A Geometric Approach to Music Pattern Matching for "The Algorhythmics"

Anna Lubiw and Luke Tanur
School of Computer Science
University of Waterloo

August 6, 2004

Abstract

The music pattern matching problem is to find occurrences of a small fragment of music called the "pattern" in a larger body of music called the "score". For symbolic (rather than audio) representation of music, a geometric approach models this as a problem of translating a set of horizontal line segments in the plane to find the best match in a larger set of horizontal line segments. We report on our algorithm to do this; it uses fairly general weight functions to measure the quality of a match, thus enabling approximate pattern matching. The algorithm's running time is $O(nm \log m)$, where n is the size of the score and m is the size of the pattern. We show that the problem, in this geometric formulation, is unlikely to have a significantly faster algorithm because it is at least as hard as a basic problem called 3-SUM that is conjectured to have no subquadratic algorithm. We apply our algorithm to measure similarities and differences among a collection of seven rock/blues pieces played by The Algorhythmics, a band composed of computer scientists from McGill.

1 Introduction

Music information retrieval is a rapidly evolving, multi-disciplinary research area [7, 5]. One of the problems at its core is the "music pattern matching problem"—to find occurrences of a small fragment of music (the "pattern") in a larger body of music (the "score").

The techniques required for this problem differ depending on whether the music is represented symbolically or as audio. This paper focuses on the former; for literature on audio representations and the pattern matching problem in that context, see [11].

With music represented symbolically, there are still a variety of approaches to the music pattern matching problem. Efforts are underway to compare these approaches on large data sets, see Downie [8]. Techniques based on string matching have been most heavily explored [15]. These include edit distance [20] and n-gram [9] techniques. Since these algorithms work on sequences, polyphonic music poses a great challenge, though there have been attempts to handle polyphony in this framework [16, 6].

For polyphonic music, the pattern matching problem is more tractable when music is represented in a richer, more geometric format than as a 1-dimensional string—when it is represented as line segments in the plane [23], weighted point sets in the plane [22], or multi-dimensional point sets [24].

Our work explores the possibilities of a particular geometric approach to music pattern matching. We model each note as a line segment in the plane—see Figure 1. The vertical axis corresponds to pitch and the horizontal axis corresponds to time; in particular, the length of a line segment indicates the duration of the note. This representation is a natural one and has been used by many others, for example in the Music Animation Machine [19] and by Brinkman and Mesiti [4].

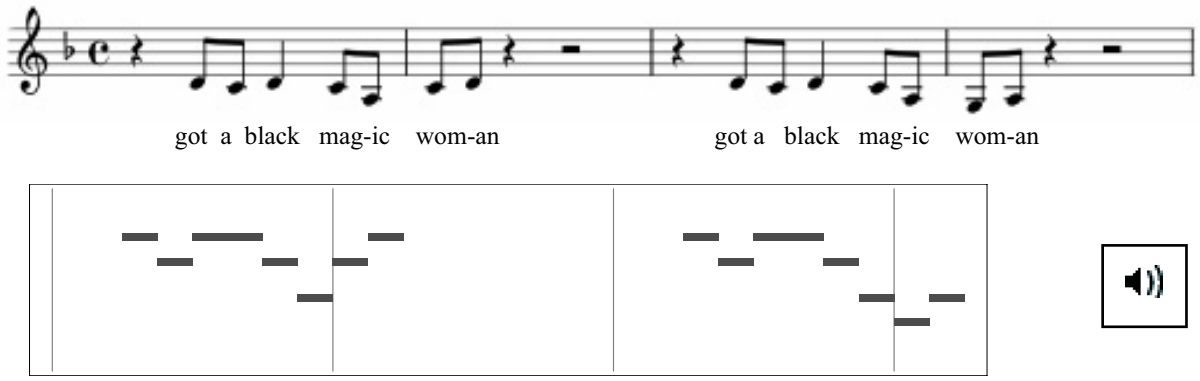

Figure 1: The main theme from "Black Magic Woman" by Santana, and the representation as line segments. If you are viewing this with Acrobat Reader, click on the sound icon.

Matching the pattern into the score means translating the pattern relative to the score, where "translation" is used in its mathematical sense. Imagine the pattern drawn on a transparent sheet that can be shifted horizontally and vertically over the score to find the best position. The vertical shift corresponds to transposing the pattern. The horizontal shift corresponds to locating the pattern in time. Some matches are better than others. An *exact match* is a translation of the pattern so that each line segment of the pattern exactly matches a line segment of the score. Exact matches have limited applicability—they encompass transposition, but allow no other variation. For a richer set of possibilities, we introduce weight functions and we search for matches of optimum weight.

Algorithms using this approach have been developed for some specific weight functions. Ukkonen, Lemström, and Mäkinen [23] define the weight of a match to be the sum of the lengths of the overlaps of pattern and score line segments. They give an algorithm to find maximum weight matches of a monophonic pattern in a polyphonic score.

A weight function that measures the area between a monophonic pattern and a monophonic score was used in a series of papers. The first paper, by Ó Maidín [18], introduced the weight function. The second, by Francu and Neville-Manning [12], gave an algorithm to find maximum weight matches. The algorithm is straightforward, and assumes the notes are expressed in terms of equal time steps (effectively the running time depends on the lengths of the notes, not just their number). The third paper was by Aloupis, Fevens, Langerman, Matsui, Mesa, Rappaport, and Toussaint [1]. They concentrated on the problem of match-

ing a repeated or circular melody against itself, though the results apply more generally. Their algorithm uses a more sophisticated and efficient method of finding maximum weight matches.

1.1 Results

Our results are being presented at the International Symposium on Music Information Retrieval [17] and in the Master's thesis of the second author [21]. In the present paper we give a brief version of our results, and we apply our music pattern matching algorithm to measure the similarities and differences between the main themes of some pieces played by Godfried Toussaint's band, "The Algorhythmics" [2].

Our approach is to use the geometric model described above. We introduce a weighting scheme that encompasses both of the measures mentioned above, and many more. We can, for example, assign weights depending on the interval between a note of the pattern and a note of the score; for example, matching notes an octave apart could contribute more weight than matching notes an augmented 4^{th} apart. Mongeau and Sankoff [20] used such a weighting scheme in their edit distance algorithm.

In Section 2 we describe our algorithm to solve the music pattern matching problem in time $O(nm \log m)$ where m is the size of the pattern and n is the size of the score. This is the same running time as that achieved by Ukkonen, Lemström, and Mäkinen [23] in their algorithm to maximize the length of pattern-score overlap. It is also the same running time as achieved by Aloupis, Fevens, Langerman, Matsui, Mesa, Rappaport, and Toussaint [1] in their algorithm to minimize the area between pattern and score. The running time of our algorithm is also competitive with other approaches to the music pattern matching problem, such as edit distance techniques. It is, however, disappointing in the sense that string pattern matching can be done much more efficiently, in linear time $O(n + m)$. The quadratic time behavior for music pattern matching is acceptable for small input sizes, but is prohibitively slow for huge ones, such as those envisioned in google-style music query systems.

However, we argue in Section 3 that for this geometric approach, quadratic behaviour is the best that can be achieved without a significant breakthrough in some basic algorithm design problems. In particular, our model of the music pattern matching problem includes as a special case a problem about containment of points in line segments. This latter problem is known to be equivalent, in terms of computational complexity, to other problems for which no one has a subquadratic algorithm, and for which it is conjectured that no such algorithm exists [3]. This is not a proved lower bound, but it is evidence towards a lower bound, which, given the dismal state of lower bound techniques, is something. We know of no previous lower bound arguments in music pattern matching.

In Section 4 we apply our algorithm to measure similarities and differences between some rock/blues pieces played by "The Algorhythmics". Applications of our algorithm to pattern matching in classical music, both monophonic and polyphonic, can be found in the full version [17, 21].

2 Algorithm

In this section we give an overview of our algorithm. Further details can be found in the full version [17, 21].

2.1 Overview

For the music pattern matching problem, we are given a pattern of m notes and a score of n notes, represented as line segments. We are also given a weight function with which to evaluate a translation of the pattern in the score. We wish to find the translation of the pattern in the score that has maximum weight. More generally, we want not only "the best" match, but a number of good matches.

Our algorithm is an efficient version of the most basic approach to this music pattern matching problem: to try all possible translations of the pattern in the score, and compute the weight of each, in order to find the translations that have maximum weight. The algorithm of Ukkonen et al. [23] uses this same approach, and our algorithm can be viewed as an extension of theirs to more general weight functions.

There are two main ingredients for an efficient implementation. One is to identify a bounded-size set of candidate translations that includes all possible optimum solutions to the music pattern matching problem. We show a bound of $O(nm)$ on the number of candidate translations. The other ingredient is to avoid computing the weight of each translation from scratch, but rather to go through the translations in an appropriate order and update efficiently from one translation to the next. This is possible for many, though not all, weight functions. We discuss the allowable weight functions in Section 2.3, and show how to preprocess the score in time $O(n)$ to achieve an update time of $O(\log m)$ to find the next translation and $O(1)$ to compute its weight.

Putting these together, we obtain an $O(nm \log m)$ algorithm for the music pattern matching problem.

In the analysis of our algorithm, we make crucial use of the assumption that musical pitches come from a discrete set. Our examples use the 128 MIDI values based on semi-tones, but our algorithm would apply to any discrete set, for example scale degrees, or the base-40 representation of Hewlett [14]. Our running time of $O(nm \log m)$ hides the dependence on 128. To put it more precisely, our running time is actually $O(nm(d + \log m))$ where d is the size of the discrete pitch set. We remark that, although 128 is a constant, it is a rather large constant, and an algorithm whose running time does not depend on d would certainly be desirable. This is possible for specialized weight functions and/or monophonic music, as we discuss in Section 2.7. It remains an open problem to achieve this independence from d for polyphonic music and our general weight functions.

2.2 Notation and Input Data

A note s is represented by its starting time, $\sigma(s)$, its ending time, $\tau(s)$, and its pitch, $\pi(s)$. The note s corresponds to the horizontal line segment from the point $(\sigma(s), \pi(s))$ to the point $(\tau(s), \pi(s))$. We assume that the notes of the score are given sorted by $\sigma(s)$. This is

true for data coming from a MIDI file, but other data may need to be sorted at an extra cost of $O(n \log n)$.

For the purpose of our algorithm, we need an ordered list of all the distinct $\sigma(s)$ and $\tau(s)$ values. These are called the *time points* of the score. There are at most $2n$ time points, and they can be computed in $O(n)$ time assuming a constant bound l on the maximum polyphony of the score (i.e. the maximum number of notes being played at any one time).

2.3 The Weight Model

We use a weight function to measure deviations of the translated pattern from the score. Note that translating the pattern is "free"; only the differences between the translated pattern and the score count. Our weight functions are additive—i.e. the weight of a particular translation of the pattern is the sum of the weights of its notes.

It seems natural that matching a long note should count more than matching a short note. We effect this by setting the weight of a translated note to be proportional to its length. For example, a half note that perfectly matches into the score counts twice as much as a quarter note that perfectly matches into the score. Thus the weight of a translated note p matching a score note s that occupies the same time interval will be $(\tau(p) - \sigma(p))f(\pi(s), \pi(p))$, where f is a function of the pitches of s and p. More generally, if s and p overlap in time, we use the length of the overlap instead of $(\tau(p) - \sigma(p))$.

When a translated pattern note overlaps in time with several notes of a monophonic score, we allow pieces of the pattern note to match with different notes of the score. This captures what Mongeau and Sankoff [20] call "fragmentation", where one note is replaced by several. The opposite transformation, "consolidation", is captured when several pattern notes match to the same note of the score. A portion of a translated pattern note may match a portion of a note of the score only if they occupy the same time span. See Figure 2(a).

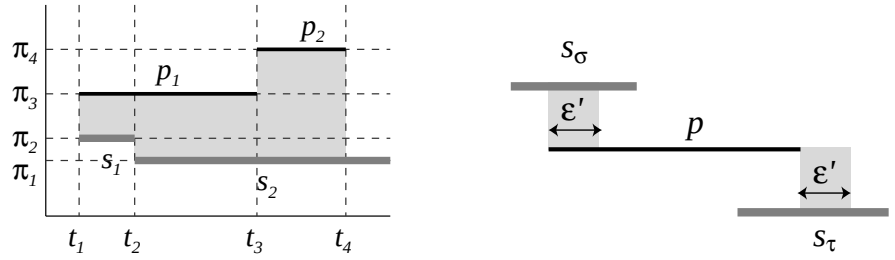

Figure 2: Computing the weight function: (a) weight is $(t_2 - t_1)f(\pi_2, \pi_3) + (t_3 - t_2)f(\pi_1, \pi_3) + (t_4 - t_3)f(\pi_1, \pi_4)$; (b) the effect of a shift by ϵ'. Notes of the score are shaded.

Polyphonic music may have several pattern notes and several score notes occupying the same time span. In this case we match each (piece of a) pattern note to the single note of the score that gives the best weight.

A very simple version of such a weight function sets $f(\pi(s), \pi(p))$ to be 1 if $\pi(s) = \pi(p)$, and 0 otherwise. In geometric terms, the weight of a translation of the pattern is then the

sum of the lengths of the overlap of pattern and score line segments. This is the weight function used by Ukkonen et al. [23].

A more complicated version of such a weight function sets $f(\pi(s), \pi(p))$ to be the difference between $\pi(s)$ and $\pi(p)$. Using MIDI pitches, this is the number of semi-tones in the interval between the two notes. In geometric terms, this weight function measures the area between the translated pattern and the score. This weight function was used for the case of monophonic music by Ó Maidín [18], Francu and Neville-Manning [12], and Aloupis et al. [1].

More generally, we can define $f(\pi(s), \pi(p))$ to depend on the interval between the two notes in a more complicated way. For example, we can assign a better value to an interval of 7 semi-tones (a perfect 5^{th}) than to the smaller interval of 6 semi-tones. Mongeau and Sankoff [20] use a scheme like this in their edit-distance algorithm, assigning weights to intervals in increasing order of dissonance. The particular weighting of intervals that we use in our examples is shown in Table 1. We make no claim about these weights being ideal; further experimentation would be good.

Our method can extend to functions $f(s, p)$ that depend on other properties of the notes s and p than pitch—for example stress, dynamics, relative position in the bar, etc.

Interval apart (in semi-tones)	Weight
perfect unison (0)	0
minor 2^{nd} (1)	0.1
major 2^{nd} (2)	0.4
minor 3^{rd} (3)	0.6
major 3^{rd} (4)	0.6
perfect 4^{th} (5)	0.8
perfect 5^{th} (7)	0.4
minor 6^{th} (8)	0.7
major 6^{th} (9)	0.7
major 7^{th} (11)	0.3
perfect octave (12)	0.2
minor 9^{th} (13)	0.3
all other intervals	1

Table 1: The weighting scheme used for our experiments

2.4 The Set of Candidate Translations

We can think of the score as lying in a grid formed by the 128 MIDI pitches along the vertical axis, and the time points of the score along the horizontal axis. This grid has size at most $2n \times 128$.

Claim 1 *With any weight function as described above, there will be an optimum match of the pattern into the score that has some line segment of the pattern starting or ending at one of these grid points.*

The proof of this claim can be found in the full version [17, 21]. Thus, for any weight function as described above, there are at most $128 \cdot 4 \cdot nm$ candidate translations.

2.5 Preprocessing

In order to quickly determine the weight of a translated note, we precompute a weight matrix W based on the score and the given weight function. Matrix W has a row for each of the 128 MIDI pitches, and a column for each of the time points of the score, of which there are at most $2n$. Thus it has size at most $128 \times 2n$, which is $O(n)$.

For pitch π and time point t, the corresponding matrix entry, $W(\pi, t)$, contains the weight factor to be applied to a note of the pattern translated to pitch π, and going from time point t to the next time point t'. Thus such a translated pattern note contributes $(t'-t)W(\pi, t)$ to the weight of a match. In terms of the function f described above, $W(\pi, t) = \max\{f(\pi(s), \pi) : s$ is a score note that includes the time interval $(t, t')\}$.

We compute W by iterating through the notes s of the score, and updating $W(\pi, t)$ as π ranges through the 128 pitch values, and t ranges through the time points from $\sigma(s)$ up to, but not including, $\tau(s)$. Each of the $O(n)$ matrix positions will be updated at most l times, where l is the maximum polyphony of the score. The total work is thus $O(n)$.

2.6 The Main Matching Algorithm

We try each possible candidate translation (t, π), where t is the translation applied to the time coordinate, and π is the translation applied to the pitch coordinate.

We try each value of t, in order. We call it an *event* when the start or the end of a translated pattern note lines up with a time point of the score. We go through the events in order of their translation values. Note that several events may occur at the same translation value, but for book-keeping purposes we handle them one at a time. The number of events is at most $4nm$. Using a heap we can find the next event in $O(\log m)$ time.

As we go through the events—i.e. the values of t—we maintain information for each value of π—i.e. each transposition of the pattern. There are at most $2 \cdot 128$ values of π. The information we maintain includes the weight for the candidate translation (t, π), but also other information that allows us to update each of these weights in $O(1)$ time. Details can be found in [17, 21].

2.7 Running Time

We analyze the running time in terms of n, m, d. We go through $O(nm)$ events. Finding the next event takes $O(\log m)$ time. Updating the information for each of the $O(d)$ values of π takes $O(1)$ time. The total is $O(nm(\log m + d))$. The space used is $O(n + d)$. Finding the k best matches adds $O(k)$ to the time and space, using linear time median finding.

Previous algorithms for specialized weight functions—the total overlap length of Ukkonen et al. [23], and the area-between of Aloupis et al. [1]—do not have the bad dependence on d that our algorithm has. In essence, this is because the number of candidate translations is reduced in these cases from our bound of $O(dnm)$ to $O(nm)$. The crucial property is

that there will be an optimum match in which some line segment of the pattern starts [or ends] exactly where a line segment of the score starts [or ends]. This happens if the weight function only measures exact overlap of pattern and score line segments. It also happens for the distance weight function if the score and pattern are monophonic. With a polyphonic score and the distance weight function this property fails, and we need the larger bound. In the next section we consider the biggest factor in the running time, the quadratic $O(nm)$ behavior.

3 Barriers to a Faster Algorithm

Our algorithm for the music pattern matching problem takes time $O(nm \log m)$ for a score of size n and a pattern of size m. A subquadratic algorithm with running time $O(n + m)$, such as is achievable for string pattern matching, or even $O((n + m) \log n)$, would be vastly preferable, and would make the algorithm practical for use in large music databases. In this section we show that such an efficient algorithm will be a major challenge.

More specifically, we show that the music pattern matching problem, in this geometric formulation, includes as a very special case a problem called "Segments Containing Points" which is at least as hard as the 3-SUM problem—and *that* problem is conjectured to have no algorithm with a subquadratic running time. We expand on these points in the remainder of this section. The two problems are as follows:

3-SUM: Given a set of n integers, does it contain numbers a, b, c with $a + b + c = 0$?

Segments Containing Points (SCP): Given a set P of m numbers and a set Q of n pairwise-disjoint intervals, is there a translation u such that $P + u \subseteq Q$?

The geometric formulation of the music pattern matching problem includes SCP as the very special case where the pattern and the score have notes only on one pitch, the pattern notes are very short, and the 0-1 weight function is used.

Barequet and Har-Peled [3] prove that an algorithm with running time $o(nm)$ for the SCP problem would imply an algorithm with running time $o(n^2)$ for the 3-SUM problem. Thus SCP is "3-SUM hard". The class of 3-SUM hard problems was introduced by Gajentaan and Overmars [13], who show that a number of different problems are 3-SUM hard. Although this is not a proved lower bound (see [10]) a subquadratic algorithm for any of these problems would be a major breakthrough.

Thus a subquadratic algorithm for the geometric version of the music pattern matching problem would have implications far beyond music information retrieval.

Recall that in terms of d, the pitch set size, our running time was $O(nm(d + \log m))$. From a practical point of view, at least for small patterns, the factor of $d = 128$ in our algorithm is probably more prohibitive than the factor of m. Certainly, given the above, it is a more tractable challenge to attempt to reduce the dependence on d.

4 Applying our Algorithm to "The Algorhythmics"

In this section we describe the results of running our algorithm on some of the songs performed by The Algorhythmics.

The Algorhythmics [2] is a band formed by David Avis, David Eu, Gena Hahn, Jörg Kienzle and Godfried Toussaint, all computer scientists at McGill University. We concentrated on their first set, which consists of the following rock and blues songs: "Lay Down Sally" by Eric Clapton; "I Got My Mojo Working" by Muddy Waters; "Crazy Little Thing Called Love" by Queen; "Black Magic Woman" by Santana; "Fire" by Bruce Springsteen; "Hoochie Coochie Man" by Muddy Waters; and "Gloria" by Van Morrison.

We extracted small themes from each piece and compared all pairs, but we will present here only the results of comparing one pattern against all the themes. The pattern we chose (because it gave some interesting results) is a 2-bar phrase from "Black Magic Woman"—the first half of the 4-bar theme shown in Figure 1. Our results are shown in Table 2.

Song Title	Value of Best Match (%)
Lay Down Sally	83.8
I Got My Mojo Working (theme 1)	89.2
I Got My Mojo Working (theme 2)	70.2
Crazy Little Thing Called Love	70.0
Fire	83.8
Hoochie Coochie Man	65.8
Gloria	66.9

Table 2: values of the best match of the "Black Magic Woman" pattern into other themes.

In the remainder of this section we show some of the actual matches, and discuss the results. The best match found by our algorithm is a remarkable similarity between the "Black Magic Woman" pattern and the first theme from "I Got My Mojo Working". (This is especially appropriate given that "mojo" means "magic spell", which is also what "Black Magic Woman" is about!)

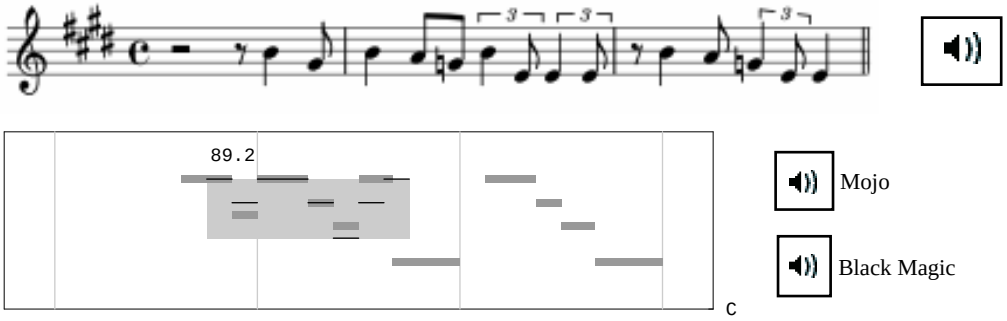

Figure 3: The first theme from "I Got My Mojo Working" (above) and the best match of the "Black Magic Woman" pattern into that theme (below). The translated pattern is indicated with solid line segments and the theme with shaded line segments. Clicking on the sound icons in Acrobat Reader plays the associated parts.

Our algorithm does not always do so well. One of matches tied for second place is the match with the "Fire" theme shown in Figure 4. Although our algorithm gives the

match a good weight, the musical similarity is not strong. Our algorithm gave this match a good weight because each note of the pattern is very close to a note of the score under our weighting scheme. One of the reasons this match is not musically significant is because several different pattern notes are matched to one longer note of the score. We do know how to fix our algorithm to lessen the weight given to matches like this: we should allocate extra points when the start of a pattern note matches the start of a score note. This will be easy to do.

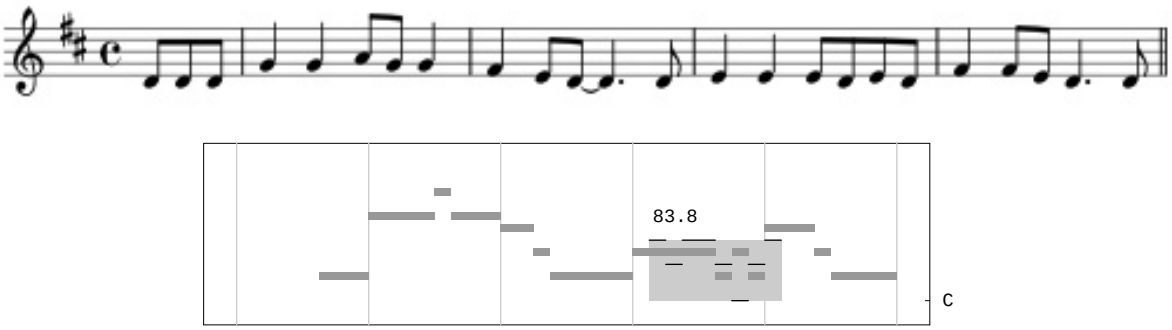

Figure 4: The theme from "Fire" (above) and the best match of the "Black Magic Woman" pattern into that theme (below). The translated pattern is indicated with solid line segments and the theme with shaded line segments.

We now turn to some of the bad matches found by our algorithm. In all these cases, our musical judgement agrees with our algorithm—the pieces are different enough that no sensible match is possible. In the case of the themes from "Hoochie Coochie Man" (see Figure 5) and "Gloria" (see Figure 6), the best matches include pattern notes that align with rests in the theme—something that our algorithm penalizes heavily. There are also pattern notes that differ significantly in pitch from the notes of the theme.

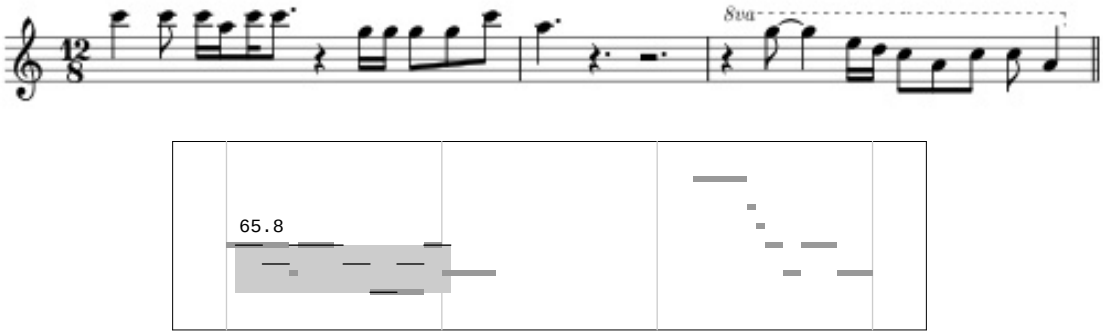

Figure 5: The theme from "Hoochie Coochie Man" (above) and the best match of the "Black Magic Woman" pattern into that theme (below). The translated pattern is indicated with solid line segments and the theme with shaded line segments.

The third example of a bad match uses as a theme a fragment of the "I Got My Mojo Working" melody that occurs later in the piece. See Figure 7. This behaves quite differently

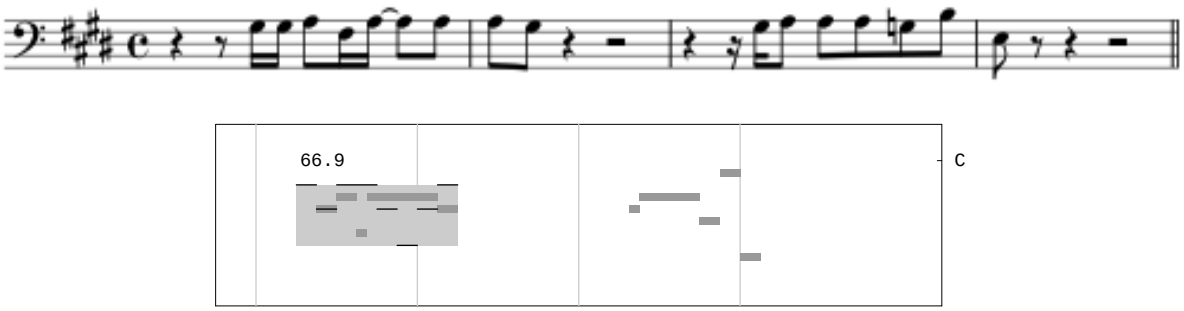

Figure 6: The theme from "Gloria" (above) and the best match of the "Black Magic Woman" pattern into that theme (below). The translated pattern is indicated with solid line segments and the theme with shaded line segments.

from the first "Mojo" theme that gave us such a good match! In this case, although there are no rests in the matched portion of the theme, the algorithm sensibly detects that the profile of the pitch direction (whether pitches move up or down) is very different in the pattern and the theme.

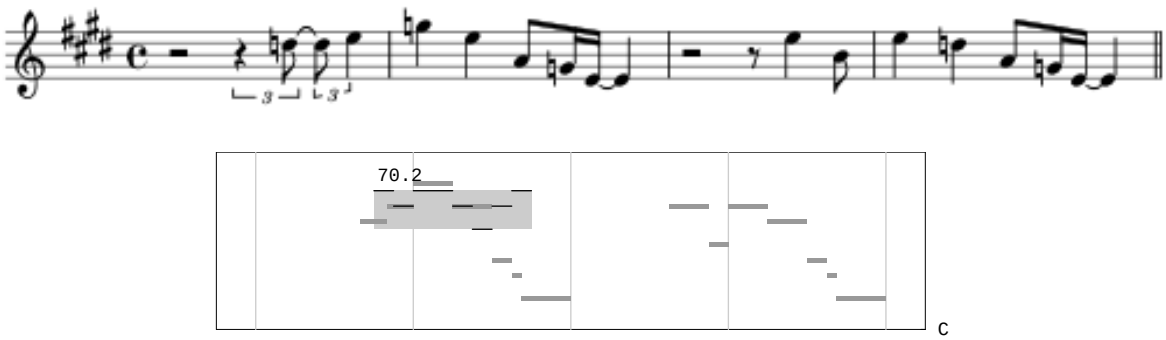

Figure 7: The second theme from "I've Got My Mojo Working" (above) and the best match of the "Black Magic Woman" pattern into that theme (below). The translated pattern is indicated with solid line segments and the theme with shaded line segments.

5 Conclusions

In this paper we have summarized our exploration of the possibilities and limitations of an approach to music pattern matching that uses a very natural geometric representation of music, and turns the problem into that of finding a "best" translation of a small set of line segments into a larger set. This geometric approach has been used before, but has not been explored as thoroughly as string matching techniques, even though it successfully deals with polyphony.

Our contribution is to show how this approach can be used together with fairly general weight functions to measure the quality of a match. This opens up the possibility of a rich range of approximate music pattern matching techniques. Our experiments here and in

[17, 21] only scratch the surface of what might be achievable. It is easy to imagine a variety of enhancements, for example: incorporating information about the key of tonal music; adding information about dynamics and stress; weighting more heavily those matches that occur at the beginnings of notes; allowing the user to specify which notes of the pattern are more important, etc.

Our algorithm runs in time $O(nm(\log m + d))$ where d is the size of the pitch set, n is the size of the score, and m is the size of the pattern. This is fine for small patterns, but too expensive for larger ones. We have argued that the factor $O(nm)$ is likely to be very hard to improve. Improving the dependence on d is perhaps more tractable, and also more relevant for reasonably sized patterns.

6 Acknowledgements

We thank Erna Van Daele for musical discussions and advice; Ian Munro for the idea of how to find the best k matches; and Therese Biedl, Dan Brown, and Anne-Marie Donovan for useful suggestions.

References

[1] G. Aloupis, T. Fevens, S. Langerman, T. Matsui, A. Mesa, D. Rappaport, and G. Toussaint. Computing the similarity of two melodies. In *Proceedings of the 15th Canadian Conference on Computational Geometry*, pages 81–84, 2003.

[2] D. Avis, D. Eu, G. Hahn, J. Kienzle, and G. Toussaint. The Algorhythmics. http://www.cs.mcgill.ca/%7Ejoerg/personal/algorhythmics/.

[3] G. Barequet and S. Har-Peled. Polygon-containment and translational min-Hausdorff-distance between segment sets are 3SUM-hard. *International Journal of Computational Geometry and Applications*, 11(4):465–474, 2001.

[4] A. Brinkman and M. Mesiti. Graphic modeling of musical structure. *Computers in Music Research*, 3:1–42, 1991.

[5] D. Byrd and T. Crawford. Problems of music information retrieval in the real world. *Information Processing and Management*, 38:249–272, 2002.

[6] M.J. Dovey. A technique for "regular expression" style searching in polyphonic music. In *Proceedings of the 2nd International Conference on Music Information Retrieval (ISMIR 2001)*, pages 179–185, 2001.

[7] J.S. Downie. Music information retrieval. *Annual Review of Information Science and Technology*, 37:295–340, 2003.

[8] J.S. Downie. Toward the scientific evaluation of music information retrieval systems. In *Proceedings of the 4th International Conference on Music Information Retrieval (ISMIR 2003)*, pages 25–32, 2003.

[9] S. Downie and M. Nelson. Evaluation of a simple and effective music information retrieval method. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 73–80, 2000.

[10] Jeff Erickson. Lower bounds for linear satisfiability problems. *Chicago Journal of Theoretical Computer Science*, 1999(8), 1999.

[11] J. Foote. An overview of audio information retrieval. *Multimedia Systems*, 7:2–11, 1999.

[12] C. Francu and C.G. Nevill-Manning. Distance metrics and indexing strategies for a digital library of popular music. In *Proc. IEEE International Conference on Multimedia and EXPO (II)*, pages 889–894, 2000.

[13] A. Gajentaan and M.H. Overmars. On a class of $o(n^2)$ problems in computational geometry. *Computational Geometry:Theory and Applications*, 5(3):165–185, 1995.

[14] W.B. Hewlett. A base-40 number line representation of musical pitch notation. *Musikometrika*, 50:1–14, 1992.

[15] K. Lemstrom. *String Matching Techniques for Music Retrieval*. PhD thesis, University of Helsinki, Department of Computer Science, 2000.

[16] K. Lemstrom and J. Tarhio. Transposition invariant pattern matching for multi-track strings. *Nordic Journal of Computing*, 10:185–205, 2003.

[17] A. Lubiw and L. Tanur. Pattern matching in polyphonic music as a weighted geometric translation problem. In *Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR 2004)*, 2004.

[18] D. Ó Maidín. A geometrical algorithm for melodic difference. In W.B. Hewlett and E. Selfridge-Field, editors, *Melodic Similarity: Concepts, Procedures, and Applications*, volume 11 of *Computing in Musicology*, pages 65–72. MIT Press, 1997–1998.

[19] S. Malinowski. Music animation machine. http://www.well.com/user/smalin/mam.html.

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

[21] L. Tanur. A geometric approach to pattern matching in polyphonic music. Master's thesis, School of Computer Science, University of Waterloo, 2004. to appear.

[22] R. Typke, P. Giannopoulos, R. C. Veltkamp, F. Wiering, and R. van Oostrum. Using transportation distances for measuring melodic similarity. In *Proceedings of the 4th International Conference on Music Information Retrieval (ISMIR 2003)*, pages 107–114, 2003.

[23] E. Ukkonen, K. Lemström, and V. Mäkinen. Geometric algorithms for transposition invariant content-based music retrieval. In *Proceedings of the 4th International Conference on Music Information Retrieval (ISMIR 2003)*, pages 193–199, 2003.

[24] G. A. Wiggins, K. Lemström, and D. Meredith. SIA(M)ESE: An algorithm for transposition invariant, polyphonic content-based music retrieval. In *Proceedings of the 3rd International Conference on Music Information Retrieval (ISMIR 2002)*, pages 283–284, 2002.