A SEARCH METHOD FOR NOTATED POLYPHONIC MUSIC WITH PITCH AND TEMPO FLUCTUATIONS

Rainer Typke, Frans Wiering, Remco C. Veltkamp
Utrecht University
Institute of Information and Computing Sciences

ABSTRACT

We compare two methods of measuring melodic similarity for symbolically represented polyphonic music. Both exploit advantages of transportation distances such as continuity and partial matching in the pitch dimension. By segmenting queries and database documents, one of them also offers partial matching in the time dimension. This method can find short queries in long database documents and is more robust against pitch and tempo fluctuations in the queries or database documents than the method that uses transportation distances alone. We compare the use of transportation distances with and without segmentation for the RISM A/II collection and find that segmentation improves recall and precision. With everything else being equal, the segmented search found 80 out of 114 relevant documents, while the method relying solely on transportation distances found only 60.

1. INTRODUCTION

Our goal is a search engine for notated polyphonic music that would allow musicologists to search large databases of notated music, to trace musical themes as they spread from composer to composer and as they develop over the course of music history. Generally, once the “holy grail” of music information retrieval, automatic polyphonic transcription from audio, is achieved, there will be an increased need for an efficient and effective method for searching notated music. Such a method should be able to deal with variations in tempo and pitch as they occur with human performers. This would enable a search engine to deal with queries entered by humans or to search databases of transcribed performances by humans.

Related Work. Byrd and Crawford [2] provide an overview of the challenges of music information retrieval. They discuss symbolic retrieval and audio retrieval, and they show that polyphonic matching is challenging. Most methods for comparing monophonic sequences of notes, for example string matching, cannot be easily modified so that they become also useful for polyphonic music.

A natural way of searching polyphonic music for the occurrence of a polyphonic pattern is to view the symbolically represented music as sets of notes, characterized by onset time, pitch, and duration, and search for pieces that are supersets of the query. This idea and some variations were explored by Michael Clausen et al. with the PROMS/notify system [3], [4] and Lemström et al. with the C-Brahms system [8], [9], [15]. For example, they looked for supersets of (possibly fuzzy) queries, maximized the overlap of set elements, or searched for occurrences of monophonic patterns represented by a string in polyphonic pieces represented by some parallel strings. Most of these methods put some constraints on the data that can be searched, such as the requirement that the measure structure be known or the note durations and onset times be quantized.

In our previous work [13], we describe how transportation distances such as the Earth Mover’s Distance (EMD) can be used for measuring melodic similarity. Transportation distances have the advantage of not requiring a quantization or the knowledge of the measure structure; however, they work well only for comparing segments of music of corresponding length. Finding an occurrence of a short query in a long piece cannot be done with transportation distances alone.

Our Contribution. Our new, segmented search method, still uses the advantages of transportation distances. It overcomes many restrictions of Clausen’s and Lemström’s methods [14]. By segmenting the music before applying a transportation distance, we are able to match pieces of music of differing length, and the segmentation also makes our method robust against tempo and pitch fluctuations. Our comparison of the new segmenting method with the old transportation distance method shows an improved performance for the RISM A/II collection.

We exploit the following advantages of transportation distances:

- **Continuity:** If differences between queries and database documents are small, transportation distances deliver accordingly small values. When a query is distorted, there is no point at which the distance would suddenly become larger.

- **Support for many distortions:** Many kinds of differences such as grace notes, differences in pitch, note durations, and rhythm are taken into account

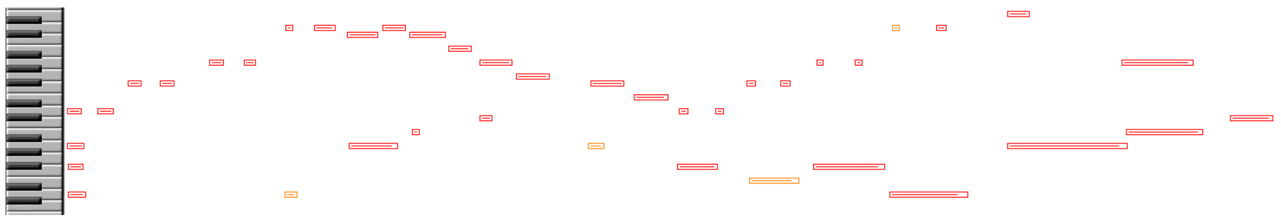

Figure 1. A polyphonic query for Bach's Brandenburg concerto No. 5 (violin part plus the left hand of the cembalo). Although the rhythm fluctuates, a segmented EMD search for this query brought up the monophonic incipit of the Brandenburg concerto shown in Figure 2 as the first match.

by transportation distances without the need for their explicit anticipation.

- **Partial Matching for any combination of polyphonic and monophonic music:** With some transportation distances, examples of which include the EMD, any combination of monophonic and polyphonic music can be matched.
- **Flexibility:** Transportation distances can be fine-tuned to genres and human perception by modifying the weighting scheme and ground distance.

We would like to benefit from these properties of transportation distances and improve on them in robustness and partial matching in the time dimension. In particular, we wish to be able to find occurrences of short queries in large pieces of music and make our method robust against pitch and tempo fluctuations, like those in Figure 1, without requiring explicit tempo tracking. We find that segmenting both queries and database documents into short, overlapping groups does indeed improve the results.

- **Robustness against pitch and tempo fluctuations:** If queries are entered by humans, the pitch and/or tempo frequently fluctuate. While such fluctuations can greatly distort a query, they either do not have a large impact on short segments, or only on a few of them.
- **Partial matching in the time dimension:** Transportation distances do not give meaningful results if the durations of the compared pieces of music vary too much. By matching segments of similar durations, we overcome this problem and are able to find short queries in long pieces.

None of the previously known distance measures for notated music combine all of these properties, and most are discrete in some way. Our contribution is a continuous distance measure that combines the desired properties mentioned above.

2. MEASURING MELODIC SIMILARITY WITH TRANSPORTATION DISTANCES

2.1. Representing notes as weighted point set

To be able to use transportation distances, we represent notated music as weighted point sets. Every note is represented as one point whose coordinates are given by the onset time and pitch. The weight represents the note duration. Depending on the information available, it is possible to make the weights depend on other features, such as the inter-onset intervals, metric stress, melodic contour, position within a measure, piece, or chord, accents, or a combination of these and possibly other features. However, for this paper, we only make the weights depend on note durations.

Figure 2 shows an example of notated music and its associated point set. Rests are represented only implicitly with the surrounding notes' coordinates and weights. Therefore, the point set in Figure 2 only contains one point for each note, but none that would represent the rest. Note that the horizontal distance between the last two notes is twice that between any other pair of notes.

As pitch coordinates, we use Hewlett's base-40 representation [7], a number-line representation that distinguishes between notes with the same pitch but different notation, such as an a$\sharp$ versus a b$\flat$. For the time coordinates, we arbitrarily assign 1 to the duration of a quarter note. Because of the transformations described in Section 2.4 (scaling and translation) which we apply to point sets before calculating a transportation distance, it does not matter which number is associated with the duration of a quarter note, as long as the range of numbers in the pitch dimension is similar enough to that in the time dimension. This is important because it affects the way notes in one point set are matched with notes in the other. If the range of time coordinates is too small in comparison with that of the pitch coordinates, notes tend to be matched with notes that occur much later or earlier, but have similar pitches.

2.2. Transportation Distances

We work with two transportation distances, the Earth Mover's Distance and the Proportional Transportation Distance.

2.2.1. The Earth Mover's Distance

The Earth Mover's Distance (EMD) measures the minimum amount of work needed to transform one weighted point set into another by moving weight. Intuitively speaking, a weighted point can be seen as an amount of earth or mass; alternatively it can be taken as an empty hole with a certain capacity. We can arbitrarily assign the role of the supplier to one set and that of the receiver/demander to the other one, setting, in that way, the direction of weight movement. The EMD then measures the minimum amount of work needed to fill the holes with earth (measured in weight units multiplied with the covered ground distance). See Cohen's Ph.D. thesis [5] for a more detailed description of the EMD.

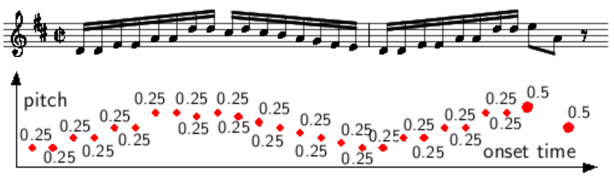

Figure 2. The beginning of the violin 1 part of Bach's Brandenburg concerto No. 5, in common music notation (top) and as a set of weighted points in the two-dimensional space of pitch and onset time (bottom). Weights here represent note durations. In this and all other diagrams, the weight is shown as the surface covered by the disks that represent points. Here we assign a weight of 0.25 to a sixteenth note and 0.5 to an eighth note.

Definition Let $A = \{a_1, a_2, .., a_m\}$ be a weighted point set such that $a_i = \{(x_i, w_i)\}, i = 1, .., m$, where $x_i \in \mathbb{R}^k$, with $w_i \in \mathbb{R}^+ \cup \{0\}$ being its corresponding weight. Let W=$\sum_{j=1}^{m} w_i$ be the total weight of set A.

The EMD can be formulated as a linear programming problem. Given two weighted point sets A, B and a ground distance d, we denote as f_{ij} the elementary flow of weight from x_i to y_j over the distance d_{ij}. If W, U are the total weights of A, B respectively, the set of all possible flows $\mathcal{F} = [f_{ij}]$ is defined by the following constraints:

1. $f_{ij} \geq 0, i = 1, ..., m, j = 1, ..., n$
2. $\sum_{j=1}^{n} f_{ij} \leq w_i, i = 1, ..., m$
3. $\sum_{i=1}^{m} f_{ij} \leq u_j, j = 1, ..., n$
4. $\sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij} = min(W, U)$

These constraints say that each particular flow is non-negative, no point from the "supplier" set emits more weight than it has, and no point from the "receiver" gets more weight than it needs. Finally, the total transported weight is the minimum of the total weights of the two sets.

The flow of weight f_{ij} over a distance d_{ij} is penalized by its product with this distance. The sum of all these individual products is the total cost for transforming A into B. The EMD(A, B) is defined as the minimum total cost over $\mathcal{F}$, normalized by the weight of the lighter set; a unit of cost or work corresponds to transporting one unit of weight over one unit of ground distance. That is:

$$\mathrm{EMD}(A, B) = \frac{\min_{F \in \mathcal{F}} \sum_{i=1}^{m} \sum_{j=1}^{n} f_{ij} d_{ij}}{min(W, U)}$$

See Figure 3 for an illustration of an optimal flow and the matching of notes.

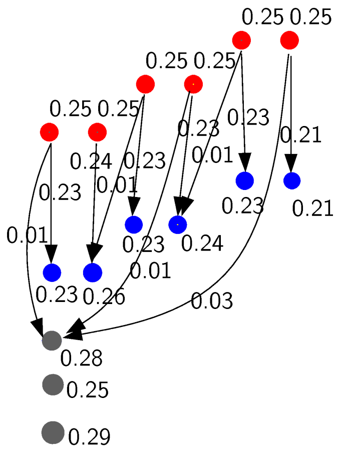

Figure 3. An example of a flow. This illustrates the distance calculation between the first segments of the polyphonic query shown in Figure 4 and the monophonic database entry shown in Figure 2 using the EMD. The point set representing the monophonic segment is shifted upwards in this picture to make the flow more visible. Its six points are actually aligned with the six corresponding points of the polyphonic segment so that most of the flow components (and all flow components involving large amounts of weight) have a ground distance close to zero.

Properties and Computation

The most important properties of the EMD can be summarized as follows:

1. The EMD is a metric if the ground distance d is a metric and if the EMD is applied on the space of equal total weight sets.

2. It is continuous. In other words, arbitrarily small changes in position and/or weight of existing points cause only arbitrarily small changes in its value. Moreover, the addition of a point with an arbitrarily small weight leads to an arbitrarily small change in the EMD's value.

3. It does not obey the positivity property if the sums of the weights of the two sets are not equal. In that case, some of the weight of the heavier distribution remains unmatched. Therefore, the EMD allows for partial matching. As a result, there are cases where it does not distinguish between two non-identical sets. This can be useful, for example when matching a monophonic melody to a piece that contains the same melody, but with an accompaniment.

4. In the case of unequal total weights, the triangle inequality does not hold.

The triangle inequality is relevant for the indexing method described in Section 4.2.

The EMD can be computed efficiently by solving the corresponding linear programming problem, for example by using a streamlined version of the simplex algorithm for the transportation problem (Hillier and Lieberman 1990). We used Rubner's EMD function [12], which implements Hillier's and Lieberman's algorithm. It is possible that the simplex algorithm performs an exponential number of steps. One could use polynomial algorithms like an interior point algorithm, but in practice that would outperform the simplex algorithm only for very large problem sizes. Since the transportation problem is a special case of the minimum cost flow problem in networks, a polynomial time algorithm for that could be used as well.

2.2.2. The Proportional Transportation Distance

Giannopoulos and Veltkamp [6] proposed a modification of the EMD in order to get a similarity measure based on weight transportation such that the surplus of weight between two point sets is taken into account and the triangle inequality still holds, which is useful for the indexing method described in Section 4.2. They call this modified EMD the "Proportional Transportation Distance" (PTD) because any surplus or shortage of weight is removed in a way that the proportions are preserved before the EMD is calculated. The PTD is calculated by first dividing, for both point sets, every point's weight by its point set's total weight, and then calculating the EMD for the resulting point sets.

The PTD is a pseudo-metric. In particular, it obeys the triangle inequality. It still does not have the positivity property since the distance between positionally coinciding sets with the same percentages of weights at the same positions is zero. However, this is the only case in which the distance between two non-identical point sets is zero. The PTD will distinguish between two sets which differ in only one point. It has all other properties that the EMD has for equal total weight sets.

2.3. Ground Distance

For all results in this paper, we use the Euclidean distance as ground distance. Thus, the distance between two notes with the coordinates (t_i, p_i) and (t_j, p_j) is

$$d_{ij} = \sqrt{(t_i - t_j)^2 + (p_i - p_j)^2}.$$

A variation possibly interesting for polyphonic music would be to make the distance in the pitch dimension depend on harmony instead of just calculating the difference between pitches.

2.4. Transformations used for achieving transposition and tempo invariance

2.4.1. Transposition invariance

In order to achieve transposition invariance, we calculate the minimum distance for a range of transpositions. Because we store pitch as discrete values, there are only finitely many transpositions with a constant upper bound that we need to try.

This translation in the pitch dimension such that the distance is minimized does not invalidate the triangle inequality.

2.4.2. Tempo invariance

Our segmenting method (see Section 3) aims at cutting the music into segments of corresponding duration. Therefore, we can translate and scale all point sets to a constant range of time coordinates before comparing them by using a transportation distance. After segmenting music, the time coordinate of the last note within a segment depends on the tempo. By scaling segments so that the maximum time coordinate is always the same constant number, we eliminate this dependence on tempo.

Note that this scaling does not invalidate the triangle inequality.

3. SEGMENTING

The aims of segmenting are to improve partial matching in the time dimension, to increase robustness against pitch and/or tempo fluctuations, and to ensure that the transportation distances are applied to comparable groups of notes. With "comparable", we mean that the groups of notes should contain similar numbers of consecutive notes, and not too many.

We are not necessarily concerned with segments that make musical sense. For our experiments, we worked with segment lengths in the range from 6 to 9 consecutive notes. Segments of this length are usually distinctive enough so that we did not get too many matches from pieces that are not really similar, but still short enough for getting the desired robustness against tempo and pitch fluctuations.

Our segmenting algorithm must fulfill certain conditions for our method to work properly. We would like to be able to process manually recorded MIDI queries with free, possibly fluctuating tempo and unknown measure structure. Also, we want the segmenting results to be largely independent of how many voices are present at the same time. Therefore, we cannot just take a certain number of notes and declare them a segment. Rather, we must look at a certain number of consecutive notes.

We work with overlapping segments to reduce the influence of the position of a query within a piece, and we create multiple segments with different lengths, but the same starting point, in order to be able to match single long notes with corresponding multiple shorter notes.

For our experiments, we segmented queries and database documents as follows:

First, we set a pointer to the onset time of the first note that is to become part of the next segment. This is the beginning time of a new segment.

Then, we move the pointer to the next end of any note whose onset time lies within the current new segment, then to the next beginning of a note. We do this until we have the desired number of consecutive notes in the segment.

We include all notes with an onset time within the closed interval from the beginning of the segment to the current pointer position in the next segment.

For example, segment number 1 in Figure 4 is found like this: First, we move a pointer to the onset time of note number 1, the first note we want to include in the segment. Then, we move the pointer to the end of the longest note in the beginning chord (the lowest note), because that is the next ending of any note whose onset time lies in the current segment. Now we move the pointer to the beginning of note 2 since this is the next onset time after the pointer. This way, we have included the whole chord at the beginning in the new segment, but count it as only one out of six steps. The next five steps work the same way (go to the next end of any note after the pointer, then to the next onset time). As a result, we identify the first segment as shown in Figure 4 with its 9 notes as a segment with 6 consecutive notes.

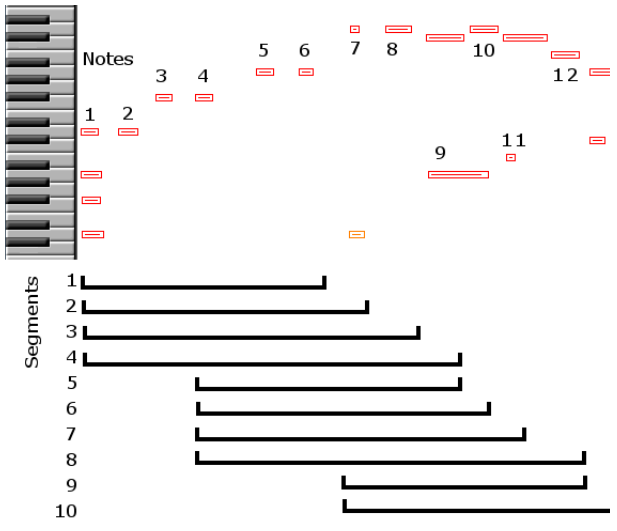

Figure 4. The first ten segments of the polyphonic query shown in Figure 1.

As illustrated in Figure 4, we generate overlapping segments that are three notes apart, and at every starting point, we create segments of length 6, 7, 8, and 9. All of those are scaled to the same distance between the onset times of the first and last note, as described in Section 2.4.2, before transportation distances are applied. Because every note is the last note of some segment, there are no leftover notes at the end that would not be part of any segment.

In order to correctly recognize consecutive legato notes in MIDI queries as consecutive (for getting a legato effect, the player releases piano keys only after the following note has started), it was sufficient to treat all notes as if they were only 80% of their length for the purpose of segmenting.

By segmenting queries and database documents, we increase the number of comparisons of point sets that are necessary for answering a query. On the other hand, the individual comparisons become simpler since smaller point sets need to be compared, and the size of point sets is bounded. The number of comparisons grows linearly. If we segment as illustrated in Figure 4, the number of segments is always less than or equal to the number of consecutive notes times 4/3 (every 3 notes, there are up to 4 beginnings of a segment). The actual search time grows only logarithmically if one uses the vantage indexing method described in Section 4.2.

4. SEARCHING

Our database contains pre-calculated segments of all pieces. To answer a query, we segment it, then for each query segment, we search the set of all segments for the most similar ones, and finally combine the results of the segment searches.

Each segment search yields a list of pieces that contain at least one matching segment. The overall result should be a list of pieces with many closely matching segments. For this, we need to compute a distance for each piece that occurs in at least one segment search result. To do this, we first determine the maximum distance M that occurs in any segment search result. For each segment search result in which a piece P occurs, we add the distance of the highest ranked segment of P to the overall score for P. For each segment search result in which P does not occur, we do not know the distance of the corresponding segments because it was high enough for the segment of P to not occur in this result list. Therefore, it is at least the maximum distance in this result list, but probably clearly higher. We get good results if we add twice M to the overall score for P in such cases. For each segment search result without a segment from P that is both preceded and followed by segment search results with segments from P, we add 4 times M to the overall score for P. If the query is really a subset of the database document P, there should not be a section within P that does not match, therefore there should be a higher penalty for missing segments within the query than for missing segments at the beginning or end of a query.

The resulting overall score is a distance measure. It is zero if for every segment of the query a matching segment with distance zero was found in the same database document P. The distance measure grows with the individual distances of segments and with the number of segments for which no matches were found. While the underlying transportation distance is symmetric, the resulting distance measure is not. The triangle inequality does not hold, and it is not always positive for unequal pieces of music. Therefore, it is not a metric.

4.1. Adjusting the search radius for different segments

For each segment, we perform an n nearest-neighbours search up to a given maximum search radius m.

When using the vantage indexing method (see Section 4.2), we cannot directly search for n nearest neighbours, but need to work with a search radius. This radius has to be different for different segments if we want to retrieve similar numbers of neighbours. For typical musical patterns, like many repeated notes within one segment, there tend to be many more neighbours within a small radius around the segment than for very distinctive patterns of notes.

We do not want to impose the task of selecting an appropriate search radius for each segment on the user, who should not need to be aware of the segmenting in the first place. Our search engine, therefore, adjusts the search radius during the search as follows: The search starts with a given low initial value which is unlikely to be too large for any segment. If during the search we find more than n neighbours with distance zero, the segment is not distinctive enough to be considered at all, and this segment search can be stopped immediately. There are segments that do not contain enough characteristic musical material for being helpful. If at the end of the search, not enough matches (less than the n nearest neighbours we are looking for) were found within the search radius, we increase the radius and search again. In this case, it is sufficient to search the area outside the original search radius, but within the new, enlarged one. We do this only while the search radius is less than the given maximum search radius m.

4.2. Nearest neighbour searches with the vantage indexing method

Since it would be prohibitively time consuming to compute a transportation distance to a query point set for all point sets in the database, we use the vantage indexing method described by Vleugels and Veltkamp [16]. If the triangle inequality holds for the transportation distance, this method allows us to rule out almost all database objects without having to calculate the time consuming transportation distance. We can rule out all objects whose distance to any of the vantage objects differs by more than our search radius from the distance of the query object to the same vantage object.

Before searching, we pick some vantage objects, for example v randomly selected point sets that are already in the database. Then, for each point set in the database, we calculate the transportation distance to each of the vantage objects.

For the search, we first determine the distance of the query object to each vantage object. If the query object is in the database, these distances are already calculated. Otherwise, we calculate them now. Then, we retrieve all database objects whose distance to the query object, measured with the L_∞ norm in the v-dimensional space of distances to vantage objects, is less than or equal to the search radius. This can be done with an approximate nearest-neighbour search with $O(k \log n)$ L_∞ norm calculations [1] plus k expensive transportation distance calculations, where k is the number of reported point sets. If one prefers an exact nearest neighbour search, one can query a v-dimensional kd-tree using $O(n^{1-\frac{1}{v}} + k)$ L_∞ norm calculations, which is more expensive for sensible numbers of vantage objects (a larger v will allow us to rule out more database objects).

Only for the objects that could not be ruled out based on the triangle inequality do we have to compute the transportation distance. With our constant segment length, the complexity is $O(k)$, where k is the number of reported objects.

In practice, searching the RISM A/II collection usually takes a few minutes on a PC with 1 GB of main memory. The most important factors determining how long the search really takes are the number of segments in the query and whether previous queries were similar. In that case, the contents of the cache containing part of the database indices are useful, and the search takes only a few seconds. The large impact of caching effects makes it seem that with enough main memory for holding the database indices, response times of a few seconds would be possible for all queries. We use two tables, one containing the distances to vantage objects for every segment and one containing the weighted point sets for all segments. Together, these tables including the MySQL indices take up about 1.7 GB of space, so with about 2 GB of main memory, there would be a good chance of attaining search times of a few seconds.

By using the vantage indexing method, we do not change the search result, we just calculate it faster. When working with a transportation distance for which the triangle inequality holds, e. g. the PTD, calculating the transportation distances only for the candidates with similar distances to vantage objects yields the same result as an exhaustive database search. One might argue that human similarity measures are not even symmetric and usually also do not obey the triangle inequality, therefore using a method that relies on the triangle inequality for indexing seems suspect. However, our distance measure as described in this section does not obey the triangle inequality although it is based on the PTD. It performs well in experiments where human experts judge its results [14]. We do not reduce the quality of our results by exploiting the fact that for the underlying transportation distance, the PTD, the triangle inequality holds.

For the EMD, not even a weak triangle inequality such as EMD(A,B) $\leq k$ (EMD(A,C) + EMD(C,B)) holds (with $k \geq 0$). Counterexamples exist where EMD(A,B)>0, EMD(A,C)=0, and EMD(B,C)=0; see Figure 5.

However, our experiments show that with the RISM A/II[11] collection, when using the vantage indexing method with the EMD, usually all matches within a third of the search radius are retrieved. Hence if the search radius is increased accordingly, the vantage indexing method can still be used for polyphonic searches with the EMD, albeit without a guarantee for the completeness of the matches.

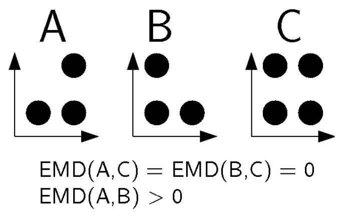

Figure 5. For the EMD, not even the weak triangle inequality holds. In this example, EMD(A,B) > k (EMD(A,C) + EMD(C,B)) for all $k \geq 0$.

5. COMPARISON

To see how well the segmented search method works, we manually entered rhythmically distorted queries using a MIDI keyboard. For example, a segmented EMD search finds the monophonic incipit shown in Figure 2 as the first match for the query shown in Figure 1 when we search the RISM A/II collection with about half a million musical incipits (incipits are the beginnings of pieces, typically about 20 notes long).

We also compared the original method described by Typke et al. in [13] to our improved method using the RISM A/II collection. To avoid any bias, we randomly selected 16 incipits out of the database as queries. For each of them, we used both methods for retrieving the 25 most similar incipits. As transportation distance, we used the PTD since the collection is mostly monophonic, and the EMD's partial matching in the pitch dimension is not needed. This gave us a total of 800 matches (with some overlap), for each of which we decided whether it was melodically similar to the query and therefore relevant.

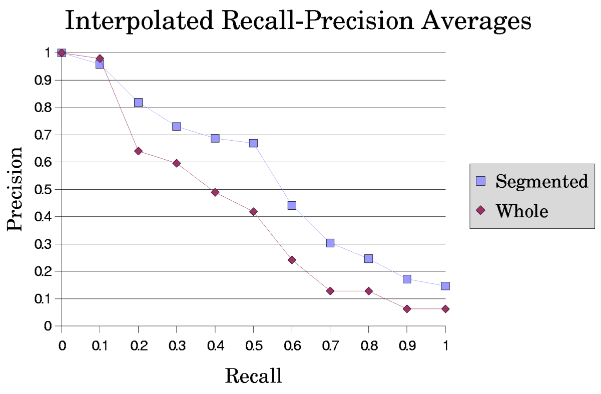

Figure 6. Interpolated recall-precision averages. Since every query was contained in the database, and both methods correctly recognize identity, the difference is small for very similar documents. The advantages of segmenting become apparent for documents that are less similar, but still similar enough to be considered relevant.

We decided about the relevance in a way that minimized the influence of any bias towards one method. For each query, we created one combined result list that contained all documents which were returned by any of the two search methods. Those that were returned by both were listed only once. These lists were not sorted by method or by the ranks of documents, but by the library holding the source manuscripts. Therefore, for every relevance decision it was very hard to tell which method had retrieved the document in question. The relevance decisions were taken by two people, each of whom covered half the queries. As Müllensiefen et al. point out [10], "subjects with stable similarity judgements seem to have the same notion of melodic similarity". Thus, a high number of human experts making the similarity judgements is not always necessary.

Since for our comparison, we searched the RISM A/II collection for pieces similar to queries taken from the same collection, there were no pitch or rhythm distortions. Therefore, this comparison does not show all strengths of the segmented search method. But the segmented search still performs better than the non-segmented one.

See Figure 6 for a recall-precision graph. For the purpose of this graph, we assumed that all relevant documents were retrieved by one of the two methods. Among all documents that any method retrieved, 114 were judged to be relevant. The comparison of whole incipits produced 60 relevant documents, while the segmented search found 80.

The increased retrieval performance of the segmented search is largely due to the improved partial matching in the time dimension. Figure 7 shows an example where this matters.

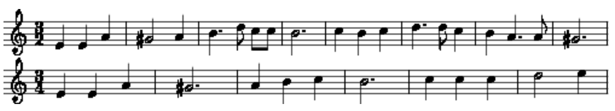

Figure 7. Query (top): John Dowland, "If fluds of tears could clense my follies past". A segmented search finds the match (bottom) by Josephus Fodor, a violin duet. This match is not found by the search method relying purely on the transportation distance because although the melodies are similar, the durations of the incipits do not correspond.

6. CONCLUSIONS

Our comparison of segmented and non-segmented searches using the RISM A/II collection showed that the improved partial matching in the time dimension, which is achieved by segmenting, improves precision and recall. We have also used segmented searches with transportation distances for matching polyphonic queries with fluctuating tempo with similar monophonic incipits from the RISM A/II collection with constant tempo as illustrated with Figures 1 and 2. This method supports any combination of monophonic and polyphonic notated music with or without pitch and tempo fluctuations.

The same indexing methods can be used for searches with or without segmenting. For segmented searches, indexing is very important since without it, the number of transportation distance calculations gets unbearably high. For example, with segment lengths of 6, 7, 8 and 9 and a distance of 3 notes between beginnings of segments, we need about 4.5 million segments for covering approx. 0.5 million incipits in the RISM A/II collection. A typical query is cut into 20 segments. Without indexing, this would mean almost 100 million transportation distance calculations just for answering one query.

Possible improvements

There are some ways in which our method could still be improved:

In order to improve the ranking of the retrieved candidates for matches, we could add a second ranking step after deciding which documents should be listed at all. In this second step, we could also take those segments into account that did not lead to the candidates' inclusion because they were either not similar enough to any segment in the query or not distinctive enough to be considered. Also, we could work with a finer overlap (start a new segment at every note instead of just every three notes) and more segment lengths for the final step, where the added effort would not be very noticeable.

Figure 3 shows that transportation distances sometimes match notes with multiple other notes, some of which can be quite far away. It is conceivable that a transportation distance that would only take the flow component to the closest point in the receiving point set into account would perform better. Such a transportation distance, however, would introduce discontinuities whenever points are added or removed. It would still be continuous if only weights and positions of points are modified.

7. REFERENCES

[1] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Wu (1994). An optimum algorithm for approximate nearest neighbor searching. *Proceedings of the Fifth ACM-SIAM Symposium on Discrete Algorithms*, pp. 573–582.

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

[3] M. Clausen, R. Engelbrecht, D. Meyer, and J. Schmitz (2000). PROMS: A Web-based Tool for Searching in Polyphonic Music. *Proceedings of the 1st International Symposium on Music Information Retrieval (ISMIR 2000)*

[4] M. Clausen and F. Kurth (2003). A Unified Approach to Content-Based and Fault-Tolerant Music Recognition. *IEEE Transactions on Multimedia*. To appear.

[5] S. Cohen (1999). *Finding Color and Shape Patterns in Images*. Ph.D. thesis, Stanford University, Department of Computer Science.

[6] P. Giannopoulos and R. C. Veltkamp (2002). A Pseudo-Metric for Weighted Point Sets. In Heyden, A., Sparr, G., Nielsen, M. & Johansen, P. (Ed.), *Proceedings of the 7th European Conference on Computer Vision (ECCV)*, pp. 715–730. Copenhagen, Denmark: Springer-Verlag.

[7] W. B. Hewlett (1992). A Base-40 Number-line Representation of Musical Pitch Notation. *Musikometrika*, 4, 1–14. Retrieved April 1, 2003, from http://www.ccarh.org/publications/reprints/base40/

[8] K. Lemström, V. Mäkinen, A. Pienimäki, M. Trkia, and E. Ukkonen: The C-BRAHMS Project. *Proceedings of the 4th Internationoal Conference on Music Information Retrieval (ISMIR 2003)*, pp. 237–238, Johns Hopkins University, Baltimore (MD), USA, 2003.

[9] K. Lemström and J. Tarhio: Transposition Invariant Pattern Matching for Multi-Track Strings. *Nordic Journal of Computing*, 10 (3) 2003 (to appear). Retrieved from http://www.cs.helsinki.fi/group/cbrahms/publications/lemstrom_tarhio.pdf

[10] D. Müllensiefen and K. Frieler: Measuring melodic similarity: Human vs. algorithmic Judgments. *R. Parncutt, A. Kessler & F. Zimmer (Eds.) Proceedings of the Conference on Interdisciplinary Musicology (CIM04) Graz/Austria, 15-18 April, 2004*

[11] *Répertoire International des Sources Musicales (RISM). Serie A/II, manuscrits musicaux après 1600.* (2002) K. G. Saur Verlag, München, Germany. http://rism.stub.uni-frankfurt.de

[12] Rubner, Y. *Source code for the Earth Mover's Distance software*. (1998). Retrieved April 1, 2003, from http://robotics.stanford.edu/~rubner/emd/default.htm

[13] R. Typke, P. Giannopoulos, R. C. Veltkamp, F. Wiering and R. van Oostrum (2003). Using Transportation Distances for Measuring Melodic Similarity. *ISMIR 2003: Proceedings of the Fourth International Conference on Music Information Retrieval*, pp. 107–114.

[14] R. Typke, R. C. Veltkamp, F. Wiering (2004). Searching notated polyphonic music using transportation distances. *Proceedings of the ACM Multimedia Conference 2004, New York.*

[15] E. Ukkonen, K. Lemström, and V. Mäkinen (2003). Geometric Algorithms for Transposition Invariant Content-Based Music Retrieval. *ISMIR 2003: Proceedings of the Fourth International Conference on Music Information Retrieval*, pp. 193–199.

[16] Vleugels, J. and Veltkamp, R. C. (2002) Efficient Image Retrieval through Vantage Objects. *Pattern Recognition*, 35(1) pp. 69–80.