Multiscale Branch and Bound Image Database Search *†‡

Jau-Yuen Chen, Charles A. Bouman, Jan P. Allebach
School of Electrical Engineering
1285 EE building
Purdue University
West Lafayette, IN 47907-1285

ABSTRACT

This paper presents a formal framework for designing search algorithms which can identify target images by the spatial distribution of color, edge and texture attributes. The framework is based on a multiscale representation of both the image data, and the associated parameter space that must be searched. We define a general form for the distance function which insures that branch and bound search can be used to find the globally optimal match. Our distance function depends on the choice of a convex measure of feature distance. For this purpose, we propose the L_1 norm and some other alternative choices such as the Kullback-Liebler and divergence distances. Experimental results indicate that the multiscale approach can improve search performance with minimal computational cost.

Keyword: multiscale search, image similarity, content-based retrieval, color histogram, convex function

1 INTRODUCTION

One method for searching large databases of natural images is to retrieve images with color, texture and edge properties which are similar to a query image provided by the user. Previous approaches to this problem have largely been based on the extraction of a single global feature vector from the query image and each target image in the database. The search is then performed by comparing the query image feature vector to each target image feature vector. This approach is very computationally efficient, but it tends to be limited by the simple global attributes that can be effectively represented in the feature vector.

A more natural approach is to directly compare the spatial attributes of the query and target images. However, less attention has been paid to this approach perhaps because it is perceived as being too computationally expensive. Template matching can be computationally expensive for two reasons. First, spatial comparison of color and texture attributes is intrinsically more computationally expensive then comparison of a single global feature vector. Second, spatial matching assumes that the query and target images are properly aligned. Direct implementation of this alignment step can be very computationally expensive.

*This work was supported by Apple Computer, Inc.
†To try this algorithm visit the URL http://www.ece.purdue.edu/~bouman
‡Appear in the Proceedings of the *SPIE/IS&T Conf. on Storage and Retrieval for Image and Video Databases V*, vol. 3022, pp. 133-144, San Jose CA, Feb. 10-13, 1997.

To capture information about the spatial distribution of color, Gong *et. al.*[1] divided an image into 9 subareas (3x3) and created a histogram for each of the sub-areas. Later Zhang *et. al.*[2] concluded that spatial histograms, such as that used by Gong *et. al.*, significantly improving the retrieval rate. Recently, Jacobs *et. al.*[3] proposed a strategy based on direct spatial matching of wavelet coefficients in the query and database images. In each of these cases, the information from all scales was treated as a single feature vector, and the query and target images were match without relative shifting.

This paper presents a novel framework for image search which allows the use of spatial template matching approaches while retaining the computational efficiency required to search large databases. The key to the approach is a synergistic combination of four techniques: multiscale representation of query and target images; branch and bound search; convex distance metrics based on statistical image models; and multiscale representation of the translation parameters. By using these techniques we are able to efficiently search image databases using a distance metric based on the spatial matching of the color, texture, and edge attributes with 2-D relative translation of the query and target images. This work builds on previous research presented in references.[4,5]

We first show that the branch and bound technique is well suited for searching images in a multiscale framework. This is because the branch and bound algorithm can eliminate poor matches at coarse scales with minimum computation. This insures that each image is only searched to a resolution necessary to determine if it is a potential match candidate. We also note that since the branch and bound algorithm finds the global minimum, this computational savings comes at no cost in the quality of the match.

The key to the effectiveness of the branch and bound algorithm is the formulation of a tight lower bound to the dissimilarity measure. In the multiscale framework, this lower bound can be computed at any resolution, but becomes tighter as the resolution is increased. To formulate this lower bound, we propose a class of dissimilarity criteria formed by summing distances between feature vectors at corresponding locations in the query and target images. We show that by restricting ourselves to distance functions that are convex, we can formulate a useful lower bound on the total dissimilarity measure.

Finally, we discuss the extension of this search method to allow translations of the relative alignment of the query and target images. This is done by quantizing the space of shift parameters in a multiscale pyramid structure. The branch and bound search is then applied to the forest of trees in which the nodes of each tree represents a potential shift of the query image relative to a specific target image.

We present experimental results of searches using a database of $10,000$ images containing a wide variety of natural images including people, wildlife, business, art, landscapes and textures. In these experiments we illustrate how spatial distribution can enhance the quality of the search results without requiring excessive computation. Current versions of the algorithm search the $10,000$ image database in approximately 15 seconds on a Unix workstation.

2 MULTISCALE BRANCH AND BOUND SEARCH

Formally, we define the distance function $D(i)$ to be the distance of the i^{th} image in the database to the query image. Our objective is then to search the N images in the database to find the best match.

$$i^* = \arg \min_{1 \leq i \leq N} D(i) \tag{1}$$

However, solving (1) can be a very difficult problem because with current storage technology N can easily take on values greater than $10,000$. While very simple metrics for $D(i)$ can be rapidly computed and minimized, these metrics may not provide sufficient discrimination to identify the images of interest. Alternatively, more complex metrics may yield better results, but require computation that makes them impractical or impossible to implement for large databases.

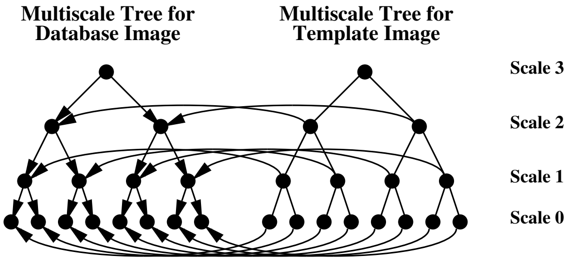


Figure 1: Structure used to compute similarity metric. Both the query and target images are represented by multiscale tree structures where each node contains a feature vector for the corresponding region of the image. The distance function is computed by summing the distance between corresponding nodes of the query and target images.

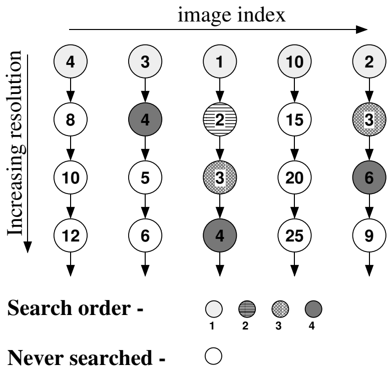


Figure 2: Branch and bound search example. At the first iteration, all the images in the database are checked at the courses scale. Then the search proceeds to finer scales in order of similarity.

Our approach will be to use a multiscale distance function. We will see that such a function can potentially compare fine image differences, but can still be efficiently minimized. The structure of the multiscale distance function is illustrated in Fig. 1 for the 1-D case. Both the query and target images are first decomposed into a pyramid of feature vectors. Each level of the tree represents a scale, k ranging from 0 to $L-1$, and a node in the tree at scale k is denoted by $s \in S_k$. At the coarsest scale, the entire image is represented by a single node. As k is reduced, the resolution is increased, and each feature represents a more spatially localized region of the image. The distance function $D(i)$ is computed by adding the distances between corresponding features of the query and target trees.

The difficulty in using a multiscale distance function is that computing the match at fine scales is very computationally expensive. However, this problem can be circumvented by formulating a lower bound to $D(i)$ which is more efficient to compute. More specifically, we will define a function $\underline{D}_k(i)$ such that $D(i) \geq \underline{D}_k(i)$, $D(i) = \underline{D}_0(i)$, and $\underline{D}_k(i)$ can be computed using only nodes at scales greater than or equal to k.

The following algorithm implements branch and bound search. Notice that k_i and $\underline{D}(i)$ are temporary storage variables which contain the current finest resolution of search and lower bound to the distance for image i.

1. Set $k_i \leftarrow L-1$ for all i.
2. Compute $\underline{D}(i) \leftarrow \underline{D}_{L-1}(i)$ for all i.
3. $i^* = \arg\min_{1 \leq i \leq N} \underline{D}(i)$
4. While $k_{i^*} \neq 0$
   (a) $k_{i^*} \leftarrow k_{i^*} - 1$
   (b) $\underline{D}(i^*) \leftarrow \underline{D}_{k_{i^*}}(i^*)$
   (c) $i^* = \arg\min_{1 \leq i \leq N} \underline{D}(i)$

[1] Figure 2 illustrates this branch and bound search with a simple example. At each step, the smallest nodes are searched to the next resolution. If the algorithm terminates with the value i^*, then $D(i^*) = \underline{D}_0(i^*)$ is the smallest distance among all searched nodes. Since each node is a lower bound on nodes that fall below it, $D(i^*)$

[1] Note there was a typo in the original publication of equation 4b)

must then be the globally minimum solution. When the lower bound is tight, the value of $\underline{D}_k(i)$ may be used to eliminate many poor matches and thereby reduce the size of the search. This search strategy, known as branch and bound or more generally as A^* search,[6] leads to an optimal solution to the search problem, but often with dramatically reduced computation. In practice, the minimization of step 4c may be efficiently implemented using a data structure known as a heap. This reduces the time required to find the minimum to order $\log N$. While in theory this heap maintenance dominates the computation, in practice N is generally small enough that evaluation of the distance function dominates computation.

2.1 Multiscale distance function

In this section, we define a general form for the distance function which insures that the branch and bound search will be optimal. Figure 1 shows that our distance function will be formed by summing the distances between corresponding nodes of the query and target images. Each node for the i^{th} target image is assumed to contain a feature vector, h_{iks}, and a pixel count, N_{ks}, where k is the scale, and s is the index of the node. The set S_k will denote the finite set of all node indices at scale k. Notice that the pixel count, N_{ks}, is assumed to *not* depend on the image i. This is true since the regions represented by each node are generally of the same size and shape. [2] In particular, we will always use $N_{ks} = 4^k$. The tree for the query image is similar, but contains feature vectors, u_{ks}. Notice that u_{ks} is not indexed by i since there is only a single query image. The children of a node $s \in S_k$ are denoted by the set $C(s)$. In the 1-D case, each node has two children, but for images, most nodes have four children.

Given this notation, we will assume that the distance function has the form

$$\begin{aligned} D(i) &= \sum_{k=0}^{L-1} w_k d_k(i) \\ d_k(i) &= \sum_{s\in S_k} N_{k\,s} I(h_{i\,k\,s}, u_{k\,s}) \end{aligned} \qquad (2)$$

where $w_k \geq 0$ are scalar weights, and $I(\cdot,\cdot)$ is a positive functional which assigned a distance to corresponding nodes in the query and target images.

A basic assumption of our approach is that feature vectors at each node are formed by a weighted average of the feature vectors from child nodes. This assumption is formally stated by the following recursions.

$$\begin{aligned} N_{k+1\,s} &= \sum_{r\in C(s)} N_{k\,r} \\ h_{i\,k+1\,s} &= \sum_{r\in C(s)} \frac{N_{k\,r}}{N_{k+1\,s}} h_{i\,k\,r} \\ u_{k+1\,s} &= \sum_{r\in C(s)} \frac{N_{k\,r}}{N_{k+1\,s}} u_{k\,r}\ . \end{aligned} \qquad (3)$$

This is a very reasonable assumption for many useful feature vectors. For example, both normalized histograms and moments are formed by spatial averaging, and therefore obey this assumption.

The following property gives a computationally efficient method for computing a lower bound to the distance $D(i)$. This lower bound results from assumptions (2), (3), and the additional assumption that $I(\cdot,\cdot)$ is convex. The importance of the lower bound, $\underline{D}_k(i)$, is that it is only a function of features at scales k and above. This

[2] This assumption may cause some difficulties if the shape of the query and target images are different. However, such shape mismatch is generally a problem when comparing spatial attributes of images.

means that $\underline{D}_k(i)$ may be computed with much less computation than $D(i)$. As discussed in the previous section, a tight lower bound may be used to eliminate many poor matches.

Property 1: Let $D(i)$ be a cost function with the form of (2), and let the feature vectors satisfy the recursions of (3). Further assume that the functional $I(h,u)$ is a convex function of (h,u). Then the function

$$\underline{D}_k(i) = \sum_{l=0}^{k-1} w_l d_k(i) + \sum_{l=k}^{L-1} w_l d_l(i)$$

is a lower bound on $D(i)$ with $D(i) = \underline{D}_0(i)$.

Proof:
It is enough to show that $d_k(i) \geq d_{k+1}(i)$.

$$\begin{aligned}
d_{k+1}(i) &= \sum_{s\in S_{k+1}} N_{k+1\,s} I(h_{i\,k+1\,s}, u_{k+1\,s}) \\
&= \sum_{s\in S_{k+1}} N_{k+1\,s} I\left(\sum_{r\in C(s)} \frac{N_{k\,r}}{N_{k+1\,s}} h_{i\,k\,s}, \sum_{r\in C(s)} \frac{N_{k\,r}}{N_{k+1\,s}} u_{k\,s}\right) \\
&\leq \sum_{s\in S_{k+1}} N_{k+1\,s} \sum_{r\in C(s)} \frac{N_{k\,r}}{N_{k+1\,s}} I(h_{i\,k\,s}, u_{k\,s}) \\
&= \sum_{s\in S_{k+1}} \sum_{r\in C(s)} N_{k\,r} I(h_{i\,k\,s}, u_{k\,s}) \\
&= \sum_{s\in S_k} N_{k\,r} I(h_{i\,k\,s}, u_{k\,s}) \\
&= d_k(i)
\end{aligned}$$

2.2 Convex distance functions

In this section, we discuss some potential choices for convex distance functions $I(h,u)$ where h and p are the feature vectors of length M. Table 1 summarizes these choices.

In fact, a variety of convex metrics have been proposed for this purpose. Swain and Ballard proposed the use of an L_1 norm, and argued that it had a useful interpretation as a measure of intersection when used with histograms.[7] Alternatively, L_2 norms have also been applied,[8,9] and L_q norms are an obvious generalization. We also note that any of these metrics may be generalized by preprocessing the feature vector with a linear transformation. This leads to the special cases of L_1 norms of the cumulative histogram,[9] or weighted L_2 norms.[8]

In many applications, the feature vectors are constrained to be positive. For example, normalized histograms (which we will use) must be a member of the set $\Omega^M = \left\{h \in I\!R^M : h_j \geq 0 \text{ and } \sum_{j=1}^{M} h_j = 1\right\}$. Under these conditions, there are a number of other interesting convex distance functions to consider. In particular, the Kullback-Liebler distance listed in Table 1 is known to be a convex function of (h,u).[10] Furthermore, it has an interesting statistical interpretation when h and p are the normalized histograms.

Consider a region of the image containing N pixels with a normalized histogram of h. Assume that these pixels are independent and identically distributed with marginal distribution given by the M dimensional vector

Distance Type	Formula	Convex	Comments
L_1 norm	$I_{l1}(h,u) = \sum_{i=1}^{M} \lvert h_i - u_i \rvert$	yes	histogram intersection
L_2 norm	$I_{l2}(h,u) = \sum_{i=1}^{M} \lvert h_i - u_i \rvert^2$	yes	
L_p norm	$I_{lp}(h,u) = \sum_{i=1}^{M} \lvert h_i - u_i \rvert^p$	yes	$1 \leq p \leq 2$
Kullback-Liebler	$I_{KL}(h,u) = \sum_{i=1}^{M} h_i \log \frac{h_i}{u_i}$	yes	$\sum_{i=1}^{M} h_i = \sum_{i=1}^{M} u_i = 1$
divergence	$I_d(h,u) = \sum_{i=1}^{M} (h_i - u_i) \log \frac{h_i}{u_i}$	yes	symmetric
generalized divergence	$I_{gdp}(h,u) = \sum_{i=1}^{M} (h_i - u_i)({h_i}^{p-1} - {u_i}^{p-1})$	Conjectured for $p > 1$	defined for $h_i = u_i = 0$

Table 1: Five similarity metrics used in this research.

θ. Under this assumption, the probability of the pixels, x, are given by

$$\log p(x|\theta) = N \sum_{j=1}^{M} h_j \log \theta_j \ .$$

Using this model, we may formulate the distance as the log likelihood ratio between hypothesis 1 that the $\theta = u$ and hypothesis 0 that $\theta \neq u$.

$$\begin{aligned} d &= \log \left(\frac{\max_{\theta \in \Omega^M} p(x|\theta)}{p(x|\theta = u)} \right) \\ &= N \sum_{j=1}^{M} h_j \log \frac{h_j}{u_j} \\ &= N I_{KL}(h,u) \end{aligned}$$

We note that these distance terms are exactly the $d_k(i)$ terms of (2).

The Kullback-Liebler distance is useful because differences are effectively normalized to the number of pixels. Therefore, a 10% deviation yields the same difference independently of the absolute value. The divergence distance is a symmetrized version of the Kullback-Liebler distance form by $I_{KL}(h,u) + I_{KL}(u,h)$. In addition, the divergence distance is appropriate to use when $\sum_{i=1}^{M} h_i \neq 1$ or $\sum_{i=1}^{M} u_i \neq 1$. In practice, we have found the divergence distance to perform better.

However, a problem with either the Kullback-Liebler or divergence distance is that they are not well defined when h_i or u_i are 0. One solution is to add a small quantity (we use 10^{-5}) to both h_i and u_i and renormalize them. Perhaps a more elegant solution is the generalized divergence distance proposed in Table 1. We conjecture based on numerical evaluation of the Hessian that the generalized divergence is also a convex function of (h,u).

In practice, we have found the L_1 norm to yield accurate matches with minimum computation. However,

		Dynamic Range	Number of Bins	Resolution
Color histogram		Dynamic Range	Number of Bins	Resolution
	L^*	$0 \cdots 100$	16	$6\Delta E$
	a^*	$-100 \cdots 100$	32	$6\Delta E$
	b^*	$-100 \cdots 100$	32	$6\Delta E$
Texture histogram		Dynamic Range	Number of Bins	Resolution
	L^*	$0 \cdots 100$	16	$6\Delta E$
	a^*	$0 \cdots 200$	32	$6\Delta E$
	b^*	$0 \cdots 200$	32	$6\Delta E$
Edge histogram		Dynamic Range	Number of Bins	Resolution
	L^*	$0 \cdots 2\pi$	16+1	$\pi/8$
	a^*	$0 \cdots 2\pi$	16+1	$\pi/8$
	b^*	$0 \cdots 2\pi$	16+1	$\pi/8$

Table 2: Structure of feature vector.

the divergence appears to be somewhat superior particularly for color histogram features. We believe that the generalized divergence may have the potential for the best overall matching accuracy.

3 FEATURE VECTORS

Our feature vectors, h and p, will each consist of three individual feature vectors representing image color, texture and edges respectively. Each of these component feature vectors will be a histogram of the color, texture or edge values in a region of the image. The specific form of these histogram features is described below. Table 2 lists the properties of the feature vector discussed below. In practice, many of the histogram entries are empty so the feature vectors can be efficiently stored using run length coding.

The color features are formed by independently histograming the three components of the CIELa^*b^* color space. La^*b^* is a good choice because it is visually uniform and it is also a reasonable approximation to a true opponent color space. In fact, Gargi *et. al.*[11] evaluated six color spaces, and found that La^*b^* performed well. Table 2 shows that we chose the number of histogram bins so that the resolution of each bin is approximately $6\Delta E$. In addition, we found that it was important to smooth the histogram so that small color shifts do not excessively effect the match. Therefore, we apply a Gaussian filter kernel with a standard deviation of $6\Delta E$.

The texture feature is formed by histograming the magnitude of the local image gradient for each color component. More specifically, $D_x L$ and $D_y L$ are defined as the x and y derivatives of the L component computed using the conventional Sobel operators. Then $T_L = \sqrt{(D_x L)^2 + (D_y L)^2}$, and T_a, T_b are defined similarly. These three features are histogramed as shown in Table 2.

The edge features are formed by thresholding the edge gradient and then computing the angle of the edge for all points that exceed the threshold.

$$\Theta_L = \begin{cases} \arctan(D_x L, D_x L) & T_L \geq \sigma_L \\ \emptyset & T_L < \sigma_L \end{cases}$$

The threshold σ_L is given by the standard deviation of the L component for the particular image. The values of Θ_a and Θ_b are computed similarly. These values are then histogramed as shown in Table 2. Notice that a 17^{th} bin is reserved for the case of $\Theta = \emptyset$.

4 TEMPLATE SHIFTING

One limitation of the distance function $D(i)$ is that it can be sensitive to spatial translations of the image regions. Therefore, we extend the search method to allow translations of the query image.

Let Δ be a 2-D displacement vector, and let $u_{k\,s}(\Delta)$ be the new query tree that results from displacing the query image by Δ. The new distance function is then given by

$$d_k(i,\Delta) = \sum_{s\in S_k} N_{k\,s} I(h_{i\,k\,s}, u_{k\,s}(\Delta))$$

The new search problem requires optimization over both i and Δ which is a computationally intractable problem since Δ is a continuously valued parameter. However, we may discretize the search process by forming a multiscale tree of parameter values $\Delta_{k\,t}$ for $t \in T_k$ where T_k is the discrete set of displacement nodes at scale k. Once again, for $t \in T_k$ with $0 < k < L$, we use the notation $C(t)$ for to denote the children of node t at scale $k-1$. We use the notation $C^n(t)$ to denote the n^{th} descendent of the node t, and we denote the set of all descendents by $C^L(t) = \cup_{l=1}^{k} C^l(t)$ where $t \in T_k$. Using this definition of $d_k(i,\Delta)$, we define a new distance function

$$D(i,t) \;=\; \sum_{k=0}^{L-1} w_k d_k(i,\Delta_{0\,t}) \tag{4}$$

Assumption: Let $t \in T_k$, and let $r, s \in C^L(t)$ be descendents of t at scales $l, m < k$ respectively. Then we assume that there is a fixed ϵ such that

$$|d_k(i,\Delta_{l\,r}) - d_k(i,\Delta_{m\,s})| \le \epsilon\ .$$

The value of ϵ may be made small by choosing a sampling period for the parameter space Δ which is sufficiently fine. This is because a sufficiently fine sampling will insure that $||\Delta_{l\,r} - \Delta_{k\,t}||$ is small.

Property 2: Let $D(i,t)$ be a cost function with the form of (4), and let the feature vectors satisfy the recursions of (3). Further assume that the functional $I(\cdot,\cdot)$ is convex. Define the function

$$\underline{D}_k(i,t) = \sum_{l=0}^{k-1} w_l d_k(i,\Delta_{k\,t}) + \sum_{l=k}^{L-1} w_l d_l(i,\Delta_{k\,t})$$

where $t \in T_k$. Then using Assumption 1,

$$D(i,r) \ge \underline{D}_k(i,t) - \epsilon \sum_{k=0}^{L-1} w_k$$

where $r \in C^k(t)$, and $D(i,r) = \underline{D}_0(i,r)$ for all $r \in T_0$.

Proof:
Let $t \in T_k$ and $r \in C^k(t)$, then

$$\begin{aligned} D(i,r) - \underline{D}_k(i,t) &= \sum_{l=0}^{k-1} w_l \left(d_l(i,\Delta_{0\,r}) - d_k(i,\Delta_{k\,t})\right) \\ &\quad + \sum_{l=k}^{L-1} w_l \left(d_l(i,\Delta_{0\,r}) - d_l(i,\Delta_{k\,t})\right) \\ &= \sum_{l=0}^{k-1} w_l \left(d_l(i,\Delta_{0\,r}) - d_k(i,\Delta_{0\,r})\right) \end{aligned}$$

$$
\begin{aligned}
& + \sum_{l=0}^{k-1} w_l \left(d_k(i, \Delta_{0\,r}) - d_k(i, \Delta_{k\,t}) \right) \\
& + \sum_{l=k}^{L-1} w_l \left(d_l(i, \Delta_{0\,r}) - d_l(i, \Delta_{k\,t}) \right) \\
\geq \quad & \sum_{l=0}^{k-1} w_l\,(0) + \sum_{l=0}^{k-1} w_l\,(-\epsilon) + \sum_{l=k}^{L-1} w_l\,(-\epsilon) \\
= \quad & -\epsilon \sum_{l=0}^{L-1} w_l
\end{aligned}
$$

5 EXPERIMENTAL RESULTS

In the following experimental results, we our algorithm to search a image data base containing $10,000$ images. We applied a weighting of $w_k = 1/2^k$ at each scale $k \geq L - 3$. We only searched to a 4×4 resolution which is equivalent to setting $w_k = 0$ for of $k < L - 3$.

Figure 3 shows the computational advantages of the multiscale search algorithm. The first graph shows that only a small number of images are search to finer resolutions. As the resolution increases, the number of possible shifts of the query image rapidly increases, so the number of positions searched does not decrease as rapidly as the number of images searched. The third graph shows the number of nodes compared. This measure is approximately proportional to computation; so we see that the computational cost is approximately constant at each resolution. The final graph shows a histogram of computation time for search of the $10,000$ images on an HP755/100. Average search time is approximately 15 seconds.

The large database is useful for making realistic assessments of computational speed and qualitative performance. However, it is difficult to make accurate measurements for match quality. To make these more accurate measurements, we used the methodology suggested by Frese *et. al.*[12] Frese *et. al.* collected data from subjects on smaller 200 image randomized subsets of the full database. Our algorithm was then used to rank the complete set of 200 images. Given this ranking, one can compute the probability of the subjects "best" match falling among the first n images as ranked by the algorithm. This plot of probability of detection versus $\log n$ gives a useful measure of algorithm performance.

Figure 4 shows the increased search accuracy of the multiscale search versus the search using only the single coarse scale feature vector. It shows a 5 percent improvement for images ranking from 3 to 10 out of 200. Figures 5 6 and 7 compare the search results using various combinations of color, texture and edge features. In particular, Fig. 7 indicates that texture and edge features are important when used with color features.

Figure 8 illustrates how multiscale search improves match quality using a variety of alternative feature vectors.

6 ACKNOWLEDGEMENTS

Special thanks to John Dalton of Apple Computer for all his insights and assistance in this research.

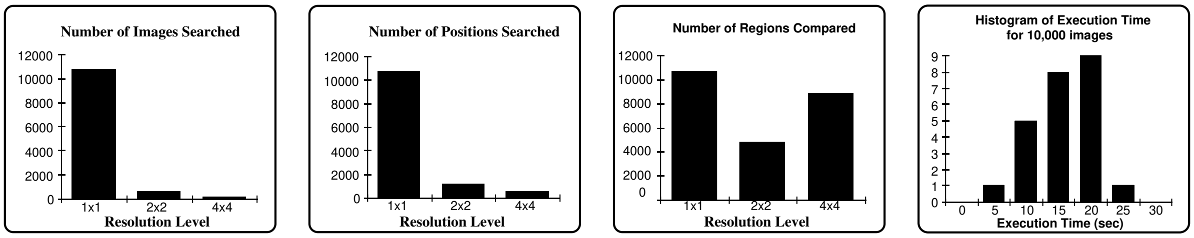


Figure 3: Computational analysis for a set of 30 query images.

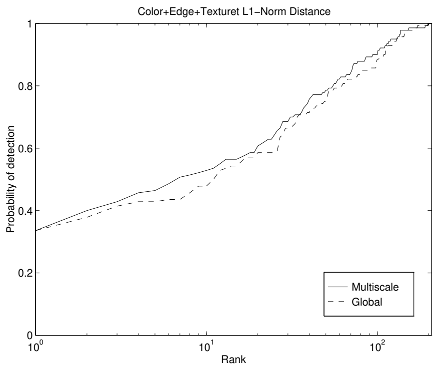


Figure 4: Comparison of multiscale and global search using L_1norm with color, texture and edge features.

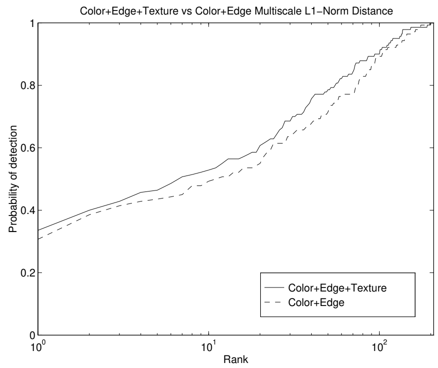


Figure 5: The effect of adding texture features to multiscale search using L_1norm.

7 REFERENCES

[1] Y. Gong, H. Zhang, and C. H. Chuan, "An image database system with fast image indexing capability based on color histograms," *Proceedings of 1994 IEEE Region 10's ninth annual international conference : Frontiers of computer technology*, August 22-26 1994, Singapore, pp. 407–411.

[2] H. Zhang, Y. Gong, C. Y. Low, and S. W. Smoliar, "Image retrieval based on color features: An evaluation study," *Proc. of SPIE/IS&T Conf. on Storage and Archiving Systems*, vol. 2606, October 25-26 1996, Philadelphia, PA, pp. 212–220.

[3] C. E. Jacobs, A. Finkelstein, and D. H. Salesin, "Fast multiresolution image querying," *ACM Siggraph 95 proceedings*, August 9-11 1995, Los Angeles, CA, pp. 277–285.

[4] S. Sista, C. A. Bouman, and J. P. Allebach, "Fast image search using a multiscale stochastic model," *Proc. of IEEE Int'l Conf. on Image Proc.*, October 23-26 1995, Washington, DC, pp. 225–228.

[5] J.-Y. Chen, C. A. Bouman, and J. P. Allebach, "Stochastic models for fast multiscale image search," *Joint Conference of Classification Society of North America and Numerical Taxonomy Group*, June 13-16 1996, Amherst, MA.

[6] J. Pearl, *Heuristics: Intelligent Search Strategies for Computer Problem Solving.* Reading, Massachusetts: Addison-Wesley Publishing Company, 1984.

[7] M. J. Swain and D. H. Ballard, "Color indexing," *Intern. Journal of Computer Vision*, vol. 7, no. 1, pp. 11–32, 1991.

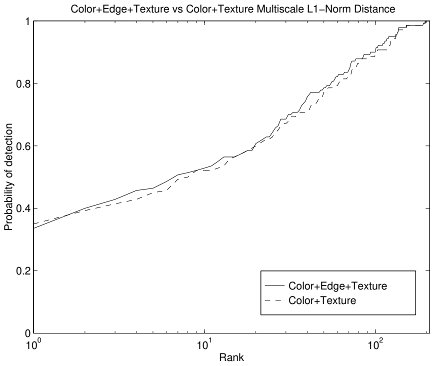


Figure 6: The effect of adding edge features to multiscale search using L_1norm.

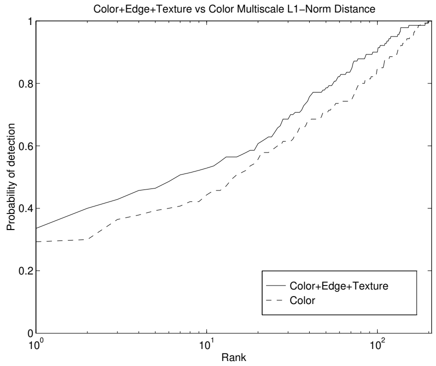
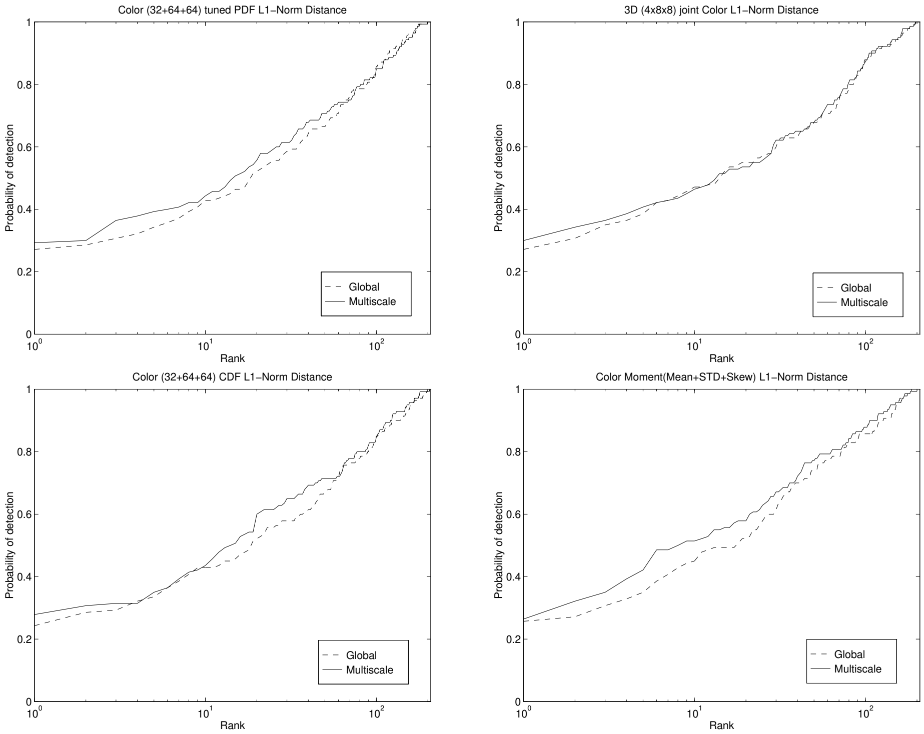


Figure 7: The effect of adding texture and edge features to multiscale search using L_1norm.



a	b
c	d

Figure 8: Each plot compares multiscale and global search using different features and L_1 norm. (a) color features (no texture or edge); (b) 3D color histogram; (c) color cumulative histogram (no texture or edge); (d) color central moments (no texture or edge).

Figure 9: This figure illustrates the effect of multiscale search for some selected examples using a database of 10,000 images. Notice that the multiscale criteria tends to eliminate matches with different spatial distributions of color, texture and edges.

[8] J. Hafner, H. S. Sawhney, W. Equitz, M. Flickner, and W. Niblack, "Efficient color histogram indexing for quadratic form distance functions," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 17, no. 7, pp. 729–736, July 1995.

[9] M. Stricker and M. Orengo, "Similarity of color images," *Proc. of SPIE/IS&T Conf. on Storage and Retrieval for Image and Video Databases III*, vol. 2420, February 9-10 1995, San Jose, CA, pp. 381–392.

[10] S. Ihara, *Information Theory for continuous systems*. Singapore: World Scientific, 1993.

[11] U. Gargi, S. Oswald, D. Kosiba, S. Devadiga, and R. Kasturi, "Evalutation of video sequence indexing and hierarchical video indexing," *Proc. of SPIE/IS&T Conf. on Storage and Retrieval for Image and Video Databases III*, vol. 2420, February 9-10 1995, San Jose, CA, pp. 144–151.

[12] T. Frese, C. A. Bouman, and J. Allebach, "A methodology for designing image similarity metrics based on human visual system models," *(submitted to) Proc. of SPIE/IS&T Conf. on Human Vision and Electronic Imaging*, February 1997.