

AUTOMATIC ANNOTATION AND RETRIEVAL OF IMAGES

Yuqing Song

Department of Computer and Information Science

The University of Michigan at Dearborn

Dearborn, MI 48128 USA

yqsong@engin.umd.umich.edu

Wei Wang and Aidong Zhang

Department of Computer Science and Engineering

State University of New York at Buffalo

Buffalo, NY 14260 USA

wwang3, azhang@cse.buffalo.edu

Abstract Although a variety of techniques have been developed for content-based image retrieval (CBIR), automatic image retrieval by semantics still remains a challenging problem. We propose a novel approach for semantics-based image annotation and retrieval. Our approach is based on the monotonic tree model. The branches of the monotonic tree of an image, termed as *structural elements*, are classified and clustered based on their low level features such as color, spatial location, coarseness, and shape. Each cluster corresponds to some semantic feature. The category keywords indicating the semantic features are automatically annotated to the images. Based on the semantic features extracted from images, high-level (semantics-based) querying and browsing of images can be achieved. We apply our scheme to analyze scenery features. Experiments show that semantic features, such as sky, building, trees, water wave, placid water, and ground, can be effectively retrieved and located in images.

Keywords: Content-based image retrieval, semantics, monotonic tree, image annotation

1. Introduction

With the advance of multimedia technology and growth of image and video collections, content-based multimedia (image/video) retrieval has been an active research area in the last ten years. Image retrieval can be based on low-level visual features (such as color [Swain and Ballard, 1991; Smith and Chang, 1996; Pass et al., 1996], texture [Manjunath and Ma, 1996; Smith and Chang, 1994; Sheikholeslami and Zhang, 1997], and shape [Shahabi and Safar, 1999; Tao and Grosky, 1999; Safar et al., 2000]), high-level semantics [Forsyth et al., 1996; Torralba and Oliva, 1999], or both [Zhao and Grosky, 2001]. Retrieving images via low-level features has proven unsatisfactory. Automatic image retrieval by semantics still remains a challenging problem due to the difficulty in object recognition and image understanding. It is an urgent need to build image retrieval systems which support high-level (semantics-based) querying and browsing of images. Keyword indexing is a common scheme used by many picture libraries. For example, Getty Images [Bjarnestam, 1998] used over 10,000 keywords to index their collection of contemporary stock photographs. Current image indexing by keywords can only be done manually. According to [Eakins and Graham, 1999], the process of manual indexing suffers from two significant drawbacks. Firstly, it is inherently very labor-intensive. Secondly, manual indexing does not appear to be particularly reliable as a means of subject retrieval of images. In recognizing the existing problems in the CBIR field, we believe that research efforts are needed to bridge the gap between the high-level semantics users are interested in and the low-level features that can be extracted.

The semantics of images can be classified into two levels:

- *Local semantic level*: Local semantic features describe the presence of individual objects in images. Two examples of queries by local semantic features are “find pictures with a bridge” (object of a given type) and “find pictures with sky and trees” (combination of objects).
- *Thematic level (or global semantic level)*: Thematic features describe the global meanings or topics of images. Two examples of queries at this level are “find pictures of a Chinese garden” and “find pictures of an earthquake.”

The thematic features of an image are based on all objects in the image along with their spatial relationships. High-level reasoning is needed to derive the global meaning of all objects in the scene and to determine the topic of the image.

The difficulty of detection and recognition of general objects presents a significant challenge to the development of a CBIR system which can

extract general semantic features from images. However, if we restrain ourselves to specific domains, we can have a quick success. Among all objects in a natural image, it's quite common that a foreground object (or a group of foreground objects) indicates the theme of the image, while the background objects provide contextual and complementary information. Needless to say, successfully recognizing background objects in an image is a necessary step in determining the theme of the image. In addition, the contextual information provided by background objects helps and in many cases is necessary in recognizing foreground objects in two aspects:

- it can narrow down the possible types of foreground objects and make foreground object recognition more efficient; and
- it can be used to resolve the ambiguity in recognizing foreground objects.

Thus finding effective ways to recognize common background objects will be a milestone in solving the problem of image understanding. In outdoor pictures, scenery features such as sky, trees, ground, and water are common types of background objects.

In this paper, we will introduce a novel approach for semantics-based image annotation and retrieval. We choose scenery features as a testbed for our approach, which can be generalized to analyze other features, though. Our approach is based on the concept of monotonic tree [Song and Zhang, 2002], a derivation of contour tree for use with discrete data. Branches (subtrees) of the monotonic tree are termed as a *structural elements* if their areas are within a given scale. The structural elements are classified and clustered based on their low level features such as color, spatial location, harshness, and shape. Each cluster corresponds to some semantic feature. The category keywords indicating the semantic features are automatically annotated to the images. Based on the semantic features extracted from images, high-level (semantics-based) querying and browsing of images can be achieved.

The body of this paper is organized as follows. Section 2 introduces the concept of the monotonic tree. In Section 3, techniques are described for extracting semantics and annotating images. Section 4 presents case studies of scenery features, while Section 5 introduces semantics-based image querying. Section 6 offers a performance evaluation of the proposed approach. A summary and concluding remarks appear in Section 7.

2. Monotonic tree

Contour trees [Morse, 1969; Roubal and Poiker, 1985; van Kreveld et al., 1997] have been used in geographic information systems (GIS)

and medical imaging to display scalar data. For example, the elevation in the landscape can be modeled by scalar data over the plane, where a contour (also called an isoline) is a line along which the elevation function assumes the same value. Contours are only defined for continuous functions. For discrete data, a continuous function is first defined as an interpolation of the data. Then the contour tree is defined on this continuous function.

In computer imaging, the discreteness of image data is one main aspect which makes image processing and understanding so difficult. The discreteness of image data itself is a research topic. So we introduced the concept termed monotonic tree [Song and Zhang, 2002], which retrieves similar structures as the contour tree does while reserving the discreteness of image data. Monotonic tree is used as a hierarchical representation of image structures.

Corresponding to a positive/negative contour line [Morse, 1969], an *outward-falling/climbing monotonic line* of an gray image is a boundary where the image assumes higher/lower values in the pixels adjacent ¹ to the boundary from inside than those from outside. All monotonic lines in an image form a rooted tree, called *monotonic tree*. A maximal sequence of uniquely enclosing monotonic lines is called a *monotonic slope*. All monotonic slopes in an image form the *topological monotonic tree*. A monotonic slope is called *outward-falling/climbing* if all monotonic lines in it are outward-falling/climbing. See Figure 1. For a color image, the monotonic tree and topological monotonic tree are constructed on its gray copy.

For computation of monotonic trees, a top-down algorithm is available. The algorithm begins with the root monotonic line, browsing the private region of the root to find all child monotonic lines. The private region of a monotonic line is a region enclosed by the line and excluding the regions of the child monotonic lines. This process is repeated at each internal node to find its children. When all monotonic lines are found, they are stored in the monotonic tree. The algorithm is based on the properties of monotonic trees. For a detailed discussion of the algorithm and the properties of monotonic trees, readers may refer to [Song, 2002].

For a natural image of size 300×200 , the size (the number of monotonic lines) of its monotonic tree is around 10,000, mainly depending on the number of objects and their texture in the image. To demonstrate the monotonic trees of natural images, we will show their structures on several selected scales. Given a scale threshold t , a *maximal*

¹In [Song and Zhang, 2002], an image on the square grid is first transferred to the hexagonal grid, where monotonic lines and hence the monotonic tree are defined.

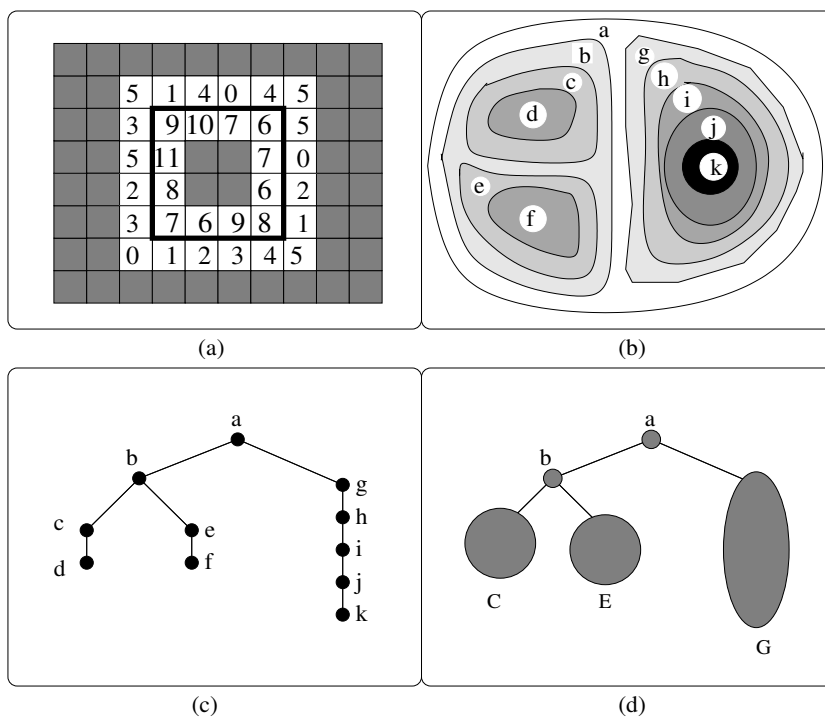


Figure 1. (a) An outward-falling monotonic line (the solid line in the figure), (b) a set of monotonic lines, (c) the monotonic tree, (d) the topological monotonic tree.

branch b of the monotonic tree at this scale is a branch such that (1) $Area(b) \leq t$ and (2) there is no other branch b' such that $Area(b') \leq t$ and $Region(b) \subset Region(b')$, where $Region(b)$ is the region in the image covered by branch b and $Area(b)$ is the area of $Region(b)$. At scale t , we decompose the image by removing all maximal branches at this scale. Figure 2 shows how to decompose an one-dimensional function at one scale, where a maximal branch at scale t is a maximal peak (or valley) such that the length of its base is no more than t . There are two types of maximal branches: upward and downward. Given a sequence of increasing scales, we can decompose the image scale by scale and get a multi-scale decomposition of the original image. See Figure 3.

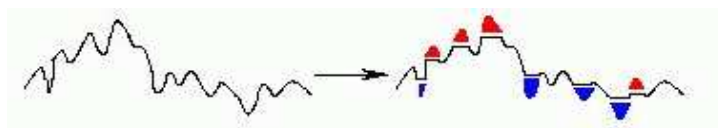
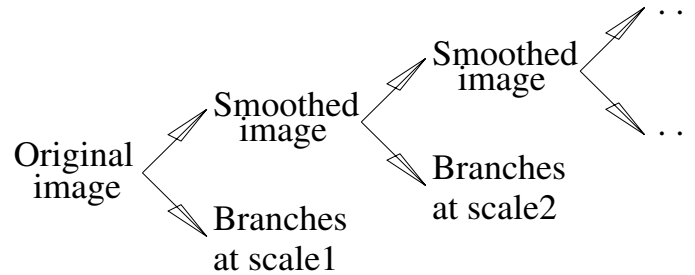


Figure 2. Decomposition of an one-dimensional function at one scale.



Given that: $\text{scale1} < \text{scale2} < \dots$

Figure 3. Multi-scale decomposition.

An example is shown in Figures 4-6. The original image is decomposed at two scales 50 and 150. The upward and downward maximal branches are shown in white and black, respectively. The second example is given in Figure 7, where the image is decomposed at three scales 10, 50, and 300. In this example, the noise part is mainly extracted in the maximal branches at scale 10, while the concentric characteristics of the texture is captured at scales 50 and 300.



Figure 4. An example image of size 554×362.

The third example is shown in Figure 8. In this example, the stars in the US flag form very regular texture. Each star appears in an upward maximal branch. The characteristic features (such as eyes, nose, mouth) in the face are also retrieved as maximal branches. From these examples we can see that the monotonic tree models the structures of an image at all scales. Semantics extraction and image annotation can be achieved



Figure 5. Decomposition of the image in Figure 4 at scale 50: (a) the maximal branches, (b) the smoothed image.

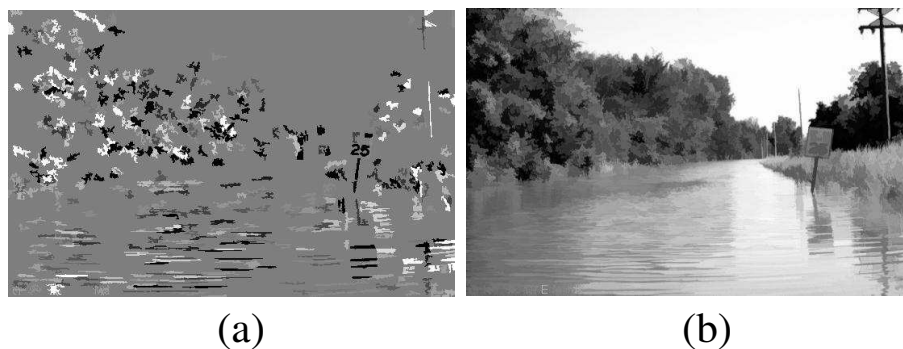


Figure 6. Decomposition of the image in Figure 4 at scale 150: (a) the maximal branches, (b) the smoothed image.

by analyzing the shape of these structures and their spatial relationship. A detailed discussion is given in the following sections.

3. Semantics Extraction and Image Annotation

Our feature extraction scheme is based on the topological monotonic tree. We use the branches of the topological monotonic tree to model the basic structures in an image, which are termed as *structural elements*. Structural elements have low-level features such as color, shape, harshness, and spatial location. They are clustered to form high-level features.

Feature extraction consists of four consecutive steps: (a) classifying structural elements; (b) clustering structural elements; (c) rendering semantic regions; and (d) annotating the input image. See Figure 9.

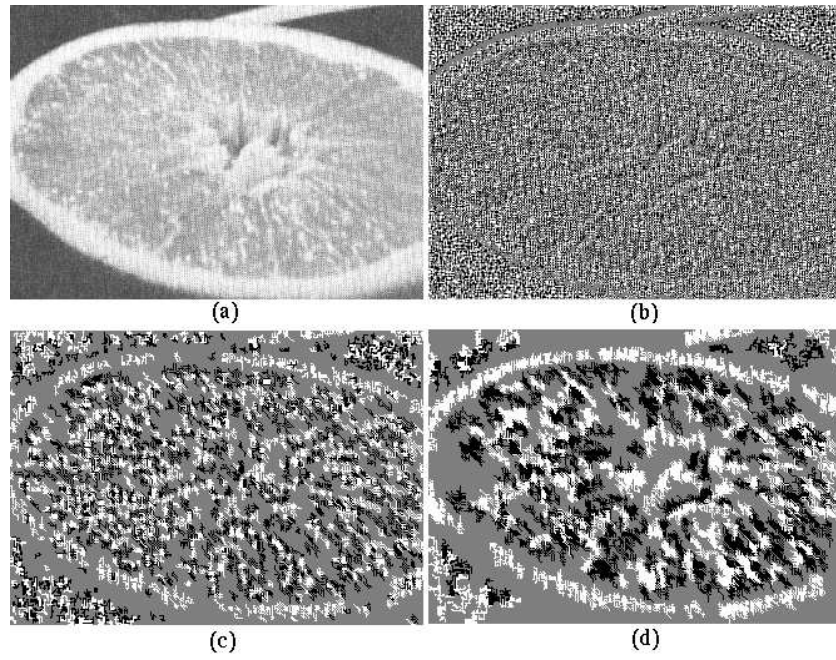


Figure 7. (a) Original image of size 330×236 , (b) maximal branches at scale 10, (c) maximal branches at scale 50, and (d) maximal branches at scale 300.

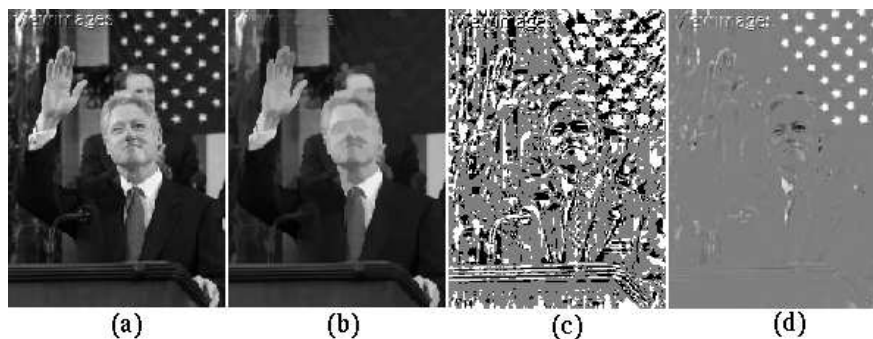


Figure 8. (a) Original image of size 148×204 , (b) the smoothed image at scale 80, (c) the maximal branches (in black-white) at scale 80, and (d) the maximal branches (in gray colors) at scale 80.

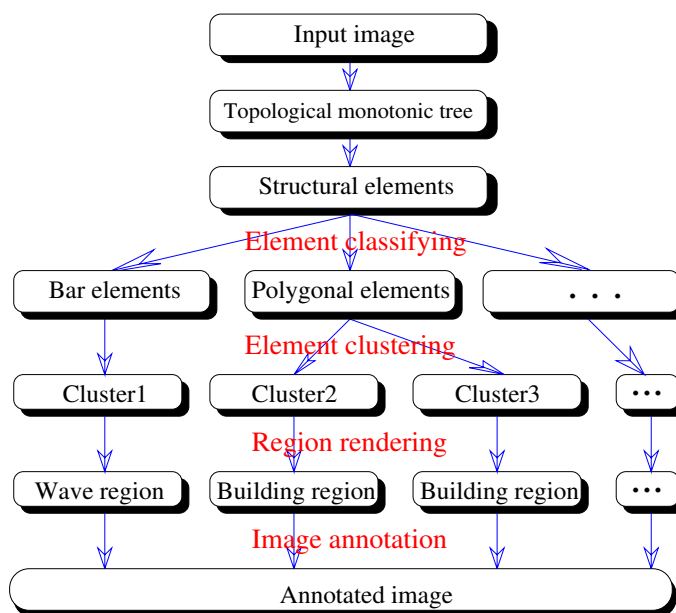


Figure 9. System design for feature extraction.

3.1. Classifying Structural Elements

Each branch (a subtree) of the topological monotonic tree is called a *structural element* if its covered area is no more than a threshold, which gives the scale in which we are interested. A structural element is called *positive/negative* if its root (root of the subtree) is outward-falling/climbing. See Figure 10(a). Positive/negative elements are like

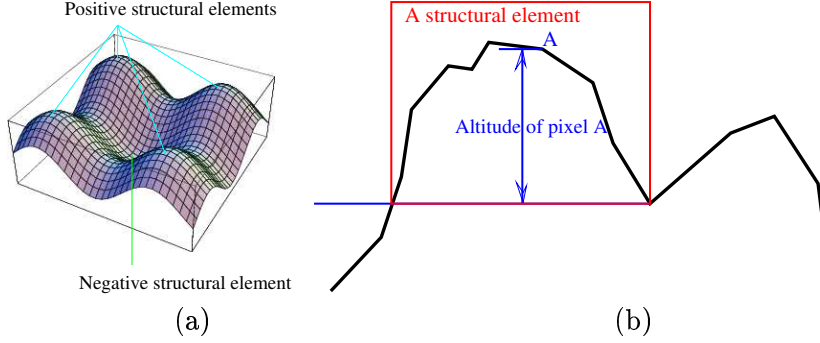


Figure 10. (a) Positive/negative structural elements; (b) the altitude of a pixel in a structural element.

peaks/valleys. For a positive/negative element, we define its *altitude* to be the absolute value of the average altitude of all its pixels above/below the highest/lowest pixels adjacent to the structural element. See Figure 10(b). The *harshness* of a structural element is determined by the number, area and altitude of its sub-elements.² We define the harshness of an element t by

$$Harshness(t) = \frac{\sum_{b \in SubElementSet(t)} Altitude(b) * Area(b)}{Area(t)},$$

where $SubElementSet(t)$ is the set of sub-elements of t , $Altitude(b)$ is the altitude of b and $Area(b)$ is the area of the region covered by b .

A structural element can be classified by its (1) color (the average color of all pixels in this element), (2) altitude, (3) harshness, and (4) shape (the shape of its covered region). By shape, we can classify elements as: (a) bars, (b) polygons, (c) irregular elements, (d) boundary-smooth elements, and (e) others. For a bar element, the ratio of its length to its width is high. A polygon element is a structural element whose boundary mainly consists of line segments. For a smooth-boundary element, its boundary is a smooth curve. The irregular elements are those whose boundaries are irregular. Figure 11 shows different cases of elements.

The semantic features of scenery images are characterized by the categories of structural elements. Three examples of the categories are polygon elements (for building), horizontal bar elements (for wave), and green harsh irregular elements (for tree). See Figures 12 and 13.

²An element is a branch of the topological monotonic tree. All sub-branches of this branch are sub-elements.

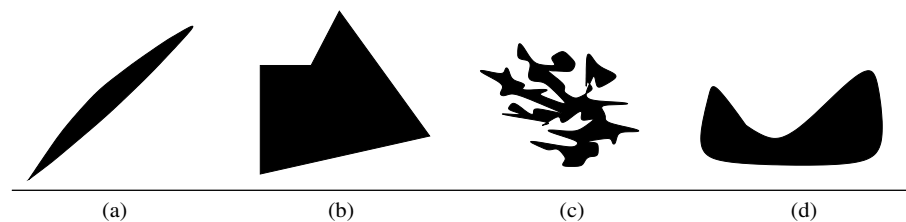


Figure 11. (a) A bar element, (b) a polygon element, (c) an irregular element, and (d) a smooth-boundary element.

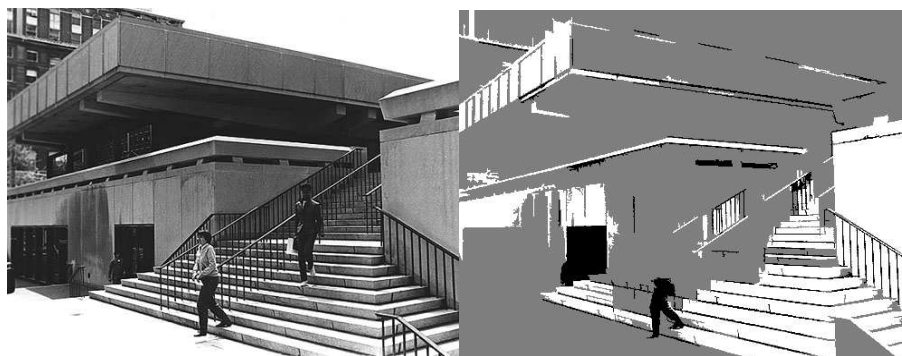


Figure 12. An image of building (in the left) and the polygon elements (in the right) in it. The polygon elements are shown in black and white.

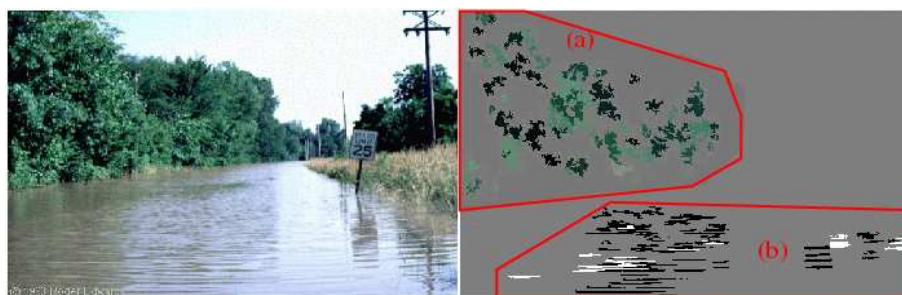


Figure 13. A image of river (in the left); and (a) green harsh irregular elements (shown in green or dark green), (b) horizontal bar elements (shown in black and white).

3.2. Clustering Structural Elements

Given an image, for each category of structural elements we are interested in, we apply clustering algorithms to find the clusters.

For a given category, we first find the set of qualified elements (i.e., the elements belong to this category). For two qualified elements, if they

overlap³, then the one with lower qualifying score⁴ is removed. This process is called *element sifting*. After sifting, we reduce multi-level elements of the image into one level elements, which all belong to the given category. The elements after sifting form some element pattern in the 2D plane.

For the element pattern, we construct its Delaunay graph, which is the neighboring graph of the element pattern. See Figure 14. We then apply clustering algorithms on the neighboring graph to find the clusters in the element pattern. In our implementation, the clustering algorithm is based on the minimal spanning tree of the neighboring graph. The main idea of clustering by the minimal spanning tree is as follows. Let V be the vertex set of the graph, and D be the distance threshold. Then V is grouped into disjoint sets by joining all edges in the minimal spanning tree whose lengths (or weights) are less than or equal to D . Each set obtained this way is said to be a cluster at level D . Reference on pattern processing by neighboring graph can be found in [Ahuja, 1982; Ahuja and Tuceryan, 1989]. Reference on clustering by minimal spanning tree can be found in [Zahn, 1971].

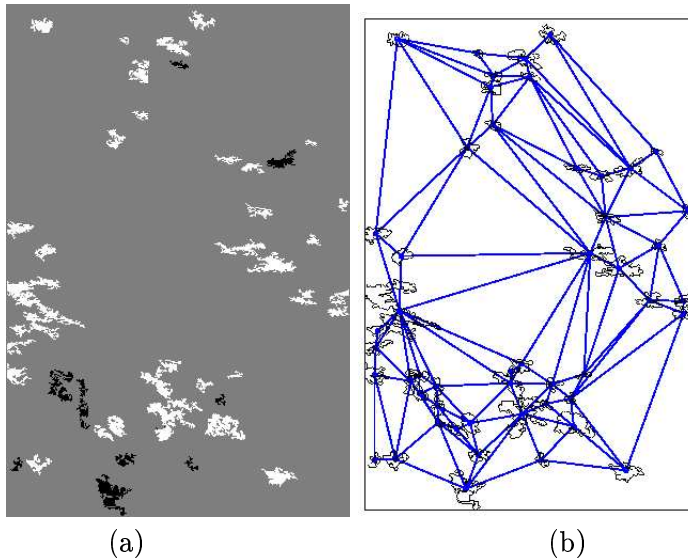


Figure 14. (a) Some element pattern, (b) the neighboring graph of the pattern.

³By the definition of monotonic tree, when two elements overlap, then the covered region of one element is a super set of the covered region of the other.

⁴For a given category, the qualifying score of an element indicates how qualified the element is to belong to this category.

3.3. Rendering Semantic Regions

Given a cluster of structural elements, the region rendering process consists of three steps: (1) element connecting; (2) hole filling; and (3) boundary smoothing. At the first step, we connect all elements in the cluster by line segments whose lengths are within a threshold. At the second step, we fill the holes whose areas are less than an area threshold. At the last step, we smooth the boundary of the region by removing those irregular angles and branches. Figure 15 shows an example of these steps.

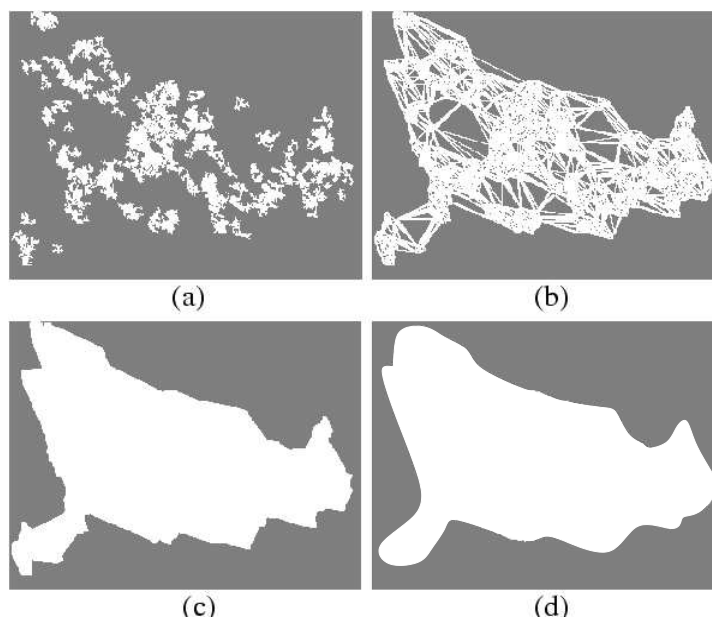


Figure 15. (a) A cluster of structural elements, (b) element connecting (c)hole filling, and (d)boundary smoothing.

3.4. Annotating the Input Image

For the regions rendered in the last subsection, we measure their qualifying scores to determine their semantics and annotate the corresponding keywords to the input image.

In general, qualifying scores must be established for structural elements, regions, and images. The qualifying score of a structural element measures the degree to which the element is considered to belong to a given category of elements. Qualifying scores for different element categories are determined in different ways.

Similarly, the qualifying score of a region with respect to a semantic feature measures the degree that the region manifests this particular feature. For a region generated from a cluster, its qualifying score is determined by the qualifying scores of the elements in the cluster and the area of the region. For example, a wave region is rendered from a cluster of horizontal bar elements. Let the input image be defined on the domain Ω and R be a wave region generated from a cluster S of horizontal bar elements. The qualifying score for R with respect to wave is defined as

$$Score_{wave}(R) = \frac{\sum_{t \in S} Score_{hbar}(t)}{\alpha_{wave}} * \frac{Area(R)}{Area(\Omega)},$$

where $Score_{hbar}(t)$ is the qualifying score for t to be a horizontal bar element, and α_{wave} is a parameter to scale the qualifying score.

In contrast, sky and ground regions are not generated from clusters. As will be discussed in the next section, we will assume that these regions are smooth. Coarse regions are identified first; the smooth regions are then the complement of the coarse regions. The qualifying score of a smooth region is determined by the smoothness, color, and area of the region. As with the harshness of a structural element, the smoothness of a region is determined by the area and altitude of the structural elements intersecting this region:

$$Smoothness(R) = \frac{Area(R)}{\sum_{t \in ElementSet} Altitude(t) * Area(Region(t) \cap R)},$$

where $ElementSet$ is the set of all structural elements of the input image and $Region(t)$ is the region covered by t . The qualifying score for a region R to be considered a sky region is defined as ⁵

$$Score_{sky}(R) = \frac{Smoothness(R)}{\alpha_{sky}} * \frac{Area(R)}{Area(\Omega)} * SkyColorRatio(R),$$

where α_{sky} is a parameter to scale the qualifying score and $SkyColorRatio(R)$ is the ratio of sky-colored pixels in R . ⁶ The qualifying scores for ground and placid-water regions are defined similarly.

For a region rendered in the last subsection, its qualifying score with respect to a semantic feature is measured; if the score exceeds a threshold, then the region is considered to have this particular feature and a corresponding keyword is annotated to this region.

⁵Region R should first be examined to ascertain that it satisfies our assumptions about sky regions.

⁶We assume that the sky color is either blue or gray (the color of clouds). These assumptions will be discussed in the next section.

An input image may have several regions which manifest a specific feature. The qualifying score for the input image with respect to this feature is defined as the sum of qualifying scores of these regions. For example, let I be the input image and R_1, R_2, \dots, R_n be the tree regions found in this image. The qualifying score of I with respect to “tree” is

$$Score_{tree}(I) = \sum_{i=1}^n Score_{tree}(R_i).$$

If an input image contains no regions which manifest a specific feature, then the qualifying score for the image with respect to this feature is zero. The features of the input image are extracted and stored in a feature vector. A feature vector is an array which records the qualifying scores for all semantic features. Figure 16 illustrates the data structure of a feature vector.

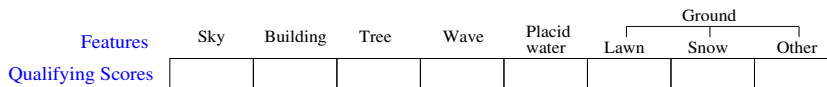


Figure 16. Data structure of feature vector.

Several sample images and their annotated copies are provided in Figure 27 (in Section 6).

4. Case Study of Semantic Features

In this section, we discuss the identification of high level scenery features: sky, building, tree, water wave, placid water, and ground. Water wave and placid water have different structures in images, thus they are treated with different schemes. The ground feature in images can be further split into snow, lawn and other kinds of ground.

4.1. Sky

Without clouds, a sky region is a homogeneous region consisting of blue pixels. In natural images, clouds tend to change smoothly at pixel level, due to their physical properties. In location, there is usually no other object above a sky region. To retrieve sky regions, We make three simple assumptions:

- (sky.a1) A sky region is smooth;
- (sky.a2) A sky region occupies an upper part of the image; and
- (sky.a3) The color of sky regions is either blue or the color of clouds.

For our current implementation, we assume that the color of clouds is gray.⁷

To find the sky regions, we first find the smooth regions in the image. The smooth regions are the complement of the harsh regions, which is characterized by intensity peaks and valleys. Under monotonic tree, the intensity peaks and valleys are modeled as small structural elements whose altitudes are high. Thus we detect the harsh regions of the image by clustering the small elements whose altitudes are high, as we discussed in last section. When we get the harsh regions, we also get the smooth regions. Then we check the location and color of the smooth regions to find the sky regions.

Figure 17 shows an example of sky and its annotation.



Figure 17. An example of sky and its annotation.

4.2. Ground and Placid Water

We make three assumptions about ground regions.

- (ground.a1) A ground region is smooth;
- (ground.a2) A ground region is in the lower part of the image; and
- (ground.a3) In a ground region, micro-structures are more horizontal than vertical.

When scenery pictures are taken, the direction of projection is usually horizontal or nearly horizontal. Thus, as the natural scene is projected to the image plane, the structures on the ground appear more horizontal than vertical, which is the reason why we make the third assumption.

Similar to detecting background regions, we first find the smooth regions in the image. Then for each smooth region, we check if the last

⁷Clouds can have more colors. In the following discussion, we also make some simple assumptions about the colors of trees, water and snow. More comprehensive color patterns of sky, trees, water, and snow can be integrated into our system, which is not the focus of this paper.

two assumptions hold or not. For the last assumption, we count the horizontal and vertical elements in the smooth region. The last assumption holds if the horizontal elements are more than the vertical elements in the region.

A ground region found this way could be lawn, snow, or other kinds of ground. We distinguish these kinds of regions by their colors. We assume that lawn regions are green, snow regions white.

For placid water, we make four assumptions. The first three are the same as the three assumptions of ground. Besides, we assume that the color of placid water is blue.

Figures 18 and 19 show examples of ground regions. Figure 20 shows an example of placid water.



Figure 18. An example of ground and its annotation. The building is annotated with “bldg”.



Figure 19. An example of snow ground and its annotation.

4.3. Wave

Small water waves have very regular patterns. When the small waves projected to a picture, they appear to be horizontal bars if the projecting direction is nearly horizontal. See the image in Figure 13. Wild waves have complicated structures. In the images with wild waves, we usually

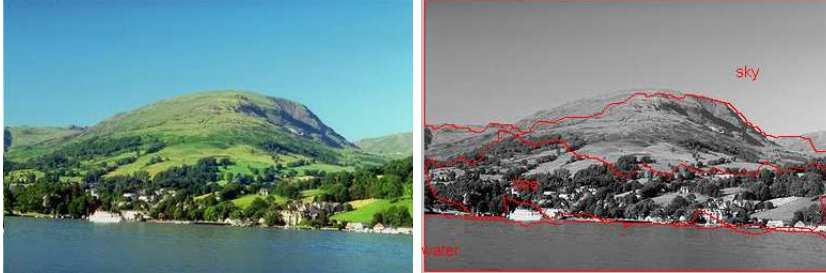


Figure 20. An example of placid water and its annotation.

can detect a piece of surface consisting of parallel bar structures. See the image in Figure 21. We assume that a wave region is a region consisting of horizontal bar elements. We detect wave regions by clustering horizontal bar elements in the image.

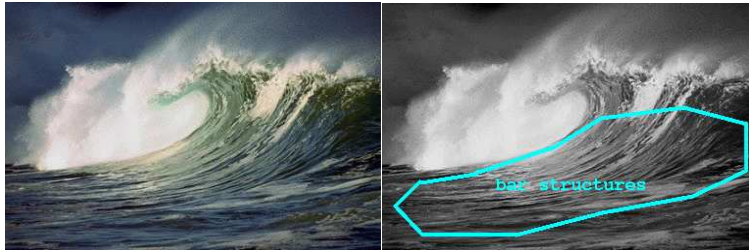


Figure 21. An example of image with big wave.

4.4. Green Tree

If we look at the tree region in the Figure 13 carefully, we can find that the micro structures in the region are very irregular. Based on this observation, we assume that a tree region is a region consisting of green⁸ harsh, irregular elements. The tree regions in an image are found by clustering the green, harsh irregular elements in the image.

4.5. Building

The shapes of most buildings are characterized by the line segments inside them. We assume that a building region in an image is a region consisting of polygon elements. To check whether a structural element is

⁸Again, we make a simple assumption about the color of trees.

a polygon element, we first partition its boundary into line segments and other kinds of segments. A structural element is a polygon element if its boundary mainly consists of line segments. Reference on curve partitioning can be found in [Robl and Farber, 1998; Rosin and West, 1989; Rosin and West, 1992]. Figure 22 shows an example of sky, building, and tree.



Figure 22. An example of sky, building and tree.

All of the above cases can be viewed in our demo system located at <http://monet.cse.buffalo.edu:8888/>.

5. Image Querying

Our demo system, named SceneryAnalyzer, has three main components: (1) image databases, (2) feature extraction, and (3) image querying. The feature extraction is done off-line. During feature extraction, each image is processed to generate semantic features, which are stored in a feature vector. The semantic features are automatically annotated in the gray copy of the original image. All feature vectors are stored in a vector base.

Figure 23 illustrates the interface for image querying. To retrieve images, users can select any combination of the listed features. After the query is submitted, the icons of images with the selected features are output in pages (Figure 24). The output images are sorted by their qualifying scores with respect to the submitted list of semantic features. The qualifying score of an image with respect to a list of semantic features is the product of the qualifying scores of that image with respect to the features in that list. If users click an icon, a window is pop up with the original and annotated images. Users can also try with images from the Internet or their local machines. After an image is uploaded, it's analyzed in the server side, and its semantic features are reported to the user. Then the user can submit the list of these semantic features as an image query.

SceneryAnalyzer: Image Retrieval By Monotonic Tree

Search by features

Choose image bases:
 COREL (31646 images) PhotoDisc (1444 images)

Choose features (we are adding more ...):

<input type="checkbox"/> Background (BK_GD)	<input checked="" type="checkbox"/> Tree (TREE)	Ground:
<input checked="" type="checkbox"/> Sky (SKY)	<input checked="" type="checkbox"/> Building (BLDG)	<input type="checkbox"/> Lawn (LAWN)
	<input type="checkbox"/> Wave (WAVE)	<input type="checkbox"/> Snow (SNOW)
	<input type="checkbox"/> Water (WATER)	<input type="checkbox"/> Other (GROUND)

Search by an image

Upload an image from web or your local machine:

web file or

local file

to upload the image.

Online
Search
for
Scenic
Features

Figure 23. Interface of SceneryAnalyzer.

6. Experiments

We conducted experiments to compare the performance between our approach and traditional CBIR techniques including keyblock model [Zhu, L., Rao, A., and Zhang, A., 2000], traditional color histogram [Swain and Ballard, 1991], color coherent vector [Pass et al., 1996], and wavelet (Haar and Daubechies) texture techniques [Smith and Chang, 1994; Strang and Nguyen, 1996]. The comparison is made by the precisions and recalls of each method on six scenery features: sky, building, tree, wave, ground, and placid water.

We used 6776 COREL images in the experiments. They are selected from CD7 and CD8 of COREL Gallery 1,300,000. The COREL images can be split into two parts: scenery and nonscenery parts. There are

Image Retrieval By Monotonic Tree
Features selected: sky tree/grass building
203 images retrieved (page 1)

Next 20 [UP](#) [HOME](#)

 734079.jpg	 881097.jpg	 47010.jpg	 393072.jpg	 232036.jpg
 819082.jpg	 838087.jpg	 102097.jpg	 438062.jpg	 374043.jpg
 54059.jpg	 281022.jpg	 279028.jpg	 54095.jpg	 242016.jpg
 67023.jpg	 463021.jpg	 149002.jpg	 82058.jpg	 67027.jpg

Next 20 [UP](#) [HOME](#)

Figure 24. Retrieval results.

4125 scenery images, which are pictures taken at countries all around the world. The nonscenery part has 2651 images, which cover a large variety, including different kinds of textures, marine fishes, molecules, space scenes, and insects. Table 1 shows the statistics of the scenery features in the COREL images (each entry shows the number of images with some feature).

Table 1. The statistics of scenery features in 6776 COREL images.

<i>feature</i>	sky	building	tree	wave	placid water	ground		
						lawn	snow	other
<i>images</i>	2369	1920	1479	161	882	298	68	659

For each scenery feature, to show the performance of SceneryAnalyzer, we calculate the precisions and recalls of first $\frac{n}{30}$, $\frac{2n}{30}$, $\frac{3n}{30}$, ..., and n images retrieved by SceneryAnalyzer with this feature, where n is the total number of images retrieved by SceneryAnalyzer.

Traditional CBIR techniques accept only queries by examples. Let's take an example to show how we choose query sets and calculate the precision-recall for these methods. The example is about keyblock on sky feature. There are 2369 COREL images with sky regions. For each sky image, we use it as a query on the COREL database to select top 100 images by the keyblock approach, and count the number of sky images in this retrieved set. Then we sort the 2369 sky images descendingly by the numbers of sky images in their corresponding retrieved sets. Let the sorted list be *SKYLIST*. Then we select the first 5%, i.e., 118 images of *SKYLIST* as the query set, which is denoted as *QUERYSET*. Then for each COREL image I , we calculate its distance to *QUERYSET* - $\{I\}$ by the keyblock approach. The COREL images are sorted ascendantly by this distance. Top 2369⁹ COREL images are retrieved. Based on the retrieved images by keyblock, we calculate and plot the precision-recall of keyblock on sky, as we did for SceneryAnalyzer.

The result is in Figure 25. By comparing the graphs in Figure 25, we can see that our method outperforms all the others on each feature. To see the comparison clearly, we calculate the average precision-recall on the six scenery features, which is shown in Figure 26. From these comparisons, we can see that our method is much better than the traditional techniques on scenery features.

Figure 27 provides sample images and their copies as annotated by SceneryAnalyzer.

⁹This is the number of COREL images with sky.

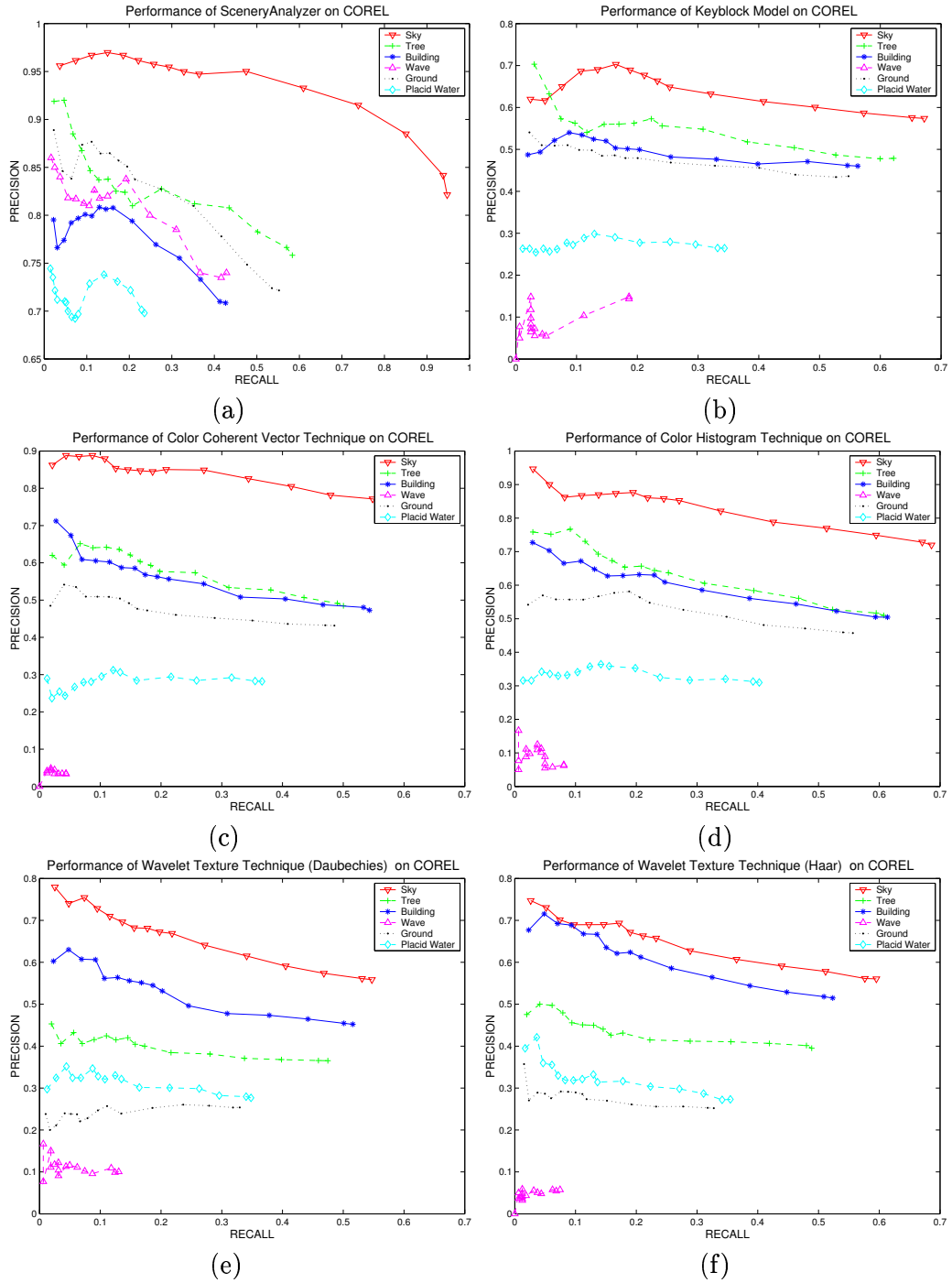


Figure 25. Performance of (a) SceneryAnalyzer, (b) keyblock model, (c) color coherent vector, (d) color histogram, (e) Daubechies wavelet, and (f) Haar wavelet on COREL images

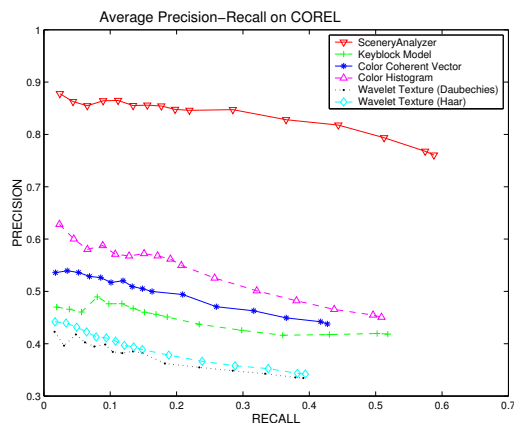


Figure 26. Average precision-recall on six features: sky, building, tree, wave, ground and placid water.

7. Conclusion

In this paper, we used a model termed monotonic tree to model high-level scenery features. Based on the monotonic tree representation, primitive elements of low-level features such as color, shape, harshness, and spatial location can be easily identified, clustered and combined to form semantically meaning regions (or features) for images. Thus, images can be automatically annotated with category keywords, including sky, building, tree, wave, lawn, water, snow, and ground. With this annotation, high-level (semantics-based) querying and browsing of images can be supported.

In future work, we intend to extend the proposed approach to categories other than scenery images. It may be cumbersome to provide a hand-crafted strategy for the identification of structural elements for each semantic feature. It will therefore be necessary for us to use machine-learning techniques to assist in the process of identifying structural elements for general semantic features.

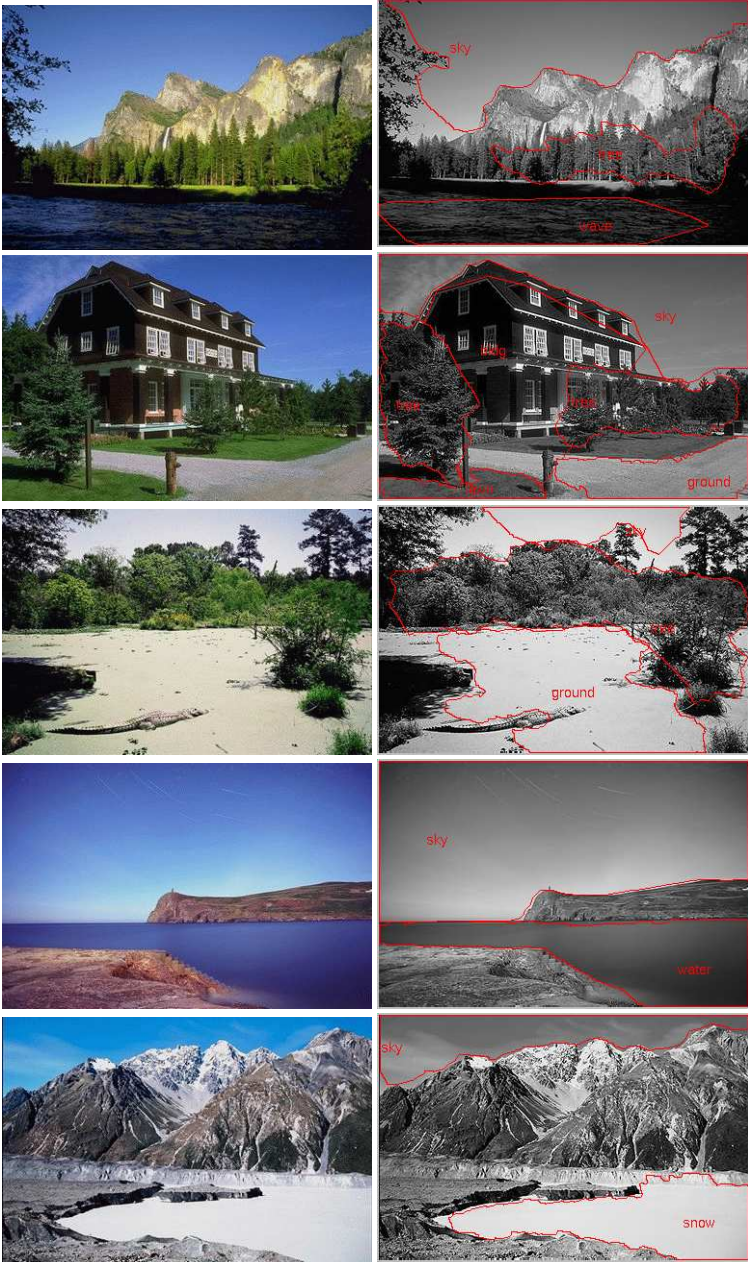


Figure 27. Examples of scenery images and their annotated copies.

References

- Ahuja, N. (1982). Dot pattern processing using voronoi neighborhoods. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 4(3):336–343.
- Ahuja, N. and Tuceryan, M. (1989). Extraction of early perceptual structure in dot patterns: integrating region, boundary, and component gestalt. *Computer Vision, Graphics, and Image Processing*, 48(3):304–356.
- Bjarnestam, A. (Feb.5, 1998). Description of an image retrieval system. In *The Challenge of Image Retrieval research workshop, Newcastle upon Tyne*.
- Eakins, J. and Graham, M. (Jan. 10, 1999). Content-based image retrieval. In *Reports of JISC Technology Applications Programme*. URL=<http://www.jtap.ac.uk/reports/htm/jtap-039.html>.
- Forsyth, D., Malik, J., Fleck, M., Greenspan, H., Leung, T., Belongie, S., Carson, C., and Bregler, C. (1996). Finding pictures of objects in large collections of images. In *Report of the NSF/ARPA Workshop on 3D Object Representation for Computer Vision*, page 335.
- Manjunath, B. and Ma, W. (1996). Texture Features for Browsing and Retrieval of Image Data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(8):837–842.
- Morse, S. P. (1969). Concepts of use in computer map processing. *Communications of the ACM*, 12(3):147–152.
- Pass, G., Zabih, R., and Miller, J. (1996). Comparing images using color coherence vectors. In *Proceedings of ACM Multimedia 96*, pages 65–73, Boston MA USA.
- Robl, C. and Farber, G. (1998). Contour tracer for a fast and precise edge-line extraction. In *IAPR Workshop On Machine Vision Applications (MVA98)*.
- Rosin, P. and West, G. (1989). Segmentation of edges into lines and arcs. *Image and Vision Computing*, 7(2):109–114.
- Rosin, P. and West, G. (1992). Multi-stage combined ellipse and line detection. In *British Machine Vision Conference (BMVC92)*, pages 197–206.
- Roubal, J. and Poiker, T. (1985). Automated contour labelling and the contour tree. In *Proc. AUTO-CARTO 7*, pages 472–481.
- Safar, M., Shahabi, C., and Sun, X. (2000). Image retrieval by shape: A comparative study. In *Proceedings of IEEE International Conference on Multimedia and Exposition ICME, USA*.
- Shahabi, C. and Safar, M. (1999). Efficient retrieval and spatial querying of 2d objects. In *IEEE International Conference on Multimedia Computing and Systems (ICMCS99)*, pages 611–617, Florence, Italy.

- Sheikholeslami, G. and Zhang, A. (1997). An Approach to Clustering Large Visual Databases Using Wavelet Transform. In *Proceedings of the SPIE Conference on Visual Data Exploration and Analysis IV*, pages 322–333, San Jose.
- Smith, J. R. and Chang, S. (1994). Transform Features For Texture Classification and Discrimination in Large Image Databases. In *Proceedings of the IEEE International Conference on Image Processing*, pages 407–411.
- Smith, J. R. and Chang, S.-F. (1996). VisualSeek: a fully automated content-based image query system. In *Proceedings of ACM Multimedia 96*, pages 87–98, Boston MA USA.
- Song, Y. (2002). Monotonic tree and its application to multimedia information retrieval. *PhD dissertation, Department of Computer Science and Engineering, State University of New York at Buffalo*. URL=<http://www.cse.Buffalo.EDU/~ys2/publication/thesis.pdf>.
- Song, Y. and Zhang, A. (April 3-5, 2002). Monotonic tree. In *The 10th International Conference on Discrete Geometry for Computer Imagery, Bordeaux, France*.
- Strang, G. and Nguyen, T. (1996). *Wavelets and Filter Banks*. Wellesley-Cambridge Press, Wellesley, MA.
- Swain, M. and Ballard, D. (1991). Color Indexing. *Int Journal of Computer Vision*, 7(1):11–32.
- Tao, Y. and Grosky, W. (1999). Delaunay triangulation for image object indexing: A novel method for shape representation. In *Proceedings of the Seventh SPIE Symposium on Storage and Retrieval for Image and Video Databases*, pages 631–942, San Jose, California.
- Torralba, A. and Oliva, A. (1999). Semantic organization of scenes using discriminant structural templates. In *International Conference on Computer Vision (ICCV99)*, pages 1253–1258.
- van Kreveld, M., van Oostrum, R., Bajaj, C., Pascucci, V., and Schikore, D. (1997). Contour trees and small seed sets for iso-surface traversal. In *Proceedings of the 13th ACM Symposium on Computational Geometry*, pages 212–220.
- Zahn, C. (1971). Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Trans. on Computers*, C-20:68–86.
- Zhao, R. and Grosky, W. (2001). Bridging the semantic gap in image retrieval. In Shih, T., editor, *Distributed Multimedia Databases: Techniques and Applications*, pages 13–36. Idea Group Publishing, Hershey, Pennsylvania.
- Zhu, L., Rao, A., and Zhang, A. (2000). Theory of keyblock-based image retrieval. *ACM Transactions on Information Systems*.