A New Approach to Indexing in High-Dimensional Space.

Conference Paper · January 1999

Source: DBLP

CITATIONS
0

READS
26

2 authors, including:

Tom Gedeon
Australian National University
430 PUBLICATIONS **5,943** CITATIONS

SEE PROFILE

A New Approach to Indexing in High-Dimensional Space

B. J. Briedis
T. D. Gedeon

School of Computer Science & Engineering
The University of New South Wales
Sydney NSW 2051
bbriedis@cse.unsw.edu.au
tom@cse.unsw.edu.au

Abstract. SVI is a promising new scheme for indexing high-dimensional points and vectors for use in vector retrieval and for finding the k-nearest neighbours. SVI performs an approximate search; that is, it trades off the completeness of the search for speed. The indexing scheme is built around a rule that was found by applying data mining techniques to sets of random vectors. This approach could well lead to further improvements in the indexing scheme.

1 Introduction

This paper introduces a promising new scheme for indexing high-dimensional points and vectors called *sub-vector indexing* (SVI). SVI is used to reduce the time expended in processes such as *vector retrieval* and finding the *k-nearest neighbours*. Also described is the approach taken in developing SVI, for it is believed that this approach may lead to further improvements in the indexing scheme. It must be noted that the search performed in SVI is incomplete (or *approximate*), for it generally fails to locate some of the desired items.

In describing the approach taken to the development of SVI it is assumed that there is a collection of items, and that queries may be made of the collection that result in some subset of the collection being returned. A scalar function is used to compare each query against each item in the collection in determining which items are to be retrieved. In the specific case of SVI, it is further assumed that the queries and the items in the collection may be represented using vectors.

Proceedings of the Tenth Australasian Database Conference, Auckland, New Zealand, January 18–21 1999

To date SVI has been tested for use in two types of system. In the first type, the queries and the items in the collection are represented using vectors with a common origin. The cosine of the angle between the two vectors is taken as a measure of their similarity. This measure is commonly used in vector retrieval systems (see for example [Salton, 1989]). In the second type, the queries and the items in the collection are represented using points, and the distance between them is used as a measure of their dissimilarity. This is the approach taken when finding the k-nearest neighbours. The distance used in this paper is the Euclidean distance.

The techniques that are currently used for indexing points and vectors are discussed in Section 2. Section 3 then presents the mechanics of SVI with various test results. Section 4 describes the development of SVI and analyses its operation. Section 5 then provides an outline of the work still in progress.

2 Existing Techniques

There has long been considerable interest in reducing the computation required in finding the k-nearest neighbours of a point, due in part to the wide-ranging applicability of the technique. The growing popularity of retrieval systems has helped to ensure that this interest has continued. In particular, the use of dimension reduction techniques has increased the importance of being able to find nearest neighbours quickly. Even text retrieval systems that incorporate a dimension reduction process [Deerwester et al., 1990] are now candidates for the use of fast nearest neighbour algorithms, so long as the large dimensionality of the vectors can be handled.

An overview of pre-1991 nearest neighbour algorithms are presented together with a number of the more significant papers in [Dasarathy, 1991]. Since then a considerable amount of work has been done on tree-based structures such as the R-tree [Guttman, 1984], TV-tree [Lin et al., 1994], SS-tree [White and Jain, 1996], SS+-tree[Kurniawati et al., 1997], and X-tree [Berchtold et al., 1996]. Some of these structures may be used for solving problems other than just that of finding nearest neighbours. The popular k-d tree [Friedman et al., 1977] also continues to be improved by many authors ([Grother et al., 1997], [Zakarauskas and Ozard, 1996], [Beis and Lowe, 1997], [Arya, 1995]).

It has generally been found that the available techniques have been of little benefit when the dimensionality is high. One possible solution to this problem is to search in parallel [Berchtold et al., 1997]. Another possible approach is to perform incomplete, or approximate, searching. An early paper that investigates this possibility is [Miclet and Dabouz, 1983], while more recent contributions include [Arya et al., 1998], [Kapoor and Smid, 1996] and [Nene and Nayar, 1997]. While these techniques have probably allowed for improved performance, the dimensionality of the points being indexed is still quite restricted. In [Arya et al., 1998] the authors state explicitly that their technique is limited to around 20 dimensions. Nene and Nayar's algorithm [Nene and Nayar, 1997] is also restricted to around 20 dimensions for random data, as is shown in their results. Most

authors are vague about the limits of their techniques as regards dimensionality, although few consider more than 20 dimensions. Some of the techniques have been to be found to be of some benefit in dimensionality as high as 40 for limited types of data sets ([Grother et al., 1997], [Nene and Nayar, 1997]). A number of new techniques are emerging, including those of [Berchtold et al., 1998] and [Weber et al., 1998]. SVI is yet to be compared to these new techniques.

3 Sub-Vector Indexing (SVI)

3.1 Constructing an Index

This subsection contains a description of the mechanics of SVI but does not attempt to explain the rationale underlying its operation. This discussion is deferred until Section 4. In order for SVI to be applied it is necessary that the queries and the objects being indexed may be represented using vectors. These vectors may be compared using a measure of similarity to result in what will be referred to as a *score*. Given a query, it is assumed that the items to be returned have a score greater than a specified *cutoff value*. The selection of the cutoff value is described in Section 3.2.

Firstly the queries and the items to be indexed are converted to *sign vectors*. The elements of the sign vectors are pluses and minuses, with each element being equal to the sign of the corresponding elements in the original vector. An example of this is given in the top part of Table 1. The original vector shown may represent either a query or an item to be indexed.

Next it is necessary to construct a set of sub-vectors from each sign vector. Each element of each new sub-vector is equal to an element from the original full-length sign vector. An example of how these sub-vectors are constructed is given in the bottom part of Table 1. Although the original vector here is only of length 8, in reality it could be much longer. Note that the sub-vectors do *not* need to consist of contiguous elements from the original vector. The indices used

Original vector	2	-1	-4	-3	4	6	-1	4
Full sign vector	+	−	−	−	+	+	−	+
Original indices	1	2	3	4	5	6	7	8
Sub-vectors								
Sub-vector 1			−	−	+	−		
Original indices			2	4	6	7		
Sub-vector 2			+	−	+	+		
Original indices			1	3	6	8		
Sub-vector 3			+	−	+	−		
Original indices			1	2	5	7		

Table 1. An example of how to construct a set of sub-vectors.

to construct the sub-vectors may be chosen at random or by using some more sophisticated method (see Section 5). The indices used will be the same for every original vector that is indexed. So for example, assuming the indices in Table 1 are used, sub-vector 1 of every query vector and of every vector representing an item in the collection will be constructed from elements 2, 4, 6 and 7 of the original vector.

The sub-vectors are used to predict whether a given pair of the original vectors are likely to have a score that exceeds the cutoff value. If a pair of the original full-length vectors have one or more sign sub-vectors that are constructed using the same indices which are identical, then it is comparatively likely that the original vector pair has a score that exceeds the cutoff. Those pairs of vectors with no identical sub-vector pairs are unlikely to exceed the cutoff. When a search is conducted, it is restricted to those items whose vectors have at least one sub-vector that is identical to the corresponding sub-vector of the query's.

The two major parameters that must be set when constructing an index are the number of sub-vectors to store per item in the collection (s) and the length of these sub-vectors (l). Simulations may be performed to decide upon these values. An example of such a simulation (for 100 dimensions, with s set to 100) is given in Table 2. Note that increasing the sub-vector length reduces the completeness of the search, but allows the search to proceed more quickly.

Each item is referenced to by s keys in the index. These keys are derived from the item's sub-vectors by performing a binary decoding, such as that depicted below:

$$\begin{array}{rc} \text{Sub-vector:} & - - + - - + - - + \\ & \Downarrow \\ \text{Binary vector:} & 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ 1 \\ & \Downarrow \\ \text{Key:} & 2^6 + 2^3 + 2^0 = 73 \end{array}$$

On making a query, s keys are created in a similar fashion. The index is then consulted, and the vectors that do not share any keys in common with the query are eliminated. The number of vectors that remain is likely to be large, although still much smaller than the entire collection. The remaining vectors are then tested against the query vector, and those whose scores exceed the cutoff are returned.

3.2 Calculating the Cutoff Value

In order to calculate the cutoff value it is necessary to decide upon the approximate number of items that are to be returned. This number may be represented as a percentage of the entire collection. Given the cumulative distribution function (CDF) of the scores that result when comparing vectors, it is possible to predict a cutoff that is likely to result in a retrieval list of a length similar to the desired length. The CDF is, of course, dependent on the probability density function (PDF) of the scores. It is possible to approximate the PDF using randomly generated vectors.

Sub-vector	Top items found (%)					Collection
length	0.0001%	0.0010%	0.0100%	0.1000%	1.0000%	searched (%)
3	100.0	100.0	100.0	100.0	100.0	100.0
4	100.0	100.0	100.0	100.0	100.0	98.8
5	100.0	100.0	100.0	99.9	99.7	90.3
6	100.0	99.9	99.6	98.9	97.2	71.0
7	99.4	98.7	97.0	94.3	88.4	48.0
8	98.1	95.1	89.8	83.5	72.9	29.0
9	90.0	86.6	77.6	67.7	54.5	16.3
10	80.6	74.3	62.8	50.7	37.5	8.7
11	66.0	61.1	47.2	35.6	24.3	4.6
12	53.0	46.6	33.5	23.8	15.1	2.3
13	41.0	33.3	22.7	15.3	9.1	1.2
14	30.0	21.8	14.9	9.6	5.4	0.6
15	21.9	15.2	9.6	6.0	3.1	0.3
16	13.1	10.4	6.2	3.7	1.8	0.2
17	10.6	7.4	4.0	2.2	1.0	0.1
18	7.5	5.2	2.5	1.3	0.6	0.04
19	4.4	3.2	1.6	0.8	0.3	0.02
20	3.1	2.1	1.0	0.5	0.2	0.01

Table 2. The central 5 columns give the percentage of the items with scores in the top 0.0001%, 0.001%, 0.01%, 0.1% and 1% that are found. The right-most column gives the percentage of the collection that needs to be searched. In this example the measure used is the cosine of the angle between the query vector and the vector representing a given item. The number of dimensions is 100, the collection size is 10^7 and the number of sub-vectors used for each item is 100.

For uniformly spread random vectors it was found that the PDF of the scores approximated the normal curve when the vectors were of high dimension. For lower dimensions, the PDF resembled a normal curve lacking its tails. When the cosine of the angle between two vectors was used as the score, the standard deviation was found to be approximately $1/\sqrt{d}$, where d was the dimensionality of the vectors. When the Euclidean distance was used, the standard deviation was approximately $1/(3\sqrt{d})$. In both cases the range of the scores was from -1 to 1. Unfortunately when the number of items to be retrieved is a very small percentage of the entire collection, the normal curve is somewhat inaccurate for deciding cutoff values. This is not surprising as the true PDF reaches probability 0 at 1 and -1, whereas the normal curve is asymptotic. It may be preferable to use cutoffs calculated from simulations on real or random data to taking values from the normal curve.

3.3 Complexity

Each item is indexed by s keys, and each key may take one of 2^l values, where l is the length of the sub-vectors. The simplest search strategy involves generating

s keys for a query, consulting the index for each key in turn to find those items which have the same key, and then comparing the item vectors found against the query vector. This results in a search time of $O(Nsd/2^l)$, where d is the number of dimensions of the original vectors and N is the number of items in the collection. The time overheads involved here are small, so $s/2^l$ should provide an accurate estimate of the search time as a proportion of the time taken to perform an exhaustive search. The memory requirement for the index is sN words, where each word is typically between about 6 and 14 bits. In addition, it is necessary to store the original vectors. The index creation and update times are quick. Each key on average refers to $N/2^l$ items, but there is no need to keep these sorted unless deletion needs to be particularly quick. Bearing this is mind, the update times are:

One insertion:	O(s)
One deletion (average):	O($sN/2^l$)
Full index creation:	O(Ns)

More complex indexing and search strategies are possible. The lower limit that may be achieved for search time is $Ndt(c,d)$ where c is the percentage of the collection that is considered to be relevant to the query and t is a function that returns the proportion of the collection that has a key in common with the query. Some values for t may be read from Figure 1 and from Table 2. One way of approaching this lower limit is to record those items that have already been compared to the query, so that future needless repeated comparisons may be avoided. In this case there are only $Nt(c,d)$ vector comparisons, although the record will need to be searched $N/2^l$ times. As dimensionality is high the former operation is likely to be more significant than the latter. Keeping such a record requires a memory overhead of $O(N)$, although index creation, insertion and deletion times are as before.

3.4 Results

SVI was tested on two types of measure: the Euclidean distance between two points, and the cosine of the angle between two vectors. The points and vectors were generated randomly using a uniform PDF. The vectors tested variously had 5, 10, 20, 50, 100, 200 and 500 dimensions. In each case, other than for 500 dimensions, 16 simulations were made of collections of size 10^7. In the case of 500 dimensions only 4 simulations were run due to time constraints. Figures were derived for sub-vector lengths ranging from 3 to 20 (where appropriate) and the cutoff values were variously defined so as to retrieve the top 0.0001%, 0.001%, 0.01%, 0.1% and 1% of the collection. In each case there were 100 sub-vectors per item, and the indices that were used in their creation were chosen at random. Table 2 gives the results for 100 dimensions, where the measure used is the cosine. It can be seen that the searches in the columns on the left are more successful than those on the right. The columns on the left correspond to cutoff values that are set very high and result in few items being returned. In

a retrieval system it is possible to make the search more restrictive when the system is large so consequently SVI will be most beneficial in large systems.

The six graphs in Figure 1 show the percentage of a collection that needs to be searched for a variety of dimensions, cutoff values and degrees of completeness. The trade-off between the completeness of the search and the search time can again be seen clearly. The trade-off becomes more severe as the dimensionality grows, and becomes less severe as the cutoff is made more restrictive. It is also the case that SVI almost always performs better when the cosine is used as the measure, rather that the Euclidean distance. This is more evident from the raw data than it is from Figure 1. The roughness of the curves is due to the small number of choices available when choosing the sub-vector length. It is to be noted that SVI may be useful even when the dimensionality is as large as 500, particularly when the cutoff is restrictive and the search does not need to be very accurate.

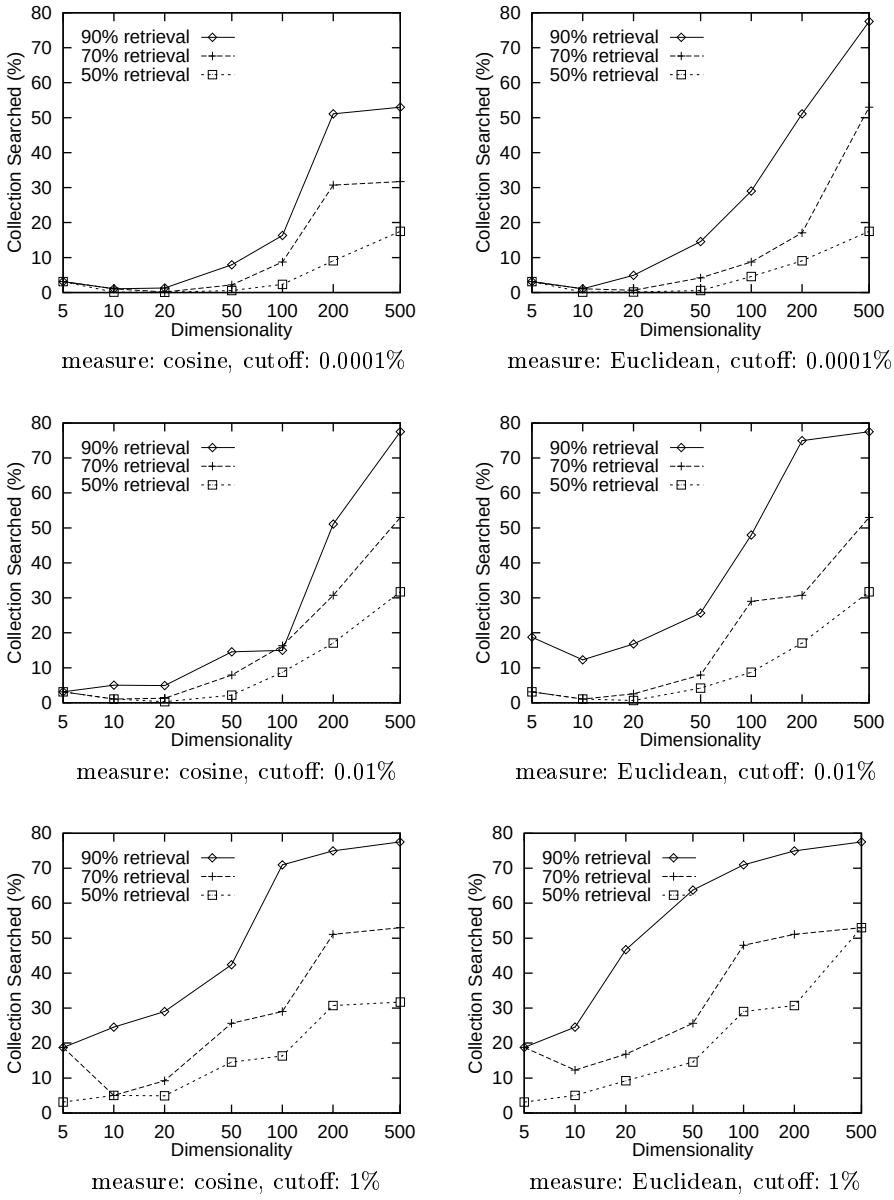
3.5 Applicability to Different Data Distributions

From the preceding discussion of SVI it might appear that SVI is only suitable for indexing vectors whose elements are uniformly distributed around zero. The conditions that actually do need to be met are somewhat less stringent than this. They are:

1. Each element must be independent of the others and
2. Each element must have a median value that divides the values the element takes into two equal parts.

It is easy to see that these conditions are sufficient. When a vector is indexed, each of its elements is compared to some *split point* using a simple greater or less than test. The values of the elements of the vectors are used in no other way. It is thus sufficient to merely consider the position and operation of the split points. The purpose of the sub-vectors is to divide the vectors into groups of equal size, and this occurs (on average) when the medians are used to split the ranges of the elements, provided the elements are independent of one another. For simplicity this paper has assumed the medians are all equal to zero, but different split points could easily be used if the data are not centred around zero. In a practical system it may be necessary to use estimates of the medians, which can be obtained by sampling the data.

These requirements allow SVI to be applied to a large number of data distributions, not just the uniform distribution. The requirement of independence suggests that SVI may be most suitable for indexing vectors that have undergone dimension reduction, as such vectors will generally have elements that are fairly independent of one another. The requirement imposed on the medians will normally exclude the indexing of sparse data, as such data has elements whose values cannot be divided into two even approximately equal parts.



measure: cosine, cutoff: 0.0001% measure: Euclidean, cutoff: 0.0001%

measure: cosine, cutoff: 0.01% measure: Euclidean, cutoff: 0.01%

measure: cosine, cutoff: 1% measure: Euclidean, cutoff: 1%

Fig. 1. Each of these graphs shows the percentage of the collection that needs to be searched in order to locate at least 90%, 70% and 50% of the items whose scores exceed the cutoff value for a range of dimensions. The cutoff values are set so as return the items with the highest 0.0001%, 0.01% and 1% scores respectively. The two measures tested are the the cosine of the angles between two vectors and the Euclidean distance between two points.

4 The Development of SVI

SVI was derived using a data mining-style process. A large set of pairs of vectors was randomly generated and for each pair a score was calculated. The vector pairs were divided into two sets according to whether or not they exceeded a cutoff value. A number of statistics that described the relationship between two vectors were tested to see how well they discriminated between the two sets of vector pairs. One important aspect of these statistics was that they had to lend themselves to being incorporated in a rule that could form the basis of an indexing scheme (we will call such a rule an *indexing rule*).

The most promising of the initial statistics tested was the count of the signs that matched in the two (full-length) vectors. The distribution of these counts varied greatly between the two sets of vector pairs. Figure 2 shows the distribution of these counts for 100 dimensions when the cosine is used as the measure of similarity. While this count provides good discrimination between the two types

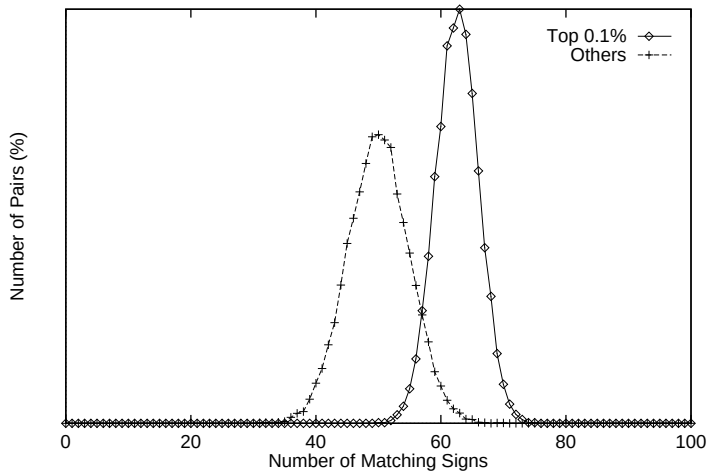


Fig. 2. The distribution shown represents the counts of the signs that match in random pairs of vectors. The distributions correspond to those pairs of vectors whose scores are in the top 0.1% of scores, and to those that are not. The measure of similarity used is the cosine of the angle between the vectors, and each vector has 100 dimensions. Both samples contain 10000 elements.

of vector pair, it does not directly lead to an indexing rule. It may, for example, be decided that it would be desirable to return vectors that have between 58 and 74 signs that match the query's. This, however, results in $\sum_{i=58}^{74} \binom{100}{i}$ possible sign vectors that would need to be searched.

The solution chosen to this problem treats the signs of the full vectors as being a population from which it is possible to draw samples. The probability

of a sign of a vector representing an item in a collection matching the sign of a query vector varies depending on whether the pair of vectors has a score that falls above or below the cutoff. Let the probability of a single sign matching be m. Assuming replacement, the likelihood of l matches is m^l. If s sub-vectors are chosen at random and they are assumed to be independent of one another, then the probability of there being at least one match is:

$$p = 1 - (1 - m^l)^s$$

l and s may be set so that the probability p is quite sensitive to the relevant variation in m. In practice the assumption of independence does not hold, but the equation does help to illustrate the principle behind SVI.

5 Work in Progress

The choice of the indices used to create the sub-vectors can obviously affect the performance of SVI. It is believed that randomly chosen indices will generally be close to optimal, although further reducing the interdependence of the sub-vectors could improve performance to some degree. Some initial testing has been done using a set of indices that has been created using a genetic algorithm, but so far the initial results have been inconclusive.

The authors are in the process of constructing a mathematical model of SVI to avoid the need for extensive simulations and to aid analysis. For the most recent work in this area, please contact the authors.

SVI still needs to be tested against a number of other techniques, particularly newer techniques such as [Weber et al., 1998] and [Berchtold et al., 1998]. On the face of the literature the older techniques do not make significant computational savings in many of the dimensions to which SVI may be applied.

Other types of statistical rule are still being considered, and these will be tested in a fashion similar to the method described in Section 4. One possibility is to replace the simple sign comparisons with more complicated range comparisons. To aid this process it would be useful to formally define the characteristics a statistic must have to produce an indexing rule.

6 Conclusion

SVI shows considerable promise as a technique for indexing in high dimensions; it is capable of saving considerable computation even in hundreds of dimensions. Unlike many other indexing techniques, there is a trade-off between the completeness of the search and the speed of the search. The technique is still in its infancy with many areas yet to be fully investigated. It is believed that the continued use of data mining techniques will lead to further improvements in the indexing scheme.

References

Arya, Sunil (1995). *Nearest Neighbour Searching and Applications.* PhD thesis, Computer Vision Laboratory, Center for Automation Research, University of Maryland, College Park, Maryland 20742-3275.

Arya, Sunil, Mount, David M., Netanyahu, Nathan S., Silverman, Ruth, and Yu, Angela Y. (1998). An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. *Journal of the ACM.* To be published.

Beis, Jeffrey S. and Lowe, David G. (1997). Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *Conference on Computer Vision and Pattern Recognition*, pages 1000–1006, Puerto Rico.

Berchtold, Stefan., Böhm, Christian, Braunmüller, Bernhard, Keim, Daniel A., and Kriegel, Hans-Peter (1997). Fast parallel similarity search in multimedia databases. *SIGMOD Record*, 26(2):1–12. Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data, Tucson, Arizona, May 13–15, 1997.

Berchtold, Stefan, Böhm, Christian, and Kriegel, Hans-Peter (1998). The pyramid-technique: Towards breaking the curse of dimensionality. *ACM SIGMOD Record*, 27(2):142–153. Proceedings of the ACM SIGMOD International Conference on Management of Data.

Berchtold, S., Keim, D., and Kriegel, H.-P. (1996). The X-tree: An index structure for high-dimensional data. In *22nd Conference on Very Large Databases*, pages 28–39, Bombay, India.

Dasarathy, Belur V. (1991). *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques.* IEEE Computer Society Press, Los Alamitos, California.

Deerwester, Scott, Dumais, Susan T., Furnas, George W., Landauer, Thomas K., and Harshman, Richard (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Friedman, Jerome H., Bently, Jon Louis, and Finkel, Raphael Ari (1977). An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3(3):209–226.

Grother, Patrick J., Candela, Gerald T., and Blue, James L. (1997). Fast implementations of nearest neighbor classifiers. *Pattern Recognition*, 30(3):459–465.

Guttman, A. (1984). R-trees: A dynamic index structure for spatial searching. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 47–57, Boston, MA.

Kapoor, Sanjiv and Smid, Michiel (1996). New techniques for exact and approximate dynamic closest-point problems. *SIAM Journal of Computing*, 25(4):775–796.

Kurniawati, R., Jin, J. S., and Shepherd, J. A. (1997). The SS+ -tree: An improved index structure for similarity searches in a high-dimensional feature space. In *SPIE Storage and Retrieval for Image and Video Databases V*, San Jose CA.

Lin, King-Ip, Jagadish, H. V., and Faloutsos, Christos (1994). The TV-tree: An index structure for high-dimensional data. *VLDB Journal*, 3(4):517–549.

Miclet, L. and Dabouz, M. (1983). Approximative fast nearest-neighbour recognition. *Pattern Recognition Letters*, 1:277–285.

Nene, Sameer A. and Nayar, Shree K. (1997). A simple algorithm for nearest neighbor search in high dimensions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(9):989–1003.

Salton, Gerard (1989). *Automatic Text Processing: The Transformation Analysis and Retrieval of Information by Computer*. Addison-Wesley Publishing Company, Reading, Massachusetts.

Weber, Roger, Schek, Hans-J., and Blott, Stephen (1998). A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *Proceedings of the 24th VLDB Conference*, New York, USA.

White, D. A. and Jain, R. (1996). Similarity indexing with the SS-tree. In *Proc. 12th IEEE International Conference on Data Engineering*, New Orleans, LA.

Zakarauskas, Pierre and Ozard, John M. (1996). Complexity analysis for partitioning nearest neighbor searching algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(6):663–668.