ALGORITHMS FOR POLYPHONIC MUSIC RETRIEVAL: THE HAUSDORFF METRIC AND GEOMETRIC HASHING

Christian André Romming
Stanford University
Dept. of Computer Science
romming@cs.stanford.edu

Eleanor Selfridge-Field
Stanford University
Dept. of Music
esfield@stanford.edu

ABSTRACT

We consider two formulations of the computational problem of transposition-invariant, time-offset tolerant, meter-invariant, and time-scale invariant polyphonic music retrieval. We provide algorithms for both that are scalable in the sense that space requirements are asymptotically linear and queries are efficient for large databases of music. The focus is on cases where a query pattern M consisting of m events is to be matched against a database N consisting of n events, and $m \ll n$. The database is assumed to be polyphonic, and the algorithms support polyphonic queries. We are interested in finding exact and proximate occurrences of the query pattern. The first problem considered is that of finding the minimum directed Hausdorff distance from M to N. We give a $(2 + \epsilon)$-approximation algorithm that solves this problem in $O(nm)$ query time and $O(n)$ space. The second problem is that of finding all maximal subset matches of M in N, and we give an algorithm that solves this problem in $O\left(m^3 (k + 1)\right)$ query time and $O\left(w^2 n\right)$ space, where w represents the maximum window size and k is the number of matches. Using the same method, the problem can be solved in $O(m (k + 1))$ query time and $O(wn)$ space if we do not require the time-scale invariance property. The latter query time is asymptotically optimal for the given problem.

1 INTRODUCTION

We consider two formulations of the computational problem of content-based, polyphonic music retrieval. We suggest algorithms for both, and they are linked in the sense that they employ the same point-set representation of music. The first problem formulation, referred to as the *minimum Hausdorff distance* problem, considers the dissimilarity of two point sets to be related to the Euclidian distance between points in the set. The second, which we call the *maximal subset matching* problem, takes the similarity to be related to the number of points in one set that are in the same location as points in the other. Although these two problems might seem similar, the methods required to solve them are inherently different.

1.1 Related Work

It is argued by Di Lorenzo and Di Maio [3] that the Hausdorff distance has significant relationships with perceived musical similarity. The Hausdorff distance as formulated in this paper is related to the Earth Mover Distance studied by Typke [9]. The main differences lie in the metric used to measure similarity and the indexing methods employed in order to make queries efficient. Another related approach is that taken by Lubiw and Tanur [5], whose measure of similarity is based on the consonance of events that overlap in time. The maximal subset matching problem (and variants of it) has been studied by several groups. Some of the most notable algorithms are the SIA family, an excellent overview of which is given by Meredith [6]. A recent randomized algorithm building on the SIA algorithms is MSM, developed by Clifford et al. [2], which achieves a running time of $O(n \log n)$. Furthermore, problem P2 described by Ukkonen et al. [10] is closely related to the maximal subset matching problem as defined here. The most notable differences are that we also require note durations to match and that we allow arbitrary scaling of the pattern set in addition to translation.

1.2 Representation

We represent music as three-dimensional point sets, where the three dimensions are pitch, duration, and onset time. Point-set representations of music are common in the context of melodic similarity algorithms [2, 9]. Another popular representation format is sets of line segments, where the length of the line segments specifies the duration of a musical event [5, 10]. We choose the point-set representation for two main reasons. First, it is extensible in the sense that more features such as e.g. harmony and beat can be included by simply adding more dimensions. The algorithms in this paper will still work after such a modification (with slightly higher running times). Second, a point-set representation makes it easier to apply some of the ideas used in this paper, such as the Hausdorff metric and geometric hashing. Only notes are explicitly represented, and ornaments and grace notes are ignored.

Let us first give the details of the point-set representation we use. The pitch of an event is encoded using Hewlett's base-40 notation [4], so that translations in the pitch dimension leave all intervals unchanged. A logarithmic scale is used to represent duration, so that translations in the duration dimension correspond to scaling of actual durations. For example, the distance from a quarter note to an eighth note is the same as the distance from a whole note to a half note.

We identify two different ways of representing onsets, each with different musical properties. The first is to use a measure-relative scale, that is, we let all measures have a fixed length (independent of meter) and set onset coefficients accordingly. For example, a note that begins half-way through the fifth measure will have onset $4.5 \cdot l$, where l is the measure-length constant. The second is to use an absolute scale. In this system, the meter information is ignored, and the onset coefficient of a point would be proportional to the aggregate duration since the beginning of the work. Using either the measure-relative or the absolute scale, translations in the onset dimension clearly correspond to a time-shift of the music. To see the difference between the two scales, observe that a piece of music in 4/4 meter will have different onset coefficients from a piece with exactly the same notes in 3/4 meter only when the measure-relative scale is used.

Although translations have clear meanings in each of the dimensions in isolation, the effect in the onset dimension of a translation upon the duration dimension is slightly more involved. Let us assume that a translation of one positive unit in the duration dimension corresponds to a doubling of the duration of each note. Observe that note onsets are left unchanged by this transformation. Using the measure-relative scale, this transformation can equivalently be seen as a change in meter. This is illustrated in Fig. 1: The duration of each note in fragment (b) is twice that of the corresponding note in fragment (a), but the relative position of each note is the same in both sequences. We will exploit this feature when approaching the issue of time-scale invariance. An artifact of this representation is that fragment (a) is indistinguishable from fragment (c), which implies that though we might achieve the meter-invariance property, this interpretation is musically ambiguous. We would lose meter invariance if we use absolute onsets: The meter information is ignored, and so in order to achieve the transformation from (a) to (b) in Fig. 1 (ignoring the time signature and the bar lines), the translation in the duration dimension would have to be combined with a scaling by a factor of two in the onset dimension. Our algorithms can be used with both scale types.

1.3 Time-Scale Invariance

One element of the richness of polyphonic music lies in melodic imitation between voices. Imitations can be exact in every respect, but commonly the melody varies from one iteration to the next by octave or by transposition. In these cases, a query in which pitch and duration are defined should be adequate for retrieval. Since the need for pitch-invariance is widely accepted, we turned our attention to time-scale invariance. An imitative device in widespread use in European art music of several centuries

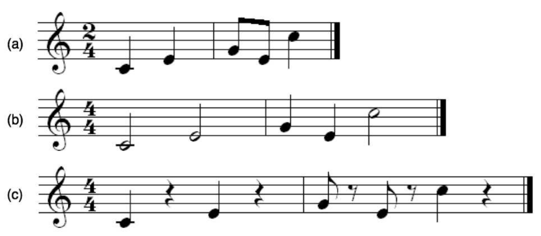

Figure 1. Three related fragments. (a) A simple note sequence. (b) The note sequence translated in the duration dimension. (c) The note sequence from (a) shown in a different meter with measure-relative onsets left unchanged.

was the mensuration canon, in which an imitation could be contracted or elongated by doubling or halving note-values. Contrapuntal techniques involving the augmentation or diminution of note-durations often noted in the work of J. S. Bach, left a significant mark on motivic procedures in later orchestral music.

1.4 Windowing

Windowing refers to the concept of splitting the music into (overlapping) segments whose events fit into a certain onset interval (referred to as *window length*). This idea is somewhat analogous to the concept of *n*-grams, which is used extensively in string matching. The *n*-grams technique has been applied in monophonic music query algorithms, but does not easily generalize to polyphonic music, where the target sequence may occur in any voice and may even move between voices [2]. Windowing is made possible by the fact that query patterns are local in time, that is, they usually span only a limited range in the onset dimension.

We use two forms of windowing. In section 2.1 we suggest that the running time of our algorithm can be improved if one of the subroutines queries a data structure consisting of a small window of points rather than the entire model point set. In the maximal subset matching algorithm presented in section 3.1 windowing is fundamental: The preprocessing step essentially takes windows as input, and the algorithm makes assumptions about the relative lengths of these windows and the query pattern lengths.

2 MINIMUM HAUSDORFF DISTANCE PROBLEM

We begin by specifying the first of the two problems we study in this paper, which is that of finding the minimal directed Hausdorff distance between a pattern M and a model N over all possible translations of the pattern M. Formally, the distance we want to find is

$$\min_{w \in \mathbb{R}^3} \left(\max_{m \in M} \left(\min_{n \in N} d\left(m + w, n\right) \right) \right), \tag{1}$$

where w is any translation vector and d is the Euclidean distance function. To understand what is meant by Hausdorff distance, consider a particular translation of the pat-

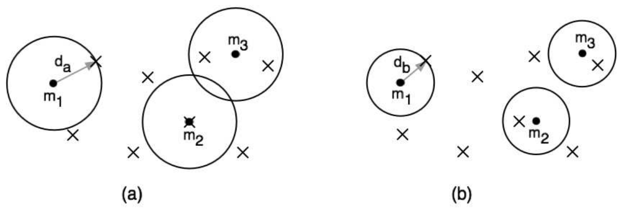

Figure 2. The concept of Hausdorff distance.

tern M. For each point in M, consider the distance to the closest point in N. The directed Hausdorff distance is the maximum of these distances. This is illustrated in Fig. 2(a): The crosses represent the point set N and the dots represent the pattern point set M. The Hausdorff distance from the dots to the crosses is the furthest distance from a dot to the cross that is nearest to it. Another way of seeing this distance is as the smallest radius that circles around each of the dots can have in order for each of the circles to contain at least one cross. As stated earlier, we allow the pattern point set M to undergo any translation w. Such a translation that decreases the Hausdorff distance is shown in Fig. 2(b): The dots have been shifted horizontally to the right, and this has decreased the distance from m_1 to the nearest cross from d_a to d_b. The task is to determine the translation vector which gives the smallest Hausdorff distance. The example presented here is for a two-dimensional representation, but the problem clearly generalizes to any point dimension.

2.1 Our Algorithm

There are two main steps in our algorithm. The first step is to *align* the pattern set M and the model set N with respect to some pair of points (a, b), where $a \in M$ and $b \in N$. Concretely, this means that we apply a translation w to every point in the pattern set M such that $a + w = b$. The next step is to find the nearest neighbor of each of the pattern points in the model point set under this translation. The Hausdorff distance is then computed based on this. We repeat this procedure for each of the points in the model set N. Alg. 1 gives the details of this procedure. We will provide details about the complexity of the algorithm, and also show that it is indeed a $(2+\epsilon)$-approximation to the directed Hausdorff distance problem outlined in the previous section.

Subsequent to the alignment, the nearest neighbor is found in the model set N for each of the points in the pattern set M. We use the approach proposed by Arya et al in [1], which during preprocessing of the data decomposes the set N into cells in a so-called BBD-tree structure. At query time, the cell to which the query point belongs is found, and a priority search routine processes nearby cells until the nearest neighbor is found. Note that this algorithm actually finds an *approximate* nearest neighbor, that is, it returns any point that is at most $(1+\epsilon)$ times as far away from the query point as its true nearest neighbor, where $\epsilon > 0$ is a constant of our choosing. The time required for each such nearest neighbor computation is $O\left(d\left\lceil 1+6\frac{d}{\epsilon}\right\rceil^d \log n\right)$, where d is the point dimension (in our case $d = 3$). The preprocessing time for the data structure is $O(dn \log n)$, and the space requirement is $O(dn)$.

Algorithm 1 Minimum Hausdorff distance algorithm

```
1. Pick any point $p$ in $M$
2. For every point $n_i$ in $N$
    (a) Translate $M$ by $(n_i - p)$
    (b) For every point in $m_j$ in $M$
    (c) Find the nearest neighbor $n_j$ of $m_j$ in $N$
            i. Compute the distance $d_{ij} = d(m_j, n_j)$
    (d) Set distance $i$, $d_i = \max_j \{d_{ij}\}$
3. Output the minimal alignment distance, $\min_i \{d_i\}$
```

2.2 Analysis

For each alignment, there are $(m-1)$ nearest neighbor queries to process, and there is a total of n candidate alignments. The total time complexity of our algorithm is thus $O(mn \log n)$. The only preprocessing required is creating the BBD-tree for the model points, and the space requirement is linear in the size of the model point set. The idea of exploiting the spread in the onset dimension through windowing (see section 1.4) can be used to reduce the time complexity of queries. To do this, we require an upper bound on the potential difference in onset between a query point and its nearest neighbor in the model set. This will determine a window length and hence a maximum size that the set of model points that fall within the limits of such a window can have. If we build the BBD-tree for each window, the query time is reduced to $O(nm \log w)$, where w is largest possible window point set size. Note that although w may be large, it is inherently linked to the representation and does not grow with either m nor n. It can thus be regarded as a constant, and the asymptotic running time of our algorithm is thus $O(mn)$. Time-scale invariance is achieved in a brute-force way by repeating the query process for different meters applied to the query sequence (see section 1.2). We do not discuss the details about the set of applicable meters here, other than to note that the set of applicable meters is finite and relatively small. We also omit the proof that the alignment method guarantees that the distance output is at most twice the true Hausdorff distance. From the promises of the approximate nearest neighbor subroutine it thus follows that our algorithm produces a distance which is at most $(2+\epsilon)$ times the true distance, where $\epsilon > 0$ is arbitrarily small.

2.3 Example

In order to illustrate the algorithm outlined above, we apply our algorithm to the melody “Fuggi, fuggi, da questo cielo”, which belongs a melodic family studied by Tagliavini [8]. The result of matching this theme against three other members of this melodic family is shown in Fig. 3.

The example is meant to illustrate meter aspects of the algorithm as well as to show the output when perceptually similar pieces are compared. Note that our algorithm is designed for polyphonic music, and in particular instances where the models are entire works of music that are orders of magnitude larger than the query.

The query consists of the first 19 events of “Fuggi, fuggi, da questo cielo”; the last event has been left out so that the query is no longer than any of the model melodies. Note that the query is in 4/4 meter whereas the three models are in 6/8, 6/8, and 2/4 meter respectively. The query melody matches each of the models when the onsets of the query events are scaled according to 2/4 meter (see section 1.2). Applying this meter has the effect of doubling the number of measures. The numbers above each of the models show the matches (i.e. geometrically nearest neighbors) of each query event in the optimal alignment of the query. Note that more than one query note can be matched to a model note. The translation vector that gives the optimal alignment is also given, where the three coefficients are onset, pitch, and duration, respectively. The first two matches have a shift in the duration dimension of 1, meaning that every query note has been doubled in duration. This shift happens because the algorithm aligns the first note of the query (an eighth note) with notes in the models, and in these two cases the best matches happen to be alignments with quarter notes. For model (b) it turns out that the distance would be smaller if we did not translate the durations. As the algorithm is a $(2+\epsilon)$-approximation, however, we are guaranteed that the distance output is at most about twice the optimal. Hence, match (a) is in any case better than match (b), since the distance of model (b) must be at least ~ 1.4. In each case the note matches that are furthest apart, i.e. those that dictate the Hausdorff distance, are given.

3 MAXIMAL SUBSET MATCHING

This problem is a variation of problem P2 proposed by Ukkonen et al. [10]. In their representation, events are line segments in two-dimensional pitch-time space where event durations are proportional to line segment lengths. The problem is to find all translations of a pattern M such that a subset of the onset events of M match onset events in N. They give an algorithm that solves this problem in $O(nm \log m)$ time and $O(m)$ space based on the idea of sweeplining N. The problem we consider here is the same, with three important modifications: First, we will allow time-scalings (i.e. scaling in the onset dimension) of M in addition to translations. Second, we require that the durations of each event match (in addition to pitch and onset). Third, we are only interested in subsets of size two or greater. The reason for the last modification is that in our representation, any event can be transformed into any other by a translation. Thus, for every query there are nm trivial subset matches of size one that result from simply translating each event in M to each event in N. A solution to the maximal subset matching problem is a

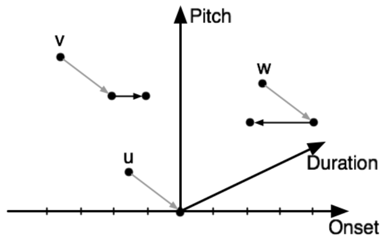

Figure 4. Two steps of the maximal subset matching algorithm. Translation (gray) with respect to point u and subsequent scaling (black) with respect to point v.

list of ordered triplets of the form $\langle v, s, M' \rangle$, where v is a translation vector, s is a scaling factor, $M' \in M$ is the subset of points that match points in N under translation v and scaling s, and $|M| \geq 2$.

3.1 Our Algorithm

Our algorithm employs the idea of geometric hashing, an excellent overview of which is given in [11]. The main idea is to preprocess the models so that the representation is invariant to translation and scaling. In order to keep the preprocessing time and data structure size reasonable, we window the models as described in section 1.4. For each model point u, we consider each other point v that lies within a time window W centered at u. We refer to each such pair of points $\langle u, v \rangle$ as a *basis*. For each basis we translate the window point set W so that u is at the origin. This is illustrated by the gray arrows in Fig. 4. We then scale the onsets of all points in the set by a positive factor such that the difference in the onset of u and v is 1. In other words, we scale all onsets by the inverse of the absolute value of the onset of v. This is shown by the black arrows in Fig. 4. The pitch and duration are left unchanged by this scaling. We now compute a hash value of the points in the point set using a hash function that assigns a unique value to each possible point location. The hash values are then used as indices in a hash table, the entries of which are referred to as $bins$. Each bin consists of a list of *labels*, which contain information about the basis for instances of the aforementioned process where a point in the window set ended up in this specific bin location. For example, to each of the lists associated with the bins for (the translated and scaled) points u, v, and w in Fig. 4, we would add the label identifying the basis $\langle u, v \rangle$. We repeat this process for all pairs of points in all windows.

Processing a query follows a similar procedure. For a pair of query points, we translate and scale as in the preprocessing step. We then consider the lists of labels of the bins into which the query points fall. Each label that occurs in one of these lists identifies a subset match, since for this particular translation and scaling the query and the model have at least one event in common. All that remains is to aggregate the lists, i.e. for each label create a set of all the points whose bins contain that label. Note that every possible label will be present in the bin at the

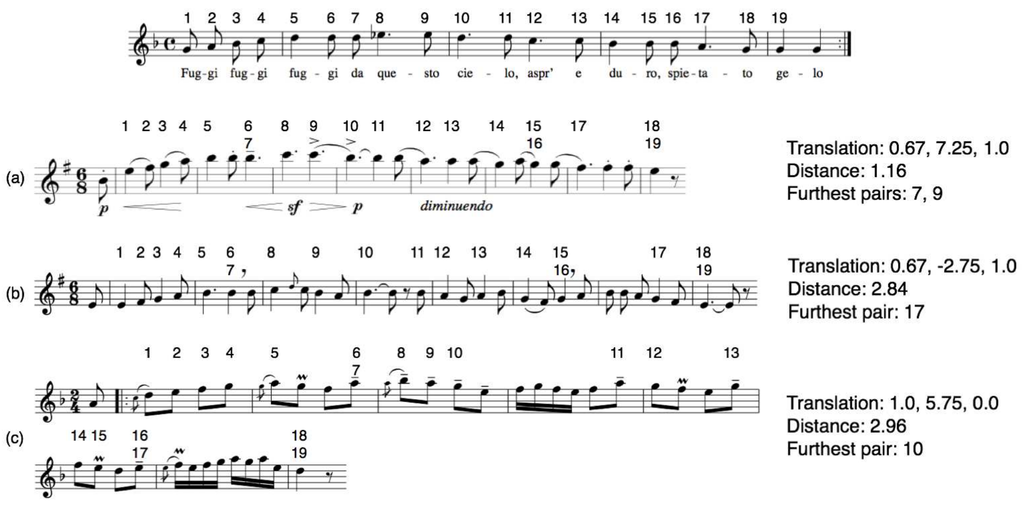
Fug-gi fug-gi fug - gi da que - sto cie - lo, aspr' e du - ro, spie-ta - to ge - lo
(a)
p
sf
p
diminuendo
Translation: 0.67, 7.25, 1.0
Distance: 1.16
Furthest pairs: 7, 9
(b)
Translation: 0.67, -2.75, 1.0
Distance: 2.84
Furthest pair: 17
(c)
Translation: 1.0, 5.75, 0.0
Distance: 2.96
Furthest pair: 10

Figure 3. Example of the minimum Hausdorff distance algorithm. Top: "Fuggi, fuggi da questo cielo", a canzonette from c. 1625 as notated by Gherardo Pedali. (a) "Moldau" theme used by Smetana in his symphonic suite Ma Vlast. (b) A transcription by Coussemaker (a noted musicologist) from c. 1850 of a popular Flemish song, "Ik zag Cecilia komen". (c) A transcription by Tagliavini from a twentieth-century arrangement by Melchiade Benni called the "Ballo di Baraben" or the "Ballo di Mantova".

Algorithm 2 Maximum subset matching algorithm

```
Preprocessing:
1. For each window $W_u$ centered at point $u \in N$
   (a) For each pair $v \in N \cap W_u$, where $u \neq v$
       i. Translate $W$ by $-u$
       ii. Scale $W$ by $\frac{1}{|onset(v)|}$
       iii. For each $w \in W \setminus \{u\}$
           A. Add label $\langle u, v \rangle$ to the bin at location $w$
Query:
1. For each $\langle a, b \rangle$, where $a, b \in M$ and $a \neq b$
   (a) Translate $M$ by $-a$
   (b) Scale $M$ by $\frac{1}{|onset(b)|}$
   (c) For each $c \in M \setminus \{a\}$
       i. For each label $\langle u, v \rangle$ in the bin at location $c$
           A. Add $c$ to the match set for $\langle u, v \rangle$
   (d) Output labels whose match sets are of size $\geq 2$
```

origin, corresponding to subset matches of cardinality 1 as described earlier. To find all maximal subset matches, we repeat this process for all pairs of query points.

3.2 Analysis

We now turn to the time and space complexity of the algorithm. Denote by l the maximum pattern length and by ϵ the smallest unit in the duration dimension. By using a window of length $2l$ centered at the first basis point we are guaranteed to find all subset matches. Furthermore, let w be the maximum number of events that occur in any window. An event forms a basis with at most $(w-1)$ other events. For each basis, at most w new entries are created in the hash table, and so an upper bound on the time and space complexity of the preprocessing step is $O\left(w^2 n\right)$. Although the complexity seems high for big window lengths, for applications where one can define an upper bound on pattern lengths the time and space complexity of the algorithm grows linearly with the database size.

Let us now consider the query time complexity. Each element in the model set N is the first element of a basis at most w times. For each basis, at most one entry is made into the data structure per bin. Hence, each bin's list will have length at most wn. For a query of size m there will be m such lists to consider per basis. By using a hash table, the histogram step (step 1(c) of alg. 2) can be completed in $O\left(wnm\right)$ time. The total running time of the query algorithm is thus $O\left(wnm^3\right)$, since the computation is repeated for $O\left(m^2\right)$ bases. As the running time of our algorithm is in fact proportional to the number of matches, we can restate it as $O\left(m^3\left(k+1\right)\right)$, where k is the number of maximal subset matches. This entails that the running time does not depend on the size of the database. Our algorithm is thus asymptotically optimal up to a m^2 factor, and this at least partly justify the higher space requirements of this relative to previous methods. Finally, we also observe that we were not interested in time-scale invariance, the basis in the preprocessing step and queries would consist of one point. Thus we would get $O\left(wn\right)$ space and $O\left(m\left(k+1\right)\right)$ query time, as we would only need to consider a single basis in the query. This query time is asymptotically optimal assuming match event enumeration is required.

3.3 Example

As an example of the maximal subset matching algorithm given in this section, we consider an excerpt from the appendix to the Goldberg Variations by J. S. Bach (BWV 1087, No. 14) shown in Fig. 5. The second phrase in the soprano (a series of eleven sixteenth notes) is later repeated in the tenor as quarter notes. Taking either of these two note sequences as the query, the algorithm returns a

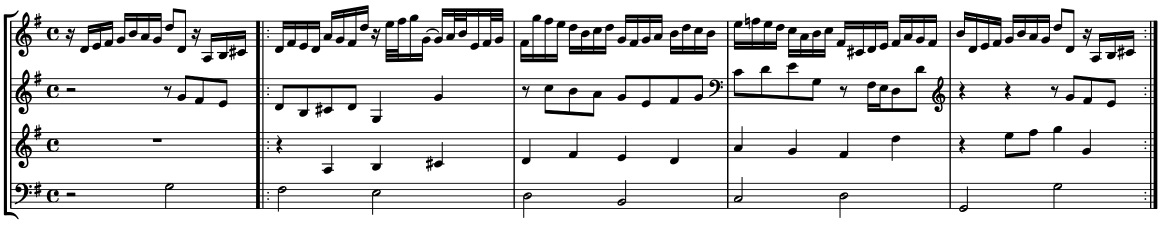

Figure 5. Excerpt from the appendix to the Goldberg Variations by J. S. Bach (BWV 1087, No. 14).

complete match for each basis (see previous section), corresponding to the scaled occurrence. Similarly, the first seven eighth-notes in the alto are repeated in the bass as half notes, with one difference: The 6th note in the sequence is a C# in the eighth-note occurrence and a C in the half-note occurrence. Applying the algorithm to either of these sequences thus gives a match of size six corresponding to the scaled occurrence. This will be the case for any basis that does not have the 6th note as the first note. The relative onsets of the 6th note are the same in the sequences, and so the bases with the 6th note as the second note will also have a match of size six.

4 DISCUSSION AND CONCLUSIONS

We have implemented our two algorithms on the Java platform. The input format for both the database and queries is the Kern data format, and for testing our algorithms we have made extensive use of data from the KernScores website (`http://kern.humdrum.net`) [7], including music by J. S. Bach, Corelli, and Beethoven, as well as ragtime pieces composed by Scott Joplin. The software produces on-the-fly output using the GUIDO NoteServer (`http://www.noteserver.org`).

We continue to explore the contexts in which each of the algorithms are useful. Provisionally, we expect that the Hausdorff distance algorithm should be effective in situations where there is noise present in the encoded music. Future research will focus on ways to improve the space requirements of the time-scale invariant maximal subset matching algorithm. Furthermore, a three-dimensional representation of music is insufficient to capture all perceived features of music. More work is required to determine if the point-set model can be extended to encompass more features, such as harmony and beat information, for example.

5 ACKNOWLEDGEMENTS

We are thankful to Don Anthony for typesetting musical examples for this paper, to Walter B. Hewlett for useful discussions, and to Jürgen Kilian for maintaining the GUIDO NoteServer website.

6 REFERENCES

[1] S. Arya, D. M. Mount, N.S. Netanyahu, R. Silverman, and A. Y. Wu. “An Optimal Algorithm for Approximate Nearest Neighborhood Searching,” *Proc. Symp. Discrete Algorithms*, pp. 573-582, 1994.

[2] R. Clifford, M. Christodoulakis, T. Crawford, D. Meredith, and G. A. Wiggins. “A Fast, Randomised, Maximum Subset Matching Algorithm for Document-Level Music Retrieval,” *Proceedings of the 7th International Conference on Music Information Retrieval* (ISMIR 2006), pp. 153-155, 2006.

[3] P. Di Lorenzo and G. Di Maio. "The Hausdorff Metric in the Melody Space: A New Approach to Melodic Similarity," *Ninth International Conference on Music Perception and Cognition*, Bologna, 2006.

[4] W. B. Hewlett. “A Base-40 Number-Line Representation of Musical Pitch Notation," *Musikometrika 4*, pp. 1-14, 1992.

[5] A. Lubiw and L. Tanur. “Pattern Matching in Polyphonic Music as a Weighted Geometric Translation Problem,” *Proceedings of the 5th International Conference on Music Information Retrieval* (ISMIR 2004), pp. 289-296, 2004.

[6] D. Meredith. “Point-Set Algorithms for Pattern Discovery and Pattern Matching in Music,” In T. Crawford and R. C. Veltkamp (Eds.), *Proceedings of the Dagstuhl Seminar on Content-Based Retrieval*, Dagstuhl, Germany, 2006.

[7] C. Sapp. “Online Database of Scores in the Humdrum File Format,” *Proceedings of the 6th International Conference on Music Information Retrieval* (ISMIR 2005), pp. 664-665, 2005.

[8] L. F. Tagliavini. “’Il ballo di Mantova’, ovvero, ’Fuggi, fuggi, da questo cielo’, ovvero, ’Cecilia’, ovvero,” in Max Lutolf zum 60. Geburtstag: Festschrift (Basel: Wiese), pp. 135-175, 1994.

[9] R. Typke. “Music Retrieval based on Melodic Similarity,” Doctoral thesis. Utrecht University, 2007.

[10] E. Ukkonen, K. Lemstrom, and V. Makinen. “Geometric Algorithms for Transposition Invariant Content-Based Music Retrieval,” *Proceedings of the 4th International Conference on Music Information Retrieval* (ISMIR 2003), pp. 193-199, 2003.

[11] H. J. Wolfson and I. Rigoutsos. “Geometric Hashing: An Overview,” *IEEE Computational Science and Engineering*, 4(4), pp. 10-21, 1997.