A TUNNELING-VANTAGE INDEXING METHOD FOR NON-METRICS

R. Typke[1]
Austrian Research Institute
for Artificial Intelligence (OFAI), Vienna

A. C. Walczak-Typke
Kurt Gödel Research Center for Mathematical Logic
University of Vienna

ABSTRACT

We consider an instance of the Earth Mover's Distance (EMD) useful for comparing rhythmical patterns. To make searches for r-near neighbours efficient, we decompose our search space into disjoint metric subspaces, in each of which the EMD reduces to the l_1 norm. We then use a combined approach of two methods, one for searching within the subspaces, the other for searching between them. For the former, we show how one can use vantage indexing without false positives nor false negatives for solving the exact r-near neighbour problem, and find an optimum number and placement of vantage objects for this result. For searching between subspaces, where the EMD is not a metric, we show how one can guarantee that still no false negatives occur, and the percentage of false positives is reduced as the search radius is increased.

1 INTRODUCTION

Searching a database for a melody or rhythm is often equivalent to a special version of the nearest neighbour problem: one wants to find items which are similar, but not necessarily identical to a given query, and be sure to retrieve all such items up to a certain level of dissimilarity. With a distance measure which adequately captures similarity, this amounts to retrieving all items which lie inside a ball around the query, where the radius of the ball is the dissimilarity threshold up to which retrieval results are considered relevant. We will call this search radius r, and items whose distance from a query is at most r will be called *r-near neighbours*.

If the database is large, one needs a data structure which makes it possible to retrieve the r-near neighbours without comparing the query to each point in the database. For high numbers of dimensions, where exact methods such as kd-trees do not offer much improvement over a linear search, a number of approximate methods have been suggested. Andoni and Indyk [2] point to many approximate methods and describe one of the most popular among them, *locality sensitive hashing* (LSH), in detail. With LSH, one applies a hash function to every data point. The hash function must be chosen so that there is a high probability for points which are close to each other to be hashed into the same bin. To search for nearest neighbours, one then applies the hash function to the query and searches the bin into which the query was hashed. However, LSH relies on the distance measure being a metric.

We describe a method which combines *vantage indexing* with *tunneling*, an approach applicable to certain non-metric distance measures. Our specific distance measure is non-metric; however, we can decompose our search space into disjoint metric subspaces. Within the metric subspaces, the non-metric reduces to the l_1 norm. We use a new variant of the vantage indexing method which can be used for solving the exact r-near neighbour problem (i. e., without resorting to approximation) for the l_1 norm in moderately high dimensional spaces (no more than about 9 dimensions). It guarantees to retrieve exactly the items within the ball of interest, without any false positives or false negatives, with a time complexity of essentially $\mathrm{O}(\log n)$ (n=number of data points), but at the cost of using $\mathrm{O}(n2^{j-1})$ storage space (j=number of dimensions). It is possible to use less storage space, but at a cost: false positives can occur.

For certain locality sensitive hash functions, LSH is similar to vantage indexing. Andoni and Indyk propose hash functions for the l_1 and l_2 norms which involve calculating the distances to an arbitrary number of randomly chosen vantage objects and then putting all data points into the same bin whose distances to the vantage objects lie in a certain range. We, on the other hand, use exactly as many vantage objects as needed, placed in the space so that we are guaranteed to retrieve all data points we should retrieve, and no others. Also, we avoid the need to pick a bin size which might not fit very well with the actual search radius.

While our method for vantage indexing is specific to instances where the EMD reduces to metric subspaces with regular ball shapes, the second part of our combined method is more generally applicable to prametric spaces that can be decomposed into metric subspaces. By "tunneling", we can efficiently search between metric subspaces, still without introducing false negatives. False positives can occur, but their percentage among the retrieved items shrinks as the search radius grows, making them tolerable.

2 MEASURING MELODIC OR RHYTHMIC SIMILARITY WITH THE EMD

2.1 The Earth Mover's Distance

The distance measure discussed in this paper measures the distance between weighted point sets. Intuitively speaking,

[1] R. Typke gratefully acknowledges support by the Austrian Research Fund (FWF), Project Number M1027-N15.

a weighted point set a_i can be imagined as an array of piles of dirt each equal to w_i units, situated at position x_i. The role of the supplier is arbitrarily assigned to one array and that of the receiver to the other one, and the arrays of piles are made to look as similar as possible by shifting dirt from piles in the supplier array to piles in the receiver array. The Earth Mover's Distance (EMD) then measures the minimum amount of work needed to make two arrays of piles as similar as possible in this way. See [3] for a more detailed description of the EMD. We now define the EMD formally:

Definition. Fix a *ground distance* d on $\mathbb{R}^k$. The ground distance can, but need not be, a metric.

Let $A = \{a_1, a_2, \ldots, a_m\}$, $B = \{b_1, b_2, \ldots, b_n\}$ be *weighted point sets* such that $a_i = \{(x_i, w_i)\}, i = 1, \ldots, m$, $b_j = \{(y_j, v_j)\}, j = 1, \ldots, n$, where $x_i, y_j \in \mathbb{R}^k$ with $w_i, v_j \in \mathbb{R}^+ \cup \{0\}$ being the respective weights.

Let $W_A = \sum_{j=1}^{n} w_i$ be the total weight of set A; the total weight W_B of the set B is defined analogously.

Let $d_{ij} = d(x_i, y_j)$ denote the ground distance between individual coordinates in A and B, without regard to their weight.

A *flow matrix* $F = (f_{ij})$ between A and B is an $m \times n$ matrix of non-negative real numbers, such that for each $1 \leqslant i \leqslant m$, $\sum_{j=1}^{n} f_{ij} \leqslant w_i$, and for each $1 \leqslant j \leqslant n$, $\sum_{i=1}^{m} f_{ij} \leqslant v_j$. Furthermore, we require that $\sum_i \sum_j f_{ij} = \min(W_A, W_B)$. Denote by $\mathcal{F}$ the collection of all possible flow matrices between A and B.

The *Earth Mover's Distance*, $\mathrm{EMD}_d(A, B)$, between A and B is defined as

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

For the remainder of this paper, we will assume that the ground distance for the EMD is the Euclidean metric l_2. With this assumption in mind, we drop the subscript referring to the ground distance from our formulas, writing $\mathrm{EMD}(A, B)$ instead of $\mathrm{EMD}_{l_2}(A, B)$.

The EMD is a useful measure for music similarity, as was demonstrated at the annual MIREX comparison of music retrieval algorithms in 2006. Useful properties of the EMD include its continuity, its ability to support partial matching, and its robustness against distortions of tempo and pitch when measuring melodic similarity for symbolically encoded music. For doing so, one can represent every note by a point in the two-dimensional space of onset time and pitch. The weights of points can be used to encode the importance of notes. See [6] for details.

The EMD has one major disadvantage which can cause technical difficulties: it is in general not a metric. Specifically, the triangle inequality does not hold, and while the EMD of a point to itself is 0, there can exist distinct points that are also EMD 0 from one another. While there is no universally accepted terminology in the mathematical literature for weak distance measures, there is some precedent for calling weak distance measures with properties like the EMD *symmetric prametrics*.

It should be emphasized that the EMD does behave as a metric if one restricts the domain of the EMD to point sets having a given weight, assuming that the ground distance is a metric [3]. One can take advantage of this property when working with an EMD which is a prametric by decomposing the space of possible point sets into subspaces each containing only point sets having a given weight. We will refer to such subspaces as *metric subspaces* of the EMD space.

2.2 Comparing symbolic rhythmic patterns: a nice special case

The instance of the EMD described in this subsection is of particular interest for searching rhythmic patterns. We call this EMD the *Manhattan EMD*, and note that it is well-behaved from a geometric point of view.

Rhythmic patterns can naturally be represented as sequences of onset times. See Figure 1 for an illustration. To render irrelevant a musical segment's tempo and location within a piece of music, we scale their numeric representations to a fixed duration (say, 60) and translate them so that they start at position 0.

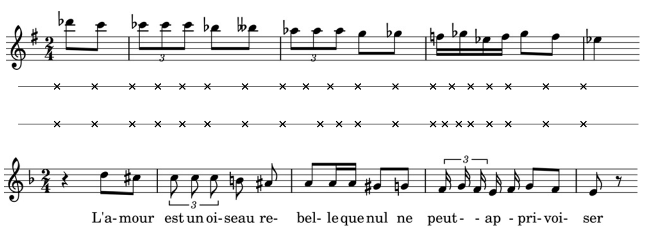
L'a-mour est un oi-seau re- bel- le que nul ne peut- - ap - pri-voi- ser

Figure 1. Comparing rhythms using sequences of onset times. *Top:* Ferruccio Busoni: Sonatina No.6: Chamber Fantasia on Bizet's Opera Carmen (3rd theme); *bottom:* Georges Bizet: Carmen: Habanera; *middle:* two sequences of onset times representing the incipits at the top and bottom. A possible numeric representation of the two series of onset times: **Busoni**: (0, 3, 6, 8, 10, 12, 15, 18, 20, 22, 24, 27, 30, 31.5, 33, 34.5, 36, 39, 42); **Bizet**: (0, 3, 6, 8, 10, 12, 15, 18, 21, 22.5, 24, 27, 30, 31, 32, 33, 34.5, 36, 39, 42).

When comparing normalized sequences of onsets containing the same number of onsets, the Manhattan EMD equals the sum of the absolute differences between corresponding onsets a_i and b_i in onset sequences $A = a_1 \ldots a_n$ and $B = b_1 \ldots b_n$, divided by the number of onsets: $\mathrm{EMD}(A, B) = \frac{\sum_{i=1}^{n} |a_i - b_i|}{n}$. Thus, if we restrict ourselves to point sets of a certain given length, the Manhattan EMD (with l_2 as ground distance) is a metric and is equal to the l_1 norm (also known as "Manhattan norm").

Since every normalized segment starts with 0 and ends with 60, we omit these two numbers and view n-onset segments as points in an $(n-2)$-dimensional space. All segments lie in the subset of $\mathbb{R}^{n-2}$ where the coordinates are strictly increasing.

3 VANTAGE INDEXING FOR MANHATTAN EMD

Vantage indexing (introduced by Vleugels and Veltkamp [7]) is an approach to the retrieval of objects from a large database which are similar to a given query. The search is restricted to items whose pre-calculated distances to a small set of pre-chosen *vantage objects* are similar to the query's distances to the same vantage objects.

3.1 Ball shapes

If one works with the l_1 norm, a ball (the set of all points whose distance lies within a certain radius around a point of interest) has the shape of a cross-polytope. A one-dimensional cross-polytope is a line segment, a two-dimensional cross-polytope is a square, for three dimentions, an octahedron, and so forth.

3.2 Creating a ball by intersecting "onion layers"

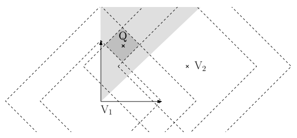

Figure 2. In this two-dimensional example, only the light gray area is inhabited by database objects since only there the second coordinate is greater than the first. The dark gray ball around Q with a given radius can be created by intersecting onion layers around V_1 and V_2 whose thickness equals the radius.

We call the area between the surfaces of two balls of differing radii around a vantage object an "onion layer". [1] An onion layer contains all items in the database whose distance from the vantage object lies within a certain range. Searching an onion layer can be done efficiently if one has built an index (for example, a range tree) for the database column that contains the distance to a vantage object. By building an index for multiple columns containing distances to different vantage objects (for example, a nested range tree), one can also search intersections of onion layers efficiently.

If one can cover the ball around the query object whose radius equals the search radius by using intersections of onion layers in a way that the intersection of onion layers equals exactly the ball of interest, neither false negatives nor false positives will occur. Since the intersection of onion layers can be searched efficiently, the ball of interest can as well. See Figure 2 for an illustration.

Certain metric spaces have the property that there exists a finite number of points $x_1, \ldots, x_n$ such that $\bigcap_i (B(x_i, R + r) \setminus B(x_i, R)) = B(y, r)$ for some point y (here we denote the ball with radius r around a point x with $B(x, r)$). In other terms, some spaces have the property that one needs only finitely many vantage objects so that the intersection of onion layers is exactly that of a ball. The l_1 space has this property, as do other spaces whose balls are regular objects with finitely many faces. An example of a metric space that does not have this property is Euclidean space.

[1] Inspiration has many sources, and onions have other good uses [5].

3.3 Optimum placement and number of vantage objects necessary for optimum indexing

In the case of one dimension, one vantage object is enough to always achieve the desired effect of perfectly covering only the interesting ball with onion layers. However, one has to place the vantage object correctly. See Figure 3 for good and bad examples.

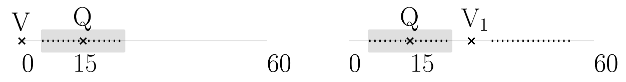

Figure 3. Of interest is the ball around Q (at 15) with radius 10, shown as a grey bar. *Left:* This ball is identical to the intersection of an onion layer around V with distance range of 5 to 25 and the inhabited part of the space. *Right:* If the vantage object is not put either to the left or to the right of inhabited space, corresponding onion layers can intersect with the inhabited space in two disjoint places (shown as dotted lines) – one would need to use another vantage object for the intersection of onion layers to be identical to the ball of interest.

For two dimensions, two vantage objects are necessary, and they are sufficient if they are placed, for example, like in Figure 2. Here, the onion layers intersect in two square-shaped regions, but only one of them (the one in the gray area) can ever contain data points. Therefore, no false positives can occur.

One can cover exactly the ball of interest and no other region (except for regions outside the inhabited space) by using 2^{j-1} vantage objects (j is the number of dimensions of the space) and placing them as follows:

- Let m be some number which is larger than any coordinate that can ever occur in the inhabited subspace (in the Bizet and Busoni example from Section 2.2, we could let $m = 60$).
- For one dimension, place one vantage object at the coordinate (0) (as illustrated in Fig. 3, left).
- For j dimensions, use twice as many vantage objects as for $j - 1$ dimensions. The coordinates of the vantage objects for j dimensions are based on those for $j - 1$ dimensions: for each vantage object position in $j - 1$ dimensions, create two new lists of coordinates for j dimensions by appending 0 or m, respectively,

to the list of coordinates for $j-1$ dimensions.

We need at least 2^{j-1} vantage objects because a cross-polytope has 2^j facets (a *facet* is a $(j-1)$-dimensional structure). For the intersection of onion layers to equal the ball of interest, every facet of the ball needs to coincide with the surface of an onion layer. One can use as few vantage objects as possible if one covers opposite facets of the cross-polytope with the surfaces of an onion layer around the same vantage object. This is exactly what happens if one uses 2^{j-1} vantage objects and places them as above.

4 PARTIAL MATCHING BY TUNNELING BETWEEN METRIC SUBSPACES

When searching sequences of onsets that were detected in audio files, there are two problems: the detection is not always completely reliable, producing both false negatives and false positives, and often there will be multiple voices with onsets, while a user might want to search for rhythmic patterns which occur only in one voice. Therefore, for successfully searching sequences of onsets from audio data, one should be able to ignore a certain number of onsets and still find matching subsequences or supersequences. In this section, we will show how one can use tunnels between metric subspaces to achieve partial matching while still benefiting from the optimum vantage indexing within the subspaces of sequences with equal numbers of onsets.

The EMD provides partial matching as described above for point sets whose weight sums are unequal. Unfortunately, the EMD does not obey the triangle inequality in such a case. This makes it impractical to directly apply vantage indexing since there would be no way of controlling the number of false negatives. Also, the locality sensitive hashing method, which also relies on the triangle inequality, becomes unusable.

Our approach is inspired by the case of the Manhattan EMD, which is rather easy to visualize. In particular, one can precisely visualize the neighbourhood, or "ball" of r-near neighbours in the n-dimensional metric subspace for an m-dimensional query point: assume that $(q_1, q_2, \ldots, q_m)$ is a query, where the initial 0 and final 60 are dropped, so that the query is an m-dimensional object.

First, we visualize the neighbourhoods in higher-dimensional metric subspaces, say of dimension $m+i$. In this case, the set of points which are of distance 0 from the query comprise $\binom{m+i}{i}$ many i-dimensional hyperplanes which are parallel to axes, and pass through the points that are the various possible completions of the query vector to $m+i$ dimensional space using i-many zeros. Then, the r-near neighbours of the query are found in the union of the r-balls, as calculated in the $m+i$-dimensional metric subspace, around each of the points of distance 0. The shape of these neighbourhoods is rather awkward, and is no longer a ball, but rather the cartesian product of a ball and a hyperplane. This shape is difficult to search efficiently. The reader may wish to think about the case of $m=2$ and $i=1$.

The neighbourhoods in lower-dimensional metric subspaces are a easier to describe. The points of distance 0 from the query in the $m-i$-dimensional metric subspace are simply the $\binom{m}{i}$ many points obtained by reducing the query to an $m-i$-dimensional point. The neighbourhood is then the union of the r-balls around these points.

Our intuition is that a given point can be somehow connected with "tunnels" to the points in other metric subspaces which are of distance 0, and then take advantage of the triangle inequality in the metric subspaces to use vantage indexing. However, the points in the database can possibly not include any such points of distance 0, so we have to modify this simplistic idea of a tunnel by instead linking to nearest neighbours. We precalculate, for every point set in all spaces except for that with the lowest dimensionality, links to its nearest neighbours in lower-dimensional spaces. If such links exist, one can efficiently retrieve not only the nearest neighbours in the same space as the query (e.g. as described in Section 3), but also the nearest neighbours in higher-dimensional and lower-dimensional spaces which are linked to the retrieved items from the space with the same dimensionality as the query. This yields almost the same result as an exhaustive linear search, but requires only logarithmic complexity.

One needs, however, to limit the number of connections per high-dimensional object by limiting the range of dimensionalities across which one wants to search. By introducing such a constant bound, one ensures that the number of connections grows only linearly with the number of items in the database. For the application of searching for rhythmic and melodic patterns, this means that one should limit the partial matching to a certain maximum number of missing or extra notes which may be ignored when the dissimilarity is calculated. Such a limit is probably desirable: without it, a short (low-dimensional) pattern could become very similar to very different long (high-dimensional) patterns.

4.1 Efficiently building tunnels

By projecting high-dimensional objects onto lower-dimensional subspaces and then using vantage indexing as described in Section 3 for finding the nearest neighbour of the projection, one can benefit from the logarithmic search complexity of vantage indexing for the purpose of building connections between nearest neighbours in metric subspaces of differing dimensionality.

Algorithm 1 can be used for building connections between objects of different dimensionality. Note that its runtime lies in $\mathrm{O}(n \log n)$ since we limit the number of dimensionalities to cross (the number of notes to ignore) to a constant t. The outermost loop is executed $\mathrm{O}(n)$ times. The

Algorithm 1 Build connections between nearest neighbours in subspaces of different dimensionality.

```
for all point sets $p$ do
    for $i$ := 1 to max. number of dimensions to cross do
        for all subspaces $s$ with $i$ dimensions less than $p$ do
            project point set $p$ onto subspace $s$
            use the vantage index of subspace $s$ for finding $p$'s nearest neighbours in $s$
            create a connection between $p$ and its nearest neighbour in $s$ (or all of them if they have equal distances).
        end for
    end for
end for
```

two inner loops each are executed only a constant number of times. Each point set p with j dimensions can be projected onto $(j - i)$-dimensional space in $\binom{j}{i}$ many ways, and this is bounded by a constant as long as the maximum possible number of dimensions and the number of subspaces to cross are bounded. Using the vantage index within the subspace onto which p is projected takes $\mathrm{O}(\log n)$ steps, which leads to an overall runtime of $\mathrm{O}(n \log n)$.

The space complexity can be expected to lie in $\mathrm{O}(n)$ since the number of connections to create should be bounded by $\binom{j}{i}$. There is one exception: if there are many nearest neighbours in the subspace which all have the same distance from the projected point, we might store more than $\binom{j}{i}$ connections for one projected point.

4.2 Using tunnels for retrieving items from higher dimensions than the query

To find rhythmic patterns which are similar to a given query, but contain up to t additional notes (which should be ignored when calculating dissimilarity), one can tunnel into higher dimensions by using the connections previously described. Given a query Q, search radius r, and a database with a vantage index for each metric subspace and connections between items and the nearest neighbours of their projections in subspaces with up to t fewer dimensions, one can retrieve Q's r-near neighbours with the same dimensionality as Q or up to t dimensions higher as follows:

```
Retrieve $Q$'s nearest neighbours $n$ within radius $r$ with as many dimensions as $Q$, using the vantage index for this subspace.
for all $r$-near neighbours $n$ do
    Retrieve every item which is connected to $n$ and has a dimensionality higher than that of $Q$.
end for
```

This will retrieve exactly the desired objects from the subspace where the query Q resides, but for the higher-dimensional objects, both false positives and false negatives may occur.

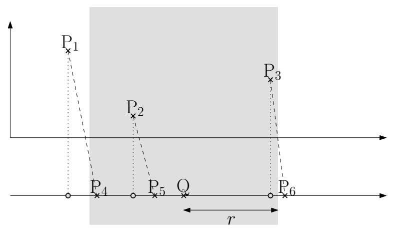

Figure 4. False negatives and false positives resulting from tunneling, and how to avoid them.

The database shown in Figure 4 contains 6 point sets $P_1, \ldots, P_6$. Three, $P_1, \ldots, P_3$, are two-dimensional, the others, one-dimensional. The query Q is one-dimensional. The area of interest within the search radius r around Q is marked grey.

False positives: It is conceivable that the projection of a higher-dimensional object onto Q's subspace lies just outside the search radius, but its nearest neighbour in Q's subspace happens to lie within the search radius. An example is P_1, whose projection onto the subspace (shown as a circle) has a nearest neighbour beyond the border of the grey area.

False negatives: It is also possible that while the projection of a higher-dimensional object onto Q's subspace lies inside the search radius, the closest object in Q's subspace lies outside the search radius. In this case, illustrated with P_3 and P_6, the higher-dimensional object will not be retrieved. In the extreme case that there is no single object inside the search radius in Q's subspace, no higher-dimensional objects whatsoever will be retrieved.

Controlling false negatives and false positives. To avoid all false negatives and limit the badness of false positives to a threshold e, one can add the projections as additional "ghost points" to the database if their nearest neighbour in the subspace is further away than $e/2$, and extend the search radius by $e/2$.

The distance of false positives to the query will be at most e higher than desired because in the worst case, the nearest neighbour of the projection will lie on the line connecting the projection with the query. The nearest neighbour can be up to $r + e/2$ away from the query, while the projection can be another $e/2$ away from the nearest neighbour, leading to a total maximum distance of $r + e$ for false positives.

Such additional ghost points would be added as part of the task of building the index, and so would not slow down the search process. It would also not increase the computational complexity of building the index – the only price is some extra space for storing ghost points wherever some point from higher dimensions gets projected into a sparsely populated area in a subspace. There is a tradeoff between the required additional space and the maximum possible distance error for false positives.

4.3 Using tunnels for retrieving items from lower dimensions than the query

As we have seen in Algorithm 1, subspaces can be searched efficiently for a query even without using the pre-computed connections. Since these connections are already there, they can be used instead of projecting the query onto each possible subspace. To avoid false negatives, one still needs to search the area around the nearest neighbour in the subspace with the search radius $r + a$, using the vantage index of the subspace, where a is the distance between the projection of Q onto the subspace and its nearest neighbour in that subspace. Whenever a is greater than zero, the possibility of false positives gets introduced. If such false positives cannot be tolerated, one can resort to not using the connections but instead projecting the query onto all possible subspaces.

5 EXPERIMENTAL EVALUATION OF VANTAGE INDEXING WITH TUNNELING

For an experimental evaluation, we used Dixon's onset detector from his "BeatRoot" system [4] for extracting onsets from various recordings with piano music (piano concertos and solo works by Beethoven, Liszt, and Schumann). We used piano music because piano onsets are relatively easy to detect correctly. The detected onsets were grouped into overlapping sequences of 5 to 8 onsets. $40,000$ of these sequences were put into a database and indexed for optimum retrieval within subspaces as described in Section 3. We also added connections between subspaces as described in Section 4, with connections spanning up to 3 dimensionalities, thus allowing for ignoring up to 3 extra or missing notes.

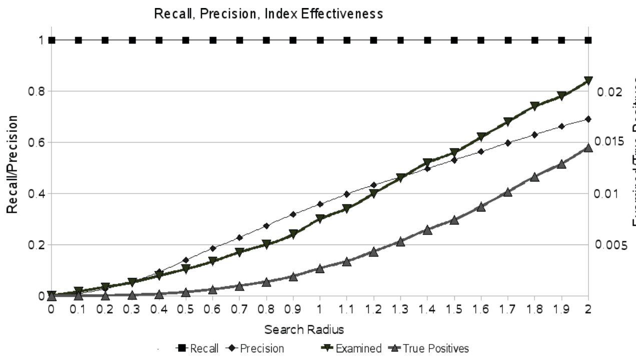

Figure 5. Recall ■, precision ♦ (left scale), ratio of retrieved items ▼, and ratio of true positives ▲ (right scale) for the tunneling method and different search radii, averaged over 20 queries.

For every search radius in a range from 0 to 2, we randomly picked 20 items from the database as queries and used the index to retrieve the most similar items according to the EMD. To calculate precision and recall, we also did exhaustive searches for each of these queries (by calculating the EMD between the query and each of the $40,000$ items in the database) and counted false positives and false negatives.

With this experiment, we verified that it is indeed possible to reduce the number of false negatives to zero and limit the distance error of false positives to e by adding the projections of high-dimensional points to the database in cases where their nearest lower-dimensional neighbour is further away than $e/2$, and by increasing the search radius by $e/2$. Figure 5 shows that our index works well: for example, with a search radius of 1.9, we need to retrieve 1.3% of the whole database (curve: ▲), but we actually retrieve 1.95% (▼ Examined $= 0.0195$), which still relieves us from looking at 98.05% of the database. For this experiment, the value of e was 1, i.e., we inserted ghost points only if the projection and its nearest neighbour were more than 0.5 apart. With this threshold, we needed about as many ghost points as real points. The nested range tree we used for the vantage table [1] has a space complexity of $O(n \log n)$.

The larger our search radius, and thus the more important it is that we do not get swamped with false positives, the lower their percentage in the search results. This is probably due to the fact that false positives can only occur near the surfaces of searched balls, and when the search radius is increased, one expects an ever larger percentage of objects inside the ball rather than near its surface. Although the precision is not very good for very low search radii, this does not matter much since for low search radii, very few items are retrieved at all from the database. As the number of retrieved database items grows with the search radius, the pain resulting from false positives is relieved by the fact that at the same time, the percentage of false positives among the retrieved items shrinks.

References

[1] CGAL, Computational Geometry Algorithms Library. http://www.cgal.org.

[2] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *CACM*, 51(1):117–122, 2008.

[3] S. Cohen. *Finding Color and Shape Patterns in Images*. Stanford University, 1999. Ph.D. thesis.

[4] S. Dixon. Onset detection revisited. In *Proc. of the 9th Int. Conf. on Digital Audio Effects (DAFx06)*, 2006.

[5] T. Keller. The importance of onion soup. In *Bouchon*, pages 47–50. Artisan, 2004.

[6] R. Typke. *Music Retrieval based on Melodic Similarity*. Doctoral Thesis, Universiteit Utrecht, http://rainer.typke.org/books.html, 2007.

[7] J. Vleugels and R. C. Veltkamp. Efficient image retrieval through vantage objects. *Pattern Recognition*, 35(1):69–80, 2002.