A Fast, Background-independent Retrieval Strategy for Color Image Databases*

M. Das B. A. Draper
W. J. Lim R. Manmatha E. M. Riseman
Department of Computer Science
University of Massachusetts
Amherst, MA 01003
e-mail : mdas@cs.umass.edu

CMPSCI Technical Report TR-96-79
November, 1996.

Abstract

We describe an interactive, multi-phase color-based image retrieval system which is capable of identifying query objects specified by the user in an image in the presence of significant, interfering backgrounds. The system uses a split and merge histogram peak detection technique, efficient indexing and a color neighborhood graph-based refinement phase for removing false matches. The method is fast and

*This material is based on work supported in part by the National Science Foundation, Library of Congress and Department of Commerce under cooperative agreement number EEC-9209623, in part by Defense Advanced Research Projects Agency/ITO under ARPA order number D468, issued by ESC/AXS contract number F19628-95-C-0235, in part by ARPA (via USAF Rome Laboratory) under contract F30602-94-C-0042 and in part by NSF Multimedia CDA-9502639. This material is also based on work supported in part by the National Science Foundation, Central Intelligence Agency, Depart of Defense (DARPA) and National Security Agency under grant number IRI-9619117. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsor.

has low storage overhead. Good retrieval results are obtained with multi-colored query objects even when they occur in arbitrary sizes, rotations and locations in the database images. The overall scheme is flexible so that one or more phases could be used for retrieval depending on the precision desired. The experimental results on a database of advertisement images highlights the capabilities of the system.

1 Introduction

With the advent of large image databases, efficient content-based retrieval of images has become an important issue. The aim of content-based retrieval is to find images in a database which contain objects same as or similar to the object represented in a query image. The simpler problem of finding global similarity between a query image and candidate images has been addressed in [2, 4, 9]. However, in the more general case studied here, the queried object may be embedded in images which have nothing else in common apart from the presence of the queried object. This is a very complex task where the queried object may appear in candidate images in various sizes and orientations with a wide variety of background colors and forms as shown in Fig 1.

When the database has images of multi-colored objects with distinctive color signatures, the color of the object seems to be an obvious choice for indexing. There are many examples of such databases - flags, logos, consumer products, textile patterns and postal stamps among man-made objects and

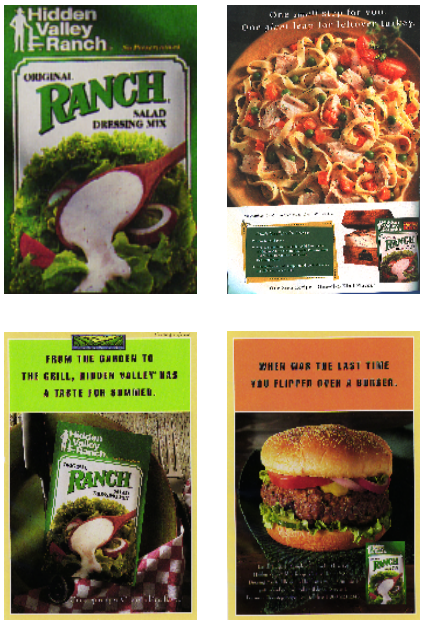


Figure 1: Example of a query image (top left) and correctly retrieved images

flowers, birds, fishes and butterflies in the natural domain.

We have chosen a database of advertisements from magazines to demonstrate the effectiveness of our method. The goal is to retrieve all advertisements containing the object depicted in the query.

Unlike other databases on which color-based retrieval has been tried (flags, logos, products), in most of the advertisement images, the products do not spatially dominate the image, nor are they necessarily in the center. There is no concept of foreground and backgound - what is backgound clutter for this application may actually be the foreground of the image. So,

no focus of attention based pre-segmentation is possible. In addition, most advertisements include natural backgrounds that do not provide sharp color boundaries. In the rest of the report, background refers to all objects in the image which are not a part of the queried object.

This choice of domain also gives us some advantages. Occlusion of the product is rare and the same aspect of the product is presented in all advertisements. There may be small out-of-plane rotations causing some occlusion, but all major colors of the object remain visible. There is very little color distortion across different advertisements of the same product. However, the color constancy problem still arises because light and shadow effects in images create differences in the apparent color of objects. Fortunately, the color variations are not severe and can be handled by the selection of a robust color representation and by allowing some tolerance in the matching strategy.

2 Previous Work

There has been some work in color-based retrieval in the last six years. The earliest significant work in color-based retrieval is by Swain and Ballard who used *color histograms* for indexing [2]. They introduced the *histogram intersection* technique for matching the histograms of a query image and a candidate image. More efficient histogram indexing strategies have been developed

in [6]. Though histogram matching is efficient, it cannot handle differences in scale between the query and candidate image, color shifts due to differences in illumination, out-of-plane rotations of the queried object or the presence of similar colored objects in the background. Since the size, orientation and background in which the queried object appears in a database image are not known in general, these drawbacks limit the use of this approach in databases with complex images.

The color structure of an object is described well by the *color adjacency graph* (CAG) representation proposed by Kittler *et al* [7], where each node in the graph represents a distinct color region and edges indicate the adjacency relations between the regions. Unfortunately, CAG construction requires distinct boundaries between colors which are uncharacteristic of most natural objects. So, CAG construction is not possible where the queried objects occur with natural objects in the background. This representation is not suitable for online retrieval as the matching speed is slow due to the complexity of the CAGs.

Some methods use features in addition to color histograms [4], but these require manual annotation by the user during offline processing. The histogram cluster-based matching described by Kankanhalli *et al* [9] is very efficient but it does not handle the presence of interfering backgounds in an image.

Thus, even though color has been recognized as an important tool in content-based retrieval, fast color-based retrieval strategies which can handle complex, interfering backgrounds are not yet available. Our work is motivated by the need to develop an effective retrieval strategy which deals with this problem for large databases. We need a description of a multi-colored object which will keep the retrieval process fast but will be general enough to handle the differences in the appearance of the object in an image and the presence of significant background clutter.

3 Overview of the system

The design concept behind the system is to have multiple layers of processing involving increasing amounts of computation and resultant precision. The aim of the first phase is to retrieve a set of possible candidate images as fast as possible. Subsequent phases improve precision by filtering out false matches from the image list generated by the first phase. Since each subsequent phase needs to work on the image list produced by its previous phase and not the whole database, more precise and time-consuming matching algorithms can be used. The user has the option to view the retrieved results after any phase. This allows the system to handle applications where speed is more important than precision and vice-versa.

At present, we have implemented a two phase retrieval system as shown in Fig 2. The first phase quickly extracts images which have the signature colors of the query object. This phase uses color histogram peak matching.

During offline processing, histogram peaks are computed for each image in the database and stored in an index table. At query time, the query image peaks are used as *keys* in an efficient indexing scheme. The first phase is fast, requiring only about 0.5 sec[1] on a database of 300 images. At the end of the first phase, a ranked list of images is generated which contains, in the best case, all the images that match the query and a number of false matches. For each image on this list, correspondence between the query peaks and the image peaks is noted and used in the next phase.

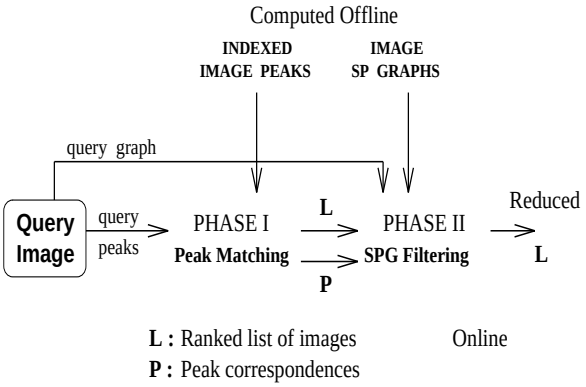


Figure 2: System Overview

The number of false matches depends on the number and uniqueness of

[1]on an SGI Onyx/Extreme with two 200 MHz R4400 processors

the colors of the queried object. Further processing may not be neccessary for objects with unusual colors or many distinct colors due to the small number of false matches, but for most objects using the presence of peaks as the sole match criterion may result in some false matches which can be filtered out using more information about the color structure of the candidate image. We use neighborhood information between color regions in the second phase. Although actual color adjacency is difficult to compute, spatial neighborhood relations between color regions can be computed to produce much smaller *color neighborhood graphs* (CNG). The second phase checks the query object CNG against the reduced CNGs of candidate images returned by the first phase. A restricted form of subgraph isomorphism is used to filter the false matches from the images retrieved by the first phase. This phase is also fast enough (about 1 second for checking 20 retrieved images) for use in retrieval schemes with an online user interface. The color information extracted offline for each database imgae is shown in Fig 3.

More details of the first two phases are provided in the next two sections. Additional phases can be added to the system if greater precision is desired by using more of the information present in the images.

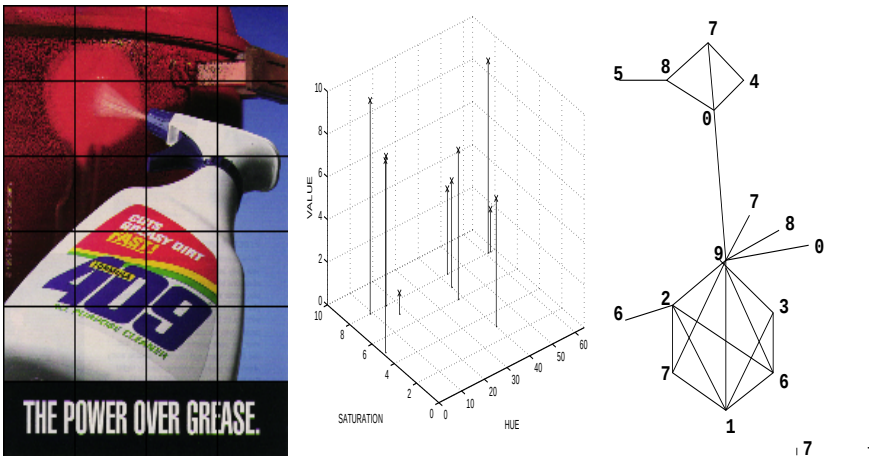


Figure 3: Information extracted offline from an image (i) Peaks in quantized HSV space (ii) Color Neighborhood Graph

4 Histogram Peak Detection and Matching

As observed in [7], the *locations* of peaks in a histogram are stable under viewpoint change and scale transformation. Histogram bin *counts* used in [1], on the other hand, are sensitive to these factors. If the object colors were not present anywhere in the background, the ratio of the histogram bin counts of peaks corresponding to the query object colors would also be invariant to scale transformations. However, this assumption cannot be made for this or any other general database. In fact, even the peak locations are affected by the presence of background in the image and methods have to be developed to minimize this effect so that peak locations can be used as

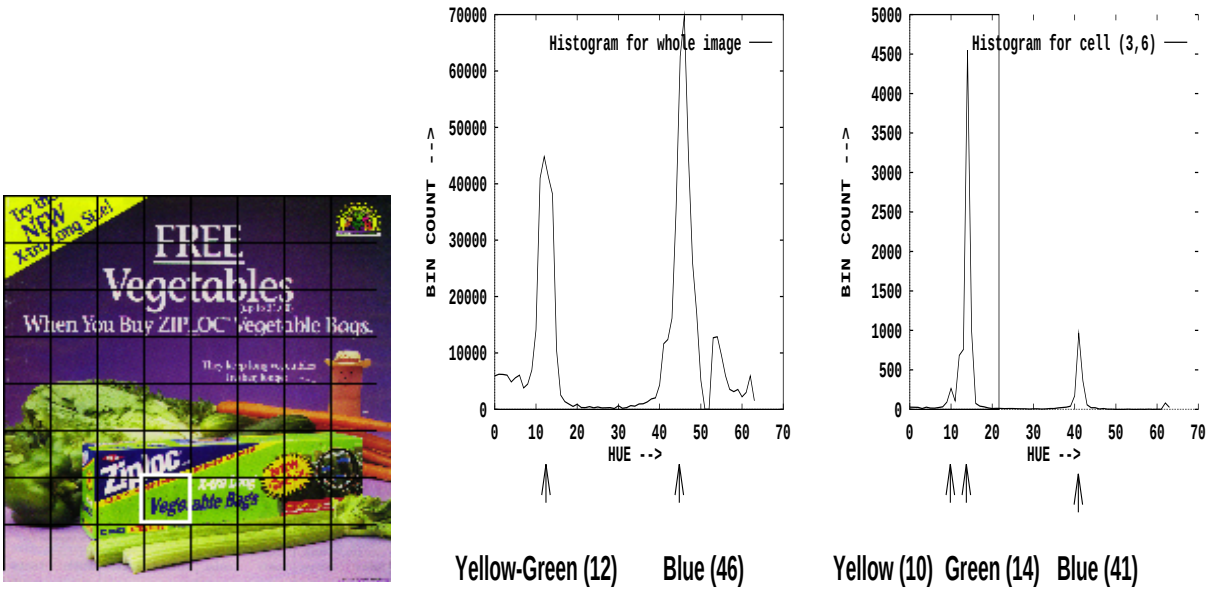
accurate descriptors of the color content of an image. We use a *split and merge* technique for peak detection which produces accurate peaks inspite of the presence of interfering backgounds.

Peaks are detected offline from 3-D color histograms of the images in the database. Peak detection requires a number of practical choices to be made. The histograms should be constructed in a color space which is as stable as possible under differences in lighting conditions and the histogram bin size should be able to distinguish visually different but similar colors. For this work, we use the HSV (hue, saturation, value) color space, since it is more stable than RGB under variations in illumination. In addition, Since the hue component is the most stable and value component the least stable, we use HSV histograms with finely discretized hue axis and coarsely quantized saturation and value axes (64x10x10).

When constructing histograms, all "grey" (ranging from black to white) pixels in the image are ignored since these pixels map to arbitrary hue locations in HSV space. Very little discrimination power is lost because grey pixels are present in almost all images, usually as background or text.

There are two conflicting issues to be considered in deciding the peak detection strategy. On the one hand, all colors in the image should be represented accurately and on the other hand the number of peaks should not be large as this would make the match less robust and increase the size of the

neighborhood graph used in the second phase.



(a) (b)

Figure 4: Shift in the hue component of histogram peaks due to the presence of interfering background (a) global hue histogram of the "Ziploc" advertisement on left (b) hue histogram of cell (3,6)

The histogram peak location of a query object color can be masked or shifted in a global histogram in the presence of background objects which are larger than the query object and have similar colors. Peak shifts can also occur when the background has gradually varying colors which overlap the query object colors. For example, Fig 4 shows the global hue histogram of an image of a "Ziploc" bag along with various vegetables which are different

shades of green and yellow. The histogram of cell (3,6) which covers the "Ziploc" package only, shows the actual peak locations of the colors present on the package. These peaks are lost in the global histogram, being subsumed by the colors in the background. To avoid this problem, we use a *split and merge* strategy for peak detection. The image is split into m x n cells. Local histograms are constructed in these image cells and peaks are detected in the local histograms. The *split* step reduces the area of the image covered by the histogram to a small locality and thus reduces the number of colors present in each cell and the chance of interference between colors. Due to the smaller number of pixels used in the histogram construction, even small peaks become significant and can be detected. The peaks obtained by this method describe the various colors present in the image more accurately than global histogram peaks. For this database, a cell size of 100x100 pixels was used.

Splitting the image has another major benefit which is used to advantage in the second phase of matching. Each peak is spatially localized within a small area of the image. If the union of all peaks detected over the image cells is considered, the total number of peaks would be large (100-200), with many identical peaks generated from the same color area split across cells. The *merge* step leaves just one copy of each peak in the final peak list for the image. Each peak in the final list of an image is assigned a color label

unique in that image.

The histogram peaks are detected by finding *local maxima* in a 3-D neighborhood window passed over the 3-D histogram. This approach produces more peaks than the histogram clustering method [8] but all distinct colors are more likely to be correctly represented. Though the histogram clustering approach [9] produces fewer clusters, interfering background colors will produce large clusters where the cluster means are not representative of the whole cluster. Since the query object colors are not known a priori, it is necessary to have peaks representing every distinct color in the image.

During the matching process, candidate images which have peaks matching all the query colors are extracted. A perfect match is detected when the candidate peak falls within a small tolerance window around the query peak. The tolerance window is asymmetric - smaller along hue and larger along value, reflecting the different sensitivity of each component to variations in illumination. A tolerance window of $(h \pm 1, s \pm 2, v \pm 4)$ is found to be suitable for this database. It is desirable to consider matches between peaks even when they do not match exactly so that the matching process is more robust to variations in color and degrades gracefully. Any peak beyond the tolerance window produces an *inexact match* with some *penalty*. The penalty is given by the *city block* distance between the candidate peak and the nearest perfect match. All match penalties greater than four are treated

as *mismatches*.

5 Indexing and Retrieval

The aim of indexing is to limit the search process to the images in the database which can be possible candidates for retrieval instead of trying to match every image in the database with the query. The index table is constructed in the form of a *three-dimensional grid* of the same dimensions as the discretized HSV space (64x10x10) with each cell corresponding to a HSV value. This format makes it possible to directly retrieve the contents of each cell indexed by the HSV value of a query peak. An example of a cell in the index table is shown in Fig 5.

Content of cell (h1, s1, v1) of index table

Image Number	Match Penalty	Matched Pk Label
m1	**k0**	**b0**
m12	**k1**	**b1**
m43	**k2**	**b2**
.	.	.
m241	**k4**	**b4**
.	.	.

Figure 5: A cell in the 3-D index table

Each cell (h, s, v) has a list of triples (n, p, l) where n is a unique image identifier, p is the match penalty of the peak in n closest to (h, s, v) and

l is the color label of the peak which matched (h, s, v) in image n. Image identifier n is present in cell (h, s, v) only if $p \leq 4$. The list is stored in increasing order of n.

The number of occurances of each peak for the entire database is counted and stored in a *frequency table* to be used to prioritize the order in which query peaks are matched because the frequency of occurance is inversely proportional to the discriminatory power of the peak.

The peaks in the query image provided by the user are ordered by increasing frequencies from the frequency table stored offline. The query image is assumed to have no background, so all query peaks need to be matched. The indexing starts with the list of triples from the index table at the first (most rare) peak of the ordered query peak list onwards. The next rarer peak from the query peak list is then selected and the list for this peak is read from the index table. An intersection or *join* is performed on lists from two consecutive query peaks.

The *join* step performs three tasks as illustrated in Fig 6 :

- Finds the image identifiers common to both lists and retains only these identifiers in the joined list. In Fig 6, image numbers **n1** and **n4** are common to lists for both query peaks P and $P+1$. This process is fast because both lists are sorted by the image identifiers.

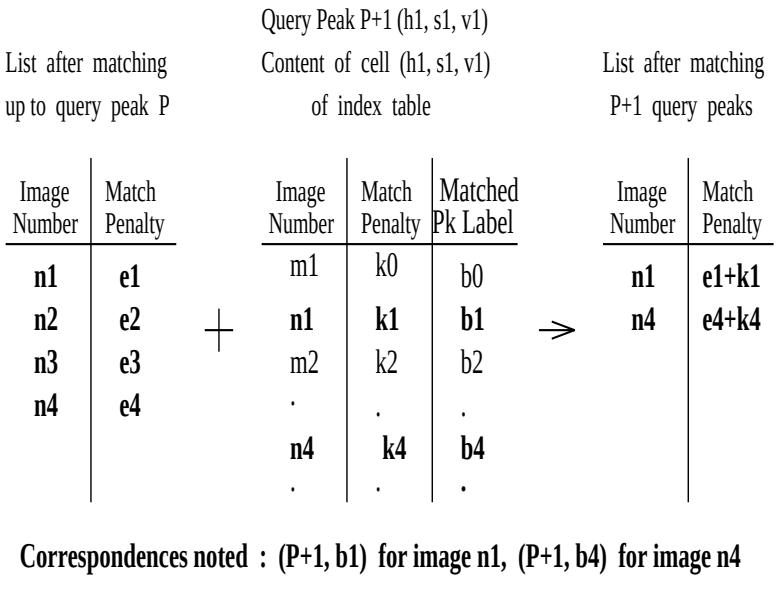


Figure 6: A step in the indexing process

- Updates the match penalty for the retained images by *adding* the penalties in the original lists. In Fig 6, the match penalties for **n1** and **n4** are updated to $e1 + k1$ and $e4 + k4$ respectively.

- Notes the correspondence between query peak and image peak label. In Fig 6, query peak $P + 1$ matched the peak labelled $b1$ in image **n1** and $b4$ in image **n4**.

The smaller list obtained in the above step is then intersected with the list from the next peak in the query list. The match penalty is cumulative and reflects the degree of match for all the peaks upto the current peak. This process is continued until all the peaks in the query list have been exhausted

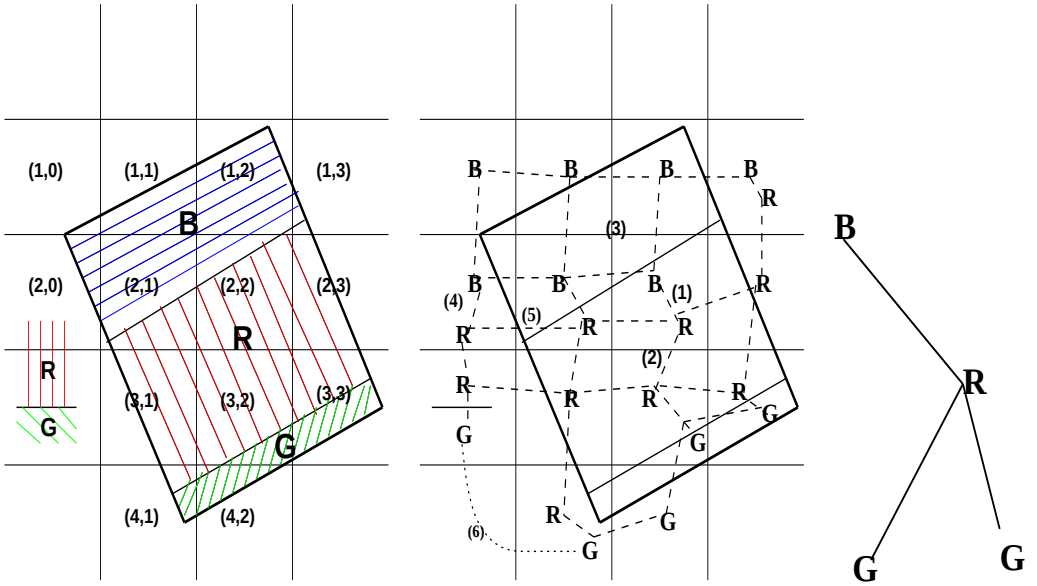
and a final list of images has been obtained. The final list is then sorted in increasing order of match penalties. This ranked list is the output from the first phase. For each of the images in this final list, the correspondence between the image color labels and query peak color labels is stored.

6 CNG construction and Matching

Of the false matches retrieved after the first phase, there will be a large number of images in which the colors of the query object are scattered across the image and do not form a spatially coherent cluster. These false matches could be eliminated if color adjacency information was used. Two color regions are adjacent if they share a boundary at the pixel level. Although actual color adjacency is difficult to compute in the presence of natural objects and colored backgrounds, approximate 'neighborhood' information between color peaks can be obtained from the spatial localization of the peaks into particular cells obtained during the peak detection process. Two regions are considered neighbors if they are spatially close and could be adjacent.

The difference between color adjacency and 'neighborhood' as used in this context is in the level of confidence that can be attached to the adjacency relation. Regions which are adjacent are always neighbors but there may be some neighbors which are not actually adjacent color regions in the

original image. This approximation is handled by using sub-graph isomorphism during the matching phase instead of exact graph matching. However, to maximize the number of false matches removed, the aim of neighborhood computation is to approximate color adjacency as closely as possible by eliminating edges where no adjacency is possible due to the configuration of peaks detected.



(a) (b) (c)

Figure 7: Example of CNG construction (a) Synthetic image divided into cells (b) Cells marked with nodes (peaks) contained in them. The edges between the nodes are shown in broken lines (c) Final CNG after merging

Fig 7 shows how the approximate neighborhood relations between the

color peaks is condensed into a compact graph - the *color neighborhood graph* (CNG). Fig 7 (a) shows two synthetic objects with three color regions (R,G,B). The peaks detected in each cell are shown in Fig 7 (b) by including the peak color label within the cell. These peaks form the nodes in the intermediate CNG. The edges in the intermediate CNG indicate that the two peaks could be from adjacent color regions in the original image. The edges marked are generated from the following observations.

- Identically labelled nodes in adjacent cells could be a part of the same color region in the image and therefore are connected by an edge. An example of such an edge where the two nodes connected are from the same region is labelled (2) in Fig 7 (b). The edge labelled (5) shows a case where two nodes are connected as neighbors but are actually not from the same color region.

- When two nodes occur in the same cell, they could be from adjacent color regions in the original image, so they are connected by an edge e.g. edge labelled (1). Some nodes may be connected which are not actually adjacent but we cannot determine the exact adjacency relation within a cell without pixel level processing. The edge marked (4) in Fig 7 (b) is an example of such an edge.

- Diagonal edges are not considered because they would be redundant

e.g. label (3), and may add some edges where no adjacency is possible e.g. label (6).

Putting the above discussion concisely, let a_m^i be a node in the intermediate CNG where m is the label of node a and i is the cell in which a is located. There is an edge $E(a, b)$ between nodes a and b of the graph if the following condition is met.

- $E(c_m^i, c_n^j)$ if $i = j$ OR $m = n$ and (i, j) are 4-neighbors.

This reasoning fails only if the color boundaries of the query object are perfectly aligned with the cell boundaries, which is a very unlikely occurance. This problem is handled by using overlapping cells.

The intermediate graph obtained above is not suitable for use in graph matching to detect false matches because it is not scale invariant. If the same object is larger or the cell size is smaller, there would be more nodes of each color label added to the graph. To eliminate this dependence on scale, identical labels which are connected by an edge are merged into a single node as shown in Fig 7 (c). This produces the final CNG which is much smaller and scale and orientation invariant. Information about the spatial extent of the object is lost but the color neighborhood information remains intact. The CNG may still have multiple nodes of the same color label, but only if these peaks were spatially disconnected in the image. The graph is computed

offline for each of the database images and stored using an adjacency matrix representation.

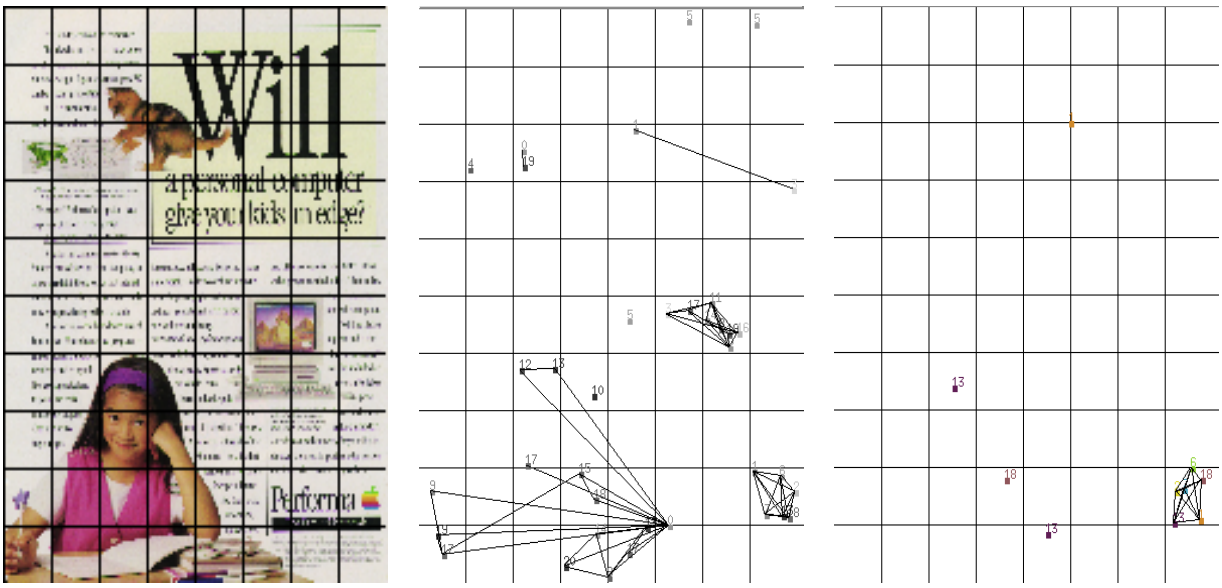


Figure 8: Reducing CNGs by deleting nodes not matched in phase 1 (left) Original image (mid) CNG stored offline (right) CNG left after reduction

The CNG stored offline is still quite large because most of the advertisement images have a large number of distinct colors. However, it should be noted that at the end of the first phase of retrieval, the correspondence between color labels in the image and the query peaks are available for each image in the retrieved list. So, the nodes in the CNG which have labels not matching any of the query peaks can be deleted from the graph as they are not part of the queried object. Fig 8 shows the drastic reduction in the CNG

when only nodes which matched a query peak are considered. The query in this case was the "Macintosh" logo. The color label of the nodes in the reduced image CNG are replaced by the query peak number they matched using the correspondence available from the first phase of matching. This reduced, labelled CNG is used for graph matching.

(a) (b)

Figure 9: Example of CNG mismatch (a) "Blueberry Morning" query image with CNG superimposed (b) A false match with CNG superimposed

An example showing the labelling of the query CNG and the reduced image CNG just before the graph match is computed is shown in Fig 9. At this point, both the graphs are labelled using the same labels - the query peak number. For clarity, the labels have been placed in the region of the image where the peaks were detected. We can see here that though the peaks labelled 0-4 are present in both query and image, the spatial neighborhood

relationships are not identical. Peaks 0 and 3 are connected in the query graph but not in the image graph.

Fig 14 shows an example of a true match and a false match when the query object CNG is matched with the reduced CNG of an image.

The query CNG is a sub-graph of the reduced image CNG. Detecting a given sub-graph in a larger graph is the well-known subgraph isomorphism problem which is known to be NP-complete. However, due to the restricted nature of our problem where the reduced CNG nodes are labelled, the matching computation is feasible. The running time is of the order of $O(n^m)$ where n is the size of the query adjacency matrix and m is the number of nodes of each color label in the reduced image CNG, typically less than 3. Without labelling, m would be the size of the reduced CNG adjacency matrix which is in the range of 10-100.

The search is further reduced by starting the matching process with the query peak which has the minimum number of copies in the reduced CNG of the image.

7 Query construction and processing

The query is an image of the object which needs to be detected in the database images. It can be constructed by taking a sub-part of one of the

database images or can be provided from some other source. The query image should not contain any background colors but it is not necessary to include the whole object exactly - including the salient colors of the object is sufficient. Example query images can be seen in Fig 15.

The processing required on the query image includes the detection of query peaks and the construction of the query color neighborhood graph. The techniques used for both these computations is somewhat different from the techniques used offline on the database images for the reasons explained in the following paragraphs.

The histogram from the whole query image is used for peak detection. The reasons for using split and merge histogram peak detection technique do not apply here. There are no background colors and the peaks which are too small to detect in the query image itself would be even more difficult to detect in the database images and should not be included. Also, the query image may be quite small making it infeasible to sub-divide it. The CNG construction also does not use the localization of peaks in cells, eliminating the need for sub-dividing the image.

If the CNG was constructed from peak localization in cells, the scale invariance property would be lost. In a small query image, two peaks could be in the same cell and thus be connected by an edge in its CNG. However, when a bigger copy of the query object appears in a database image, these

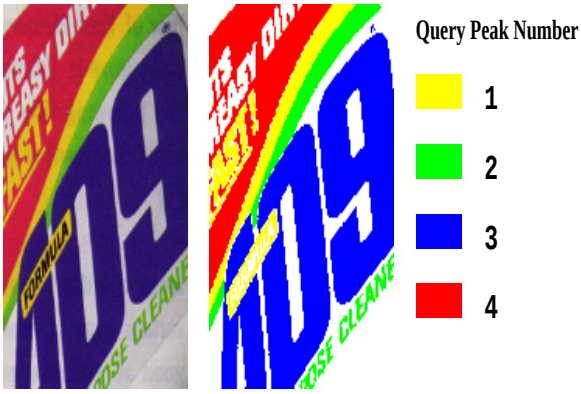
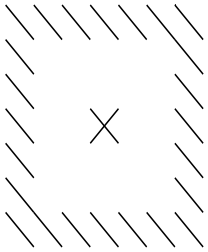


Figure 10: Labelling the query image pixels with the peak colors (left) query image (right) labelled image

two peaks could now be in different cells with no edge between them. This would create a mismatch with the query CNG. So, we need the exact color adjacencies from the query image. This is obtained by using pixel adjacencies described below.

After the peaks in the query image are detected, the pixels in the query image are labelled by the peak number under which they can be classified as shown in Fig 10. The labelled query image is scanned and a color adjacency matrix is computed by counting the number of neighbors of each color for each of the pixels as shown in Fig 11. There are regions of intermediate or mixed color at the boundaries between two colors even though they may not be visible when the image is viewed. To take care of boundary effects, a mask of the type shown in Fig 11 (a) is used to find neighbors of the center pixel.



	1	2	3	4
1	9292	381	1	150
2	1730	10436	42	0
3	2	82	10042	0
4	59	0	0	27301

	1	2	3	4
1	**1**	**1**	**0**	**1**
2	**1**	**1**	**1**	**0**
3	**0**	**1**	**1**	**0**
4	**1**	**0**	**0**	**1**

(a) **(b)** **(c)**

Figure 11: Constructing the adjacency matrix from pixel neighborhoods (a) A typical neighborhood mask (b) Table constructed from "409" query (c) Adjacency matrix obtained from table

8 Results

The database has 300 images of various sizes upto 8"x11" scanned at 100dpi resolution. The overhead incurred in storing the information extracted offline is about 3% of the size of the database. The average number of target images for each query image is 3.3.

The retrieval results obtained by this system can be judged by the criteria used in text retrieval, *precision* and *recall*. Precision is the proportion of correct retrievals in the images retrieved upto the last correct image. Recall is the proportion of correct retrievals out of all the images in the database that should have been retrieved for the given query. On a query set of 25, the

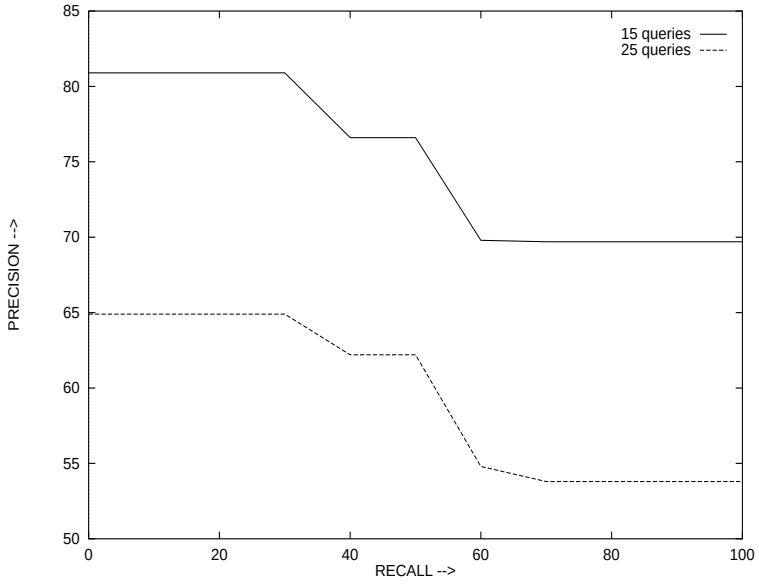


Figure 12: Recall-Precision graph after Phase 2 for a set of 25 randomly selected queries and 15 queries with more than three colors each

recall was 95%. The average precision after phase 1 was 42% and after phase 2 it improved to 60%. The recall-precision graph obtained after phase 2 is shown in Fig 12 (a). In terms of this database, which has 3.3 target images on an average, a precision of 60% means that one needs to view only the top 6 images on an average to see all the target images. The performance was better when the query had more than three colors. The average precision score after phase 2 in this case was 75%. The recall-precision graph for this case is shown in Fig 12 (b).

Some of the retrieved results are shown in Fig 16 for the queries shown

Name	Recall	Prec 1	Prec 2
Breathe Right	3/3	3/3	3/3
L'oreal Casting	4/4	4/7	4/5
Comet	2/2	2/23	2/9
Dannon	3/3	3/5	3/5
Fresh Step	2/2	2/3	2/2
Hidden Valley	7/7	7/20	7/15
Macintosh	2/2	2/2	2/2
Merit	6/6	6/18	6/12
Reynolds	4/4	4/13	4/6
Sun Crunchers	2/3	2/13	2/8

Figure 13: Retrieval results for 10 queries : (Recall) Images retrieved/No. of correct images in database (Prec 1) Precision after Phase 1 (Prec 2) Precision after Phase 2

in Fig 15. These results show that good results are obtained even when the query object is present in different sizes, orientations and with different backgrounds in the candidate images. Fig 13 shows a table for some of the other queries of the query set. These results and results on other queries can be viewed in the online demonstration of the system which can be accessed from `http://hobart.cs.umass.edu/~ mmedia`. The user can query the database and view the retrieved images after the computations are performed online.

9 Future Work

Though the results obtained using this retrieval strategy are very promising, we would like to try the system on much larger databases of atleast 1000 images to evaluate its performance more critically.

Some aspects of the system need additional work to be able to handle larger databases. Currently, the retrieved images are ranked by increasing match scores which are computed using a simple city block metric. This results in many images with identical scores. In a large database, the number of retrieved images in the first phase will definitely increase, so the ranking mechanism is crucial in keeping the images of interest in front of the list. A more sophisticated ranking strategy which uses the ratio of the color peak sizes *after* the second phase (when the object of interest has been localized in the image) is being developed. In this case, the images would be re-ranked, based on the new information obtained during the second phase computation.

The other aspect which could be improved is the indexing strategy. In the current indexing scheme, the number of entries in each index cell could increase linearly with the size of the database. A modification of the hierarchical R-tree based indexing strategy [8] is being developed which is logarithmic in the size of the database. There is also a possibility of using indexing during the graph matching phase as well which will speed up the

second phase. A third phase could also be added which computes the approximate size, location and orientation of the object detected. This would not be needed for a human user but could be used to feed information to an automated system as a focus-of-attention mechanism.

This system was optimized to work on the database of advertisements, but the color attributes used for matching could be used for other databases with multi-colored objects. One of the extensions of this work would be to generalize this method to other domains, selecting system parameters based on the color characteristics of the database images.

10 Conclusion

In this technical report, we have presented a fast, background independent color image retrieval system which produces good results with multi-colored query objects. The retrieval is robust to differences in the scale, orientation and location of the query object in candidate images and degrades gracefully in the presence of color variations. The speed of the system and the small storage overhead makes it suitable for use in large databases with online user interfaces.

Acknowledgements

We would like to thank R.Manmatha for his valuable suggestions regarding this work and Jonathan Lim for building the web-page for the online demonstration of the system. This work is a part of the Multimedia Indexing and Retrieval group of Center for Intelligent Information Retrieval, UMass and the Computer Vision Laboratory, UMass.

References

[1] M.J.Swain and D.H.Ballard, "Indexing via Color Histograms", *Third International Conference on Computer Vision*, pp 390-393, 1990.

[2] M.J.Swain and D.H.Ballard, "Color Indexing", *International Journal of Computer Vision, Vol. 7:1*, pp 11-32, 1991.

[3] M.A.Stricker, "Bounds for the Discrimination Power of Color Indexing Techniques", *SPIE Conference on Storage and Retrieval for Image and Video Databases II, Vol. 2185*, pp 15-24, Feb 1994.

[4] W.Niblack, R.Barber et al, "The QBIC project: Querying Images by Content using Color, Texture and Shape", *SPIE Conference on Storage and Retrieval for Image and Video Databases , Vol. 1908*, pp 173-87, Feb 1993.

[5] R.Barber, W.Niblack et al, "Ultimedia Manager: Query by Image Content and its Applications", *Spring COMPCON 94*, pp 424-429, 1994.

[6] J.Hafner, H.Sawhney, W.Niblack et al, "Efficient Color Histogram Indexing for Quadratic Form Distance Functions", *PAMI, Vol. 17, No. 7*, pp 729-736, July 1995.

[7] J.Matas, R.Marik and J.Kittler, "On Representation and Matching of Multi-Coloured Objects", *Fifth International Conference on Computer Vision*, pp 726-732, June 1995.

[8] G.P.Babu, B.M.Mehtre and M.S.Kankanhalli, "Color Indexing for Efficient Image Retrieval", *Multimedia Tools and Applications, Vol 1, No. 4*, pp 327-348, Nov 1995.

[9] M.S.Kankanhalli, B.M.Mehtre and J.K.Wu, "Cluster-Based Color Matching for Image Retrieval", *Pattern Recognition, Vol. 29, No. 4*, pp 701-708, Apr 1996.

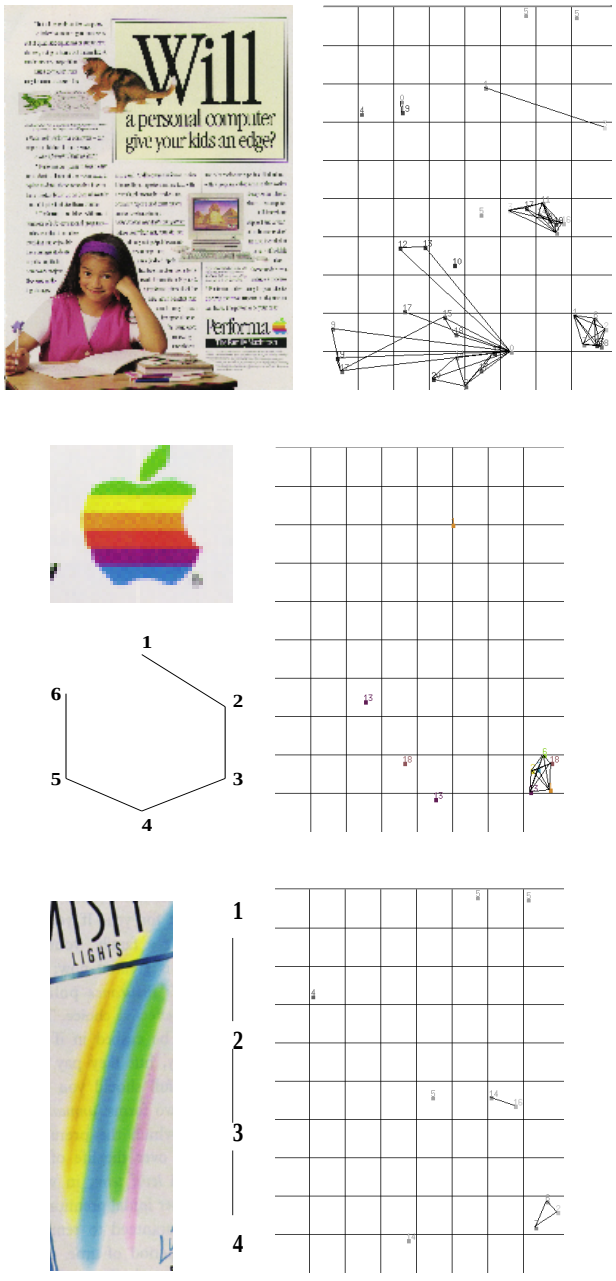


Figure 14: Matching CNGs (top) Original image and CNG stored offline (mid) Query, query CNG and reduced CNG of above image for a match (bot) Query, query CNG and reduced CNG of above image for a mismatch

Figure 15: Query images for results shown in Fig 16 (left to right) “Blueberry Morning”, “409”, “Misty”

(X)

(X)

(X)

(X)

Figure 16: Retrieved results on three queries shown in Fig 15 after phase 1 (top) “Blueberry Morning” (mid) “409” (last) “Dannon”. (X) indicates false matches that were removed by phase 2