High Dimensional Search Using Polyhedral Query

Richard Connor, Stewart MacKenzie-Leigh, and Robert Moss

Department of Computer and Information Sciences,
University of Strathclyde, Glasgow, G1 1XH, United Kingdom
{richard.connor,s.mackenzie-leigh,robert.moss}@strath.ac.uk

Abstract. It is well known that, as the dimensionality of a metric space increases, metric search techniques become less effective and the cost of indexing mechanisms becomes greater than the saving they give. This is due to the so-called *curse of dimensionality.*

One effect of increasing dimensionality is that the ratio of unit hypersphere to unit hypercube volume decreases rapidly, making the solution to a similarity query (the query ball, or hypersphere) ever more difficult to identify by using metric invariants such as triangle inequality.

In this paper we take a different approach, by identifying points within a query polyhedron rather than a ball. We show how this can be achieved by constructing a surrogate metric space, such that a query ball in the surrogate space corresponds to a polyhedron in the original space. If the polyhedron contains the ball, the overall cost of the query is likely to be increased in high dimensions; however, we show that shrinking the polyhedron can capture a surprisingly high proportion of the points within the ball, whilst at the same time giving a more efficient, and more scalable, search.

We show results which confirm our underlying hypothesis. In some cases we can retrieve significant volumes of query results from spaces which are otherwise intractable.

1 Introduction

In this paper, we show a novel conceptualisation of an approximate indexing technique based on the geometry of high-dimensional metric spaces. This is based on the following observations:

1. the relationship between the shared volume of a hypersphere and a hypercube centred around the same point in high-dimensional space; especially that, as the side length of a containing hypercube is reduced, much of the hypersphere may still be contained
2. that points within an approximate hyper-polyhedron centred around an arbitrary query point can be defined, relying upon triangle inequality, by a set of inequalities based on distances from a fixed set of reference points[1]

[1] Corresponding to existing multiple-pivot indexing mechanisms.

A.J. Machado Traina et al. (Eds.): SISAP 2014, LNCS 8821, pp. 176–188, 2014.
DOI: 10.1007/978-3-319-11988-5_16 © Springer International Publishing Switzerland 2014

3. that a finite metric space can be re-indexed, using the Chebyshev distance over pre-calculated distances to the reference points, to allow the efficient extraction of points within a hyper-polyhedron centred around a query

The re-indexed (*surrogate*) space can often be queried more efficiently than the original: it may have smaller data points, a faster metric, and lower intrinsic dimensionality. However when queried at the same threshold, the result will be a much larger proper superset of the query in the original space, and the cost of filtering against the original space is likely to outweigh any efficiency gain.

However, observation (1) means that, as the surrogate query threshold is reduced, a corresponding increase in efficiency may be achieved without a corresponding significant loss in correct results. This makes the mechanism as proposed useful in the context of high-dimensional metric spaces, where known indexing techniques are completely ineffective. In this context, it may give a tractable and scalable approach to at least achieving some kind of imperfect search, and we show results from searching against GIST image characterisations [13] which we have been unable to otherwise achieve.

2 Dimensionality: Curse and Counter-Curse

In the domain of metric search, we are very familiar with the so-called *curse of dimensionality* [7]. The observable effect is that, as the dimensionality of a metric space increases, then for arbitrarily selected distances within the space standard deviation decreases and there are ever fewer very small values.

One explanation of this is that, as the dimensionality increases, the ratio of the volume of the unit hypersphere to the unit hypercube decreases rapidly. Points within the space fill a hypercube, while the solution to a threshold query fills a hypersphere (the query ball) centred around the query point.

It is instructive to observe the magnitude of this effect. The volume of a hypersphere with radius r in $2k$ dimensional space[2] is $\frac{\pi^k}{k!}r^{2k}$ which starts to decrease rapidly after 6 dimensions. The ratio of this volume to the volume of a containing hypercube is given by $\frac{\pi^k}{2^{2k}k!}$ which clearly becomes very small, very quickly, as k increases. At 6 dimensions the ratio is 0.08; at 10 it is 0.002 and it drops to 2×10^{-8} at 20. This sharp drop-off fits well with the generally known rule-of-thumb that metric indexing mechanisms become ineffective at an intrinsic dimensionality [7] of more than around 6-10.

2.1 Shrinking the Search Hypercube

Imagine that, for a threshold search, all points within the containing hypercube could be efficiently discovered. Even if this were true, it would be of little practical value in high dimensions, as in an evenly-distributed space almost all of the points contained would not be within the solution to the search.

[2] Even dimensions are used as the formula is slightly simpler, an equivalent formula exists for odd dimensions.

However, if for example all points within a hypercube of the *same volume* and centre as the query hypersphere could be discovered, these points will have a significant overlap with those in the query ball. As the number of dimensions increases, the overlap becomes proportionally smaller, but not rapidly.

The side length of this cube actually *decreases* as the dimensionality increases. Therefore, for example, in 10 dimensions a set of points which will coincide significantly with a query ball of radius t can be found by extracting points within a hypercube with a half-side length of just over $\frac{1}{2}t$, and around half of the space contained in this hypercube will also be in the hypersphere.

There is one further effect of which we can take advantage. As the half-side length increases up to the sphere radius, the contained volume of the sphere increases rapidly after a threshold of around the equivalent volume. It very slowly approaches full containment as the half-side length approaches the query radius, but includes almost all of the points within the sphere at a much smaller value than this. With higher dimensions, this effect is greater and starts at a lower threshold. Figure 1 shows the volume overlap in 10 dimensions, the graphs corresponding to recall and precision in an evenly distributed Euclidean space as the half-side length increases from 0 to 1. It can be seen that there is an overlap where both recall and precision are usefully far from zero: for example, if we could efficiently retrieve a hypercube with a half-side of little more than 0.6 of a query radius, we would retrieve over 80% of the true results, while of all the results retrieved, around 20% of them would be correct. In 10-dimensional space, this may well be a reasonable compromise if there is an associated reduction in query cost.

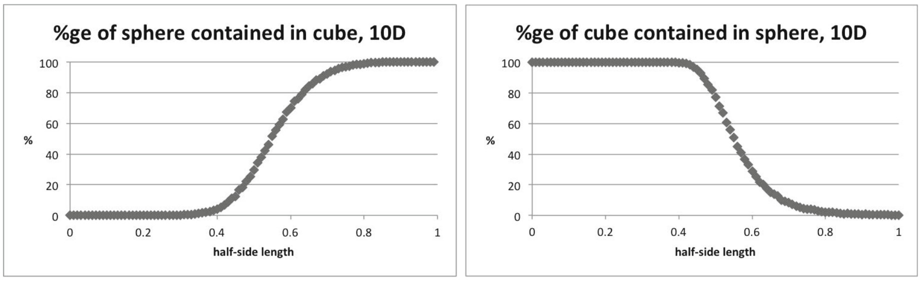

Fig. 1. With half-side ranging from 0 to 1, graphs show percentage of volumes: (a) of unit hypersphere in hypercube, and (b) of hypercube in unit hypersphere

These observations are used as follows. A surrogate metric space will be constructed such that a query ball in the surrogate space corresponds to a hyper-polyhedron within the original space. The hyper-polyhedron is expected to have similar volume-ratio properties as those determined for hypercubes. The surrogate space then allows points within an approximate hyper-polyhedron to be discovered using standard metric indexing techniques. Search at the same threshold corresponds to the minimum containing polyhedron, and will typically give a huge proportion of false positive results. However, efficiency gains from dropping the search threshold should maintain a high proportion of the true results because of the effect shown above.

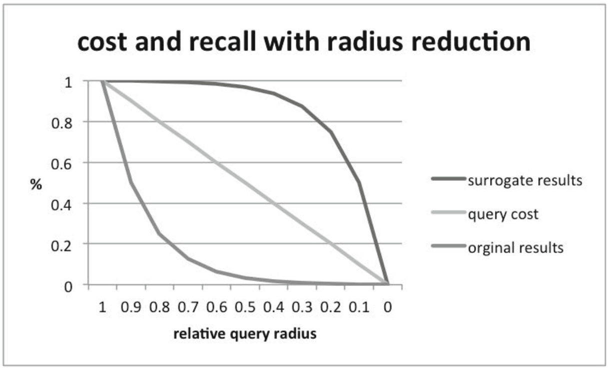

Fig. 2. Hypothesis: percentage of correct results with threshold reduction

In essence, we would hope to see a pattern as shown in Figure 2. As the query threshold is reduced, the query cost decreases approximately linearly in both original and surrogate spaces. In the original space, the amount of data returned will drop off very rapidly, due to the decrease in the hypersphere volume; however in the surrogate space, it will drop off at first very slowly, due to the effects just outlined. This should allow a high proportion of the correct results in return for a substantial reduction in query cost.

It is worth noting that, although the above discussion implicitly assumes a Euclidean space, the same patterns occur in other spaces as well, and the technique proposed works correctly over any metric space.

3 Defining Approximate Hyper-Polyhedra

There remains the issue of finding a scalable mechanism which will identify the points within the reduced hypercube. For a Euclidean space, this could be done by setting up an independent search structure for each dimension and finding the intersection of all points within the appropriate range on all dimensions; with unlimited parallel hardware this could be extremely efficient. However, our primary interests include performing search over high-dimensional, non-Euclidean metric spaces.

Instead of calculating the actual hypercube, we form an approximation of a hyper-polyhedron by use of reference points within the space. Figure 3 shows a simple example. Reference points p_1 and p_2 have been selected. For a query q, with threshold t, the property of triangle inequality means that for any u_i in the space, if $|d(q, p_1) - d(u_i, p_1)| > t$ or $|d(q, p_2) - d(u_i, p_2)| > t$, then u_i cannot be in the result set.

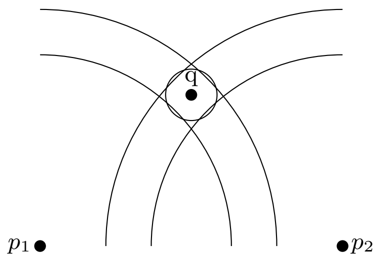

Fig. 3. An Approximate Hyper-Polyhedron in 2 dimensions

Generalising to a metric space $(\mathcal{S}, d)$, and a set of reference points $\mathcal{R}$, then an approximate hyper-polyhedron constructed for a query q with threshold t is defined as:

$$Poly(q) = \{u \leftarrow \mathcal{S} \,\text{where}\, \forall p \in \mathcal{R},\ |d(q,p) - d(u,p)| < t'\}$$

where $t' \leq t$, and is chosen according to the tradeoffs highlighted above.

This statement of inclusion essentially corresponds to the pivoting exclusion principle used in various multiple-pivot mechanisms, and is discussed further in Section 5. However we now show how to turn the pivot-based exclusion into a metric search in its own right. The value of doing this is that, as the value of t' is reduced, the scalability of the search increases.

4 Re-Indexing for Hyper-Polyhedral Search

If we use the denotation u^j to mean $d(u, p_j)$, then the inclusion criterion for $Poly(q)$ can be rewritten as $\max_j(|q^j - u^j|) \leq t'$, as for any $p_j \in \mathcal{R}$, $|q^j - u^j|) > t'$ means that u does not lie within the polyhedron around the query point.

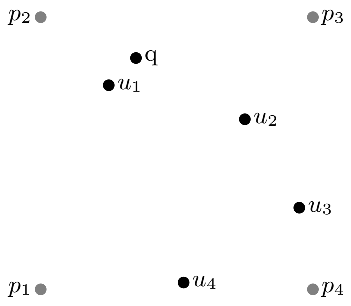

Fig. 4. Euclidean space: data u_i, reference points p_i and query q

This condition is captured by applying the Chebyshev distance (L_∞) to the ordered sets of values constructed from the distances of u and q to each reference point in turn. As Chebyshev is itself a proper metric, this means that elements of $Poly(q)$ can be found by using metric search over a metric space constructed from the original by pre-calculating these distances, and using Chebyshev as the distance metric.

Table 1. The left-hand column shows the Euclidean distances (d) from the query point in the original space (Figure 4). The right-hand column shows the corresponding Chebyshev distance (L_∞) in the surrogate space.

Original Space		Surrogate Space				
Point	$d(q,u)$	$d(p_1,u)$	$d(p_2,u)$	$d(p_3,u)$	$d(p_4,u)$	$L_\infty(q',u')$
q	0	3.68	1.52	2.67	4.28	0
u_1	0.57	3.16	1.41	3.14	4.24	0.52
u_2	1.84	3.91	3.35	1.80	2.69	1.83
u_3	3.26	3.98	4.72	2.81	1.22	3.20
u_4	3.37	2.10	4.43	4.34	1.90	2.91

Figure 4 demonstrates this by example, for data drawn in 2D Euclidean space. There are four reference points (p_i), four data points (u_i) and a single query point (q). Table 1 gives the corresponding distance values used to populate the surrogate space. It can be seen that, in all cases, the Chebyshev distance over the surrogate set gives a smaller value than the original distance, this property deriving from the triangle inequality property of the original space.

4.1 Formal Definition

Consider a metric space $(\mathcal{X}, d)$ over which a threshold search is required: that is, for some finite subset $\mathcal{S} = \{u_0, u_1, \ldots, u_n\}$ of $\mathcal{X}$, those objects within the close proximity of some $q \in \mathcal{X}$ require to be found. Note that $\mathcal{X}$ is not necessarily a Cartesian space, but d must be a proper metric.

Let $\mathcal{R}$ be an ordered set of m arbitrarily chosen points in $\mathcal{X}$, where r_j denotes the jth element of $\mathcal{R}$. $\mathcal{R}$ can be thought of as a set of *reference* points within the space.

A *surrogate* set $\mathcal{T}_\mathcal{R}$ of $\mathcal{S}$ is a set in m-dimensional Cartesian space where, for each $u_i \in \mathcal{S}$, there exists a corresponding $v_i \in \mathcal{T}_\mathcal{R}$ such that $v_i^j = d(u_i, r_j)$, where v_i^j denotes the value of the jth dimension of v_i.

The surrogate set $\mathcal{T}_\mathcal{R}$ will be used to perform queries without reference to either the actual values of $\mathcal{S}$, or the metric d. For a query q, a surrogate query $q_\mathcal{R}$ will be constructed, such that $q_\mathcal{R}^j = d(q, r_j)$.

The Chebyshev (L_∞) distance metric is defined as

$$L_\infty(x, y) = \lim_{n\to\infty} \sqrt[n]{\sum_j (|x^j - y^j|)^n}$$

which can be conveniently calculated as

$$L_\infty(x, y) = \max_j(|x^j - y^j|)$$

Being an element of the family of Lebesque metrics with $n \geq 0$, this is a proper metric. Therefore $(\mathcal{T}_\mathcal{R}, L_\infty)$ is also a metric space.

4.2 Properties

1. $L_\infty(q_\mathcal{R}, v_i) \leq d(q, u_i)$
   That is, if $\mathcal{Q}_t(q, \mathcal{S}, d)$ denotes the set of values returned by a threshold query for metric d over $\mathcal{S}$ for the point q and the threshold t, then $\mathcal{Q}_t(q, \mathcal{S}, d) \subseteq \mathcal{Q}_t(q_\mathcal{R}, \mathcal{T}_\mathcal{R}, L_\infty)$. The proof of this derives from the triangle inequality property of d.
2. The conditional probability of $u_i \in \mathcal{Q}_t(q, \mathcal{S}, d), v_i \in \mathcal{Q}_{t'}(q_\mathcal{R}, \mathcal{T}_\mathcal{R}, L_\infty)$ reduces at first, very slowly, from 1 as t' reduces from t downwards, while the cost of evaluating the query reduces more rapidly.

The following tradeoffs exist, according to a given search scenario:

1. Querying $L_\infty(q_\mathcal{R}, v_i)$ at the same threshold value t will always give a superset of the required results, which for a precise search will then have to be tested back in the original space before being returned as results of the threshold query. The relative sizes of the true and false results returned by a query at the same threshold depend upon the size, and individual points chosen, for the set $\mathcal{R}$; however, this aspect of query performance is likely to be in contention with choosing $\mathcal{R}$ to give the best search performance.
2. Conversely, although $L_\infty(q_\mathcal{T}, v_i) \leq d(q, u_i)$ is the only guarantee, it may be the case that, for some given ϵ, there is an acceptable probability that $L_\infty(q_\mathcal{T}, v_i) < d(q, u_i) - \epsilon$. If so, then a smaller threshold value can be used to produce an approximate result set. In many dimensions, it is extremely unlikely that a distance very close to the threshold will be reached in the surrogate space, as this can happen only with very close alignment of three points in the original space, which is increasingly less likely as the number of dimensions increases, although more likely as the size of $\mathcal{R}$ increases.

The same core mechanism can thus be used either as an accurate, or an approximate, threshold search, depending on the context of the requirements.

4.3 Choosing Reference Points

In common with other methods which use reference points, the choice of points appears to be critical to the performance of the mechanism. However, we are at an early stage of investigation in this respect.

We have tried various strategies for various spaces, and the only general deduction is that a random choice of points is relatively safe, as often the use of apparently appealing strategies only makes things worse.

For Euclidean space, it seems that the best strategy may be to choose artificial points in the "corners" of the space rather than points within the existing data. Thus in unitary space we use the origin, and then the points $(0, 1, 0, 0, \ldots)$, $(0, 1, 1, 0, \ldots)$, $(0, 1, 1, 1, \ldots)$ etc. We have not yet found equivalent series of points for other metrics, which are harder to reason about in terms of their multidimensional geometry, although we suspect they exist.

In terms of the number of reference points, we have seen some surprising results which show that many less points than might be expected can be used. This seems to depend very much upon the distribution of points within the space, and may be beyond theoretical analysis in an uneven distribution. There is however a clear law of diminishing returns: if each dimension in the surrogate space gives approximately a constant probability of excluding that point from the result set, and assuming this probability is reasonably large, then small numbers of reference points will be much more efficient than large numbers as better use of memory is made in the surrogate space. The tradeoff is that larger numbers of reference points will always give a smaller number of false positive returns, but the magnitude of this effect will depend heavily on how well the reference points can be chosen.

5 Related Work

There are already a large number of approximate methods suitable for use in higher-dimensional spaces, classified in [14], many of which use reference points.

Permutation indexing [2,4] is essentially another surrogate space mechanism. In common with our mechanism, a set of reference points is chosen and the distance to each is pre-calculated for all points in the data set. However these distances themselves are then abstracted into only their order from each point. Searching by these orders should be strongly correlated with a metric search, especially for nearest-neighbour searches. Many strategies have been suggested, with the best scaling being produced by using a relatively large number of reference points and then testing against only a much reduced view of these, allowing the resulting sparse space to be searched using inverted index techniques e.g. [1,12]. Our observation is that these techniques require more reference points, and give rather larger numbers of false positive results, than our technique, although the use of inverted indices can give impressive performance.

Re-indexing a space according to a proxy based on reference points was, to our knowledge, first suggested by Figueroa and Frediksson [10] in which a permutation space is re-indexed to give an improved metric performance.

As noted, our core semantics is identical to a set of multiple exclusions based on triangle inequality, and therefore relates closely to the pivoting exclusion principle used in various multiple-pivot mechanisms which use pre-computed distances, notably *LAESA* [15] and Extreme Pivots [16].

Our mechanism has much in common with Extreme Pivots, and in fact was derived from attempts to use this over GIST data. Perhaps because of the high dimensionality, or uneven distribution, we failed to find useful pivot groups as described in [16], and in the course of running experiments to find the best combination of pivots and pivot groups, we discovered that the best size of pivot group we could find was in fact 1. The results returned by our surrogate search are the same as those returned by a degenerate case of Extreme Pivots, using each reference point as one pivot group of a single value. However this allows the same search to be conducted in the surrogate space, rather than performing a serial scan of the data as is more generally required. The comparison of cost between our mechanism and Extreme Pivots is therefore only the increased efficiency of performing an indexed search over the space, versus a potentially greater number of false positive exclusions from using larger pivot groups. We believe our mechanism will therefore work better with larger, higher dimensional data. However, the observations based on the efficiency/recall tradeoffs via threshold reduction should apply equally to both mechanisms.

Since originally proposing this mechanism, we have discovered that exactly the same tradeoffs between efficiency and recall when reducing the search threshold within a mulitple-pivot space have been observed by Chávez and Nararro in [5]. Their explanation of the gains is based on the probability distribution function of distances within the original space, and is fully compatible with our observations on hyperspheres and hypercubes; the relationship seems worthy of further investigation. They do not propose reindexing the space using Chebyshev, which we believe gives much greater efficiency gains as the search threshold decreases.

In common with the motivation for the List of Clusters [6], the use of memory is critical in real index performance. This can be seen to be the reason for much of the performance gain we can achieve. As our indexing mechanism works over the surrogate space of the hyper-polyhedron, we can substantially reduce the query threshold without significantly reducing the number of results.

6 Results

For all experiments reported here, we have used the Euclidean "corners" strategy for Euclidean spaces, and randomly selected reference points for other metrics. We have results against three different types of data set: generated Cartesian spaces of various dimensions[3]; the SISAP *colors* data set, and a data set of GIST characterisations of images. This last set is the real target of the described mechanism: with 420 Cartesian dimensions, it is essentially intractable for metric indexing techniques.

[3] Not included due to space constraints, please contact the authors if interested.

In each case, we compare our technique using a balanced Vantage Point Tree (VPT) in both the original and surrogate spaces. This is just to give a point of reference against which two searches, the original and the surrogate, can be compared. It is quite likely that, for any given original or surrogate search, there are better indexing techniques available.

6.1 SISAP *colors*

The SISAP *colors* data was used with Euclidean, Cosine and SED [8, 9] distances, each of which has very different cost implications for both metric cost and scalability. 256 random points were removed from the 112,682 data points to use as queries. For each metric a query threshold sufficient to return around 1k results, i.e. mean of 4 per query, was used.

In each case, the surrogate space was searched and the results from these queries were then post-processed by comparing the original metric over the original data. These, and all other calculations, were performed on a non-optimised system, written in Java, and executed on a laptop computer, and so only the relative timings are important; each timed test was repeated until the standard error of the mean was less than 1%.

For Euclidean queries, only 5 reference points were used, this giving the best overall performance. This is many fewer than we would have expected, but as more points are used, only marginally better precision is achieved and the search cost is substantially increased. It is worth noting that each data point is therefore represented in less than one-twentieth of the memory required for query against the original 112-dimensional vectors. For Cosine and SED, the relatively high costs of the distance metrics themselves imply using larger number of reference points, to reduce the number of post-processing distances calculated.

Table 2 shows some key measurements for each metric. The figures given are: the cost of a sequential search; the cost using a standard balanced VPT, and the surrogate costs for retrieving 90%, 70% and 50% of the query results by reducing the query threshold. In all cases, achieving this through reducing the search threshold in the original space makes a negligible difference to cost. All costs are given in absolute time measured, to highlight the tradeoffs in the different metrics. The pattern of cost and recall as the surrogate threshold is decreased is shown in Figure 5.

The relative saving is quite complex, depending on a number of factors. For SED, the surrogate method is cheaper even to fetch 100% of the query results, as the number of results returned by the surrogate metric is less than the number of SED calculations performed in indexing via the VPT, and the cost of the metric makes this the dominant factor.

Depending on the context of the search, these speedups could already be quite useful. In all cases, however, the metrics over this space are already relatively tractable, with VPT indexing being substantially faster than sequential query; this is not the intended domain of our surrogate mechanism, and we turn our attention to a higher-dimensional space where this is not the case.

Table 2. Times (ms per 256 queries) for queries over the colors dataset

	Original Space		Surrogate Space		
Metric	Sequential	VPT-indexed	90% recall	70% recall	50% recall
Euclidean	1059	148	88	46	19
Cosine	9617	45	52	33	20
SED	79033	2849	1196	558	250

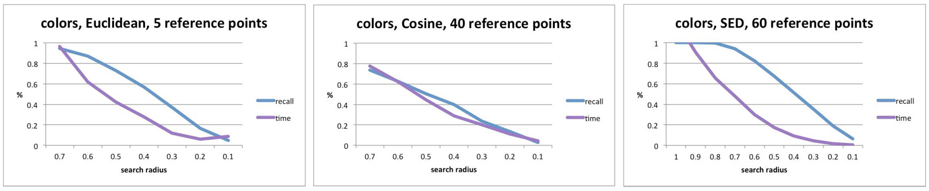

Fig. 5. Colors: three metrics, each showing cost reducing faster than recall. Time measured corresponds closely to memory use, which is optimised by choosing an appropriate number of reference points.

6.2 MirFlickr/GIST

The data used here comprises the GIST [13] characterisations of first 10k images taken from the Mir-Flickr collection [11]. A balanced VPT was built using the data, and then each value was queried against it at a threshold which returned 10k results (excluding the query itself), i.e. a mean of one per query. As would be expected with data of this complexity, the VPT gave little or no cost saving over sequential search for any metric.

Figure 6 shows the result of using polyhedral search for Euclidean and Cosine distances. The surrogate space was constructed, and a VPT used to query at different thresholds between the original, and one-tenth of the original, threshold. All values shown are relative to the cost of the original search.

Values shown are, from top to bottom in the graphs: *recall*, i.e. how many of the correct results are returned; *tree distances*, the relative number of distance calculations performed during the tree search; the actual *measured time* for the queries to complete; and the *memory use.* These last two figures include both the surrogate tree search, and the post-processing of the results using the original metric and data.

Notably for both searches even searching at the containment threshold is cheaper than the original metric search. This is because the cost is dominated by the memory cost of the original distance metric searching over the original points, each of which require 50-100 times more memory than the surrogate points. Even at the full threshold, a smaller number of results is obtained than the number of calculations performed during a tree search using the original metric and data.

For Euclidean search, the cost of retrieving 99.7% of the true results is just under half of the original, whereas 75% can be retrieved for just over one-fifth of the cost. The lack of a good strategy for choosing reference points means that more surrogate distance calculations are required for Cosine search, however this is more than compensated for by higher recall at lower relative thresholds, and the measured cost of retrieving 99.7% of the correct results is just over one-quarter of the cost of the original search, and 70% of correct results can be obtained for one-twentieth of the cost.

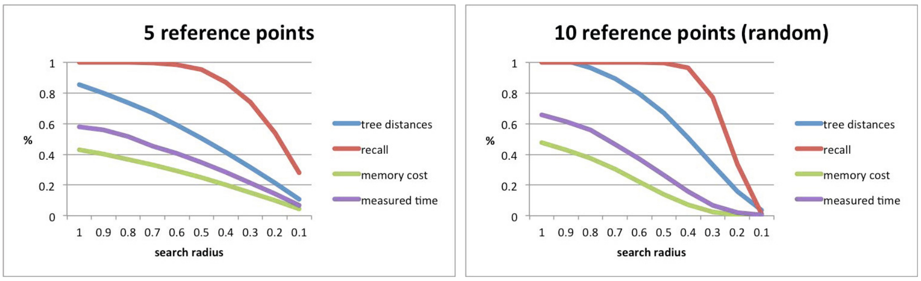

Fig. 6. Euclidean and Cosine distance over MirFlickr/GIST

7 Conclusions and Further Work

We have presented a novel strategy for approximate search in intractable, high-dimensional metric spaces. The essence of the mechanism is to re-cast the original space, via a set of reference points, into another metric space which can be usefully searched at lower thresholds. This allows, at least in some cases, a relatively predicable proportion of the correct results to be obtained for an acceptably low cost.

We are at an early stage of investigation of this technique, however we have already used it to obtain some real results that were previously unavailable to us in complex domains such as image similarity.

One area of investigation which could greatly improve the performance of the technique would be a better selection of reference points for non-Euclidean spaces, which may be possible to achieve by analysis of the geometry of these spaces as we believe we have achieved for Euclidean distance.

Finally, we have seen some interesting preliminary results from approximating nearest-neighbour (kNN) search in the surrogate space, for appropriately increased values of k. This takes advantage of an observation that there may be better correlation of the surrogate and original distances at lower threshold values, but requires further investigation.

References

1. Amato, G., Esuli, A., Falchi, F.: Pivot selection strategies for permutation-based similarity search. In: Brisaboa, et al. (eds.) [3], pp. 91–102
2. Amato, G., Savino, P.: Approximate similarity search in metric spaces using inverted files. In: Proceedings of the 3rd International Conference on Scalable Information Systems, InfoScale 2008, pp. 28:1–28:10. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), Brussels (2008)
3. Brisaboa, N., Pedreira, O., Zezula, P. (eds.): SISAP 2013. LNCS, vol. 8199. Springer, Heidelberg (2013)
4. Chávez, E., Figueroa, K., Navarro, G.: Effective proximity retrieval by ordering permutations. IEEE Trans. Pattern Anal. Mach. Intell. 30(9), 1647–1658 (2008)
5. Chávez, E., Navarro, G.: Probabilistic proximity search: Fighting the curse of dimensionality in metric spaces. Inf. Process. Lett. 85(1), 39–46 (2003)
6. Chávez, E., Navarro, G.: A compact space decomposition for effective metric indexing. Pattern Recognition Letters 26(9), 1363–1376 (2005)
7. Chávez, E., Navarro, G., Baeza-Yates, R., Marroquín, J.L.: Searching in metric spaces. ACM Comput. Surv. 33(3), 273–321 (2001)
8. Connor, R., Moss, R.: A multivariate correlation distance for vector spaces. In: Navarro, G., Pestov, V. (eds.) SISAP 2012. LNCS, vol. 7404, pp. 209–225. Springer, Heidelberg (2012)
9. Connor, R., Simeoni, F., Iakovos, M., Moss, R.: A bounded distance metric for comparing tree structure. Inf. Syst. 36(4), 748–764 (2011)
10. Figueroa, K., Fredriksson, K.: Speeding up permutation based indexing with indexing. In: Second International Workshop on Similarity Search and Applications, SISAP 2009, pp. 107–114 (August 2009)
11. Huiskes, M.J., Lew, M.S.: The mir flickr retrieval evaluation. In: MIR 2008: Proceedings of the 2008 ACM International Conference on Multimedia Information Retrieval. ACM, New York (2008)
12. Mohamed, H., Marchand-Maillet, S.: Quantized ranking for permutation-based indexing. In: Brisaboa, et al. (eds.) [3], pp. 103–114
13. Oliva, A., Torralba, A.: Modeling the shape of the scene: A holistic representation of the spatial envelope. Int. J. Comput. Vision 42(3), 145–175 (2001)
14. Patella, M., Ciaccia, P.: Approximate similarity search: A multi-faceted problem. J. of Discrete Algorithms 7(1), 36–48 (2009)
15. Ruiz, E.V.: An algorithm for finding nearest neighbours in (approximately) constant average time. Pattern Recognition Letters 4(3), 145–157 (1986)
16. Ruiz, G., Santoyo, F., Chávez, E., Figueroa, K., Tellez, E.S.: Extreme pivots for faster metric indexes. In: Brisaboa, N., Pedreira, O., Zezula, P. (eds.) SISAP 2013. LNCS, vol. 8199, pp. 115–126. Springer, Heidelberg (2013)