This is a post-peer-review, pre-copyedit version of a paper published in Amato G., Gennaro C., Oria V., Radovanović M. (eds) *Similarity Search and Applications. SISAP 2019*. Lecture Notes in Computer Science, vol 11807. Springer, Cham. The final authenticated version is available online at: https://doi.org/10.1007/978-3-030-32047-8_21

# Query Filtering with Low-Dimensional Local Embeddings

Edgar Chávez[1], Richard Connor[2], and Lucia Vadicamo[3]

[1] Centro de Investigación Científica y de Educación Superior de Ensenada (CICESE), México
[2] Division of Mathematics and Computer Science, University of Stirling, Scotland
[3] Institute of Information Science and Technologies (ISTI), CNR, Via Moruzzi 1, 56124 Pisa, Italy
elchavez@cicese.mx
richard.connor@stir.ac.uk
lucia.vadicamo@isti.cnr.it

**Abstract.** The concept of *local pivoting* is to partition a metric space so that each element in the space is associated with precisely one of a fixed set of reference objects or pivots. The idea is that each object of the data set is associated with the reference object that is best suited to filter that particular object if it is not relevant to a query, maximising the probability of excluding it from a search. The notion does not in itself lead to a scalable search mechanism, but instead gives a good chance of exclusion based on a tiny memory footprint and a fast calculation. It is therefore most useful in contexts where main memory is at a premium, or in conjunction with another, scalable, mechanism.

In this paper we apply similar reasoning to metric spaces which possess the four-point property, which notably include Euclidean, Cosine, Triangular, Jensen-Shannon, and Quadratic Form. In this case, each element of the space can be associated with two reference objects, and a four-point lower-bound property is used instead of the simple triangle inequality. The probability of exclusion is strictly greater than with simple local pivoting; the space required per object and the calculation are again tiny in relative terms.

We show that the resulting mechanism can be very effective. A consequence of using the four-point property is that, for $m$ reference points, there are $\binom{m}{2}$ pivot pairs to choose from, giving a very good chance of a good selection being available from a small number of distance calculations. Finding the best pair has a quadratic cost with the number of references; however, we provide experimental evidence that good heuristics exist. Finally, we show how the resulting mechanism can be integrated with a more scalable technique to provide a very significant performance improvement, for a very small overhead in build-time and memory cost.

**Keywords:** metric search · extreme pivoting · supermetric space · four-point property · pivot based index

## 1 Introduction

In a metric space, the distances among a set of any three objects may be used to construct a triangle in Euclidean space, with corresponding vertices and edge lengths. If any two of these distances are known, the triangle inequality property can be used to determine upper and lower bounds for the third.

For any supermetric space [12], the distances among a set of any four objects can be used to construct a tetrahedron in Euclidean space, with corresponding vertices and edge lengths. An equivalent lower-bound calculation can be made for a final edge length, given any four objects, when any five of the six distances among them are known. In simple terms, the five distances can be used to fix two adjacent faces of a tetrahedron, and lower and upper bounds for the last edge can be easily determined by considering the rotation of the two triangles around their common edge.

We show a novel way of exploiting this situation, as follows. From a finite metric space $S$, a relatively small set of reference objects $P$ is selected. For all $p_i, p_j \in P$, the distance $d(p_i, p_j)$ is calculated and stored. For each element $s_i$ in $S$, a single pair of reference objects $\langle p_x, p_y \rangle$ is selected, and the the distances $d(s_i, p_x)$ and $d(s_i, p_y)$ are stored. Thus the space $S$ is represented as a set of tuples $\langle x, y, d(s_i, p_x), d(s_i, p_y) \rangle$, indexed by $i$, therefore requiring only a few bytes per object.

When a query is executed, the distances $d(q, p_i)$ for each $p_i \in P$ are first calculated. At this point, considering the objects $q, p_x, p_y$ and any $s_i \in S$, five of the six distances among them can be retrieved, leaving only $d(q, s_i)$ as an unknown. Therefore, for each element of $s_i$ a lower-bound for $d(q, s_i)$ can be calculated, with a cheap geometric calculation, without any requirement to access the original value $s_i \in S$.

The approach we take is based on the observation that, for a selection of $n$ reference points, there exist $\binom{n}{2}$ pairs from which the representation of each data point can be selected. This number, of course, becomes rapidly very large even with modest increases in $n$. If for each element of $S$ we can find a particularly effective pair $p_i, p_j$, within this large space, then this tiny representation of $S$ can be used as a powerful threshold query filter. This exclusion mechanism leads to a sequential scan, which is virtually unavoidable in light of a recent conditional hardness result in [18] for nearest neighbor search, even in the approximate setup, computing a $(1 + \epsilon)$-approximation to the nearest neighbor requires $\Omega(N - \delta)$ time, with $N$ the size of the database.

The above hardness result has been suspected for a long time by the indexing community, and it has been named the *curse of dimensionality*. It is known, for example, that a metric inverted index [1] has high recall rates only if a substantial part of the candidate results is revised. We aim our approach at this final part of query filtering or re-ranking.

The contributions of this paper are as follows:

1. We show that the outline mechanism is viable. For the well-known SISAP benchmark data sets we show exclusion rates of over 99% can be achieved using our small memory footprint and cheap calculations.
2. We examine the problem of finding the best pair of reference points per datum; this can be done perfectly, but expensively, by an exhaustive search of the pair space. We show that much cheaper heuristics are also effective.
3. Finally, we show one example of how the mechanism can be combined with another, by describing its incorporation with the List of Clusters index. We use a pragmatic selection of pivot pairs to ensure that no new distances are measured at either construction or query time, and show a halving of overall query cost.

## 2 Related Work

Pivot based indexes have populated the metric indexing scene for long time. A pivot table, with the triangle inequality, is just the direct product of one dimensional projections obtained from a single pivot at a time. Each coordinate gives a lower bound to the actual distance from database points to the query. A safe choice is to take the maximum over all the available lower bounds. The most competitive algorithm published for searching is AESA [21], which proceeds as following. All the $O(n^2)$ distances between every object in the database of $n$ elements is pre-computed. With this, every object in the database is a potential pivot. At query time a subset of the $n$ pivots is selected, one at a time, using a heuristic which consist in selecting the $j+1$ pivot, the closest to the query, using as bound the $j$ pivots known so far and the first pivot at random. The output of this heuristic is both a set of *good* pivots for the query, and the nearest object to it. Two things can be noticed from this basic approach, first, the number of pivots actually used is way smaller than $n$, and second, they are tailored for each query on the fly. Since the space usage is quadratic, the approach is impractical. Also notice that a sequential scan is implied to obtain the closest next pivot in the interaction. Linear space approaches of the same idea were used in [17], and a better heuristic for selecting the next pivot is proposed in [14]. The sequential scan can be avoided using a tree [5, 2].

Selecting the best pivot for a given query is not possible offline. A weaker alternative is to select the best pivot for each database object, increasing the probability of exclusion at query time. Two options have been explored in the literature, in [6] each pivot in the pool only keep distances to objects in the extreme of the distribution, those objects near and far the pivot. This process is sub-optimal and may end with a few objects guarded by many pivots, and many objects guarded by a few or none pivots. A second alternative, ensuring some fairness in the coverage, was proposed in [19], this time each object can select the best pivot. This latter approach is called *extreme pivoting*. In those heuristics the gain is in filtering power, when the amount of available memory is fixed. Pivot tables are useful for post filtering in a hierarchical metric index, as in [20], or they can be used as a stand alone index using directly the table as in [7, 9].

The exclusion based on the four point property was firstly proposed in [11], and generalized to $n + 1$ polytopes in [13]. The exclusion increased with the dimension of the polytope.

For post-filtering, when a primary index is applied to filter the data and only a small fraction of the databases should be checked against the original metric, a table is useful. A high rate of exclusion will prevent the use of the more expensive distance computation, and moreover, it will require to fetch a smaller number of objects from secondary memory. Hence a small table, with just a couple of coordinates, is an excellent trade-off because it can be kept in main memory. In the same spirit as the extreme pivots for unidimensional mapping, in this paper we are aiming at building a table of small memory footprint using the four point property.

### 2.1 The Four-Point Property and Supermetric Spaces

Much work on finite isometric embeddings was conducted in the $20^{th}$ century, by e.g. Blumenthal [4], Wilson [22] and Menger [16]. Blumenthal uses the phrase *four-point property* to mean a space that is 4-embeddable in 3-dimensional Euclidean space: that is, that for any four objects in the original space it is possible to construct a distance-preserving tetrahedron.

More recently we have applied these results in theoretical mathematics to the practical domain of metric search [10–12]. For this context, the important result is that the four-point property applies to many commonly-used distance metrics, including Euclidean, Cosine[4], Jensen-Shannon, Triangular and Quadratic Form distances, all of which can be safely used in conjunction with the mechanisms described here.

### 2.2 The Four-Point Planar Lower Bound

For two points that have not been directly compared, $q$ and $s_i$, it is shown in [12] how a lower bound of their distance can be established by comparing the distances between both points and two further reference points. For reference points $p_1$ and $p_2$, two triangles with a common base, $\triangle p_1 q p_2$ and $\triangle p_1 s_i p_2$, can be used to form two adjacent faces of a tetrahedron. Because of the four-point property, the unmeasured distance $d(q, s_i)$ must form the sixth edge of a tetrahedron. It is then clear, by consideration of the rotation of these triangles around the common baseline $\overline{p_1 p_2}$, that upper and lower bounds for the distance $d(q, s_i)$ can be determined as the two cases where the triangles lie in the same plane.

A lower bound of their distance can therefore be calculated by notionally plotting Cartesian points $p'_1$ and $p'_2$ arbitrarily on 2D axes, say at positions $(0, 0)$ and $(d(p_1, p_2), 0)$ respectively, and then plotting points $q'$ and $s'_i$, both above the X-axis, according to their respective distances from $p_1$ and $p_2$. Then the distance $\ell_2(q', s'_i)$ is a lower bound of $d(q, s_i)$.

[4] for the correct formulation, see [10].

The value of this is that, independently of the size of individual data values and the cost of the distance metric, any value can be represented, for a fixed choice of reference points, as a small 2D coordinate, and compared with a cheap 2D $\ell_2$ distance; the result of this comparison may mean that there is no requirement for the full comparison to be made. Of course, the value of the method depends heavily upon the probability of its success.

## 3 Distribution of Values in the 2D plane

To visualise this property we use scatter diagrams constructed as follows. The two selected reference points are plotted on the X-axis according to the distance between them, and a data set is represented as points in the 2D space plotted above the X-axis, according to their respective distances from these reference points. The triangle inequality property gives the ability to create such a plot.

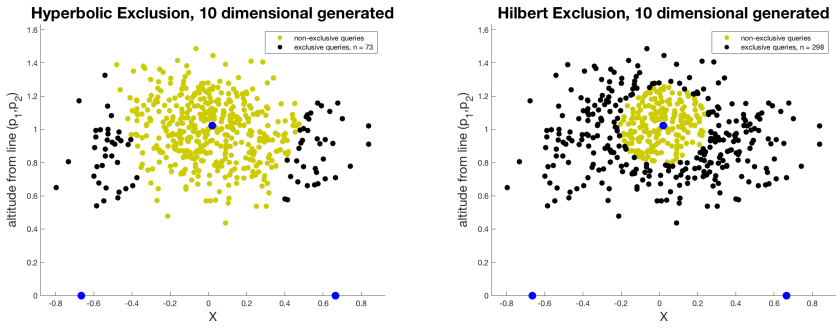


**Fig. 1.** 500 points from a generated Euclidean space plotted against randomly selected reference points. Left and right plots show the exclusion potential based on simple metric (left) and supermetric (right) properties.

Figure 1 shows two versions of such a scatter plot created from a 10-dimensional Euclidean space, using the same data and reference points. Although the triangle inequality property guarantees the ability to create such a plot, the relationship among the plotted points is more subtle.

An example query point is selected from the centre of the diagram, coloured blue. For every other point plotted in the plane, we then consider whether it *might* be within a threshold distance $t$ from this blue-coloured point, based only on the distances calculated to the two reference points. Here we have chosen $t = 0.24$, representing around one-millionth of the volume of the generated space.

The diagrams are then colour-coded so that those points which *may* be within that distance, i.e. those that cannot be excluded from a search, are highlighted, plotted in yellow. The four-point planar lower bound is illustrated on the right-hand side, clearly represented by a simple exclusion radius in the 2D plane. On

the left-hand side, only the triangle inequality property is used, giving much wider hyperbolic bounds.

These diagrams represent a situation where only two reference points have been used, with respect to a single query. The left-hand side shows the effectiveness of local pivoting, where this very small amount of information allows 73 out of 500 data points to be excluded from the candidate solution space. It can be seen on the right-hand side that, if the four-point property can be used, then 298 ex 500 potential solutions can be excluded, using exactly the same information.

## 4 Independence of Reference Points

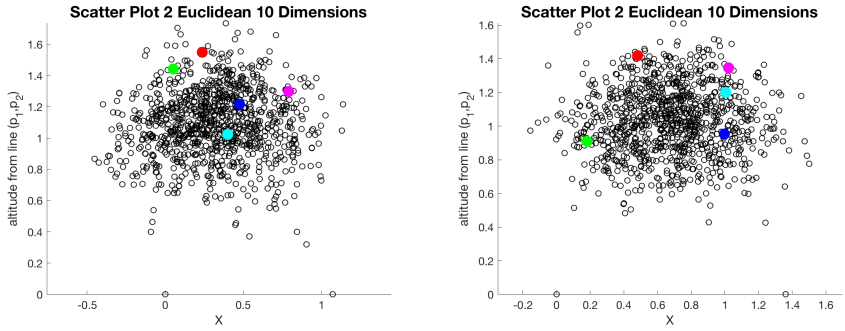


**Fig. 2.** 1,000 points plotted in the 2D plane based on two different, randomly selected, pairs of reference points. The data plotted is the same in each diagram, and the colour-coded points represent the same values. The $(X, Y)$ scatter is similar in both cases, close to uncorrelated normal distributions on both axes, but it can be seen that where an individual point lands depends on the choice of reference points.

It appears that, for a given choice of reference points, the *distribution* of other points in the 2D plane with respect to these points is fairly predictable. However, where *individual* data points land within the scatter varies widely with the choice of reference points.

Figures 2 and 3 show some diagrams to illustrate this. In each figure, a single set of data points is plotted in the $XY$ plane according to their distances from two randomly-selected reference points; the left and right sides of each figure now represent the same data plotted against a *different* choice of reference points.

In the two figures, a random selection of five data points has been made and these are highlighted in colour in the charts; that is, the coloured spots in the left and right sides of the figure represent the same data point and its position with respect to the different reference points. It can be seen that there is a relationship among the positions where the coloured dots are plotted, but only a relatively weak one.

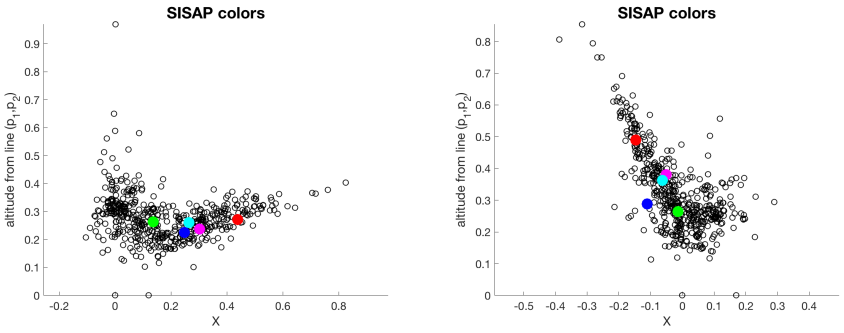


**Fig. 3.** 500 points from the *colors* data set plotted in the 2D plane, again with two randomly selected pairs of reference points. It can be seen how much the distribution changes in a non-uniform set with the choice of reference points. Again, where an individual point lands within the scatter depends on the choice of reference point.

### 4.1 Choice of Reference Points

An underlying hypothesis in our work is that the distribution of queries within $U$ will be similar to the distribution of $S$ within $U$. Thus, looking at the scatter diagrams in Figures 2 and 3, we could be viewing the distribution of either data or queries with respect to those same reference points. The probability of successful elimination, for a given $q$ and $s_i$, therefore depends upon the choice of reference points, and the relative position of both $s_i$ and $q$ with respect to them.

If the hypothesis is correct, then the notion of a "good" pair of reference points for an individual $s_i \in S$ corresponds to the (inverse) *density* of the region where $s_i$ lands, within a representative set. If query and datum lie further than the query threshold within the 2D plot, then the datum cannot be a solution to the query; this is most likely to occur when either query or datum lie within a sparsely populated region of the plane. If queries and data follow the same distribution patterns, then the best pair of reference points can be selected with reference to a representative set of data points from within $S$.

## 5 Selection and Query

### 5.1 Selection of Best Reference Pair

It is possible to use a statistical technique to select a good reference point pair per individual datum. A sample set of data is used, the *witness* set.

For a query over a finite metric space $(S, d)$, first a set of $n$ objects is taken from $S$ and used to form a set $P$ comprising numbered reference points $p_i$. For a given set of $n$ reference objects, each of the $\binom{n}{2}$ pairs $p_i, p_j$ is considered. For each, a 2D Euclidean space is built, exactly corresponding to those depicted in the earlier figures. Each space is built using the data from the witness set,

according to the distances of each element to the pair of reference objects. These spaces may be efficiently searched using normal metric indexing techniques, and as the space is a genuine 2D space very efficient mechanisms such as the KD-Tree [3] can be used.

Each element of the data set is now considered as a query against each of these $\binom{n}{2}$ metric indexes, and the one with the least *local density* is selected to represent that element. There are various mechanisms for assessing local density, for example the smallest number of results for a threshold query, or the largest distance in the result set of a $k$NN query. We tested various ways over some different data sets and found relatively little difference in the cost or outcome, and settled on the strategy of picking the pair which gave the largest distance to the third-nearest 2D point.

While this mechanism is effective, it is of course extremely expensive, with a quadratic cost according to the number of reference points. In general, for high-dimensional queries, a relatively large number of reference points will be required. We discuss linear geometric approximations in Section 7.

### 5.2 Query

Having selected the most promising pair of reference points for each element $s_k \in S$, it is now represented as a tuple $\langle i, j, x, y \rangle$ where $i$ and $j$ are the identifiers of the reference points, and $x$ and $y$ are the 2D coordinates of the point where these reference points cause $s_k$ to be projected onto the corresponding plane[5]. At query time, each distance $d(q, p_i)$ is first calculated; then for each tuple in the data, $i$ and $j$ are used to select the appropriate distances from which $x_q$ and $y_q$ can be calculated. Finally, the 2D Euclidean distance $\ell_2((x, y), (x_q, y_q))$ is calculated, which gives a lower bound to the distance $d(s_k, q)$ in the original space.

## 6 Initial Measurements

Table 1 shows the results of applying this strategy to the SISAP *colors* and *nasa* data sets [15]. The figures reported represent the proportion of the data set excluded when searched, using the reported technique, at each of the standard thresholds[6]. Note that the left-hand column reports the number of reference points used; while this represents of the number of distance calculations necessary, both per datum at build time and per query at query time, the number of available pairs is $\binom{n}{2}$ for $n$ reference points, thus ranging from 45 to 11,175.

It is immediately apparent that the proposed mechanism is very effective. With only 10 reference points, already 97% of *colors* and 99% of *nasa* is successfully excluded at the smallest threshold. To put this in context, the top two rows of the table give the exclusion rates reported in [12] for the Distal SAT operating with both normal metric and supermetric exclusion mechanisms. However,

[5] as this is marginally more efficient than storing the distances to $p_i$ and $p_j$

[6] *colors*: 0.052, 0.083, 0.131; *nasa*: 0.12, 0.285, 0.53

**Table 1.** Exclusion Rates for different numbers of reference points, taking the statistically best pair available. The top two rows give comparable figures for the Distal SAT index structure (see text).

| | colors | | | nasa | | |
|---|---|---|---|---|---|---|
| | $t_0$ | $t_1$ | $t_2$ | $t_0$ | $t_1$ | $t_2$ |
| DiSAT: 3pt | 0.960 | 0.910 | 0.805 | 0.985 | 0.941 | 0.824 |
| DiSAT: 4pt | 0.980 | 0.943 | 0.840 | 0.991 | 0.964 | 0.851 |
| no. of refs | | | | | | |
| 10 | 0.973 | 0.927 | 0.821 | 0.988 | 0.928 | 0.761 |
| 30 | 0.987 | 0.959 | 0.880 | 0.996 | 0.967 | 0.851 |
| 50 | 0.991 | 0.969 | 0.902 | 0.997 | 0.975 | 0.872 |
| 70 | 0.993 | 0.974 | 0.912 | 0.998 | 0.981 | 0.894 |
| 90 | 0.994 | 0.977 | 0.918 | 0.998 | 0.984 | 0.903 |
| 110 | 0.995 | 0.979 | 0.924 | 0.999 | 0.986 | 0.910 |
| 130 | 0.995 | 0.981 | 0.929 | 0.999 | 0.987 | 0.915 |
| 150 | 0.996 | 0.982 | 0.932 | 0.999 | 0.988 | 0.920 |

even although much better exclusion rates are achieved here, the mechanism is explicitly sequential.

Apparently, the value of the mechanism goes on increasing as the number of reference points is increased, with what appears to be a slow asymptotic approach towards perfect exclusion.

### 6.1 Build Cost

The dominant cost is in searching the 2D pair space at build time; the tables show results up to 150 reference points which of course also requires 150 distance calculations per datum. However these distance calculations are likely to be amortised within another search mechanism as shown in Section 8.

The cost of searching the pair space however increases quadratically with the number of reference points, making it infeasible for larger numbers. This cost is almost independent of the cost of distance calculations or size of data in the metric space: the cost of searching $\binom{n}{2}$ 2D spaces becoming quickly predominant as $n$ increases. The cost is perfectly quadratic, in our experiments we have measured the cost $C(n) = 0.007n^2$ milliseconds for $n$ pivots; even with only 150 reference points this is approaching $0.2s$ per datum. In the context of searching a very large, high-dimensional, data set, then thousands of extra distance calculations are unlikely to be significant, but this would result in a huge potential space of reference point pairs that is intractable to search.

This leaves an interesting problem. The number of reference points does not typically constitute a performance problem in terms of distance calculations; the large cost is in the exhaustive search for the best pair of points. The reason the cost is high is because there are a huge number of potential pairs, which is the reason the mechanisms works so well. We have shown tremendous potential

when the best pair of points is calculated from the very large number of pairs available. If we can find a way of finding these cheaply, ideally in a manner that scales linearly rather than quadratically with the number of reference points, the mechanism should become even more useful.

In the context of searching a very large, high-dimensional, data set, then thousands of extra distance calculations are unlikely to be significant, but this would result in a huge potential space of reference point pairs that is intractable to search; thus we seek linear-scaling solutions using geometric analysis instead. For once, it is not reasonable to assume an arbitrary amount of pre-processing time is acceptable in order to achieve a small improvement in query time.

## 7 Geometric Approach

A number of intuitively-derived methods for the selection of first and second reference points for were tested. In all cases, sets of 10, 50, 150 and 500 objects were chosen to act as reference points, and these were scanned linearly in two passes according to the following strategies. The intent is to find a strategy that gradually improves with respect to the number of reference points, but where the construction cost remains linear.

The strategies used for each of two linear-cost scans were as follows:

1. random, to act as a benchmark
2. for each data point, associate the closest reference point
3. for each data point, associate the farthest reference point
4. for each of the $n$ reference points, associate it with the $\frac{1}{n}$ closest subset of the data (and do not consider these data points again)
5. for each of the $n$ reference points, associate it with the $\frac{1}{n}$ farthest subset of the data (and do not consider these data points again)
6. having selected a first reference point, choose the second to minimise the altitude (Y-coordinate) of the plotted 2D apex point
7. having selected a first reference point, choose the second to minimise the horizontal displacement (X-coordinate) of the plotted 2D apex point

The first five strategies were tried for each of first and second reference point choice, whereas the last two were used only for the choice of the second point; thus a total of 35 different strategies were tested.

Methods (2) and (3) in any combination proved no better than random, and actually became slightly worse with a larger number of reference points; we believe this is because of non-uniformity within the sets and the presence of outliers in the reference points. This problem was fixed by use of methods (4) and (5), where the closest or farthest $\frac{1}{n}$ of the data is associated with each reference point.

Table 2 shows a few of the results. The first row shows a purely random choice for comparison. The second shows method (4) used for the first point, and method (6) for the second. Finally the third row shows the use of method (4) for the first point and method (5) for the second, which gives the best compromise

**Table 2.** Results shown only for the lowest threshold of the *colors* data set, other results are consistent. We give the build cost (msec per object) and exclusion rate for some of the strategies tested.

| Pivot strategy | | | Number of Pivots | | | |
|---|---|---|---|---|---|---|
| First | Second | | 10 | 50 | 150 | 500 |
| random | random | build cost | 0.0008 | 0.0008 | 0.0010 | 0.0016 |
| | | exclusion | 0.929 | 0.925 | 0.926 | 0.924 |
| low dist | low alt | build cost | 0.014 | 0.022 | 0.045 | 0.124 |
| | | exclusion | 0.930 | 0.958 | 0.967 | 0.966 |
| low dist | high dist | build cost | 0.023 | 0.030 | 0.045 | 0.089 |
| | | exclusion | 0.946 | 0.962 | 0.971 | 0.973 |

for these data sets and thresholds. The final effect of achieving 97% exclusion – as much as is achieved by a very sophisticated indexing structure over the full data set – through a linear cost construction of a 10-byte data representation is really a significant achievement. Note that in the cost comparisons, the "random" benchmark cost is effectively zero; at 500 pivots the cost of either mechanisms is restricted to around $0.1ms$ per datum independent of the size of the data set, when the thorough search described in Section 5.1 would have cost $1.75s$.

## 8 Incorporation within List of Clusters

Finally, we report results where our mechanism is incorporated with another, scalable, indexing mechanism. We have chosen a well-known indexing structure, and give a very simple technique which extends this using the four-point exclusion mechanism as a post-filter. That is, the mechanism is embedded within the original structure to act as an internal filter, avoiding the calculation of original-space distances where the lower-bound calculation makes this unnecessary.

For this purpose we choose the List of Clusters [8], generally regarded as the most scalable mechanism known. We have measured this, with and without our optimisation, over the SISAP benchmark data sets *colors* and *nasa*, to perform theshold search using the three standard benchmark thresholds; we show a very significant improvement in performance.

As the list of clusters is built, at each node a pivot point is selected and a fixed number of objects, those being closest to this pivot point, are stored in an associated "bucket". Especially towards the start of this process, the cover radius of these objects from the pivot point is likely to be very small, therefore maximising the probability of the bucket being excluded from a search. When each cluster is constructed, the distances between every object in that cluster, and every pivot point from the root to that point in the list, will have been to be calculated as a part of the construction algorithm.

To this structure, we add only our small representations of the objects within each bucket, and cause no extra distance calculations at either build or query

time. The local pivot point is used as the first reference point, and the furthest pivot from the so-far constructed spine of the tree as the second. This gives an approximation to the geometric technique (low dist, high dist) described in Section 7, and the only extra construction-time cost is the calculation of the 2D coordinate from these distances; in experiments, this was literally undetectable. The extra space cost is 10 bytes per object, for the *colors* data set representing an increase of around 1%.

At query time, the mechanism is used in the normal way based on the measured distance between the query and each pivot point down the spine of the list. In cases where the local "cluster" requires to be searched, then the four-point representations are first checked. The four-point representation of the query requires only the calculation of the 2D representative point, as all of the distances required have already been measured as the query algorithm progresses down the spine of the list. The lower-bound computation then comprises a 2-dimensional $\ell_2$ distance. If the lower-bound distance is greater than the query threshold, there is no requirement to access the corresponding object and check its true distance against the query object. This saves not only an expensive distance calculation, but also the movement of the object within memory.

### 8.1 Experimental Results

Table 3 shows the number of distance calculations made against the original data sets, along with the percentage improvement shown; the same values are plotted in Figure 4. It can be seen in almost all cases that the query cost is better than halved, in return for only a small increase in memory size.

**Table 3.** Improvement shown on List of Clusters using four-point post-filtering. Values given are mean number of distance calculations per query.

| | standard | | | optimised | | |
|---|---|---|---|---|---|---|
| threshold | $t_0$ | $t_1$ | $t_2$ | $t_0$ | $t_1$ | $t_2$ |
| SISAP colors | 5645 | 11649 | 24401 | 2256 | 3987 | 10402 |
| SISAP nasa | 1381 | 3258 | 8790 | 1007 | 1402 | 3384 |

## 9 Conclusions and Future Work

We have shown how the four-point property can be used in conjunction with the concept of a pivot table in order to produce a minimally-sized table comprising only two reference objects identifiers, and two distances, per database object. These are used to construct a coordinate in a two-dimensional Euclidean space which gives a lower-bound on a query distance. The combination of the very large space of object pairs available from a relatively small set of reference objects,

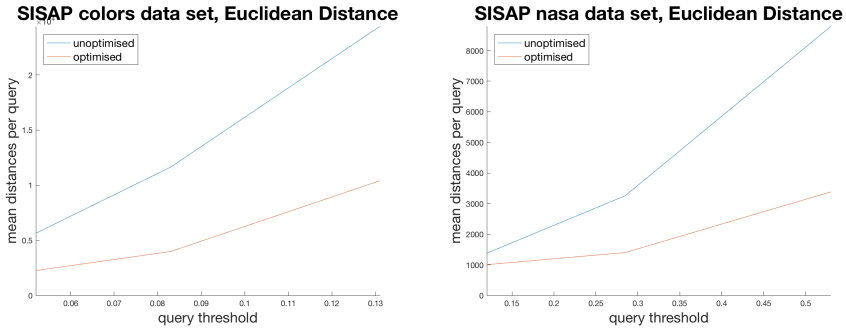


**Fig. 4.** SISAP benchmark space results with and without optimisation

and the observation that each pair gives a significantly different projection of the space, combines to allow a very high rate of successful exclusion for a typical range search, with exclusion rates of 99.6 and 99.9% obtained for the SISAP benchmark *colors* and *nasa* data sets, with only 150 reference objects being used. For a data size of around 10 bytes per object and a cheap arithmetic check these results are impressive.

It is remarkable that a random selection of pairs of reference points produce exclusion rates quite close to the more expensive exhaustive search. Other linear cost pair selection heuristics are closer to the ground truth. There is room for trying to match the almost perfect exclusion rate with other heuristics.

Finally, since it is theoretically impossible to avoid a sequential scan for nearest neighbour search, even in the approximate sense, a cheap exclusion mechanism that is trivially parallelizable is competitive. We remark that this mechanism can be used in conjunction with probabilistic methods requiring post-filtering or re-ranking, like metric inverted files. We have given one successful example of this: for an almost immeasurably small increase in build cost and memory, the performance of the List of Clusters indexing structure has been shown to be radically improved. It is likely that many similar examples exist.

## References

1. Giuseppe Amato, Claudio Gennaro, and Pasquale Savino. Mi-file: using inverted files for scalable approximate similarity search. *Multimedia tools and applications*, 71(3):1333–1362, 2014.
2. Ricardo Baeza-Yates, Walter Cunto, Udi Manber, and Sun Wu. Proximity matching using fixed-queries trees. In *Annual Symposium on Combinatorial Pattern Matching*, pages 198–212. Springer, 1994.
3. Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.
4. Leonard M. Blumenthal. A note on the four-point property. *Bulletin of the American Mathematical Society*, 39(6):423–426, 1933.

5. Walter A. Burkhard and Robert M. Keller. Some approaches to best-match file searching. *Communications of the ACM*, 16(4):230–236, 1973.
6. Cengiz Celik. Priority vantage points structures for similarity queries in metric spaces. In *EurAsia-ICT 2002: Information and Communication Technology*, pages 256–263. Springer, 2002.
7. Edgar Chávez, José L Marroquín, and Ricardo Baeza-Yates. Spaghettis: An array based algorithm for similarity queries in metric spaces. In *String Processing and Information Retrieval Symposium, 1999 and International Workshop on Groupware*, pages 38–46. IEEE, 1999.
8. Edgar Chávez and Gonzalo Navarro. A compact space decomposition for effective metric indexing. *Pattern Recognition Letters*, 26(9):1363–1376, 2005.
9. Edgar Chavez, Ubaldo Ruiz, and Eric Tellez. Cda: Succinct spaghetti. In *International Conference on Similarity Search and Applications*, pages 54–64. Springer, 2015.
10. Richard Connor, Franco Alberto Cardillo, Lucia Vadicamo, and Fausto Rabitti. Hilbert Exclusion: Improved metric search through finite isometric embeddings. *ACM Transactions on Information Systems*, 35(3):17:1–17:27, December 2016.
11. Richard Connor, Lucia Vadicamo, Franco Alberto Cardillo, and Fausto Rabitti. Supermetric search with the four-point property. In *International Conference on Similarity Search and Applications*, pages 51–64. Springer, 2016.
12. Richard Connor, Lucia Vadicamo, Franco Alberto Cardillo, and Fausto Rabitti. Supermetric search. *Information Systems*, 2018.
13. Richard Connor, Lucia Vadicamo, and Fausto Rabitti. High-dimensional simplexes for supermetric search. In *International Conference on Similarity Search and Applications*, pages 96–109. Springer, 2017.
14. Karina Figueroa, Edgar Chávez, Gonzalo Navarro, and Rodrigo Paredes. Speeding up spatial approximation search in metric spaces. *Journal of Experimental Algorithmics (JEA)*, 14:6, 2009.
15. Karina Figueroa, Gonzalo Navarro, and Edgar Chávez. Metric spaces library. *Online http://www. sisap. org*, 2007.
16. K. Menger. Untersuchungen ber allgemeine metrik. *Mathematische Annalen*, 100:75–163, 1928.
17. María Luisa Micó, José Oncina, and Enrique Vidal. A new version of the nearest-neighbour approximating and eliminating search algorithm (aesa) with linear preprocessing time and memory requirements. *Pattern Recognition Letters*, 15(1):9–17, 1994.
18. Aviad Rubinstein. Hardness of approximate nearest neighbor search. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1260–1268. ACM, 2018.
19. Guillermo Ruiz, Francisco Santoyo, Edgar Chávez, Karina Figueroa, and Eric Sadit Tellez. Extreme pivots for faster metric indexes. In *International Conference on Similarity Search and Applications*, pages 115–126. Springer, 2013.
20. Tomáš Skopal, Jaroslav Pokornỳ, and Václav Snášel. Nearest neighbours search using the pm-tree. In *International Conference on Database Systems for Advanced Applications*, pages 803–815. Springer, 2005.
21. Enrique Vidal. New formulation and improvements of the nearest-neighbour approximating and eliminating search algorithm (aesa). *Pattern Recognition Letters*, 15(1):1–7, 1994.
22. Wallace A Wilson. A relation between metric and euclidean spaces. *American Journal of Mathematics*, 54(3):505–517, 1932.