Hierarchical Clustering for Image Databases

Sanjiv K. Bhatia
Department of Mathematics & Computer Science
University of Missouri – St. Louis
St. Louis, MO 63121
sanjiv@cs.umsl.edu

Abstract

The organization of an image database is one of the important issues in efficient storage and retrieval of images. Most of the existing image databases are based on flat structures, with the possibility of an index into the database that can help in narrowing down the images to be searched. In this paper, I'll present a technique to create a hierarchical data structure based on the clustering approach such that a user can select or discard a number of images for subsequent operations. The presented technique is based on application of wavelet analysis to scale the images in hierarchy, and can take advantage of the structure of compressed images in the JPEG *2000 standard.*

1. Introduction

An image can be abstracted as data in multiple dimensions, defined by $I = f(x,y)$, where $f(x,y)$ indicates the value of a discrete pixel at spatial coordinates (x,y). This abstraction implies that there is no easy way to impose an order on a set of images based on content. However, an image can be manipulated in a number of ways to make the data contained in the image provide more information that can be used to create an index that can be searched readily.

In a flat environment, an image needs to be compared to every other image in the database to determine the closest match for retrieval or other operations. This leads us to a linear search algorithm but with the caveat that each comparison involves comparing a number of pixels in each image – data in each dimension of our multidimensional abstraction – to compute a coefficient of match. This structure was used in the celebrated QBIC project [1, 7, 8]. The indexing algorithms reduce the computation time of each comparison but still, require a comparison with a representation of every image, yielding a linear algorithm with improved efficiency due to working with a computationally efficient structure rather than raw images [3, 6]. There is a need for a technique that can reduce the number of comparisons while still performing the search in an efficient manner. One of the techniques successfully employed to achieve such an objective in multidimensional data is provided by clustering.

Clustering is defined as an organization technique where the objects of interest, that have some similarity along a dimension of interest, are kept close together while the objects that differ from each other, are kept further apart. The dimension of interest is defined based on the application. In case of images, we abstract the dimension of interest as the images that have similar color distributions in corresponding areas. This may make the images perceived as similar to each other in appearance in perceptual terms but the images could be farther apart based on actual color distribution. As an example, consider the images presented in Figure 1 which are similar in form but are very different if we look at color distribution. Comparing these images poses yet another challenge in clustering.

Clustering results in improved performance for

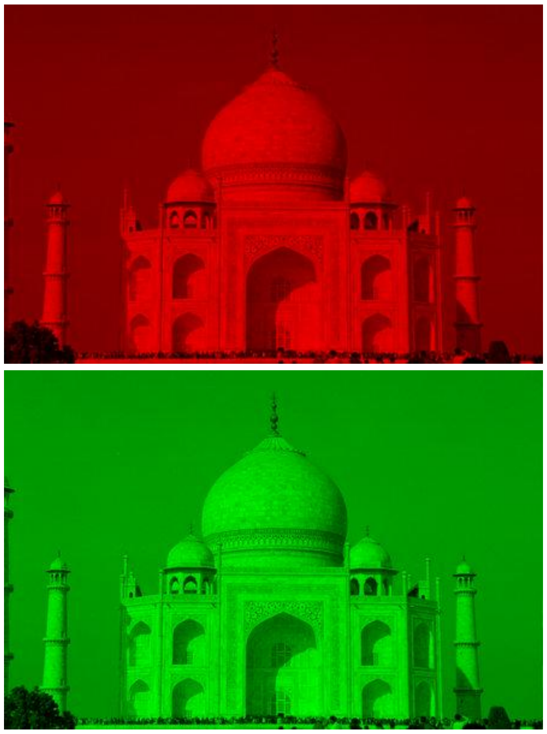

Figure 1. A picture of Taj Mahal in red and green bands

search and retrieval as the system can select or discard a number of objects based on a limited number of comparisons with the items being searched for. A hierarchical cluster goes one step further by collecting similar clusters at different levels of detail into a single cluster using a multi-branch tree structure. Such a system allows for selection of a set of clusters for further exploration.

In this paper, I present the use of a hierarchical clustering data structure for image database organization. This data structure has evolved from an adaptive clustering scheme for multidimensional data that has resulted in a search time of $O(\sqrt{n})$ in an average case over n multidimensional objects [4]. I have successfully used this technique to achieve performance gains in assignment of thermal properties to composite material data for FLIR simulation as well as to generate a color table in which the colors have been uniformly distributed in a perceptual sense. In this paper, I'll show that clustering can be applied to image databases to achieve similar gains in efficiency.

In the next section, I'll present a review of the existing literature on image database organization. In Section 3, I'll present the use of hierarchical cluster data structure to organize the images. This will be followed by some examples and conclusion in the last section.

2. Organization of Images in Databases

Most of the image database systems can be characterized as *content-based image retrieval* (CBIR) methods. These systems extract image characteristics, known as *signatures*, and perform reasoning on images based on some rules defined on image signatures.

Most of the CBIR systems can be classified into one of three categories broadly defined as histogram-based systems, color distribution-based systems, or region-based systems [9]. Some systems, such as QBIC [7], Walrus [9], and JPEG-coefficient indexing-based system [6] have characteristics of more than one category.

The histogram-based systems create a histogram of each of the three primary colors independent of each other. The number of bins in each color at different quantization level are tallied to create the signature of the image. The signature of an image can be compared against the collected signatures of images in the database to determine the closest match. The main drawback of this scheme is that there is no regard for the shape, texture, and object location in the image. It works only on the distribution of color *anywhere* in the image and give a high degree of match in semantically unrelated images.

The systems based on color layout perform somewhat better by dividing the image into a number of small blocks and using the average color of each block to create a signature for the image. Some systems, such as WALRUS [9] and WBIIS [12], use significant wavelet coefficients instead of average values to capture sharp color variations within a block. These systems show low tolerance for object translation and scaling in an image.

Region-based systems, including QBIC [7], Blobworld [2], and SIMPLIcity [11], use local properties of regions with the assumption that each region identifies an object. However, a problem can arise if an object is divided into multiple regions.

The system presented here considers a more holistic approach by using a query image at different resolutions to navigate to the images that match the query. The multiresolution nature of the data structure uses wavelets to get a mipmap-like scheme for each image. In the current implementation, I have used the Haar wavelet for its simplicity to derive multiple resolutions of the image and by throwing away the detail coefficients, retaining only the scale coefficients.

In the next section, I describe clustering phenomenon in relation to images followed by the use of scaling coefficients to create the hierarchical cluster data structure as well as the overall implementation of this structure.

3. Cluster Scheme for Images

The hierarchical cluster data structure is based on the development of a representation of each cluster that adequately captures the contents of objects contained in the cluster. This representation forms the *centroid* of the cluster. The centroid of cluster, in case of images, is simply the average image for the cluster where the average image is formed by taking the average value of corresponding pixels in the images that form the cluster. This condition requires that all images in the cluster be of the same size, or scaled to be the same size for internal matching.

To develop the clusters, consider all images to be randomly distributed along a two-dimensional plane. We develop clusters by looking at each image and adding it to the cluster that is closest in match to this image. Another point to be considered is that the quantitative measure of match is within a specified threshold from the centroid of the cluster. If we cannot find such a cluster, we create a new cluster that is made up of just this one image. Whenever an image is added to a cluster, its centroid may shift because of the change in average value of corresponding pixels due to the new image. An example of images divided into clusters is shown in Figure 2.

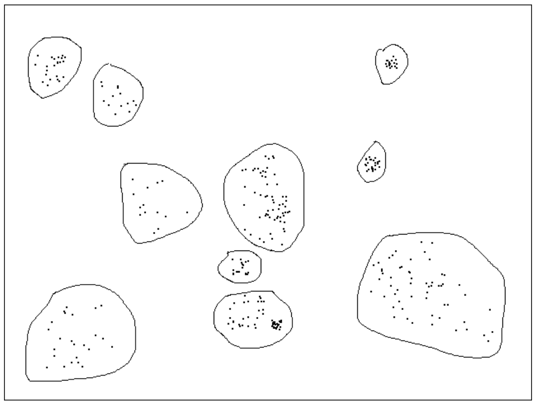

Figure 2. Clustered images in a 2D plane

The hierarchical cluster data structure needs a measure of distance between two given images. A simple definition of this distance is provided by the average difference in each of the red, green, and blue bands of the two images. If the two images of size $r \times c$ pixels are denoted by I and I', the distance d between them is given by

$$d = \frac{1}{3rc} \sum_{x=0}^{c-1} \sum_{y=0}^{r-1} \sum_{z=RGB} |I_{x,y,z} - I'_{x,y,z}|$$

The distance computed in this fashion is extremely simplistic and suffers from the problem illustrated in Figure 1. The problem is fixed by discarding the RGB color space in favor of a perceptual color space such as HSV or `L*a*b*`.

Whenever a new image is added to a cluster, the averaging for new cluster representation is performed based on the number of images already in the cluster. If a cluster represented by $\mathscr{C}$ contains n images, and I is a new image being added to $\mathscr{C}$, the new cluster representation $\mathscr{C}'$ is given by

$$\mathscr{C}' = \frac{1}{n+1}(n \cdot \mathscr{C} + I)$$

The first cluster representation is created from the first image itself. Thus the image is at the center of the cluster. As an image is added to this cluster, the new cluster representation is created by taking the average of two images in the cluster. This changes the location of centroid to the point in the middle of the two images. From this point on, the representation keeps on shifting but never goes further than the tolerance specified by the threshold due to the weighted averaging. Thus, the algorithm guarantees that an image cannot be classified to more than one cluster. Therefore, we can assign the image to a cluster as soon as a suitable match is found. Moreover, a corollary to this implies that two clusters cannot have an overlapping boundary.

3.1. Hierarchical Cluster

The clustering described above can degenerate into worst case scenario if the threshold is selected to be small. In such a case, each image is a cluster by itself. The above clustering algorithm performs a linear scan of the image against the clusters already produced. The only way to improve on the algorithm will be if we can compare a query image against clusters in order such that the cluster with maximum number of images is compared first and the one with least number of images is compared last. A better way to improve the algorithm performance is by using a wavelet analysis to create a hierarchy of clusters. The hierarchy is created in the form of cluster of clusters at multiple levels, generated by full averaging of images. This cluster of clusters organizes the images at different resolution levels in the form of a multi-branch hierarchical tree.

The hierarchical clustering algorithm performs the full averaging using Haar wavelet on the image that is to be clustered. It performs this analysis recursively till the entire image is reduced to a single pixel, saving the intermediate results in a stack. It should be recalled that at each step, the image is reduced by half in both height and width. Thus, a 256×256 ($2^8 \times 2^8$) pixel image will yield 9 levels of averaging, from $2^0 \times 2^0$ pixels to $2^8 \times 2^8$ pixels. Each hierarchy in the new data structure is thus $1 + \lg n$ levels deep where n is the larger of the number of rows and columns in the texture. As in the case of *flat* clusters, we will require each image in the cluster to be of the same size, or scaled to be of the same size.

The hierarchical cluster data structure is implemented by a recursively defined vector of clusters with each cluster represented by a single image, and containing a hook for another hierarchy below it. The image representing the cluster is simply an average of all the images that are contained in the cluster at that resolution level. The hook provides for the hierarchy starting at the next higher resolution level.

As an image I is added to the cluster, we create multiple resolutions of I by applying the Haar wavelet and discarding the wavelet or detail coefficients until we have an image of just one pixel, denoting it by I_0. This pixel captures the average value of all pixels in the image, in respective bands. This image is then compared with the vector of clusters at the root of hierarchical cluster data structure to determine the closest match.

Once we have found the matching cluster, and it is within the cluster radius, we select the cluster vector at the next higher resolution level that is associated with this cluster. This will lead us to the cluster vector of 2×2 pixel images. We compare the 2×2 version of current image I, denoted by I_1 to each member of this cluster vector. The closest match is then selected and we continue with the selection at the next level. The recursive application stops when we find the image that matches within a threshold limit, or when we discover that the image is outside the radius of any cluster at that resolution level within the selected cluster. At such a point, we stop and create a new cluster to add the image into the new cluster.

3.2. Results

We'll illustrate the results using the images categorized as *art* in the Smithsonian database. There are 31 images in this category and their representation is given in Figure 3.

I tested this set of images separately using 256 × 256 pixel images and created the lustering by comparing in `L*a*b*` color space. The hierarchy contains a number of clusters as shown in Table 1. The table shows the multiresolution nature of hierarchical cluster structure. At the lowest resolution level or root, each image is reduced to a single pixel that captures the average of all the color bands separately. At this level, the images were divided into four different clusters. As we go down the structure to higher resolutions, the number of clusters start to increase and by the time we arrive at level 7, all images are separated from each other and form a cluster by themselves.

Table 1. Number of clusters at different levels in hierarchy

Level	Picture size	Number of clusters
0	1 × 1	4
1	2 × 2	7
2	4 × 4	13
3	8 × 8	18
4	16 × 16	22
5	32 × 32	26
6	64 × 64	28
7	128 × 128	31
8	256 × 256	31

The system is implemented in C++ on a Sun SPARCstation running Solaris. The query for a matching image in this structure is completed in less than one second on an average once the cluster has been loaded into memory and the stack of multiresolution images created. In case the system does not find the exact match of the queried image, it returns the closest match.

4. Conclusion and Future Work

The hierarchical cluster data structure is implemented using Haar wavelet for scaling the images at different resolution levels. The system includes separate programs to create the clusters as well as to add individual images to the data structure. The distance measure uses RGB color space though the user can specify `L*a*b*` or HSV color spaces as command line options. The system has been tested with two separate databases.

The first test was performed with a database that contains about 4000 textures of size 256 × 256 pixels. In each query, I was able to satisfactorily retrieve an exact or approximate match to the query image. The current implementation uses the simplest form of wavelets – the Haar wavelets – to perform scaling at different resolutions.

The system has also been tested with Smithsonian database containing 747 images, divided into seven different categories. Each of these categories contains between 31 and 216 images. The images are scaled to 512 × 512 pixel size for internal use with our data structure, giving a depth of 9 levels in the hierarchy.

Currently, the system gives only one image as a response to the query. This image could be the exact match, if available in a cluster, or it could be the one that is at the smallest distance from the query image. One of the enhancements in the system will be to add the capability to output multiple images and provide a ranking to those images in the output.

The Haar wavelets perform a simple averaging of color values in different picture elements. They are simple to program, compared with other wavelets. I used them to prove a concept that they can help with hierarchical clustering. However, to create a robust environment, I believe I need to use a different type of wavelet. In future, I plan to replace the use of Haar wavelets with Debauchies 5/7 and 7/9 wavelets [5] to scale the images to create the hierarchical data structure that will further improve the retrieval process. I chose the Debauchies wavelets because they form the basis for the JPEG 2000 standard [10]. This selection leads me to believe that I may be able to work on images compressed with JPEG 2000 algorithm directly by extracting the scaling coefficients at different levels for use in the hierarchical cluster data structure.

References

[1] R. Barber, W. Equitz, M. Flickner, W. Niblack, D. Petkovic, and P. Yanker. Efficient query by image content for very large image databases. In *COMPCON*, pages 17–19. IEEE Computer Society Press, Los Alamitos, CA, 1993.

[2] S. Belongie, C. Carson, H. Greenspan, and J. Malik. Color- and texture-based image segmentation using the expectation-maximization algorithm and its application to content-based image retrieval. In *ICCV98: Proceedings of the International Conference on Computer Vision*, pages 675–682, 1998.

[3] S. K. Bhatia. Image database indexing using JPEG coefficients. In *Proceedings of the 10th FLAIRS Conference*, Daytona Beach, FL, May 1997.

[4] S. K. Bhatia. Adaptive *k*-means clustering. In *Proceedings of the FLAIRS 2004 Conference*, South Beach, FL, May 2004.

[5] C. K. Chui. *Wavelets: A Mathematical Tool for Signal Analysis*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1997.

[6] S. Climer and S. K. Bhatia. Image database indexing using JPEG coefficients. *Pattern Recognition*, 35(11):2479–2488, November 2002.

[7] M. Flickner, et. al. Query by image and video content: The QBIC system. *IEEE Computer*, 28(9):23–32, September 1995.

[8] B. Holt and L. Hartwick. Retrieving art images by image content: the UC Davis QBIC project. *ASLIB Proceedings*, 46(10):243–248, October 1994.

[9] A. Netsev, R. Rastogi, and K. Shim. WALRUS: A similarity retrieval algorithm for image databases. In A. Delis, C. Faloutsos, and S. Ghandeharizadeh, editors, *SIGMOD 1999: Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 395–406, Philadelphia, PA, June 1999. ACM Press.

[10] `www.jpeg.org`. *Coding of Still Pictures: JPEG 2000 Part I Final Committee Draft*, version 1.0 edition, April 2000.

[11] J. Z. Wang, J. Li, and G. Wiederhold. SIMPLIcity: Semantics-sensitive integrated matching for picture libraries. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23, 2001.

[12] J. Z. Wang, G. Wiederhold, O. Firschein, and S. X. Wei. Content-based image indexing and searching using Daubechies' wavelets. *International Journal on Digital Libraries*, 1(4):311–328, 1997.

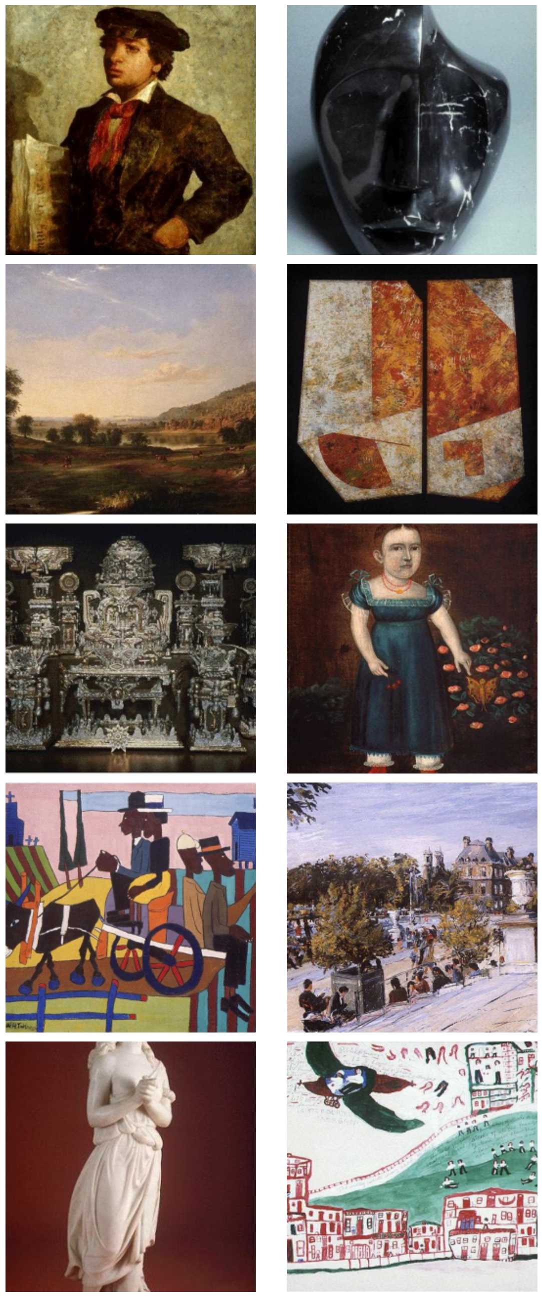

Figure 3. Representative images from art category in Smithsonian database