pp 496-507, SPIE Int. Conf. 3106, Human Vision and Electronic Imaging II, Feb. 1997.

Tools for texture/color based search of images

W. Y. Ma, Yining Deng, and B. S. Manjunath

Department of Electrical and Computer Engineering
University of California, Santa Barbara, CA 93106
E-mail: {wei, deng}@iplab.ece.ucsb.edu, manj@ece.ucsb.edu

ABSTRACT

Currently there are quite a few image retrieval systems that use color and texture as features to search images. However, by using global features these methods retrieve results that often do not make much perceptual sense. It is necessary to constrain the feature extraction within homogeneous regions, so that the relevant information within these regions can be well represented. This paper describes our recent work on developing an image segmentation algorithm which is useful for processing large and diverse collections of image data. A compact color feature representation which is more appropriate for these segmented regions is also proposed. By using the color and texture features and a region-based search, we achieve a very good retrieval performance compared to the entire image based search.

Keywords**:** color, texture, content-based image retrieval, image database, image segmentation

1 INTRODUCTION

Due to the rapid advance of technologies in information processing and internet communication, people nowadays have been overwhelmed by the fast accumulation of digital information such as text, image, video, and audio around the world. However, the tools which can help people manage such a large volume of digital data and facilitate the information retrieval are still very limited. Information retrieval based on content is emerging as an important topic in many applications including multimedia databases and digital libraries.

In this paper, we present some of our on-going work in the UCSB Alexandria Digital Library (ADL) project [10]. The goal of this project is to establish an electronic library of spatially indexed data, providing internet access to a wide collection of geographic information. We have identified various components and research problems which are required to construct such an image retrieval system. An initial prototype which enables the search of large aerial photographs based on texture features has been implemented [6]. Currently, we are expanding this prototype to handle other types of image features such as color, shape, and spatial relationships. Figure 1 shows a schematic of this prototype system under development.

The image processing and analysis component contains three different layers. The first layer extracts a variety of image features such as color, texture, and shape for characterizing the low-level image information. The second layer segments images into homogeneous regions for further analysis. The results of these two layers form the input to the third layer for image interpretation. While much of the current research has focused on low level image features such as color and texture for image retrieval, not much attention has been given to issues related to segmentation and subregion retrieval. Some of the work related to extracting salient image regions include a foreground/background segmentation scheme [2], and a feature back-projection approach [9]. In this paper we present an image segmentation algorithm which is designed to meet two goals: easy to use (in terms of the number of parameters need to be specified) and capable of processing large and diverse collections of images. A color feature representation which is appropriate for the segmented regions is also proposed. As can be seen from the experimental results, image segmentation helps in developing a better representation for region-based search, and thus dramatically improves the retrieval performance compared to a global search.

In this paper, we focus on the integration of color and texture features for image retrieval. Details of the visual thesaurus construction and learning similarity measures can be found in [3, 4, 6]. In the next section, we introduce a compact color feature representation scheme for segmented regions and briefly review our texture feature representation. Section 3 discusses the

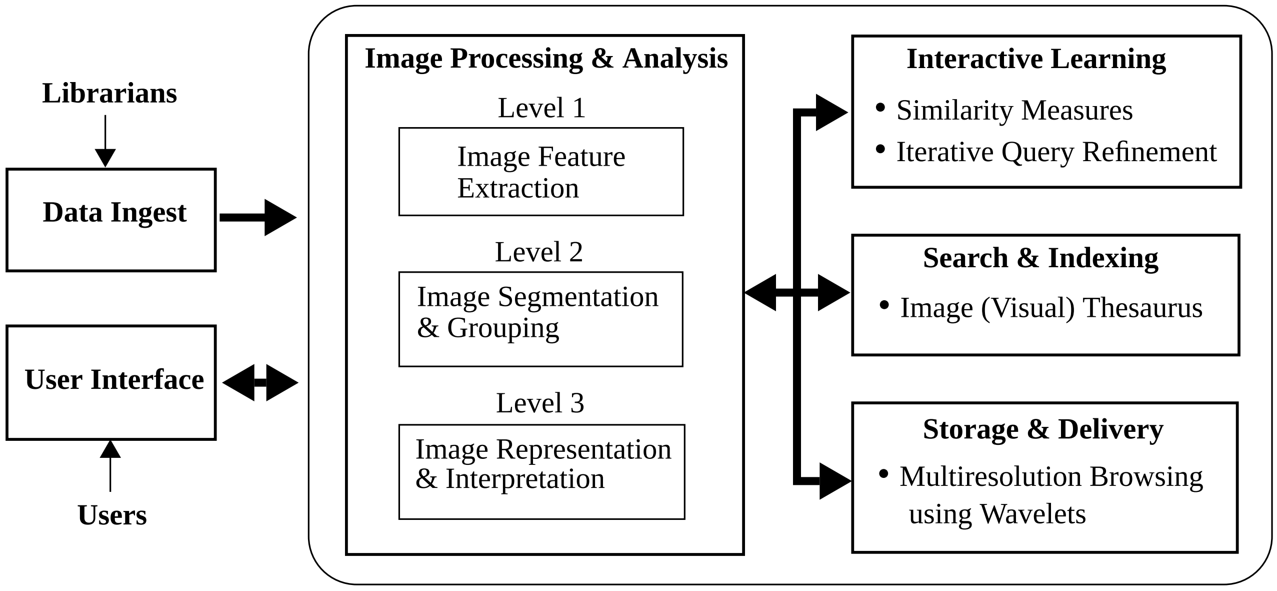

Librarians
Data Ingest
User Interface
Users
Image Processing & Analysis
Level 1
Image Feature Extraction
Level 2
Image Segmentation & Grouping
Level 3
Image Representation & Interpretation
Interactive Learning
• Similarity Measures
• Iterative Query Refinement
Search & Indexing
• Image (Visual) Thesaurus
Storage & Delivery
• Multiresolution Browsing using Wavelets

FIGURE 1. Various components of the image retrieval system which is currently under development.

image segmentation algorithm. The experimental results of using color and texture features for region-based image search in a color photo database is provided in Section 4. We conclude with some discussions and future work in Section 5.

2 COLOR AND TEXTURE FEATURE REPRESENTATION

2.1 Color Features

Color histogram has been widely used as a color feature representation for image retrieval [8, 11]. Although it has shown to be an effective method for searching images based on the entire image property, it may not provide a good representation for subregion/object retrieval. Besides, the large volume of feature vectors make the storage and indexing problem more complicated.

In this paper we propose a compact color representation scheme which is applicable when the input image has been segmented into a number of regions. The color features are extracted from each segmented region. Most natural scenes only require a few number of colors to describe its color content without significantly affecting the perceptual quality. For example, a scene of yellow flowers with green leaves basically has two colors; yellow and green. For this reason, we design a color feature that directly stores the best representing colors and their corresponding percentages in a given region.

Let us first consider the construction of global color representation. In order to have a manageable number of colors to represent the images in the database, a vector quantization scheme called Generalized Lloyd algorithm (GLA) [1] is used to quantize the RGB color space. In our experiments, we set the number of colors in the color codebook to 256. Let us denote this color codebook as $\boldsymbol{C}$ where $\boldsymbol{C} = \{C_1, C_2, \ldots, C_{256}\}$ and each color $C_i = (r_i, g_i, b_i)$ is a 3-dimensional RGB color vector. The 2,500 color images in the database are used as the training data. The GLA basically contains the following steps:

1. Begin with an initial codebook $\boldsymbol{C}_1$. Set $m = 1$.
2. Given a codebook, $\boldsymbol{C}_m = \{C_i\}$, find the optimal partition into quantization cells, that is, use the nearest neighbor condition to form the nearest neighbor cells (notice X is the color of pixels in the training images):

$$S_i = \{X | d(X, C_i) \le d(X, C_j);\ \text{all } j \ne i\}$$

3. Using the centroid of each cell S_i to form the new codebook $\boldsymbol{C}_{m+1}$.
4. Compute the average distortion for $\boldsymbol{C}_{m+1}$. If it has changed by a small enough amount since the last iteration, stop. Otherwise set $m+1 \rightarrow m$ and go to step 2.

The color feature extraction is to find the fewest number of colors from the color codebook $\boldsymbol{C}$ that adequately describe a given image region. This is done by sequentially increasing the number of colors to cluster the colors in that region until either of the following stopping criteria is met:

1. The number of colors has reached the maximum number of colors allowed (20 in our experiments).
2. The mean squared error of the color clustering is below a pre-defined threshold.

The resulting number of colors in our experiment is in the range of 10 to 15. These colors are then represented by their closest match in the color codebook $\boldsymbol{C}$. The percentage of each color (i.e., the number of pixels of that color divided by the total number of pixels in the region) is also computed and stored. The color feature is defined as

$$\boldsymbol{f}_c = \{ (I_j, P_j) \mid I_j \in \{1, 2, \ldots, 256\}, 0 \le P_j \le 1, \sum_{1 \le j \le N} P_j = 1, \text{and } 1 \le j \le N \} \tag{1}$$

where I_j is the index into the color codebook $\boldsymbol{C}$, P_j is the corresponding percentage, and N is the total number of colors in the region.

This color feature representation can be considered as a quantized version of the color histogram. This representation scheme has several advantages. First, it best represents the original color content in terms of minimizing the mean square error using a small number of colors. Second, this color feature is very compact. By taking advantage of the fact that human eyes can not distinguish close colors very well and that most segmented images regions contain only a very small set of colors, this method extracts the most prominent and distinctive colors from the region. It greatly reduces the amount of feature data for storage and indexing. Furthermore, this representation facilitates queries such as “Find me all image regions that have 50% red and 30% green.”

2.1.1 Color Distance Measure

Given two image regions A and B, suppose region A has color feature $\{ (I_a, P_a) \mid 1 \le a \le N_a \}$ and region B has color feature $\{ (I_b, P_b) \mid 1 \le b \le N_b \}$, where N_a and N_b denote the corresponding number of colors in their color features. Now let us first define

$$W(I_a, I_b) = \left\| C_{I_a} - C_{I_b} \right\| = \sqrt{\left(r_{I_a} - r_{I_b}\right)^2 + \left(g_{I_a} - g_{I_b}\right)^2 + \left(b_{I_a} - b_{I_b}\right)^2} \tag{2}$$

which is the Euclidean distance measure between any given two colors from the color codebook $\boldsymbol{C}$. It can be pre-computed and stored as a table. Now identify the best matched color k from the region B which has the minimum distance to the color I_a:

$$k = \arg\min_{1 \le b \le N_b} W(I_a, I_b) , \tag{3}$$

Use this to compute:

$$D[(I_a, P_a), B] = \left| P_a - P_k \right| \cdot W(I_a, I_k) \tag{4}$$

where $D[(I_a, P_a), B]$ is a distance measure between the given color element (I_a, P_a) and the set of color elements $\{ (I_b, P_b) \mid 1 \le b \le N_b \}$ in region B. $D[(I_b, P_b), A]$ can be computed in a similar manner. Thus, for each color in A, the closest color in B is found and the distance is calculated. These distance are then summed over all colors in A with the distance of color percentages as a weighting factors. The same process is also performed for each color in B. The distance between the regions A and B is then defined as follows:

$$d(A, B) = \sum_{1 \le a \le N_a} D[(I_a, P_a), B] + \sum_{1 \le b \le N_b} D[(I_b, P_b), A] \tag{5}$$

Note that $d(A, B) = d(B, A)$.

2.2 Texture Features

In [7] we have proposed a texture feature extraction scheme based on a Gabor decomposition. A comprehensive evaluation and comparison with other multi-resolution texture features using the Brodatz texture database was also provided. The conclusion was that these Gabor features provide excellent pattern retrieval performance. A brief review of the texture feature extraction from [7] is given below. First consider a prototype Gabor filter:

$$h(x, y) = \left(\frac{1}{2\pi\sigma_x\sigma_y}\right)\exp\left[-\frac{1}{2}\left(\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2}\right)\right] \cdot \exp\left[2\pi j W x\right] \tag{6}$$

A bank of Gabor filters can be generated by dilating and rotating the above function:

$$h_{i,j}(x, y) = a^{-i} h(x', y'),\ i, j = \text{integer} \tag{7}$$

$$x' = a^{-i}(x\cos\theta + y\sin\theta),\ y' = a^{-i}(-x\sin\theta + y\cos\theta),$$

where $\theta = j\pi/K$ and K is the total number of orientations. The scale factor a^{-i} is meant to ensure the equal energy among different filters. These Gabor filters can be considered as orientation and scale tunable edge and line (bar) detectors. The statistics of the detected features can be used to characterize the underlying texture information. Given an image $I(x, y)$, a Gabor decomposition can be obtained by

$$O_{i,j}(x, y) = \int I(x, y)\, h^*_{i,j}(x - x_1, y - y_1)\, dx_1 dy_1 \tag{8}$$

where * indicates the complex conjugate. Because $O_{i,j}(x, y)$ is a complex number, it can be further represented as $O_{i,j}(x, y) = m_{i,j}(x, y) \exp[\phi_{i,j}(x, y)]$. A simple texture feature representation can be constructed using the mean and standard deviation of the amplitude information:

$$\mu_{ij} = \iint m_{i,j}(x, y)\, dxdy,\ \sigma_{ij} = \sqrt{\iint (m_{i,j}(x, y) - \mu_{ij})^2 dxdy} \tag{9}$$

$$\boldsymbol{f}_t = \left[\mu_{00}\ \sigma_{00}\ \mu_{01}\ \cdots\ \mu_{(S-1)(K-1)}\ \sigma_{(S-1)(K-1)}\right]. \tag{10}$$

Four different scales, $S = 4$, and six orientations, $K = 6$, are used in the following experiments. This results in a feature vector of length 48. The normalized Euclidean distance is used to measure the distance between two texture features.

3 IMAGE SEGMENTATION AND GROUPING

In order to make the individual object or region of the images indexable, the input images have to be first partitioned into homogeneous regions or objects at the ingest time into the database. Each of the regions is represented by a set of image features which can be matched during the query time. However, as can be seen from most of the existing systems for content-based image retrieval, the image segmentation component is either missing or not performing well. This is due to the fact that it is difficult to design a robust image segmentation scheme, especially for a large variety of image data. Although considerable research have been done in this area, image segmentation continues to be a very challenging research problem in computer vision.

In the following we will present our recent work in developing an image segmentation tool. Its objective is to provide users an easy way to process large and diverse collections of images. The user defined parameters that control segmentation are:

1. Type of image property which should be used to detect image boundaries: color (gray intensity) only or combination of texture and color (gray intensity).
2. Scale parameter (for controlling the smoothing and resolution) to localize the desirable image boundaries.
3. Desirable number of image regions and objects (roughly).

This information is then used to adjust the internal parameters of the image segmentation scheme which consists of the following three stages:

1. Edge flow computation for identifying and integrating different types of image boundaries.

2. Edge flow propagation and boundary detection.

3. Boundary connection and region merging.

The edge flow computation is controlled by the given image property and scale parameter and the last region merging procedure is guided by the user's preferred number of regions and objects. Each of these stages is briefly explained in the following, and a more detailed description of the whole algorithm can be found in [5].

3.1 Edge Flow [5]

Most natural images are usually made of various types of boundaries created by the changes in cues such as color, texture, or phase. A unified "edge flow" model is proposed to detect discontinuities in these image features. Let us define the general form of edge flow vector F at image location s with an orientation θ as:

$$F(s, \theta) = \boldsymbol{F}[E(s, \theta), P(s, \theta), P(s, \theta + \pi)] \tag{11}$$

where $E(s, \theta)$ is the edge energy at location s along the orientation θ, $P(s, \theta)$ represents the probability of finding the image boundary if the corresponding flow at location s "flows" in the direction θ, and $P(s, \theta + \pi)$ represents the probability of finding the image boundary if the corresponding flow at location s flows backwards, i.e., in the direction $\theta + \pi$. The steps for detecting image boundaries is summarized as follows:

- At each image location, we first compute its local edge energy and estimate the corresponding flow direction.
- The local edge energy is iteratively propagated to its neighbor if the edge flow of the corresponding neighbor points in a similar direction.
- The edge energy stops propagating to its neighbor if the corresponding neighbor has an opposite direction of edge flow. In this case, these two image locations have both their edge flows pointing at each other indicating the presence of a boundary between the two pixels.
- After the flow propagation reaches a stable state, all the local edge energies will be accumulated at the nearest image boundaries. The boundary energy is then defined as the sum of the flow energies from either side of the boundary.

In contrast to the traditional "static" edge detection approaches which directly localizes edges at the local maxima of the gradient in intensity, the edge flow model results in a "dynamic" boundary detection scheme. The flow direction gives the direction with the most information change in feature space. Since any of the image attributes such as color, texture, or their combination can be used to define the edge flow, this scheme provides an easy framework for integrating different types of image information for boundary detection.

3.1.1 Color Edge Flow

Suppose the user has specified the preferable scale as σ, then the image at this given scale can be obtained by smoothing the original image, denoted as $\{r(x, y), g(x, y), b(x, y)\}$, which correspond to the three color bands RGB, with a Gaussian kernel $G_\sigma(x, y)$. The edge flow energy $E_r(s, \theta)$ of the red color band $r(x, y)$ is defined to be the magnitude of the gradient of the smoothed color $r_\sigma(x, y)$ along the orientation θ:

$$E_r(s, \theta) = \frac{\partial}{\partial \boldsymbol{n}} r_\sigma(x, y) = \left| \frac{\partial}{\partial \boldsymbol{n}} [r(x, y) * G_\sigma(x, y)] \right| = \left| r(x, y) * \frac{\partial}{\partial \boldsymbol{n}} G_\sigma(x, y) \right| \tag{12}$$

where $s = (x, y)$ and $\boldsymbol{n}$ represents the unit vector in the gradient direction. We can rewrite (12) as

$$E_r(s, \theta) = |r(x, y) * GD_{\sigma, \theta}| \tag{13}$$

where $GD_{\sigma, \theta}$ represents the Gaussian derivative along the orientation θ. The edge flow energy $E_g(s, \theta)$ and $E_b(s, \theta)$ for the other two color bands are computed in a similar manner.

For each of the edge energy, we now consider two possible flow directions; the forward (θ) and the backward ($\theta + \pi$), and estimate the probability of finding the nearest boundary in each of the two directions. These probabilities can be obtained by looking into the prediction errors toward the surrounding neighbors in the two directions. Consider the use of red color information at location s to predict its neighbor in the direction θ. Ideally they should have similar color intensity if they belong to the same object and the prediction error can thus be computed as

$$Error_r(s, \theta) = \left| r_\sigma(x + d\cos\theta, y + d\sin\theta) - r_\sigma(x, y) \right| = \left| r(x, y) * DOOG_{\sigma, \theta}(x, y) \right| \tag{14}$$

where d is the distance of the prediction and $DOOG_{\sigma, \theta}(x, y) = G_\sigma(x, y) - G_\sigma(x + d\cos\theta, y + d\sin\theta)$. The distance d should be proportional to the scale which the image is being analyzed. In the experiments, we choose $d = 4\sigma$. Because a large prediction error implies a higher probability of finding a boundary in that direction, we assign the probabilities of edge flow direction in proportion to its corresponding prediction errors:

$$P_r(s, \theta) = \frac{Error_r(s, \theta)}{Error_r(s, \theta) + Error_r(s, \theta + \pi)} \tag{15}$$

3.1.2 Texture Edge Flow

Based on a similar strategy, we can also formulate an edge flow for detecting texture boundaries. Consider the use of texture features extracted from a Gabor decomposition. By taking the amplitude of the filtered output across different filters at the same location (x, y), we form a texture feature vector

$$\Psi(x, y) = [m_1(x, y), m_2(x, y), \ldots, m_N(x, y)] \tag{16}$$

which characterizes the local spectral energies in different spatial frequency bands. The N represents the lowest spatial frequency of the Gabor filters, and it should be controlled by the scale parameter σ because only those “patterns with scales smaller than σ pixels” are considered as texture. In our experiments, we choose the lowest frequency of the Gabor filters to be $1/(4\sigma)$ cycles/pixel.

By using the texture features Ψ, the texture edge energy $E_t(s, \theta)$, which is used to measure the change in local texture information, is given by

$$E_t(s, \theta) = \sum_{1 \le i \le N} \left| m_i(x, y) * GD_{\sigma, \theta}(x, y) \right| \cdot w_i \tag{17}$$

where $w_i = 1/\|\alpha_i\|$ and $\|\alpha_i\|$ is the total energy of the subband i. The weighting coefficients w_i normalize the contribution of edge energy from the various frequency bands.

Similar to the color edge flow, the direction of texture edge flow can be estimated based on the texture prediction error at a given location:

$$Error_t(s, \theta) = \sum_{1 \le i \le N} \left| m_i(x, y) * DOOG_{\sigma, \theta}(x, y) \right| \cdot w_i \tag{18}$$

which is the weighted sum of prediction errors from each texture feature map. Thus, the probabilities $P_t(s, \theta)$ and $P_t(s, \theta + \pi)$ of the flow direction can be computed using (15).

3.1.3 Combining Color and Texture Edge Flow

Now let us consider the use of color and texture information for image segmentation. The edge flows obtained from both color and texture attributes can be combined together to form a single edge flow field:

$$E(s, \theta) = \sum_{a \in \{r, g, b, t\}} E_a(s, \theta) \cdot w(a) \text{ , and } \sum_{a \in \{r, g, b, t\}} w(a) = 1 \tag{19}$$

$$P(s, \theta) = \sum_{a \in \{r, g, b, t\}} P_a(s, \theta) \cdot w(a) \tag{20}$$

$w(a)$ is the weighting coefficient among different types of image properties, and it is adjusted according to the user's preference. In our experiments, if the user decides the segmentation process to be color only, then we set $w(t) = 0$ and $w(r) = w(g) = w(b) = 1/3$. If the user specifies the use of color and texture together, then we set $w(t) = 0.4$ and $w(r) = w(g) = w(b) = 0.2$.

After the previous integration, we identify the best direction of finding the nearest boundary using the following scheme. Given edge flows $\{\boldsymbol{F}[E(s,\theta), P(s,\theta), P(s,\theta+\pi)]\,|_{0 \le \theta < \pi}\}$, we first look for a continuous range of flow directions which maximizes the sum of probabilities in that half plane:

$$\Theta(s) = \underset{\theta}{\arg\max} \left\{ \sum_{\theta \le \theta' < \theta + \pi} P(s, \theta') \right\} \tag{21}$$

Then, the final resulting edge flow is defined to be the vector sum of the edge flows with their directions in the identified range, and is given by

$$\vec{\boldsymbol{F}}(s) = \sum_{\Theta(s) \le \theta < \Theta(s) + \pi} E(s, \theta) \cdot \exp(j\theta) , \tag{22}$$

where $\vec{\boldsymbol{F}}(s)$ is a complex number with its magnitude representing the resulting edge energy and angle representing the flow direction.

3.2 Edge Flow Propagation and Boundary Detection

After the edge flow $\vec{\boldsymbol{F}}(s)$ of an image is computed, boundary detection can be performed by iteratively propagating the edge flow and identifying the locations where two opposite direction of flows encounter each other. At each location, the local edge flow is transmitted to its neighbor in the direction of flow if the neighbor also has a similar flow direction (the angle between them is less than 90 degrees). After the edge flow propagation reaches a stable state, the image boundaries are detected at the locations which have non-zero edge flow coming from two opposing directions. Figure 2(c)-(d) illustrate the detected image boundaries.

3.3 Boundary Connection and Region Merging

After boundary detection, disjoint boundaries are connected to form closed contours and result in a number of image regions. The basic strategy for connecting the boundaries are summarized as follows.

- For each open contour, we associate a neighborhood search size proportional to the length of the contour. This neighborhood is defined as a half circle with its center located at the unconnected end of the contour.
- The nearest boundary element which is within the half circle is localized.
- If such a boundary element is found, a smooth boundary segment is generated to connect the open contour to another nearest boundary element.
- This process is repeated few times (typically 2-3 times) till all salient open contours are closed.

At the end, a region merging algorithm is used to merge similar regions based on a measurement that evaluates the distances of region color and texture features, the sizes of regions, and the percentage of original boundary between the two neighboring regions. This algorithm sequentially reduces the total number of regions each time by checking if the user's preferred number has been approached to the best extent. Figure 2(e) shows the result of this stage.

This image segmentation algorithm has been applied to segment about 2,500 real natural images from Corel color photo gallery, and has resulted in visually acceptable performance on this diverse image collection. A tool which displays the segmentation result and allows some minor modification is also provided (as shown in Figure 2(f)). The user can remove the boundaries which he thinks are not necessary. Because no additional boundary is created, this task can be performed very fast. Figure 3 shows some of the image segmentation results.

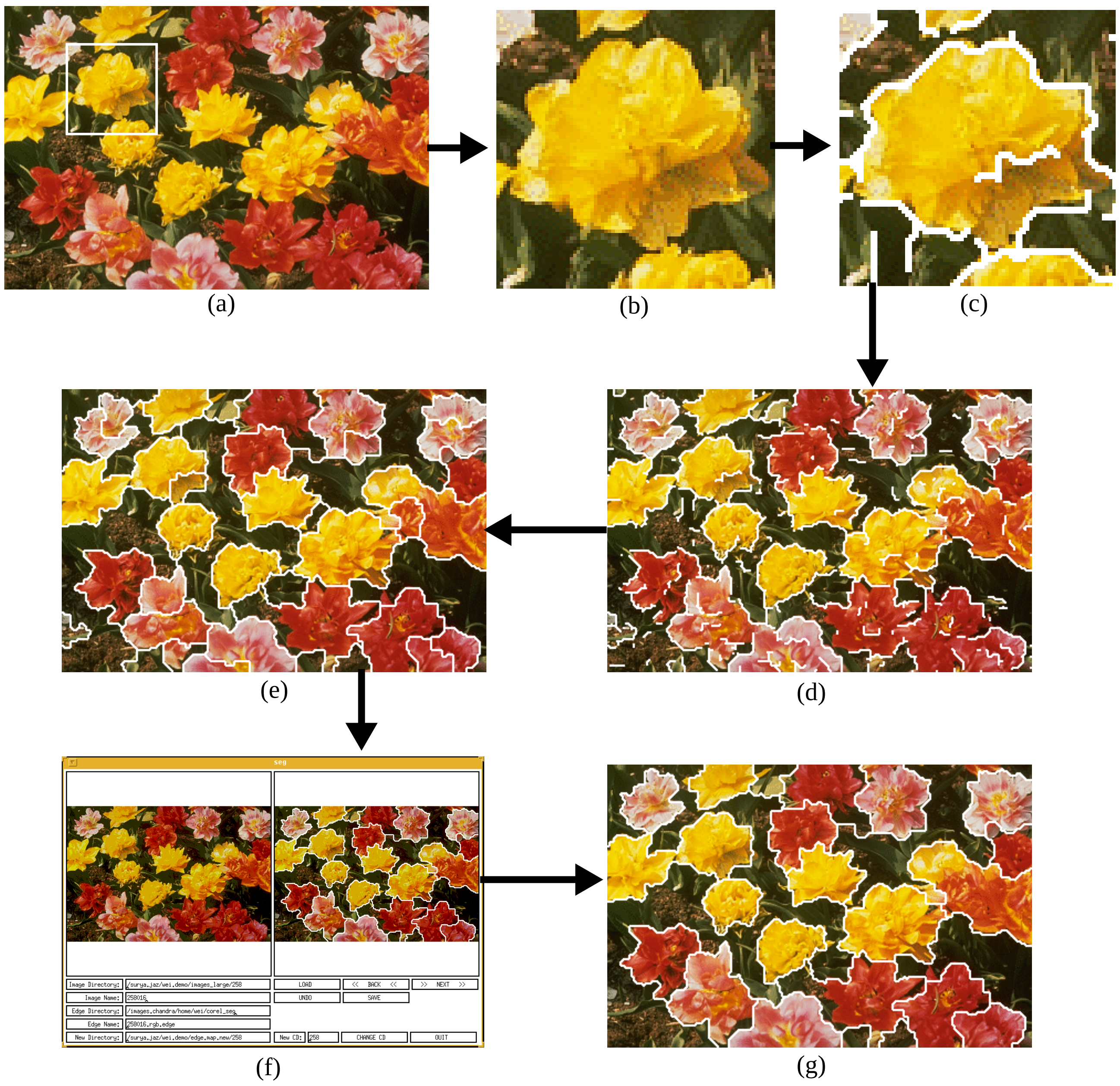


Image Directory: /surya.jaz/wei.demo/images_large/258
Image Name: 258016
Edge Directory: /images.chandra/home/wei/corel_seg
Edge Name: 258016.rgb.edge
New Directory: /surya.jaz/wei.demo/edge.map.new/258
New CD: 258

FIGURE 2. (a) Input image, (b) a small portion of the input image, (c)-(d) the results of boundary detection, and (e) after the boundary connection and region merging, (f) shows the tool where the user can further improve the image segmentation result by removing unnecessary boundaries, and (g) shows the final segmentation result.

4 EXPERIMENTS

We have implemented a system which allows users to specify a particular image region as a query, then use the color and texture features to search similar image regions from the database. Currently our database contains 2,500 color images which are obtained from Corel photo gallery (volume 7, nature). They are organized into 25 different categories with 100 images per category.

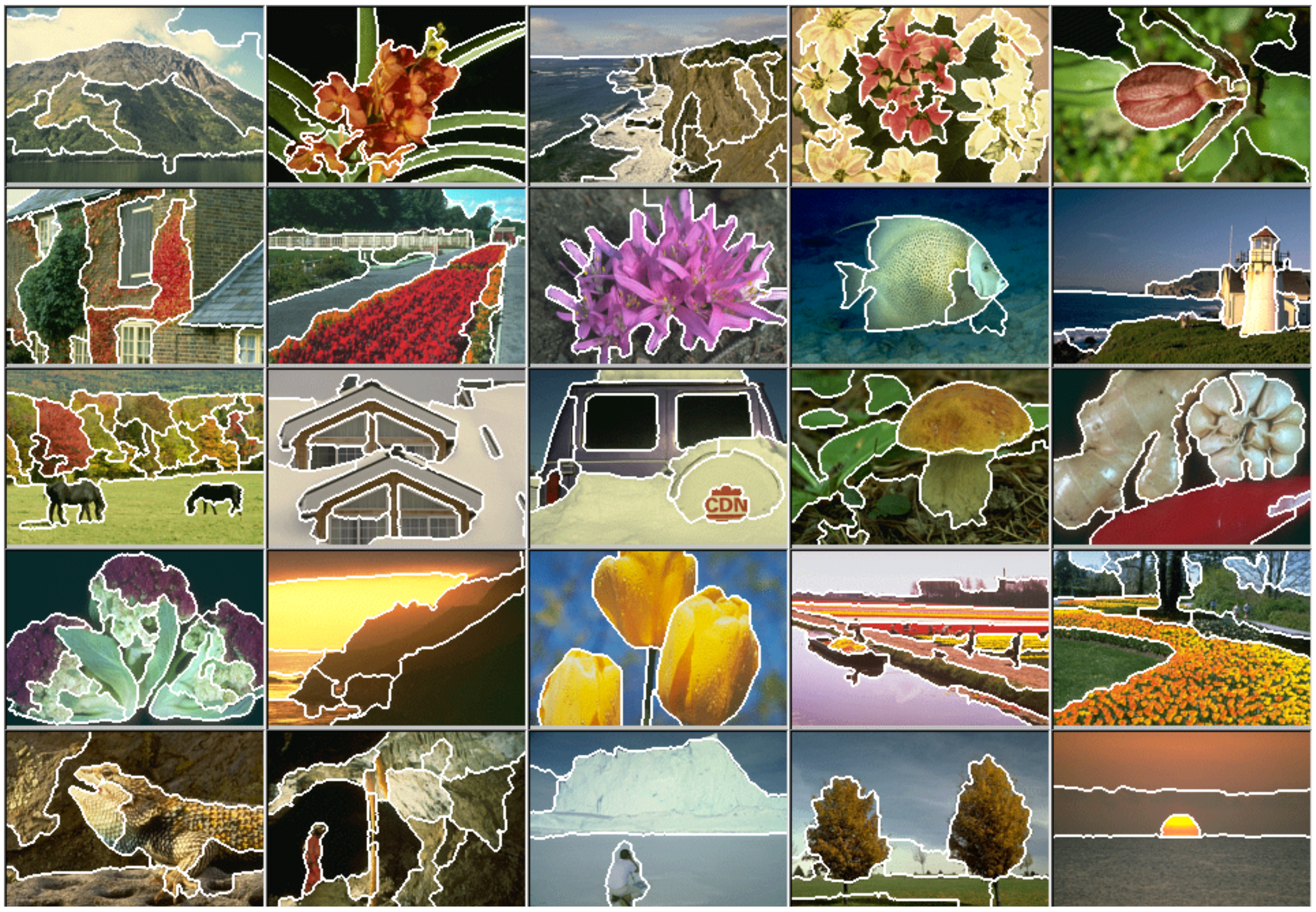


FIGURE 3. Examples of the image segmentation results. For color pictures, see http://vivaldi.ece.ucsb.edu/projects/ipinadl/wei_demo/demo_segment.html.

All the image features and the corresponding segmentation of these images are pre-computed and stored into the database. The parameters for controlling the segmentation algorithm are specified on a category-by-category base. For those images with uniform color and no texture, we specify the type of processing to “color only” and also set the scale parameter smaller (ranging from 2 to 6 pixels) to obtain more precise object boundaries. For those images with textures such as gardens and mountains, we use both color and texture to perform image segmentation and set a larger scale parameter (ranging from 7 to 20 pixels). The number of preferable regions is set to 6-12 for each image, and the total number of segmented regions in our database is about 26,000. In other words, each image is partitioned into 10 regions on the average. Following the segmentation process, the image region map (or logical map) is used to guide the color and texture feature extraction so that only the pixels of same region contribute the corresponding region features. Both the region map and the region feature vector are stored as part of the image meta-data.

Figure 4(a) shows the retrieval system which displays segmented images so that users can select one of the regions as a query. The users can also specify the relevant importance of the color and texture features for conducting the search. Figure 4(b) shows an image retrieval example using the region of yellow flowers with green leaves as a query. As can be seen, the system successfully retrieves images with perceptually similar regions. Without the pre-segmentation, the global feature representation will mix the information of irrelevant background and objects with the desirable region, so that the colors of sky, building, river, and boat will deteriorate the retrieval performance of yellow flowers with green leaves. As a result, Image segmentation helps in developing a better representation for region based search and indexing, and significantly improves retrieval results compared to a global search (based on the entire image). Figure 5 provides two other retrieval examples where the corresponding matched regions are outlined.

5 DISCUSSIONS AND FUTURE WORK

In this paper, we have presented some of our recent work in the ADL project, which include a robust and general-purpose image segmentation scheme, a color feature representation appropriate for segmented regions, and a tool which provides region-based search of images using both color and texture information. The proposed segmentation algorithm is shown to be powerful and reliable in processing large and diverse collections of image data. With the capability of analyzing and representing individual image regions, the performance of image retrieval improves dramatically compared to the global search. Much work remains to be done in evaluating the performance of image retrieval systems of this type.

Currently we are expanding the size of the database (to about 12,500 images), and developing a search mechanism which facilitates the use of combined color and texture information, so that the retrieval can be done more efficiently. Segmentation also will help in addressing more complex queries which include spatial relationships, another research direction that we are pursuing at present.

ACKNOWLEDGMENTS

This research is supported by the Alexandria Digital Library project at the University of California, Santa Barbara under NSF grant number IRI-94-11330.

REFERENCES

[1] A. Gersho and R. M. Gray, Vector Quantization and Signal Processing, Kluwer Academic Publishers, 1992.

[2] Qian Huang et al., “Foreground/background segmentation of color images by integration of multiple cues,” IEEE Int. Conf. on Image Processing, Vol. 1, pp. 246-249, Washington, DC, Oct. 1995.

[3] W. Y. Ma and B. S. Manjunath, “Texture features and learning similarity,” Proc. of IEEE Int. Conf. on Computer Vision and Pattern Recognition, pp. 425-430, San Francisco, CA, June 1996.

[4] W. Y. Ma and B. S. Manjunath, “A Pattern thesaurus for browsing large aerial photographs,” ECE Technical Report #96-10, University of California, Santa Barbara, June 1996.

[5] W. Y. Ma and B. S. Manjunath, “Edge flow: a framework of boundary detection and image segmentation,” ECE Technical Report #97-02, University of California, Santa Barbara, Jan. 1997.

[6] B. S. Manjunath and W. Y. Ma, “Browsing large satellite and aerial photographs,” IEEE Int. Conf. on Image Processing, Vol. 2, pp. 765-768, Lausanne, Switzerland, Sep. 1996.

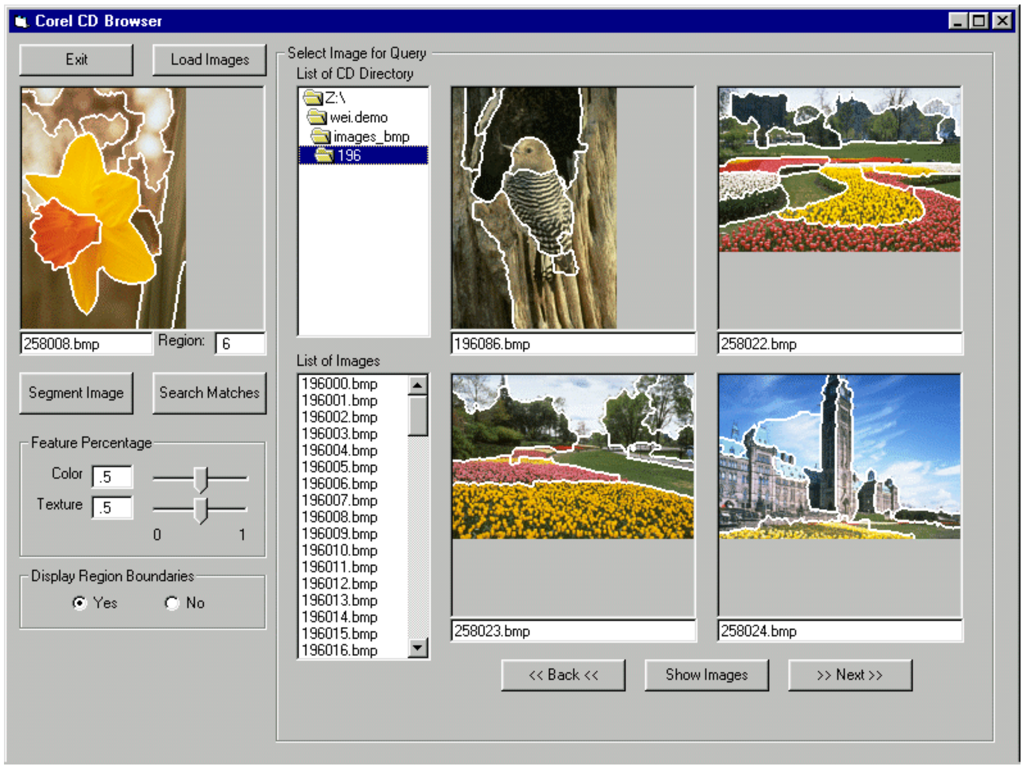
[7] B. S. Manjunath and W. Y. Ma, “Texture features for browsing and retrieval of image data,” IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 18, No. 8, pp. 837-842, Aug. 1996.

[8] W. Niblack et al., “The QBIC Project: Querying images by content using color, texture and shape,” Proc. SPIE 1908, Storage and Retrieval for Image and Video Databases, pp. 173-187, San Jose, CA, Feb. 1993.

[9] J. R. Smith and S. F. Chang, “Local color and texture extraction and spatial query,” IEEE Int. Conf. on Image Processing, Vol. 3, pp. 1011-1014, Lausanne, Switzerland, Sep. 1996.

[10] Terence R. Smith, “A digital library for geographically referenced materials,” *Computer*, pp. 54-60, May 1996.

[11] M. J. Swain and D. H. Ballard, “Color indexing,” International Journal of Computer Vision, Vol. 7, No. 1, pp.11-32, 1991.



258008.bmp Region: 6

Feature Percentage
Color .5
Texture .5
0 1

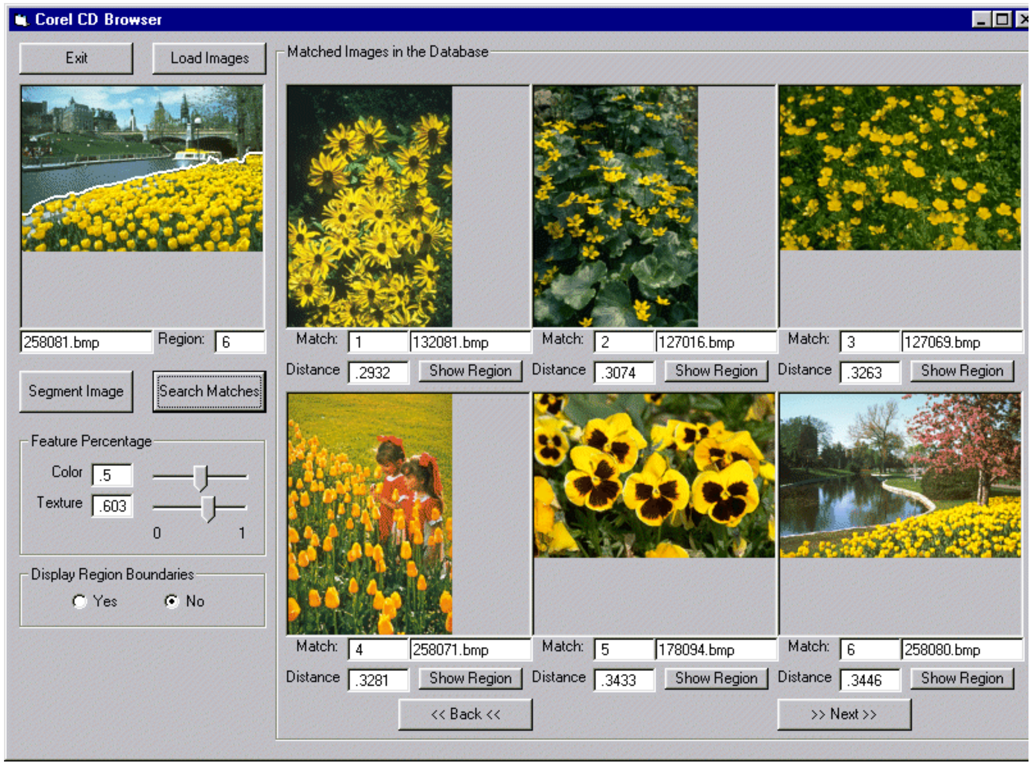
Display Region Boundaries
[✓ Yes] No

Z:\
wei.demo
images_bmp
196

196000.bmp
196001.bmp
196002.bmp
196003.bmp
196004.bmp
196005.bmp
196006.bmp
196007.bmp
196008.bmp
196009.bmp
196010.bmp
196011.bmp
196012.bmp
196013.bmp
196014.bmp
196015.bmp
196016.bmp

196086.bmp
258022.bmp
258023.bmp
258024.bmp

(a)



258081.bmp Region: 6

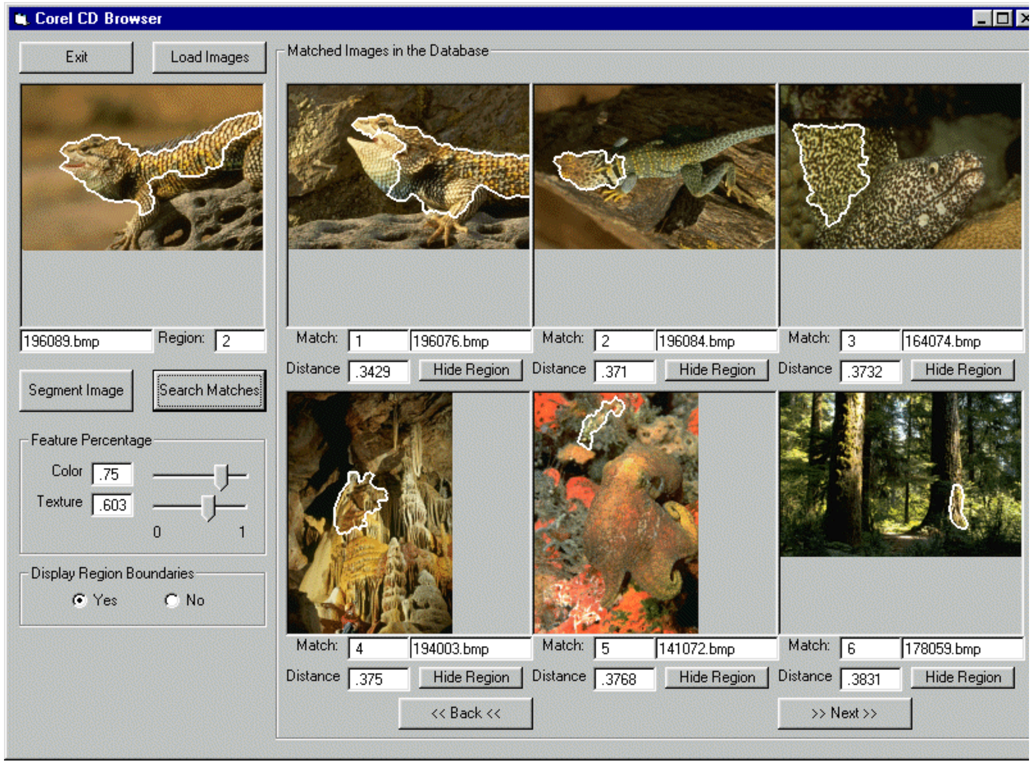
Feature Percentage
Color .5
Texture .603
0 1

Display Region Boundaries
Yes [✓ No]

Match: 1 132081.bmp Distance .2932
Match: 2 127016.bmp Distance .3074
Match: 3 127069.bmp Distance .3263
Match: 4 258071.bmp Distance .3281
Match: 5 178094.bmp Distance .3433
Match: 6 258080.bmp Distance .3446

(b)

FIGURE 4. (a) A snapshot of the image retrieval system which displays images with segmented regions. The user can select an individual region as a query to conduct the search. (b) A retrieval example using a region which contains yellow flowers and green leaves. Color pictures are available on the web at http://vivaldi.ece.ucsb.edu/projects/ipinadl/wei_demo/demo_corel.htm.



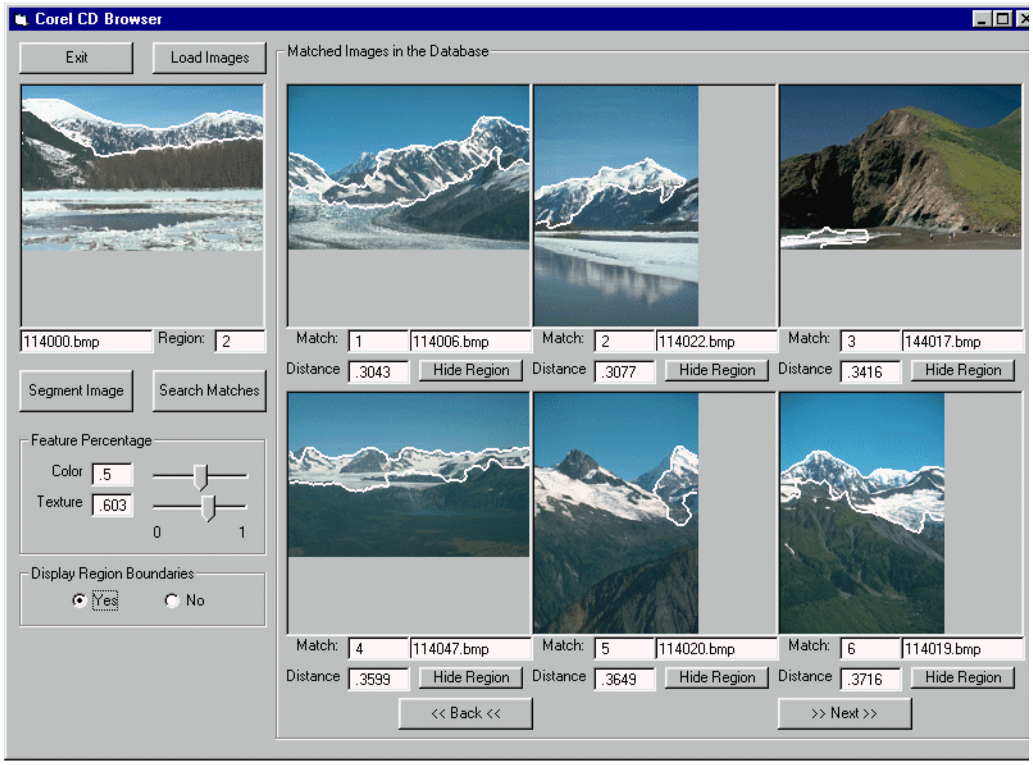
Matched Images in the Database

196089.bmp Region: 2

Feature Percentage
Color .75
Texture .603
0 1

Display Region Boundaries
[✓ Yes] No

Match: 1 196076.bmp Distance .3429
Match: 2 196084.bmp Distance .371
Match: 3 164074.bmp Distance .3732
Match: 4 194003.bmp Distance .375
Match: 5 141072.bmp Distance .3768
Match: 6 178059.bmp Distance .3831



Matched Images in the Database

114000.bmp Region: 2

Feature Percentage
Color .5
Texture .603
0 1

Display Region Boundaries
[✓ Yes] No

Match: 1 114006.bmp Distance .3043
Match: 2 114022.bmp Distance .3077
Match: 3 144017.bmp Distance .3416
Match: 4 114047.bmp Distance .3599
Match: 5 114020.bmp Distance .3649
Match: 6 114019.bmp Distance .3716

FIGURE 5. Examples of region-based image retrieval. Both of the query regions and the best matched regions are outlined. For color pictures, see http://vivaldi.ece.ucsb.edu/projects/ipinadl/wei_demo/demo_corel.html.