

NeTra: A toolbox for navigating large image databases

Wei-Ying Ma¹, B. S. Manjunath²

¹ Hewlett-Packard Laboratories, Palo Alto, CA 94304-1126, USA; e-mail: wei@hpl.hp.com

² Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106-9560, USA; e-mail: manj@ece.ucsb.edu

Abstract. We present here an implementation of NeTra, a prototype image retrieval system that uses color, texture, shape and spatial location information in segmented image regions to search and retrieve similar regions from the database. A distinguishing aspect of this system is its incorporation of a robust automated image segmentation algorithm that allows object- or region-based search. Image segmentation significantly improves the quality of image retrieval when images contain multiple complex objects. Images are segmented into homogeneous regions at the time of ingest into the database, and image attributes that represent each of these regions are computed. In addition to image segmentation, other important components of the system include an efficient color representation, and indexing of color, texture, and shape features for fast search and retrieval. This representation allows the user to compose interesting queries such as “retrieve all images that contain regions that have the color of object A, texture of object B, shape of object C, and lie in the upper of the image”, where the individual objects could be regions belonging to different images. A Java-based web implementation of NeTra is available at <http://vivaldi.ece.ucsb.edu/Netra>.

Key words: Color – Texture – Shape – Query by spatial location – Content-based image retrieval – Image databases – Image segmentation

1 Introduction

Rapid advances in computers and communication technology is pushing the existing information-processing tools to their limits. The past few years have seen an overwhelming accumulation of media-rich digital data such as images, video, and audio. The internet is an excellent example of a distributed database containing several millions of images. Other examples of large image databases include satellite and medical imagery, where often it is hard to describe or annotate the image contents. Even if it is possible for a user

to describe the contents in an unambiguous way, the large amount of data that need to be processed in applications such as medicine or geographic information systems makes it necessary that robust image analysis tools be developed for automated image annotation.

In this paper, we present NeTra, a toolbox for navigating large image collections. NeTra is implemented on the World Wide Web (<http://vivaldi.ece.ucsb.edu/Netra>) using the platform-independent Java language. NeTra uses color, texture, and shape information to organize and search the database. One of the distinguishing aspects of this system is that it automatically segments and uses localized region information in indexing images in the database. This is in contrast to global image attributes that many of the existing content-based retrieval systems use.

Several systems have been developed recently to search through image databases using color, texture, and shape attributes. These include QBIC [13], Photobook [19], Virage [3], and VisualSEEK [28]. The initial version of QBIC provided querying of the entire image and manually extracted regions. Its most recent versions have incorporated an automated foreground/background segmentation scheme [2, 11] to improve the retrieval performance. The Photobook shares many similar features to QBIC, but utilizes a more sophisticated texture and shape feature representation in addition to image segmentation. Its recent emphasis has focused on interactive learning to incorporate the user’s feedback to adjust classification and segmentation parameters based on a variety of feature representation models [19]. The Virage system [3] uses features such as color, texture, composition, and structure to search images. However, its most recent version has used several new techniques to include spatial information in the image representations. The VisualSEEK [28] proposed a feature back-projection scheme to extract salient image regions and, therefore, the system is able to provide joint content-based and spatial search capability.

1.1 Use of color, texture, and shape features for image query

Using color to index and search images dates back to some of the early work by Swain and Ballard [30] on color his-

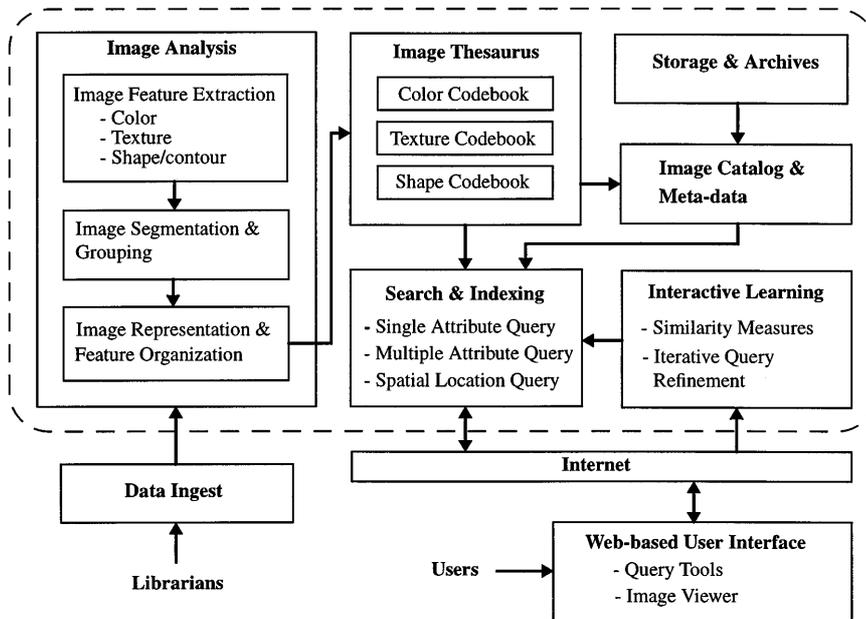


Fig. 1. Various components in the image retrieval system NeTra

tograms. Since then many variants of the histogram indexing have been proposed. The basic idea is to reduce the space of all possible colors such that the features can be efficiently organized for future search. Very often there is a transformation from the traditional RGB color space to some other space where the similarities between colors are better preserved while computing the Euclidean distances. Since color is one visual primitive that humans can easily relate to, it has received considerable attention within the image database community [4, 10, 22, 26, 31].

While texture analysis has a very rich history in image processing and vision, image databases are perhaps the first large-scale applications demonstrating the use of texture. The texture search component of the UCSB's Alexandria Digital Library (ADL) project is a very good example [15, 18, 29]. The ADL collections include aerial photographs and satellite imagery besides other geospatially referenced data such as the Gazetteer. The ADL testbed can be accessed on the web at <http://alexandria.sdc.ucsb.edu> and the users can, by various means, retrieve spatially referenced data. For the air-photos in the database, the users can also use texture as one of the attributes. Texture in these pictures has turned out to be a surprisingly powerful representation, as one can search and retrieve patterns, such as those consisting of parking lots, orchards, runways, and airplanes by using the corresponding texture features.

Shape is another low-level attribute that can be used to represent local image information. However, it is not as widespread as color and texture, as it involves extraction of region/object boundaries, a hard problem by itself. Most of the current work on using shape is for single simple to segment objects in the scene [13, 20].

Automated image segmentation is clearly a bottleneck for enhancing the retrieval performance. Although some of the existing systems have demonstrated a certain capability in extracting image regions and providing a region-based search [5], the performance of their segmentation algorithms in terms of processing large and diverse collections of image

data has not been clearly demonstrated. Lack of robust segmentation algorithms is one of the reasons why shape and other local image features have not been extensively used in image queries.

In contrast, NeTra builds on a robust segmentation algorithm [16]. Figure 1 shows a schematic of NeTra. The system advances the current technology in many ways: (a) it incorporates a very robust image segmentation scheme that partitions the image into homogeneous regions. Segmentation enables searching based on regions/objects in the image, as well as facilitating specifying spatial relationships in the query language. (b) A color representation and indexing for segmented regions is described that facilitates fast search. (c) Texture and shape attributes, together with spatial information, further help the user to compose queries more accurately, improving retrieval performance.

The entire system is implemented on the web using the Java programming language. An initial prototype of this system has been implemented in the UCSB ADL project [29], where it is used to search through large collections of aerial images using image texture [14, 18].

1.2 Overview

Figure 1 shows the main components of NeTra. At the time of data ingest, images are analyzed to extract texture and color information. Texture and color are used to partition the image into non-overlapping homogeneous segments. The region boundaries/shape, together with the region color, texture, and its spatial location, are used in representing the region within the database. Vector quantization techniques [8] are used to cluster each of the features to create a visual thesaurus [14].

One of the outstanding issues in database retrieval is that of similarity matching in the feature space. While the concept of a visual thesaurus addresses this problem to some extent, interactive learning and iterative query refinement

are needed to further improve retrieval results. These components are not yet part of the current version of NeTra. For details of the image thesaurus construction and learning similarity measures, we refer to our previous work [14, 15, 18].

The organization of the paper is as follows: the next section (Sect. 2) describes a color feature representation that is quite compact and well suited to represent color in homogeneous regions. An efficient color-indexing scheme which involves only boolean operations is presented. Section 3 briefly reviews the shape and texture features used in NeTra. Section 4 summarizes a new image segmentation scheme. Section 5 discusses query specification and retrieval and Sect. 6 provides experimental results. Section 7 concludes with discussions.

2 Color features

Color histogram is a popular color representation scheme that has been used in many image retrieval applications [13, 27, 30]. It works quite well in quantifying global color content in images. Several algorithms have been developed for matching color histograms efficiently. However, within homogeneous regions, the color content is much more sparsely distributed in the color space than the color of the whole image. Fewer colors (typically 5–15 per region) can thus be used to represent region color without affecting the perceptual quality. For example, a field of yellow poppy flowers has typically two dominant colors, yellow and green. The image itself may contain more than just this flower bed, such as a river or skyline or other objects, and may require a significantly wider spectrum of colors to adequately represent the global content.

In NeTra, each image region is represented by a subset of colors from a color codebook. The color codebook itself could be context dependent, and a different codebook can exist for different applications. From a training dataset of image samples, the codebook is constructed using the generalized Lloyd Algorithm (GLA) [8] to vector quantize colors in the RGB color space. The codebook construction is discussed in the Appendix. The codebook in our prototype contains a total of 256 colors.

In order to represent the color within each homogeneous region, we again use the GLA to cluster the local colors. One of the objectives is to represent the region with as few colors as possible. Starting with one color, the number of clusters is progressively increased until either of the following stopping criteria is met.

1. The number of color clusters 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.

Note that the color descriptor can be of varying length, depending upon the color complexity of the homogeneous region. The resulting number of color clusters in the experiments is in the range 5–15. The color feature is then defined as

$$\mathbf{f}_c = \left\{ (I_j, P_j) \mid I_j \in \{1, 2, \dots, 256\}, 0 \leq P_j \leq 1, \sum_{1 \leq j \leq N} P_j = 1, \text{ and } 1 \leq j \leq N \right\}, \quad (1)$$

where I_j is the index into the color codebook \mathbf{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 cannot distinguish close colors very well and that most segmented image 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 A color dissimilarity measure

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

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

which is the Euclidean distance between any given two colors from the color codebook \mathbf{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 :

$$k = \arg \min_{1 \leq b \leq N_b} W(I_a, I_b). \quad (3)$$

Use this to compute

$$D[(I_a, P_a), B] = |P_a - P_k| \cdot W(I_a, I_k), \quad (4)$$

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 $\{(I_b, P_b) \mid 1 \leq b \leq N_b\}$ 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. Using the color percentages as weighting factors, a cumulative distance over all colors in A is calculated. The same process is also performed for each color in B . The distance between the regions A and B is then defined as follows:

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

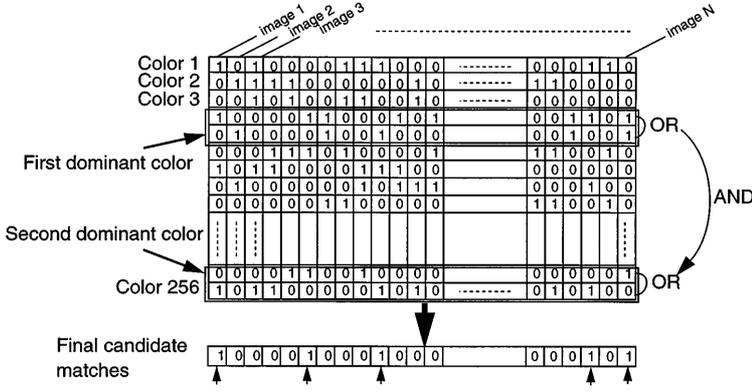


Fig. 2. An example of efficient color indexing. The table contains $256 \times N$ elements with 1 representing the existence of color in the region and 0 for non-existence. The final set of matches is obtained from the intersection (AND operation) of the two candidate lists according to the dominant colors of the query image

Note that $d(A, B) = d(B, A)$. However, $d(\cdot)$ is not a distance metric, as the triangular inequality does not hold.

2.2 Efficient color indexing

We now describe an efficient search algorithm to identify similar colors in the database. Towards this objective, we first construct an $M \times N$ binary color table $T(i, j)$, where $M = 256$ is the number of colors in the color table and N is the number of image regions in the database. Figure 2 shows an example. $T(i, j) = 1$ if the i th color is present in the j th region, otherwise it is set to zero. One bit is used to represent each element in the table.

A query object color information is represented using a feature vector similar to the one described in the previous section. The K color elements in the query color feature vector $\mathbf{f}_q = \{(I_j^{(q)}, P_j^{(q)}) | 1 \leq j \leq K\}$ are sorted such that the first color is the most dominant color (in terms of the percentage of pixels belonging to that color in the region), the second index representing the second most dominant color and so on, i.e., $P_a^{(q)} \geq P_b^{(q)}$ if $a \leq b$.

The search for images with similar colors as the query is then conducted in two phases. In the first phase, the binary color table is used to significantly prune the search space. This is described in greater detail in the following. For those image regions that survive the first phase, the more expensive dissimilarity measure (Eq. 5) is computed.

Note that the elements in binary table $T(\cdot)$ indicate either the presence or absence of a color. However, one needs to allow for “similar” colors close to the colors in the query image as well for browsing applications. This is done by specifying a similarity color table $S(i, j)$, where an entry $S(i, j)$ represents the j th most similar color index to the color C_i . Consider a color C_i that belongs to the query image. A vector $A(i)$ is now constructed as

$$A(i) = T(i, 1 : N) | T(S(i, 1), 1 : N) \dots | T(S(i, p), 1 : N), \quad (6)$$

where $|$ means the element-by-element OR operation, and p is the number of similar colors considered. The similarity is determined based on the Euclidean distance in the RGB color space. The binary vector $A(i)$ is now used to select a subset of image regions for further search as follows.

1. Begin with a $1 \times N$ binary vector $L = [1 \ 1 \ \dots \ 1]$ (with all elements as 1). Set $k = 1$.

2. Set $L \& A(I_k^{(q)}) \rightarrow L$ where the operator $\&$ represents the element-by-element AND operation.
3. Count the number of 1s in L . If it is smaller than a given threshold, or if $k = K$, then go to step 4. Otherwise set $k + 1 \rightarrow k$ and go to step 2.
4. Compute the color distance measure with the image regions whose corresponding index in the vector L is 1. Show the sorted results to the user.

Figure 2 shows an example with $p = 1$ and the top two dominant query image colors are used to reduce the search space.

Because the computations required in color indexing are mostly simple boolean operations such as bit-to-bit AND and OR, they can be implemented efficiently. Besides, the storage space required for these tables is very small. When new data is entered into the database, one can simply update the table by adding a new column vector at the end.

3 Shape and texture features

3.1 Shape features

The shape of objects or of regions of interest is another important image feature for content-based image retrieval. Over the past few decades many approaches to the characterization of shape and shape similarity have been proposed in the literature. An important class of shape analysis algorithms is based on the representation of the outer boundaries of objects [32]. We have adapted some of the existing well-known shape representations with some minor modifications. For the sake of completeness, we briefly describe the actual implementation used in NeTra.

3.1.1 Contour representation

The contour of a 2D object is considered as a closed sequence of successive boundary pixel coordinates (x_s, y_s) , where $0 \leq s \leq N - 1$ and N is the total number of pixels on the boundary. An example of this coordinate chain is shown in Fig. 3. In our experiments, three types of contour representations are derived from these boundary coordinates, which include *curvature* function, *centroid distance*, and *complex coordinate* functions.

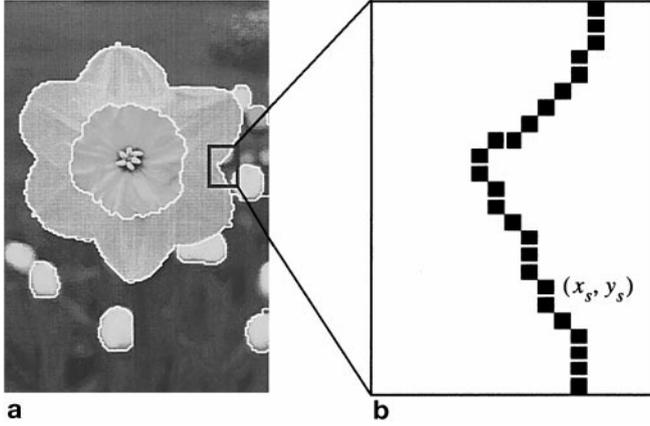


Fig. 3a,b. The object boundary is represented as a discrete coordinate chain. **a** A segmented flower image. **b** A small portion of the outer boundary of flower

The curvature at any point along the contour is defined as the rate of change in tangent direction of a contour, as a function of arc length. Let us denote the curvature function as $K(s)$, which can be expressed as

$$K(s) = \frac{d}{ds} \theta(s), \quad (7)$$

where $\theta(s)$ is the tangent angle function of the contour and is defined as

$$\theta(s) = \text{atan} \left(\frac{\dot{y}_s}{\dot{x}_s} \right), \quad \left(\dot{y}_s = \frac{dy_s}{ds} \text{ and } \dot{x}_s = \frac{dx_s}{ds} \right). \quad (8)$$

When implementing the above formula in a digital image where the contour is represented as a discrete coordinate chain (as shown in Fig. 3b), we can use the following equation to compute the curvature:

$$K(s) = \text{atan} \left(\frac{y_s - y_{s-w}}{x_s - x_{s-w}} \right) - \text{atan} \left(\frac{y_{s-1} - y_{s-1-w}}{x_{s-1} - x_{s-1-w}} \right), \quad (9)$$

where $w > 1$ is used to reduce the effect of contour noise in computing the differentiation. Also note that $y_t = y_{t+N}$ and $x_t = x_{t+N}$, because the boundary representation is a closed chain.

The centroid distance function is the second contour representation which will be used in extracting the shape features. It is defined to be the distance of boundary pixels from the centroid (x_c, y_c) of the object (see Fig. 4):

$$R(s) = \sqrt{(x_s - x_c)^2 + (y_s - y_c)^2}. \quad (10)$$

The third contour representation is the complex coordinate function, which can be obtained by simply representing the coordinates of the boundary pixels as complex numbers:

$$Z(s) = (x_s - x_c) + j(y_s - y_c), \quad (11)$$

where $j = \sqrt{-1}$. Note that an object-centered coordinate system is used here to make the representation translation invariant.

3.1.2 Fourier-based shape description

In the area of shape analysis and classification, several shape feature representation schemes based on autoregressive (AR)

models [7, 25] and Fourier descriptors [1, 21, 35] of contours have been proposed. Recently, an experimental comparison of shape classification methods based on these two principles has been carried out in [12], which indicates that Fourier-based methods provide better performance than AR-based approaches, especially for noisy images. For this reason, we use the Fourier-based shape descriptions in our image retrieval system.

In order to ensure that the resulting shape features of all image objects in the database have the same length, the boundary function $((x_s, y_s), 0 \leq s \leq N - 1)$ of each object is re-sampled to M samples before performing the Fourier transform. In our experiments, we choose $M = 2^6 = 64$ so that the transformation can be conducted efficiently using FFT.

Fourier transform of a contour representation generates a set of complex coefficients. These coefficients represent the shape of an object in the frequency domain, with lower frequency describing the general shape property and higher frequency denoting the shape details. The shape feature can be extracted from these transform coefficients. In order to achieve rotation invariance, we only use the amplitude information of the coefficients and discard the phase component. This allows the encoding of the contour to begin at any point along the contour. Scale invariance is achieved by dividing the amplitude of the coefficients by the amplitude of the DC component or the first non-zero frequency coefficient [12]. Note that translation invariance is obtained directly from the contour representation.

Three Fourier-based shape feature representations are now computed. For curvature and centroid distance functions, we only need to consider the positive frequency axes, because these functions are real and, therefore, their Fourier transforms exhibit symmetry, i.e., $|F_{-i}| = |F_i|$. The shape feature for the curvature is

$$\vec{f}_K = [|F_1|, |F_2|, \dots, |F_{M/2}|], \quad (12)$$

where F_i denotes the i th component of Fourier transform coefficients. Similarly, The shape feature for the centroid distance is

$$\vec{f}_R = \left[\frac{|F_1|}{|F_0|}, \frac{|F_2|}{|F_0|}, \dots, \frac{|F_{M/2}|}{|F_0|} \right]. \quad (13)$$

For a complex coordinate function, we use both negative and positive frequency components. The DC coefficient is dependent on the position of shape and, therefore, is discarded. The first non-zero frequency component is used to normalize the other transform coefficients. The shape feature for complex coordinate representation is

$$\vec{f}_Z = \left[\frac{|F_{-(M/2-1)}|}{|F_1|}, \dots, \frac{|F_{-1}|}{|F_1|}, \frac{|F_2|}{|F_1|}, \dots, \frac{|F_{M/2}|}{|F_1|} \right]. \quad (14)$$

In the prototype system, any of the three features from Eqs. (12, 13, and 14) can be specified by the user for computing shape similarity. A Euclidean metric is used to compute the distance between two shape feature vectors.

3.2 Texture features

In [17], we presented a texture feature representation scheme based on a Gabor decomposition. A comprehensive evalua-

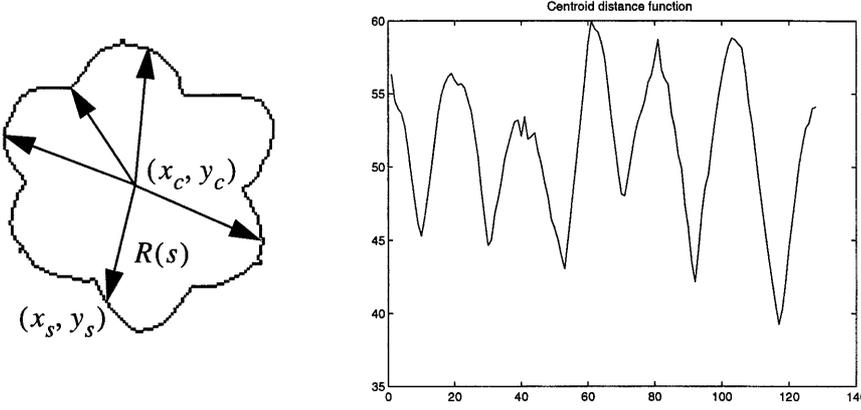
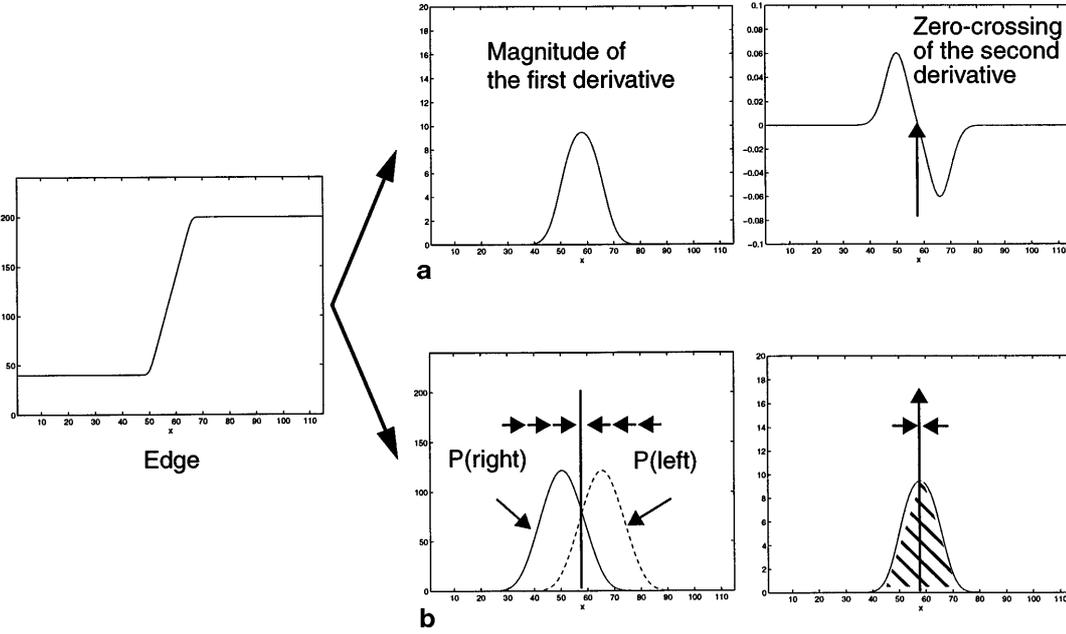


Fig. 4. An example of the centroid distance function

Fig. 5a,b. A comparison of the edge flow model with the conventional approach to detecting edges. **a** Traditional method of edge detection. **b** Edge flow model

tion and comparison with other multiresolution 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 [19] is given below. First consider a prototype Gabor filter:

$$h(x, y) = \left(\frac{1}{2\pi\sigma_x\sigma_y} \right) \exp \left[-\frac{1}{2} \left(\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2} \right) \right] \cdot \exp[2\pi j W x]. \quad (15)$$

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

$$h_{i,j}(x, y) = a^{-i} h(x', y'), \quad i, j = \text{integer} \quad (16)$$

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

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

$$O_{i,j}(x, y) = \int I(x, y) h_{i,j}^*(x - x_1, y - y_1) dx_1 dy_1, \quad (17)$$

where $*$ indicates the complex conjugate. A simple texture feature representation can be constructed using the mean and standard deviation of the amplitude information:

$$\mu_{ij} = \int \int |O_{i,j}(x, y)| dx dy$$

$$\sigma_{ij} = \sqrt{\int \int \int (|O_{i,j}(x, y)| - \mu_{ij})^2 dx dy}, \quad (18)$$

$$\mathbf{f}_t = [\mu_{00} \sigma_{00} \mu_{01} \dots \mu_{(S-1)(K-1)} \sigma_{(S-1)(K-1)}]. \quad (19)$$

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. For more details on this representation, we refer the reader to [17].

4 Image segmentation and grouping

One of the distinguishing features of NeTra is the automated image segmentation algorithm. Images are segmented and local region features are computed and stored at data ingest time. Segmentation is based on an edge flow model that we recently developed [16]. The basic ideas of the edge flow algorithm are outlined here and for details we refer to [16].

The usefulness of the segmentation scheme lies in the fact that very little parameter tuning is needed. The only free parameters controlling segmentation that the user needs to provide are:

1. Image features to be used (gray/color, texture, or both).
2. The preferred scale to localize the desired image boundaries.
3. The approximate number of image regions for the region-merging algorithm.

Discontinuities in natural images can occur in texture, color, or both. A segmentation algorithm should consider these different image attributes together in computing a partition of the image. Towards this, a general framework for boundary detection called “edge flow” is proposed in [16]. This framework utilizes a predictive coding model to identify and integrate the direction of change in color, texture, and filtered phase discontinuities at each image location. From this, an edge flow vector which points to the closest image boundary is constructed. This edge flow is iteratively propagated to its neighbor if the edge flow of the corresponding neighbor points in a similar direction. The edge flow stops propagating if the corresponding neighbor has an opposite direction of edge flow. In this case, the two image locations have 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. Figure 5 provides a comparison of the edge flow model with the conventional approaches to detecting edges using a 1D edge as an example.

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. This whole process including image smoothing, feature extraction, and prediction for identifying the flow direction is designed in a way that it can be controlled by a single scale parameter.

After boundary detection, disjoint boundaries are connected to form closed contours, thus partitioning the image into a number of regions. This is followed by a region-merging algorithm. Region merging utilizes dissimilarity in color and texture of the neighboring regions, as well as the length of the original boundary (before boundary connection) between those regions. One of the stopping criteria for region merging is the user-provided preferred number of segments in the image. The user’s preference may not be

strictly enforced if it requires merging two largely dissimilar regions.

Figure 6 illustrates the various stages of the image segmentation algorithm. This algorithm has been applied to over 2,500 images from a Corel photo gallery. This is one of few instances where a segmentation algorithm has been demonstrated to give good results on a wide class of images. The system also provides an optional tool for the user to further merge some of the regions if necessary. However, no additional boundaries apart from the ones given by the algorithm are created and the user-assisted region merging can be performed extremely fast. Figure 7 shows some of the image segmentation results.

A note regarding performance evaluation: Since no ground truth is available for the color images, no quantitative performance evaluation can be provided at this time. However, our experiments with some of the synthetic texture mosaics have given results better than most of the algorithms that we are currently aware of in the segmentation literature. A visual inspection of the results indicate that the segmentation is of acceptable quality, particularly for applications such as image browsing.

5 Query processing

Summarizing the discussions so far, a given image is first segmented into a number of homogeneous regions. Each region is represented using color, texture, and shape attributes as described in Sects. 2 and 3. For color features, a color existence table is generated/updated for fast indexing.

The texture and shape features are represented using a data structure similar to the SS-tree [33]. To construct the tree data structure, a modified k-means clustering algorithm is used. The modification is mainly to balance the tree, so that browsing can be supported efficiently. The balancing of the tree is achieved by imposing a constraint on the minimum number of nodes in each cluster. If the clustering in any iteration results in clusters with fewer than the minimum, such clusters are deleted and their members are assigned to other nearby clusters. The cluster centers are then re-computed and used as the initial condition for the next iteration. The process is repeated until no cluster is smaller than the specified threshold or if the number of iterations exceed a given number.

Color, texture, and shape of each of the regions in the database are indexed separately. For a query consisting of more than one of these image features, the intersection of the results of search using individual features can be computed and then sorted based on a weighted similarity measure. The current implementation of NeTra uses an implicit ordering of the image features to prune the search space. The first feature that the user specifies is used to narrow down the space, within which a more detailed search is performed to similarity-order the retrieval results.

5.1 Spatial queries

In addition to the above-mentioned image features, NeTra allows users to specify spatial location to further disambiguate the retrievals. For example, consider a search for

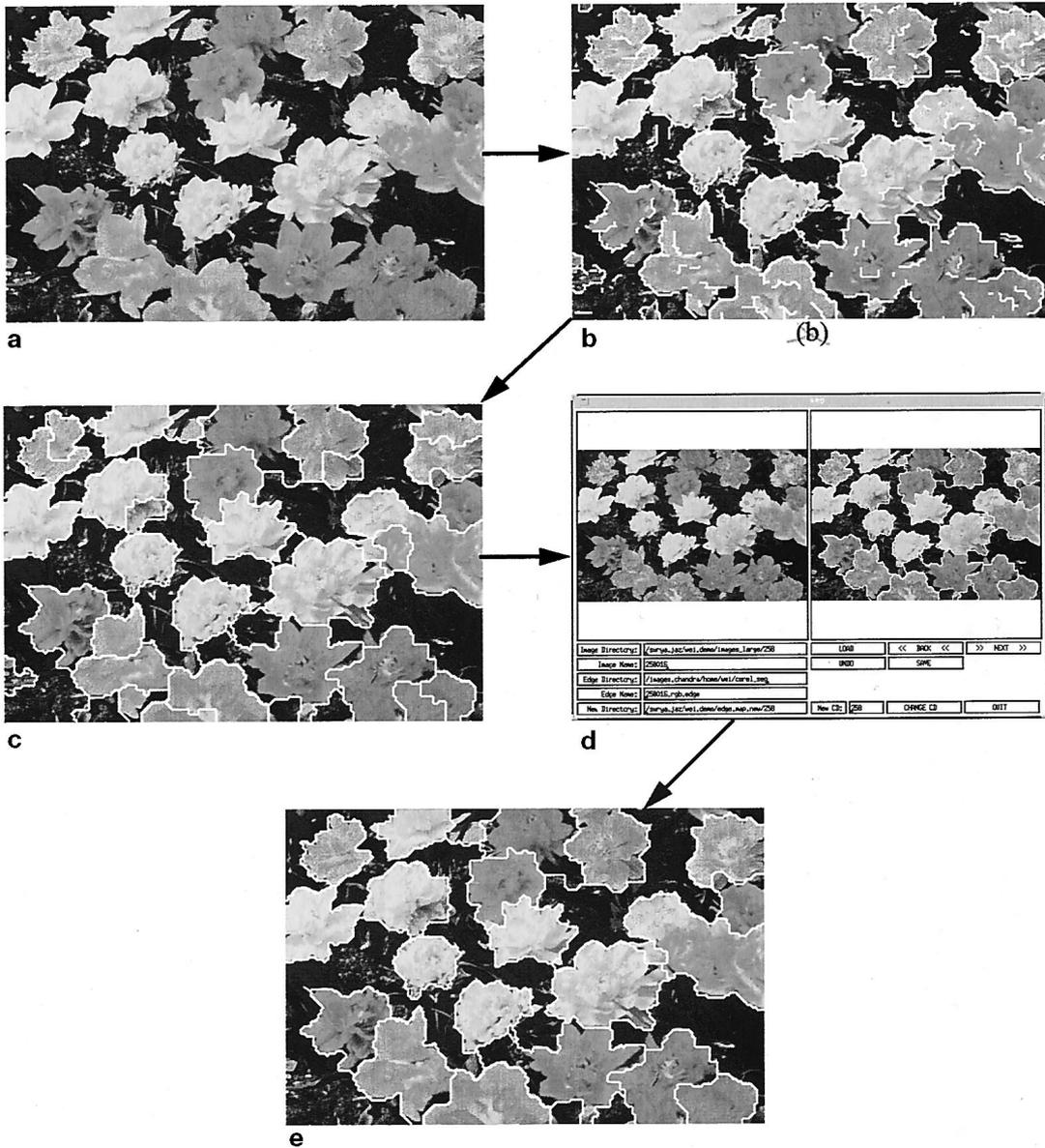


Fig. 6a–e. Various stages of the edge-flow based image segmentation. **a** Input image, **b** results of boundary detection based on edge flow, and **c** after boundary connection and region merging, **d** shows a tool (optional) where the user can further reduce the number of segments by deleting boundaries, and **e** is the final segmentation result

snow-covered-mountain pictures using color and texture. A spatially unconstrained search often finds regions of ocean surf, as such regions also have similar texture and color. A search constrained to look for solutions only in the upper half of the image would eliminate or reduce such retrieval instances.

In order to facilitate spatial constraints, NeTra organizes each region location information as part of its metadata. The spatial location of each image region in the database is represented by two sets of parameters: the region centroid (x_c, y_c) , and the coordinates of its minimum bounding rectangle (x_l, x_r, y_t, y_b) . The minimum bounding rectangle is the smallest vertically aligned box which contains the region completely as shown in Fig. 8.

There are several approaches to constructing a spatial query. One can directly specify the centroid of image re-

gion or use a rectangle window to find the image regions that overlap with it spatially. The quad-tree [24] has been widely used to provide a fast access to 2D data points and, therefore, can be employed to index the region centroids. The R-trees [9] can be utilized to efficiently search the image regions whose minimum bounding rectangles overlap with a specified area. However, in our experiments, we noticed that the range or area of image region is more intuitive and effective for forming a spatial query than the region centroid. For example, a blue sky is a common feature in many scenery pictures, and it is usually located in the upper half of image. In order to search such images, one might want to impose a spatial constraint on the blue area to enhance the retrieval performance. The region centroid might not be sufficient to convey a message for “upper half.”

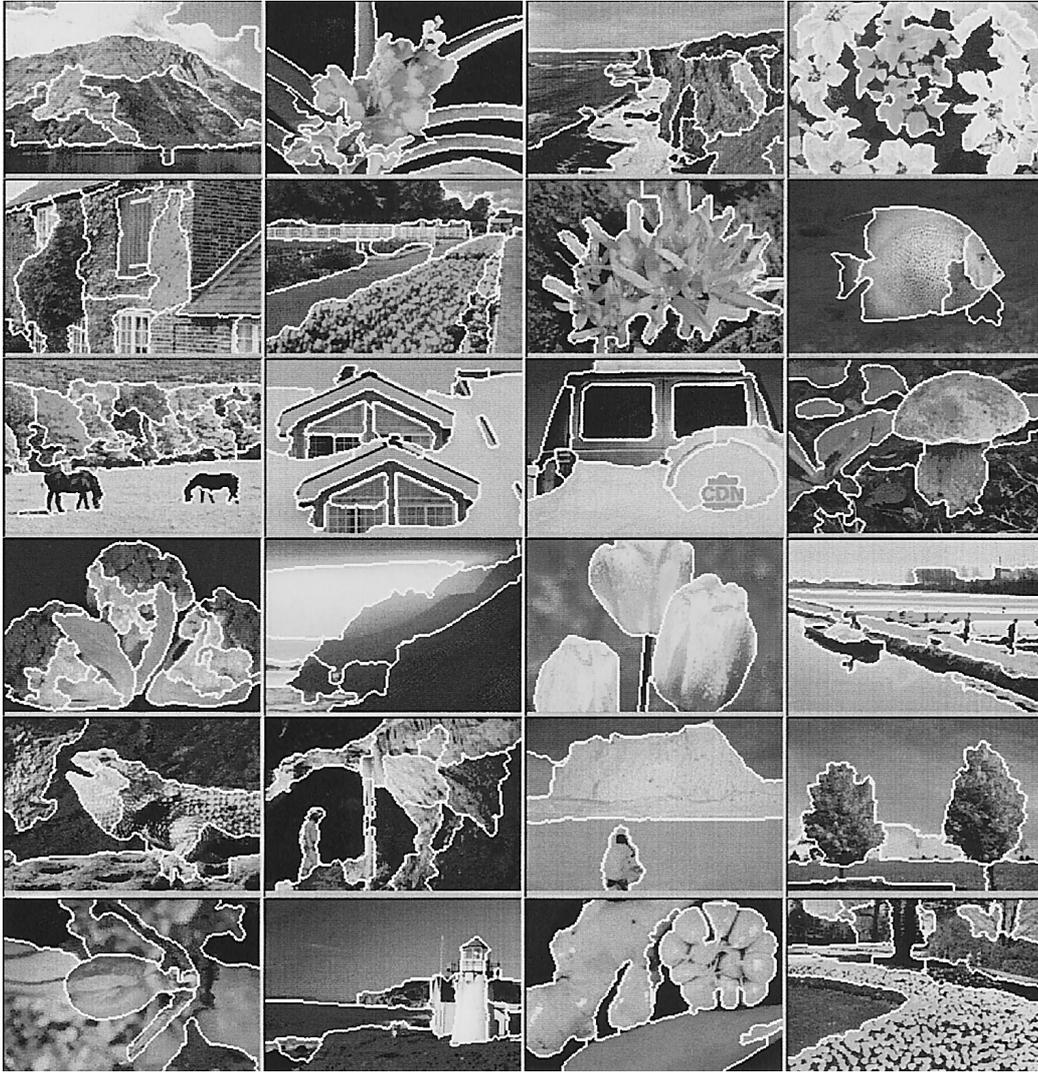


Fig. 7. Examples of the image segmentation results

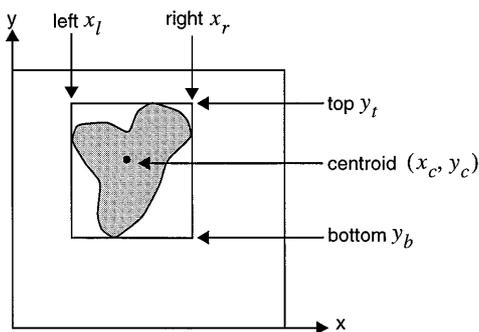


Fig. 8. The spatial location of image region is represented by its region centroid and its minimum bounding rectangle (x_l, x_r, y_t, y_b)

In order to provide an effective tool to query the spatial location, we propose the use of two query rectangle windows to define the area of interest for image search. The first window, the inner rectangle, is used to find the image regions whose bounding boxes have at least some overlap with it. The second window, the outer rectangle, is used to retain only those regions whose bounding boxes are com-

pletely contained inside of this rectangle. These inner and outer rectangles are represented by their corresponding coordinates $(x_l^{(i)}, x_r^{(i)}, y_t^{(i)}, y_b^{(i)})$ and $(x_l^{(o)}, x_r^{(o)}, y_t^{(o)}, y_b^{(o)})$, respectively (see Fig. 9).

The query for spatial location is performed using four image region lists which are sorted according to the top left and bottom right coordinates of their minimum bounding rectangles, and four additional tables which provide indices into these image region lists. Let us consider the list of bottom positions, $\{y_b\}$, denoted as L_b , and its associated index table T_b as an example. L_b contains the index to all the image regions in the database with their y_b sorted, and thus its length is equal to the total number of image regions N . T_b then stores the pointers to the list L_b so that it knows which portion of the list should be accessed if the range of y_b is specified. Note that the length of T_b is equal to the image height.

Given the positions of two query rectangles, the top of inner rectangle $y_t^{(i)}$ will be used to eliminate those image regions which have $y_b > y_t^{(i)}$. This can be done by first looking at the pointer $T_b(y_t^{(i)})$, then deleting the upper portion of the

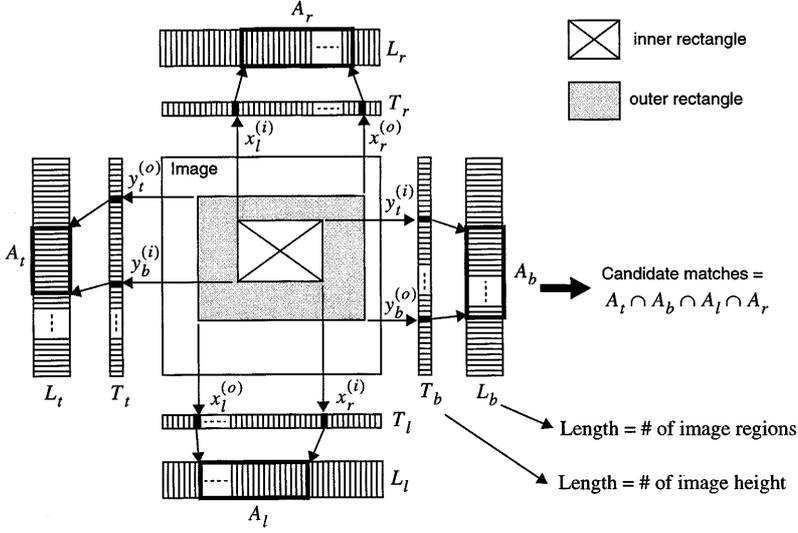


Fig. 9. Indexing scheme for query spatial location. Two rectangle windows are used to define the area of interest. Four sets of sorted image region lists and the corresponding index tables based on each side of the region bounding rectangle (top, bottom, left, and right) are used to perform a quick search

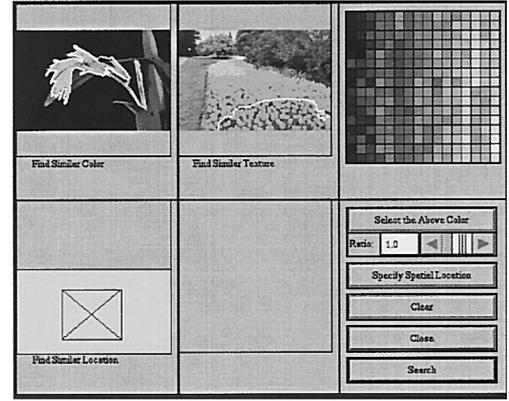
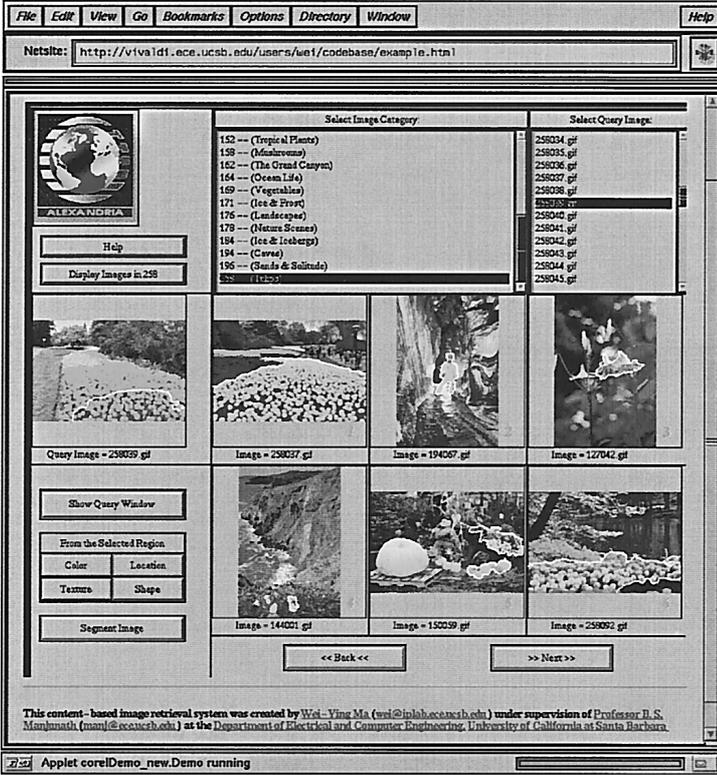


Fig. 10. **a** Window for browsing the images in the database and displaying the search results, **b** tool for constructing the queries

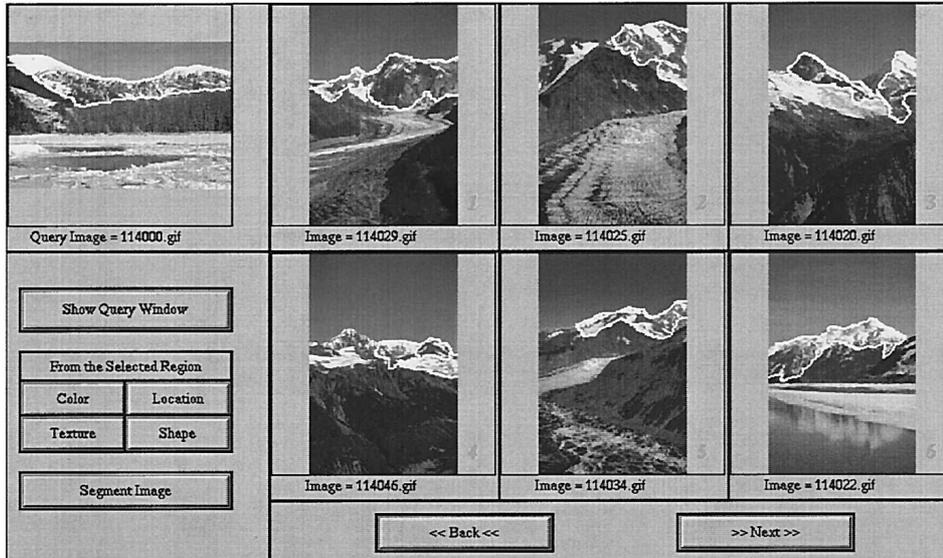
list L_b . On the other hand, the bottom of outer rectangle $y_b^{(o)}$ will be used to delete those regions which have $y_b < y_b^{(o)}$. This is done by using the pointer $T_b(y_b^{(o)})$ to remove the lower portion of the list L_b . Therefore, the resulting set of candidate matches is

$$A_b = L_b \left(T_b(y_t^{(i)}) : T_b(y_b^{(o)}) \right). \quad (20)$$

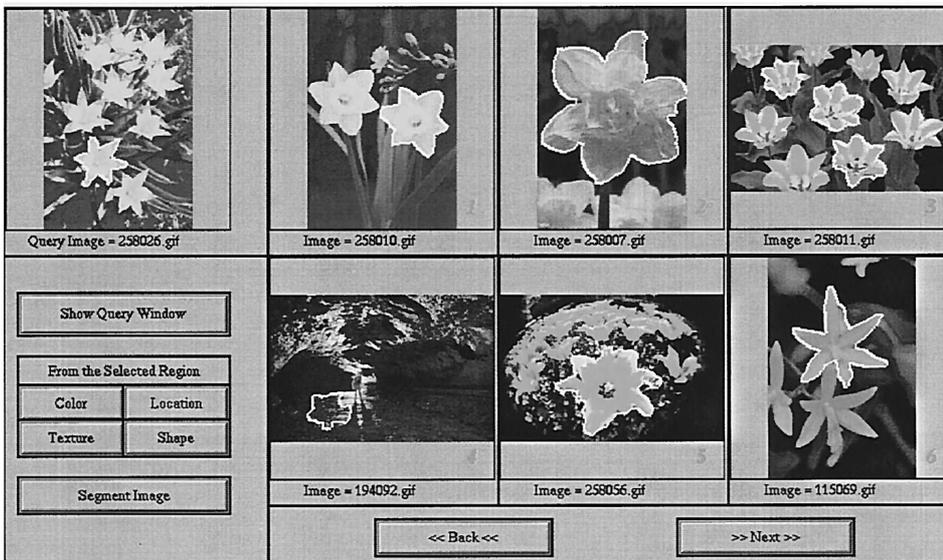
Similar operations are performed on the other three sides using the corresponding lists and tables, and thus computing three sets of candidate image regions, which are

$$\begin{aligned} A_t &= L_t \left(T_t(y_t^{(o)}) : T_t(y_b^{(i)}) \right), \\ A_l &= L_l \left(T_l(x_l^{(o)}) : T_l(x_r^{(i)}) \right), \\ A_r &= L_r \left(T_r(x_l^{(i)}) : T_r(x_r^{(o)}) \right) \end{aligned} \quad (21)$$

where L_t , L_l , and L_r are the lists for the top, left, and right position of region-bounding rectangle, and T_t , T_l , and T_r are their corresponding index tables. The final candidate matches should satisfy all the previous requirements and, therefore, are the result of $A_t \cap A_b \cap A_l \cap A_r$. Figure 9 illustrates this search mechanism. Based on a similar strategy, the region



a



b

Fig. 11a,b. Image retrieval using color and shape features. The image in the upper left window is the query image, and the selected query region is outlined. The top 6 best matched image regions are displayed on the right. **a** uses the color feature, and **b** is based on the shape attribute

centroid can also be integrated into the search process to eliminate image regions whose centroids fall outside of the inner rectangle.

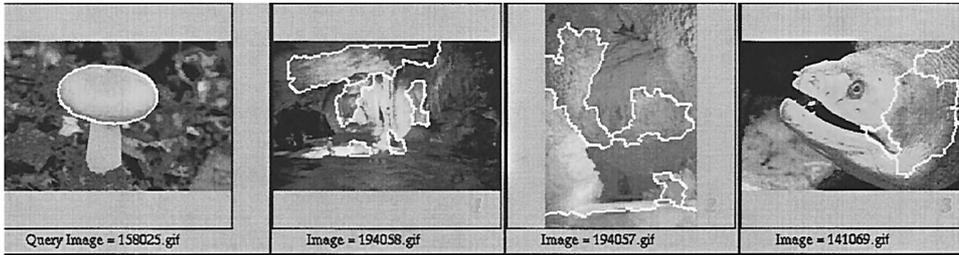
6 Experimental results

The current implementation of NeTra utilizes color, texture, shape, and location information for region search and retrieval. The system is developed in JAVA language to provide client platform independence and accessibility on the web. Currently, our database contains 2,500 color images from a Corel photo gallery. The collection is organized into 25 different categories with 100 images per category.

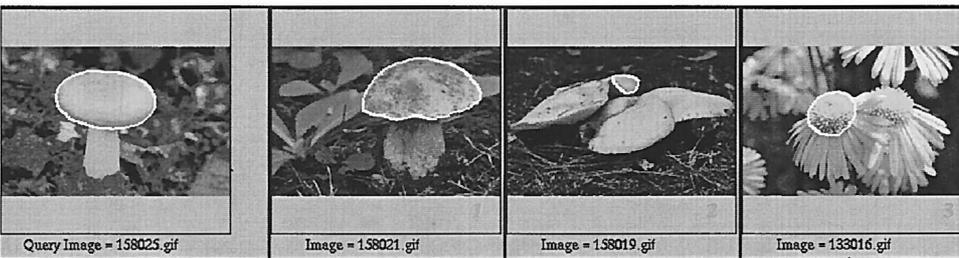
Image segmentation and region feature extraction are done off-line. The parameters for controlling the segmentation algorithm are specified on a category-by-category base. For those images with uniform color and no texture, we specify the type of processing to “color only” and use a smaller

scale parameter 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 use a larger scale parameter. 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 local region color, texture, and shape feature extraction. Regions which contain more than 30% of image borders are not considered for shape-based retrievals. The resulting region features and their spatial locations are stored as part of the image metadata.

Orange colored flower beds. A snapshot of the user interface of NeTra is shown in Fig. 10. The users can choose color, texture, shape, spatial location, or any combination of them to form their queries in the query window as shown



a



b

Fig. 12. **a** Image retrieval based on the color information only. **b** Combining with shape information improves retrieval performance

in Fig. 10b. In this example, we illustrate the case when a user is interested in finding images containing orange-colored flower beds. An example of a large orange flower is used to specify the color of interest. Since a flower bed has texture associated with it, the user further uses an example of such a “flowers” pattern to specify the texture of interest. In addition, these orange flowers are more likely to be located in the lower portion of an image and, therefore, the spatial location is also included to favor those regions. Figure 10a shows the search results based on these three requirements. This example demonstrates how one can use low-level image attributes such as color, texture, shape, and spatial location to form a complete query so that image regions of interest can be retrieved.

Snow-covered peaks. Figure 11a shows an example which uses the color to retrieve several similar snow covered peaks, and Fig. 11b shows an example of using the shape information for image retrieval. Integrating shape and color enhances retrieval performance as shown in Fig. 12.

Blue sky. Figure 13 shows another example where the user is interested in retrieving images of blue sky. The user chooses the color table (256 colors from the codebook) to specify the blue colors and includes the spatial location to help the search. The top 24 retrievals are shown. As can be seen, except for matches #20 and #23, the other retrieved images all contain blue sky.

Figure 14 provides two more retrieval examples based on the joint color and texture similarity. As can be seen from these examples, the system successfully retrieves images with perceptually similar regions. These and other examples shown emphasize the need for segmentation to index image regions and not whole images. It should be noted that global image attributes would have mixed background information with desired local region attributes. Image segmentation distinctly helps in developing a better representation for region-based search and indexing, and significantly improves retrieval results compared to using whole-image attributes.

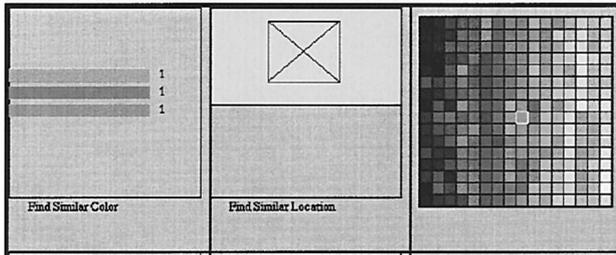
7 Discussion

We have described an implementation of NeTra, a toolbox for organizing and searching image regions based on local image properties. The system includes a robust image segmentation scheme, and color, texture, and shape features representing region information. The edge-flow-based segmentation is used to process a large and diverse collection of images with very little parameter tuning. With the capability of analyzing and representing individual image regions, the image retrieval performance improves dramatically. However, much work remains to be done in evaluating and quantifying performance of such image retrieval systems.

NeTra uses a compact color feature representation appropriate for segmented regions. In contrast with traditional color histogram methods which use a fixed number of color bins to characterize the color information, our approach sequentially increases the number of colors to cluster the colors in the region until the mean squared error of the clustering is below a pre-defined threshold. Since segmented regions are quite homogeneous in color and/or texture, much fewer colors, typically 5–15, are usually sufficient for representing a region color.

An efficient color-indexing scheme based on the compact color representation is also proposed. This indexing scheme utilizes dominant colors in the query image to prune the search space. This initial search involves only boolean operations such as AND and OR, and thus can be efficiently implemented. Color quantization and similarity computations are currently performed in the RGB space and, as the results indicate, do provide visually acceptable retrievals. In color vision research it is shown that color spaces such as the CIE $L^*a^*b^*$, CIE L^*u^*v and Munsell color space [34] correspond better to human color perception. However, Euclidean distance metric for distance computations may not be appropriate in these spaces and new quantization schemes need to be developed. These issues are being investigated.

In addition to color, NeTra uses texture and shape of the segmented regions in indexing them in the database.



a



b

Fig. 13a,b. Image search for “blue sky.” **a** The query formed by the user (colors/percentage) **b** The top 24 retrievals from the database. Except the match 20 and 23, the other retrieved images all contain blue sky

A fast algorithm for spatial queries which use the region bounding box information is also developed. Some typical retrieval results are provided, which illustrate that region-based representation and querying images using local image properties significantly enhance the perceptual quality of the retrieved results.

Currently we are expanding the size of the database (to about 12,500 images), and developing a search tool which allows the user to form a query using multiple image regions which include spatial relationships. Modifying the retrieval results based on user feedback is another interesting direction that is being explored. This problem of incorporating

relevance feedback from users has recently become an important and active research area [6, 23].

Appendix Color codebook construction using vector quantization

Let us denote this color codebook as C , where $C = \{C_1, C_2, \dots, C_{256}\}$ and each color $C_i = (r_i, g_i, b_i)$ is a 3D RGB color vector. The 2,500 color images in our database (from the Corel photo CDs) are used as the training data. The GLA basically contains the following steps.

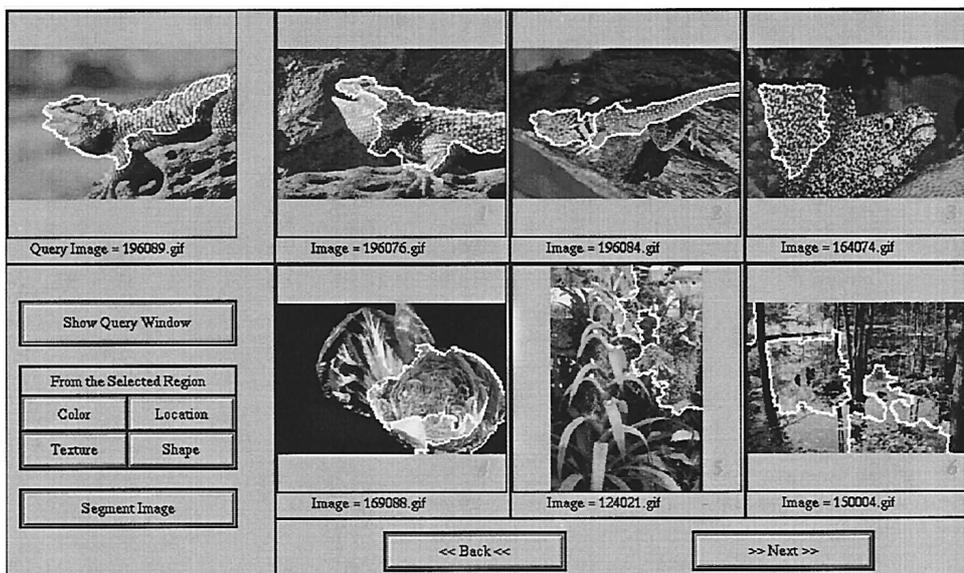
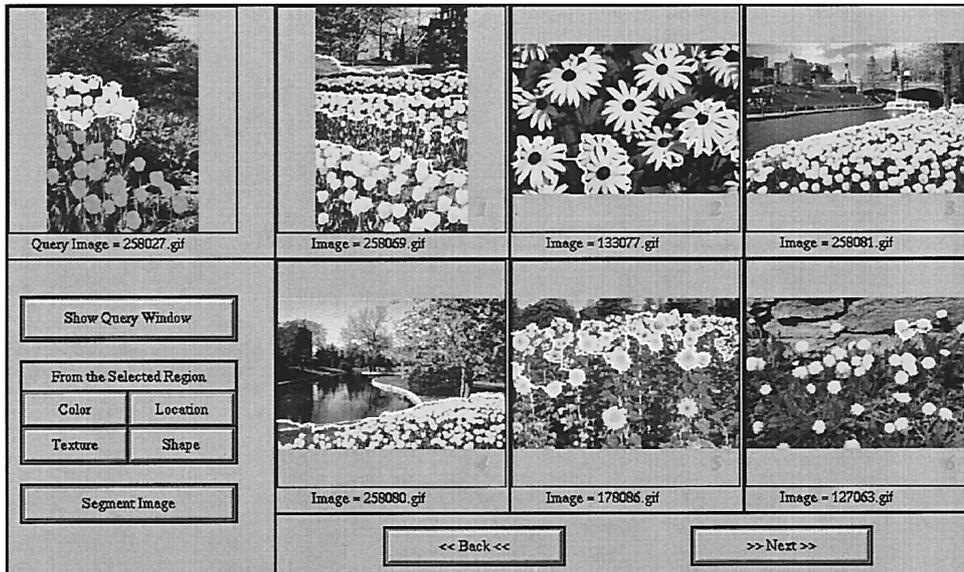


Fig. 14. Examples of region-based image retrieval using the joint color and texture information. Both of the query regions and the best matched regions are outlined

1. Begin with an initial codebook C_1 . Set iteration number $m = 1$.
2. Given a codebook, $C_m = \{C_i\}$, find the optimal partition into quantization cells. That is, $S_i = \{X | d(X, C_i) \leq d(X, C_j); \text{ all } j \neq i\}$, where S_i is a collection of colors belonging to C_i .
3. Use the centroid of each S_i to form the new codebook C_{m+1} .
4. Compute the average distortion for 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.

Acknowledgements. This research is supported in part by the Alexandria Digital Library project at the University of California, Santa Barbara under NSF grant number IRI-94-11330. We thank Yining Deng for his help in the design and implementation of color image features.

References

1. Arbter K, Snyder WE, Burkhardt H, Hirzinger G (1990) Application of affine-invariant Fourier descriptors to recognition of 3D objects. *IEEE Trans Pattern Anal Machine Intell* 12: 640–647
2. Ashley J, Barber R, Flickner MD, Hafner JL, Lee D, Niblack W, Petkovic D (1995) Automatic and semiautomatic methods for image annotation and retrieval in QBIC. *Proc SPIE (Storage and Retrieval for Image and Video Databases III)* 2420: 24–35
3. Bach JR, Fuller C, Gupta A, Hampapur A, Horowitz B, Humphrey R, Jain RC, Shu C (1996) Virage image search engine: an open framework for image management. *Proc SPIE (Storage and Retrieval for Image and Video Databases IV)* 2670: 76–87
4. Carson C, Ogle VE (1996) Storage and retrieval of feature data for a very large online image collection. *IEEE Comput Soc Bull Techn Comm Data Eng* 19(4)
5. Carson C, Belongie S, Greenspan H, Malik J (1997) Region-based Image Querying. In: *IEEE Workshop on Content-Based Access of Image and Video Libraries*, June 1997, Puerto Rico, San Juan, pp.42–49
6. Delanoy RL (1995) Toolkit for image mining: user-trainable search tools. *Lincoln Lab J* 8(2): 145–160

7. Dubois SR, Glanz FH (1986) An autoregressive model approach to two-dimensional shape classification. *IEEE Trans Pattern Anal Mach Intell* 8: 55–66
8. Gersho A, Gray RM (1992) *Vector Quantization and Signal Compression*. Kluwer Academic, Dordrecht
9. Guttman A (1984) R-trees: a dynamic index structure for spatial searching. *ACM Proc. Int. Conf. Manag. Data*, June 1984, pp 47–57
10. Hafner J, et al. (1995) Efficient color histogram indexing for quadratic form distance functions. *IEEE Trans Pattern Anal Mach Intell* 17(7): 729–736
11. Huang Q et al. (1995) Foreground/background segmentation of color images by integration of multiple cues. In: *IEEE Int. Conf. on Image Processing*, Vol. 1, October 1995, Washington, DC, pp 246–249
12. Kauppinen H, Seppänen T, Pietikäinen M (1995) An experimental comparison of autoregressive and Fourier-based descriptors in 2D shape classification. *IEEE Trans Pattern Anal Mach Intell* 17(2): 201–207
13. Niblack W, Barber R, Equitz W, Flickner M, et al. (1993) The QBIC project: querying images by content using color, texture, and shape. *Proc SPIE (Storage and Retrieval for Image and Video Databases)* 1908: 173–187
14. Ma WY, Manjunath BS (1996) A texture thesaurus for browsing large aerial photographs. *Journal of the American Society for Information Science*, vol. 49, no. 7, 1998, pp 633–648
15. Ma WY, Manjunath BS (1996) Texture features and learning similarity. In: *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, June 1996, San Francisco, Calif., pp 425–430
16. Ma WY, Manjunath BS (1997) Edge flow: a framework of boundary detection and image segmentation. *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, pp 744–749, Puerto Rico, June 1997
17. Manjunath BS, Ma WY (1996) Texture features for browsing and retrieval of image data. *IEEE Trans Pattern Anal Mach Intell* 18(8): 837–842
18. Manjunath BS, Ma WY (1996) Browsing large satellite and aerial photographs. *IEEE Int. Conf. on Image Processing*, Vol. 2, September 1996, Lausanne, Switzerland, pp 765–768
19. Minka TP, Picard RW (1995) Interactive learning using a society of models. Technical Report No. 349, MIT Media Laboratory, Cambridge, Mass.
20. Pentland A, Picard RW, Sclaroff S (1994) Photobook: tools for content based manipulation of image databases. *Proc. SPIE (Storage and Retrieval for Image and Video Databases II)* 2185: 34–47
21. Persoon E, Fu K (1977) Shape discrimination using Fourier descriptors. *IEEE Trans Syst Man Cybern* 7: 170–179
22. Rubner Y, Guibas LJ, Tomasi C (1997) The earth mover's distance, multi-dimensional scaling, and color-based image retrieval. In: *Proc. of the APRA Image Understanding Workshop*, May 1997, pp 661–668
23. Rui Y, Huang TS, Mehrotra S, Ortega M (1997) A relevance feedback architecture in content-based multimedia information retrieval systems. In: *IEEE Workshop on Content-Based Access of Image and Video Libraries*, June 1997, Puerto Rico, pp 82–89
24. Samet H (1984) The quadtree and related hierarchical data structures. *ACM Comput Surv* 16(2): 187–260
25. Sekita I, Kurita T, Otsu N (1992) Complex autoregressive model for shape recognition. *IEEE Trans Pattern Anal Mach Intell* 14: 489–496
26. Slater D, Healey G (1996) The illumination-invariant recognition of 3D objects using local color invariants. *IEEE Trans Pattern Anal Mach Intell* 18(2): 206–210
27. Smith JR, Chang SF (1996) Local color and texture extraction and spatial query. In: *IEEE Int. Conf. on Image Processing*, Vol. 3, September 1996, Lausanne, Switzerland, pp 1011–1014
28. Smith JR, Chang SF (1996) VisualSEEK: a fully automated content-based image query system. *ACM Multimedia*, November 1996, Boston, Mass.
29. Smith TR (1996) A digital library for geographically referenced materials. *IEEE Computer Society Press*, pp 54–60
30. Swain MJ, Ballard DH (1991) Color indexing. *Int J Comput Vision* 7(1): 11–32
31. Stricker M, Orengo M (1995) Similarity of color images. *Proc SPIE (Conf. on Storage and Retrieval for Image and Video Databases III)* 2420: 381–392
32. Van Otterloo PJ (1991) *A Contour-Oriented Approach to Shape Analysis*. Prentice Hall, Englewood Cliffs, N.J.
33. White DA, Jain R (1996) Similarity indexing with the SS-tree. In: *Proc. 12th IEEE Int. Conf. on Data Engineering*, February 1996, New Orleans, La., pp 516–523
34. Wyszecki G, Stiles WS (1982) *Color Science*. John Wiley & Sons, New York
35. Zahn CT, Roskies RZ (1972) Fourier descriptors for plane closed curves. *IEEE Trans Comput* 21(3): 269–281



WEI-YING MA received the B.S. degree in electrical engineering from the national Tsing-Hua University in Taiwan in 1990, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of California at Santa Barbara (UCSB) in 1994 and 1997, respectively. From 1994 to 1997 he was engaged in the Alexandria Digital Library project in UCSB while completing his Ph.D. In June 1997, he joined the Hewlett-Packard Laboratories at Palo Alto, where he is currently a Staff Engineer in the Internet Systems and Applications lab. His research interests include content-based image/video retrieval, image processing, computer vision, and neural networks.



MANJUNATH BS received the B.E. in Electronics (with distinction) from the Bangalore University in 1985, M.E. (with distinction) in Systems Science and Automation from the Indian Institute of Science in 1987, and the Ph.D. degree in Electrical Engineering from the University of Southern California in 1991. He joined the ECE department at UCSB in 1991, where he is now an Associate Professor. During the summer of 1990, he worked at the IBM T.J. Watson Research Center at Yorktown Heights, NY. Dr. Manjunath was a recipient of the national merit scholarship (1978–85) and was awarded the university gold medal

for the best graduating student in electronics engineering in 1985 from Bangalore University. He has served on the program committees of many international conferences and workshops and was on the organizing committee of the 1997 International Conference on Image Processing (ICIP'97). His current research interests include computer vision, learning algorithms, image/video databases and digital libraries. He is currently an Associate Editor of the *IEEE Transactions on Image Processing* and is a guest editor of a special issue on image and video processing for digital libraries to be published in the *IEEE Image Processing Transactions* in 1999.