Semantics Retrieval by Content and Context of Image Regions*

Wei Wang, Yuqing Song and Aidong Zhang
Department of Computer Science and Engineering
State University of New York at Buffalo
Buffalo, NY 14260 USA

Abstract

We propose a novel approach for semantics retrieval from images in multimedia databases. In our approach, we use color-texture classification to generate the codebook which is used to segment images into regions. The content of a region describes the lower-level features of the region, including color and texture. The context of regions in an image describes their relationships in the image. The content and context of image regions provide a way for semantics retrieval. On top of semantics retrieval, high-level (semantics-based) querying and query-by-example are supported. The experimental results demonstrate that our approach outperforms the traditional CBIR approaches.

1 Introduction

Although content-based image retrieval (CBIR) techniques based on low-level features such as color, texture, and shape have been extensively explored, their effectiveness and efficiency are not satisfactory. The ultimate goal of image retrieval is to provide the users with the facility to manage large image databases in an automatic, flexible and efficient way. Therefore, image retrieval systems should be armed to support high-level (semantics-based) querying and browsing of images. The basic elements to carry semantic information are the image regions which correspond to semantic objects, if the image segmentation is effective. After the regions are obtained, proper representation of their content and context remains a challenge. The extraction of content and context of image regions is a necessary step for semantics retrieval.

Many methods have been proposed for region-based image retrieval. Under keyblock model [4], images are partitioned into equal-sized blocks and features are extracted by training blocks to form the codebook. Images are indexed using the codebook. Image retrieval is performed based on the indexed images. The equal-sized blocks ignore the boundary of regions, therefore they cannot represent the objects correctly. Another image querying system developed by Smith et al [7] first decomposes an image into regions with characterizations pre-defined in a finite pattern library. With every pattern labeled by a symbol, images are represented by region strings. Region strings are converted to descriptor matrices of composite region templates(CRT), which reflect the relative ordering of symbols. The CRT library is solely dependent on color feature. If texture or shape features are added to distinguish patterns, the size of library will increase dramatically which makes the retrieval process inefficient. Li et al proposed IRM [3] which allows matching a region of one image to several regions of another image. That is, the region matching between two images is a many-to-many relationship. As a result, the similarity between two images is defined as the weighted sum of distances in the feature space between all regions from different images. All the approaches introduced above could not integrate the semantic descriptions into the regions (or blocks), therefore cannot support the high-level querying of images.

Semantics retrieval from images can be performed by annotating the images. In [9, 8], the monotonic tree is used as a hierarchical representation of image structures. Microstructure called *structural elements* are classified and clustered using methods based on minimum spanning tree. The images are rendered with semantic annotation keywords. The system performs well on scenery images. However, it does not take the context of image regions into consideration.

In this paper, we propose a novel approach for semantics retrieval from images based on the content and context of image regions. Our method consists of three levels. At pixel level, color-texture classification is used to form the semantic codebook. At region level, the semantic codebook is used to segment the images into regions. At image level, the content and context of image regions are defined and represented to support the semantics retrieval from images. The content of a region describes the lower-level features of the region, including color and texture. The context of regions in an image describes their relationships in the image. The three levels are illustrated in Figure 1.

The remainder of the paper is organized as follows. From Section 2 to Section 4, each step of our approach is elaborated. In Section 5 experimental results will be pre-

*This research was supported by NSF Digital Government Grant EIA-9983430.

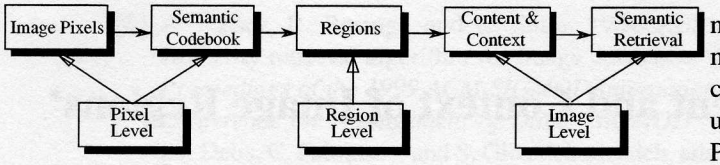


Figure 1: System design of our approach.

sented, and we will conclude in Section 6.

2 Semantic codebook based on color-texture classification

We intend to generate the semantic codebook to represent the low level features including color and texture such that the high-level semantic description can be supported. The formation of semantic codebook is based on the color-texture feature vectors for pixels in the images.

2.1 Color-texture feature vector

2.1.1 Color

We will use a feature vector to represent the color-texture features of each pixel. For each pixel, the three color features we choose are the averaged *RGB* values in a neighborhood. For any pixel q, let Ω be a neighborhood and S be the area of Ω. We define the color features of q as

$$\begin{aligned} R_{FV}(q) &= \frac{\sum_{p\in\Omega} Red(p)}{S}, \\ G_{FV}(q) &= \frac{\sum_{p\in\Omega} Green(p)}{S}, and \\ B_{FV}(q) &= \frac{\sum_{p\in\Omega} Blue(p)}{S}, \end{aligned}$$

where $Red(p), Green(p)$, and $Blue(p)$ are the *RGB* components of pixel p, respectively. For pixel q, the color feature vector is $[R_{FV}(q),\ G_{FV}(q),\ B_{FV}(q)]$, denoted as $Color_{FV}$.

2.1.2 Texture

To characterize the texture patterns, we use texture features based on the *second moment matrix* [1, 2, 5]. The second moment matrix can be thought of as a covariance matrix of a two-dimensional random variable, or, with a mechanical analogy, as the moment of inertia of a mass distribution in the plane. Its eigenvalues represent the amount of "energy" in the two principle directions in a neighborhood. When one eigenvalue is larger than the other, the local neighborhood possesses a dominant orientation and can be characterized as 1D texture. When the eigenvalues are comparable, there is no preferred orientation. When both eigenvalues are negligible in magnitude, the local neighborhood is approximately a constant intensity and can be characterized as low contrast (e.g., sky). For the case of two significant eigenvalues, the region is characterized as 2D texture (e.g., grass). Pixels with local texture characterized as 1D texture may be part of a pattern in one direction (called "flow", e.g., zebra stripes), or an edge (see Figure 2).

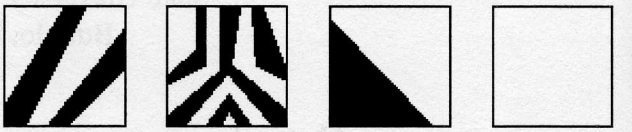


Figure 2: Different texture characteristics. From left to right, Flow, 2D pattern, Edge and Low Contrast.

Based on these texture patterns, we use three values to describe the texture characteristics of a certain pixel in its neighborhood: *anisotropy, contrast and flowOrEdge*. *Anisotropy* measures the energy comparison between the dominant orientation and its orthogonal direction. *Contrast* reflects the contrast, or harshness of the neighborhood. *FlowOrEdge* can distinguish whether an 1D texture is a flow or an edge. In following discussion, we will give a formal description of these values.

Let $L : R^2 \to R$ be the image brightness, and let $\nabla L = (L_x, L_y)^T$ be its gradient. The second moment matrix around a pixel q is defined as: [2]

$$\begin{aligned} \mu_L(q) &= \begin{pmatrix} \mu_{11} & \mu_{12} \\ \mu_{21} & \mu_{22} \end{pmatrix} = E_q \begin{pmatrix} L_x^2 & L_xL_y \\ L_xL_y & L_y^2 \end{pmatrix} \\ &= E_q((\nabla L)(\nabla L)^T), \end{aligned}$$

where E_q denotes an averaging operator [1] around pixel $q = (x,y)^T \in R^2$. Let $\lambda_1(q)$ and $\lambda_2(q)$ denote the eigenvalues of $\mu_L(q)$. Without loss of generality, we assume $\lambda_1(q) \geq \lambda_2(q)$. For pixel q, anisotropy and contrast are defined, respectively, as

$$\begin{aligned} anisotropy(q) &= 1 - \lambda_2(q)/\lambda_1(q), \\ contrast(q) &= 2\sqrt{\lambda_1(q) + \lambda_2(q)}. \end{aligned}$$

In [5], a method is provided to distinguish whether a 1D texture is an edge or a flow by looking at the sign of the gradient vectors. Based on the method, let $E_+(q)$ and $E_-(q)$ represent how many gradient vectors in the neighborhood are on the "positive" and "negative" sides of the dominant orientation, respectively. we define flowOrEdge of pixel q as[2]:

$$flowOrEdge(q) = 1 - \frac{min(E_+(q), E_-(q))}{max(E_+(q), E_-(q))}.$$

[1] we use Gaussian smoothing kernel.

[2] Definitions of E_+ and E_- are not introduced here due to the space limit. Refer to [5].

The texture feature vector for pixel q is $[anisotropy(q),\ contrast(q),\ flowOrEdge(q)]$, denoted as $Texture_{FV}$. For each pixel, we have a six-dimensional feature vector where three dimensions are for color, and three for texture.

2.2 Color-texture classification

We intend to classify the pixels based on their feature vectors. For color, we uniformly quantize the color space into $4 \times 4 \times 4 = 64$ cells. For texture, we classify the texture patterns into 7 classes: one class for low contrast and edge, respectively; two classes for flow and three for 2D texture. The classification illustrated in Figure 3 is based on $Texture_{FV}$.

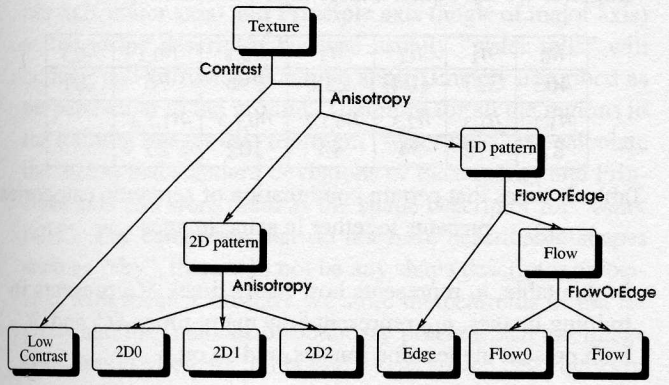


Figure 3: Texture Classification

Based on texture feature vector defined for each pixel, we give the definitions of the texture classes below. We first define $LowContrast, Flow, Edge$, and $2D\ pattern$:

$$
\begin{aligned}
LowContrast &= \{q | contrast(q) \le \varepsilon\}, \\
Flow &= \{q \mid \begin{array}{lcl} contrast(q) & > & \varepsilon \\ anisotropy(q) & \ge & \eta \\ flowOrEdge(q) & < & \delta \end{array}\}, \\
Edge &= \{q \mid \begin{array}{lcl} contrast(q) & > & \varepsilon \\ anisotropy(q) & \ge & \eta \\ flowOrEdge(q) & \ge & \delta \end{array}\}, \\
2D &= \{q \mid \begin{array}{lcl} contrast(q) & > & \varepsilon \\ anisotropy(q) & < & \eta \end{array}\},
\end{aligned}
$$

where we choose $\varepsilon = 0.01$, $\eta = 0.75$ and $\delta = 0.5$. We split $Flow$ into two classes:

$$
\begin{aligned}
Flow_0 &= \{q \mid q \in Flow, flowOrEdge(q) \in [0, \delta/2)\}, \\
Flow_1 &= \{q \mid q \in Flow, flowOrEdge(q) \in [\delta/2, \delta)\}.
\end{aligned}
$$

For $Flow_0$, the bigger value of E_+ and E_- is relatively close to the smaller value. $Flow_0$ represents thin flows, while $Flow_1$ represents fat flows. We split $2D\ pattern$ into three classes:

$$
\begin{aligned}
2D_0 &= \{q \mid q \in 2D, anisotropy(q) \in [0, \eta/3)\}, \\
2D_1 &= \{q \mid q \in 2D, anisotropy(q) \in [\eta/3, 2\eta/3)\}, \\
2D_2 &= \{q \mid q \in 2D, anisotropy(q) \in [2\eta/3, \eta)\}.
\end{aligned}
$$

$2D_0$ represents the texture patterns with random 2D structure. $2D_2$ refers to the texture patterns similar to a flow but major orientation is not dominant. $2D_1$ is the middle case between $2D_0$ and $2D_2$. We now have 7 fine-tuned texture classes:

$$LowContrast, 2D_0, 2D_1, 2D_2, Edge, Flow_0, Flow_1$$

as shown in Figure 3. Because edges cannot reflect the color-texture characteristics of image regions semantically (although it does have important cue in the perception), we classify all the pixels, except those pixels corresponding to edges, to 6 texture classes $LowContrast, 2D_0, 2D_1, 2D_2, Flow_0, Flow_1$. In the remaining part of this paper, we will focus on pixels which don't belong to the class $Edge$.

2.3 Generation of semantic codebook

We define the major semantic categories and select the training images for each category. Then color-texture classification is applied to the training images to form the codebook. The concept of *codebook* is from the vector quantization. In general, a vector quantizer Q of dimension k and size N is a mapping from a vector in k dimensional space into a finite set C, $Q : R^k \longrightarrow C$. The set C is called the *codebook*.

Based on the semantic constraints of the images in the database, major semantic categories are selected as:

$$SC = \{SC_0, ..., SC_{M-1}\}.$$

For example, if we are interested in the scenery images, SC might include "Sky", "Tree", "Flower", "Water", etc. For our experiments in Section 5, we define SC of 10 different categories. For each semantic category SC_i, certain number of images are chosen to be the training images. The training images are carefully selected such that the semantic codebook generated from them can represent as much as possible the color-texture characteristics of all images belonging to the SC. The set of all pixels in the training images, except those in class $Edge$, is denoted as $TrainingPixels$.

As stated before, we uniformly quantize the RGB color space to $C(C = 64)$ cells. Therefore, corresponding to $T(T = 6)$ texture classes, we now split the color-texture space (denoted as CTS, excluding the pixels on the edges) to $C \times T$ cells. For each pixel in $TrainingPixels$, its 6-dimensional color-texture feature vector will fall into one cell of CTS. After all the pixels in $TrainingPixels$ are classified, for each cell in CTS we count the number of pixels in it.

Only those cells whose number of pixels exceeds a threshold will be one entry of the semantic codebook. Therefore the size of semantic codebook will be less or equal to $C \times T$, no matter how large the image database might be. In this way, the semantic codebook represents the color-texture characteristics of the training images which in turn represent the whole database, effectively and efficiently. In following discussion, we use *SCDB* to denote the semantic codebook for color-texture space, and suppose its size is N.

3 Image segmentation based on the semantic codebook

We segment images by the semantic codebook. For each image I, we extract the color-texture feature vector for each pixel q of I. For each feature vector, we find the cell in CTS where the pixel belongs. If the cell is one of *SCDB*'s entries, q is replaced with the *index* of that entry; otherwise its value is set to $C \times T + 1$ to distinguish it from any valid entry of the semantic codebook. After the process, I becomes a matrix of indices corresponding to entries in *SCDB*. Because the number of valuable regions in an image is usually less than 5 in our experiments, we only choose 5 most dominant indices (referred as *DOMINANT*) and use *DOMINANT* to re-index the pixels with indices not present in *DOMINANT* [3]. Finally we run the encoded images through a connected-components algorithm and remove the small regions (with area less than 200 pixels).

4 Representation of Content and Context of regions and semantics retrieval of images

4.1 Generation of statistical data from training images

For each entry e_i in the semantic codebook *SCDB*, and each semantic category SC_j, we count the number of regions R in the training images such that (i) the pixels in R belong to e_i; and (ii) the region R represents an object belonging to the semantic category SC_j. The result is stored in a table, called *cell-category statistics table*. For example, suppose we have four semantic categories $\{Sky, Tree, Flower, Water\}$, and the table of the cell-category statistics is given below. From the table, we can see that the first codebook entry (with index 0) has large probability to represent sky regions, the second entry mostly represents tree regions, and so on.

[3] Re-index means to find the closest codebook entry in *DOMINANT*, not in the whole *SCDB*.

SC_i	*SCDB*'s entry index				
	0	1	2	...	$N-1$
Sky	134	0	0	...	54
Tree	0	86	0	...	0
Flower	0	0	15	...	0
Water	40	0	0	...	12

Table 1: Statistics that certain *SCDB*'s entry represents certain semantic category

In addition, we count the times that two or three different categories present in the same training images. The result is stored in a table, called *category-category statistics table*. For example, let the number of categories in *SC* be 4, and the index for each category of *SC* is 0, 1, 2, 3, respectively. Suppose we have such a table after the counting:

n_0	n_1	n_2	n_3	$n_{0,1}$	$n_{0,2}$	$n_{0,3}$
465	327	104	258	298	76	201
$n_{1,2}$	$n_{1,3}$	$n_{2,3}$	$n_{0,1,2}$	$n_{0,1,3}$	$n_{1,2,3}$	
87	178	54	43	102	25	

Table 2: times that certain combination of semantic categories presents together in same images

In this table, n_i represents how many times SC_i presents in training images, $n_{i,j}$ represent how many times SC_i and SC_j both present in the same images, and so on.

Based on the cell-category statistics table, for each codebook entry of *SCDB*, we can calculate its probability of representing a certain semantic category. Let N be the size of *SCDB*, M be the number of the semantic categories, $i \to SC_j, i \in [0...N-1], j \in [0...M-1]$ denote the event that index of *SCDB* i represents semantics described in SC_j. The probability of the event $i \to SC_j$ is

$$P(i \to SC_j) = \frac{T(i \to SC_j)}{T(i)},$$

where $T(e)$ represents event e's presence times in the training images.

Based on the category-category statistics table, we define the Bi-correlation factor and Tri-correlation factor, for representing the context of regions.

Definition 1 For any two different semantic categories SC_i and SC_j, we define the **Bi-correlation factor** $B_{i,j}$ as:

$$B_{i,j} = \frac{n_{i,j}}{n_i + n_j - n_{i,j}}.$$

Definition 2 For any three different semantic categories SC_i, SC_j and SC_k, we define the **Tri-correlation factor** $T_{i,j,k}$ as:

$$T_{i,j,k} = \frac{n_{i,j,k}}{n_i + n_j + n_k - n_{i,j} - n_{i,k} - n_{j,k} + n_{i,j,k}}.$$

In above definitions, n_i, $n_{i,j}$ and $n_{i,j,k}$ are the entries in the category-category statistics table. Bi and Tri-correlation factors reflect the correlation between two or three different categories within the scope of training images. If the training images are selected suitably, they reflect the relationship of pre-defined semantic categories we are interested in.

So far we did not mention how shape and/or spatial description of the image regions play a role in our approach. Due to the extreme difficulty of segmentation and complexity of objects' shapes, no single shape or spatial descriptor can generally satisfy the requirements of recognition. Therefore, for each semantic category, we adopt a particular shape and/or spatial descriptor. For instance, assume we have a semantic category "water falls", we can use Eccentricity (minor axis/major axis) and Principle axis (angle of major axis) as the shape descriptor because usually "water falls" will be long and narrow, and can be approximately described as perpendicular to the ground. Therefore for all the regions in the training images that represent "water falls", we calculate the means and standard deviations of Eccentricity and Principle axis and store them as the shape descriptor for "water falls". For categories that do not have describable shapes such as "sky", there will not be any shape descriptor associated with them. Currently we store the centroids of the regions and the (number of boundary pixels which are image boundary)/(whole region boundary) as spatial descriptors of the regions.

4.2 Representation of Content and Context of regions

Armed with the statistical data we have generated from the training images, we can define and extract content and context of regions for all the images in the database. Assume we have an image I with number of Q regions. The *SCDB*'s codebook indices of regions are $C_i, i \in [0...Q-1]$, and regions are represented by $R_i, i \in [0...Q-1]$, and each C_i is associated with i_N possible semantic categories $SC_{i_{j(i)}}, i_{j(i)} \in [0...N-1], j(i) \in [0...i_N-1]$. For I, let P_{all} be the set of all possible combinations of *indices* of semantic categories represented by regions $R_i, i \in [0...Q-1]$. We have

$$P_{all} = \{(0_{j(0)},...,i_{j(i)},...,Q-1_{j(Q-1)}) \mid \forall i, P(C_i \rightarrow SC_{i_{j(i)}}) > 0, \; i \in [0...Q-1], \; i_{j(i)} \in [0...N-1], \; j(i) \in [0...i_N-1]\}$$

Note there are totally $\prod_{i=1}^{Q} i_N$ possible combinations for P_{all}, therefore P_{all} has $\prod_{i=1}^{Q} i_N$ tuples of semantic categories indices, with each tuple having Q fields.

Corresponding to each tuple $\kappa \in P_{all}$, for $Q >= 2$ we have

$$B(\kappa) = \sum_{p,q \in \kappa, p<q} B_{p,q}, \quad \kappa \in P_{all},$$

and for $Q >= 3$ we have

$$T(\kappa) = \sum_{p,q,r \in \kappa, p<q<r} T_{p,q,r}, \quad \kappa \in P_{all}.$$

Here p,q,r are the indices belonging to tuple κ, $B(\kappa)$ represents the sum of Bi-correlation factors of different semantic categories with regard to tuple κ, and $T(\kappa)$ represents the sum of Tri-correlation factors of different semantic categories with regard to tuple κ.

Definition 3 we define **Context** $C_{Score}(\kappa)$ of I as:

$$C_{Score}(\kappa) = \frac{1}{Norm(\kappa)}(B(\kappa) + \beta T(\kappa)), \quad \kappa \in P_{all}$$

here $Norm(\kappa)$ is the normalization function with tuple κ, β is the weight for $T(\kappa)$, since $T(\kappa)$ will be more effective in distinguishing contexts of images than $B(\kappa)$ [4]. We normalize the C_{Score} because several region indices may point to the same semantic category, we need to guarantee that removal the redundant semantic category will not influence the effectiveness of C_{Score}.

Definition 4 we define **ProbScore** $P_{Score}(\kappa)$ of I as:

$$P_{Score}(\kappa) = \sum_{i=1}^{Q} w(R_i, SC_{i_j}) P(C_i \rightarrow SC_{i_j}), i_j \in \kappa, \kappa \in P_{all}$$

$P_{Score}(\kappa)$ represents the probability score corresponding to tuple κ, here $w(R_i, SC_{i_j})$ is the weight function with regard to region R_i and semantic category SC_{i_j}. $w(R_i, SC_{i_j})$ is relevant to the shape and/or spatial descriptors of R_i. Suppose the region R_i's shape and/or spatial descriptors with regard to SC_{i_j} is compatible with SC_{i_j}'s stored shape and/or spatial descriptors, $w(R_i, SC_{i_j})$ will be increased, otherwise it will be decreased.

Definition 5 We define the **TotalScore** T_{Score} of image I as

$$T_{Score} = Max\{P_{Score}(\kappa) + \gamma C_{Score}(\kappa) \mid \kappa \in P_{all}\}$$

where $Max\{t\}$ represents the maximum value of value t.

Definition 6 We define the **Content** of image I as the semantic categories corresponding to T_{Score}. By computing the maximum value of $P_{Score}(\kappa) + \gamma C_{Score}(\kappa)$ over all tuples in P_{all}, we find the semantic categories that best interpret the semantics of the regions in image I as the *Content* of I. We store the *Content*, T_{Score} and each region's *SCDB* codebook index as the final features for I. Note that for those regions whose codebook indices are invalid corresponding to semantic codebook, we will mark its semantic category as "UNKNOWN".

[4] In our experiments, we select $\beta = 10$.

4.3 Semantics Retrieval

Our approach supports both semantic keyword query and query-by-example. According to the submitted semantic keywords (corresponding to semantic categories), the system will first find out those images that contain all of the categories (denoted as set *RI*), then rank the documents by sorting the T_{Score} stored for each image in the database. For those images belonging to *RI* that has "UNKNOWN" category, its T_{Score} will be multiplied a diminishing factor. When the user submits the query-by-example, if the query image is in the database, its *Content* will be used as query keywords to perform the retrieval, otherwise it will first go through the above steps to obtain the *Content* and T_{Score}.

5 Experiments

We conducted experiments to compare the performance between our approach and traditional CBIR techniques including Keyblock [4], color histogram [10], color coherent vector [6]. The comparison is made by the precisions and recalls of each method on all the semantic categories. Following is the statistics of the 10 semantic categories we defined [5]:

Category	Explanation	No. in *db*
SKY	Sky (no sunset)	1323
WATER	Water	474
TREE_GRASS	Tree or Grass	778
FALLS_RIVER	Falls or Rivers	144
FLOWER	Flower	107
EARTH_ROCK _MOUNTAIN	Earth or Rocks or Mountain composed of	998
ICE_SNOW _MOUNTAIN	Ice or Snow or Mountain composed of	204
SUNSET	Sunset scene	619
NIGHT	Night scene	171
SHADING	Shading	1709

We used an image database with name *COREL* and size 3865. The *COREL* images can be considered as scenery and non-scenery. Scenery part has 2639 images consisting images containing the defined semantic categories, while non-scenery part has 1226 images including different kinds of textures, indoor, animals, molecules, etc. We choose 251 training images from scenery images as training images and form the semantic codebook with size 149. For each semantic category SC_i, we calculate and plot the precision-recall of our approach in the following way. Let *RETRIEVELIST* denote the images retrieved with SC_i. Suppose *RETRIEVELIST* has n images. We calculate the precisions and recalls of first $\frac{n}{30}$, $\frac{2n}{30}$, ... , and n images in *RETRIEVELIST*, respectively. The precisions and recalls are plotted to show the performance of our approach.

Since traditional CBIR approaches accepts only query-by-example, we have to solve the problem of comparing the approach of query-by-semantics with query-by-example. Let us take Keyblock as the example of traditional CBIR to show how we choose query sets and calculate the precision-recall for these methods. Suppose user submits a semantic keyword query of semantic category of *Sky*. There are total of 1323 images in *COREL* containing *Sky*. For each image containing *Sky*, we use it as a query on *COREL* to select top 100 images by Keyblock, and count the number of images containing *Sky* in the retrieved set. Then we sort the 1323 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% of *SKYLIST* as query set, denoted as *QUERYSET*. Then for each *COREL* image I, we calculate shortest distance to $QUERYSET - \{I\}$ by Keyblock [6]. The *COREL* images are sorted ascendantly by this distance. Top 1323 *COREL* images are retrieved and we calculate and plot the precision-recall of Keyblock on *Sky*, as we did for our approach.

The precision-recall for each semantic category is shown in Figure 5, 6 and the average precision-recall on all semantic categories is in Figure 4. We can see our method outperforms traditional approaches. In the experiments, traditional approaches benefit from the way of comparison in that only top 5% are selected as query set in their own way. Furthermore, the overlap of boundaries of different semantic categories degrades the our method's performance to some extent. Examples include "sunset" with "night", "sky" with "water" and "snowmountain" with "snowmountain-like cloud in the sky". Our approach performs still pretty well under such circumstances.

6 Conclusion

A new approach is proposed for semantically retrieving images based on content and context of image regions. By semantic codebook based on color-texture classification, images are segmented into regions, whose content and context are extracted and represented. The content and context of image regions provide a way to bridge the high-level semantics and low-level features. The experimental results show that our approach outperforms the traditional CBIR approaches we compared with.

[5] For now we only generate the semantic categories in the image database that scenery images consists a major percentage. We will certainly enroll more semantic categories into our system in the future.

[6] images in the *QUERYSET* will have distance zero to *QUERYSET*. Thus the query images will be automatically be retrieved as the top 5% images, which is unfair when making comparison.

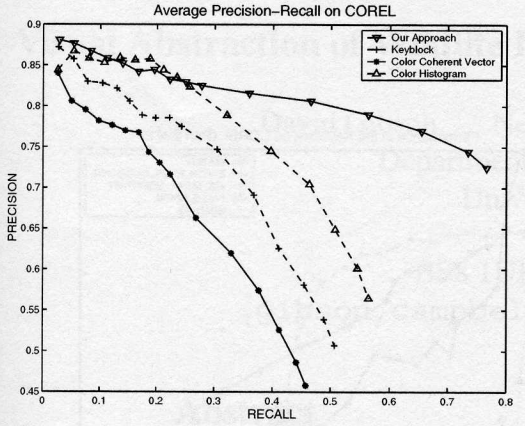
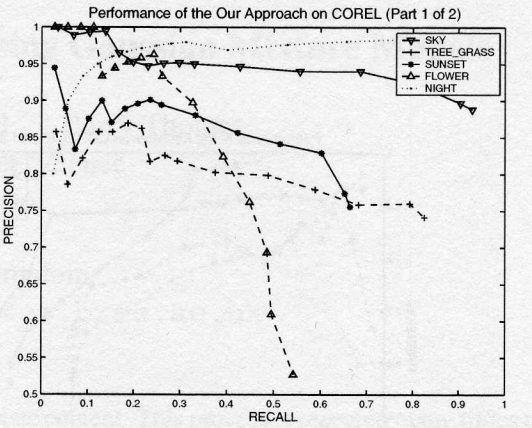
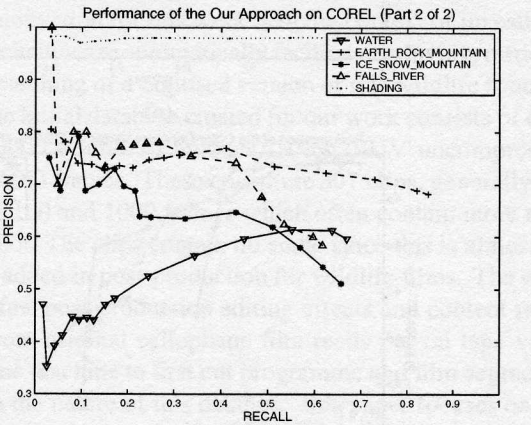


Figure 4: Average Precision-recall on all the semantic categories



(a)



(b)

Figure 5: Performance of (a) Our approach result 1, (b) Our approach result 2

References

[1] W. Forstner. A framework for low level feature extraction. In *Proc. Euro. Conf. Comp. Vis.*, pages pp 383–394, 1994.

[2] J. Garding and T. Lindeberg. Direct computation of shape cues using scale-adapted spatial derivative operators. In *Int. J. Comp. Vis.*, pages 17(2):163–191, 1996.

[3] Jia Li, J. Z. Wang and Gio Wiederhold. Integrated region matching for image retrieval. In *Proceedings of ACM Multimedia 2000*, pages 147–156, Los Angeles, California, USA, Oct 30 - Nov 3 2000.

[4] L. Zhu, A. Zhang, A. Rao and R. Srihari. Keyblock: An approach for content-based image retrieval. In *Proceedings of ACM Multimedia 2000*, pages 157–166, Los Angeles, California, USA, Oct 30 - Nov 3 2000.

[5] T. Leung and J. Malik. Detecting. localizing and grouping repeated scene elements from an image. In *Proc. Euro. Conf. Comp. Vis.*, pages pp 546–555, 1996.

[6] Greg Pass, Ramin Zabih, and Justin Miller. Comparing images using color coherence vectors. In *Proceedings of ACM Multimedia 96*, pages 65–73, Boston MA USA, 1996.

[7] J. Smith and C. Lee. Decoding image semantics using composite region templates. In *Proc. of the CVPR, volume Workshop on Content-Based Access of Image and Video Libraries*, 1998.

[8] Y. Song, W. Wang, and A. Zhang. Automatic annotation and retrieval of images. In *The Sixth IFIP Working Conference on Visual Database Systems*, 2002.

[9] Yuqing Song and Aidong Zhang. Monotonic tree. In *The 10th International Conference on Discrete Geometry for Computer Imagery, Bordeaux, France*, 2002.

[10] M.J. Swain and D. Ballard. Color Indexing. *Int Journal of Computer Vision*, 7(1):11–32, 1991.

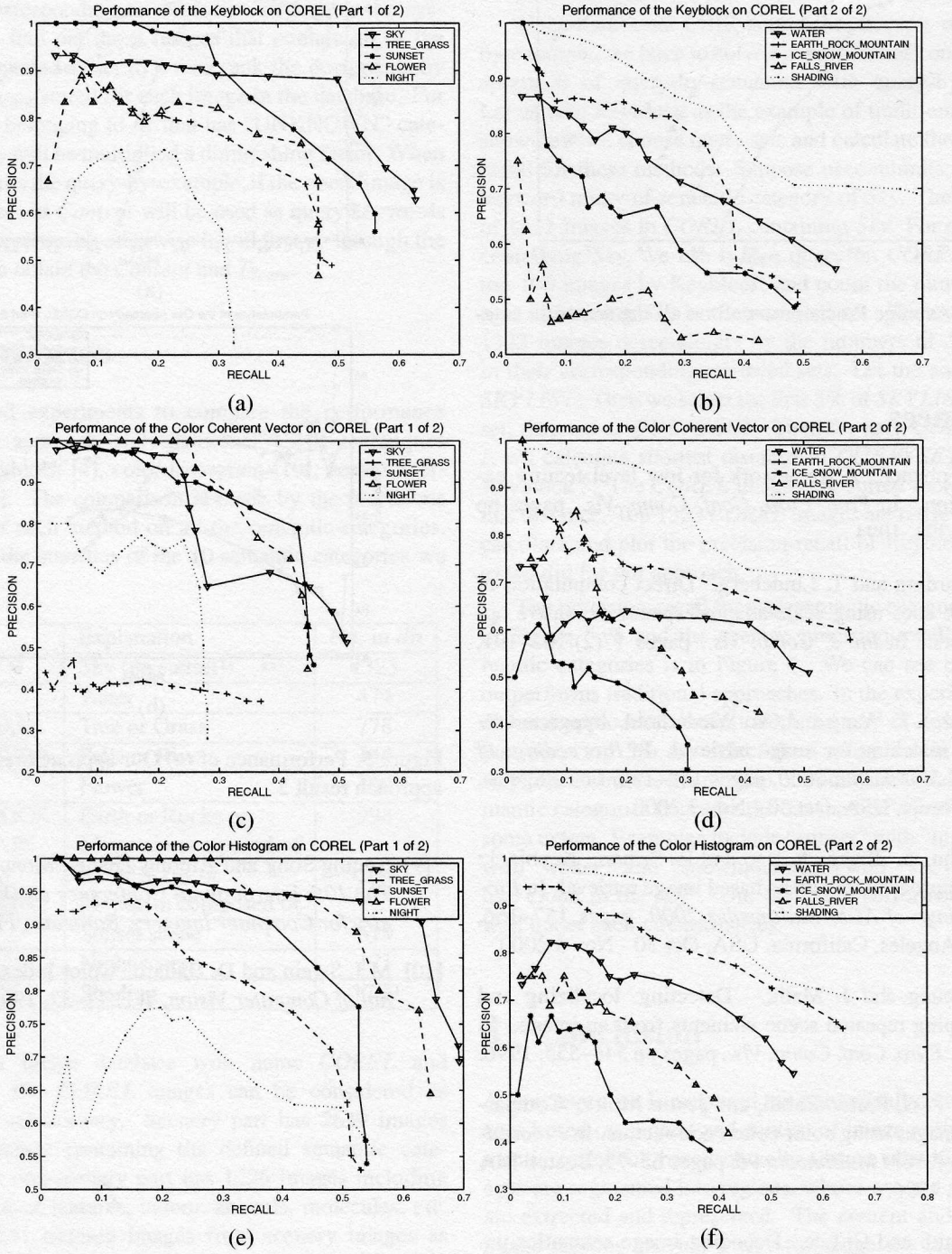


Figure 6: Performance of (a) Keyblock result 1, (b) Keyblock result 2, (c) Color Coherent Vector result 1, (d) Color Coherent Vector result 2, (e) Color Histogram result 1, (f) Color Histogram result 2