Searching Notated Polyphonic Music Using Transportation Distances

Rainer Typke
Utrecht University
Padualaan 14
3584 CH Utrecht,
The Netherlands
rainer.typke@cs.uu.nl

Remco C. Veltkamp
Utrecht University
Padualaan 14
3584 CH Utrecht,
The Netherlands
remco.veltkamp@cs.uu.nl

Frans Wiering
Utrecht University
Padualaan 14
3584 CH Utrecht,
The Netherlands
frans.wiering@cs.uu.nl

ABSTRACT

We present a method for searching databases of symbolically represented polyphonic music that exploits advantages of transportation distances such as continuity and partial matching in the pitch dimension. By segmenting queries and database documents, we also gain partial matching in the time dimension. Thus, we can find short queries in long database documents, and have a method more robust against pitch and tempo fluctuations in the queries or database documents than we would with transportation distances alone. We compare our method with three algorithms from the C-Brahms project by Lemström et al. and with PROMS by Clausen et al. and find that our method is more generally usable, retrieves a higher number of relevant documents than all three compared algorithms, and that it is faster than C-Brahms. This is the first comparative study of these algorithms involving a large database with about half a million of documents.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing—*Symbolic music representation*; J.5 [**Arts and Humanities**]: Performing arts (music)

General Terms

Algorithms, Measurement, Performance

Keywords

melodic similarity, Proportional Transportation Distance, Earth Mover's Distance, polyphonic matching

1. INTRODUCTION

Currently, those music information retrieval systems that get the most attention from the general public are audio fingerprinting systems like Shazam (www.shazam.com). Those systems can reliably and quickly identify a recording, but not a composition. Because our goals differ from those behind Shazam, we take an entirely different approach. We are concerned with notated music instead of audio because our goal is to create a tool 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. A search engine for notated music can also be useful for attributing composers to anonymous pieces by comparing the music from one library to that in many other libraries in the hope of finding another occurrence of the same piece with more complete metadata. Finally, 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.

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]. 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 Section 5, we compare our method with three of their algorithms.

Our Contribution. We overcome many restrictions of Clausen's and Lemström's methods by representing music with weighted point sets and comparing these using transportation distances such as the Earth Mover's Distance

MM'04, October 10-16, 2004, New York, New York, USA.

(EMD). In particular, 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 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 all 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.

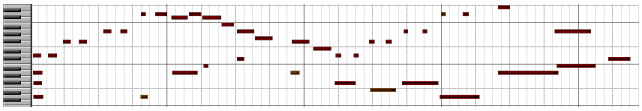


Figure 1: A polyphonic query (Violin 1 plus cembalo, left hand) for Bach's Brandenburg concerto No. 5, entered manually with a MIDI keyboard. Note the strong inaccuracies in rhythm and note durations.

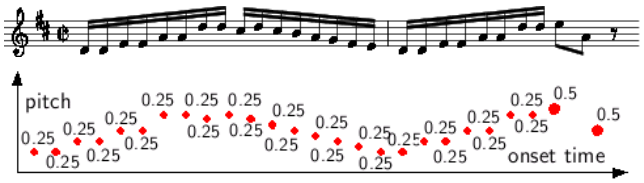


Figure 2: The violin part of the beginning of Bach's Brandenburg concerto No. 5, in common music notation (top) and as a set of weighted points in the two-dimensional space of pitch and onset time (bottom). Weights here represent note durations. In this and all other diagrams, the weight is shown as 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.

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♯ versus a b♭. 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 speak-

ing, 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.

2.2.1.1 *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.

2.2.1.2 *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 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.

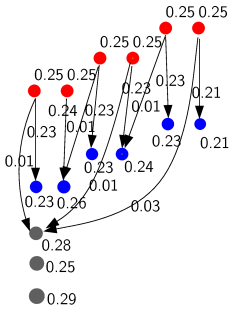


Figure 3: An example of a flow. This illustrates the distance calculation between the first segments of the polyphonic query shown in Figure 4 and the monophonic database entry shown in Figure 2 using the EMD. The point set representing the monophonic segment is shifted upwards in this picture to make 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.

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 [11], 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 prop-

erty 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 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.

A comparison of query results for the RISM A/II collection with and without segmenting [13] showed greatly improved precision and recall with segmenting.

We are not necessarily concerned with segments that make musical sense. For our experiments, we worked with a length of 6 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 a length of 6 also means that segments are 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: The segmenting algorithm should not rely on a known measure structure. 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. Finally, we work with overlapping segments to make the position of a query within a piece less relevant.

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 six times (because we want six 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.

As illustrated in Figure 4, we generate overlapping segments that are three notes apart.

We ignore any notes that may be left over at the end of the segmented music. This would happen, for example, for a query that contains 7 consecutive notes. In this case, we ignore one note.

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.

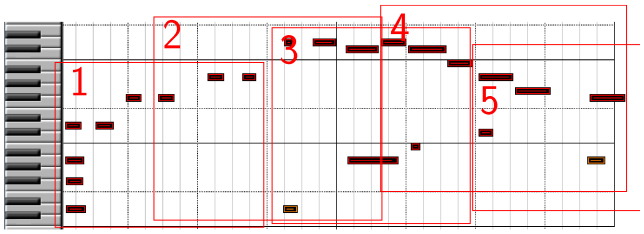


Figure 4: The first five segments of the polyphonic query shown in Figure 1.

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. Although the transportation distances are symmetric, the combined distance measure is not. Also, 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 [15]. 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.

When working with a transportation distance for which the triangle inequality holds, e. g. the PTD, using the vantage indexing method yields the same result as an exhaustive database search.

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.

However, our experiments show that with the RISM A/II 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.

5. COMPARISON

Lemström et al. [8], [9], [14] propose a number of algorithms for searching notated music. Their search engine prototype, which uses these algorithms, is called "C-Brahms". We compare our search results to those produced by three of their algorithms that support polyphonic queries:

- **P1:** Find translations of the query pattern such that *all* onset times and pitches of notes in the query match with some onset times and pitches of notes in the database documents. Time complexity: $O(mn)$ (where m is the number of notes in the query and n the number of notes in the database document).
- **P2:** Find translations of the query pattern such that *some* onset times and pitches of the query match with some onset times and pitches of database documents.

Complexity: $O(mn \log m)$.

- **P3:** Find translations of the query pattern that give longest common shared time (i. e., maximize the times at which query notes sound at the same time and with the same pitch as notes from the database documents). This algorithm does not take into consideration whether onset times match. Complexity: $O(mn \log(mn))$ or, for a constant, finite number of possible pitch levels: $O(mn \log m)$.

PROMS [3] is an efficient system for finding supersets of queries, where queries are sets of notes described by onset time and pitch. Therefore, it essentially performs algorithm P1 in an efficient way, so we do not need a separate comparison with the results that PROMS would deliver for the same queries and database.

Lemström et al. do not propose any indexing method for their algorithms, so for any search, the query has to be compared with each database document. PROMS uses an inverted file index that requires a quantization of note onset times and the knowledge of the measure structure of queries and database documents. Our method can be used with the vantage indexing method described in Section 4.2, without requiring the measure structure to be known or the onset times to be quantized beyond what is needed for representing real numbers using a computer's finite memory.

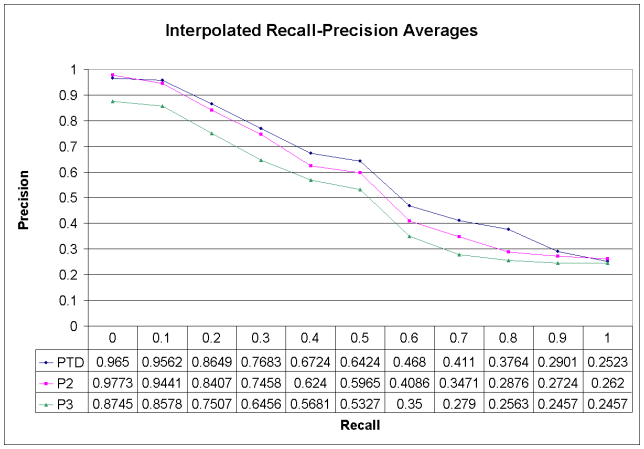
5.1 Experiment

We randomly selected 44 incipits from our database of about 476,000 incipits and used them as queries. We cut off every result list at position 25 and, if the 26th document was ranked the same as the 25th, removed all documents of that rank (otherwise we would have made a very arbitrary selection of matches since there were cases with hundreds of documents with the same rank as the 25th). This gave us a total of 3563 matches (4 methods, 44 queries, and up to 25 matches each), for each of which we decided whether it was melodically similar and therefore relevant.

Using van Zwol's testbed [16], 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 containing all documents that were returned by any search method. Those that were returned by more than one method 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. We had four people make the relevance decisions, two of whom had not worked on our project before. Each person covered half of the 44 queries, and every match was judged by two people.

Generally, P1, P2, and P3 do not properly match patterns if for some notes, the pitch is slightly distorted, and they also do not work if the tempo is not constant. 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. But even if we do not take this advantage of using transportation distances into account, our method still compares favourably with P1, P2 and P3 (and therefore also PROMS).

See Figure 5 for a recall-precision graph and numbers of retrieved relevant documents. For the purpose of this graph, we assumed that all relevant documents were retrieved by some method. We do not have a ground truth for our 44 queries. There are probably some relevant documents that were not retrieved by any method. Therefore, the actual precision is probably lower for all methods, but a comparison is still possible. Out of all 275 relevant documents found by all methods combined, our method retrieved 179, P1 only 76, and P2, the second-best method, 166. Thus, our method found 8 % more relevant documents than P2 and 136 % more than P1. For every method, we looked at queries where it performed badly, and describe the individual weaknesses in the following subsections.



	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
PTD	0.965	0.9562	0.8649	0.7683	0.6724	0.6424	0.468	0.411	0.3764	0.2901	0.2523
P2	0.9773	0.9441	0.8407	0.7458	0.624	0.5965	0.4086	0.3471	0.2876	0.2724	0.262
P3	0.8745	0.8578	0.7507	0.6456	0.5681	0.5327	0.35	0.279	0.2563	0.2457	0.2457

Figure 5: P1 is not included in the graph because it does not rank matches. Among all documents that any method retrieved, 275 were judged to be relevant. Out of these, P1 found 76, P2 found 166, P3 found 125, and PTD found 179.

5.2 P1 (all onset times match)

Algorithm P1 finds only supersets of the query. For most queries, only database documents containing exactly the same notes as the query turned out to be supersets. Whenever this was not the case, the retrieved document tended to be a false positive like the example shown in Figure 6. For this query, P1 only found two occurrences of the piece containing the query and two occurrences of the same false positive. The problem here is that P1 completely ignores any non-matched notes which occur between the query notes.

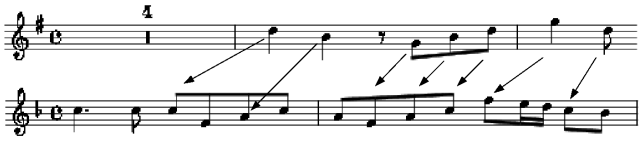


Figure 6: Top: Joseph Eybler: "Dies sanctificatus" (Query). Bottom: False positive, Pietro Guglielmi: "Domine ad adjuvandum". Arrows connect the notes that P1 matched with one another.

5.3 P2 (some onset times match)

We ranked the P2 matches by the number of matching onset times. The highest-ranked matches are always identical to the P1 matches. Therefore, the problem illustrated with Figure 6 also applies to P2. For this query, Joseph Eybler's Gradual "Dies sanctificatus", P2 returns more than

500 matches, but fails to find two occurrences of Gregor Rösler's sacred song "Dies sanctificatus", which is melodically similar and has the same title and therefore should be ranked highly (see Figure 9).

For an example of a match that was found by P2, but missed by P1, see Figure 7.

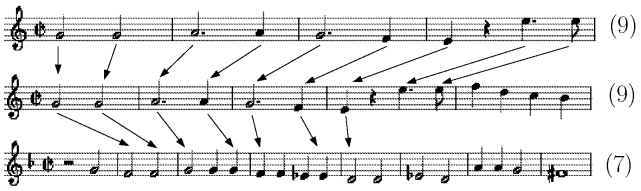


Figure 7: The top three matches for a Kyrie by Jan Vitásek, as returned by P2. The query is shown at the top, and the number of matched onset times is written in parentheses. There are many more matches with 7 matching onsets. The partial match shown here (a song titled "En jungfru stålter haver lag") is clearly not the same piece, but it is similar enough to be relevant.

5.4 P3 (maximize shared common time)

As P1 and P2, P3 finds all matches that are a superset of the query. False positives are caused by a problem somwhat similar to P1's: P3 does not distinguish between short, repeated notes versus longer notes of the same pitch and therefore frequently ignores additional note onsets between those in the query. An illustration of this problem can be found in Figure 8, which shows a query for which P3 finds 113 quite different matches, all with the same maximum shared common time.

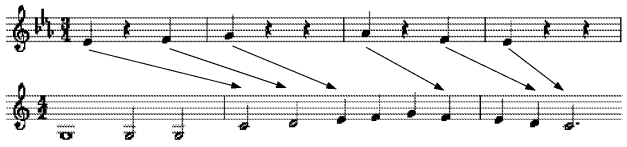


Figure 8: Top: Joseph Haydn: Symphony in E flat, Hob. 1.91, first movement (query). Bottom: G. P. da Palestrina: Offertorium "Confitebuntur caeli". This is one of the 113 matches where the common shared time is maximal (equal to the total time of the query). Most of the other 112 matches have just as little in common with the query.

5.5 PTD (transportation distance)

We used the PTD for comparing our segmented search method since for the almost exclusively monophonic RISM A/II collection, the EMD's partial matching capability is not needed. We would expect very similar results with the EMD.

Like P1, P2, and P3, the segmented PTD searches always resulted in top ranks for the exact matches. An example of a true positive found by PTD, but missed by P1, P2, and P3 is shown in Figure 9.

Although the PTD performs better than the other methods, P2 found some relevant documents that the PTD

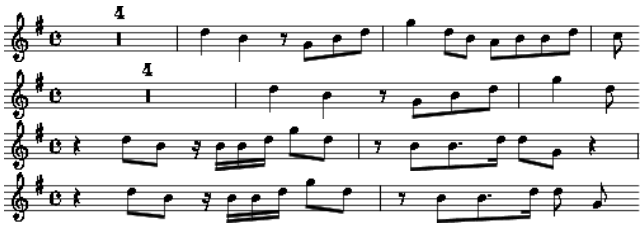


Figure 9: The first four matches for Joseph Eybler's "Dies sanctificatus", returned by a segmented PTD search. The distance is 0 for the top two matches, which are both occurrences of the Eybler piece, and 1.333 for the next two, which are both occurrences of Gregor Rösler's sacred song "Dies sanctificatus".

missed. Usually, the reason was the same as for the example shown in Figure 7. Here, our segmenting method does not extract corresponding groups of notes from the query and the database document. This could be addressed by creating multiple segments with different numbers of consecutive notes, all starting at the same note, instead of only one segment with 6 consecutive notes. This would increase the number of segment searches, but probably improve precision and recall.

It is still an open question whether 6 really is the optimum number of consecutive notes, and whether our search performance can be improved with a better method of combining partial search results.

6. CONCLUSIONS

Our comparison of our method with PROMS and C-Brahms indicates that our method delivers better results, is faster than C-Brahms for realistically large numbers of documents (hundreds of thousands or more), and puts fewer constraints on data and queries.

6.1 Better results

For the RISM A/II collection, our method finds more relevant documents than any of the three polyphonic C-Brahms algorithms P1, P2 and P3. Our method finds more than twice as many relevant documents as P1, of which PROMS is essentially an efficient implementation. However, we did not evaluate the possibility of fuzzy queries that PROMS/notify offers. Those are equivalent to sets of non-fuzzy queries.

6.2 More efficient algorithm

Our algorithm needs $O(k \log n)$ L_∞ norm calculations plus k transportation distance calculations (for point sets representing music segments whose duration is bounded by a constant), where k is the number of reported point sets and n the number of music segments in the database. This is better than the complexity of the best C-Brahms algorithm in our comparison, P2. Because of the lack of a suitable indexing structure, P2 needs $O(d)$ comparisons, each of them with a complexity of $O(ij \log i)$, where d is the number of documents, i the query length, and j the length of the compared document. Even if there was a suitable indexing structure which would reduce the number of comparisons, P2 would still need more time for longer documents, while for our method, only the sum of all document sizes matters, not the sizes of individual documents. In practice, with half a

million of database documents on a 1-GHz machine with 1 GB of memory, answering queries with the C-Brahms algorithms takes hours instead of seconds or minutes with transportation distances.

6.3 Fewer constraints on data

Unlike PROMS/notify or C-Brahms, our method is continuous, and it supports both tempo and pitch fluctuations. The "notify" system supports tempo fluctuations, but no pitch fluctuations, and C-Brahms works only if neither tempo nor pitch fluctuate. Our database only contains pieces without tempo or pitch fluctuations, but when used with the EMD, our method can correctly match the distorted query shown in Figure 1 with Bach's Brandenburg concerto No. 5.

Our method is based on a symmetric distance measure. Therefore, the query does not have to be a subset of database documents, like for C-Brahms and PROMS/notify.

6.4 Possible improvements

There are some ways in which our method could still be improved: To avoid missing desirable matches such as the third match shown in Figure 7, we could use several segment lengths instead of only one. That is, for each beginning of a segment, create segments containing, for example, 6, 7, 8, 9, 10, 11, and 12 consecutive notes. Currently, we have about 1.5 million segments extracted from 476,000 melodies. Every additional segment size would add a bit less than another 1.5 million segments to the database.

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. ACKNOWLEDGEMENTS

We thank Roelof van Zwol for letting us use his evaluation testbed [16], Agatha Walczak-Typke for proofreading this article, and Marc den Hoed and Justin de Nooijer for participating in our experiment.

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.

[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] *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

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

[12] 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.

[13] R. Typke, F. Wiering, and R. C. Veltkamp (2004). A search method for notated polyphonic music with pitch and tempo fluctuations. *ISMIR 2004: Fifth International Conference on Music Information Retrieval*

[14] 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.

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

[16] R. van Zwol and H. Oostendorp (2004). Google's "I'm feeling lucky", Truly a Gamble? *Proceedings of the Fifth International Conference on Web Information Systems Engineering*, Brisbane, Australia