# QUERYING IMPROVISED MUSIC: DO YOU SOUND LIKE YOURSELF?

**Michael O. Jewell, Christophe Rhodes and Mark d'Inverno**
Department of Computing
Goldsmiths, University of London
New Cross, London, SE14 6NW
United Kingdom
{m.jewell,c.rhodes,dinverno}@gold.ac.uk

## ABSTRACT

Improvisers are often keen to assess how their performance practice stands up to an ideal: whether that ideal is of technical accuracy or instant composition of material meeting complex harmonic constraints at speed. This paper reports on the development of an interface for querying and navigating a collection of recorded material for the purpose of presenting information on musical similarity, and the application of this interface to the investigation of a set of recordings by jazz performers. We investigate the retrieval performance of our tool, and in analysing the 'hits' and particularly the 'misses', provide information suggesting a change in one of the authors' improvisation style.

## 1. INTRODUCTION

Query-by-Example systems for musical search offer the promise of rich interaction for their users with collections of music. The purpose of a search can be goal-driven or exploratory, while the musical content being searched can be highly focused (as in a curated collection in a sound archive), heterogenous and largely known to the user (a personal collection on a user's personal music player) or heterogenous and largely unknown (an online music vendor's catalogue). The first Query-by-Example systems [8, 16] stored their collections in MIDI format; they admitted audio queries (hence the 'Query-by-Humming' term in the Music Information Retrieval community), and one of the technical hurdles in those systems was a sufficiently accurate transcription of the hummed input – and a search relevance filter that could account for error from imperfect human humming as well as from imperfect transcription algorithms. This mode of interacting with a collection of MIDI-encoded music is available over the web at *Musipedia* [1] .

However, for usable systems, Query-by-Example needs to be augmented by some means of navigating the collection; typically, that navigation mode is specialized to the particular use case enviseaged (and details of the collection being investigated); there exist numerous interfaces and visualisations of collections (such as [9, 12, 17, 18]) and their use for music discovery has been discussed in tutorial sessions [2] .

Achieving intuitive navigation through collections requires some kind of notion of similarity (of which there are many kinds [2]); for systems using primarily content-based information, this means that the audio features or descriptors must encode not only identity but one or more similarity relationships at some level of specificity. Viewed from this perspective, systems based on audio features for classifying and clustering musical tracks [10, 18] or segments [1] are Query-by-Example systems, just as are more modern implementations of the original idea (*e.g.* [7]).

In our work, we are interested in both small-scale and large-scale collections, and in particular at allowing the user to search for and retrieve fragments of tracks (rather than track-to-track or fragment-to-track matches); in principle if given a 5-second audio snippet as a query, we consider all similarly-sized segments in the database – up to some reasonable granularity – as potential matches. This means that collections of even a small number of tracks have a large number of effective database entries to be considered. Achieving fast search through large databases of musical content has been considered in a few applications [14, 15], including the ability to search for specific content within a track in a manner which can still be implemented efficiently [3] and can be generalized [6].

In this paper, we describe a practical use-case for exploratorily searching for fragments of audio by similarity within a small collection. In section 2, we describe in more detail the use case in question; in section 3, we describe how the technology we have developed can meet this need. Our preliminary experiments are reported in section 4, and we draw conclusions and suggest further work in section 5.

## 2. CASE STUDY

It is often the case that when amateur and semi-professional musicians hear themselves play they cringe at just how

[1] http://www.musipedia.org/

[2] *e.g.* http://musicviz.googlepages.com/home

far away they are from being like the professional heroes that have influenced them. There is a sense of 'I wish I could sound a bit less like me and more like someone *really* good'. We propose to build a tool that provides a general framework for the analysis of performance, allowing performers to both self-analyse and also discover how they relate to their influences.

Performers are often concerned with knowing if their playing has improved in over a time period; whether they can learn about their approach and technique from how professional musicians play particular phrases; or whether they play differently depending on the instrument, event, ensemble, etc.

As such, we are interested in building a tool that enables performing musicians to analyse certain performance characteristics. We propose an iterative development cycle where we increase the scope incrementally in terms of what performance characteristics may be considered, the range of media, the range of extractors, the type of searches (point, track, catalogue) and the options which we make available to a user in the interface. The planned functionality includes, but is not limited to, investigating the following queries:

1. How do performance characteristics of a musician develop over time?

2. How does the performance context (*e.g.* home recording, studio recording) affect performance characteristics?

3. How does the ensemble (*e.g.* solo, duo, trio, big band) affect performance characteristics?

4. How does the type of instrument (*e.g.* in the case of piano, grand, upright, electric) affect performance characteristics?

5. How do certain performance characteristics compare with great musicians?

6. How do performance characteristics develop through a single piece performance?

One of the authors is a reasonable jazz pianist (he has received good reviews in the UK *Guardian* and *Observer* newspapers), so we chose to focus on jazz piano performance, with our ultimate goal as being able to ask the question: 'How much of a performer's improvisation is genuinely improvised, and how much is made from stock patterns?'

Many jazz musicians can come up with phrases or 'licks' that work over chord changes but it is only the greats who can actually approach improvisation as 'instant composition': where what they play is not only appropriate to the sequence but an original passage of notes. The co-author would ideally like to find out where the stock patterns arise in their playing in order to remove them to free up space for more creative improvisation.

## 3. TECHNOLOGY

### 3.1 Similarity Measurement

The necessary functionality for our application is the insertion and storage of numerical audio feature information extracted from tracks, and their subsequent searching for similarity. These two aspects are illustrated in figure 1: in the left panel, we schematically show a track which has had $d$-dimensional audio features extracted for a number of regions of audio. Subsequently, a user wishes to search using a query of region length $sl$, so successive feature vectors are concatenated (illustrated by the arrows in the left panel) to arrive at shingled [4] feature vectors (right panel). These shingled feature vectors are then compared against the query by summing squared Euclidean distances, and a retrieved list is assembled.

### 3.2 Interface

iAudioDB is an application developed for Mac OSX in Objective C which provides an intuitive user interface for the creation and exploration of feature databases. As such, it binds directly to the audioDB libraries for creation and querying, and employs Sonic Annotator to extract features from files provided by the user.

Usage of iAudioDB follows a straight-forward process, with the interface providing intuitive abstractions to parameters where possible. The first step is to create the database itself, which is achieved via the interface in Figure 2. The user is prompted for the feature they wish to extract, which corresponds directly to the VAMP plugin [3] which is used with Sonic Annotator, and then a selection of parameters which are database-specific. The first two, 'Max Tracks' and 'Max Length' correspond to the number of audio files the user expects to import into the database and the maximum length in seconds of those tracks. The hop size and window size, equivalent to the step and block size detailed above, are used in conjunction with these values to determine the initial size (in bytes) of the database. Furthermore, the chosen parameters are stored alongside the database to remove the need to enter the settings at the import stage.

Once created, the user imports any audio files, both ground truth and queries. Aside from a standard file dialogue, there is no interface for this, as all parameters required are obtained at creation time. Multiple files may be selected, and progress is indicated as files are imported. At this stage, Sonic Annotator extracts feature information as n3-serialized RDF, which is then imported into the database. Future increments of the software will see it acting as a VAMP host, allowing the use of extractors via a native library. The filenames of the audio files are preserved alongside the unique keys of the tracks in the audioDB instance, thus easing the playback process.

The query process again has an intuitive user interface, shown in Figure 3. The user selects the audio file they wish to use as the query, and from this the length is determined. This length is displayed in the Query Length fields in units

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

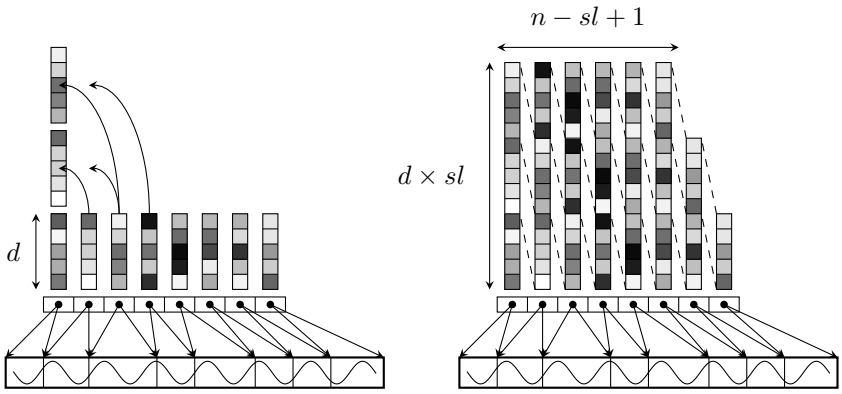


**Figure 1**. Illustration of the construction of concatenated or shingled feature vectors for our search. Note that while in principle this construction can be done for features over audio regions with temporally-varying extent and step (adjusted to the local tempo), in this paper the step size and block size were kept constant.

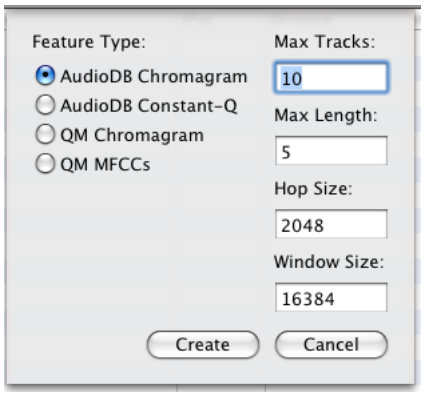


**Figure 2**. Creating a new database in iAudioDB. The feature extractor is chosen on the left-hand side, while parameters related to the database and the extractor are on the right.

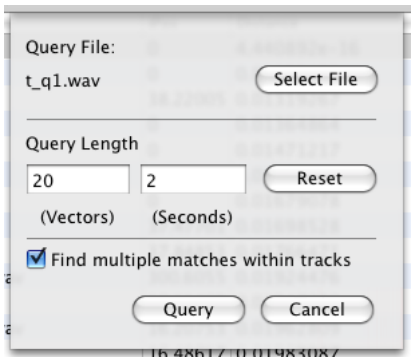


**Figure 3**. Querying a database in iAudioDB. The query length is generated dynamically from the query audio file, and may then be customized by the user.

of Vectors and Seconds, and both of these fields may be customized by the user to vary the length of the query. The fields are dynamically updated, so a change to the seconds value reflects instantly in the vectors value. If desired, the length may also be reset to the full duration of the query file. Finally, the user may opt to locate multiple matches of a query within the corpus, or to only determine the best match per track.

Once queried, results are displayed in the main application window (see Figure 4). By default, these are sorted by ascending distance values, but this may be customized by clicking the column headers. The other columns are, from left to right, a visual indicator of the closeness of the match (though this varies depending on extractor, so should not be used for comparison), the unique key within the audioDB instance, and the position in seconds at which the query occurs in the track. Results may be played in isolation from the match position, or synchronized with the original query.

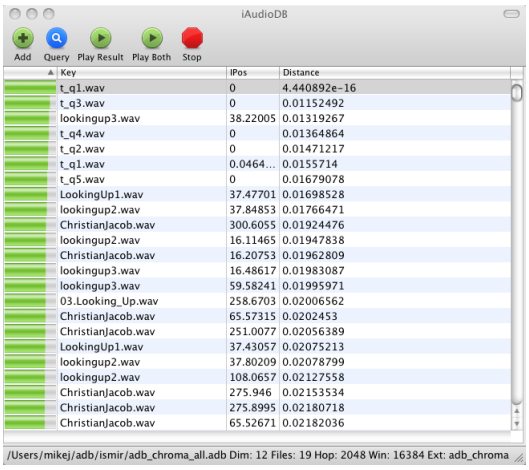


**Figure 4**. Results generated from an iAudioDB query.

## 4. FEATURE SPACE INVESTIGATION

The first step in this investigation was to turn our attention to one track and to focus on a single element of the tune. This would at least give us some ground truths whereby we could start to map out a method for getting to our ultimate goal. The track we chose was *Looking Up*, written by the late great jazz pianist Michel Petrucciani. Specifically, we chose the following performance scenarios that would in time enable us to look at all the issues of our case study:

1. The co-author at home using the internal microphone of a laptop recorded three versions of *Looking Up* solo on a Kawai grand piano in an informal setting. These were stored as stereo 44100Hz WAV files.

2. The co-author again, in the same session, but recording three versions of two other tracks – *Ambleside Days* by John Taylor and *My Romance* by Rogers and Hart. (The significance of recording these will become clear later.) As above, these were stored as 44100Hz WAV files.

3. The co-author again, but recorded in a studio context, in trio ensemble and on a Technics electric piano ten years previously.

4. The composer of *Looking Up* and an influence of the co-author, Michel Petrucciani recorded in a concert on a solo grand piano.

5. Michel Petrucciani again but in a band context on a grand piano in a live setting.

6. Another well-regarded pianist and influence of the co-author, Christian Jacob in a trio ensemble, a recording studio with a grand piano.

To begin our iterative development cycle for this application, we focus on one specific phrase in the tune Looking Up (the very first phrase, an 8-note Mixolydian scale in E). This run appears several times in the piece, though the frequency and positions vary on a per-recording basis. The co-author recorded this phrase five times in the same setting as 1 and 2 above to build a library of different queries. These query tracks were played at an even tempo, with no missing or muffled notes.

From this set of tracks three feature databases were built, all with a step size of 2048 samples (0.046s) and a block size of 16384 (0.372s):

1. An MFCC feature database with 20 cepstral coefficients.

2. A constant-$Q$ feature database with 12 bins per octave, a minimum frequency of 65.4064Hz, and a maximum frequency of 1046.5Hz.

3. A chromagram database with the same bins per octave and frequency range as the constant-$Q$ database, and a sum of squares accumulation method.

| Track | Position (s) | Missing Notes | Muffled Notes | Rhythm Alterations | Chord Additions | Sustain Pedal |
|---|---|---|---|---|---|---|
| LU1 | 15 | | x | | | |
| | 37 | | | | | |
| | 59 | | x | | | |
| | 86 | | x | | | |
| | 162 | x | | | | x |
| LU2 | 15 | x | | | | x |
| | 37 | | | | | |
| | 59 | | | | x | |
| | 107 | x | | x | | |
| LU3 | 16 | x | | x | | x |
| | 38 | | | | | |
| | 59 | | | | | |
| | 80 | | | | x | |

**Table 1**. Locations and comments of fragments corresponding to our queries in the three single-take recordings through a laptop microphone.

The 3 *Looking Up* tracks were examined to locate the positions of the queried tune and thence act as a ground truth. The resultant locations and notes on these instances are shown in Table 1.

Each feature database was then queried with each of the 5 recorded queries, with a maximum length of 20 vectors (1.3s). The recordings of *My Romance* and *Ambleside Days* were used as a boundary, with results examined up to the first match of a track in this set, and duplicated results were discarded. From this set, it was possible to determine those which matched the segments in Table 1 and those which did not. Note that with queries of this length, and with the audio features extracted every 2048 audio samples, there are over 50,000 candidate matching points in our 9-track database; the fact that we are searching for fragments of track rather than whole tracks enlarges the problem.

The mean precision and recall values from these queries can be seen in Table 2, and it is immediately apparent that chromagram features produce the most useful results. While the precision is not as high as that of the constant-$Q$ database the recall is significantly improved, and thus of most benefit to this case study, where the user is looking for a variety of similar matches rather than a small number of exact matches.

Within the results, some notable differences between feature performance were present. Riff instances with muffled notes (15s, 59s, and 86s in *Looking Up* 1) were located in 73% of queries using the chromagram database, 47% using constant-$Q$, and 20% using MFCCs. Instances with rhythm alterations (107s in *Looking Up* 2 and 16s in *Looking Up* 3) were found in 100% of queries using the chromagram database, 50% using constant-$Q$ (matching the *Look-*

| Feature | Precision | Recall | F-Score |
|---|---|---|---|
| MFCC | 0.89 | 0.29 | 0.44 |
| Constant-$Q$ | 1.00 | 0.57 | 0.73 |
| Chromagram | 0.97 | 0.83 | 0.89 |

**Table 2**. Average precision, recall, and balanced F-score for our queries against recordings in the same recording environment.

*ing Up* 2 instance throughout), and none using MFCCs. Finally, the chromagram and constant-$Q$ databases were more resilient to missing notes, matching 75% of the cases in the former and 40% in the latter, with MFCC matching 10%. Interestingly, the riff at 162s in *Looking Up* 1 was entirely unmatched, possibly due to the number of notes missing from the melody.

As a second case, 4 performances of *Looking Up* by professional jazz pianists were added to the databases: a trio studio recording (MDI), a solo piano studio recording (CJ), a live band recording (MP(B)), and a live solo piano recording of the same (MP(S)). The ground truth for this collection is shown in Table 3, and the precision/recall means for the MFCC and chromagram databases in Table 4.

As before, chromagrams provided the most useful results, with a comparatively high mean precision and recall. The CJ recording obtained a mean recall of 1.00 and a mean precision of 0.72, while the MDI recording resulted in a mean recall of 0.43 and a mean precision of 1.00. MP(B) and MP(S) both obtained low recall (0.27 and 0.32 respectively) and good precision (1.00 and 0.78 respectively). Both MP(B) and MP(S) were recorded in a live setting, which may suggest the distance from the query, but notably the queries which didn't match often occurred in locations where the sustain pedal was employed. The CJ recording, while in a studio, was classically precise in terms of note velocity, timing, and consistency, with no sustain pedal employed during the riff instances. The MDI recording only missed matches across all queries when the sustain pedal was used. Further investigation will examine this characteristic more closely.

## 5. CONCLUSIONS

Our study, while still at a preliminary stage, is promising: we can achieve good precision and recall for fragments of audio, both for queries recorded under the same conditions as the test database and for queries recorded on consumer hardware against a database of professional studio recordings.

Treated as a pure retrieval task, recall performance is perhaps not as good as might be desired; our observation is that our audio features are not sufficiently robust to the kinds of difference that arise in practice between the query and the matches desired by our userbase. Enhancements in this area would be to incorporate more aspects of desired invariance [11] into our feature, such as for example: constant-$Q$ translations or chroma rotations to model transposition invariance; and beat-based analysis windows

| Track | Position (s) | Missing/Altered Notes | Muffled Notes | Rhythm/Tempo Alterations | Chord Additions | Sustain Pedal |
|---|---|---|---|---|---|---|
| MDI | 9 | | | | | x |
| | 37 | | | | | x |
| | 64 | | | | | x |
| | 258 | | | | | |
| | 285 | | | | | x |
| | 310 | | | | | |
| CJ | 15 | | | | | |
| | 40 | | | | | |
| | 66 | | | | | |
| | 250 | | | | | |
| | 276 | | | | | |
| | 300 | | | | | |
| MP(B) | 17 | x | | | | |
| | 43 | | | x | | |
| | 70 | | | | x | |
| | 342 | x | | | | x |
| | 368 | | | x | | x |
| | 394 | | | | x | x |
| MP(S) | 32 | x | | x | | |
| | 65 | | | x | | x |
| | 92 | | | x | x | |
| | 202 | | | x | | x |
| | 227 | | | | x | |

**Table 3**. Locations and comments of fragments corresponding to our queries in the three professional-quality recordings.

| Feature | Precision | Recall | F-Score |
|---|---|---|---|
| MFCC | 0.77 | 0.04 | 0.08 |
| Chromagram | 0.80 | 0.51 | 0.62 |

**Table 4**. Average precision, recall, and balanced F-score for our queries against the professional, studio recordings.

to incorporate tempo invariance. Because we desire to allow our users to search large databases of audio as well as small ones, we wish to avoid providing invariants using methods scaling worse than linearly with the database size (such as dynamic time warping [14, Chap. 4] for tempo invariance).

However, these invariants are not desired for all applications of our searching technology; in particular, when exploring a corpus for changes in stylistic aspects of performance, it is important for sufficiently different renditions *not* to match a query. The success of our initial experiment in this respect is the observation that one apparently robust characteristic of the ground truth matches in the professionally-recorded corpus that are not found by our current features is that they are executed in the recordings with the sustain pedal on (which has previously been identified as a problem in other MIR tasks [5, 10]); designing a feature to cope with this would be very desirable, but the distinction between the performance practice with sustain and without was new information to our co-author pianist.

We expect to go through several more design-and-test iterations for our implementation of a user interface; known currently-missing features include: a quasi-live interface for rapid, experimental search; and a means for navigation between regions [1, 13]. However, we believe that what we have already developed is good enough for a sophisticated user to be able to explore his own performance practice, or for a composer to use as a thesaurus. The software will be available to download from the OMRAS website [4] shortly after publication, and we welcome feedback from users.

## 6. REFERENCES


[1] Dominikus Baur, Tim Langer, and Andreas Butz. Shades of Music: Letting Users Discover Sub-song Similarities. In *Proc. ISMIR*, pages 111–116, 2009.

[2] Donald Byrd. A Similarity Scale for Content-Based Music IR. Available at `http://www.informatics.indiana.edu/donbyrd/MusicSimilarityScale.html`, 2008.

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

[4] M. Casey and M. Slaney. The Importance of Sequences in Music Similarity. In *Proc. ICASSP*, volume V, pages 5–8, 2006.

[5] Arshia Cont. Realtime Multiple Pitch Observation using Sparse Non-negative Constraints. In *Proc. ISMIR*, 2006.

[6] Mark d'Inverno, Christophe Rhodes, Michael Casey, and Michael Jewell. Content-based Search for Time-based Media. in preparation.


[4] `http://www.omras2.org`


[7] Alexander Duda, Andreas Nürnberger, and Sebastian Stober. Towards Query by Singing/Humming on Audio Databases. In *Proc. ISMIR*, pages 331–334, 2007.

[8] Asif Ghias, Jonathan Logan, David Chamberlin, and Brian C. Smith. Query by humming: musical information retrieval in an audio database. In *Proc. ACM Conference on Multimedia*, pages 231–236, 1995.

[9] Masataka Goto and Takayuki Goto. Musicream: New Music Playback Interface for Streaming, Sticking, Sorting, and Recalling Musical Pieces. In *Proc. ISMIR*, pages 404–411, 2005.

[10] Maarten Grachten and Gerhard Widmer. Who is who in the end? Recognizing pianists by their final ritardandi. In *Proc. ISMIR*, pages 51–56, 2009.

[11] Kjell Lemström and Geraint A. Wiggins. Formalizing invariances for content-based music retrieval. In *Proc. ISMIR*, pages 591–596, 2009.

[12] M. Magas, M. Casey, and C. Rhodes. mHashup: fast visual music discovery via locality sensitive hashing. In *SIGGRAPH '08: ACM SIGGRAPH 2008 new tech demos*, pages 1–1, Los Angeles, 2008. ACM.

[13] Michela Magas and John Wood. A More User-Centric Approach to the Retrieval of Music Data. submitted to JNMR, 2010.

[14] Meinard Müller. *Information Retrieval for Music and Motion*. Springer-Verlag, Berlin Heidelberg, 2007.

[15] Dominik Schnitzer, Arthur Flexer, and Gerhard Widmer. A Filter-and-Refine Indexing Method for Fast Similarity Search in Millions of Music Tracks. In *Proc. ISMIR*, pages 537–542, 2009.

[16] Yuen-Hsien Tseng. Content-based retrieval for music collections. In *Proc. ACM SIGIR*, pages 176–182, 1999.

[17] George Tzanetakis, Andreye Ermonlinskyi, and Perry Cook. Beyond the Query-By-Example Paradigm: New Query Interfaces for Music Information Retrieval. In *Proc. ICMC*, pages 177–183, 2002.

[18] Hugues Vinet, Perfecto Herrera, and François Pachet. The CUIDADO Project. In *Proc. ISMIR*, pages 197–203, 2002.