Manipulation of Music For Melody Matching

Alexandra. L. Uitdenbogerd *Justin Zobel*
Department of Computer Science, RMIT, Melbourne, Victoria, Australia
sandrau@rmit.edu.au *jz@cs.rmit.edu.au*

Abstract

Many types of user would find it valuable to search collections of music via queries representing music fragments, but such searching requires a reliable technique for identifying whether a provided fragment occurs within a piece of music. The problem of matching fragments to music is made difficult by the psychology of music perception, because literal matching may have little relation to perceived melodic similarity, and by the interactions between the multiple parts of typical pieces of music. In this paper we analyse the properties of music, music perception, and music database users, and use the analysis to propose alternative techniques for extracting monophonic melodies from polyphonic music; we believe that such melodies can subsequently be used for matching of queries to data. We report on experiments with music listeners, which rank our proposed techniques for extracting melodies.

1 Introduction

Content-based retrieval is being explored for many different forms of media. In the case of audio data, the content may consist of sounds, speech, or music, the topic of this paper. Content-based retrieval of music has many applications, from casual users wanting to identify a nagging fragment of music to artists verifying that a new composition is indeed original. A likely form of query for such retrieval is a short melody fragment, input via a keyboard or possibly even hummed or sung [12, 17]. The task of a retrieval system is to identify the pieces of music that contain music fragments that sound similar to the query. To achieve this task requires a technique for matching melody fragments. By analogy with the conceptually similar problem of text retrieval [19], the matching technique need not be perfect—users are likely to be tolerant of sloppiness in matching—but it needs to be reasonably reliable, so that users can find matches fairly rapidly. We examine possible ways of matching pieces of music, including the sequences of notes, intervals between the notes, and melody contour. We argue that a combination of techniques is likely to be the most effective solution: evidence of similarity does not solely depend on the notes played. However, a successful indexing technique needs to account for both user needs and the psychology of music perception.

We use music perception results and the likely uses of music databases to propose several alternative methods for extracting melodies from polyphonic music, thus allowing searching: previous work on music searching has assumed that the music is monophonic, and indeed it is far from clear how searching of polyphonic music might proceed. We test our melody extraction techniques by asking listeners to identify which of the extracted melodies most resembles the original piece. These experiments, although limited in scope, identify a clear winner, which surprisingly is the most naive of the methods we considered.

Music can be thought of as a sequence of notes and chords. The pitch or frequency distance between notes is called an *interval*. The smallest interval in western music is called a *semitone*. An *octave* is the interval that occurs when one note's frequency is double that of the other. Notes that are an octave apart sound very similar and have the same note names. Usually, music is written in a *key*, which refers to a subset of possible notes, conforming to a standard pattern of intervals. A melody *transposed* from one key to another will sound like the same melody, but be perceived as higher or lower than the original. If a piece of music consists largely of notes from a specific key, it is called *diatonic*, whereas if it ignores keys it is called *atonal*.

In this paper, we use the term *monophonic* to refer to music consisting of a sequence of notes of which none are simultaneous. We use the term *polyphonic* to mean music in which there exist simultaneous notes. Where the term *melody* is used, it is assumed to be monophonic.

2 Music Database Users

Being able to retrieve data from a music database using a melody fragment as a query would be desirable for several applications. It is useful for copyright searches, locating a particular musical work for which one doesn't have the title, performing musicological studies of music collections, and as a composition tool. In the case of

ACM Multimedia'98, Bristol, UK
© 1998 ACM 1-58113-036-8/98/0008 $5.00

copyright searches, all works that are sufficiently similar to the query should be retrieved, whereas for the music library patron who is locating a particular work of which they can only remember a part, only one answer is being searched for. To this type of user, a query result will only be relevant if it is the same as the musical work that they remember. Other results may have some similarity that leads them to be considered relevant in a general sense, but will not be of use to the user; this difference between "topicality" and "utility" is discussed by Blair [3].

Composers may also be interested in types of queries that aid the composition process. For example, if a melody is being harmonised or arranged, a composer might like to see how other composers handled a particularly difficult sequence of notes.

Different types of users have different levels of skills when it comes to describing their query. Composers, musicologists and other musicians could be expected to be able to prepare a query using a keyboard or notation with a good degree of accuracy. The lay person would have some difficulty in preparing a music query. Several systems have been described that allow the user to perform queries by humming or singing [12, 21, 17]. Problems with this method is that people do not sing accurately, especially if they are inexperienced or unaccompanied; even skilled musicians have difficulty in maintaining the correct pitch for the duration of a song. However, melody contour, that is, the pitch direction of the notes, is usually accurate.

3 Music Perception

Several aspects of music perception need to be considered when developing matching techniques for music databases. First is the type of query that users will present. In the case of someone trying to locate a half-remembered fragment of music, it is useful to understand how people remember music and in particular, how they remember melodies. Second, since most music that we hear contains both melody and accompaniment, it is necessary to determine what would be perceived as melody in an accompanied musical work. Third, since many music queries involve finding similar but not exact matches to melodies, we need to decide what similarity means in terms of music perception. We now discuss these areas and the implications for music databases.

Music Memory

There has been much research on how people build mental structures while listening to music and on how music is remembered. Dowling [7] discovered that melody contour is easier to remember than exact melodies. Contour refers to the shape of the melody, indicating whether the next note goes up, down, or stays at the same pitch. He postulates that contour and scale are stored separately in memory and that melodies are mapped onto an overlearned scale structure, so that if presented a melody with an unknown scale, we map it to the scale structure with which we are familiar. Edworthy [10], however, discovered that the length of a novel melody determines whether it is easier to detect changes in contour or changes in intervals. There are also differences in terms of short-term and long-term memory for melodies: it is easy for people to reproduce a well-known melody with exact intervals than to do so for new melodies. In terms of music database queries, a user's query melody will probably be in long-term memory, so exact intervals will be important. Memory tasks are generally performed better by experienced musicians than by those with less experience.

Figure and Ground

In order to extract melodies reliably from polyphonic music files it is necessary to determine what a person listening to the music would perceive as the melody. Several papers have explored the way that groups of notes are perceived. A melody is heard as the *figure* if it is higher than the accompanying parts. However, if the upper notes are constant and the lower notes form a more interesting pattern, then the lower notes will be heard as the figure [11]. Deutsch [6] discusses the main principles of perception of groups and showed the results for the perception of music. *Proximity* of notes is more important than the *good continuation* principle, as illustrated by the "scale illusion" experiment: if one part is descending in a scale and another part ascends so that they overlap, listeners perceive the upper notes as one part and the lower notes as a second part (if it is perceived at all). Similar amplitude or loudness of notes was found to be fairly unimportant in the perception of musical parts compared to proximity, but is also used for grouping of notes.

If a melody consists of rapid notes where the alternating notes are of a different frequency range, then two musical parts are perceived. This does not occur to the same extent when the melody is slowed down. There is a considerable overlap in terms of how the music will be perceived, so that a melody could be perceived as being two musical parts or just one over a range of speeds. This suggests that a melody extractor should consider the speed of notes as well as the relative frequencies.

Composers sometimes present us with other difficulties, for example, Tchaikovsky's sixth symphony has the theme and accompaniment distributed between two violin parts, but the theme is perceived to come from one set of instruments and the accompaniment from another. This implies that the result of the melody extraction process may not be as originally organised by a composer, even if it correctly simulates how it will be perceived. Conversely, if stored music is organised according to the sheet music for a composition, then the musical work may not be retrieved given the user's perception of the melody.

Melody Similarity Perception

It is highly likely that many melodies will match a user's query, so a similarity ranking based on perception results may be useful. Our analysis of music perception research [2, 7, 15, 22] suggests the following ordering of factors in music similarity, from good evidence of similarity to poor.

1. Exact transposition in the same key.
2. Exact transposition in a closely related key.
3. Exact transposition in a distant key.
4. Exact transposition except for a chromatically altered note.
5. Exact transposition except for a diatonically altered note.
6. Same contour and tonality.
7. Same contour but atonal.
8. Same note values but different contour.
9. Different contour, different note values.

For our application, the first three items can be condensed into one, since relative intervals will be used for melody comparisons. The music perception survey performed thus far has not revealed any comparisons of rhythm or timing in the perception of similarity. Our experience suggests that difference on a stressed note is more significant than difference on an unstressed note.

4 Extracting Melodies from MIDI Files

Music is represented in many ways. It can be stored as: a waveform representing the sound, an image of the sheet music, performance information, or music notation layout information. The MIDI (Musical Instrument Digital Interface) standard is a commonly-used format for musicians who exchange music sequences and for lay people who play music on their computers. It usually includes performance information only, that is, when a note is played, how loudly, and for how long. It can only be used for instrumental music and does not store notation information such as the division of music into bars. Standard MIDI files contain one or more tracks of MIDI events. MIDI files usually store each instrument's note events on a separate track. Sometimes, however, a track consists of several channels in which case each instrument is assigned a separate channel within a track.

The Internet has made large quantities of MIDI data available. One site archives MIDI files sent to a newsgroup and contains about 15 000 files. MIDI data from the Internet is used as a basis for the music data analysis in this paper.

The individual instruments occurring in a MIDI file can be playing chordal parts, consisting of several simultaneous notes; there may be more than one "countermelody" occurring simultaneously; and the perceived melody can move from instrument to instrument. That is, the melody may not be present in any individual part, thus it is far from clear which sequence (or sequences) of notes should be used in the comparison. Additionally, some "notes" are actually percussion and therefore have no pitch. Extraction of all sequences would lead both to combinatorial explosion and large numbers of sequences with little perceived connection to the original music; indeed, most such sequences would sound like no more than a random series of notes.

The purpose of a melody extraction technique is to identify sequences of notes that are likely to correspond to the perceived melody. Given the volumes of music available online, it is not practicable to do this by hand.

There does not appear to have been much research on the problem of extracting a melody from a piece of music. The approach of Ghias et al. [12] was to ignore percussion and apply other simple heuristics (not described in the paper) to collect melodies. Several papers have discussed splitting polyphonic-style music into its parts using rules, largely based on proximity and usually for a set of music with a fairly uniform style [4, 16].

To obtain melodies, we trialled several approaches that make use of music perception principles: the highest musical part is usually perceived to be the melody, and notes that are close together in pitch are usually considered to belong to the same musical part. We also used first-order predictive entropy as a measure of how interesting a part was, to see if this could be used to predict melody. This calculation is in effect based on the probabilities in a simple Markov model with one state for each note. In highly repetitive tracks, such as some forms of accompaniment, each note is reliably predicted by its predecessor, giving a low entropy, whereas in more varied tracks the entropy is high.

Four methods of extracting a melody from a MIDI file were developed. All the methods made a single pass through the note stream to select the melody notes, ignoring the note events from channel 10 as these are percussion events in standard MIDI files. In some circumstances, the melodies generated are identical. This occurs when the music consists of a melody with a simple chordal accompaniment below it, with the chords occurring at the same time as melody notes. The four algorithms are as follows; the effect of each algorithm is illustrated in Figure 1.

- In algorithm one, all musical information is combined into one stream of events. Whenever a note starts, the algorithm chooses the top note of all notes starting at that instance. This will result in many overlapping notes, as a note that is sustained in one track will cover the start of other notes. For the purpose of evaluation, note lengths were truncated until the result was monophonic. As can be seen in Figure 1b, extra notes are often included in the melody.
- Algorithm two makes use of the structure of MIDI

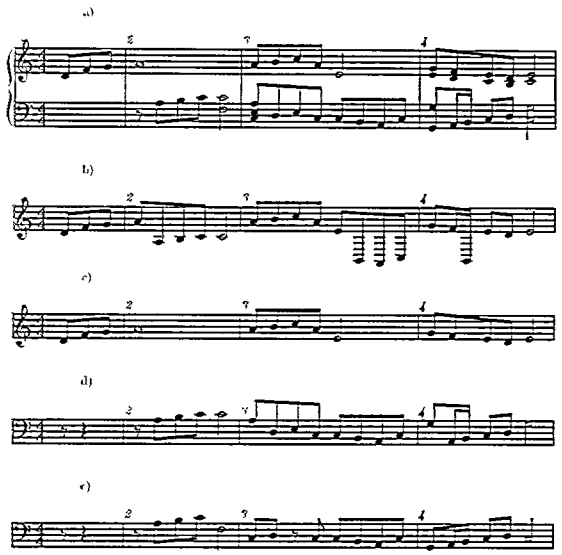

Figure 1: *A short piece of music (a) and the resulting monophonic "melodies" (b–e) extracted by each of the four algorithms.*

files (tracks and channels) by processing each channel separately; for each channel the information is processed as in algorithm one above. The channel with the highest average pitch is then chosen as the melody. The algorithm, illustrated in Figure 1c, works well for the example but in practice sometimes wrongly identifies a high accompanying part.

- In algorithm three the top notes for each channel are chosen as in algorithms one and two, then the channel with the highest first-order predictive entropy is selected. In the example the lower part had the greatest entropy and so was chosen, as in Figure 1d, instead of the melody.
- Algorithm four uses heuristics to split each channel into parts. It does so by allocating successive notes to parts that have the closest proximity in pitch to the current note in the stream of notes. A new part is only created if the current note occurs at the same time as the existing parts. The part with the highest entropy is then chosen as the melody. The splitting algorithm is a little simplistic as it only compares the current note to previous notes. As can be seen in Figure 1e, the part selected goes from middle C to F, instead of back to middle C, because in the note stream the F occurred before the second middle C. If the simultaneous notes C and F were reversed in the note stream a rather different part would be produced.

From our analysis of the needs of music databases users and from well-known results in music perception, we believe that each of these algorithms has potential as a method of extracting melodies from music. However, application of these methods to actual music shows that they can produce very different results. We applied the four algorithms to ten MIDI files, representing a range of styles and methods of organisation within a file. Each of our eight listeners was presented with the original MIDI file and the four automatically-extracted melodies for each of the ten files.[1]

The melodies generated for each piece of music were presented in random order. The listeners were able to play the pieces as often as they wished and sometimes only played a small portion of the melodies if that was sufficient to make a decision. The melodies were ranked from one to four according to how well they represented the melody of the original piece. The listeners were asked to consider the presence of extra notes and absence of melody notes in their evaluation, and were permitted to give melodies equal ranking.

Results

The sum of the ratings for each algorithm for each piece are shown in the table at Table 1. Low scores indicate a high rating. The range of possible scores is 8 to 32. As can be seen from the results, most algorithms performed well for some pieces and badly for others. The only algorithm to work consistently well was algorithm one—the most naive of the methods we considered. Interestingly, for pieces 5 and 8 the first three algorithms produced identical melodies but were ranked somewhat differently. For piece number 5 ("Scotland the Brave") it is the authors' opinion that algorithm four performed the best, as accompanying notes at the start and during the piece were removed. This difference was not detected by the evaluators, however.

Sometimes music is perceived differently if it is recognised. In this experiment, however, the algorithm rankings were similar regardless of whether or not the music was recognised. The music experience of the evaluators did not reveal any clear cut trends in algorithm preference. The least skilled group (three people) and the most skilled group (three people) ranked algorithm one as the best whereas the middle group (two people) ranked algorithm three the best overall.

In conclusion, choosing the top notes usually best selects the melody of a piece of music, but garners extra notes. The other three methods often select a part which has no notes in common with the melody of the music; as our results show, there can be almost no notes in common between the melodies generated by these algorithms.

There are several improvements that can be made to the above algorithms to make them more effective. The first algorithm often includes notes that have a much lower or higher pitch than the majority of the notes. In

[1]The MIDI files are available at **http://www.mds.rmit.edu.au/~sandra/melexp.**

	Algorithm				Best
	One	Two	Three	Four	Alg
1	18, 0.70/0.13	30, 0.00/0.00	14, 1.00/1.00	17, 0.76/1.00	3
2	10, 1.00/1.00	24, 0.00/0.03	19, 0.16/0.47	21, 0.16,0.47	1
3	9, 1.00/1.00	32, 0.00/0.00	20, 0.13/0.19	19, 0.13/0.19	1
4	14, 1.00/1.00	14, 1.00/1.00	13, 1.00/1.00	32, 0.35/0.55	3
5	11, 1.00/1.00	22, 1.00/1.00	13, 1.00/1.00	13, 0.93/1.00	1
6	16, 0.99/0.34	12, 1.00/1.00	16, 0.91/0.38	32, 0.00/0.00	2
7	14, 0.99/0.69	12, 1.00/1.00	26, 0.00/0.00	24, 0.00/0.00	2
8	22, 1.00/1.00	14, 1.00/1.00	16, 1.00/1.00	25, 0.43/0.93	2
9	13, 1.00/1.00	31, 0.11/0.59	21, 0.02/0.03	20, 0.02/0.03	1
10	13, 1.00/1.00	14, 0.78/1.00	32, 0.22/0.25	19, 0.30/0.49	1

Table 1: *Results of ranking melody extraction algorithms. The first number in each algorithm column is the sum of the rankings given for the melodies by the 10 evaluators. The second number is the proportion of notes in common with the melody selected as the best by the evaluators, expressed as a ratio of number of common notes over the number of notes in the best melody. The third number is the ratio of the number of common notes over the number of notes in the melody generated by that algorithm. For example, the most successful algorithm for piece number one was judged to be algorithm number 3. The number of notes it had in common with algorithm one divided by the number of notes in algorithm three's melody equals 0.70. The number of common notes divided by the number of notes in algorithm one's melody is 0.13.*

some circumstances these could be removed. The part extraction algorithm could be improved to consider all notes that start simultaneously as a group. Entropy could be combined with average pitch when selecting the melody part. We are currently exploring these approaches.

5 Searching Music

Several researchers have explored the problem of searching music databases for melodies. Approaches have included: extending the relational model for the storage of music using temporal database techniques [9]; performing exact matches using grep on melodies that are stored as a series of relative pitches in a text format [13]; using Pat trees to index melodies, where the notes of each measure of the melody were combined to determine the "chord" and this chord sequence was used as an index term [5]; creating indexes of melodies, indexing from the start of each phrase, reasoning that users usually start a query at the start of a phrase [21].

In work on contour Ghias et al. [12] used a collection of MIDI files as data. Melodies were extracted from these files as contour strings. These were stored in a text file and searched for matches to users' hummed queries. McNab [17, 18] converted melodies to contour strings, but also explored the effect of storing exact interval and contour and exact rhythm information. He discovered that after exact interval and rhythm, exact contour and rhythm was the combination with the greatest discriminatory power. Other methods explored include the encoding of whether a note is stressed or not [1, 14, 20], which would be important for ranking similar melodies, since experience suggests that those melodies that only differ on an unstressed note are more similar than those that differ on a stressed note. The approach of both Schaffrath [20] and Bakhmutova et al. [1, 14] was to use diatonic information, that is, the notes were numbered according to the note number in the scale or key of the music. This did not allow for music outside the key. Each of these techniques could be used in conjunction with a melody extraction algorithm.

To allow for queries that start anywhere within a melody, indexing via n-grams could be useful, where an n-gram is a sequence of n consecutive notes and each such sequence is extracted from the music; similar techniques are used for the related problems of string indexing and genomic indexing [23, 24] and have been explored to some extent for music [8]. With melodic data, however, it is not particularly useful to represent the data as an absolute pitch, since the same melody can be played or sung at different pitches.

Contour could be used instead, by representing melodies with an alphabet of three symbols, say "u" for up, "d" for down and "s" for same pitch. For long strings of contour information, it may be possible to uniquely identify a melody in a database. The advantage of using contour is that if a query is entered by humming the contour is usually correct, but the pitch intervals may not be, due to the difficulty in singing accurately. In a large database, however, more precise information may be needed. We examined the distribution of the contour n-grams found in a collection of 500 MIDI files. These files contained a total of 2 697 tracks that were each processed individually to extract their melody n-grams. A typical query of six notes would result in about 330 tracks being returned. In the collection of nearly 15 000 files, about 1 220 tracks of about

65 000 are retrieved. Assuming that the user needs to listen to a 10-second excerpt of each melody, it would take over three hours to determine which answers are relevant. Clearly a longer string of notes would be required in queries, or greater precision in the search for melodies. To retrieve less than 10 tracks from 2 697 using contour requires a query consisting of 12 notes. A more precise approach combines the use of contour with either more precise information such as exact interval n-grams.

6 Conclusions

Much music is now available online, as either performance information or music notation. The value of this music would be greatly enhanced by the existence of techniques for searching it for familiar melodies. A central problem in such searching is that stored music typically consists of multiple simultaneous parts, each of which can be chordal, and it is often the case that none of these parts has a simple correspondence to the perceived melody.

We have reviewed the needs of likely users of music databases and the psychology of music perception, and based on this review proposed several techniques for extracting likely monophonic melodies from polyphonic music. Our experiments with ten pieces of music and eight subjects indicate that the simplest technique—in which the extracted melody consists of highest-pitch notes appearing in any track—is the most effective, despite the fact that it often garners additional notes.

We have also outlined possible approaches to techniques for searching of music databases, which we expect to be based on a combination of relative pitch, contour, and possibly stress. These matching techniques, like previous matching techniques described in the literature, rely on monophonic melodies: extraction of such melodies is a necessary precursor to searching of music.

Acknowledgements

We thank John Harnett for invaluable sys-admin assistance.

References

[1] V. D. Gusev Bakhmutova, I. V. and T. N. Titkova. A search and classification of imperfect repetitions in song melodies. *Acta et Commentationes Universitatis Tartuensis: Quantitative Linguistics and Automatic Text Analysis*, 827:20–32, 1988.

[2] James C. Bartlett and W. Jay Dowling. Recognition of transposed melodies: A key-distance effect in developmental perspective. *Journal of Experimental Psychology: Human Perception and Performance*, 6(3):501–515, 1980.

[3] David C. Blair. STAIRS redux: Thoughts on the STAIRS evaluation, ten years after. *Journal of the American Society for Information Science*, 47:4–22, 1996.

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

[5] Ta-Chun Chou, Arbee L. P. Chen, and Chih-Chin Liu. Music databases: Indexing techniques and implementation. In *Proceedings IEEE International Workshop in Multimedia DBMS*, 1996.

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

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

[8] J. Stephen 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.

[9] Barry M. Eaglestone. Extending the relational database model for computer music research. In Alan Marsden and Anthony Pople, editors, *Computer Representations and Models in Music*, pages 41–66. Academic Press, 1992.

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

[11] R. Frances. *La Perception de la Musique*. 1958. Translated by W. Jay. Dowling.

[12] 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.

[13] Michael Hawley. Structure out of sound. Email discussion with the author about his thesis.

[14] Vladimir D. Gusev Irene V. Bakhmutova and Tatiana N. Titkova. The search for adaptations in song melodies. *Computer Music Journal*, 21(1):58–67, 1997.

[15] Carol L. Krumhansl and Roger N. Shepard. Quantification of the hierarchy of tonal functions within a diatonic context. *Journal of Experimental Psychology: Human Perception and Performance*, 5(4):579–594, 1979.

[16] Alan 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.

[17] Rodger J. McNab. Interactive applications of music transcription. Master's thesis, Department of Computer Science, University of Waikato, New Zealand, 1996.

[18] Rodger J. McNab, Lloyd A. Smith, Ian H. Witten, Clare L. Henderson, and Sally Jo Cunningham. Towards the digital music library: Tune retrieval from acoustic input. In *Digital Libraries Conference*, 1996.

[19] G. Salton. *Automatic Text Processing: The Transformation, Analysis and Retrieval of Information by Computer*. Addison-Wesley, 1989.

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

[21] Kazuhiro Mochizuki Tetsuya Kageyama and Yosuke Takashima. Melody retrieval with humming. In *ICMC Proceedings 1993*, 1993.

[22] Rene van Egmond and Dirk-Jan Povel. Perceived similarity of exact and inexact transpositions. *Acta Psychologica*, 92:283–295, 1996.

[23] 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.

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