to the nature of the data and the activity. For example, a search of the database using an error-prone query (e.g. a contour extracted from humming) requires a different parameterization of the matching algorithm to searching for a MIDI selection in a linkbase.

Whereas for text documents the Open Hypermedia approach must assert the advantages of not embedding ‘links’, this is less of an issue for audio documents where there is no well established format with embedded links and where data is often read-only and therefore enforces separate links. In fact, without the read-only constraint, our work suggests that endpoints and links can perhaps be embedded in audio formats whilst remaining ‘open’, thanks to multichannel formats.

Our experience of contours is that, while suitable for QBH, the contour representation is not sufficiently specific for some of our applications, in particular linking from selections in MIDI files when a precise query can be expected. A lot of information is discarded, which might otherwise help reduce the search space, such as rhythm information and the pitch intervals. We are investigating other representations, such as secondary contours. We are now also considering the time contour: as 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 may be classified in one of three ways: it is either a repetition of the previous time within a tolerance (R); 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 helps avoid one bar of a melodic part matching another part that changes note just once per bar, such as a bass line.

Contours are just one feature which we can extract from musical content and we are working on others. We are particularly interested in a hybrid approach, combining the results from different analyses, and the architecture of the tools will evolve to support this. We are reviewing pitch tracking techniques, as our requirements are different to other applications.

We support streaming via RTSP [16] and we see this as the means for transporting branching audio, hence we are exploring the use of RTSP for transporting endpoint information, linkbase information and for carrying queries. The server uses current and new generation Internet protocols (IPv6), and we are interested in developing these ideas further for multicast applications.

Our experiments with system architectures show the value in both the query routing and the agent approaches, and we plan to bring these together. The agent approach appears to be the most general and we envisage the system expanding to a community of agents exhibiting the characteristics of *weak agency* as defined in [27]; e.g. they will be reactive, proactive and will cooperate with other agents.

We believe that developments in digital broadcast technologies and standards provide an increasing role in the ideas and techniques we propose. In particular, the MPEG-4 Structured Audio Format [28] and MHEG-5 [29] are very relevant and we are tracking their development.

Hypermedia techniques are now commonplace; open hypermedia techniques, where appropriate, are increasingly viable thanks to evolution in the standards and infrastructure. Hypermedia and temporal media tools are becoming available. In this paper we have shown another step, towards a more generic linking mechanism which is particularly powerful for authors and provides functionality for readers too. We have shown how content based retrieval techniques can be applied to temporal media and how one content based retrieval technique, the melodic pitch contour, can be used in a content based navigation role.

Acknowledgement. We are grateful to our colleague Paul Lewis for discussions on content based navigation, and to Colin Bird of IBM UK Laboratories for his valuable input to this project based on experience with QBIC. The audio ontology for the agents experiment was designed by Danius Michaelides and the agent framework was designed and implemented by Luc Moreau. This work is partially supported by EPSRC grant GR/K73060.

References

1. S. Blackburn and D. De Roure, “A tool for content based navigation of music,” in *Proceedings of ACM Multimedia 1998*, pp. 361–368, 1998.
2. D. C. De Roure and S. G. Blackburn, “Amphion: Open hypermedia applied to temporal media,” in *Proceedings of the 4th Open Hypermedia Workshop* (U. K. Wiil, ed.), pp. 27–32, June 1998. Technical Report CS-98-1, Department of Computer Science, Aalborg University Esbjerg, Denmark.
3. L. Carr, D. De Roure, W. Hall, and G. Hill, “The distributed link service: A tool for publishers, authors and readers,” in *Proceedings of the Fourth International World Wide Web Conference: The Web Revolution*, (Boston, Massachusetts, USA), Dec. 1995.
4. H. C. Davis, D. E. Millard, and S. Reich, “OHP - communicating between hypermedia aware applications,” in *Proceedings of the Workshop ‘Towards a New Generation of HTTP’, held in conjunction with the 7th International WWW Conference* (J. Whitehead, ed.), (Irvine, CA 92697-3425), University of California, Irvine Department of Information and Computer Science, Apr. 1998.
5. L. Hardman, D. Bulterman, and G. van Rossum, “The Amsterdam Hypertext Model: adding time and context to the Dexter model,” *Communications ACM*, vol. 37, pp. 50–62, Feb. 1994.
6. A. Ghias, J. Logan, D. Chamberlin, and B. C. Smith, “Query by humming - musical information retrieval in an audio database,” in *Proceedings of ACM Multimedia 95*, (San Francisco, California), Nov. 1995.
7. 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 *Proceedings of Digital Libraries 96*, 1996.
8. “SMIL, XML, Xlink, XPointer,” tech. rep., World Wide Web Consortium. These are World Wide Web Consortium (W3C) documents and can be obtained from the W3C Web site `http://www.w3.org/`.
9. H. C. Davis, W. Hall, I. Heath, G. J. Hill, and R. J. Wilkins, “Towards an integrated information environment with open hypermedia systems,” in *Proceedings of the Fourth ACM Conference on Hypertext*, (Milan, Italy), pp. 181–190, ACM Press, Nov. 1992.

10. D. De Roure, W. Hall, S. Reich, A. Pikrakis, G. Hill, and M. Stairmand, "An open framework for collaborative distributed information management," in *Seventh International World Wide Web Conference (WWW7)*, vol. 30, (Brisbane, Australia), pp. 624–625, Elsevier, Apr. 1998. Published in Computer Networks and ISDN Systems.

11. A. M. Fountain, W. Hall, I. Heath, and H. C. Davis, "Microcosm: An open model for hypermedia with dynamic linking," in *Proceedings of the ECHT'90 European Conference on Hypertext, Building Hypertext Applications*, pp. 298–311, 1990.

12. S. Goose and W. Hall, "The development of a sound viewer for an open hypermedia system," *The New Review of Hypermedia and Multimedia*, vol. 1, pp. 213–231, 1995.

13. J. van Ossenbruggen and A. Eliens, "Music in time-based hypermedia," in *Proceedings of the ECHT'94 European Conference on Hypermedia Technologies, Technical Briefings*, pp. 224–227, 1994.

14. P. H. Lewis, H. C. Davis, S. R. Griffiths, W. Hall, and R. J. Wilkins, "Media-based navigation with generic links," in *Proceedings of the 7th ACM Conference on Hypertext*, (New York), pp. 215–223, ACM Press, Mar. 1996.

15. D. C. A. Bulterman, L. Hardman, J. Jansen, K. S. Mullender, and L. Rutledge, "GRiNS: A GRaphical INterface for creating and playing SMIL documents," *Computer Networks and ISDN Systems*, vol. 30, pp. 519–529, Apr. 1998. Proceedings of the 7th International World Wide Web Conference.

16. H. Schulzrinne, A. Rao, and R. Lanphier, "Real Time Streaming Protocol (RTSP)," apr 1998. RFC 2326.

17. W. Hall, "Ending the tyranny of the button," *IEEE Multimedia*, vol. 1, pp. 60–68, Spring 1994.

18. G. Golovchinsky, "Queries? links? is there a difference?," in *Proceedings of CHI'97*, (Atlanta), pp. 407–414, ACM Press, Mar. 1997.

19. M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker, "Query by image and video content: The QBIC system," *IEEE Computer*, vol. 28, Sept. 1995.

20. W. Dowling, "Scale and contour: Two components of a theory of memory for melodies," *Psychological Review*, 1978.

21. M. D. McAll, *Melodic Index of the Works of Johann Sebastian Bach*. New York: C. F. Peters, 1962.

22. W. B. Hewlett and E. Selfridge-Field, eds., *Melodic Similarity: Concepts, Procedures and Applications.* No. 11 in Computing in Musicology, MIT Press, 1998.

23. E. Wold, T. Blum, D. Keislar, and J. Wheaton, "Content-based classification, search and retrieval of audio," *IEEE Multimedia magazine*, vol. 3, no. 3, pp. 27–36, 1996.

24. K. D. Martin, E. D. Scheirer, and B. L. Vercoe, "Music content analysis through models of audition," in *ACM Multimedia 98 Workshop on Content Processing of Music for Multimedia Applications*, Sept. 1998.

25. A. L. Uitdenbogerd and J. Zobel, "Manipulation of music for melody matching," in *Proceedings of ACM Multimedia 1998*, pp. 235–240, 1998.

26. P. Deutsch, R. Schoultz, P. Faltstrom, and C. Weider, "Architecture of the WHOIS++ service," Tech. Rep. RFC1835, Aug. 1995.

27. M. Wooldridge and N. Jennings, "Intelligent agents: Theory and practice," *Knowledge Engineering Review*, vol. 10, no. 2, 1995.

28. ISO/IEC FCD 14496-3 Subpart 5, "Low bitrate coding of multimedia objects," May 1998. Part 3: Audio, Subpart 5: Structured Audio.

29. M. Echiffre, C. Mrchisio, P. Marchisio, P. Panicciari, and S. D. Rossi, "MHEG-5 - aims, concepts, and implementation issues," *IEEE Multimedia*, vol. 5, Mar. 1998.

DAVID C. DE ROURE is a Senior Lecturer in the Multimedia Research Group in the Department of Electronics and Computer Science at the University of Southampton, UK, where he leads the distributed systems research activity. A graduate in Mathematics with Physics, he received a PhD in Computer Science from the University of Southampton in 1990. His research interests include distributed open hypermedia, distributed programming models, emergent computing and music.

STEVEN G. BLACKBURN is a graduate student in the Multimedia Research Group in the Department of Electronics and Computer Science at the University of Southampton, UK. He received a BSc in Computer Science from the University of Southampton in 1997. His research interests include content based navigation systems, music representations and the Open Hypermedia Protocol.

This article was processed by the author using the LaTeX style file *pljour2* from Springer-Verlag.