A Flexible Image Database System for Content-Based Retrieval*

Andrew P. Berman, Linda G. Shapiro
University of Washington
PO Box 352350
Seattle WA 98195-2350 USA
aberman@cs.washington.edu, shapiro@cs.washington.edu

Track: Systems and Applications
Keywords: Content Queries, Digital Libraries, Data Structures

Abstract

There is a growing need for the ability to query image databases based on similarity of image content rather than strict keyword search. As distance computations can be expensive, there is a need for indexing systems and algorithms that can eliminate candidate images without performing distance calculations. As user needs may change from session to session, there is also a need for run-time creation of distance measures. In this paper, we present FIDS, "Flexible Image Database System". FIDS allows the user to query the database based on complex combinations of dozens of predefined distance measures. Using an indexing scheme and algorithms based on the triangle inequality, FIDS can often return matches to the query image without directly comparing the query image to more than a small percentage of the database.

1 Introduction

There is a growing interest in image databases that can be queried based on image content rather than just with keywords. Such queries are based on the use of distance measures–scoring functions that rate the similarity of two images based on pre-defined criteria. There are several challenges that arise from this approach. The first challenge is that of flexibility. The user's definition of similarity may change from session to session, begetting a need for runtime creation of distance measures. Another challenge is that of speed. Distance measure computation requires accessing either the images being compared, or pre-computed data associated with the images. Furthermore, distance computation can be somewhat expensive. Thus, a system that must compute the distance from the query image to each image in a large database may exhibit prohibitively unsatisfactory performance. For certain distance measures and data sets, indexing or clustering schemes can be used to reduce the number of direct comparisons. Yet standard clustering or indexing schemes may not be efficient for some combinations of data sets and distance measures.

We have designed and implemented a prototype database system that allows the user a great deal of flexibility in runtime distance measure creation. The system can also reduce the number of direct distance measure calculations between a given query object and the database elements. FIDS, or "Flexible Image Database System," has been tested on a database of eighteen hundred images, and is being upgraded to handle one hundred thousand images. FIDS allows the user to find approximate matches to query images using complex combinations of dozens of pre-defined distance measures. FIDS can often return results without directly comparing the query image to more than a small percentage of the original database.

There are algorithms in the literature based on the triangle inequality that can reduce the number of distance measure calculations[3, 5, 2, 6, 1] in object retrieval. These methods have the advantage of being applicable to any distance measure which satisfies the triangle inequality. FIDS uses algorithms based on the triangle inequality that are extensions to the methods in the literature.

2 The Use of Multiple Distance Measures

2.1 The Need for Multiple Measures

One can create an innumerable number of distance measures for images based on any set of features and an arbitrary scoring mechanism. We cannot program all these distance measures in advance. This difficulty motivates the idea of giving the user a pre-defined set of base distance measures that he or she can combine to create more complex measures.

*This research was partially supported by the National Science Foundation under Grant IRI-9711771

The potential space of creatable distance measures increases as we increase the number of pre-defined measures. For example, Haering, Myles, and da Vetoria Lobo [9] demonstrated a neural network with 43 distance measures as input that was trained to detect deciduous trees.

Apart from having many different distance measures, we believe that there is a need for multiple variants of the basic distance measures. A piece of software representing a distance measure also represents many decisions made by the programmer. The user of the distance measure may disagree with, not understand, or not even know about these decisions. For example, there are innumerable ways to define bin size and placement for the underlying color histogram of a color-histogram-based distance measure. A user who does not wish to distinguish pink from red will be unhappy with a color measure that does distinguish the two colors. Giving the user multiple color histograms from which to choose increases the potential utility of the system for such a user.

The problems of color measurement only increase when dealing with texture distance measures. As in color-histogram measures, there may be many decisions made by the programmer that control the behavior of the texture measure. These decisions are opaque to the user, yet they affect the performance of the measure for the user. The difficulties are compounded in that the semantics of texture are more obtuse than the semantics of color. Thus, just giving the user multiple textures from which to choose is not a complete strategy, since he or she will have no method (other than trial and error) by which to decide what measures to use.

A more subtle problem with distance measures is that they force the user to make concrete decisions about similarity when the user may not desire to do so. The user may wish to find images that are "similar" without actually specifying any particular measure. For example, consider the user who wants the set of all closest matches to a query image over all possible color-histogram-based distance measures. This set may actually be quite small, yet reliance on any single distance measure will leave out some matches. In this case, such a fuzzy similarity can be approximated by querying the database with several different color distance measures.

2.2 Methods of Combining Distance Measures

Given a database system with several distance measures, one has to decide in what ways the user shall be allowed to combine the distance measures. There are three separate, possibly conflicting objectives. The first objective is to provide many choices to the user. The second objective is to provide an interface that allows the user to understand his choices. The third objective is to support database search of the resultant queries in an efficient manner.

Systems such as QBIC[7] and Virage[8] offer the user the ability to take weighted combinations of color, texture, shape, and position measures, combined with keyword search. But what of the user who wishes to formulate a queries such as "Match on colors, unless the texture and shape are both very close" or "two out of three of color, texture, and shape must match"? These queries cannot be expressed as a weighted sum of individual distance measures. In order to expand users' searching vocabulary, more complex combinations of distance measures must be offered.

To deal with these problems to some extent, we propose the following set of operations to enable more expressive queries: ($d_1 \ldots d_n$ represent distance measures)

- Addition: $d = d_1 + d_2$
- Weighting: $d = Cd_1$
- Max: $d = \text{Max}(d_1, d_2, \ldots, d_n)$
- Min: $d = \text{Min}(d_1, d_2, \ldots, d_n)$

These operations are all invariant under inequality. That is,

$$x_1 \leq y_1, x_2 \leq y_2 \Rightarrow x_1 + x_2 \leq y_1 + y2$$

$$x \leq y, C \geq 0 \Rightarrow Cx \leq Cy$$

$$x_1 \leq y_1, x_2 \leq y_2 \Rightarrow Min(x_1, x_2) \leq Min(y_1, y_2)$$

$$x_1 \leq y_1, x_2 \leq y_2 \Rightarrow Max(x_1, x_2) \leq Max(y_1, y_2)$$

We later show how to take advantage of this invariance to apply triangle-inequality-based pruning algorithms to distance measures that are combined together using the above operations.

The reason for Max and Min is that they enable queries that we expect to be useful. Max, for example, enables tight searches. "I want a match on color and texture and position and shape". Min enables more speculative searches similar to those required by the data mining community: "I want a match on color or texture or position or shape."

2.3 How to Query the Database

User understanding of the distance measures is a problem with any content-based retrieval system. Our proposed set of operations adds a great deal of complexity to the system. There is a need for an additional layer to bridge the gap between user understanding and system capabilities. Possibilities include example-based learning and natural language translation. It may be that different image database domains will require different interfaces.

3 Indexing with the Triangle Inequality

There are several schemes in the literature [3, 5, 2, 6, 1] that take advantage of the triangle inequality to reduce the number of direct comparisons in a database search. The intuition behind all the schemes is that the distance between two objects cannot be less than the difference in their distances to any other object. More formally, let *i* represent a database object, *q* represent a query object, *k* represent some *key* object, and *d* represent some distance measure that is a metric. The following inequality is always true:

$$d(i,q) \geq |d(i,k) - d(q,k)| \quad (1)$$

Thus, by comparing the database and query objects to a third *key* object, a lower bound on the distance between the two objects can be obtained.

We define $l(d,k,i,q)$ to be equal to the lower bound on $d(i,q)$ found by calculating $|d(i,k) - d(q,k)|$. We further shorten $l(d,k,i,q)$ to $l(d,k)$ when there is no confusion as to the values of *i* and *q*.

Burke and Keller[6] first proposed the idea of using the triangle inequality to reduce comparisons. This idea was used by Uhlmann[10] to create vantage-point trees, where each node in a tree corresponds to a carefully chosen key. The subtree rooted at that node was partitioned according to the distance of the leaf elements to the key. Berman[3] and Baeza-Yates, et. al.[1] separately refined Burke and Keller's algorithm by creating a single set of keys to use for all the objects in the database. The single key method differs from vantage-point trees in that it reduces the total number of key comparisons at the expense of increasing the number of fast operations. Barros, et. al.[2], successfully used a single set of keys and the triangle inequality in a real image database. What follows is a description of the general algorithm used with a single set of keys:

Equation (1) can be extended naturally by substituting a set of keys $K = (k_1, \ldots, k_M)$ for *k* as follows:

$$d(i,q) \geq max_{1 \leq s \leq M} |d(i,k_s) - d(q,k_s)| \quad (2)$$

We can see that this inequality is valid by noting that $d(i,q) \geq |d(i,k_s) - d(q,k_s)|$ for all values of *s*. We define $l'(d,K,i,q)$ to be equal to the lower bound on $d(i,q)$ found by using equation (2). As before, we shorten $l'(d,K,i,q)$ to $l'(d,K)$ where possible.

Consider a large set of database objects, $I = \{i_1, \ldots, i_N\}$ and a much smaller set of key objects, $K = \{k_1, \ldots, k_M\}$. Precalculate $d(i_s,k_t)$ for all $\{1 \leq s \leq M\} \times \{1 \leq t \leq N\}$. Now consider a request to find all database objects i_s such that $d(i_s,q) \leq T$ for some query image *q* and threshold value *T*. We can calculate lower bounds on $\{d(i_1,q), \ldots, d(i_N,q)\}$ by calculating $\{d(q,k_1), \ldots, d(q,k_M)\}$ and repeatedly using equation (2). If we prove that *T* is less than $d(i_s,q)$, then we eliminate i_s from our list of possible matches to *q*. After the elimination phase, we search linearly through the uneliminated objects, comparing each to *q* in the standard fashion. This algorithm involves *M+U* distance measure calculations, and $O(MN)$ simple (constant cost) operations, where *U* is the number of uneliminated objects. The hope is that *M+U* is sufficiently smaller than *N* to result in an overall time savings.

3.1 Indexing with Multiple Distance Measures

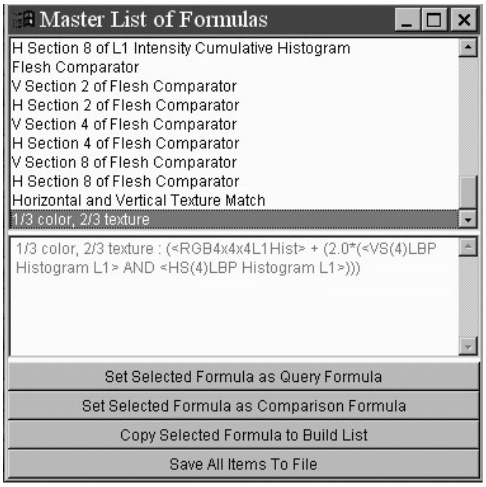
In [5], we extended the above scheme to work with combinations of distance measures. The intuition is that lower bounds on the distance between two objects for distance measures d_1 and d_2 can be used to calculate a lower bound between the objects for distance measure *d* when *d* can be calculated as a combination of d_1 and d_2.

Let $D = d_1, \ldots, d_P$ be a set of distance measures. These distance measures will be known as the *base* distance measures. Let $K = K_1, \ldots, K_P$ be a set of sets of keys, one set of keys for each distance measure. Let $L(D,K,i,q)$ be the set of lower bounds $l'(d_s,K_s,i,q)$ calculated from equation 2 for each tuple $(d_s \in D, K_s \in K), 1 \leq s \leq P$.

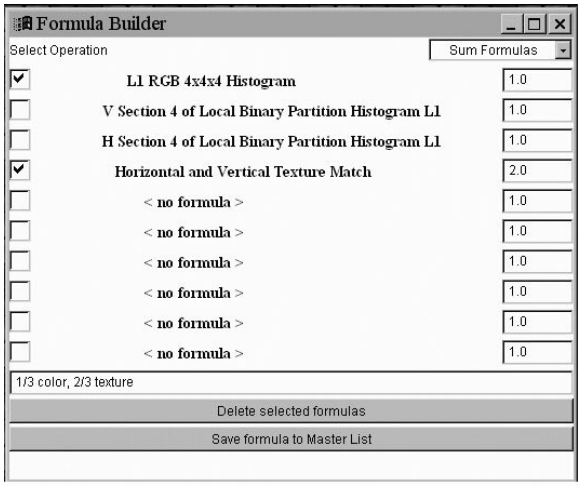
Now consider a new distance measure d' that is of the form $d'(i,q) = f(d_1(i,q), \ldots, d_s(i,q))$, where f is monotonically non-decreasing in its parameters. For example, f might describe a weighted sum of the base measures, or even combinations of minimums and maximiums of sets of the base measures. Since $l'(d_s,K_s,i,q) \leq d_s(i,q)$ for all *s*, substituting $l'(d_s,K_s,i,q)$ for each instance of $d_s(i,q)$ gives us $d'(i,q) \geq f(l'(d_1,K_1,i,q), \ldots, l'(d_s,K_s,i,q))$. Thus we can calculate a lower bound on $d'(i,q)$ given lower bounds on the base distance measures. As in the example with single distance measures, we can thus eliminate database objects as candidates for approximate matching to a query object without direct comparisons. Here we note that the operations on distance measures described earlier–Addition, Weighting, Max, and Min–can be combined to form monotonically non-decreasing functions.

3.2 Finding the best match without thresholding

The calculated lower bounds on the distance from the database objects to the queries have an interesting property. Suppose we are given two database images i_1 and i_2 and query *q* such that $d(i_1,q) < d(i_2,q)$. Experimental evidence suggests that quite often $l(d,k,i_1,q) < l(d,k,i_2,q)$. That is, the ordering of the lower bounds reflects the ordering of the closeness of the images to the query. In tests of up to 37000 images with multiple distance measures, the best match to a query can often be found by ordering the images on the basis of their calculated lower bounds and directly examin-

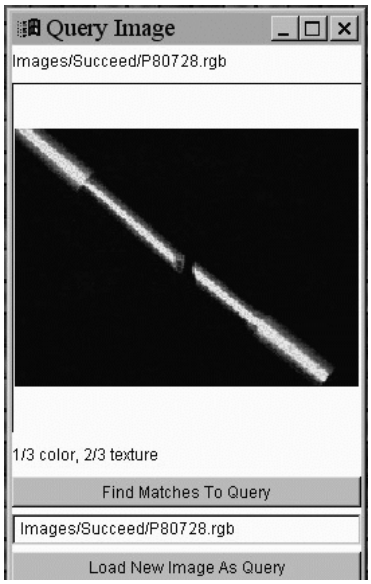


(a) List of available distance measures



(b) New distance measure creation window

Figure 1. The LIST and BUILD windows for the Fast Image Database System



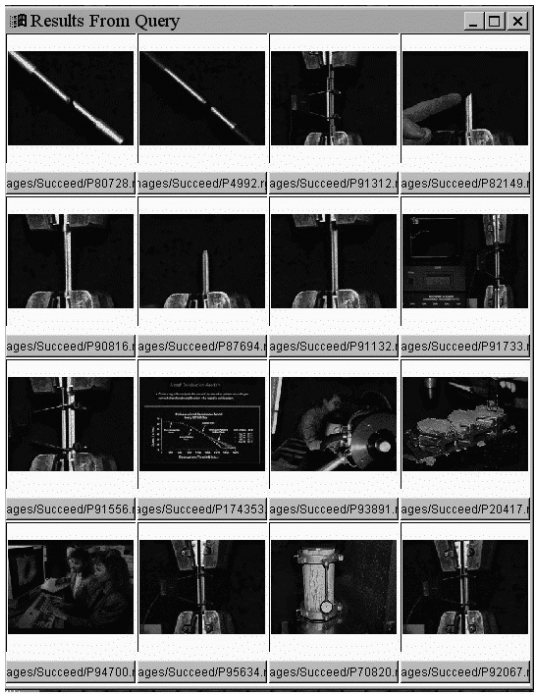


Figure 2. The QUERY and RESULTS windows for the Fast Image Database System

ing the first dozen or so images. This strategy often works even in cases where sizeable fractions of the images are not eliminated by thresholding. In our experiments, we thus not only test for how well our selected keys eliminate images by thresholding, but also how "close to the front" the true best match is placed.

4 Fast Image Database System

FIDS, the Fast Image Database System, is a prototype content based image retrieval system. It currently has eighteen hundred images and will soon be upgraded to one hundred thousand images. It contains 42 distance measures based on color, texture, and feature detection. Position matching is implemented by "gridding" the distance measures and performing distance calculations on corresponding sub-rectangles.

The FIDS interface is shown in Figures 1 and 2. There are 4 windows. The LIST window is a list of available distance measures. When the user clicks on a distance measure, a brief description is offered. The user can select distance measures from this window and copy them to either the BUILD win-

dow or the QUERY window. The BUILD window contains a smaller set of distance measures. One creates a new distance measure by performing the following four activities:

1. Selecting two or more measures in the BUILD window,
2. Selecting one of the three operations "SUM", "MIN", or "MAX",
3. Entering a name for the new distance measure,
4. Pressing the 'build' button.

This new distance measure is then added to the LIST window.

The QUERY window contains the query image and the chosen distance measure. The user runs the query by pressing the run-query button. The RESULTS window contains the sixteen images judged to be the closest by the system. In the full FIDS system, there are three ways to get results. The first way is to order the images by the lower bounds calculated from the triangle inequality. This method is the fastest as no direct comparisons of the query to the database are made. In the second method, the user selects how many images to verify. An image is verified if its true distance to the query is less than the triangle-inequality-derived lower bounds of all the remaining images. The algorithm calculates the true distances to the returned images in lower bound derived order until the required number of images have been verified. Optionally, one can set an upper limit on the number of images that need to be verified. The third method is to perform direct comparisons on all the images. In Berman and Shapiro[5], verification with an upper limit of a few percent of the database almost always returned the correct images.

The user can click on any of the images in the RESULTS window to move that image to the QUERY window. This provides an efficient browsing mechanism of groups of similar images.

5 Experiments

We performed experiments evaluating the efficiency of using the triangle inequality to reduce direct comparisons during searches, using both single and composite distance measures. For several distance measures, the triangle inequality was able to to eliminate over 90 percent of the images from direct comparison. In most cases, over 80 percent of the images were eliminated. In the worst case, up to two-thirds of the images still had to be compared directly to ensure that the best match was found. These experiments were conducted with keys randomly chosen from the database[5].

We also experimented with key selection algorithms[4]. We were able to improve performance over random keys by factors of 1. 2 to 2. 2 depending on the distance measures and search criteria. That is, the number of images that had to be directly compared to the query using random keys was from 20 percent to 120 percent greater than if we used the best keys selected by the algorithms.

References

[1] R. Baeza-Yates, W. Cunto, U. Manber, and S. Wu. Proximity matching using fixed-queries trees. In *Combinatorial Pattern Matching*, pages 198–212. Springer-Verlag, June 1994.

[2] J. Barros, J. French, W. Martin, P. Kelley, and M. Cannon. Using the triangle inequality to reduce the number of comparisons required for similarity-based retrieval. In *IS&T/SPIE - Storage and Retrieval for Still Image and Video Databases*, volume IV, Jan 1996.

[3] A. Berman. A new data structure for fast approximate matching. Technical Report 1994-03-02, Dept. of Computer Science, University of Washington, 1994.

[4] A. Berman and L. Shapiro. Selecting good keys for triangle-inequality based pruning algorithms. In *IEEE International Workshop on Content-based Access of Image and Video Databases*, 1997.

[5] A. P. Berman and L. G. Shapiro. Efficient image retrieval with multiple distance measures. In *Proceedings of the SPIE Conference on Storage and Retrieval for Image and Video Databases*, February 1997.

[6] W. A. Burkhard and R. M. Keller. Some approaches to best-match file searching. *Communications of the ACM*, 16(4):230–236, Apr 1973.

[7] M. Flickner, H. Sawhnew, W. Niblack, J. Ashley, Qian-Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by image and video content: the qbic system. *Computer*, 28(9):23–32, Sep 1995.

[8] A. Gupta. Visual information retrieval: A virage perspective. Technical report, Virage, Inc. http://www.virage.com/wpaper, 1995-1997.

[9] N. Haering, Z. Myles, and N. de Vitoria Lobo. Locating deciduous trees. In *Content-Based Access of Image and Video Libraries*, June 1997.

[10] J. Uhlmann. Satisfying general proximity/similarity queries with metric trees. *Information Processing Letters*, 40:175–179, 1991.