Matching Techniques for Large Music Databases

Alexandra Uitdenbogerd and Justin Zobel
Department of Computer Science, RMIT University
GPO Box 2476V, Melbourne 3001, Australia
{alu,jz}@cs.rmit.edu.au

Abstract

With the growth in digital representations of music, and of music stored in these representations, it is increasingly attractive to search collections of music. One mode of search is by similarity, but, for music, similarity search presents several difficulties: in particular, deciding what part of the music is likely to be perceived as the theme by a listener, and deciding whether two pieces of music with different sequences of notes represent the same theme. In this paper we propose a three-stage framework for matching pieces of music. We use the framework to compare a range of techniques for determining whether two pieces of music are similar, by experimentally testing their ability to retrieve different transcriptions of the same piece of music from a large collection of MIDI files. These experiments show that different comparison techniques differ widely in their effectiveness; and that, by instantiating the framework with appropriate music manipulation and comparison techniques, pieces of music that match a query can be identified in a large collection.

1 Introduction

Historically, the principal representations of music have been by written score and, for the last century, by recorded performance. In recent years a third form has also become important: the MIDI format, a digital representation of music designed for replay through electronic instruments. The three formats are not interchangeable: a MIDI file cannot capture the subtlety of a human performance, nor does it include the directives such as time given in a written score. In contrast to a score, however, MIDI includes duration and intensity of each note. Music is increasingly being transcribed into the MIDI format, and with the growth of the internet tens of thousands of MIDI transcriptions are publicly available.

Composers and musicians could search for pieces of music by their theme, for reasons such as copyright checks and interest in music they can recall by theme or melody alone. These searches are currently only supported by incipits of major themes in music library catalogues. However, it is reasonable to suppose that digital representations of music, and searching techniques for music collections, will eventually replace these catalogues, and public-domain transcriptions of music are likely to be an important component of such collections. In recent years there has been considerable interest in retrieval from such music collections, with for example prototype query-by-humming systems already implemented by Kageyama et al. [16], Ghias et al. [10], McNab et al. [18], and Borchers and Mühlhauser [2].

In this paper our aim is to identify a practical way of locating particular melodies in collections of MIDI files. The melodies and themes are not clearly delineated in these files as each consists of several tracks, many instrument types can be involved, and some of these can play greater than one note simultaneously—that is, it is not immediately obvious what sequence of notes in the music constitutes the melody. Previous systems working with MIDI files, including those listed above, have typically used either melody contour or short sequences of notes for matching melodies, but have been based on collections such as folk melodies that consist of a single non-chordal instrument. With the richer MIDI files available for many forms of music, it is not immediately obvious how these methods would be applied or how effective they would be.

We propose a three-stage framework for matching pieces of music. *Melody extraction* is used to reduce a MIDI representation to a sequence of non-chordal notes [25]. *Standardisation* is used to rewrite the sequence in a standard form that preserves the "feel" of the melody but eliminates performance-specific characteristics; melody extraction and standardisation are applied to both query and corpus. A *similarity function* is then used to numerically score each piece in the collection against the query.

Using our framework we have experimentally compared a range of extraction, standardisation, and similarity techniques, by exhaustively combining them and measuring their effectiveness on a test data set of just over 10,000 MIDI files garnered from the internet. Amongst these files there are many pieces of music with multiple distinct transcriptions; our experiments

test the ability of matching techniques to correctly identify different transcriptions of the same piece of music. The experiments show that, given appropriate music manipulation and comparison techniques, our framework can be used to effectively retrieve matching pieces of music from a large collection.

The experiments also show that choice of extraction, standardisation, and similarity technique all have a dramatic impact on effectiveness. For example, similarity based on local alignment is much superior to that based on n-grams; and standardisation should preserve pitch intervals between notes, because a simple pitch contour does not allow effective retrieval.

2 Music databases

Music publishing is a major industry: each year over 10,000 new albums of recorded music are released and over 100,000 new pieces are registered for copyright. Lawyers need to know that each registered piece is indeed new; composers could search music databases to confirm that their inspiration is novel, and to explore forms of music; performers could search for music remembered by melody. Music databases have the potential to greatly aid the process of matching by melody, but are currently only used by musicologists and music librarians.

Currently the commonest digital representation of music is as recorded performance information, stored as a series of amplitude measurements taken at a fixed rate such as 44.1 kilohertz. However, such information is extremely space-intensive, and it is difficult to transform it to an abstract, note-based form that might allow melody matching.

Another common representation is the MIDI format. Each MIDI file contains one or more tracks of note events (and other information); each track can consist of note events for multiple channels. Each channel is normally assigned to an instrument. Typically, two methods of organising MIDI files are used. In the first method, each instrument is allocated a separate track containing events for one channel only. In the second, there may be only one track containing note information within the file. In this case, note events for many channels are within the single track. Channels can contain multiple simultaneous notes, representing a chord. A MIDI file is similar to a score, in that it describes which notes should be played and by which instruments, but is intended for mechanical rather than human perusal. We use MIDI files throughout this paper, because they are readily available and because they contain much of the abstract information necessary for music similarity computation.

However, music similarity is a human judgement. Research into perception of music has shown that two pieces of music can seem alike even after substantial transformation, such as transposition into a different key, interpolation or omission of some notes, changing the pitch intervals between notes but preserving the contour, replacement of notes by appropriate chords, and addition or deletion of parts [6].

Any technique for matching pieces of music needs to consider that melody may be preserved by such transformations, and will inevitably be error-prone. For this reason a music retrieval system should be evaluated by its ability to highly rank the pieces of music that are similar to a query, over a collection of test queries where there has been some human evaluation of the similarity of each query to the stored pieces of music. That is, music retrieval systems need to be evaluated with the same methodology as that used for text retrieval systems [20].

3 Matching pieces of music

We propose a framework for matching pieces of music based on three stages: melody extraction, melody standardisation, and similarity measurement. We now describe each of these in detail, discussing why each stage is essential to the process of music matching.

Melody extraction

As discussed above, MIDI files usually consist of multiple tracks and channels, each representing a separate instrument. The perceived melody, however, is rarely played by a single channel: it can move from instrument to instrument. Some channels, on the other hand, consist of percussion, which has no pitch. When many parts are playing simultaneously, only some of the notes are perceived as part of the melody; for the purpose of matching music by similarity, it is necessary to automatically select these notes and discard the remainder.

There has been only a little previous work on melody extraction. Besides our previous experiments, described below, to our knowledge there has been no comparison of approaches to melody extraction. There have been several papers on splitting polyphonic music into parts using rules such as note proximity, but these have generally been designed for music with fairly uniform style [3, 17]. Another approach has been to use use the lowest of simultaneous notes, keeping all parts [1], or to simply discard percussion [10].

In previous work we developed a series of melody extraction algorithms, and evaluated them by asking listeners to rate them when applied to a series of pieces of music [25]. These algorithms were based on music perception research, which suggests that, where there

are multiple parts, the notes of highest pitch are often perceived as the melody; but that a sequence of notes that form an interesting pattern are more likely to be the melody than a repetition of a simple pattern, in which case low notes can be preferred over high notes [9]. There is only a little published work on other factors that may affect melody perception. Volume is of less importance in the grouping of notes into parts than is their relative pitch [4], but may determine what captures the listener's attention. In the case of MIDI files, many of the parts will have equal volume. Also there is no clear relationship between the specified volume and the perceived loudness for different instruments. Handel describes devices that allow soloists to be heard over other instruments [13].

We proposed several melody extraction techniques:

1. Combine all channels and keep the highest note from all simultaneous note events. In this paper, we refer to this method as *all-mono*.
2. Keep the highest notes from each channel, then select the channel with the highest first-order predictive entropy, that is, with—by a simple measure—the most complex sequence of notes. This is referred to as *entropy-channel*.
3. Use heuristics to split each channel into parts, then choose the part with the highest entropy. This is referred to as *entropy-part*.
4. Keep only the channel with highest average pitch, then keep only the highest notes as before. This is referred to as *top-channel*.

The first melody extraction algorithm (all-mono) is demonstrated in Figure 1. The example shows the situations in which extra and incorrect notes would be included. If each staff is a representation of the notes of a single channel, the upper staff would be extracted when using the top-channel method, the highest notes of the lower staff would be the choice of entropy-channel, and entropy-part would split the lower staff into several parts and choose the one with highest entropy.

The most successful of these algorithms was the simplest: the melody was best represented by choosing the highest note starting at any instance. The resulting sequence includes many more notes than those generated by the other methods, but usually succeeds in containing a significant proportion of melody notes. The larger number of notes makes the melodies more recognisable for human assessors, but potentially can make matching more difficult when using string matching techniques.

In this paper we again compare extraction techniques, as part of a music retrieval process.

Melody standardisation

Music perception research suggests that matching melodies on their exact sequences of notes is not appropriate: pieces of music can seem similar even if the notes are transformed in various ways. Contour—whether each note is of the same, higher, or lower pitch than its predecessor—can be more important than exact pitch [5], although exact pitch intervals between notes usually do help listeners to distinguish between melodies [8]. Transposition of the melody, within key or to a closely related key, has little effect on similarity perception. Melody standardisation is necessary to allow matching of pieces that have these kinds of variations.

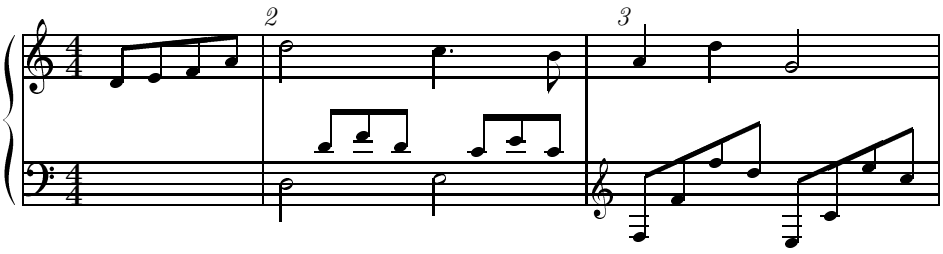
With *pitch contour*, melody standardisation consists of replacing each note by *up*, *down*, or *same*, representing the change in pitch from the previous note. For example, the familiar first phrase of Beethoven's Fifth Symphony would be represented by *same-same-down-up-same-same-down*. There has been much discussion as to whether pitch contour strings are adequate for locating melodies in databases [1, 7, 10, 18, 25], Previous work suggests that contour is probably not discriminating enough. Blackburn et al. [1] observed that contour would only be sufficient if strings were more than twelve characters long, with experiments on MIDI file database containing 8000 files. They discovered that while 60,000 of the possible 531,441 contours identify exactly two files, there are some contours that occur in over 1000 files. This agrees with McNab's results that show an average of about eleven notes to retrieve a unique melody using exact contours in a database of 9600 folk songs [18].

Note that contour strings, despite their simplicity, do not eliminate the possibility of error. If the melody extraction technique includes incorrect notes, these may be reflected in the contour.

An alternative to contour strings is to represent the melody as a sequence of changes in pitch; in this relative pitch or *interval* method, each note is represented as a change in pitch from the prior note. In the standard Western musical scale of twelve semitones to the octave, such changes in pitch are readily represented as small integers. For accurately transcribed melodies the intervals should be relatively small [5]. A variant on this technique is *modulo interval*, in which changes of more than an octave are reduced by twelve; for example, a transition of 17 semitones (from say A to D in the octave above) is reduced to 5. The range of possible pitch changes is 23 values, -11 to $+11$. This approach was used by Downie [7].

Another alternative represents notes relative to the key note or tonic. This is difficult with MIDI files, as key information is not present, adding a source of unreliability. Further, many pieces of music change

a)



b)

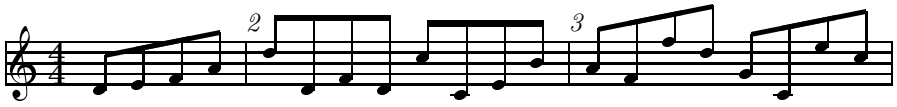


Figure 1: *The most successful of four melody extraction algorithms (all-mono) is shown. The original piece of music (a) contains the melody on the upper staff and accompaniment on the lower staff and the extracted melody (b) contains the highest note starting at any time. In the third bar, the accompaniment has changed clef and is occasionally above the melody in pitch. As a result, the melody notes in bar three have been replaced with higher accompanying notes in the extracted melody. Bars two and three show extra notes added to the extracted melody.*

key several times, adding extra problems for comparisons. Some research has used diatonic information, which is a special case of the representation of notes relative to the key note [19, 21]. In this case, notes may be represented by the note number of the scale, where the scale is the major or minor (or other) scale associated with the key of the composition.

In addition to pitch, rhythm information can be important for identifying a melody. Melodies are most easily identifiable by listeners when both pitch and rhythm are used, followed by pitch only and then rhythm only. Use of intervals and rhythm has been shown to improve melody retrieval [16] and increases the uniqueness of melody strings [18]. We are currently exploring use of rhythm. Another aspect of rhythm is the presence or absence of rests, or periods in music at which no note is sounded. We test the effect of rests in the experiments described below.

The stress or emphasis given to notes can also affect how the melody is perceived. For example, shifting a melody by one beat and adding an accompaniment that enforces the new beats completely changes its feel. Some MIDI files have time signature (or bar and beat) information explicitly provided but many do not. Volume can be used to determine note stress, but many MIDI files have all notes of equal volume making this unhelpful. Where rhythm or volume variation are available, they can be used to refine a melody matching scheme to place more emphasis on stressed note matching and less on unstressed notes. There are other situations that lead a listener to perceive a note as stressed, such as a large pitch interval between notes or a change in contour direction [15, 22, 23, 24] that may also be used to fine-tune melody matching.

McNab [18] compared contour and exact intervals, optionally combining both with rhythm contour. The combination of exact intervals and rhythm contour gave the best discrimination, tested by the number of notes required to uniquely identify a single-part piece of music. Inexact matching was also tested, but was less discriminating than exact matching. However, it is not clear what the implications are for a practical retrieval system—as we discuss below, error is inherent in music retrieval, so exact matching is unlikely to be an option.

Similarity measures

For music matching, for each piece in a music database it is necessary to apply melody extraction and standardisation, yielding a string against which queries can be compared. Queries must be transformed into strings of the same type. We assume that queries are entered as a series of notes, in which case they must be standardised; if the query is in some richer format, such as a multi-part representation, it will also be necessary to extract the melody.

Since melody extraction is inaccurate, and since—even with standardisation—the same melody can be represented by different sequences of notes, two strings

representing the same melody will not necessarily be identical. Matching must therefore be based on some measure of the similarity of query and piece. Given a standardised query string and a collection of standardised piece strings, matching involves computing a numerical score for each piece with respect to the query. The pieces can then be sorted by their score, and the highest-ranked pieces returned to the user as potential matches.

The edit distance family of string matching techniques is suitable for this task [11, 12], and have been widely applied in related applications including genomics and phonetic name matching [27, 28]. Three kinds of edit distance are applicable. The first is *longest common substring*: pieces are ranked according to the length of the longest contiguous sequence that is identical to a sequence in the query. This approach has the disadvantage, however, that a piece that matches well overall but has intermittent differences will not score highly. A more flexible method, successfully used in string matching and genomic retrieval [27], is to use *n-grams*: count the number of matching substrings of some fixed length n. The n-gram count should be normalised by string length, because long pieces are statistically more likely to have an n-gram match.

Two n-gram measures were used. In the first version, for every distinct n-gram in the query we counted the number of occurrences of the n-gram in the stored piece, in effect counting the number of n-grams in common between query and piece. For example, the query SSDUSSD represents the contour version of the opening phrase of Beethoven's Fifth symphony. The 4-grams it contains are shown in the upper part of the first column of Table 1. The third column shows the frequencies of these 4-grams in the contour for part of "Old MacDonald Had a Farm": SSDUSDUSDSDDUSSDUSDUSDSD. The score is the sum of these frequencies, in this case 6.

The other version of n-gram counting is based on the Ukkonen measure [26]:

$$\sum_{v \in \Sigma^n} |G(x)[v] - G(y)[v]|$$

where Σ^n is the set of possible n-grams, x and y represent the strings being compared and $G(x)[v]$ is the number of occurrences of the n-gram v in string x. In the above example, the Ukkonen measure results in a score of 15, which is the sum of the differences between query and piece shown in the last column of Table 1.

The second kind of edit distance is the *longest common subsequence*: in this case the query is matched against pieces with no penalty for gaps of any size between the matching symbols. Longest common subsequence has potential for this task because melody extraction often yields additional non-melody notes. For example, consider the sequence of intervals below for the Beethoven's Fifth Symphony fragment used earlier, represented as the number of semitones, with positive numbers meaning an increase in pitch.

0 0 −4 2 0 0 −3

Suppose this is to be matched against the start of "Au Clair de la Lune", which is represented as:

0 0 2 2 −2

By visual inspection it can be seen that they both contain the subsequence:

0 0 2

A dynamic programming approach is used to calculate the length of this maximal subsequence [14].

The third kind of edit distance is *local alignment*: dynamic programming is used to determine the best match of the two strings on a local basis. For example, using the two melody fragments above, we would fill a two-dimensional array of values according to the following formula, where a represents the array, q and p represent the query melody string and piece to match against respectively, and array index i ranges from 0 to query length and index j ranges from 0 to piece length:

4-gram	No. times in query	No. times in melody	Diff.
SSDU	1	2	1
SDUS	1	2	1
DUSS	1	1	0
USSD	1	1	0
DUSD	0	4	4
USDU	0	2	2
USDS	0	2	2
SDSD	0	2	2
DSDD	0	1	1
SDDU	0	1	1
DDUS	0	1	1

Table 1: *Frequency of each distinct contour 4-gram in the Beethoven's 5th query* SSDUSSD *and matched against the first phrase of "Old MacDonald Had a Farm"* SSDUSDUSDSDDUSSDUSDUSDSD. *The 4-grams above the horizontal dividing line are those occurring in the query. Their frequencies in the melody being matched are summed to determine its similarity score (which is 6). The Ukkonen measure uses the sum of the difference in n-gram frequencies in the two strings (which is 15).*

		0	0	-4	2	0	0	-3
	0	0	0	0	0	0	0	0
0	0	1	1	0	0	1	1	0
0	0	1	2	0	0	1	2	0
2	0	0	0	1	1	0	0	0
2	0	0	0	0	2	0	0	0
-2	0	0	0	0	0	1	0	0

Figure 2: *Local alignment of fragments (0 0 −4 2 0 0 −3) and (0 0 2 2 −2). Once the array has been filled, the arrows, representing how the value pointed to was calculated, are followed to determine the parts that have been aligned.*

$$a[i,j] = max \begin{cases} a[i-1,j]+d & (i \geq 1) \\ a[i,j-1]+d & (j \geq 1) \\ a[i-1,j-1]+e & (q(i)=p(j) \\ & \text{and } i,j \geq 1) \\ a[i-1,j-1]+m & (q(i) \neq p(j)) \\ 0 & \end{cases}$$

where d is the cost of an insert or delete, e is the value of an exact match, and m is the cost of a mismatch. We used $d = -2$, $e = 1$ and $m = -1$.

The resulting array for our example melody fragments is shown in Figure 2. This has a maximum value of 2, which occurs in three locations. Tracing the path that led to the local maxima results in the following three matches of the strings: the first 0 0 sequence of (0 0 −4 2 0 0 −3) is matched to the 0 0 of (0 0 2 2 −2); the second occurrence of 0 0 in (0 0 −4 2 0 0 −3) is matched to the 0 0 in (0 0 2 2 −2); the sequence (0 0 −4 2) is matched with (0 0 2 2) with one mismatch.

Musical information is multi-dimensional, however, so the techniques can be extended to capture more information for comparison. For example, Mongeau and Sankoff extended their edit distance computations to use two dimensions—pitch and rhythm [19]. The pitches are relative to the key of the melody. A set of weights was devised to distinguish between consonant and dissonant intervals. Other techniques used were fragmentation and consolidation. This involves matching a single long note to repeated short notes with only a small penalty score. The method was tested on two compositions by Mozart and successfully grouped similar variations.

Longest common subsequence and local alignment are against a series of note transitions, so that a wrong note introduces two errors. This would not happen if absolute pitch was used, where, however, a score of 0 would be calculated for two identical melodies in different keys.

4 Experimental design

Our aim in this paper is twofold: first, to confirm that the three-stage framework can be used to find matching melodies in a large corpus of musical works; and second, to identify appropriate techniques for each stage of the retrieval process.

For the measurement of a retrieval system we need [20]: a collection of melodies, a collection of queries, and, for each query, human relevance judgements as to which of the melodies are similar to which of the queries. When a system determines a ranking of melodies for a given query, the relevance judgements can be used to assign a score to the retrieval system; a system should get a high score if it is good at highly ranking similar melodies. A standard measurement technique for this task is to use recall (the proportion of the relevant melodies that have been retrieved) and precision (the proportion of the retrieved melodies that are relevant); recall and precision can be averaged at fixed recall levels to compute an overall eleven-point recall-precision average [20]. Alternatively, a system can be measured by the number of relevant melodies amongst the first k retrieved, for say $k = 20$; this is the precision-at-twenty measure.

We use both measures in the experiments described below. To our knowledge this is the first use of this standard system measurement technique for music retrieval.

Test collection and query set

The collection of musical works used for our experiment were obtained from a large collection of MIDI files kept at an internet archive site.[1] This collection consists of approximately 15,000 files, but contains many in non-standard formats. After excluding these, the number in our experimental collection was 10,466. Duplicates were retained.

The experiment used query melodies that were automatically extracted from pieces of music, using the algorithms described above. The 28 pieces of music chosen were randomly selected from a sub-collection of about one hundred pieces that we identified as having more than one distinct arrangement in the collection. An arrangement was considered to be distinct from other arrangements if one of the following conditions held: it was in a different key; there was a different number of parts in the arrangement; or there were differences in rhythm, dynamics or structure. All arrangements were designated as the relevant pieces for each of the queries. Some arrangements may not have

[1] ftp://trantor.student.utwente.nl/pub/midi/

been identified, where non-obvious names had been chosen for the files.

Typically there were two to six relevant pieces of music for each query, on the assumption that all other pieces are not similar—a reasonable assumption if the retrieval task is to find variant forms of the same piece of music. While the assumption may not be valid for other retrieval tasks, it should not discriminate between the retrieval methods.

This query set contains melodies from a range of music genres, including pop and rock music from every decade since the fifties, some jazz works, classical compositions, country music, a Christmas carol, and several TV or movie themes. After the queries were chosen, the various extraction and standardisation techniques were applied. To simulate queries of varying lengths, each string was truncated to 10, 30, and 100 notes, thus giving three versions of the collection of queries.

We believe that this query set provides an excellent test of a music retrieval system, not least because the data originated elsewhere. A common experimental methodology for a retrieval system is to ask users to specify queries, but it is difficult to eliminate the possibility that the experimenter has influenced the query development process. For example, for a query-by-contour system an experimenter may encourage less accurate queries than those that might be specified in other contexts.

Matching techniques

For our experiment, we used each of the four melody extraction techniques in turn for both the queries and the collection. In addition, we used an *all-channels* technique that generated a separate melody from each channel, so that each piece could be represented several times. This reduces the chance of missing melodies but increases the chance of false matches—that is, recall should improve at some cost to precision. Use of all-channels increases the size of the collection to about six times the original. In this case, the collection contained 69,032 separate parts.

The melody standardisation methods tested were melodic contour, modulo intervals, and exact intervals. In each case we tested inclusion and omission of rests. When rests were included, they were represented as a single symbol. Intervals following a rest were calculated using the note that occurred before the rest. Rests were inserted if there was a break of at least 5 time units between notes. This is a fairly short value whose actual duration varies between pieces, depending on the speed of the music.

The similarity measures were local alignment, longest common subsequence, and the two versions of n-gram counting, for $n = 4$. Varying n trades recall against precision—high n provides closer matching but is more likely to miss variations with omitted or additional notes. Our experience with use of n-grams in genomic and string matching suggested $n = 4$ as a good compromise, but we will test other values in further work. The scores calculated using local alignment, longest common subsequence and the first n-gram scoring method, were normalised by dividing by the log of the melody length plus one, to compensate for varying piece length.

The experiment itself was factorially exhaustive. We combined every melody extraction technique with every standardisation method and every edit distance, over the database as well as over each of the three query sets. We ran all possible experiments in which queries processed with one extraction technique were run against a version of the database processed with another extraction technique. This design is intended to allow complete analysis of which factors are important for music similarity measurement.

5 Results

Results are shown in Tables 2, 3, 4, and 5. Table 2 shows the effect of varying query length, melody extraction technique, standardisation technique, and similarity measure, where queries and the database are identically processed, measured by a recall-precision average. Perhaps the most important result in this table is that it demonstrates that the framework is effective: matching melodies can be found using the appropriate combination of extraction, standardisation, and similarity measure, with, in the best case, even for short queries one in three retrieved melodies being correct (and, in the top 20 as shown in Table 3, three of five melodies being correct). By the performance standards of document retrieval systems, these are excellent results.

Table 2 shows that contour is always the worst standardisation technique and is almost certainly not usable in practice. Modulo and exact intervals have similar performance to each other, with modulo better for long queries and exact interval better for short queries. Only for queries of 30 notes or more does contour have any success at finding matches—but a query of 30 notes is much longer than would be expected for practical retrieval. Note that in many cases 30 notes is more or less the whole melody; extending to 100 notes often just introduces repetition of the theme.

Another clear result is that local alignment is the best similarity measure, followed by n-gram counting. Both longest common subsequence and the Ukkonen measure performed poorly. Local alignment has allowed good matching with all melody extraction techniques, even for short queries. However, normalisation was an important factor with which we only conducted

Table 2: *Eleven-point recall-precision averages (as percentages) for matching without rests.*

similarity measure	extraction method	contour			modulo interval			exact interval		
		10	30	100	10	30	100	10	30	100
local alignment	all mono	0.68	20.37	35.72	26.83	44.58	49.66	31.17	44.01	45.60
	ent. chan.	1.14	20.25	29.84	21.71	36.24	38.71	25.70	37.09	35.83
	ent. part	2.94	18.83	23.28	12.33	23.51	25.99	12.77	23.53	26.17
	top chan.	1.00	21.05	29.82	21.85	37.02	39.55	21.17	36.40	39.69
longest common subseq	all mono	0.03	0.08	1.84	0.15	2.65	31.81	0.34	7.64	36.91
	ent. chan.	0.19	0.90	4.59	1.49	7.75	25.74	1.89	11.45	28.88
	ent. part	0.19	3.08	8.43	1.35	7.33	21.81	1.60	9.54	22.93
	top chan.	0.16	2.32	16.14	3.27	16.26	26.73	2.29	17.96	28.37
ngram count commons	all mono	0.05	0.04	0.07	15.75	20.18	21.22	23.95	25.48	28.31
	ent. chan.	0.12	0.08	0.08	16.08	14.65	15.74	18.11	16.92	16.73
	ent. part	0.21	0.36	0.05	7.25	9.24	9.80	7.43	10.28	10.91
	top chan.	1.57	1.61	0.20	18.49	18.80	17.79	18.82	19.44	19.75
ngram Ukkonen measure	all mono	0.04	0.04	1.10	0.04	0.05	1.67	15.77	18.78	15.13
	ent. chan.	0.15	0.79	4.68	0.14	0.69	5.62	0.15	0.87	5.65
	ent. part	0.16	3.02	7.69	0.17	3.16	9.51	0.18	3.18	9.52
	top chan.	0.10	2.50	15.99	0.05	2.59	15.80	0.06	2.53	16.00

limited experiments; we already had many variables to consider. We plan to refine normalisation in further work. The n-gram count method may have been penalised because some n-grams, such as when the same note is repeated several times, are very common, thus favouring long pieces of music when queries contained these common n-grams. Shorter queries are less likely to contain the common n-grams and perform about as well as long queries. It may be appropriate to use n-gram frequencies within the collection to determine a contributory weight for each n-gram, just as words are differentially weighted in document retrieval. The Ukkonen measure was probably poor because it is a global measure rather than a local one: longer compositions are penalised if they contain many n-grams that don't occur in the query, even if the query matches a portion of the longer compositions well.

Of the melody extraction techniques, the entropy-part method (the most complex technique tried) was the weakest; the others have all worked well, with all-mono and entropy-channel both giving good results. All-mono was clearly the best technique in conjunction with n-grams and was usually the best in conjunction with local alignment. In results not reported here, we found that using the same melody extraction method as that used for the database melodies is always the better than matching queries processed one way against a database processed in another.

Table 3 concerns the same experiments and variables as in Table 2, but using precision at 20—the number of correct answers in the first 20 pieces retrieved—instead of recall-precision. (In this and subsequent tables the Ukkonen and longest common subsequence results are omitted because performance is very poor; however, these results do exhibit the same trends as the reported figures.) This table shows that the results are independent of the performance measure. Note that, if all relevant pieces are retrieved in say the top 12, precision is calculated at that point rather than at 20.

Table 4 shows the results of experiments where rests are included in the melody standardisation; in all other respects the experiments are identical to those reported in Table 2. Comparing these two tables, it can be seen that rests are only helpful with short queries, and the best performance with rests is not as good as the best performance without. Rests have helped contour, but not enough to make it useful.

Table 5 shows the results of experiments where each piece in the data set was represented several times, by the highest-note sequences from each channel, while the queries were processed as before. The performance of the channel-based extraction methods is much improved, particularly for longer queries, while the all-mono method has not worked as well. Overall results are not quite as good as the best reported in Table 2. The fall in performance is probably because, while use of all-channels eliminates the need to guess which channel contains the melody, the addition of "noise" channels increases the likelihood of false matching.

We would expect that manual queries (in contrast to our queries derived from variant transcriptions) would most resemble those produced by entropy-channel and top-channel methods, which contain fewer extra notes from other parts of the composition.

Table 3: *Precision at 20 values for matching without rests.*

similarity measure	extraction method	contour			modulo interval			exact interval		
		10	30	100	10	30	100	10	30	100
local alignment	all mono	0.71	8.21	12.32	11.96	15.71	15.89	12.32	15.18	15.89
	ent. chan.	0.89	9.29	11.25	11.25	12.68	13.39	10.89	12.86	13.39
	ent. part	2.32	8.75	10.71	6.43	9.64	10.89	6.79	9.82	10.89
	top chan.	1.07	9.29	12.32	10.36	13.75	14.11	10.00	13.75	14.29
ngram count commons	all mono	0.00	0.00	0.00	8.21	9.29	10.54	11.07	10.89	10.71
	ent. chan.	0.18	0.00	0.00	7.32	5.89	6.61	7.32	7.50	7.14
	ent. part	0.18	0.18	0.18	5.54	5.89	6.25	5.71	6.43	6.96
	top chan.	0.36	0.36	0.00	8.39	9.11	7.86	8.04	9.11	9.11

Table 4: *Eleven-point recall-precision averages (as percentages) for matching with rests.*

similarity measure	extraction method	contour			modulo interval			exact interval		
		10	30	100	10	30	100	10	30	100
local alignment	all mono	1.88	24.04	26.04	20.78	32.40	32.35	13.03	16.27	15.60
	ent. chan.	6.47	19.16	23.15	18.00	27.04	26.24	19.87	27.02	26.24
	ent. part	4.48	16.35	21.32	14.30	20.78	20.93	13.70	21.09	20.68
	top chan.	4.90	23.81	27.00	24.11	31.14	29.98	24.35	30.92	30.25
ngram count commons	all mono	0.15	0.11	0.09	7.96	9.30	12.15	14.37	17.56	20.02
	ent. chan.	0.18	0.04	0.03	8.01	4.81	2.26	8.55	9.70	9.40
	ent. part	0.15	0.07	0.04	4.92	3.62	2.18	4.15	4.50	4.24
	top chan.	2.53	2.45	0.23	17.28	13.63	8.99	16.61	17.82	14.62

It follows that for manual queries the use of all channels from each composition may be valuable. Manual queries to a music retrieval system are likely to consist of a phrase, and thus will usually be less than 10 notes, and therefore good performance for short queries is vital.

6 Conclusions

We have proposed a methodology for matching pieces of music according to whether they are likely to be perceived as similar by a listener. We argued that a minimal matching process must involve at least melody extraction, to reduce each piece to a linear sequence of notes; standardisation, to represent each piece in a form that is independent of variables such as key that do not affect similarity perception; and a similarity measure, for scoring the similarity of two pieces of music. Using a large number of multiphonic pieces in the MIDI format extracted from a public-domain collection of music, we have shown that this methodology effectively finds pieces that are similar to each other.

The success of the matching requires effective extraction, standardisation, and similarity techniques. We selected a variety of such techniques for experimental evaluation, many of which had been previously proposed for this task. Combining simple melody extraction (taking the highest note starting at any time), relative pitch intervals, and local alignment gave excellent effectiveness: for even short queries, the top 20 answers contained 12 correct matches on average.

Some options, however, were highly unsuccessful, finding virtually no answers at all. For example, melody contour standardisation—used in some "query by humming" systems—does not work with queries of reasonable length, and longest common subsequence is a poor similarity function. Rests appeared to be unhelpful in matching.

Some questions remain unanswered and we will investigate them in future work. For example, we did not fully explore the use of normalisation to give long and short pieces equal probability of matching a query, nor did we make use of rhythm information. Both have the potential to further improve effectiveness. We are also working on efficiency issues, in particular the use of indexing to allow rapid matching. However, our results clearly answer the key question for music databases: use of the three-stage framework allows effective retrieval of music by theme.

Table 5: *Eleven-point recall-precision averages (as percentages) for matching against all channels without rests.*

similarity measure	extraction method	contour			modulo interval			exact interval		
		10	30	100	10	30	100	10	30	100
local alignment	all mono	0.53	7.14	9.09	10.57	38.04	38.38	13.72	38.35	36.23
	ent. chan.	0.10	15.53	35.79	9.15	43.65	47.65	11.61	45.33	48.77
	ent. part	0.12	5.92	17.80	3.86	20.02	25.99	4.41	19.91	24.13
	top chan.	0.36	20.58	35.22	18.64	42.58	46.69	19.56	42.53	47.05
ngram count commons	all mono	0.12	0.04	0.33	8.79	13.53	14.07	14.04	16.52	15.56
	ent. chan.	0.08	0.08	0.06	8.60	10.58	13.87	13.78	13.75	14.35
	ent. part	0.08	0.08	0.02	4.64	5.75	3.08	5.05	5.85	3.33
	top chan.	0.95	0.94	0.09	11.87	11.78	9.75	14.19	12.78	10.88

Acknowledgements

We are grateful to John Harnett for his help with the experiments and to Hugh Williams for his comments. This work was supported by the Australia Research Council.

References

[1] S. Blackburn and D. DeRoure. A tool for content-based navigation of music. In *Proceedings: ACM Multimedia 98, September 11–15, 1998 Bristol, England*. ACM, 1998.

[2] J. Borchers and M. Muhlhauser. Design patterns for interactive musical systems. *IEEE Multimedia*, 5(3):36–46, 1998.

[3] H. Charnasse and B. Stepien. Automatic transcription of german lute tablatures: an artificial intelligence application. In *Computer Representations and Models in Music*, pages 143–170. Academic Press, 1992.

[4] D. Deutsch. Grouping mechanisms in music. In *The Psychology of Music*, chapter 4, pages 99–134. Academic Press, Inc., 1982.

[5] W.J. Dowling. Scale and contour: Two components of a theory of memory for melodies. *Psychological Review*, 85(4):341–354, 1978.

[6] W.J. Dowling. Melodic information processing and its development. In *The Psychology of Music*, chapter 13, pages 413–429. Academic Press, Inc., 1982.

[7] J.S. Downie. The musifind music information retrieval project, phase iii: evaluation of indexing options. In *Canadian Association for Information Science proceedings of the 23rd Annual Conference, Connectedness: Information, Systems, People, Organisations*, pages 135–46. CAIS, 1995.

[8] J. Edworthy. Interval and contour in melody processing. *Music Perception*, 2(3):375–388, 1985.

[9] R. Frances. *La Perception de la Musique*. L. Erlbaum, Hillsdale, New Jersey, 1958. Translated by W.J. Dowling (1988).

[10] A. Ghias, J. Logan, D. Chamberlin, and B. Smith. Query by humming — musical information retrieval in an audio database. In *ACM Multimedia 95 — Electronic Proceedings*, 1995.

[11] D. Gusfield. *Algorithms on Strings, Trees, and Sequences*. Cambridge, USA, 1997.

[12] P.A.V. Hall and G.R. Dowling. Approximate string matching. *ACM Computing Surveys*, 12(4):381–402, 1980.

[13] S. Handel. *Listening: An introduction to the perception of auditory events*. MIT Press, 1989.

[14] D.S. Hirschberg. Serial computations of levenshtein distances. In A. Apostolico and Z. Galil, editors, *Pattern Matching Algorithms*, chapter 4, pages 123–141. Oxford University Press, 1997.

[15] D. Huron and M. Royal. What is melodic accent? converging evidence from musical practice. *Music Perception*, 13(4):489–516, 1996.

[16] T. Kageyama, K. Mochizuki, and Y. Takashima. Melody retrieval with humming. In *ICMC Proceedings 1993*, 1993.

[17] A. Marsden. Modelling the perception of musical voices: a case study in rule-based systems. In *Computer Representations and Models in Music*, pages 239–263. Academic Press, 1992.

[18] 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 *Digital Libraries Conference*, 1996.

[19] M. Mongeau and D. Sankoff. Comparison of musical sequences. *Computers and the Humanities*, 24:161–175, 1990.

[20] G. Salton and M.J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, 1983.

[21] H. Schaffrath. The retrieval of monophonic melodies and their variants: Concepts and strategies for computer-aided analysis. In A. Marsden and A. Pople, editors, *Computer Representations and Models in Music*, pages 95–110. Academic Press, 1992.

[22] H.G. Tekman. Interactions of perceived intensity, duration and pitch in pure tone sequences. *Music Perception*, 14(3):281–294, 1997.

[23] H.G. Tekman. Effects of melodic accents on perception of intensity. *Music Perception*, 15(4):391–401, 1998.

[24] J.M. Thomassen. Melodic accent: experiments and a tentative model. *Journal of the acoustical society of America*, 71(6):1596–1605, June 1982.

[25] A.L. Uitdenbogerd and J. Zobel. Manipulation of music for melody matching. In *Proceedings: ACM Multimedia 98, September 11–15, 1998 Bristol, England*. ACM, ACM Press, 1998.

[26] E. Ukkonen. Approximate string-matching with q-grams and maximal matches. *Theoretical Computer Science*, 92:191–211, 1992.

[27] H. Williams and J. Zobel. Indexing nucleotide databases for fast query evaluation. In *Proceedings of Advances in Database Technology (EDBT'96)*, pages 275–288, Avignon, France, March 1996.

[28] J. Zobel and P. Dart. Finding approximate matches in large lexicons. *Software—Practice and Experience*, 25(3):331–345, March 1995.