University of Alberta

# An Adaptive and Efficient Clustering-based Approach for Content Based Image Retrieval in Image Databases

by

Renato O. Stehling
Mario A. Nascimento
Alexandre X. Falcao

Technical Report TR 01-03
March 2001

DEPARTMENT OF COMPUTING SCIENCE
University of Alberta
Edmonton, Alberta, Canada

# An Adaptive and Efficient Clustering-based Approach for Content Based Image Retrieval in Image Databases

Renato O. Stehling[1]
renato.stehling@ic.unicamp.br

Mario A. Nascimento[2]
mn@cs.ualberta.ca

Alexandre X. Falcão[1]
afalcao@ic.unicamp.br

[1]Institute of Computing, Univ. of Campinas, Brazil
[2]Dept. of Computing Science, Univ. of Alberta, Canada

**Abstract**

In this paper, we present a new Content-Based Image Retrieval (CBIR) approach which is based on cluster analysis. A distinguishing aspect of CBIR is that it relies on a visual content representation (metadata) of the images. In order to produce such metadata, we propose an efficient and adaptive agglomerative clustering algorithm to segment the images into high-similarity regions. Each region is a connected component composed of pixels with a predefined degree of color similarity, thus avoiding the notorious problem of color-space quantization. This approach contrasts with those that use a single color histogram for the whole image (global methods), or local color histograms for a fixed number of image cells (partition-based methods). We compared the effectiveness among three configurations of our approach and five other existing approaches. The results show that our clustering approach offers high retrieval effectiveness with low space overhead, outperforming the other compared approaches. For example, we can reach higher retrieval effectiveness than partition-based methods with about the same space overhead of global methods, which are typically regarded as compact-storage-wise. Furthermore, the proposed approach is flexible in the sense that the user may easily calibrate the trade-off between space overhead and retrieval effectiveness.

## 1 Introduction

Cluster analysis [6, 10] is one of the most well-developed and commonly used forms of combinatorial data analysis. This area is characterized by methods concerned with the identification of homogeneous groups of objects. Clustering of similar objects into groups is an important human activity. Basically, one wants to form groups in such a way that objects in the same group (cluster) are similar to each other, whereas objects in different groups are as dissimilar as possible.

In this paper, we present a new *content-based based image retrieval* (CBIR) approach based on cluster analysis. A distinguishing aspect of CBIR is that it relies on a visual content representation (metadata) of the images. The complex nature of images and the complexity of image analysis algorithms do not allow a direct comparison between two images, and so representative and summarized metadata is needed. In this paper, our interest is geared towards content-dependent metadata – data which refers to perceptual low/intermediate-level features, such as color. Color is a visual feature which is immediately perceived when looking at an image. Retrieval by color similarity requires that models of color stimuli are used, such that distances in the color-space correspond to human perceptual distances between colors. Color stimuli are commonly represented as points in a three-dimensional color-space [3] such as RGB, HSV, and L*a*b*.

In order to extract the (color) metadata, we propose an adaptive and agglomerative clustering algorithm to segment the images into connected regions[1], composed by pixels with a predefined degree of color similarity. The proposed clustering algorithm is fully automatic, has an efficient implementation and is independent of the image's domain, allowing its application in large and heterogeneous collections of images. We represent and compare images based on the set of disjoint regions in which each image was decomposed. This approach has the advantage of avoiding the problem of color-space quantization and also of allowing to explore different types of metadata per region, e.g., their spatial location and size. Since our approach has many possibilities of configuration, we claim that it can be successfully tuned to work with images of specific domains, where domain-dependent metadata can be exploited both during the clustering process and during regions/images comparisons. In summary, the main contribution we offer in this paper is a efficient technique to extract color-based metadata from images, independently of a color-quantization scheme, and an algorithm to compute the similarity between images based on their metadata. We believe they are contributions for the area of image (and multimedia) databases and information retrieval, where one must be able to perform the aforementioned tasks effectively and efficiently.

This paper is organized as follows. Section 2 reviews some automatic image segmentation techniques used in existing CBIR approaches. We discuss in details two works related to ours: the segmentation module of QBIC project (which is also based on agglomerative clustering) and the Blobworld system (which is based on parametric density estimation). Section 3 describes our clustering-based CBIR approach. We discuss how we segment an image into disjoint connected regions using a simple variation of the single-linkage clustering algorithm. We also present an efficient implementation for this algorithm. The distance functions used to compare regions and images are also presented in this section. Section 4 describes the experiments used to compare the retrieval effectiveness among three variations of our approach and five other existing CBIR approaches. Section 5 presents the results of this work and Section 6 states our conclusions and directions for future work.

## 2 Related Work

Existing CBIR approaches can be classified in global methods and local methods. Global methods extract metadata of the whole image (globally). This is the case of global color histograms [20], color coherence vectors [12] and color moments [19]. Recent approaches, before extracting the image's metadata, segment an image into a set of regions/objects. Then each regions/object is characterized individually. The most important techniques used to segment an image in these approaches are: (1) spatial partitioning [2, 9, 16, 18]; (2) histogram thresholding [1]; (3) boundary detection [11]; (4) density estimation [4, 13, 21] and (5) hierarchical clustering [2].

Spatial partitioning is the simplest technique to segment an image. It consists of decomposing an image into a set of rectangular cells. In some approaches these cells are determined according to a grid superimposed over the image, in others, it is used cells of different sizes which may or may not overlap. It is also common to decompose an image in multiple resolutions using slightly variations of a quadtree [17]. The histogram thresholding techniques also try to segment objects/regions indirectly, according to the distribution of the image's colors. This technique depends entirely on how well the histogram can be partitioned. The boundary detection approaches assumes that the regions of an image are sufficiently homogeneous so that the transition between two regions can be determined on the basis of visual properties discontinuities. An edge is exactly the boundary between two regions with relatively distinct visual properties. Because of noises and other effects that introduce spurious discontinuities, edge-detection algorithms must be followed by some kind of edge-linking algorithm. The parametric density estimation approaches are based on the basic assumption that the underlying data density is a mixture of $g$ Gaussian densities. If this assumption holds then it suffices to estimate the $g$ means and covariances of these Gaussians and to partition the data among them, to get satisfactory regions. In non-parametric

[1]Two regions are disjoint if they do not share any pixel. One region $A$ is connected if there is a path between every pair of pixels of $A$, formed only by pixels of region $A$ [8].

approaches, the density is determined by convolving the data points with a density-kernel [13] of limited support. Clusters are identified by locating local density-maxima.

Hierarchical clustering [6, 10] are among the best-known clustering methods. There are two kinds of hierarchical algorithms: *agglomerative* and *divisive*. Agglomerative methods start when all pixels are apart, i.e., they start with $n$ singleton clusters. Then in each step two clusters are merged, until only one is left. The main steps of an agglomerative procedure can be viewed in Figure 1. On the other hand, divisive methods start when all pixels are together and, in each following step, a cluster is split up, until there are $n$ of them. In the literature, hierarchical clustering is usually meant to be agglomerative clustering. The main reason for this appears to be the computational aspect [10]. In the first step of an agglomerative algorithm all possible fusions of two data elements are considered, leading to $C_n^2 = \frac{n(n-1)}{2}$ combinations. A divisive algorithm based on the same principle would start by considering all divisions of the data set into two nonempty subsets, which amounts to $2^{n-1} - 1$ possibilities. The latter number grows exponentially fast and even for medium-size data sets, such a complete enumeration approach is computationally prohibitive.

```
Aggl-Clustering(k)
    Consider n singleton clusters, one for each data element
    Let k' = n.
    If k' ≤ k, then stop.
    Find the nearest pair of distinct clusters, say A and B
    Merge A and B, delete B, and decrement k' by one
    Go to 3
```

Figure 1: Generic agglomerative clustering algorithm

When $d_{min}(A, B) = min(d(A_i, B_j))$ is used as distance measure between clusters $A = \{A_1, A_2, ..., A_n\}$ and $B = \{B_1, B_2, ..., B_m\}$, the resulting clustering algorithm is often called *nearest-neighbor* or the *minimum* algorithm. If it is terminated when the distance between nearest clusters exceeds an arbitrary threshold, it is called the *single-linkage* algorithm, which is the oldest and simplest agglomerative clustering algorithm. The single-linkage algorithm is not able to keep two clusters clearly distinct apart when they come close to each other at even a single point. A single link between two clusters is sufficient to connect them. This characteristic leads to the notorious *chaining effect* [6, 10], by which poorly separated clusters are chained together.

Next, we will discuss two works more related to ours in the sense that they also deal with image segmentation for CBIR. The QBIC project was chosen because it has a segmentation module based on cluster analysis. Blobworld system (although it is based in parametric density estimation) was chosen because at the end of Section 5, we will compare our segmentation results with the Blobworld's segmentation results.

**QBIC –** In QBIC project [2], the pixels of an image are clustered in a way that, at the end of the process, the segmented image has only a small number of colors (5 to 15), depending on the image. During the clustering process, two clusters are merged if their mutual rank falls bellow a predefined threshold. The mutual rank of clusters $P$ and $Q$ is $n + m$, where $Q$ is the $n^{th}$ closest cluster to $P$ and $P$ is the $m^{th}$ closest cluster to $Q$. The cluster's distance is measured as the Euclidean distance between their mean colors. The clustering process also considers the similarity between the spatial distribution of clusters' pixels. For each final color obtained, the connected components of the pixel population having that color are identified and, for each connected component, a bounding rectangle is calculated. The bounding rectangles for a given color are successively clustered into groups of geometrically close rectangles until one rectangle remains. The result is a hierarchical tree structure for each cluster color. The distance between two regions is calculated as a weighted sum of the distance between the colors themselves and the distance between their associated tree. The distance between two images $I$ and $J$ is the average of the distance between each region of $I$ and its closest region in $J$. There is no reference to the computational complexity of the approach (clustering procedure and image distance calculation) and to the space overhead of the metadata.

**Blobworld –** The Blobworld system [4] cluster pixels in a joint color-texture-position eight-dimensional space. In order to segment an image automatically, the joint distribution of color, texture and position features are modeled as a mixture of Gaussians. The Expectation-Maximization (EM) algorithm is used to estimate parameters of this model. Creating the Blobworld representation of an image involves three steps: (1) selecting an appropriate scale for each pixel and then extracting color, texture and position for that pixel at the selected scale; (2) grouping pixels into the resulting 8D feature space modeling the distribution of pixels features as a mixture of Gaussians; and (3) describing the features of each region obtained. For each pixel, the three color components are the L*a*b* coordinates found after spatial averaging using a Gaussian at the selected scale. The three texture components are polarity, anisotropy and contrast for the neighborhood determined by the selected scale. Finally, the 8D feature vector of each pixel also contains its spatial coordinates. To group these points, the EM algorithm was used to determine the maximum likelihood parameters of a mixture of $K$ Gaussians in the feature space. The value of $K$ is determined using the log-likelihood in conjunction with the Minimum Description Length (MDL) principle.

The image pixels are grouped into connected-regions according to the selected model. After this, some postprocessing steps are performed to align the boundaries of the cluster with the original boundaries of the image and to enforce a minimal amount of spatial smoothness in the final segmentation. The color of each region is represented as a 500 bins local color histogram (based on L*a*b* color space). To compare the color of two regions, it was used the quadratic distance between their histograms $d^2(x, y) = (x - y)^T A(x - y)$, where $A = [a_{ij}]$ is a matrix of weights representing the similarity between bins $i$ and $j$. It is unclear from the original paper the space overhead and computational complexity of the approach. One remarking difference between Blobworld and the approach we present next is that Blobworld is mostly concerned with region-based queries whereas in using our approach, for the time being, we are concerned with full image queries.

# 3 Color-based Clustering – Our Approach to CBIR

In our color-based clustering approach to CBIR, each image is decomposed in a set of disjoint connected regions. Each region is larger (in number of pixels) than a threshold size $s_0$. Additionally, all pixels of a region have a predefined degree of color similarity, according to a threshold color-distance $d_0$. We call the procedure to obtain these regions CBC($d_0$, $s_0$), where the thresholds $d_0$ and $s_0$ are parameters defined by the user. These parameters have a direct impact on the number of regions in which each image is decomposed. We adopt the convention that parameters $d_0$ and $s_0$ are percentages of the maximum value allowed. Thus, $s_0 = 10$ means a threshold size equivalent to 10% of the image size (the maximum possible size of a region) and $d_0 = 5$ means a threshold distance equivalent to 5% of the maximum distance between two points in the chosen color-space. The CBC($d_0$, $s_0$) procedure can be conceptually divided into four main steps:

1. Convert an input image $I$ into a weighted and non-oriented graph $G(V, E)$, where the pixels in $I$ are the vertices of $V$ and each pair $(p, q)$ of 4-adjacent pixels in $I$ define an edge of $E$ whose weight is the Euclidean distance between the colors of $p$ and $q$ in the L*a*b* color-space[2].

[2]The L*a*b* space has been defined in order to make easier the evaluation of perceptual distance between colors. In fact, they are defined according to transformations that approximate a tri-stimulus color-space into an Euclidean space [3].

2. Compute an adaptive and agglomerative clustering of pixels on $G$ using $d_0$ as parameter. This algorithm outputs a graph partition where each part is a tree whose nodes form a connected region of pixels and the least similarity between distinct regions is greater or equal to $d_0$.

3. Merge all regions systematically, whose area is less than $s_0$, with their most similar neighbor until all regions have area greater or equal to $s_0$.

4. Characterize each remaining region by a 6D feature vector $(L, a, b, s, h, v)$, where $L, a, b$ are the mean values of the color components L, a, and b in the region, respectively, $s$ is the size (number of pixels) of the region normalized by the image size, and $h, v$ are the normalized horizontal and vertical coordinates of the geometric center of the region in the image.

Note that, the feature vectors of all regions that compose the image $I$ represent its visual content, and so, they are stored in the database. To complete description of the CBC procedure, we explain in details the clustering algorithm of step 2 as follows.

## 3.1 Clustering algorithm

Our clustering algorithm consists of a simple and effective variation of the single-linkage clustering algorithm. The single-linkage algorithm can be described under the language and concepts of graph theory [6]. Consider the data elements that we want to cluster as being nodes of a graph. The merging of two (initially singleton) clusters $A = \{A_i\}$ and $B = \{B_j\}$ corresponds to adding an edge between the nearest pair of nodes $A_i \in A$ and $B_j \in B$. Since edges are added always between distinct clusters nodes, the resulting graph never has any closed circuit. In fact, the resulting graph is a *tree*. If this process continues until all clusters are linked together, it can be shown that the resulting graph is a *minimal spanning tree* – MST [6]. Based on the relation between the single-linkage algorithm and minimal spanning trees, we chose to implement our single-linkage algorithm with a implementation equivalent to that of the well known Kruskal's greedy algorithm to generate a MST [5, Section 24.2]. This is the asymptotically fastest implementation known to us. The procedure used as base to implement our variation of the single-linkage algorithm is shown in Figure 2.

```
    Single-Linkage(G(V,E), d_0)
1       for each vertex v ∈ V do Make-Set(v)
2       sort the edges of E by nondecreasing weight w
3       for each edge (u,v) ∈ E, ordered by nondecreasing weight
4          if d_mean(Find-Set(u), Find-Set(v)) < d_0
5             if Find-Set(u) ≠ Find-Set(v), then Union(u,v)
6          else break
```

Figure 2: Our implementation of the Single-Linkage algorithm

The function `Make-Set(.)` creates a cluster with only one element: the node passed as parameter. The function `Find-Set(.)` returns the identity of the cluster that the element passed as parameter belongs to. Our clustering algorithm works as follows. Line 1 creates $|V|$ clusters, each with one node of $G$. The edges in $E$ are sorted by nondecreasing weight in line 2. The `for` loop in lines 3-6 checks, for each edge $(u, v)$, if the distance between the clusters of $u$ and $v$ is smaller than the threshold distance $d_0$ (the stop criteria). If so, and if the clusters are distinct, they are combined. To diminish the effects of the single-linkage chaining effect (Section 2), we introduced a new heuristic into the stop criteria. Instead of comparing the weight of the edge being analyzed directly with the threshold distance $d_0$, we compare the distance of the mean vectors of each cluster with $d_0$. To do this, we need to associate a mean vector to each cluster and update these vectors whenever a `Union(.,.)` operation is performed. As we can see, we use the $d_{min}$ distance to decide the ordering and which clusters are candidates to be merged, and $d_{mean}$ distance to decide whether the candidate clusters will really be merged or not. This solution has the advantage to be as efficient as the traditional single-linkage procedure and, at the same time, to diminish the chaining effect, the main problem of the single-linkage approach.

Our algorithm differs from the Kruskal's original algorithm in two ways. First, we do not store the edges that compose the MST. Second, we added a stop criterion (line 4) to finish the process before we obtain only one cluster (the MST). Clearly, these two modifications do not improve the original Kruskal's algorithm running time, nor they were meant to. Thus, as Kruskal's algorithm, the running time of our single-linkage algorithm is $O(E \log E)$ [5]. Since at the input graph $G$, $E = O(V)$ (because of the connectivity restriction), we can say that our clustering procedure runs in time $O(n \log n)$, where $n = |V|$ equals the number of pixels of the input image.

### 3.2 Distance Function

In this section, we describe the distance function used to compare two images in the CBC approach. The distance between two images $A$ and $B$, $d(A, B, \alpha)$, is a weighted composition of the distances between the regions that compose each image – $Rd(A_i, B_j, \alpha)$. Here, $A_i$ and $B_j$ are regions from images $A$ and $B$ respectively, obtained via the clustering algorithm detailed before. The $\alpha$ parameter defines the weights used to combine distances in the color-space with distances between the spatial positions of the regions. The region's distance function $Rd(A_i, B_j, \alpha)$ is thus defined as:

$$Rd(A_i, B_j, \alpha) = \alpha \times L_2(A_i.color, B_j.color) + (1 - \alpha) \times L_2(A_i.center, B_j.center) \tag{1}$$

In this equation, $L_2(.,.)$ is a function that returns the $L_2$ distance between its arguments. The computation of the distance between two images $d(A, B, \alpha)$ is algorithmically described in Figure 3.

The distance $d(A, B, \alpha)$ works as follows. Initially, we compute all possible distances between pairs of regions (one of each input image), according to the function $Rd(A_I, B_j, \alpha)$. Additionally, all regions are initialized as "non-matched" (status=0). The initialization steps (lines 1, 2, 3 and 4) takes time $O(nm)$, where $n$ and $m$ are the number of regions of images $A$ and $B$, respectively. In line 5, the computed distances are ordered in non-decreasing order. In this way, the first distance value correspond to the best possible match between a region of image $A$ and a region of image $B$. The second value corresponds to the second best match and so on. The ordering of $nm$ values takes time $O(nm \log(nm))$. For each distance $D_{A_iB_j}$ in non-decreasing order, we compare the size of the related regions $A_i$ and $B_j$. The smallest region determines the weight value $w$ that will multiply the value $D_{A_iB_j}$ in order to obtain the overall images' distance $\beta$. The weight $w$ represents the percentage of the two images that were matched with the distance $D_{A_iB_j}$. In each iteration of the `for` loop in line 7, the smallest region related to the $D_{A_iB_j}$ distance is marked as "matched" (status=1). At this moment, this status means that the whole (smaller) region has been matched, and any distance related to this region in next iterations will be discarded. Since at each iteration, the larger region may not be completely matched, we subtract the size of the smallest region (the percentage of the region actually matched) from the size of the biggest region. After this operation, if the size of the region equals 0 (occurs when the two compared regions have the same size), than both regions were matched and we mark them accordingly. If the size of the biggest region remains greater than 0, then we continue analyzing distances involving this region until the size of the region equals 0. Observe that the next distances involving a previous analyzed region will always be higher than the previous distance values because we are analyzing these values in non-decreasing order. Observe also that, at the end of the process, the sum of the weight values $w$ equals 1, meaning that all image's content was compared. The `for` loop in line 7 is executed $nm$ times, one for each distance $D_{A_iB_j}$. Thus, the execution of lines [7..18] takes time $O(nm)$.

```
d(A, B, α)
    for each pair of regions $A_i \in A$ and $B_j \in B$
        $A_i.status = 0$
        $B_j.status = 0$
        $D_{A_iB_j} = Rd(A_i, B_j, \alpha)$
    sort the computed distances $D_{A_iB_j}$ in non-decreasing order
    $\beta = 0$
    for each distance $D_{A_iB_j}$ in non-decreasing order
        if $A_i.status = B_j.status = 0$
            if $A_i.size < B_j.size$
                $w = A_i.size$
                $B_j.size = B_j.size - A_i.size$
                $A_i.status = 1$
            else
                $w = B_j.size$
                $A_i.size = A_i.size - B_j.size$
                $B_j.status = 1$
                if $A_i.size = 0$ then $A_i.status = 1$
            $\beta = \beta + w \times D_{A_iB_j}$
    return $\beta$
```

Figure 3: Distance function algorithm

It is important to note that an actual region $A_i$ of an image may be virtually "broken" in smaller units to find the best possible match for it. The best possible match may involve only one greater region, a set of several smaller regions, or also a combination of partial content of several regions of the other image. In any way, at the end of the process, the "real" regions of the two images were decomposed in possibly smaller "virtual" regions in such a way that: (1) the number of "virtual" regions of the two images is the same, (2) there is a one-to-one correspondence between the virtual regions of two images and (3) each pair of correspondent regions have the same (normalized) size. Thus, we can say that the images' distance is just an average of the similarities between correspondent virtual regions, weighted according to the region's (normalized) size.

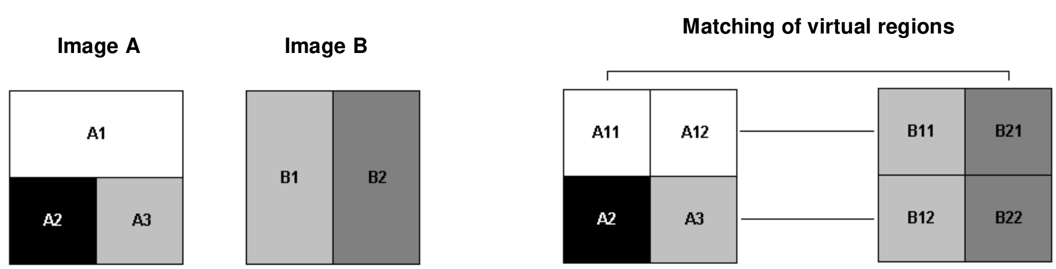


Figure 4: An example of how the function $d(A, B, \alpha)$ decompose real regions into virtual regions

We will exemplify the application of our distance function using the images in Figure 4. For the sake of simplicity, assume the use of gray scale images and the distance between two regions given only by the difference of their gray levels. In Figure 4, we have two images $A$ and $B$ that we want to compare. Image $A$ has three regions (obtained via a suitable algorithm, e.g., our proposed CBC): $A = \{A_1 \cup A_2 \cup A_3\}$. Likewise, image $B$ has only two regions: $B = \{B_1 \cup B_2\}$. As can be seen, the best possible match between regions of $A$ and $B$ is $(A_3, B_1)$ since this pair of regions has exactly the same gray level. Since $B_1$ is larger than $A_3$, we split $B_1$ into two virtual regions $B_1 = \{B_{11} \cup B_{12}\}$, in a way that one of the resulting regions ($B_{12}$) has exactly the same normalized size of region $A_3$. Thus, the first matching we obtain is $(A_3, B_{12})$ and this match will contribute to the image's similarity with a weight of 0.25 (the normalized size of the matched regions). The second best match between real regions of images $A$ and $B$ is $(A_1, B_1)$ but, as the region $B_1$ was split before and one of the resulting regions ($B_{12}$) was matched at a previous step, the only possible choice is $(A_1, B_{11})$. Again, as region $A_1$ is larger than region $B_{11}$, we decompose $A_1$ in two virtual regions $A_1 = \{A_{11} \cup A_{12}\}$ in a way that the virtual region $A_{12}$ has the same size of $B_{11}$. Thus, our second match is $(A_{12}, B_{11})$ also with weight 0.25. In spite of a distance tie between $(A_2, B_2)$ and $(A_{11}, B_2)$ let us assume the third best match is between $(A_2, B_2)$. As $B_2$ is larger than $A_1$, we decompose $B_2$ in two regions, as in previous steps. The third match we obtain is between $(A_2, B_{22})$. Finally the last possible match is $(A_{11}, B_{21})$ and, as these regions have the same size, there is no need to split regions and the process is finished. Since all the virtual regions have the same size, the four virtual matchings (evaluated according to their gray level distance) are averaged with the same weight $w = 0.25$ in order to obtain the images' similarity.

# 4 Experiments

We compared the effectiveness of our CBIR approach with the effectiveness of five other approaches: (1) Global Color Histogram – GCH [20], (2) Color-Coherence Vector – CCV [12], (3) Color-Moments – CMM [19], (4) Grid [18] and (5) Color-Shape Histograms – CSH [18]. The first three approaches are global methods and the last two are partition-based methods.

A GCH is obtained by quantizing an image's colors and counting how many pixels belong to each color. Assuming an image composed by $n$ pixels, $k$ being the $k^{th}$ color in an uniformly quantized color-space and $a[k]$ being the number of pixels whose color is $k$, a GCH $h$ is an array of real numbers, each one describing one quantized color, according to the function $h[k] = a[k]/n$. In CCV approach, for each color $k$ in a quantized color-space, there are two values associated: the number of coherent pixels vs. the number of incoherent pixels with this color. A pixel is coherent if it is part of a large similarly-colored connected region. The CMM approach, instead of representing the complete color distribution by a histogram, represents only their dominant features, via the first three moments of each color channel. The Grid approach consists in superimposing a grid of cells over the image. The image content is represented by an array of LCHs, one for each cell, according to the ratio: $h[l][k] = a[l][k]/n$, where $n$ is the number of pixels of the image, $l$ is the $l^{th}$ cell of the grid superimposed on the image, $k$ is the $k^{th}$ color in an quantized color-space and $a[l][k]$ is the number of pixels whose color is $k$ in cell $l$. The CSH approach is a slightly variation of the Grid approach The main idea behind CSHs is to also use a Grid-based partitioning of the images, but instead of using a LCH for every cell, a "cell histogram" is used for every color. Thus storage space overhead is reduced due to the empirical observation that images contain a relatively low number of all possible colors.

We compared the CBC approach with three distinct combinations of threshold color-distance ($d_0$) and threshold size ($s_0$) . Each combination resulted in a different number of regions per image. The chosen thresholds are explicit at the approach name, for example, $CBC(5, 0.3)$ means that we have $d_0 = 5\%$ of the maximum distance between two points in the chosen color-space and $s_0 = 0.3\%$ of the maximum possible size of a region (the image size). Since our distance function depends of a parameter $\alpha$, we experimentally determined that $\alpha = 0.875$ corresponds to the most adequate compromise. Confirming our intuition, this value suggests that the the distance between regions's colors is more relevant than the position of such regions when we are comparing two images. Even though it was not too relevant, using the position of the regions did affect the results. When we used $\alpha = 1$

(eliminating completely the contribution of the spatial location to the distance value), the effectiveness of our approach did became slightly smaller. A summary of the most important characteristics of the eight compared approaches can be viewed in Table 1.

Table 1: Summary of the characteristics of the compared approaches.

| **Approach** | **Color-Space** | **Cells/regions** ♯ | **Space**♭ | **Complexity**‡ | **Metric** |
|---|---|---|---|---|---|
| GCH | RGB | 1 | 64 | $O(n)$ | $L_1$ |
| CMM | HSV | 1 | 9 | $O(n)$ | $L_1$ |
| CCV | RGB | 1 | 128 | $O(n)$ | $L_1$ |
| Grid | RGB | 64 | 4096 | $O(n)$ | $L_1$ |
| CSH | RGB | 64 | 1856 | $O(n)$ | $L_1$ |
| CBC(5, 0.3) | L*a*b* | 11 | 66 | $O(n \log n)$ | $d(A, B, \alpha)^\dagger$ |
| CBC(3, 0.1) | L*a*b* | 40 | 240 | $O(n \log n)$ | $d(A, B, \alpha)^\dagger$ |
| CBC(2, 0.05) | L*a*b* | 155 | 930 | $O(n \log n)$ | $d(A, B, \alpha)^\dagger$ |

♯ Average values for the 20,000 images we used

♭ Average number of real numbers needed to represent the images in our database

‡ $n$ is the size (number of pixels) of an image

† $\alpha = 0.875$

We should to emphasize that the five approaches that we used in our comparative analysis were implemented exactly as suggested by their authors, hence not all approaches use the same underlying color model. Although it is possible that some (or all) approaches may yield better results in alternative color-spaces like HSV, L*a*b*, etc, our focus was not to investigate how the choice of the color-space affects the effectiveness of those approaches. Such study would require one to analyze aspects orthogonal to our research goals, for example, how to quantize each color-channel and the most appropriate distance function in each color-space. Thus, instead of just proposing a new CBIR approach and show absolute effectiveness values to justify its existence, we chose to compare its effectiveness with the effectiveness of other existing CBIR approaches.

It is also reasonable to question the validity of our comparative analysis since our approach is the only one using a conceptually more robust color-space (L*a*b*). The motivation to use the L*a*b* color-space in our approach is completely distinct to the motivation to use this color-space in approaches like GCH, CMM, CVV, Grid and CSH. In these approaches, the motivation to suggest the use of spaces like HSV or L*a*b* is to, at least partially, decouple the color channels [19], allowing to quantize each channel individually, according to the characteristics of the images being analyzed. Since we are working with heterogeneous images, the eventual advantage of alternative color-spaces in those approaches is minimized, since we do not know *a priori* the images characteristics and thus, we also do not know *a priori* the ideal quantization scheme. Observe also that, in these approaches, there is no computation of color's distance directly. The computed color histograms and their distance is relative exclusively to the color distribution. Thus, the fact that the $L_2$ is a natural distance for the L*a*b* color-space cannot be explored in any of the investigated approaches but ours. In a different way, our clustering approach does not require color-space quantization and thus, we do not need decouple color channels as in traditional CBIR approaches. We chose to work with the L*a*b* color-space because it is a space with a well defined distance between colors. Differently of the previous approaches, both the clustering procedure and the distance function in CBC are completely dependent of a robust distance between colors. Of course, our results are dependent of this color-space but we believe that the choice of the color-space is also part of a CBIR approach.

We evaluated the effectiveness of the approaches in processing (whole) image-based queries, in a controlled environment. Object-based queries and efficiency comparison are part of our future works. Since we do not have yet interaction with the user population, the evaluation study was based on objective criterias. For example, the set of relevant images relative to a given query was determined *a priori*, using an objective relevance criteria in which the relevance of a image relates to how well the image responds to the query that was posed. The subjective view of relevance considers not only the content of a image but also the state of knowledge of the user at the time of the search, and the other images retrieved or available that the user already knows about. We used a dataset of 20,000 JPEG images from a stock CD by Corel Corp. This database is composed of images of several different domains and, for this reason is called a *heterogeneous database*. In each domain, the images are semantically related, allowing to distinguish one domain from the others easily. Since the images belonging to the same domain may not have a very similar visual content, we called these domains *heterogeneous domains*. We chose 29 of such domains to be used in our experiments. Out of each domain, we selected a image to be used as query image and a set of images visually similar to the chosen query image. These sets were called *relevant result sets* - RRsets. In any CBIR approach, we expect to retrieve the RRSets as soon as possible, since they correspond to what we considered relevant for each query image. In the average, we have 30 relevant images per query image. We realized two distinct experiments with databases of different sizes. In one experiment, we used the 20,000 images of the whole dataset as a large and heterogeneous database. In the other experiment, we used the union of the RRSets of the first experiment as a small and less heterogeneous database with 1,023 images. We used the same queries and RRSets in both experiments. Our goal was to analyze if the relative performance between the analyzed approaches remains the same if we change the size of database, enlarging/diminishing the proportion of relevant/non-relevant images.

We compared the approaches' effectiveness using two distinct measures: (1) Precision vs. Recall [14] and (2) *Normalized Average Rank* – NavgR' [18] (a derivative of the measure presented in [7]). The effectiveness of a retrieval system is a measure related to the users satisfaction with the system output. Since it is generally difficult for users to distinguish among a large number of degrees of relevance, we choose to work with binary values: either a image is relevant or it is not. The scale for retrieval is also binary: either a document is retrieved or it is not. Precision vs. Recall [14] was chosen because it is the best known and widely used measure in information retrieval systems. Although it is not the most adequate measure in the context of ranked output [7, 14, 15], it has been widely used also in comparison of CBIR systems. Precision measures the accuracy of the search and it is defined as the fraction of the retrieved images which are relevant to the posed query. In contrast, Recall measures the extent to which the retrieval is exhaustive and it is defined as the fraction of the relevant documents which has been retrieved. Given a set of recall-precision value pairs, a recall-precision curve can be constructed by plotting the precision against the recall.

The goal of a CBIR system is to retrieve images similar to an image/sketch provided by the user. This contrasts with traditional databases where the goal is an exact-match on a set of attributes. Matching is a binary partition, being intrinsically committed to deciding whether or not the object observed corresponds to a model. On the other hand, similarity-based retrieval is the task of reordering database images according to their measured similarity to a query image. It is therefore concerned with ranking rather than classification [3]. In this context, a effectiveness measure should consider the total ranking imposed on all the collection, independent of a retrieval threshold. The Normalized Average Rank (NavgR) [7] is a single-value measure that satisfies this restriction. It is a variation of the *normalized recall* measure [14] which was first used in QBIC project [7] and then, in some other CBIR approaches [15, 18]. NavgR measures how close to the top of the ranked result, the set of relevant items appears. This value is normalized considering an ideal retrieval in which the relevant images appear ahead of any non-relevant one in the ranked result. The relative ordering of the RRset elements is irrelevant because we are using a binary judgment of relevance. The measure used in QBIC project was defined as $NavgR = A/I$, where $A$ and $I$ are, respectively, the actual average rank and the ideal average rank of the RRset. In this paper, we propose a slightly different definition of the NavgR measure, inverting the original ratio as shown in Equation 2.

$$NavgR' = \frac{I}{A} = \frac{\sum_{i=1}^{|RRSet|}(i-1)}{\sum_{i=1}^{|RRSet|} rank(i)} \tag{2}$$

In that equation, $rank(i)$ is a function that returns the rank of the $i^{th}$ relevant image. The rank values varies in the range $[0, |DB|-1]$, where $|DB|$ denotes the cardinality of the image database. With this alternative definition, the NavgR' value ranges from 0 to 1 and equals 1 in the best case, as occur in precision and recall measures. Higher values of NavgR' are associated to good effectiveness results, instead of vice-versa. This fact diminish the dependence of the measure on worst results during the process of averaging results from distinct queries. A drawback with the NavgR' measure, is that, the value obtained for a single query is still very sensitive to the rank of the last relevant document retrieved. Our future research will include an investigation of possible solutions to this problem. In this paper, we deal with this problem considering only a predefined portion of the RRSets for the purpose of NavgR' computation. We consider only the ranks of the first 80% portion of an RRSet. The last 20% of the RSSets (with the highest numerical ranks) are discarded, since any possible misleading judgment of relevance (that will strongly affect the NavgR' value) will materialize itself as a large numerical rank value.

## 5 Experimental Results

Our experiments are different from what have been used in other CBIR approaches in some aspects: (1) we used two databases of different sizes (with 1.023 and 20.000 images, hereafter referred to as Experiment 1 and Experiment 2, respectively); (2) in each experiment we averaged the effectiveness results for 29 "heterogeneous" queries; (3) we chose RRSets with higher cardinality (30 images in average); (4) we measured the retrieval effectiveness of the approaches using two distinct effectiveness measures; (5) we compared the effectiveness of three variations of our approach with five completely distinct CBIR approaches. The results of the experiments obtained with the NavgR' measure can be viewed in Table 2.

Table 2: Average NavgR' values for the 29 submitted to each approach in each experiment (the higher the better)

| Approach | Experiment 1 | Experiment 2 |
|---|---|---|
| GCH | 0.4001 | 0.0373 |
| CMM | 0.2063 | 0.0146 |
| CCV | 0.3838 | 0.0334 |
| Grid | 0.4361 | 0.0445 |
| CSH | 0.4680 | 0.0515 |
| CBC(5, 0.3) | 0.4423 | 0.0573 |
| CBC(3, 0.1) | 0.4868 | 0.0740 |
| CBC(2, 0.05) | 0.5332 | 0.0795 |

As can be seen, the numbers obtained from the first experiment (smaller database) are much higher than the number of the second experiment, which uses a database approximately twenty times larger. This shows that all approaches are affected by the database growth but, as Table 3 shows, not all approaches are affected in the same way. Table 3 shows a comparative analysis of the approaches, based on the numbers of Table 2 and using the GCH approach as reference. We also included the space overhead (in terms of floating-point numbers) and the computational complexity of each approach. Next we discuss the results in terms of global, partition-based and clustering approaches always using the GCH as a yardstick.

Table 3 shows that among the global approaches (GCH, CMM and CCV), the traditional GCH has the best performance in both experiments. Clearly, CMM is the worst approach in terms of effectiveness, about 50% worse than GCH's. However, one must note that CMM yields the smallest space overhead, seven times smaller than GCH's overhead. Among these three approaches, the GCH is also more resilient to the growth of the database, since the relative differences between CMM and CVV increased with the increase of the database size.

Table 3: Relative effectiveness using the GCH approach as reference

| Approach | Space* | Complexity | Experiment 1 | Experiment 2 |
|---|---|---|---|---|
| GCH | 64 | $O(n)$ | 0.00% | 0.00% |
| CMM | 9 | $O(n)$ | -48.44% | -61.00% |
| CCV | 128 | $O(n)$ | -4.07% | -10.60% |
| Grid | 4096 | $O(n)$ | 9.00% | 19.16% |
| CSH | 1856 | $O(n)$ | 16.97% | 38.01% |
| CBC(5, 0.3) | 66 | $O(n \log n)$ | 10.56% | 53.48% |
| CBC(3, 0.1) | 240 | $O(n \log n)$ | 21.67% | 98.09% |
| CBC(2, 0.05) | 930 | $O(n \log n)$ | 33.26% | 112.93% |

* Average number of floating-point numbers needed to represent each image

Figure 5 shows the Precision vs. Recall curves for the global approaches. As can be seen, the obtained curves are consistent with the values obtained with the NavgR' measure. Additionally, these curves show that the CVV approach is better than the GCH until about 40% of recall, when the GCH becomes slightly better.

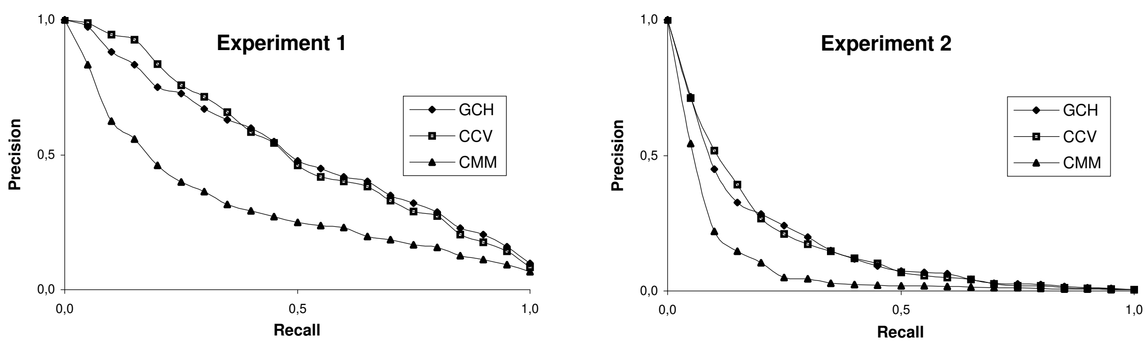


Figure 5: Precision vs. Recall graphs for the global CBIR approaches

Among the partition-based approaches (Grid and CSH), CSH is the better approach in both experiments, with a space overhead 50% smaller than Grid's. The effectiveness of CSH is about 38% better than GCH's (at the second experiment of Table 3), at the cost of a space overhead 30 times larger. The Precision vs. Recall graph of the partition-based approaches, plus the curve of the GCH approach used as reference can be viewed in Figure 6. Again, the graph confirms the results obtained with the NavgR measure.

Finally, the results in Table 3 show that the three distinct configurations of our clustering approach are more effective than CSH – the best of the compared approaches. The results also show that our approach is more robust to the growth of the database. Considering the $CBC(5, 0.3)$ configuration, we have simultaneously the space overhead of the GCH approach and an effectiveness higher than CSH's effectiveness (at the second experiment), with more robustness since the relative effectiveness of the CSH approach doubled when the database growth while the effectiveness of $CBC(5, 0.3)$ becomes 5 times higher. Using a configuration that results in more regions ($CBC(2, 0.05)$) we can achieve gains of the order of 110% relative to the GCH approach, with more robustness and with a space overhead 50% smaller than CSH's. The robustness is an important factor since it implies that, the larger the database, the larger the differences in effectiveness between our approach and the others compared approaches. The Precision vs. Recall graph of our approaches can be viewed in Figure 7.

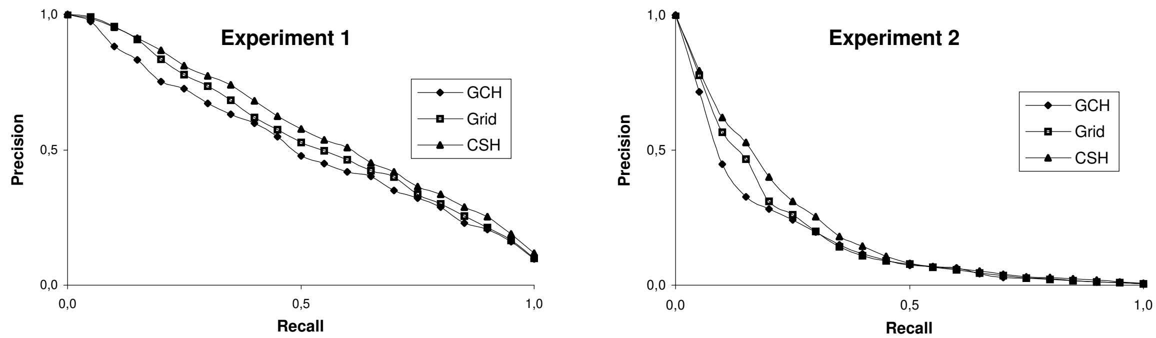


Figure 6: Precision vs. Recall graphs for the partition-based approaches

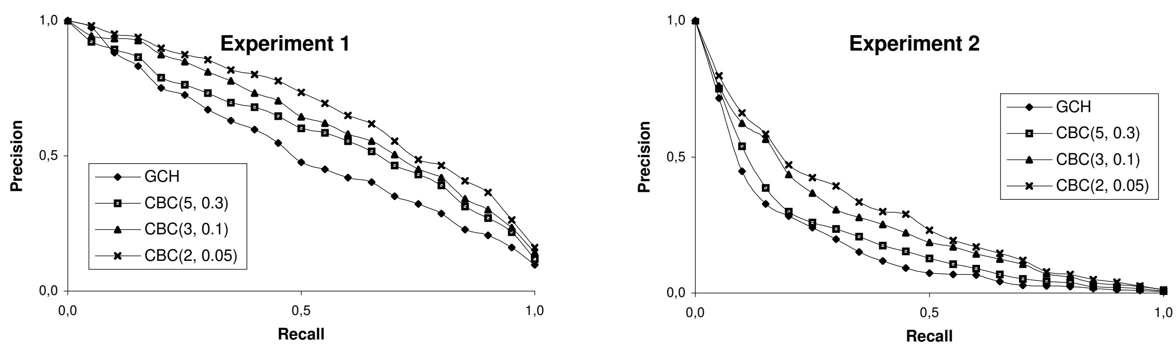


Figure 7: Precision vs. Recall graphs for our clustering approaches

The $CBC(3, 0.1)$ configuration represents an interesting compromise between the size of the metadata (number of regions), effectiveness and robustness. With this configuration, we obtain in average 40 regions per image, resulting in a space overhead 4 times larger than GCH's, but 90% smaller than CSH's. The effectiveness of $CBC(3, 0.1)$ is 98% higher than GCH's while CSH's gain is only of 38%. When the database grows from 1.023 to 20.000 images, the advantage of CSH in relation to GCH becomes approximately two times larger, while with $CBC(3, 0.1)$ this advantage becomes five times larger, suggesting that $CBC(3, 0.1)$ is also considerably more robust than the other compared approaches.

At this point it is worthwhile to discuss, at a high level, how our CBC approach would compare to the more recently proposed Blobworld system [4]. We shall use our CBC(3,0.1) variation, which is not the most effective nor the fastest, in the comparisons that follow.

Even though the computational complexity of Blobworld's algorithms is not formally analyzed in [4], it is reported that the segmentation process takes 5-7 minutes per image of size $n = 128 \times 192 = 24,576$ pixels, on a 300Mhz Pentium II. Using CBC(3,0.1), an image of size $n = 374 \times 251 = 93,874$ pixels is processed in at most 5 *seconds*, using an 500MHz AMD K6II. In Blobworld, the color of each region is represented by a 500 bins histogram while in CBC the color content of each region is described only by its average L*a*b* color (3 floating-point number). We should also observe that the use of histograms to represent the color content of a region introduce problems like color-space quantization, cross bins comparisons, and space overhead that minimizes its possible advantage. Our approach totally avoids such problems. Since the regions were obtained according to the color similarity between pixels, the average color is an adequate descriptor to the color distribution of a similarly-colored region. In order to compare CBC and Blobworld segmentation algorithms, we

Figure 8: Segmentation results in animals and outdoor scenes. Original image, Blobworld result and CBC(3,0.1) result, respectively

segmented some images available at the Blobworld home-page [3] using the CBC(3,0.1) configuration. As mentioned earlier, the CBC(3,0.1) configuration is not our most effective configuration and, under the assumption that the images segmented at Blobworld home page were carefully obtained, we believe the comparison, even though not formal, is fair. The original images, their segmented version using Blobworld and their segmented version using CBC(3,0,1) can be viewed in Figures 8 and 9.

When processing images of animals and outdoor scenes (Figure 8), it is possible to observe that the CBC results are at least equivalent to the Blobworld results. In our approach we do not have some of the problemas reported by Blobworld authors. Since we do not use the spatial coordinates of the pixels as additional dimensions of the feature vectors, but as a restriction on the input graph, we do not have large similarly colored background areas arbitrarily splitten. We also do not have problems such as regions boundaries that do not follow image boundaries (visually apparent) exactly. In Figure 9 we have an image domain in which we can observe that the CBC segmentation results seem to be better than Blobworld's ones (even though that approach consider additional texture information to cluster pixels, whereas CBC does not).

It is worthwhile to note that a direct comparison of retrieval effectiveness yielded by Blobworld and CBC is not possible simply because Blobworld queries are region-oriented in the sense that the user selects the regions

[3] http://elib.cs.berkeley.edu/photos/blobworld/

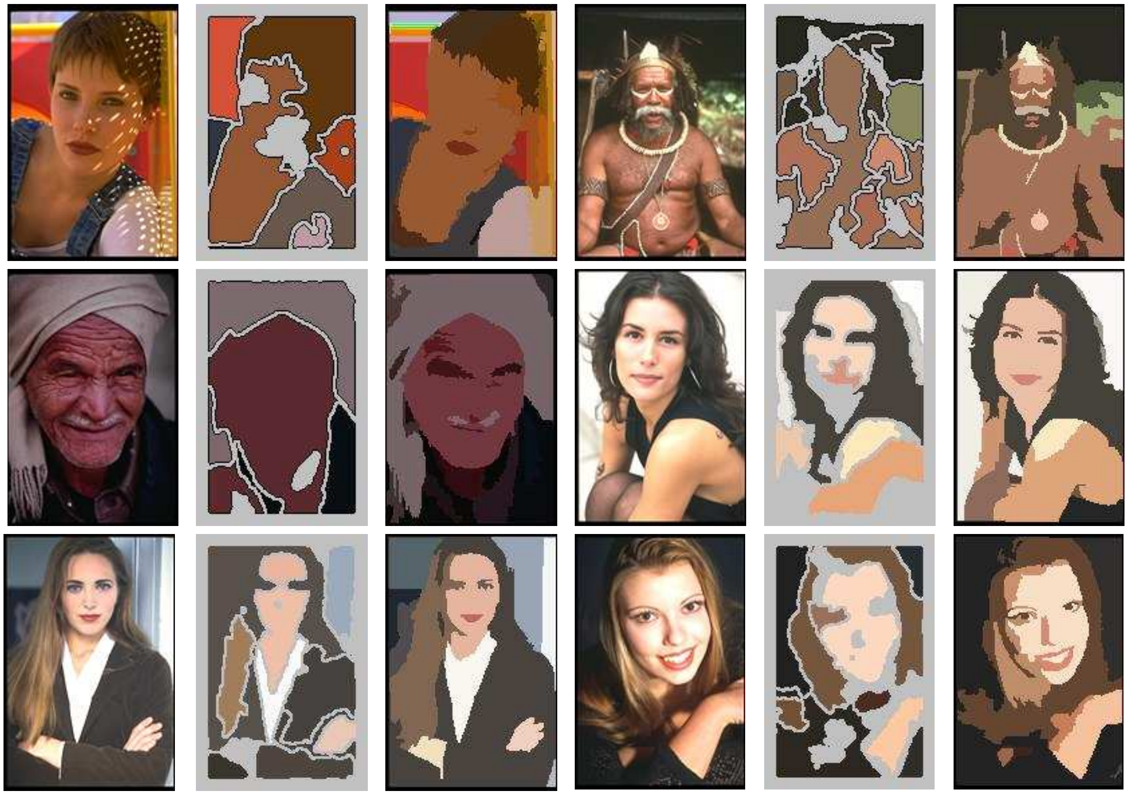

Figure 9: Persons - Original image, Blobworld result and CBC(3,0.1) result, respectively

which will guide the similarity search. In CBC all regions, i.e., the whole image content, is used to guide the similarity search. In our future work, however, we will enhance our CBIR approach, using CBC, to allow region-based queries as well.

# 6 Conclusions and Future Work

In this paper, we presented CBC, a new *content-based image retrieval* (CBIR) approach based on cluster analysis. To extract an image's color-metadata, we used a simple variation of the single-linkage agglomerative clustering algorithm to find out disjoint regions of the images, composed by pixels with a predefined degree of color similarity. This approach has the advantage of avoiding the notorious problem of color-space quantization, and of being adaptive, in the sense that the obtained segmentation depends mostly on the image itself rather than on "artificial" parameters, such as the number of clusters or the like. We represented and compared images based on the set of disjoint connected regions in which each image was decomposed. We proposed a new distance function to compare two images. We compared the effectiveness of three configurations of our CBIR approach with the effectiveness of five other approaches, in a controlled environment. We observed that the main advantages of our contributed approach were:

- **Flexibility** – using different thresholds for region's size and color's distance, it is possible to obtain different compromises between space overhead and effectiveness.
- **Configurability** – both the clustering algorithm and the distance function can be tuned to work in specific domains, i.e, to consider additionally domain-dependent metadata.

- **Effectiveness** – the three distinct configurations of CBC yielded better retrieval effectiveness than the other five compared approaches.

- **Robusteness** – Our experiments have shown that CBC is more robust with respect to the database growth. In fact, the larger the database size, the larger the relative advantage of CBC.

In a near future, we plan to investigate several other aspects related to CBIR systems, such as: (1) region-based queries effectiveness, since our approach has the potential to treat this type of query in a more elegant way; (2) use of additional metadata extraction per region (e.g., shape, texture and topological relations between regions); (3) alternative distance functions; (4) application of our approach in images of specifics domains, where domain-dependent metadata can be explore both during the clustering process and to represent/compare images (4) propose a retrieval effectiveness measure more adequate than precision, recall and NavgR to the context of similarity-based queries; (5) design and implementation of an access method to efficiently access images according to their metadata representation and the chosen distance function; and (6) realization of efficiency experiments based on the chosen access method and integration of the efficiency and effectiveness' results in order to compare distinct CBIR approaches.

## 7 Acknowledgments

Renato O. Stehling was supported by a Graduate Scholarship from FAPESP, Brazil. Mario A. Nascimento was partially supported by a Research Grant from NSERC, Canada. Alexandre X. Falcão was partially supported by a Research Fellowship Grant from CNPq, Brazil.

## References

[1] D. Androutsos, K.N Plataniotis, and A.N. Venetsanopoulos. Vector angular distance measure for indexing and retrieval of color. In *Proc. of SPIE -Storage and Retrieval for Image and Video Databases VII*, volume 3656, pages 604–613, 1999.

[2] J. Ashley, R. Barber, M. Flickner, J. Hafner, D. Lee, W. Niblack, and D. Petkovic. Automatic and semi-automatic methods for image annotation and retrieval in QBIC. In *Proc. of SPIE -Storage and Retrieval for Image and Video Databases III*, volume 2420, pages 24–35, 1995.

[3] A. Del Bimbo. *Visual Information Retrieval*. Morgan Kaufmann, 1999.

[4] C. Carson, M. Thomas, S. Belongie, J.M. Hellerstein, and J. Malik. Blobworld: A system for region-based image indexing and retrieval. In *Third Int. Conf. on Visual Information Systems - VISUAL'99*, pages 509–516, 1999.

[5] T.H. Cormem, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms*. McGraw-Hill, 1990.

[6] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*. Wiley-Interscience, 1973.

[7] C. Faloutsos, W. Equitz, M. Flickner, W. Niblack, D. Petkovic, and R. Barber. Efficient and effective querying by image content. *Journal of Intelligent Information Systems*, 3(3/4):231–262, 1994.

[8] R.C. Gonzalez and R.E. Woods. *Digital Image Processing*. Addison-Wesley, 1992.

[9] L.J. Guibas, B. Rogoff, and C. Tomasi. Fixed-window image descriptors for image retrieval. In *Proc. of SPIE -Storage and Retrieval for Image and Video Databases III*, volume 2420, pages 352–362, 1995.

[10] L. Kaufman and P.J. Rousseuw. *Finding Groups in Data - An Introduction to Cluster Analysis*. Wiley-Interscience, 1990.

[11] W.Y Ma, Y. Deng, and B.S. Manjunath. Tools for texture/color based search of images. In *Proc. of SPIE Int. Conf. - Human Vision and Electronic Imaging II*, pages 395–406, 1997.

[12] G. Pass, R. Zabih, and J. Miller. Comparing images using color coherence vectors. In *Proc. of ACM Multimedia Conference*, pages 65–73, 1996.

[13] E.J. Pauwels and G. Frederix. Finding regions of interest for content-extraction. In *Proc. of SPIE -Storage and Retrieval for Image and Video Databases VII*, volume 3656, pages 501–510, 1999.

[14] G. Salton and M.J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

[15] S. Sclaroff. Distance to deformable prototypes: Encoding shape categories for efficient search. In *Image Databases and Multi-Media Search*. World Scientific, 1997.

[16] N. Sebe, M.S. Lew, and D.P Huijsmans. Multi-scale sub-image search. In *Proc. of* $7^{th}$ *ACM Int. Conf. (part 2) on Multimedia*, pages 79–82, 1999.

[17] E. Shusterman and M. Feder. Image compression via improved quadtree decomposition algorithms. *IEEE Trans. Image Processing*, 3(2):207–215, 1994.

[18] R.O. Stehling, M.A. Nascimento, and A.X. Falcão. On 'shapes' of colors for content-based image retrieval. In *Proc. of the Intl. Workshop on Multimedia Information Retrieval - MIR'00*, pages 171–174, 2000.

[19] M. Stricker and M. Orengo. Similarity of color images. In *Proc. of SPIE -Storage and Retrieval for Image and Video Databases III*, volume 2420, pages 381–392, 1995.

[20] M.J. Swain and D.H. Ballard. Color indexing. *Intl. J. Computer Vision*, 7(1):11–32, 1991.

[21] M.-H. Yang and N. Ahuja. Gaussian mixture model for human skin color and its applicattions in image and video databases. In *Proc. of SPIE -Storage and Retrieval for Image and Video Databases VII*, volume 3656, pages 458–466, 1999.