Investigating Music Collections at Different Scales with AudioDB

Christophe Rhodes[*], Tim Crawford[*], Michael Casey[†], Mark d'Inverno[*]

January 12, 2011

Abstract

Content-based search of music collections presents differing challenges at different scales and according to the task at hand. In this paper, we consider a number of different use cases for content-based similarity search, at scales ranging between a detailed investigation of a single track to searching for fragments of a track against a collection of millions of media items. We pay particular attention to the varying tradeoff between precision and recall in these contexts, both from the point of view of system evaluation and from the point of view of a user of a system searching an unknown collection. We present the audioDB software for content-based search, and describe how it has been used to address use cases across these different collection sizes; we in addition show that the interpretation of similarity as a distance which can be modelled statistically, initially motivated by our desire to achieve sublinear retrieval time on large databases, can be used to improve the precision of searches over small and medium-sized collections.

1 Introduction

1.1 Information Retrieval from Recorded Music Collections

One aspect of Music Information Retrieval (MusicIR) is the management and navigation of music collections, which come in a range of sizes. The techniques and technologies used to manage and retrieve information about items in these collections differ substantially at small, medium and large scales.

The largest kind of collection include Internet repositories of user-contributed content, such as *YouTube*, which consist of hundreds of millions of digital items, both video and audio. To address copyright concerns these services employ robust fingerprinting techniques which identify near duplicates of a known set of target copyrighted works [Baluja and Covell, 2008]; however, their end-user interface is typically a metadata or free-text search, where such metadata can be authoritative, user-provided (through mechanisms such as collaborative filtering) or a combination.

Also at the large scale, but an order of magnitude smaller, are commercial music radio, subscription and download services that maintain collections of several million digital music items with their corresponding metadata. To use these items effectively, both metadata- and content-based information retrieval methods are employed to provide users with an experience of the collection that is suited to their music listening habits or interests [John, 2006, Wang, 2003, Wang, 2006].

[*]Department of Computing, Goldsmiths, University of London
[†]Bregman Music Audio Research Studio, Dartmouth College, NH

At the medium scale are collections of sounds or music that are oriented towards professional use. Containing tens to hundreds of thousands of items, they include production mood and incidental music libraries, sound effects libraries, global ethnomusicological collections, DJ music sites, music production sites and individual record label catalogues. Uses for music information retrieval methods at this scale include: replacing copyrighted music with similar production music for broadcast; automated identification of musical works; matching music to a production context; playlist generation; beat matching and many others.

Finally, at the small scale, are the narrowly cast music collections of specialist researchers such as those investigating performances of works by a single composer, folk song and ethnomusicological collections of a particular tradition, and music for personal pleasure stored on portable playback devices. Here the application of music information retrieval methods includes: identifying metadata from unidentified recordings; analysing short time-scale similarities between small groups of recordings and inspecting the results by listening – this is a use case for the musicologist, identifying concepts such as creative influence, musical appropriation and lineage in performance practice [Cook, 2007, Cook, 2010].

Thus, a plethora of tools and techniques have emerged, each oriented toward a single type of collection and a single task without regard for how the technique might scale, downwards or upwards in collection size, or wider or narrower in specificity. The result is that every new collection and task requires building a new system out of different music information retrieval components, and there have been no unified frameworks for MusicIR.

With the goal of unification across collection size, task specificity and feature selection, the OMRAS2 project at Goldsmiths, University of London designed an all-purpose feature vector database system called audioDB[1]. In this paper we explain our underlying model of information retrieval based on geometric methods, and show how that model can be applied to different use cases at different scales and with different tasks.

We present previous work in content-based similarity retrieval in Section 1.2. Section 2 describes the audioDB software, developed in order to unify treatment of content-based similarity search; Section 3 presents a series of use cases of the audioDB system at a variety of database scales, which motivates our experiment described in section 4 to verify whether our geometric approach improves search engine performance. Finally, we discuss our findings and conclusions in Section 5.

1.2 Content-based similarity retrieval

A MusicIR system first extracts musical features from audio or symbolic music representations which can be used for searching music. Two broad strategies are employed: those based on multi-dimensional time-series features and those using statistical models. The former are often called sequence methods and have been widely adopted for copyright detection and cover song detection tasks. Statistical features are often called bag-of-features (BOF) models as they reduce whole tracks to an aggregate feature vector representing summary statistics or probability distributions derived from time-series features.

Statistical summary features were first used for music genre classification experiments and subsequently explored for measuring similarity between audio tracks. Systems based on Gaussian mixture models were proposed by [Logan and Salamon, 2001] and [Tzanetakis and Essl, 2001]; here models of music are learnt from training data and subsequently new data is tested against each model using either the Earth Mover's Distance (EMD), in the case of [Logan and Salamon, 2001], and the Jensen-Shannon (symmetric Kullback-Leibler) divergence, in the case of [Tzanetakis and Essl, 2001]. The model with closest fit to the data was selected as the representative class in these experiments.

[1] `http://www.omras2.org/audioDB/`

Other statistical feature systems employ hidden Markov models (HMMs) [Casey, 2001] that are trained on exemplars from 20 music genre classes, then for a previously unseen music track, the HMM model with maximum probability is selected as the representative class among an ensemble of trained models. These methods were extended to general audio similarity by [Rauber et al., 2002, Flexer et al., 2005] and [Levy and Sandler, 2006], by training models for each individual track and comparing models using Kullback-Leibler divergence to get a measure of similarity between tracks.

The common elements among the above statistical feature systems are that they require training data and model fitting via a method to learn the parameters to probability models of a given type (such as 20-state HMMs), and they require a complex distance measure such as EMD which has time-complexity $O(d^2)$ in the dimensionality of the features.

While scalability with statistical features is plausible for generic tasks, where a single model exists per track or per genre, this strategy does not scale for tasks that require sub-track similarity computation. Such tasks include fingerprinting, near duplicates and cover song detection, which are the primary use cases for large collections. Here the model training and distance function complexity of statistical methods is prohibitive.

Distances between time series features can be measured by simple geometric distance functions such as Manhattan distance and Euclidean distance, i.e. L^1 and L^2 norms, that afford very efficient near constant-time implementations via geometric hashing schemes [Casey and Slaney, 2006a]. Systems that use time-series features have been proposed to solve robust identification [Wang, 2003], in which patches of time-frequency distributions are stored as acoustic landmarks and matched at query time using exact hashing. They have also been used for cover song detection [Serra and Gomez, 2007, Ellis and Poliner, 2007], remix identification [Casey and Slaney, 2006b, Casey and Slaney, 2006a], and work identification for classical music [Müller, 2007] with these systems operating at the sub-track level. Time series features were used with complex distance measures such as Dynamic Time Warping (DTW) [Ellis and Poliner, 2007, Ellis, 2007] with an improvement in accuracy for cover song retrieval, but making the time requirement prohibitive for operation at medium and large scales as DTW has complexity $O(L^2)$ in the length of the track and requires tracks to be considered pairwise.

The additional cost of a more complex distance metric is offset in the case of statistical features by having only one feature vector per track or genre. Conversely, for time-series features, the data is large because sequences often number thousands or tens of thousands of vectors per track, so for a collection with ten thousand recordings, there can be upwards of ten million vectors to be searched at query time. Our system, that we describe in the next section, is general enough to use any distance measure in principle, but for particular choices of distance measure can be made to scale to very large numbers of vectors; one way of recovering the ability to use statistical features efficiently is by embedding them into a simple metric space, such as L^1, with little distortion [Grauman and Darrell, 2004].

2 The audioDB system

AudioDB is an open-source database system, implemented in C++, for storing, indexing and searching high-dimensional feature vectors. It was designed to implement, via a single unified abstraction (described more formally in [d'Inverno et al., 2011]), many different music search use-case scenarios at different scales. While we do not claim that audioDB implements every method conceived within the field of music information retrieval, we have shown that it can implement a wide range of commonly-requested tasks, as well as new tasks. Unlike a standard relational database management system (RDBMS) audioDB has at its core an abstraction that treats each field in the database as a

vector or matrix, possibly of high dimensionality.

The database system has five components that inter-operate, each of which is analogous to standard database operations: database creation; data insertion; indexing; search; and result filtering and formatting. In the following sections, we describe aspects related to search (section 2.1) and result filtering and indexing (section 2.2).

2.1 Time-series matching

AudioDB matches, via some distance measure, time series [Casey and Slaney, 2006b] of features from audio. Feature vectors with dimension d are first extracted using an arbitrary feature extractor (such as a suitable VAMP[2] plugin, or a simple FFT-based extractor supplied with audioDB), and the resulting numerical features inserted into a database instance. At query time, the set of feature vectors for each music track are interpreted as multi-dimensional time series. The user specifies the length l of the time-series, and the search for matches then involves the distance between two $l \times d$-dimensional vectors according to the distance measure.

$$d^2_{\mathrm{Euc}}(x,y) = \sum_{i=1}^{l}\sum_{j=1}^{d} x_{ij}^2 + \sum_{i=1}^{l}\sum_{j=1}^{d} y_{ij}^2 - 2\sum_{i=1}^{l}\sum_{j=1}^{d} x_{ij}y_{ij} \tag{1}$$

$$d^2_{\mathrm{EucNormed}}(x,y) = 2 - \frac{2}{\sqrt{\sum_{i=1}^{l}\sum_{j=1}^{d} x_{ij}^2 \sum_{i=1}^{l}\sum_{j=1}^{d} y_{ij}^2}} \sum_{i=1}^{l}\sum_{j=1}^{d} x_{ij}y_{ij} \tag{2}$$

The Euclidean and normed Euclidean distance measures (as defined in Equation 1 and Equation 2) are provided out of the box, with a direct implementation in terms of a matched filter implementation with memoization of partial sums, yielding an algorithm with complexity $O(ldN)$ for a search, where N is the size of the database. For large-scale collections, a search algorithm linear in the size of the database is too slow to be practical; we have therefore implemented a probabilistic indexing technique, described in the next section, for these distance measures. While the framework allows for other distance measures to be used to compare feature vectors, not all distance measures allow indexing in this way, though many do (in particular the Euclidean distance, the cosine distance, and all L^p norms for $1 \leq p \leq 2$); the fact that the L^1 norm or Manhattan distance is indexable allows the system to be directly useful for features based on statistical models [Grauman and Darrell, 2004].

2.2 Efficiency, Indexing and Thresholding

Use of a simple distance metric admits highly efficient approximate implementations via locality sensitive hashing [Datar et al., 2004, Andoni and Indyk, 2006]; for our use cases, described in Section 3 below, we accept the approximate, probabilistic nature of LSH as a price to pay for scalable query times.

A form of geometric hashing that uses random projections, LSH yields time complexity for search that is $O(wdn^{1/c})$ for an approximation factor c. To use LSH we first have to build hash tables out of the vectors in the database. This is the *indexing* step discussed in Section 2. Then at query time, we hash the query vectors into the hash tables and retrieve those database vectors with which the query vectors collide.

[2] `http://vamp-plugins.org/`

LSH employs an ensemble of K hash functions of the following form:

$$h^{(k)}_{\mathbf{a}_k,b_k}(\mathbf{v}) = \left\lfloor \frac{\mathbf{a}_k^T\mathbf{v} + b_k}{w} \right\rfloor, k = 1 \ldots K \tag{3}$$

where $\mathbf{a} \in \mathcal{R}^d$ is a random vector drawn from an i.i.d. p-stable distribution – Gaussian for approximating the L^2 norm and Cauchy for approximating the L^1 norm; $\mathbf{v} \in \mathcal{R}^d$ is the data vector to be hashed (in our case this is a stacked sequence of time-series vectors); b is a uniform random scalar offset drawn in the interval $[0, w)$ and w is the hash bucket width. The effect of each independent hash function, $h^{(k)}$, is to project the data vector to the real line which is divided into equal parts of length w. The offset parameter, b, effectively scatters projections uniformly between neighbouring bins. Finally, the K multiple hash values, which are integers, are combined into a single hash value using a standard hashing algorithm.

Once hashed in this way, near-neighbour vectors, *i.e.* those that fall within some radius R of each other, will hash to the same location with high probability. Some vectors will not hash to the same location as their near neighbours – these are false negatives from the approximation involved – and sometimes vectors that are not near neighbours hash to the same location, these are false positives.

By manipulating the number of locality sensitive hash functions combined into a standard hash function, we control the false positive rate. Higher K means fewer false positives but more false negatives. To decrease the number of false negatives we repeat the locality sensitive hashing multiple times with H independent hash tables, increasing the true positive rate without impacting the false positive rate adversely, at the cost of doing $K \times H$ random projections.

The bin width parameter, w, is directly related to the desired retrieval radius R for searching; thus, the hashing-based method requires a radius threshold for the search decided when building the index data structures. Searches based on filtering retrieved results at that distance threshold are then approximated by the hash lookup, with matches at a distance under the threshold R retrieved with high probability and those at a distance over cR – the threshold multiplied by the approximation factor – returned with low probability. The radius threshold, and hence the bin width parameter, therefore encodes the decision boundary between retrieved and non-retrieved documents, which ideally should be the same as the relevant / non-relevant boundary.

The methods described in [Casey et al., 2008] estimate a whole-database distance threshold for a pair of data vectors to be significantly closer than expected (based on the background statistics of the database), which then allows the above probabilistic indexing scheme to be implemented. False positives resulting from the probabilistic indexed retrieval (though not from the feature itself) can be eliminated with a post-processing stage on the retrieved set; false negatives from the probabilistic nature are not recoverable. In addition, the whole-database threshold is not necessarily appropriate to all entries in the database, leading to additional false negatives from too tight a radius threshold for certain queries (see figure 1). This and other considerations of retrieval system performance are discussed further in the next section.

3 Use cases: retrieval at different scales

In this section, we describe particular use cases for content-based Music Information Retrieval that have been or are in the process of being investigated using the audioDB software described in section 2. We begin in section 3.1 by discussing the large-scale picture, where any content-based investigation at all is only possible because of the indexing admitted by the geometric interpretation of our sequence-based features described in section 2.2. In sections 3.2 and 3.3, we specify various scenarios where our sequence-based matching approach is also able to address particular tasks of

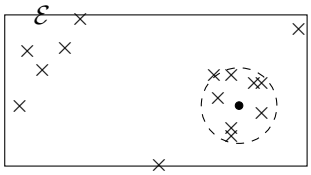

Figure 1: Distribution of nearest-neighbours in high-dimensional spaces of our musical features is highly non-uniform. A threshold appropriate for one track in the database (the filled circle) may cause musical similarity between other pairs of tracks to be missed completely.

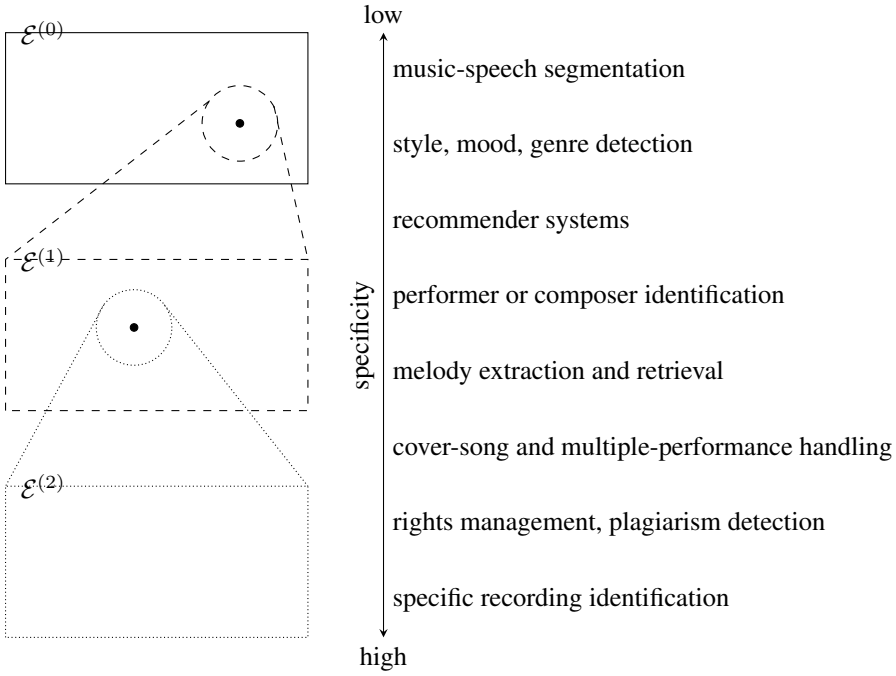

Figure 2: Illustration of different notions of similarity at different scales (left), where the notion of what is considered similar depends on the overall contents of the database under consideration. Compare and contrast with the specificity of similarity notions (right), modelled after [Byrd, 2007].

musical interest in medium and small-sized collections. This variation of tasks by scale (and the corresponding effect of the scale of the task on the success criteria for the task, described in more detail below) can be thought of as successive identifications of a coherent set of tracks from a larger database, illustrated in figure 2; notions of similarities at these different scales bear comparison with the specificity of music similarity searches, enumerated in [Byrd, 2007]. The notion of specificity is related to the size of the database but also to the size of the desired results set; thus although some of the highly-specific tasks (specific recording identification, rights management) are performed on large databases, it is not appropriate to perform some of the less specific tasks on small-scale collections.

3.1 Large-scale

Using time-series features, we can perform three types of search: single-time-point search, exhaustive track search, and database cross product search. For a million-song collection, where each song has on the order of a thousand segments, the single-time-point query would require distance computation and sorting of distances for $1{,}000{,}000 \times 1000 \sim 10^9$ vector sequences. For the whole-song version, 1000 such comparisons are required resulting in $\sim 10^{12}$ distance measurement and sort operations. This estimate of operation count is for a near-neighbour search for all portions of a single track against the entire database.

However, generally speaking, MusicIR tasks for large collections are batch operations, in the sense that the database of tracks to be matched against is known and acquired in advance. This means that the user experience of the large collection can be enhanced by a back-end system that has pre-trawled the data set and precomputed facets of interest, such as near-duplicate detection, which was first used in Web search engines to reduce the redundancy between the most highly-ranked search results [Broder et al., 1997] (and so to improve the overall relevance of the group of results returned).

One application of such methods to music is related song substitution. Here, we observe that a large music collection sometimes does not contain the specific song that a user requests in a metadata search, but may contain a different version of the same song: such as a remix, alternative language version, live versus studio version or a cover version by a different artist. Motivated by the prospect of recommending alternative songs for failed queries, shingling was applied to audio features [Casey and Slaney, 2006a] to construct a high-dimensional metric space. The application required performing a database cross product, *i.e.* all tracks against all tracks, at the subsequence level. This required $(1000 \times 1{,}000{,}000)^2 \sim 10^{18}$ distance computation and comparison operations, which, for high-dimensional feature vectors, can result in $\sim 10^{20}$ floating point operations.

To reduce the computational complexity of the search, we employed geometric hashing using locality sensitive hash (LSH) functions, see Section 2.2. AudioDB using LSH has been employed in near-duplicate detection, related-song classification and audio database exploration. The results for a subset (containing about 2×10^6 vectors) of a larger collection, used to test the identification of related songs, is shown in Figure 3. Here the precision and recall performance of the LSH based system is shown to perform with the same level of accuracy as the brute-force computational method, given a suitable choice of threshold radius hashing parameter. Tradeoffs in accuracy are manifest as lower recall rates for large databases at full scale; however, the benefit comes in the lower runtime for the query (in this experiment, the brute force search took about 100 times longer than constructing the LSH data structures and performing the indexed search).

The same approach has also been used in an experimental manner on image searches. While for images it is not possible to construct a consistent linearization of the image data that preserves locality, it is possible to take a two-dimensional cepstral feature, treating that as the feature for the

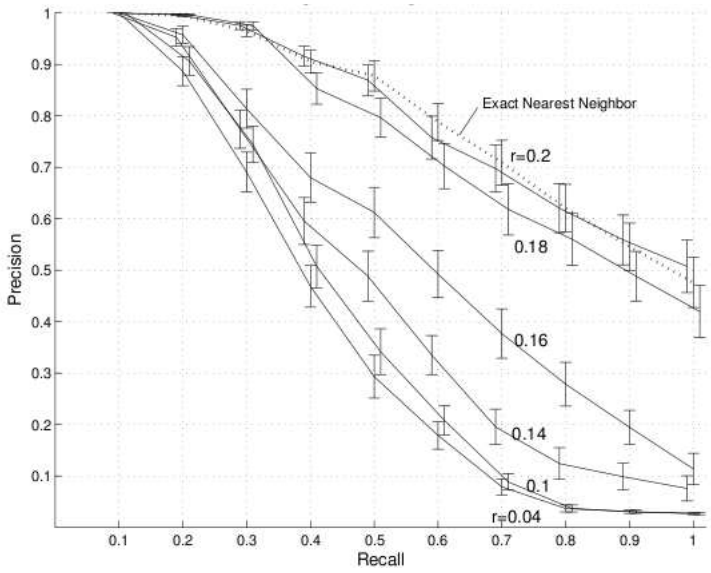

Figure 3: Results of identifying remixes in a $2,000,000$-vector subset using (dashed line) brute-force sequence comparisons and (solid lines) locality sensitive hashing for radius thresholds of between 0.04 and 0.2. (from [Casey and Slaney, 2006a]).

image. Because there is no notion of sequence in still images, the sequence approach is inapplicable; however, treating the image features as having an effective sequence length of one, the audioDB software is still able to return images with reasonable similarity to a query image, tested using a subset of the Flickr Photo Sharing website[3] [Slaney, 2009].

In all these applications of content-based searching on large collections, the primary requirement is that the returned list of results be relevant to the query: the precision should be high. It is usually less important, except for special cases[4], for recall to be high: there will usually be many items in the database that are relevant to a query, and the user is generally not interested in retrieving them all. In such circumstances, the tradeoff of reducing recall (by introducing a distance threshold; see section 2.2) in the interest of maintaining high precision while allowing for short retrieval times is an acceptable one.

It should be noted that the benefits of the probabilistic indexing using locality-sensitive hashing on large databases is task- and data-specific. In some cases, the LSH data structures themselves can directly represent the desired task result: effectively, a 'map' of close relationships between shingles in the database. Some other tasks, with less tolerance for false positives from the probabilistic nature of the index, require validation of the relationships by going back to the audio feature data to compute the exact distances – and if the data are such that large portions of the database are close to each other (relative to the LSH threshold radius) there will be no net benefit from the index, as effectively an exhaustive search will need to be done to find the nearer neighbours in any case. Nevertheless, for the various tasks we have performed, LSH affords a speedup of orders of magnitude on the larger datasets we have examined.

3.2 Medium-scale

A medium-scale database for our purposes is one where a full scan of the database for a single query is not prohibitively expensive; this search is $O(N)$ in the number of feature vectors in the database, and limits for the audioDB software are of the order of 50000 tracks, with up to 10 feature vectors per second, giving an overall estimate of $\sim 10^8$ feature vectors in the database (of course, this estimate varies substantially with the available hardware and the constraints of the task: a real-time task on consumer electronics hardware will have a substantially smaller limit than a search which can be left to run over lunch on a cluster of compute servers).

Content-based searching of large collections as a primary user-access mechanism is never likely to be attractive, since standard metadata-based techniques are more efficient. More focused medium-sized collections offer the possibility of direct access via content-based searching for a variety of uses, particularly in professional contexts. An example is the problem of work-identification: OMRAS 2 has been approached by the curators of a publicly-maintained academic online classical music collection [5] with a request to help correct errors in their online metadata: they suspected that several items in the collection (over 420 tracks) had been mis-identified in the cataloguing process, although they did not know which ones, and putting this right promised to be a very costly exercise involving intensive listening by experts.

Using the method we investigate in this article, there is a straightforward solution to this problem which would, however, depend on the existence of a trusted, medium-scale 'authority' collection with reliable metadata. This would need to be a comprehensive set of recordings of at least the most popular works in the Western classical music canon (of the order of 10,000 to 20,000 tracks). Once an audioDB database has been built from the authority collection, it is a simple matter of performing

[3] `http://flickr.com/`

[4] such as the 'googlewhack': searching for particular terms which have low absolute numbers of matches in the database.

[5] The Culverhouse Collection `http://www.filmandsound.ac.uk/collections/culverhouse.shtml`

searches with each item in the 'problem' collection in turn and substituting correct information from the authority collection's metadata as needed. Such a mechanism, if made accessible online, could act as a 'cataloguer's assistant' for music libraries.

In these scenarios, or any that might involve legal aspects, the most important consideration is the unambiguous identification of the music being performed (*i.e.* the query track); this requires that the (usually single) item in the 'authority' collection being searched comes very near the top of the result list, i.e. high precision. On the other hand, the provision of substitutes from a medium-sized collection which need be more loosely 'similar' to a given query, or the construction of playlists, will be less stringent in terms of precision, but will benefit from high recall of likely candidate tracks.

A substantially different use of similarity-based search is the provision of a navigation interface to a sizeable collection of varied music, such as a recording label's collection. We have investigated the provision of useful interfaces to a database of 50,000 tracks from the Artists Without a Label catalogue [Magas et al., 2008], where the emphasis is on exploration of a collection of music by relatively unknown artists whose work is less likely to be discovered by using metadata. By presenting the user with tracks that contain at least one passage musically similar to a passage the user has selected, fresh and sometimes surprising discoveries are easily made within the collection, with the effect of greatly enhanced exposure for the artists concerned; in this scenario consistently achieving a high precision (measured against a carefully-considered ground truth) is less important and would arguably be counterproductive.

3.3 Small-scale

A small-scale database is one where a full, database cross-product, brute-force exhaustive search – $O(N^2)$ in the number of vectors in the database – is not prohibitively slow. As with the estimate of medium-scale databases in section 3.2, this is not an absolute definition of scale, but varies depending on the use case; a task requiring near-real-time response, such as playlist generation on a consumer music player device, will only work in this fashion on smaller databases than exhaustive searches on collections of music for research purposes, where a search can be left to run overnight and the results inspected the following day. Broadly, the audioDB software can perform this kind of search on databases of up to of the order of 5000 tracks, with about 10 feature vectors per second (so of the order of 10^6 to 10^7 feature vectors in the database, assuming tracks of about 4 minutes' duration).

When investigating such a small database, in general there is a very specific kind of similarity being sought. For example, the smallest database of music tracks to be investigated is a database with a single track, whereupon the search for similar time-series of features becomes a means of discovering structural segmentation of individual tracks, as in [Rhodes and Casey, 2007, Abdallah et al., 2006] and references therein. Indeed, the approach to computing distances between pairs of points in equations 1 and 2 has a direct connection to the music structure plots and similarity matrices of [Foote, 1999] and successive investigations. We have carried out initial investigations on the recognition of similar musical motives and themes using sequence matching (for example, within musical forms such as the typical verse/chorus of songs or the classical rondo or sonata-form movement), with encouraging preliminary results.

Beyond the single track, there are many small, coherent collections of musical interest; their coherence (effectively, some form of curation) means that there is a desire to investigate the collection in a focussed way. Collections on which we have worked include one of digitized 78 rpm discs from the Kings Sound Archive (see section 4 below); a collection of about 2000 tracks, representing most of the Chopin Mazurka discography (collected by the CHARM project[6]), and of particular

[6] `http://mazurka.org.uk/`

musical interest given the particular history of some of those recordings [Cook and Sapp, 2008, Casey, 2009]; a collection of ethnomusicological field recordings [Magas and Proutskova, 2009]; a set of Charlie Parker recordings; and many more besides. The audioDB software is flexible enough to accommodate a variety of strategies for dividing tracks into segments (by external segmenter, beat detector or constant time interval) and of acoustic feature (provided the distance metrics in equations 1 or 2 represent a 'semantic' distance between feature values) and so these varied databases and associated use cases can all be handled within the same framework.

In the majority of these investigations, recall of relevant results is highly important; for many, missing any relevant results can be strongly detrimental. Precision, however, remains important, particularly given the temporal nature of the medium: often, in these smaller use cases, in order to assess the performance of a system (or to decide how to proceed), if precision is low and recall high the user of the system is forced to listen to every result returned, taking time and quickly leading to fatigue in repeated searches.

This point, that investigations on even small collections would benefit from an algorithmic means to limit the retrieved results in a principled way (in order to minimize user dissatisfaction), motivates the experiment in section 4, where we adapt our previous work regarding similarity thresholds [Casey et al., 2008], which we there used to implement efficient, probabilistic indexing, to compute adaptively a relevance threshold for each query track.

4 Experiment

The experiment that we perform here is to find the effect of using a relevance threshold, previously motivated for the purpose of indexing a large collection, to improve the usability of the audioDB software and its associated search and retrieval paradigm on small to medium collections. The approach that we take is to consider first audioDB as a 'search engine', returning vectors in the database in order of distance from a set of query vectors. This ranked list in principle contains all the vectors in the database; we can cut this down by only considering the minimum distance between all vectors in the query and all vectors in each track in the database, but this still means that each query retrieves all database tracks. We therefore adopt a common search engine paradigm and cut off the retrieved list after the 10th result, roughly corresponding to the first page of results for online text-based searches, beyond which few typical users progress[7].

However, in doing this we will usually retrieve too many results; in many collections, and certainly in the one we consider below, there are fewer than 10 true positives per query track, and many will have none at all. In practice, then, the utility of the search tool will be increased if the user is required to listen to fewer false positives, if this can be achieved without paying too high a cost in overall recall. We therefore modify the retrieved list in two ways: the first is to cut the list of 10 retrieved documents off at a distance threshold estimated by sampling the database, exactly as in our indexing strategy detailed in 2.2. However, we expect this to have a significant impact on recall, as at least some tracks are likely to have relevant documents lying outside the whole-database threshold (see figure 1). The second modification is, for each query, also to sample the database beforehand to estimate the distribution of distances between database and that *specific* query (which can be done in a small amount of time compared with a search); a threshold estimated from this distribution, again exactly corresponding with the estimation for the indexing data structure, is then used to trim the retrieved results list down from the previous limit of 10. We then evaluate the effect of these two trimmings on the recall and the effective precision, defined as the proportion of true positives in the retrieved results.

[7] `http://www.iprospect.com/premiumPDFs/iProspectSurveyComplete.pdf` (accessed 21 Jan 2010)

The use-case that forms the basis of the experiment concerns the identification of the same work in multiple performances in a collection of historical recordings on 78 rpm discs. The identification of 'covers' of this type, where each artist is principally concerned with individual interpretation of a work (usually existing in a musical score form broadly common to each recording) rather than with changes in harmony or compositional structure, is of primary importance in the use of historical recordings for the study of performance history.

The 'similarity scale' for this use-case runs from multiple digitisations of the same 78 rpm disc, via performances which are identical in instrumentation and similar in tempo, through to re-scorings, transpositions and renditions in which structure is altered in minor ways (such as the performance of repeats). We are concerned here with the identification of the "same musical content" invariant to the historical recording process, which can, especially in the case of acoustic recordings (made before about 1927), severely affect timbral quality.

The collection is a test sample of 2018 digitised recordings of 78 rpm sides, averaging around 3 minutes in length, from the King's Sound Archive (KSA) maintained at King's College, London.[8] These are archived as stereo 16-bit wav files sampled at 48 kHz which we downsampled to 44.1 kHz mono as part of our feature-extraction process. The musical repertory is predominantly classical music, vocal and instrumental, with a roughly equal balance of orchestral and chamber music; a small number of jazz, sound-effect and spoken-word recordings are also included.

As 'ground truth', from supplied but unvalidated metadata we identified 88 works which appear more than once in the collection; one important issue is that these are often spread across more than one 78 rpm side (the maximum capacity for a 12" record side being about 4 minutes); for this reason, although side-breaks do not in general consistently occur at the same instant in a work, we limit our ground-truth 'covers' to sides which represent the same musical passages. This means that matches between a query-passage in, say, side 1 of a movement and a repetition of the same passage on side 2 are not listed as covers; this leads to some loss in precision.[9] The resulting list of 322 sides for which covers exist in the collection was used as audioDB queries; while the majority have just one cover, some items are present in up to six renditions (such as J. S. Bach's Toccata and Fugue in D minor, BWV 565, represented by several of the original organ version as well as some piano and orchestral arrangements). For each item in this list there were thus for comparative evaluation purposes between one and five 'relevant' tracks in the collection, considered as known items for the purpose of evaluation.

Because historical recordings tend to be somewhat inconsistent in terms of recorded pitch owing to the mechanical difficulty of maintaining consistent recording speed, we use 36-bin chromagrams with the aim of allowing matches where pitch varies between recordings. Although fine-pitch invariance can be easily achieved in the same manner as can deliberate musical transposition, by query-feature rotation, for the present experiment we restrict ourselves to same-pitch matches only. Again, there may be a slight inherent loss of precision on this account; in fact we know some covers are performed at different transpositions. Further, while it is possible that performances of the same work differ in tempo, we use fixed-width chroma vectors; it is possible that generating beat- or onset-synchronous chromagrams would improve performance by detecting these variants better, but this would be dependent on the accuracy of the onset detector, which would need to be assessed separately.

The overall retrieval performance of our system, using all 322 queries with the aim of retrieving as many covers as possible for each query is illustrated by the conventional 11-point interpolated precision/recall curve in figure 4. While we do not claim that performance on this known-item

[8] `http://www.kcl.ac.uk/schools/humanities/depts/music/res/soundarchive.html`

[9] An alternative, concatenating multiple-side recordings into complete movements - as on a modern CD re-issue - was not done for reasons of time.

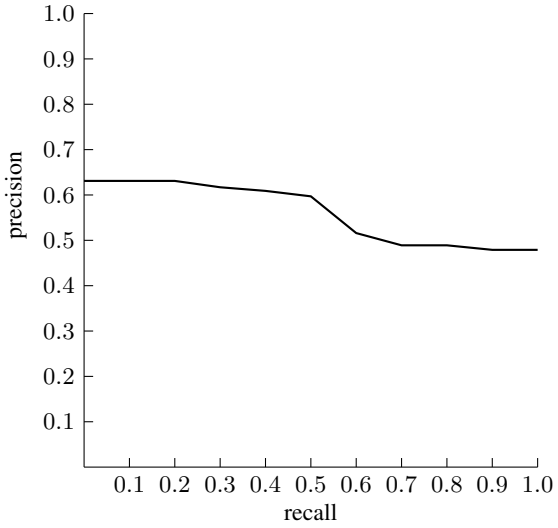

Figure 4: The overall precision-recall curves for our system, working with an exhaustive search over the entire database for the closest 20 seconds of audio in each track to any 20 seconds in the query track and ordering by distance, with ground truth for assessment generated from metadata.

Strategy	Retrieved	Relevant	Precision	Recall
1. Fixed rank (10 results)	3220	371	11.5%	65.5%
2. (1) and whole-database threshold	1667	298	17.9%	52.7%
3. (1) and per-query threshold	2235	352	15.7%	62.2%

Table 1: Retrieved and relevant results from the investigation described, with corresponding effective precision and recall figures, over the 322 tracks with at least one cover present in the database.

search task is 'good', the precision above 0.6 at low recall shows that we can fairly reliably find several covers at high positions in the ranked result lists – and using a more refined method for computing chroma features (with the rest of the experiment unchanged) yields a precision at low recall of above 0.85 and an F-score of about 0.8 [Crawford et al., 2010]. We discuss the overall performance of our system and what can be done to improve it in section 4.1.

The rest of this section examines the practical implications of using the system to search for unknown data – that is, finding 'new' covers at high rankings – and how the use of distance thresholding can make the use of retrieval systems like ours more practical and user-friendly. To begin with, first note that the overall precision-recall curve in figure 4 contains no information about query items with no known relevant results in the database (which must be considered in the unknown-item case).

Additionally, the utility of the search tool will be increased if the user is required to listen to fewer false positives, as long as this can be achieved without paying too high a cost of overall recall. The precision/recall curve shows the *best possible* effective precision at a given recall; irrelevant results beyond the last relevant one are not reflected in figure 4.

The results of our investigation are summarized in table 1. While this change in performance

from the per-query threshold may not seem like a significant improvement (the F-measure [van Rijsbergen, 1979], using these 'effective precision' numbers, increases from 9.78% to 12.5%), the change in absolute terms for the user of a search engine like this is large: each document that is no longer retrieved is 20s less music to listen to: a total of 5 hours' listening saved in this use case, at a small loss of recall.

4.1 Observations

While our per-track threshold estimation clearly reduces the number of non-relevant retrieved results, it does not eliminate them entirely; for some queries (such as a performance of R. Strauss' 'Morgen'[10], spectrogram displayed in the top panel of figure 5), the threshold performs as desired, returning only covers; for others, such as the Brahms sonata[11] whose spectrogram is displayed in the bottom panel of figure 5, the threshold is significantly larger than many of the minimum distances. In almost all of these latter cases, the large number of retrieved non-relevant results originate from a specific portion of the query track.

Our sampling method essentially assumes that all of the musical material is in some sense 'similar', not necessarily in acoustic or musical terms but in a statistical sense: drawn from the same underlying distribution; this is an assumption made in the estimation of minimum distances in [Casey et al., 2008]. However, in some of the tracks in the database in question, there are qualitatively different processes going on: for example, in the Brahms sonata displayed in figure 5 the majority of the piece is smooth, calm and slow-moving, but there are sections of fast, percussive piano activity (at 40s, 100s and 160s in this recording). These active regions, in the feature space and distance measure we are using, resemble broadband noise, and – crucially – resembles other percussive regions found in the database. Thus there are some statistically relevant (but musically non-relevant) documents in the database, as a result of a feature with insufficient discriminatory power. This is also responsible for the fact that we retrieve many documents when querying the database with a track with no known covers according to our ground truth.

Because of the nature of the tracks in the database, and the form of search that we are doing (single minimum-distance 'hit', exhaustive pairwise search of 20s sequences between tracks), our ground truth data will miss some pairs of tracks that are musically related: longer works divided into multiple sides (multiple tracks in our database) will often, though not always, share musical content judged similar with respect to our feature; those tracks are sometimes retrieved ahead of the corresponding side from a different recording of the same musical work.

In addition, a small but non-trivial number (about 20) of additional relevant results were identified from manual inspection of a subset of the results lists, indicating that our initial 'ground truth' derived from metadata was imperfect. In terms of the overall performance of the system, including these relevant results in our ground truth would have the effect of increasing the precision at low recall (and along with it the effective 'search engine' precision).

5 Conclusions

Motivated by consideration of user experience of searching for multiple performances of the same musical work in a collection, we have described how the effective precision, considering all the retrieved results in a system, can be increased, using the same statistical and geometric model as is used for indexing large-scale collections. We have demonstrated that this technique for limiting the

[10] audio file available at `http://images.cch.kcl.ac.uk/charm/liv/audio/flac/DB_1010_Cc_9888-2.flac`

[11] `http://images.cch.kcl.ac.uk/charm/liv/audio/flac/B_3098_Bb_16980-2.flac`

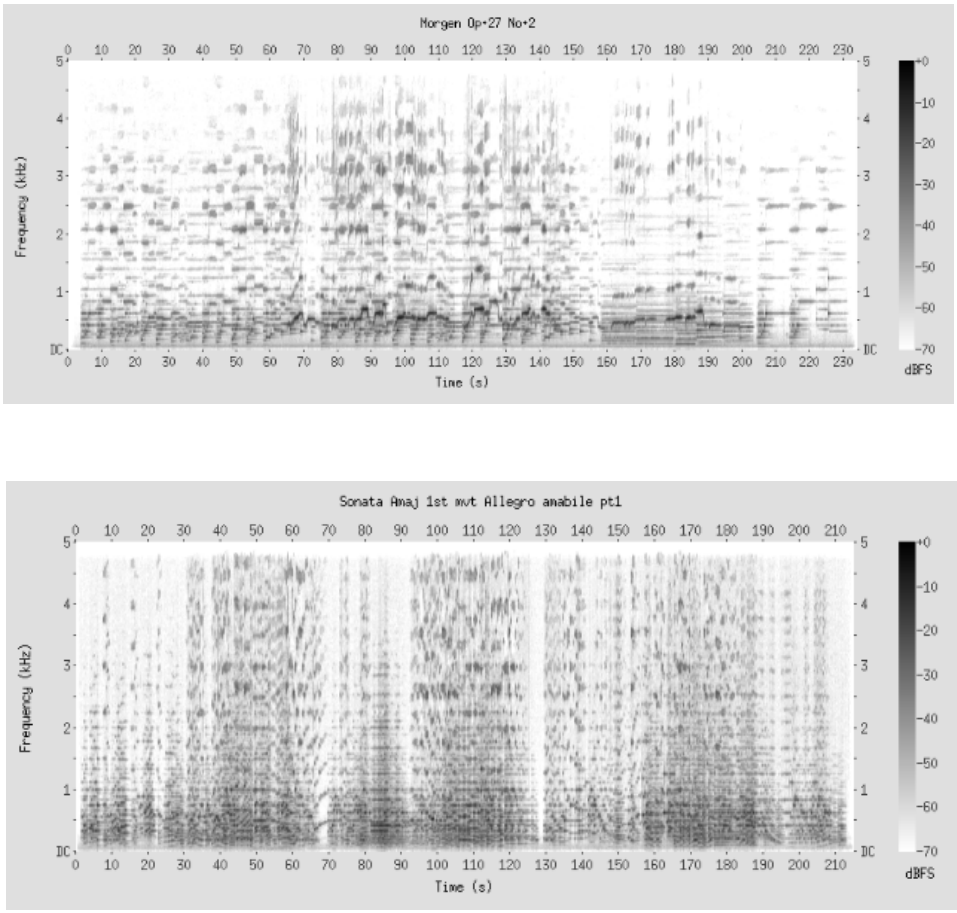

Figure 5: Spectra of R. Strauss, 'Morgen' (Elisabeth Schumann [sop], Isolde Menges [vln]), 1927 (top) and J. Brahms, Sonata in A major Op 100: I, 1 (Isolde Menges [vln], Harold Samuel [pfte]), 1929 (bottom). Observe the higher broadband noise component (darker grey background) displayed in the spectrum in the bottom panel.

retrieved set of results from searching a database can give large savings in terms of user time and energy, at a small cost in recall performance.

We have described a number of use cases for content-based similarity search, over different kinds collections of music at different scales. Consideration of the tradeoffs between precision and recall in each of these tasks allows us to use the same software framework to perform these tasks; the audioDB software has been demonstrated to scale to millions of multimedia items, each with thousands of vectors.

5.1 Future Work

The observation that, in some cases, there are many musically non-relevant documents retrieved under the per-track threshold for a given track – and in particular, the nature of the match found, being essentially a percussive section matching a percussive section – motivates the need for a better chroma feature, or alternatively a secondary feature to indicate how much of the chromagram came from 'harmonic' content in the audio; initial experiments [Crawford et al., 2010] in this direction demonstrate a dramatic improvement in retrieval performance using a more sophisticated note content estimator [Mauch and Dixon, 2010] rather than a simple FFT-based chromagram estimator.

More generally, a way of filtering for this failure mode (an outlier region matching a large proportion of the database) automatically would be valuable, as it is conceivable that there will be analogues of this phenomenon in other features, and it is only rarely desireable to return a significant fraction of a database in response to a query.

We have demonstrated the value of an adaptive threshold, working on a small database, at increasing the effective precision. There are use cases at medium and large scale where the effective precision of a search is likewise important; it is therefore important to be able to construct useful indexes over the database for variable thresholds. This must be done with some care; it is not as simple as building multiple indexes at different thresholds, because the size of such a naïve index would increase vastly as the threshold increases; we expect to have to perform some prefiltering of points to index or to consider hierarchical indexing schemes.

Finally, we anticipate that consideration of further use cases will motivate extensions and enhancements to the audioDB software described here; both the core functionality and applications using it will continue to be maintained and developed in its repository at `http://source.omras2.org/svn/audioDB`.

Acknowledgments

We are grateful to Daniel Leech-Wilkinson, Andrew Hallifax and Martin Haskell for providing files from the Kings Sound Archive collection, and to Denzyl Feigelson for providing media from the Artists Without a Label collection. Some of the larger-scale investigations we describe were performed with Malcolm Slaney on the media collections at Yahoo! Inc. We have benefited greatly from conversations with Michael Jewell, and we thank the anonymous reviewers for their perceptive comments and constructive suggestions. This work was supported by EPSRC (grant number EP/E02274X/1).

References

[Abdallah et al., 2006] Abdallah, S., Sandler, M., Rhodes, C., and Casey, M. (2006). Using duration models to reduce fragmentation in audio segmentation. *Machine Learning*, 65:485–515.

[Andoni and Indyk, 2006] Andoni, A. and Indyk, P. (2006). Near-Optimal Hashing Algorithms for Near Neighbor Problem in High Dimensions. In *Proceedings of the Symposium on Foundations of Computer Science*, pages 459–468, Berkeley, CA, USA. IEEE Computer Science.

[Baluja and Covell, 2008] Baluja, S. and Covell, M. (2008). Learning to hash: forgiving hash functions and applications. *Data Mining and Knowledge Discovery*.

[Broder et al., 1997] Broder, A., Glassman, S., Manasse, M., and Zweig, G. (1997). Syntactic clustering of the web. In *Selected papers from the sixth international conference on World Wide Web*, pages 1157–1166.

[Byrd, 2007] Byrd, D. (2007). A similarity scale for content-based music ir. `http://www.informatics.indiana.edu/donbyrd/MusicSimilarityScale.HTML`.

[Casey, 2001] Casey, M. (2001). MPEG-7 Sound Recognition Tools. *IEEE Transactions on Circuits and Systems Video Technology. Special issue on MPEG-7*.

[Casey, 2009] Casey, M. (2009). Audio Tools for Music Discovery. In Crawford, T. and Gibson, L., editors, *Modern Methods for Musicology: Prospects, Proposals and Realities*, Digital Research in the Arts and Humanities, pages 127–135. Ashgate.

[Casey et al., 2008] Casey, M., Rhodes, C., and Slaney, M. (2008). Analysis of Minimum Distances in High-Dimensional Musical Spaces. *IEEE Transactions on Audio, Speech and Signal Processing*, 16(5):1015–1028.

[Casey and Slaney, 2006a] Casey, M. and Slaney, M. (2006a). Song Intersection by Approximate Nearest Neighbour Searching. In *Proceedings of the International Symposium on Music Information Retrieval*.

[Casey and Slaney, 2006b] Casey, M. and Slaney, M. (2006b). The Importance of Sequences in Music Similarity. In *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, volume V, pages 5–8, Toulouse, France.

[Cook, 2007] Cook, N. (2007). Performance Analysis and Chopin's Mazurkas. *Musicae Scientiae*.

[Cook, 2010] Cook, N. (2010). The Ghost in the Machine: Towards a Musicology of Recordings. *Musicae Scientiae*.

[Cook and Sapp, 2008] Cook, N. and Sapp, C. (2008). Purely coincidental? Joyce Hatto and Chopin's Mazurkas. `http://www.charm.rhul.ac.uk/content/contact/hatto_article.html`.

[Crawford et al., 2010] Crawford, T., Mauch, M., and Rhodes, C. (2010). Recognising Classical Works in Historical Recordings. In *Proceedings of the International Symposium on Music Information Retrieval*. (accepted).

[Datar et al., 2004] Datar, M., Indyk, P., Immorlica, N., and Mirrokni, V. (2004). Locality-Sensitive Hashing Scheme Based on p-stable Distributions. In *Proceedings of the Symposium on Computational Geometry*, pages 253–262, Brooklyn, New York, USA. ACM.

[d'Inverno et al., 2011] d'Inverno, M., Casey, M., Rhodes, C., and Jewell, M. (2011). Content-based search for time-based media. in preparation.

[Ellis, 2007] Ellis, D. (2007). Beat tracking by dynamic programming. *J. New Music Research, Special Issue on Beat and Tempo Extraction*.

[Ellis and Poliner, 2007] Ellis, D. and Poliner, G. (2007). Identifying 'Cover Songs' with Chroma Features and Dynamic Programming Beat Tracking. In *Proc. Int. Conf. on Acous., Speech, & Sig. Proc. ICASSP-07*, pages pp.IV–1429–1432, Hawai'i.

[Flexer et al., 2005] Flexer, A., Pampalk, E., and Widmer, G. (2005). Hidden markov models for spectral similarity of songs. Technical report, Österreichisches Forschungsinstitut fr Artificial Intelligence.

[Foote, 1999] Foote, J. (1999). Visualizing Music and Audio using Self-Similarity. In *ACM Multimedia (1)*, pages 77–80.

[Grauman and Darrell, 2004] Grauman, K. and Darrell, T. (2004). Fast Contour Matching Using Approximate Earth Mover's Distance. *Procdeedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

[John, 2006] John, J. (2006). Pandora and the music genome project. *Scientific Computing*.

[Levy and Sandler, 2006] Levy, M. and Sandler, M. (2006). Lightweight measures for timbral similarity of musical audio. In *Proceedings of the 1st ACM workshop on Audio and music computing multimedia*, pages 27–36, Santa Barbara, California, USA.

[Logan and Salamon, 2001] Logan, B. and Salamon, A. (2001). A Music Similarity Function Based on Signal Analysis. *Proceedings of the IEEE International Conference on Multimedia and Expo*.

[Magas et al., 2008] Magas, M., Casey, M., and Rhodes, C. (2008). mHashup: fast visual music discovery via locality sensitive hashing. In *ACM SIGGRAPH New Technology Demos*, LA, USA.

[Magas and Proutskova, 2009] Magas, M. and Proutskova, P. (2009). A location-tracking interface for ethnomusicological collections. In *Workshop on Exploring Musical Information Spaces*, pages 12–17, Corfu, Greece.

[Mauch and Dixon, 2010] Mauch, M. and Dixon, S. (2010). Approximate Note Transcription for the Improved Identification of Difficult Chords. In *Proceedings of the International Symposium on Music Information Retrieval*. (accepted).

[Müller, 2007] Müller, M. (2007). *Information Retrieval for Music and Motion*. Springer Verlag.

[Rauber et al., 2002] Rauber, A., Pampalk, E., and Merkl, D. (2002). Content-based Music Indexing and Organization. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR02)*, pages 409–410, Tampere, Finland.

[Rhodes and Casey, 2007] Rhodes, C. and Casey, M. (2007). Algorithms for determining and labelling approximate hierarchical self-similarity. In *International Conference on Music Information Retrieval*, pages 41–46, Vienna.

[Serra and Gomez, 2007] Serra, X. and Gomez, E. (2007). A Cover Song Identification System Based on Sequences of Tonal Descriptors. *Proceedings of the International Symposium on Music Information Retrieval*.

[Slaney, 2009] Slaney, M. (2009). Image Results using audioDB. personal communication.

[Tzanetakis and Essl, 2001] Tzanetakis, G. and Essl, G. (2001). Automatic Musical Genre Classification Of Audio Signals. In *IEEE Transactions on Speech and Audio Processing*, pages 293–302.

[van Rijsbergen, 1979] van Rijsbergen, C. V. (1979). *Information Retrieval*. Butterworth.

[Wang, 2003] Wang, A. (2003). An Industrial Strength Audio Search Algorithm. In *Fourth International Symposium on Music Information Retrieval*, pages 7–13, Baltimore.

[Wang, 2006] Wang, A. (2006). The Shazam music recognition service. *Communications of the ACM*, 49(8):44–48.