

# Using the triangle inequality to reduce the number of comparisons required for similarity-based retrieval \*

Julio Barros

University of Virginia, Los Alamos National Laboratory

James French, Worthy Martin

Computer Science Dept., University of Virginia  
Charlottesville, Va 22903

Patrick Kelly, Mike Cannon

Los Alamos National Laboratory, Los Alamos, NM 87545

## ABSTRACT

Dissimilarity measures, the basis of similarity-based retrieval, can be viewed as a distance and a similarity-based search as a nearest neighbor search. Though there has been extensive research on data structures and search methods to support nearest-neighbor searching, these indexing and dimension-reduction methods are generally not applicable to non-coordinate data and non-Euclidean distance measures.

In this paper we reexamine and extend previous work of other researchers on best match searching based on the triangle inequality. These methods can be used to organize both non-coordinate data and non-Euclidean metric similarity measures. The effectiveness of the indexes depends on the actual dimensionality of the feature set, data, and similarity metric used. We show that these methods provide significant performance improvements and may be of practical value in real-world databases.

**Keywords:** image database indexing, similarity-based retrieval, best match searching, triangle inequality, similarity measures

## 1 INTRODUCTION

Similarity-based indexing is becoming an important issue in modern database applications such as image databases.<sup>1</sup> In traditional databases, queries typically retrieve records based on precise straightforward requirements, such as a word or value search. In similarity-based indexing, items are retrieved based on

---

\*To appear in IS&T/SPIE - Storage and Retrieval for Still Image and Video Databases IV. 28 Jan - 2 Feb 1996.

their similarity to a query item. The similarity of two items is measured by a user-defined function which can be quite complicated and expensive to compute. A similarity function generally returns a single real-valued similarity score for each comparison between a query item and an item in the database. Given these values we would like to answer the following questions:

1. Which  $k$  items in a database are most similar to a query item?
2. Which items in a database meet some threshold of similarity to a query item?

In this paper we concentrate on methods the answer first question more efficiently, though this work also applies to the second. We do not consider methods to answer the more traditional question of “Is item  $i$  in the database?”.

If the similarity score is viewed as a distance, similarity-based queries can be viewed as nearest-neighbor (NN) searches. Technically, similarity scores increase and dissimilarity scores decrease as two items become more similar. Consequently, dissimilarity scores are more accurately viewed as distances. In practice however, many dissimilarity and similarity functions can be easily converted and substituted, and thus we use the terms interchangeably.

Nearest neighbor searches over a small number of database items are common in pattern classification applications. With a small number of items, or prototypes, exhaustive scan is feasible and adequate. However, as similarity functions become more complex and larger databases become more common, indexing support to avoid exhaustive scan becomes increasingly important. In previous work<sup>2</sup> we explored techniques for indexing sets of point data described by a probability density function and the tradeoffs between effectiveness and efficiency.<sup>3</sup>

There has been extensive research on data structures and search methods, such as R-trees,<sup>4</sup> K-D-trees,<sup>5</sup> binning,<sup>6,7</sup> projection,<sup>8</sup> and principle components analysis, to support nearest-neighbor searching.<sup>9</sup> However, these indexing and dimension-reduction methods are not applicable to non-coordinate data and non-Euclidean distance measures. Other researchers have developed indexing methods based on the triangle inequality. These methods can be used to organize both non-coordinate data and non-Euclidean metric distance measures.

This paper extends the work of Burkhard and Keller<sup>10</sup> and Shapiro<sup>11</sup> on best match searching based on the triangle inequality and addresses some interesting questions arising from their methods. Section 2 presents an explanation and the background of the method. Section 3 discusses factors that affect the performance of the method. Section 4 discusses several empirical results and section 5 presents our conclusions.

## 2 BACKGROUND

To answer a similarity-based query, it is sufficient to exhaustively search through the database comparing each item to the query item. This approach requires one similarity computation between the query item and each database item. That similarity computation is often very expensive. Our goal is to avoid retrieving and examining every item. This paper examines the use of reference items in an effort to meet that goal. Burkhard and Keller<sup>10</sup> introduced the notion of using the distance between database items and a reference item to organize the database. Shapiro<sup>11</sup> later extended that work to use multiple reference items. The basic procedure is described below.

We wish to find the best match for a query item  $q$  from a database of  $n$  items  $X = \{x_1, \dots, x_n\}$ . The best match is defined in terms of the (dis)similarity function  $d(x, y)$ . For this approach, the similarity function must be a metric. Though this requirement excludes some applications, it is acceptable to others. For a discussion on the metric properties of some popular functions, see Gower.<sup>12</sup> A metric is a real-valued non-negative measure for which the following additional three properties hold for all  $x, y$ , and  $z$ .

$$d(x, y) = 0 \text{ implies } x = y \tag{1}$$

$$d(x, y) = d(y, x) \tag{2}$$

$$d(x, z) \leq d(x, y) + d(y, z) \tag{3}$$

The third property is commonly referred to as the triangle inequality. The Burkhard and Keller method relies on the triangle inequality to exclude from consideration items that could not possibly be nearest neighbors to the query item.

During the search for the best match to query item  $q$ , let  $b$  represent the identifier of the best item found so far. That is,  $b$  minimizes  $d(q, x)$  for all  $x$  considered so far. Let  $\xi = d(q, b)$  be the minimum distance between the query item and any item considered so far. The function *update*( $x$ ) maintains the value of  $b$  and  $\xi$  as follows. If  $d(q, x) < \xi$ , then  $b \leftarrow x$  and  $\xi \leftarrow d(q, x)$ , else do nothing. Each call to *update* requires a item-to-item similarity computation and is the only time that a complete similarity computation is done. We need not update with an item  $x$  if it is known *a priori* that  $d(q, x) \geq \xi$ .

Let  $r \in X$  be the reference item. From the triangle inequality  $d(q, x) + d(x, r) \geq d(q, r)$  or  $d(q, x) \geq d(q, r) - d(x, r)$ . This shows that if  $d(q, r) - d(x, r) > \xi$  we do not need to consider this  $x$  further. This is known as the *first cutoff criterion*. It excludes items that are too close to the reference item to be closer than  $\xi$  to  $q$ . Analogously,  $d(q, x) + d(q, r) \geq d(x, r)$  or  $d(q, x) \geq d(x, r) - d(q, r)$ . This shows that if  $d(x, r) - d(q, r) > \xi$  we do not need to consider this  $x$  further. This is known as the *second cutoff criterion*. It excludes items that are too far from the reference item to be closer than  $\xi$  to  $q$ . The two cutoff criteria can be combined into the *joint cutoff criterion*.

$$|d(x, r) - d(q, r)| > \xi \tag{4}$$

If the joint cutoff criterion is true for any  $x$ , we can be certain that  $x$  is not the best match for  $q$  and can safely and immediately eliminate  $x$  from further consideration.

The searching process, shown in Figure 1, is divided into two stages. The preprocessing stage is done once before any queries are processed. The steps of the query stage are done for each query. When this algorithm terminates,  $b$  will indicate the most similar item and  $\xi$  will be the similarity between it and the query item  $q$ . This algorithm terminates with the correct answer by working its way through items for which  $d(x, r) \approx d(q, r)$  and updating the values of  $b$  and  $\xi$ . As it proceeds, the distance between the query item and the current item under consideration,  $|d(x, r) - d(q, r)|$ , only increases and the distance to the best known item,  $\xi$ , only decreases. Eventually no remaining  $x$  will meet the joint cutoff criteria and the search terminates.

For example, suppose we had items  $r, x_1, x_2$ , and  $x_3$  as in Figure 2. For discussion purposes let's assume that these items are one-dimensional items and that Euclidean distance is our similarity measure. Let's also assume that we do *not* wish to use our knowledge of the one-dimensional values to index the items.

```

Preprocessing stage
  Select a reference item  $r$ .
  Calculate  $d(r, x)$  for all  $x \in X$  and maintain in sorted order.
Query stage
  Calculate  $d(r, q)$ .
  Locate item  $x \in X$  with  $d(r, x)$  closest to  $d(r, q)$  with binary search.
  While  $x$  meets joint cutoff criterion -
    Update with  $x$  -
      If  $d(q, x) < \xi$ 
        then  $b \leftarrow x$ 
         $\xi \leftarrow d(q, x)$ 
      else
        do nothing
  Locate new item  $x$  with next closest  $d(r, x)$  to  $d(r, q)$ .

```

Figure 1: Search algorithm

We choose  $r$  as our reference item and then calculate the dissimilarity values  $d(r, x_1)$ ,  $d(r, x_2)$ , and  $d(r, x_3)$  which are kept in sorted order. When query item  $q$  is presented, the value  $d(r, q)$  is calculated and  $b$  and  $\xi$  are initialized to  $r$  and  $d(r, q)$  respectively. The index is then searched for the distance value closest to  $d(r, q)$ . The value  $d(r, x_1)$  is found and a call is made to *update*( $x_1$ ). The value  $d(q, x_1)$  is calculated and then, since it is greater than  $\xi$ , it is discarded. The index is searched for the next closest value to  $d(r, q)$ ,  $d(r, x_2)$  is found, and a call to *update*( $x_2$ ) is made. During the update,  $d(q, x_2)$  is found to be smaller than  $\xi$  so  $b$  and  $\xi$  are updated. The index is then searched for the next closest value and  $d(r, x_3)$  is found. This time however, it is noted that  $|d(r, x_3) - d(r, q)| > \xi$  implying that  $d(q, x_3) > \xi$  (see Equation 4). Therefore we do not need to finish this similarity computation and our search is finished. The search process in this example required 3 similarity computations rather than the 4 similarity computations a sequential search would have required.

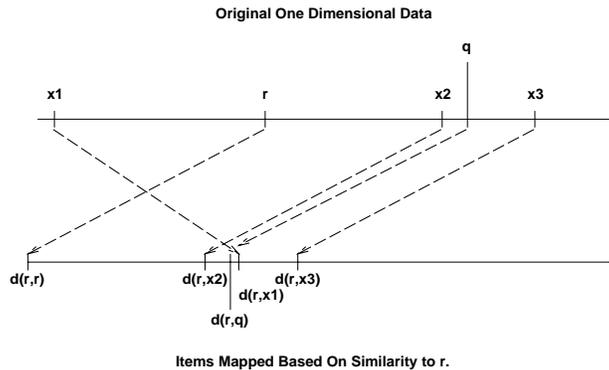


Figure 2: Search example on one-dimensional data

Sorting and searching the index requires  $O(n \lg n)$  and  $O(\lg n)$  *floating point* comparisons. In practice however, similarity-based searching is dominated by the cost of the item-to-item *similarity computations*. We therefore choose to focus on minimizing the number of item-to-item similarity computations and accept the cost of sorting and searching of floating point values. To build the index one item-to-item similarity

computation between each item in the database and the reference item is required during the preprocessing stage. The query stage requires one similarity computation between the query and the reference item as well as an unknown number of additional similarity computations between the query and database items. In the worst case there will be one additional similarity computation for each item in the database. In practice however, fewer additional similarity computations will be required and the cost of the preprocessing stage can be amortized over many queries.

## 2.1 Multiple reference items

Shapiro<sup>11</sup> shows that further reductions can be achieved by using more than one reference item. There have been several methods proposed<sup>10,13,14</sup> to organize a database when using multiple reference points. Initially they all seem very similar and further investigation of their performance and applicability to different situations may be warranted. The procedure we use was chosen due to the ease of inserting and deleting items. Though, ease of updates may not be important in a classification application, we believe it may be crucial in a database application. The procedure to use multiple reference items is very similar to the single reference item method. Let  $R = \{r_1, \dots, r_m\} \in X$  be the  $m$  reference items. For each  $r_i \in R$  and  $x_j \in X$ ,  $d(r_i, x_j)$  is calculated. The  $m$ -tuple of distances is maintained in sorted order based on the distances to a particular reference item. These tuples are searched based on this order as with the single reference item approach. However, an additional joint cutoff criterion is applied for each additional reference item before the item is considered for update. The database item under consideration must meet all  $m$  joint cutoff criteria to be considered further. Currently the reference item on which to base the search order is chosen arbitrarily. Developing a method to better choose this item is an goal for further research.

For example, Figure 3 shows the data of Figure 2 with two reference items  $r_1$  and  $r_2$ . The distance between each item and the reference items is calculated and kept in sorted order based on the distance to the reference item  $r_1$ . When the query item  $q$  is presented, the distances  $d(r_1, q)$  and  $d(r_2, q)$  are calculated and  $b$  and  $\xi$  are initialized. The index is then searched for the closest value and  $d(r_1, x_1)$  is found. However, it is noticed that  $|d(r_2, x_1) - d(r_2, q)| > \xi$  and  $x_1$  is immediately rejected. The index is searched again and  $d(r_1, x_2)$  is found and a call to  $update(x_2)$  is made. The search proceeds as usual with additional rejections being made based on the multiple joint cutoff criteria and reference items used.

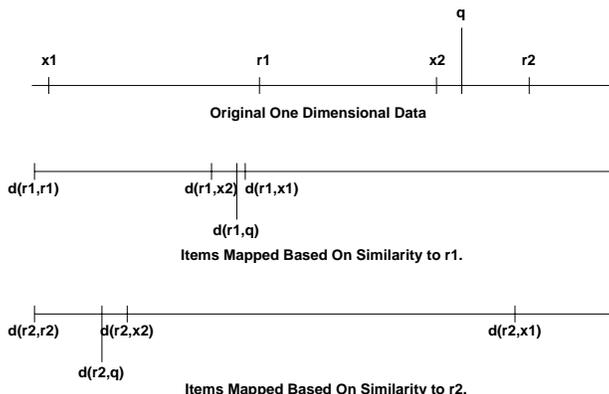


Figure 3: Search example on one-dimensional data with two reference items

To build the index, the multiple reference items approach requires one item-to-item similarity computation with each of the  $m$  reference items for each item in the database. This increases the initial pre-processing overhead but will yield fewer required similarity computations compared to the single reference item method.

### 3 PERFORMANCE OF THE REFERENCE ITEM METHOD

In this section, we begin to explore the performance of the single and multi-reference item methods. With no index, we must look through all  $N$  items in a database. With an index and  $m$  reference items, we can reduce that to  $\alpha * N$  on average where  $\alpha \leq 1$ . Understanding what affects  $\alpha$  and what the value of  $\alpha$  will be for a particular dataset and index is the goal of our present research.

In a sense, the reference item method maps all the items on the surface of a hyper-sphere to the same value on a number line. The search process examines all items mapped to the value of  $d(r, q) \pm \xi$  where  $\xi$  is the distance between  $d$  and its nearest neighbor. This range on the number line translates back to all database items in the volume between two hyper-spheres, or the shell, centered at the reference point. The number of items in that shell depends on several factors including: the particular reference items chosen; the number of reference items; the dimensionality of the dissimilarity space; the value of  $\xi$ ; the distribution of items in that space; and the density of that space.

The particular reference item chosen is important since it affects how many objects are likely to be in the shell of a particular query. For example, a reference item located in the center of a cluster of database items is likely to have many database items at approximately the same distance away. Any query point that maps to approximately this distance value will require a full similarity computation with all of these items. However, if the reference item is located on the periphery of the cluster, the database items will map to a larger range of values and the density of items along the number line (or in the shell) will be decreased and fewer complete similarity computations will be required. On the other hand, a reference item that is extremely far from the center of a cluster would tend to map all items in that cluster to similar values on the number line. This would have the affect of creating a dense range on the number line and including a large number of items in the shell.

Shapiro<sup>11</sup> experimented with data from known distributions that formed clusters and concluded that the reference items should be away from cluster centers. Selecting the reference points when the location or number of clusters is not known is not so clear. Ideally, we would want a reference item to provide high separability of items over the query distances that are likely to be encountered. We have conjectured that this may be achieved by choosing the database item with maximal variance in distance to other items. We expect to explore this idea further in future work. However, if this were true, it may be of limited practical utility, since estimating the variance of all items may require a large number of similarity computations which would make the pre-processing costs less attractive. If, however, large pre-processing costs are acceptable, Vidal<sup>15</sup> and Shasha<sup>16</sup> outline search methods that make use of available pre-computed distance information.

The multi-reference item approach attempts to reduce the number of similarity computations required by excluding items based on multiple joint cutoff criteria. Each reference item helps restrict the space that needs to be searched. This can be thought of as the intersection of multiple ranges on the number lines or of multiple shells in the dissimilarity space. When using multiple reference items, we would like to choose them to minimize the volume of the intersected shells. Again, it is not clear how to do this without nearly complete knowledge of all pairwise distances.

The dimensionality of the dissimilarity space also affects efficiency. The reference point method attempts to capitalize on the concept that the distance between the query item  $q$  and the reference item  $r$  will be similar to the distance between the answer item  $a$  and the reference item  $d(r, q) \approx d(r, a)$ . However, there is no guarantee that two items with similar distances to the reference item will have low distances between them. The higher the dimensionality of a similarity space, the less likely that two items with similar similarity values to a reference item are in fact similar to each other. For example, with a one-dimensional space, items on either side of the reference item will map to the same value on the number line; in a two-dimensional

space, items on a circle around the reference item will map to the same value; and in a three dimensional space, items on a sphere around the reference item will map to the same value. Consequently, the reference point method should be more efficient when searching a lower-dimensional similarity space as compared to a higher-dimensional space.

This brings up the interesting question of the dimensionality of similarity space if the items are not vector data and the similarity measure is not Euclidean distance. To do this, we use the classical multi-dimensional scaling technique (MDS) described in Cox and Cox.<sup>17</sup> This technique was designed to find a configuration of  $N$  items where the Euclidean distance between points is proportional to the dissimilarity between corresponding items. This technique uses only the matrix of pairwise dissimilarity values. Unfortunately, such a configuration may require that the items be mapped to  $N - 1$  dimensional points. However, this technique uses no more dimensions than is necessary and orders the dimension on the basis of variance. This allows us to explore the dimensionality of the similarity space and gives us insight into how much information is actually contained in the  $k$  most important dimensions.

This technique, however, does *not* give us a method to index the items. First of all, the  $N - 1$  required dimensions is not practical for known indexing techniques. Secondly, it is necessary to know all the pairwise distances. If this information is known already, then it is not necessary to index this data further.

## 4 EMPIRICAL RESULTS

Other researchers<sup>10,11,5</sup> have explored the reference item approach on low-dimensional spaces with Euclidean distances. However, there is still much work to be done with high-dimensional and non-Euclidean spaces. The following experiments were designed to improve our understanding of the performance of this technique with real-world data and metric similarity measures.

### 4.1 Datasets and dissimilarity measure used

We used several datasets in our experiments. The dataset descriptions are summarized in Table 1. The first two, “uniform” and “scaled”, are each composed of 200 two-dimensional randomly created items. The items in the the uniform dataset were chosen from the square formed by the points  $(0, 0)$  and  $(10, 10)$ . The items in the scaled data set were chosen from the rectangle formed by the points  $(0, 0)$  and  $(10, 1)$ . The items are composed of two values specifying the x and y locations respectively. The values in both sets were chosen using the Unix function call `drand48`. In both these datasets the Euclidean distance is used as the dissimilarity measure.

The remaining datasets were created from three collections of real-world images. The Everyday collection consists of 310 images of everyday scenes (people, sunsets, objects, symbols) from a several sources. The Landsat collection contains 120 Landsat images of Los Alamos, Albuquerque, Moscow, and Cairo. The Lung collection consists of 214 CT images of healthy and diseased lungs. For each collection several feature sets were extracted. Feature signatures were created and matched by the CANDID process described in Kelly et al.<sup>18,19</sup> In this process, a number of Gaussians are chosen to represent the feature set distribution. These Gaussians form the signature and the  $L_2$  distance between the functions is used as the dissimilarity function. For the Everyday collection, color and texture features were used to form two sets of signatures, each of 10 Gaussians. For the Landsat collection texture and spectral features were used to form three signature sets. For the Lung collection local texture statistics were used to form four signature sets.

Name	Size	Structure	Feature	Measure
Uniform	200	Tuple	Random	Euclidean
Scaled	200	Tuple	Random	Euclidean
Everyday 1	310	10 Gaussians	Color	CANDID
Everyday 2	310	10 Gaussians	Texture	CANDID
Landsat 1	120	20 Gaussians	Spectral	CANDID
Landsat 2	120	50 Gaussians	Spectral	CANDID
Landsat 3	120	20 Gaussians	Texture	CANDID
Lung 1	214	20 Gaussians	Texture 5x5 area	CANDID
Lung 2	214	20 Gaussians	Texture 7x7 area	CANDID
Lung 3	214	20 Gaussians	Texture 9x9 area	CANDID
Lung 4	214	20 Gaussians	Texture 11x11 area	CANDID

Table 1: Description of data used in experiments.

## 4.2 Dimensionality of Datasets

The performance of the reference item method is affected by the dimensionality of the dataset. We used the MDS technique to get a better understanding of the experiment datasets. Figure 4 shows the cumulative amount of variance in the first 30 dimensions of each dataset. The dimensionality of the uniform data is not shown in graph form since in our data set the first dimension represents 55% of the variance and the second represents the remaining variance. Note that the random nature of the data used caused a slight deviation from the expected 50%. Similarly, in the scaled data the first dimension accounts for 99% of the variance and the second dimension accounts for the rest. Surprisingly, Figure 4 also shows that 99% of the variance of the Landsat 3 dataset is captured in the first dimension. For the remaining datasets the first five dimensions capture between 70% and 90% of the variance.

## 4.3 Testing technique

We do not currently know how to predict the performance of a particular reference item or how to choose the best performing item. To further our understanding, we exhaustively tested each item and each pair of items in each of our datasets. This was done by choosing each item in turn to be the reference item. For each reference item, each of the remaining items is chosen in turn to be the query item. The remaining  $N - 2$  items are used as database items and searched during queries.

To measure the performance of each item, we kept track of the number of candidate items and of the number of similarity computations actually required. The number of candidate items is the number of items that can not be excluded based solely on the initial similarity computations between the query and reference items. As this list is searched, more items are excluded before they are compared and thus fewer actual similarity computations are required. Since each query for each reference item has a different number of candidate and required similarity computations we report the average for the queries run for a particular reference item.

For example, if there are 100 items in a data set. We examine 100 reference items. For each reference item we process 99 queries. For each query 98 items remain in the database. For each reference item we keep track of the average number of candidate items and the average number of actual similarity computations performed over its 99 queries. To get an idea of the best and worst performance of a reference item we

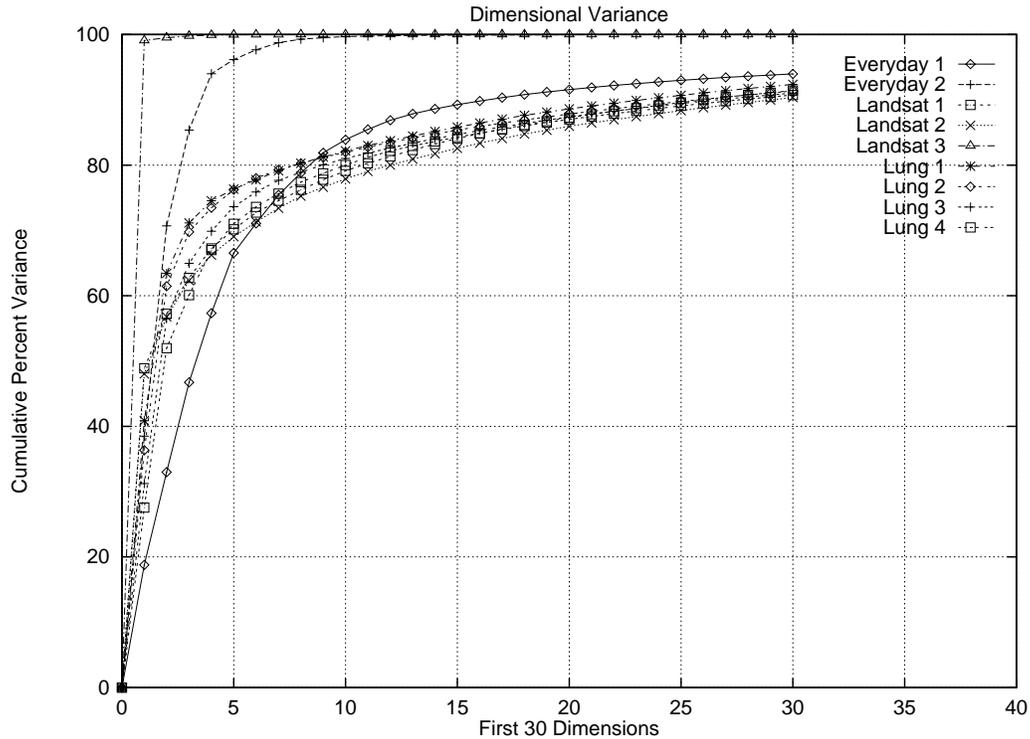


Figure 4: Dimensionality of Everyday, Landsat, and Lung data

report the minimum and maximum average.

Besides testing each reference item individually, we tested each pair of reference items. We were also interested in the performance gains of using more than two reference times. However, exhaustive testing of larger combinations is infeasible so we chose to test a small number (50) of arbitrary combinations of 10 reference items.

#### 4.4 Results

Table 2 shows the results of the retrieval experiments. In this table, the first column indicates the dataset by name. The second column indicates the number of reference items used in each group of trials. The third and fourth column indicate the lowest and greatest percentage of candidate items for each group of trials. A candidate item is an item that was not initially excluded by the reference items. The fifth and sixth columns indicate the lowest and greatest percentage of similarity computations actually required during a search. For example, when searching the uniform data with one reference item, between 78% and 88% of the database was included in the candidate set. However, only 7% to 14% of the database actually needed to be compared to the query item. When using two reference items only 2%-11% of the database needed to be compared and when using 10 reference items only 1% of the database needed to be compared. It should be noted that the one and two reference item trials were done exhaustively, while the 10 reference item trials were sparsely sampled using only 50 arbitrary sets.

Dataset	Number of Ref Items	% of Initial Candidates		% of Actual Comparisons	
		Lowest	Greatest	Lowest	Greatest
Uniform	1	78	88	7	14
	2	45	87	2	11
	10	6	15	1	1
Scaled	1	7	75	2	5
	2	22	78	2	4
	10	7	17	1	2
Everyday 1	1	81	100	74	96
	2	78	99	72	96
	10	78	87	71	77
Everyday 2	1	57	100	15	88
	2	42	100	14	88
	10	26	45	12	23
Landsat 1	1	75	100	57	91
	2	65	98	50	88
	10	53	75	47	58
Landsat 2	1	75	100	57	94
	2	65	98	53	91
	10	62	76	50	58
Landsat 3	1	61	100	23	90
	2	49	98	20	86
	10	28	48	18	25
Lung 1	1	84	100	44	73
	2	70	100	32	70
	10	43	67	25	34
Lung 2	1	84	100	45	72
	2	69	100	31	69
	10	40	65	23	31
Lung 3	1	84	100	47	72
	2	72	100	32	69
	10	40	67	21	28
Lung 4	1	85	100	49	76
	2	73	100	34	73
	10	43	61	22	29

Table 2: Performance results. The performance trials using 10 reference items do not represent true minimums or maximums since only 50 sets of 10 reference items were tested.

Table 2 also shows that significant savings (20%-50%) can be realized for most of the real-world datasets. In some cases, such as the Landsat 3 database, it is possible to achieve savings as high as 82%. It is also interesting to note that the performance increase between one and two reference is generally greater than the performance increase between two and ten reference items. Predicting these performance increases and selecting items with good performance are goals for further research.

## 5 CONCLUSIONS

In this paper we have reexamined previous work of other researchers on best match searching based on the triangle inequality. We have shown that significant (20%-50%) savings can be realized with these methods when searching many non-trivial real world datasets. More specifically, savings of at least 4%-26% were realized with one reference point; 4%-28% with two reference points; and 23%-29% with 10 randomly chosen reference points. Additionally, performance was considerably better with some data sets. For example, the Everyday 2 dataset achieved 77%-88% improvement with 10 randomly chosen reference points.

We have also shown that the savings depends heavily on the dataset in question and on several other factors such as the reference items chosen and the number used. Our future work will focus on predicting the performance increase of using more reference items and on selecting items with good performance.

## 6 ACKNOWLEDGMENTS

This work was performed under a U.S. Government contract (W-7405-ENG-36) by Los Alamos National Laboratory, which is operated by the University of California for the U.S. Department of Energy.

## 7 REFERENCES

- [1] Julio Barros, James French, Worthy Martin, Patrick Kelly, and James M. White. Indexing multi-spectral images for content-based retrieval. In *Proceedings of the 23rd AIPR Workshop on Image and Information Systems: Applications and Opportunities*, Washington, D.C., October 1994.
- [2] Julio Barros, James French, Worthy Martin, and Patrick Kelly. System for indexing multi-spectral satellite images for efficient content-based retrieval. In *Proceedings of IS&T/SPIE: Storage and Retrieval for image and Video Databases III*, San Jose, California, February 1995.
- [3] Julio Barros, James French, and Worthy Martin. Trading efficiency for effectiveness in similarity-based indexing for image databases. In *Proceedings of IS&T/SPIE: Digital image Storage and Archiving Systems*, November 1995.
- [4] Antonin Guttman. R-tree: A dynamic index structure for spatial searching. *ACM SIG MOD Proc.*, 1984.
- [5] Jerome H. Friedman, Jon Louis Bentley, and Raphael Ari Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3(3):209-226, September 1977.

- [6] Thomas P. Yunck. A technique to identify nearest neighbors. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-6(10), October 1976.
- [7] Baek S. Kim and Song B. Park. A fast  $k$  nearest neighbor finding algorithm based on the ordered partition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6), November 1986.
- [8] Jerome H. Friedman, Forest Baskett, and Leonard J. Shustek. An algorithm for finding nearest neighbors. *IEEE Transactions on Computers*, C-24, October 1975.
- [9] Jiri Matousek. Geometric range searching. *ACM Computing Surveys*, 26(4):421–461, December 1994.
- [10] W. A. Burkhard and R. M. Keller. Some approaches to best-match file searching. *Communications of the ACM*, 16(4):230–236, April 1973.
- [11] Marvin Shapiro. The choice of reference points in best-match file searching. *Communications of the ACM*, 20(5):339–343, May 1977.
- [12] J. C. Gower and P. Legendre. Metric and euclidean properties of dissimilarity coefficients. *Journal of classification*, 3:5–48, 1986.
- [13] Keinosuke Fukunaga and Patrenahalli M. Narendra. A branch and bound algorithm for computing  $k$ -nearest neighbors. *IEEE Transactions on Computers*, pages 750–752, July 1975.
- [14] Linda G. Shapiro and Robert M. Haralick. Organization of relational models for scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-4(6), November 1982.
- [15] Enrique Vidal Ruiz. An algorithm for finding nearest neighbours in (approximately) constant average time. *Pattern Recognition*, 4:145–157, July 1986.
- [16] Dennis Shasha and Tsong-Li Wang. New techniques for best-match retrieval. *ACM Transactions on information systems*, 8:140–158, April 1990.
- [17] T. F. Cox and M. A. A. Cox. *Multidimensional Scaling*. Chapman and Hall, 1994.
- [18] Patrick M. Kelly and T. Michael Cannon. Experience with CANDID: Comparison algorithm for navigating digital image databases. In *Proceedings of the 23rd AIPR Workshop on Image and Information Systems: Applications and Opportunities*, Washington, D.C., October 1994.
- [19] Patrick M. Kelly and T. Michael Cannon. CANDID: Comparison algorithm for navigating digital image databases. In *Proceedings of the Seventh International Working Conference on Scientific Database Management*, Charlottesville, Virginia, September 1994.