Cluster-driven Refinement for Content-based Digital Image Retrieval

Kyoung-Mi Lee (*)
Department of Computer Science
14 McLean Hall
The University of Iowa
Iowa City, IA 52242
TEL: 1-319-335-0972
E-mail: klee1@cs.uiowa.edu
Web: www.cs.uiowa.edu/˜klee1

W. Nick Street
Department of Management Sciences
S232 Pappajohn Business Building
The University of Iowa
Iowa City, IA 52242
1-319-335-1016
nick-street@uiowa.edu
dollar.biz.uiowa.edu/˜street

Abstract

Increasing application demands are pushing databases towards providing effective and efficient support for content-based retrieval over multimedia objects. In addition to adequate retrieval techniques, it is also important to enable some form of adaptation to users' specific needs. This paper introduces a new refinement method for retrieval based on the learning of the users' specific preferences. The proposed system indexes objects based on shape and groups them into a set of clusters, with each cluster represented by a prototype. Clustering constructs a taxonomy of objects by forming groups of closely-related objects. The proposed approach to learn the users' preferences is to refine corresponding clusters from objects provided by the users in the foreground, and to simultaneously adapt the database index in the background. Queries can be performed based solely on shape, or on a combination of shape with other features as color. Our experimental results show that the system successfully adapts queries into databases with only a small amount of feedback from the users. The quality of the returned results is superior to that of a color-based query, and continues to improve with further use.

EDICS: 4-DLIB, 4-KEEP

I. Introduction

Multimedia databases are convenient media for describing and storing image, text, voice, spatial, temporal, spectral, and physical components of information. There are several applications that need the ability to query these multimedia objects based on their content. The content of any multimedia object is represented by a collection of features that depend on the media type of the object. Typically, extracting the contents of objects from the media is a challenging goal. There is no general solution to the difficult task of content extraction, and successful solutions are limited to specific domains. There are two basic approaches, one text-based and the other based on visual information.

The popular most framework for the text-based approach was to annotate multimedia data by text [1], [2]. It caused two major difficulties, especially when the size of data collections is large. One is the vast amount of labor required in manual annotation. The other difficulty results from the rich content in the data and the subjectivity of human perception. The perception subjectivity and annotation impreciseness may cause unrecoverable mismatches in the later retrieval processes. To overcome these difficulties, content-based visual query techniques were proposed in the recent literature [3], [4], [5], [6], [7]. While these existing systems successfully establish the viability of the approach, techniques for incorporating human expertise directly during the query process to improve retrieval performance have drawn little attention.

A further consideration should be made regarding what it is that the user is looking for. By analyzing such relevance judgments, the system can then generate a new, refined query, which will likely improve the quality of the result, as experimental evidence confirms. In the information retrieval literature it has been well established that retrieval performance can be significantly improved by incorporating the user as part of the retrieval loop; this is called *relevance feedback*. The relevance feedback technique has been used in multimedia database annotation by choosing image groupings within and across images according to relevant and irrelevant judgments by the user [8]. While this process is valid in annotation, or database pre-processing before retrieval, it is not very suitable for a real-time retrieval process. Cox *et al.* viewed relevance feedback in a Bayesian framework that incorporates an explicit model of the user's selection process [9]. Nastar *et al.* introduced a relevance feedback system for query refinement using feature density estimation based on relevant and irrelevant images [10]. They estimate the distribution of relevant images and simultaneously minimize the probability of retrieving irrelevant images. Rui *et al.* examined how relevance feedback may be used to determine the appropriate features and similarity measures for retrieval, where the relevant and irrelevant examples submitted by the user's feedback are used to learn and refine the query by updating feature weights [11].

Relevance feedback allows the user to change the rank ordering of the images returned by a query. However, if a similar query is presented a second time, the system should retain previously-learned user preferences, thereby improving the efficiency and quality of the search. Furthermore, different users submitting identical queries might expect very different results. For example, given a query 'apple', some people could expect to look for fruits and others computers. Thus, our goal is to build an image indexing and retrieval system that learns user preferences.

To achieve this, we introduce the mechanism of cluster-driven refinement using user feedback. For indexing, the proposed system can cluster objects by template matching. Since the purpose of templates here is the representation of clustered objects, we also refer to them as *prototypes*. Our approach to user-oriented queries is to refine corresponding prototypes from objects provided by the user in the foreground, and to simultaneously adapt the database index in the background. Users specify their particular preferences by selecting and outlining relevant objects in the returned images. It is likely that different instances of feedback from a particular user will be related, so that over time, frequent users of the proposed intelligent retrieval system are able to tailor the system to their needs.

The rest of the report is organized as follows. Section 2 presents an overview of the object-based image retrieval system. We describe how we represent an image and an object in Section 3 and how index objects based on shape in Section 4. Section 5 describes the use of cluster-driven prototype refinement using an adaptive distance metric. The retrieval performance of the object-based query system on a sample image database is presented in Section 6. Section 7 concludes the paper.

II. Overview of System

Fig. 1 shows the functional overview of the proposed object-based image indexing and retrieval system which is implemented in Java. In this paper, we focus the use of shape for indexing because we are interested in object-based images which contain objects that have specific shapes. The proposed shape-based indexing is automatically performed off-line with no users' interaction (background processing), incorporating two queues which aid indexing of an ever-increasing volume of data.

The user presents a query, in the form of an object outline, from a local or remote file system. The system formulates the query, performing feature extraction by translating the object outline into a shape model. The extracted query feature is submitted to compute similarity between the query and prototypes first. After sorting the prototypes in terms of similarity, the system returns images containing objects indexed by the prototypes with high similarities. Based on the retrieved results, the user can supply her preferences by selecting relevant results and drawing outlines of the objects. Then the system incrementally learns her preferences by refining the prototypes. This approach, while slightly more intrusive than simple image selection, provides significantly more information on personal preferences to the system.

III. Image and Object Representation

Most existing content-based image retrieval techniques focus mainly on comparing images directly using their surface-level content such as colors, textures, shapes, and combinations of these. Although these methods work well for some types of image databases, for object-based images, similarities between images are most likely associated with interesting objects, $\mathbf{v}$, in the images [12]. The term object-based image is used to distinguish those images that contain objects that have specific shapes and colors, from other images that contain materials, such as beaches and grass, that appear as homogeneous patterns. Thus, an image $\mathbf{M}$ here is simply defined as (See Fig. 5):

$$\mathbf{M} = (N_M, \bigcup_{m=1}^{N_M} \mathbf{v}^m)$$

where N_M is the number of objects detected in $\mathbf{M}$.

For effective retrieval, a good representation for the extracted object shape is a necessary requirement. Also, for intelligent retrieval, features should be represented by some form, such as a feature vector, usable in a learning scheme. For example, feature points of shapes must be positioned consistently on each of the training examples such that a particular point always represents that same part of the shape on each example.

In order to model a shape of an object boundary scalar transform algorithms are used [13]. An object description $\mathbf{v}$ is simply a labeled set of I points $(v_i)_{i=1\cdots I}$. In this paper, we adopt the centroid-radii model to represent the shape by a set of points in polar coordinates [14], [15]. Assume that OA is an arbitrary radius of the shape. The algorithm starts from OA and moving clockwise, places I points around the boundary at the regular interval of $\frac{360}{I}$ degrees. So the shape is represented as

$$\mathbf{v} = (v_1, v_2, \cdots, v_I) = (q_{\theta_1}, q_{\theta_2}, \cdots, q_{\theta_I})$$

where q_{θ_i} is the i-th radius from the centroid to the boundary of the shape and $\theta_i = \left(\frac{360}{I}\right) i$. The reflected shape can be represented as: $\mathbf{v}' = (q_{\theta_I}, q_{\theta_{I-1}}, \cdots, q_{\theta_1})$.

This model assumes I is larger than 2. The six representations in Fig. 2 ((b)-(d) and (f)-(h)) show how this model can facilitate multiresolution representation. For example, while two objects in Fig. 2 (a) and (e) are retrieved as the similar shapes when I=4, they are dissimilar when I=8 or 16. We wish to use only as many points as necessary to adequately model the shape.

IV. Shape-based Indexing

A. Object Detection

For automatic shape-based indexing, the proposed system uses the Generalized Hough Transform (GHT) [16], a standard template matching algorithm, to detect the boundaries of objects in an image. GHT requires templates to be predefined and an array of points of the same size as the image, the accumulator, in which each point has a value, $G(x, y)$, specifying the possibility that the reference point of a shape to be detected is located at the point where $G(x, y)$ is a normalized value between 0 and 1. When the value of the point in the accumulator exceeds a certain threshold τ, then an object with the desired shape is said to be detected at the location of the point. Further, when the scale and orientation of an input shape are variant and unknown in advance, brute force is usually employed to enumerate all possible scales and orientations of the input shape in the GHT process. There have been several proposals to reduce the huge memory storage capacity and computational time required for GHT [17]. In this paper, we use an iterative approach to GHT (IGHT) proposed in [18] for memory efficiency.

For the proposed indexing, the system requires an initial definition of shapes the user wants to find. Fig. 3 shows 15 prototypes used in the first experiment. After getting the prototypes, the system performs automatic shape-based indexing with ALGORITHM I in the background.

Each image is first preprocessed to get a good edge image. Here we used a mean shift algorithm which iteratively shifts each data point to the average of data points in its neighborhood [19], [20]. Then the system applies IGHT with the prototype as a template to all images in an image queue. If the peak values in an IGHT accumulator are larger than τ, objects whose shapes look like the template are detected. The system saves each detected object (for example, two links to the corresponding image and prototype, the position, orientation angle, and scaling factor of the expected object, and the peak value as a mapping score) into the object database. Fig. 4 shows an example of the proposed shape-based indexing.

Whenever a prototype is added, it is placed the prototype queue, and all images are placed in the image queue. Whenever a new image is added the image database, the indexing procedure places it into the image queue and all prototypes into the prototype queue. All this processing is done in the background, creating an index of objects that is used for retrieval.

B. Index Tree

With tens of thousand of objects, it is not practical to sequentially compare each object in the database against the query. Since only a small number of objects is likely to match the query, a large number of unnecessary comparisons is being performed. As the comparison is expensive, we use a two-level index tree structure to provide an efficient means to reduce the number of comparisons.

The index tree shown in Fig. 5 consists of a prototype layer and an object layer. The prototype layer of the tree is a filter that reduces the search space quickly, discriminating the objects. In this paper the prototype consists shape features such as these in Fig. 3. These prototypes are used as an indexed key of the two-level indexed tree. This structure is easily extensible by adding a new layer to the index structure for a new feature, such color. Each layer can be constructed using any appropriate indexing mechanism. For example, the one-dimensional structure for color can be replaced by a three-layer tree using 3 color components. By dealing with subkeys (smaller number of dimensions), any path that differs on any subkey can be pruned away immediately, resulting in a more efficient search. More importantly, using a smaller number of dimensions can avoid performance degeneration that arises for high-dimensional data. In addition, the tree facilitates similarity retrieval. By stopping the search process at a certain level, all objects under that level are retrieved. The lower the level at which the search process terminates, the more precise the description of the objects.

V. Cluster-driven Prototype Refinement

A. Motivations

A key motivation for our work is that users have a particular context within which their queries are formed. It is likely that consecutive queries for a particular user will be related in that they will be part of the same context. Therefore, frequent users of an image retrieval system should be able to benefit from their high use of the system. In this paper, we want to use relevance feedback to give them user more control on the search criteria. We also want to allow the database system to learn the users' context. When the user looks for an object similar to one found previously, the system can give results using his context learned by feedback, instead of re-starting the search from scratch.

Our motivation is to store this feedback and avoid redundant and unnecessary steps for search in the personalization service. Fig. 6 (a) illustrates a case where there are four prototypes in a two-dimensional feature space. Many existing systems use user feedback to adjust the desired query. This process of query refinement is illustrated in Fig. 6 (b). For query $\mathbf{q}$, query refinement modifies the query utilizing user feedback provided by the user to better meet the user's need. Without prototype refinement, when the user enters the same query again, the system would need to again perform query refinement. Instead, Fig. 6 (c) shows how the prototypes are adapted to

$\mathbf{q}$ and thus the system can start the second search with those adapted prototypes.

B. Incremental Clustering

Clustering is used for forming groups of closely-related objects and for accelerating query processing by considering only a small number of representatives of the clusters, rather than the entire data set. However, existing clustering algorithms are not suitable to update clusters without frequently performing complete reclustering in such a dynamic environment. Over time, as additional indexing objects become available, the database should have the capability to adapt to the new data while re-indexing what it has already learned. Therefore, instead of re-indexing the database to the entire data set every time new samples are collected, it is more desirable to incrementally index the database to the new data.

We use an incremental learning method which enables the user to refine prototypes by specifying over time a set of relevant results. Mathematically, a given set of N objects, $\{\mathbf{v}^1, \mathbf{v}^2, \cdots, \mathbf{v}^N\}$, will be partitioned into clusters such that data within the same cluster have a high degree of similarity, while shapes belonging to different clusters have a high degree of dissimilarity.

Each of these clusters is represented by a prototype $\mathbf{P}^c$. Suppose N_c is the number of objects in the cluster $\mathbf{c}$. The ith feature of the prototype, P_i^c, can be computed by averaging the features that belong to the cluster $\mathbf{c}$. To measure the spread of a set of data around the center of the data in the cluster, the standard deviation (σ), the most commonly used measure of a width of the center in statistical practice when the mean is the measure of center, is used. Incrementally, whenever a new object $\mathbf{v}$ is most similar to an existing clister $\mathbf{c}$, the object is assigned to the cluster $\mathbf{c}$ and P_i^c and σ_i^c are updated as follows:

$$P_i^{c'} = \frac{N_c P_i^c + v_i}{N_c'} \text{ and } \sigma_i^{c'} = \sqrt{\frac{N_c s_i + (P_i^{c'} - v_i)^2}{N_c'}}, \tag{1}$$

where $N_c' = N_c + 1$ and $s_i = (\sigma_i^c)^2 + (P_i^{c'} - P_i^c)^2$. During updating, shapes are normalized for translation through the use of an object-centered coordinate system and for reflection through the use of the reflected shape. Shapes are normalized for size by scaling based on the longest radius. To avoid simple orientation differences, the comparison is performed at different rotations and the final distance is defined to be the minimum of these distances. If the new object is not similar to any of the existing clusters, a new cluster is created with the new object and the number of clusters is increased.

C. Weighted Distance Metric

The most common measure of similarity between two objects, $\mathbf{v}$ and $\mathbf{u}$, is the distance measure that is generally taken to be the weighted Euclidean norm as

$$\mathbf{d}(\mathbf{v},\mathbf{u}) = \sqrt{\sum_{i=1}^{n} \omega |v_i - u_i|^2}. \tag{2}$$

Since each dimension feature has a different influence, each feature should be rescaled so that all features contribute equally to the distance computation in Eq. (2) [21].

In Fig. 6 (a), for query $\mathbf{q}$, dimension X is more relevant, because a slight move along the X axis may change the cluster. To capture such scale information as weights, we incorporate a measure using the relative distance based on the standard deviation [21]. The weight for a feature P_i^c is defined as

$$\omega_i^c(\mathbf{q}) = \begin{cases} 1 - \exp\left(-\dfrac{|P_i^c - q_i|}{\sigma_i^c}\right), & \text{if } \sigma_i^c \neq 0 \\ \\ 1, & \text{otherwise.} \end{cases} \tag{3}$$

It is evident that $0 \leq \omega_i^c \leq 1$, where $\omega_i^c = 1$ indicates that q_i is too far from P_i^c or σ_i^c is 0, and thus q_i isn't similar to P_i^c at all. On the other hand, $\omega_i^c = 0$ states P_i^c and q_i are identical, and so q_i is totally similar to P_i^c. Values in between show the degrees of weights that q_i exerts at i. For the query $\mathbf{q}$, it reflects the influence of $\mathbf{q}$ on the variation of $\mathbf{P}^c$ at i. These weights enable the distance computation to elongate dissimilar features, and, at the same time, to constrict the closest ones. Note that the technique is shape-based because weights depend on the shape

[22]. To avoid simple orientation differences in the shape features, the distance is computed at different rotation angle and the final distance from the query to prototype $\mathbf{P}^c$ is defined to be the minimum of these distances.

D. Retrieval and Refinement

In retrieval, given a query $\mathbf{q}$ created by drawing an object on a sketch or on an image, selection of objects sharing similar characterizing features starts by comparing $\mathbf{q}$ to the set of prototypes. Using the weighted distance metric (Section V-C), the system first computes the similarity between the prototypes and $\mathbf{q}$ and sorts the prototypes based on this metric. Then, using the peak value $G(x, y)$ of each object, the system can compute the final distance between $\mathbf{q}$ and each object matching the prototypes with high similarity as

$$\mathbf{D} = \mathbf{d}_\omega(\mathbf{P}^c, \mathbf{q}) + \alpha \frac{1}{\mathrm{G}(x, y)}.$$

where G(x,y) is greater than the predefined threshold value ($\tau > 0$) and α is a constant for normalization. If τ is 0.7, $\frac{1}{\mathrm{G}(x,y)}$ is between 1 and 1.43 and is experimentally smaller than $\mathbf{d}_\omega(\mathbf{P}^c, \mathbf{q})$. In our experiments, we set α as $\tau \times n$. If multiple features are integrated, the combined distance between a query $\mathbf{q}$ and an object $\mathbf{v}$ should be computed. For instance, with the color features, $\mathbf{q}_{color}$ and $\mathbf{v}_{color}$, the combined distance is defined as

$$\mathbf{d}(\mathbf{q}, \mathbf{v}) = \omega_s \mathbf{D} + \omega_l \mathbf{d}(\mathbf{q}_{color}, \mathbf{v}_{color}), \tag{4}$$

where ω_s and ω_l are weight factors of shape features and color features, respectively. In this paper, ω_s and ω_l are defined as

$$\omega_s = \frac{1}{\text{number of shape features}} \text{ and } \omega_l = \frac{1}{\text{number of color features}}.$$

$\mathbf{d}(\mathbf{q}_{color}, \mathbf{v}_{color})$ can be calculated by the Manhattan distance between $\mathbf{q}$ and $\mathbf{v}$ based on color features (in this paper, a color histogram).

The proposed refinement algorithm is described in ALGORITHM II. At each step, the user chooses the retrieved objects as relevant or irrelevant to $\mathbf{q}$. The system has to estimate user relevance feedback, i.e., for each retrieved object, decide its relevance or irrelevance to the query. The user also has to draw boundaries of the relevant objects. These drawn boundaries are then clustered incrementally (Section V-B). If the boundary is similar to one of the existing prototypes, the prototype is updated by averaging (Fig. 7 (b)). On such an updated prototype, similarities between $\mathbf{P}^c$ and $\mathbf{q}$ and between $\mathbf{P}^c$ and each object are changed. However, it is not appropriate to perform IGHT and compute a new peak value $G(x, y)$ for an on-line retrieval system. Therefore we recompute only $\mathbf{d}_w(\mathbf{P}^c, \mathbf{q})$ during on-line retrieval, while the system recomputes the peak value by calculating the distance between the updated prototype and drawn boundaries of relevant objects in off-line (Fig. 7 (c)). Also the updated prototypes are periodically put into the prototype queue for re-indexing. Since the prototype moves toward the query, the system can get better result when an identical or similar query is presented a second time. If the feedback shape is not similar to any existing prototype, the system creates a new prototype by averaging the drawn boundary and the query, and puts it into the prototype queue (Fig. 7 (d)). Since a new prototype is presented, the system performs the indexing algorithm ALGORITHM I in the background (Fig. 7 (e)).

VI. Experimental Results

A. Experimental setup

In these experiments, we use two image databases. The first database is the well-known Columbia database which contains 1440 gray images of 20 different objects: 72 images per object taken at 5^o in pose [23]. The second database consists of 127 color images collected from the Internet, divided into five different object classes. For each object, the system maintains two features; shape and color. Both databases use a 36-dimensional shape feature (SQ). For the

Columbia database, the system uses a 64 gray-scale color histogram inside the boundary of the shape (CQ). For experiments on the color image collection, the system uses three query methods for color: a pixel-by-pixel comparison (Pixel), a color histogram (CQ) and a color coherence vector (CCV) [24]. In the pixel-by-pixel comparison, the system first quantizes the color space into representative colors. After resizing all images in the database to the size of the given query image, the system sums differences of pixel values in the quantized color space. The color histogram is computed by looking up the 64 representative colors by clustering in LUV space. The color coherence vector method is a color histogram refinement scheme that divides each bin into coherent and incoherent pixels. For shape-based indexing, we created 15 prototypes for the Columbia database (see Fig. 3) and 10 prototypes for the color image collection by drawing boundaries of representative objects.

The performance in the experiments is evaluated using two measures: precision and recall. Recall is the ratio of relevant images retrieved to the total number in the database; it measures the ability of a system to present all relevant images. Precision is the ratio of relevant images retrieved to the total number of images retrieved; it measures the ability of a system to present only relevant images. For most applications, it is impossible to maximize precision and recall simultaneously, but these values should ideally be as large as possible.

The time complexity of indexing varies depending on the number of templates applied, the number of edge points in an image, the number of rotations and scaling factors examined, and the number of shape features. In this paper, we used 16 rotations with reflective invariance and 10 scaling factors. To index an image with a single prototype, the average time for the Columbia database is 11.7 secs and that for the color image collection is 9.3 secs. While the total time for indexing all images with all prototypes (i.e., system initialization) can be one or two days, our interest in this paper is faster and more intelligent retrieval and we achieved the average retrieval time of 1.4 secs.

B. Results

For retrieval, we use three kinds of queries: color-based query (CQ), shape-based query (SQ), and shape-color-based query (SCQ) which is a combination of shape and the color histogram. The proposed system first finds the most similar prototypes and then ranks the shapes in these clusters. Fig. 8 shows the average result of 20 queries on the Columbia database. On the shape-based query, the precision and recall continue to improve with prototype refinement and thus the system works more accurately as it learns the shapes. On Columbia database, the shape query gets comparable or better results compared to the color queries after the first refinement (Fig. 8 (a)). The system also includes color features in the distance metric with the hope of improving performance. The results of the combined shape and color query are slightly better than the results with only shape features (Fig. 8 (b)). Adding the color features can correct some of the errors in shape-based indexing, caused by inexact image analysis.

Fig. 9 shows a sample result of the shape-based query drawn by a user (the upper left sketch in Fig. 9 (b)) in which the result is significantly better than the color-based query. This result can be further improved by adding color features. Due to the strong skewed line, the object at the bottom right in Fig. 9 (b) is retrieved by the query. However, the object at the bottom right in Fig. 9 (c) has the correct coloring, which in this case outweighs the effect of the shape.

Fig. 10 shows that prototype refinement improves the retrieval result of a query (the upper left image) on the Columbia database. The user selected and drew what she wanted to retrieve (white boundaries) in Fig. 10 (a). Fig. 10 (b) shows the result of on-line prototype refinement, which modified a set of prototypes and calculated the similarity between the query and prototypes. In order to take into account only the feedback effect, the system eliminates the shapes that have already been learned by the user in the first retrieval. An important aspect of this paper is that the proposed system returns Fig. 10 (c) when the user submits the same query after shape-based indexing based on the cluster-driven preference learning in the background, instead of starting again with Fig. 10 (a).

Fig. 11 shows the average result of 10 queries selected from the color image collection. This color image collection contains different objects, instead of different viewpoints of the same object. The collection was created based on the objects that appear in the image, but shape-based indexing suffers from the difficulty of finding a clear edge boundary for objects in natural images. While three color queries (Pixel, CQ, and CCV) have strong retrieval at low levels of recall, and then drop quickly as the recall increases, a shape query has relatively lower, but more stable retrieval at all levels of recall. These results of the shape-based queries again improve with the prototype refinements (Fig. 11(a)). In other words, a color-based query is better at finding a smaller number of exact matches, and a shape-based query with prototype refinements is better at finding a larger number of similar matches. Also, adding the color histogram (Fig. 11(b)) increases precision somewhat, by 0.04 on average. Note that the drop at recall 0.2 occurs because of objects that are close to the best-matching prototype, but actually distant from the query.

Fig. 12 shows a sample result of the butterfly (the upper left image) using three color-based queries. With these color-based queries, the system can find the exact matches at low levels of recall (Fig. 12). Fig. 13 presents results of a shape query drawn by user (white boundaries). With the drawn outline, the system finds the similar prototypes: two-wing butterfly, elliptical-shaped fish, sailboat, and so on. The objects in these clusters are sorted by their peak values and the similarity between their prototypes and the query (Fig. 13 (a)). When a color histogram is added to the shape query, the result can be further improved. For example, since images containing sailboats have lots of blue pixels which the query, a black butterfly on the rock (the upper left image) doesn't have, the similarities of those images go down relative to the others and they get lower ranks (Fig. 14 (a)). These results improve much more with prototype refinement. Fig. 13 (b) and Fig. 14 (b) show the results with one off-line prototype refinement, after the user outlined relevant objects (white boundaries in Fig. 13(a) and Fig. 14 (a)).

VII. Conclusions

In the past few years, many retrieval approaches based on extracting and representing visual properties of multimedia data have been developed. While these approaches establish the viability of multimedia information retrieval based on visual features, techniques for incorporating human expertise to improve retrieval performance have not been studied. To address this limitation, this paper presents cluster-driven refinement from user feedback for an intelligent image retrieval system. The proposed learning algorithm is based on an incremental clustering using shape features. The following summarizes the contribution of this paper.

• Weighted distance: Originally, the use of Euclidean distance assumes that all features are equally important. Although this is true in some cases, in many other cases this assumption is known to be false, so this could yield poor results. In this paper, we use a weighted distance metric based on both the clustered database and on relevance feedback.

• A flexible query model that can easily incorporate new features

• Intelligent retrieval via prototype refinement both on-line and off-line learning: To retrieve more relevant images, relevance feedback has been studies for years in information retrieval. In this paper, we propose a cluster-driven refinement scheme that adapts the database indexing mechanism to the user's preferences. This provides an intelligent guide to find what the users want based on their previous decisions, and so serves as an automatic personalization service.

• Learning of new shapes during system use: By incorporating newly encountered shapes into the indexing mechanism, the system removes the need for pre-defining all required shapes.

Due to inherent image analysis problems, it is very difficult to find a general approach to extract contents from image collections and to index them. Further, using a template matching method requires significant computing time. So, in its current state, our approach is more appropriate for moderately-sized collections of similar images, or images from a specific domain, such as a medical database. For example, brain tumor applications are a good target of shape-based indexing. (Semi)automatic detection and segmentation techniques can extract tumors from medical images like MRI, and extracted tumors can be translated to an appropriate shape model for indexing. We are currently implementing such a system, which will provide similarity-based queries on tumor shapes.

We are currently extending our work in several directions. For instance, we are exploring ways to incorporate irrelevance feedback into prototype refinement. We are also implementing a neural network based on using unsupervised learning to cluster shapes and supervised learning to classify and retrieve objects. Finally, the adaptive nature of our approach would seem appropriate for a multiuser, "collaborative indexing" environment. This could also be enhanced by the use of named queries, e.g., a "penguin" query submitted by one user would be useful to other users.

VIII. Acknowledgments

This work was partially supported by NIH grant CA64339-04 and NSF grant IIS-99-96044. The authors would like to thank David Eichmann for helpful suggestions of experiments.

References

[1] P. G. B. Enser, "Query analysis in a visual information retrieval context," *Journal of Document and Text Management*, vol. 1, no. 1, pp. 25–52, 1993.

[2] L. H. Keister, *Challenges in Indexing Electoric Text and Images*, chapter User Types and Queries: Impact on Image Access Systems, pp. 7–22, American Society for Information Science, 1994.

[3] M. De Marsicoi, L. Cinque, and S. Levialdi, "Indexing pictorial documents by their content: A survey of current techniques," *Image and Vision Computing*, vol. 15, pp. 119–141, 1997.

[4] Y. A. Aslandogan and C. T. Yu, "Techniques and systems for image and video retrieval," *IEEE Transactions on Knowledge and Data Engineering*, vol. 11, no. 1, pp. 56–63, 1999.

[5] Y. Rui, T. S. Huang, and S.-F. Chang, "Image retrieval: Current techniques, promising directions and open issues," *Journal of Visual Communication and Image Representation*, vol. 10, pp. 1–23, 1999.

[6] A. Yoshitaka and T. Ichikawa, "A survey on content-based retrieval for multimedia databases," *IEEE Transactions on Knowledge and Data Engineering*, vol. 11, no. 1, pp. 81–93, 1999.

[7] A. W.M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain, "Content-based image retrieval at the end of the early years," *IEEE Transactions on Pattern Analyis and Machine Intelligence*, vol. 22, no. 12, pp. 1349–1380, 2000.

[8] T. P. Minka and R. W. Picard, "Interactive learning using a society of models," *Pattern Recognition*, vol. 30, no. 4, pp. 565–581, 1997.

[9] I.J. Cox, M.L. Miller, T.P. Minka, T. V. Papathomas, and P.N. Yianilo, "The Bayesian image retrieval system, PicHunter: Theory, implementation and psychological experiments," *IEEE transactions on Image Processing*, vol. 9, no. 1, January 2000.

[10] C. Nastar, M. Mitschke, and C. Meihac, "Efficient query refinement for image retrieval," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1998, pp. 547–552.

[11] Y. Rui, T. S. Huang, and Sharad Mehrotra, "Relevance feedback techniques in interactive content-based image retrieval," in *Proceedings of IS&T/SPIE Storage and Retrieval of Image and Video Databases IV*, 1998, pp. 25–36.

[12] I. Roth and V. Bruce, *Perception and Representation: Current Issues*, Open University Press, 2nd edition, 1995.

[13] S. Loncaric, "A survey of shape analysis techniques," *Pattern Recognition*, vol. 31, no. 8, pp. 983–1001, 1998.

[14] K.-M. Lee and W. N. Street, "Automatic segmentation and classification using on-line shape learning," in *Proceedings of the 5th IEEE Workshop on the Application of Computer Vision*, 2000, pp. 64–70.

[15] K.-L. Tan, B. C. Ooi, and L. F. Thiang, "Indexing shapes in image databases using the centroid-radii model," *Data and Knowledge Engineering*, vol. 32, no. 3, pp. 271–289, 2000.

[16] D. H. Ballard, "Generalizing the Hough transform to detect arbitrary shapes," *Pattern Recognition*, vol. 13, no. 2, pp. 111–122, 1981.

[17] A. A. Kassim, T. Tan, and K. H. Tan, "A comparative study of efficient generalised Hough transform techniques," *Image and Vision Computing*, vol. 17, no. 10, pp. 737–748, 1999.

[18] K.-M. Lee and W. N. Street, "Model-based detection, segmentation and classification for image analysis using on-line shape learning," *Machine Vision and Applications, in press.*

[19] Y. Cheng, "Mean shift, mode seeking and clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 8, pp. 790–799, 1995.

[20] D. Comaniciu and P. Meer, "Mean shift analysis and applications," in *Proceedings of the IEEE Conference on Computer Vision*, 1999, pp. 1197–1203.

[21] K.-M. Lee and W. N. Street, "Incremental feature weight learning and its application to a shape-based query system," *Pattern Recognition Letters, in press.*

[22] C. Atkeson, A. W. Moore, and S. Schaal, "Locally weighted learning," *AI Review*, vol. 11, pp. 11–73, 1997.

[23] S. A. Nene, S. K. Nayar, and H. Murase, "Columbia object image library (COIL-20)," Tech. Rep. CUCS-005-96, Columbia University, 1996, http://www.cs.columbia.edu/CAVE/research/softlib/coil-20.html.

[24] G. Pass and R. Zabih, "Comparing images using joint histograms," *ACM Journal of Multimedia Systems*, vol. 7, no. 3, pp. 234–240, 1999.

```
Dequeue a prototype, $\mathbf{P}$, as a template, in the prototype queue
For each image, $\mathbf{M}$, in the image queue
    Preprocess the image
    Perform IGHT: $G(x,y) = \text{IGHT}(\mathbf{M},\mathbf{P})$
    Find peak points $(x,y)$ whose values $G(x,y)$ are larger than the predefined threshold.
    For each peak point $(x,y)$,
        Store a new object $\mathbf{v}$ into the object database:
            $((x,y)$, $G(x,y)$, orientation angle, scaling factor, $\mathbf{M}$, $\mathbf{P})$
```

ALGORITHM I. Automatic shape-based indexing

```
(1) Initialization:
    $k = 0$, $N_{c_0} = N_c$, $\mathbf{P}^{c_0} = \mathbf{P}^c$, and $\sigma^{c_0} = \sigma^c$,
    where $k = 0$ means there is no relevance feedback to $\mathbf{q}$.
(2) Retrieval:
    Calculate $\omega_i^{c_k}(\mathbf{q})$ using Eq. (3).
    Search similar prototypes using $\mathbf{d}_\omega(\mathbf{P}^c, \mathbf{q}) = \sqrt{\sum_{i=1}^{n} |\omega_i^c(\mathbf{q})(P_i^c - q_i)|^2}$.
    Retrieve relevant objects in the above similar prototypes $\mathbf{P}^{c_k}$ using Eq. (4).
(3) User relevance feedback:
    Select relevant objects from retrieved results.
    Draw the boundaries of the selected objects.
(4) On-line prototype refinement using incremental clustering:
    Update the matched prototype using Eq. (1) or create a new prototype.
(5) $k = k + 1$ and go to step (2).
(6) Off-line prototype refinement:
    Recompute the peak value between the updated prototype and objects in the prototype or
    run ALGORITHM I with the created prototype in the background.
```

ALGORITHM II. Prototype refinement

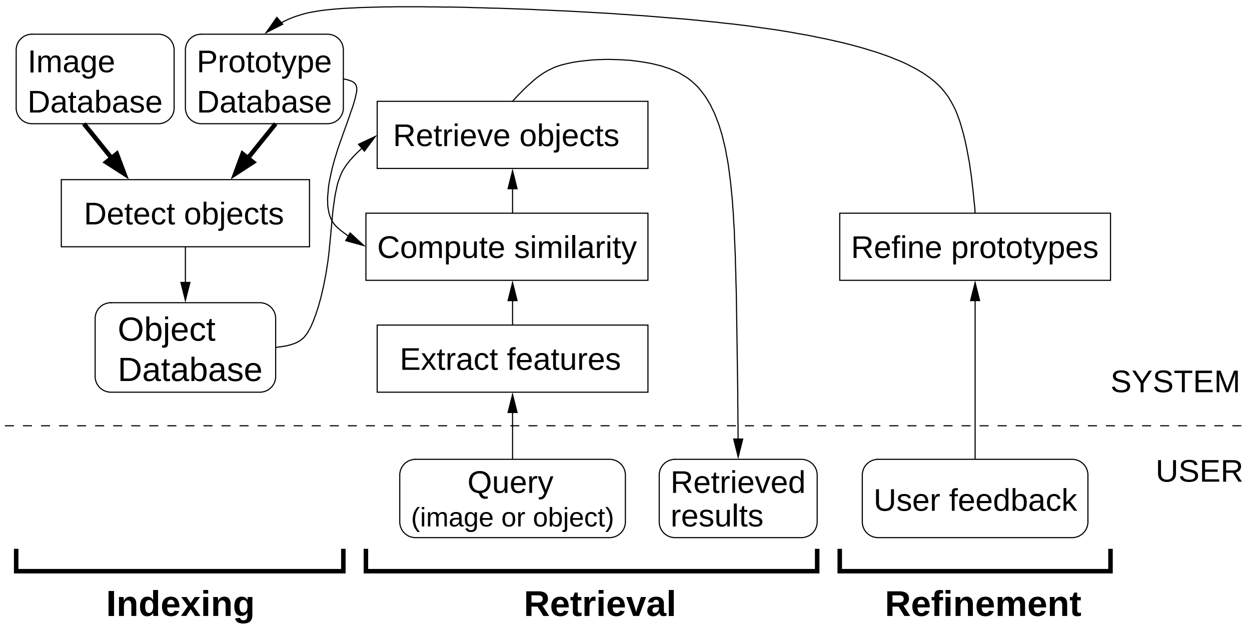
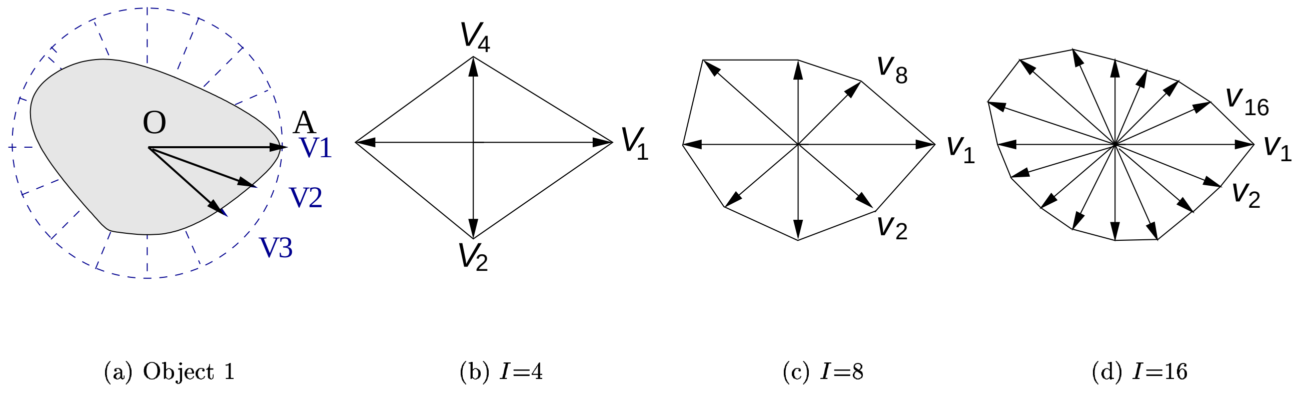


Fig. 1. System overview: box = process, rounded box = data or database, thin arrow = data flow, and thick arrow = queue



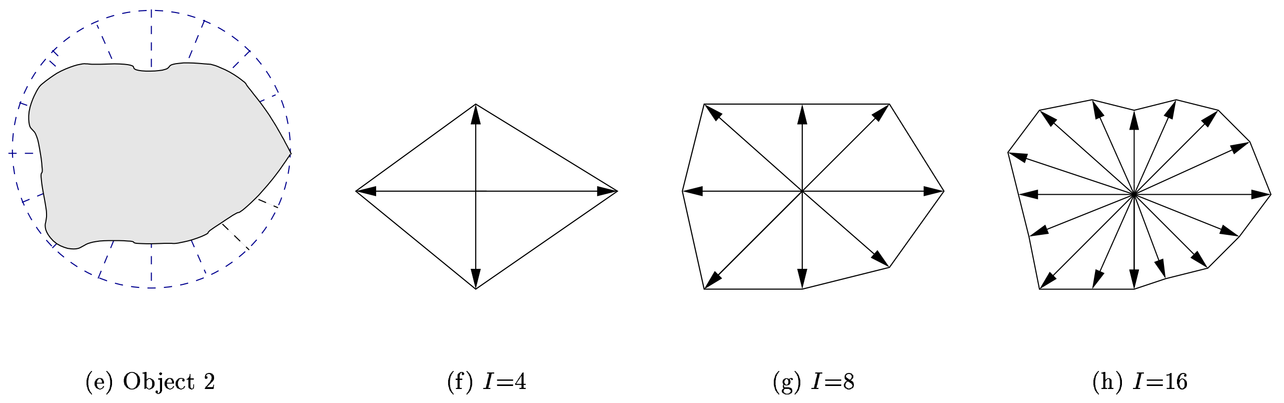


Fig. 2. The centroid-radii model of a shape query using multiresolution: when I=4, two objects are retrieved as similar shapes, but when I is 8 or 16, they are dissimilar shapes

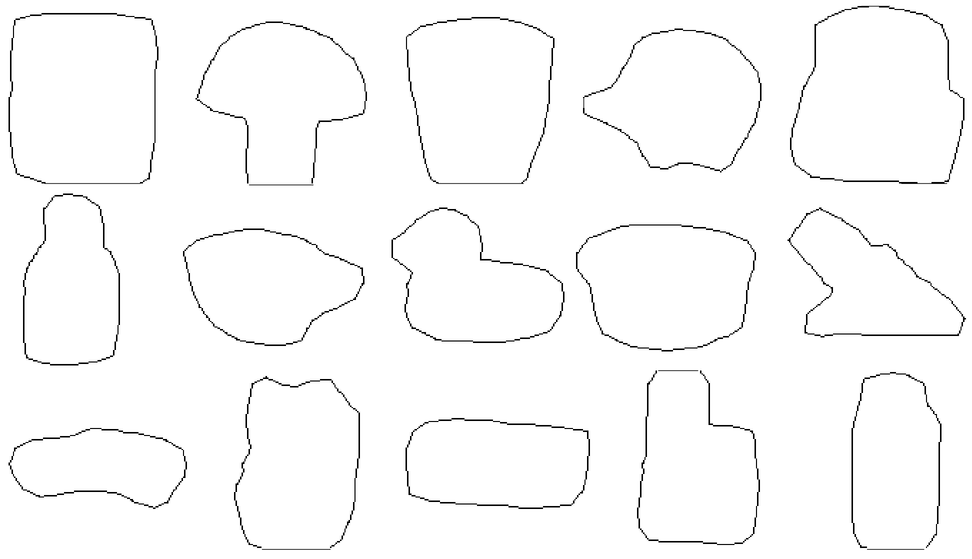


Fig. 3. 15 initial prototypes (Columbia database)

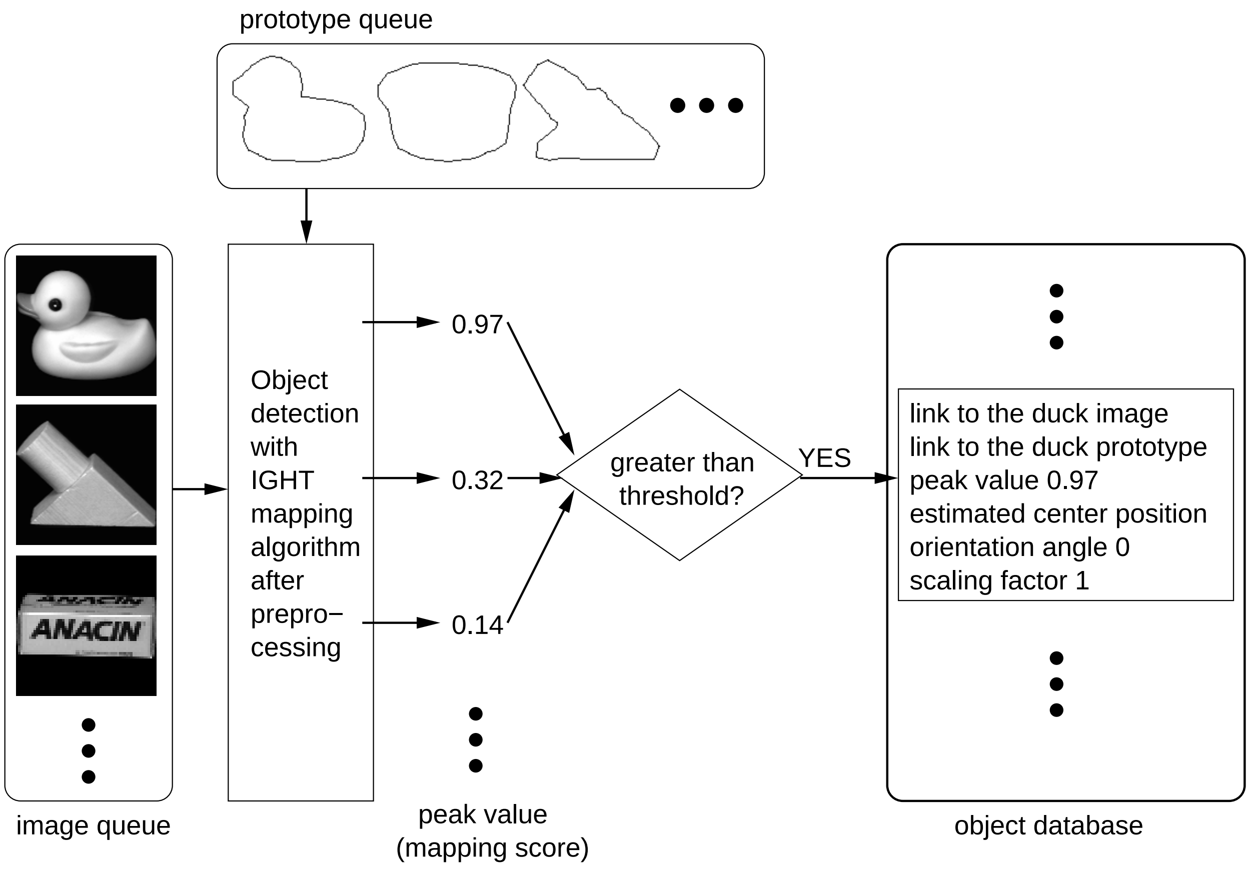


Fig. 4. An example of shape-based indexing

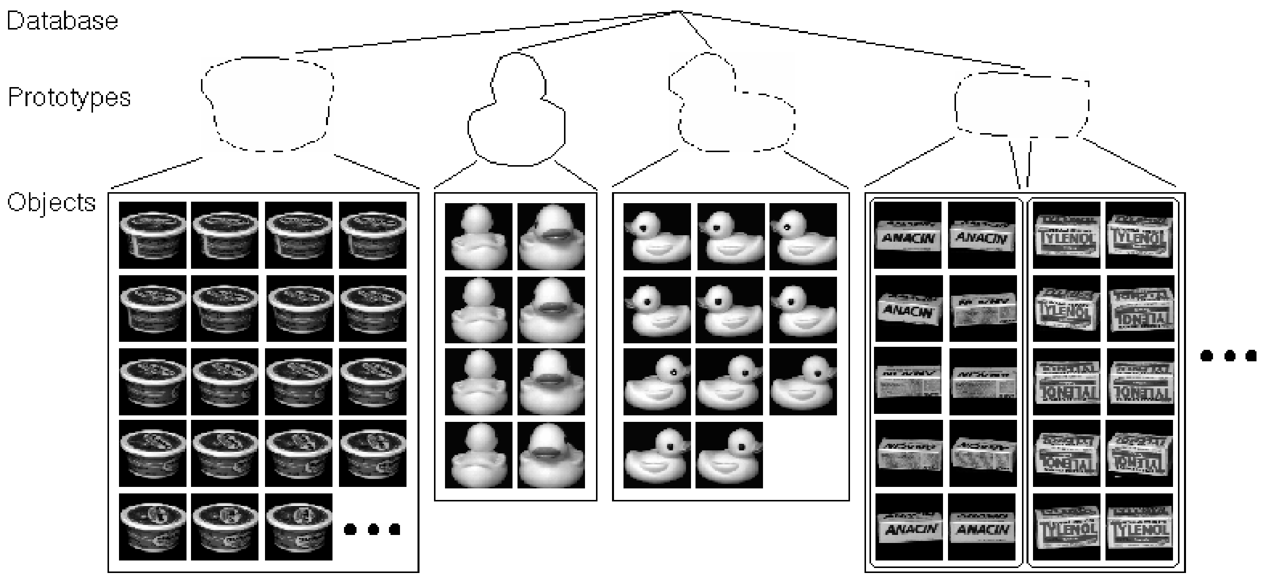


Fig. 5. Index tree of the object-based database

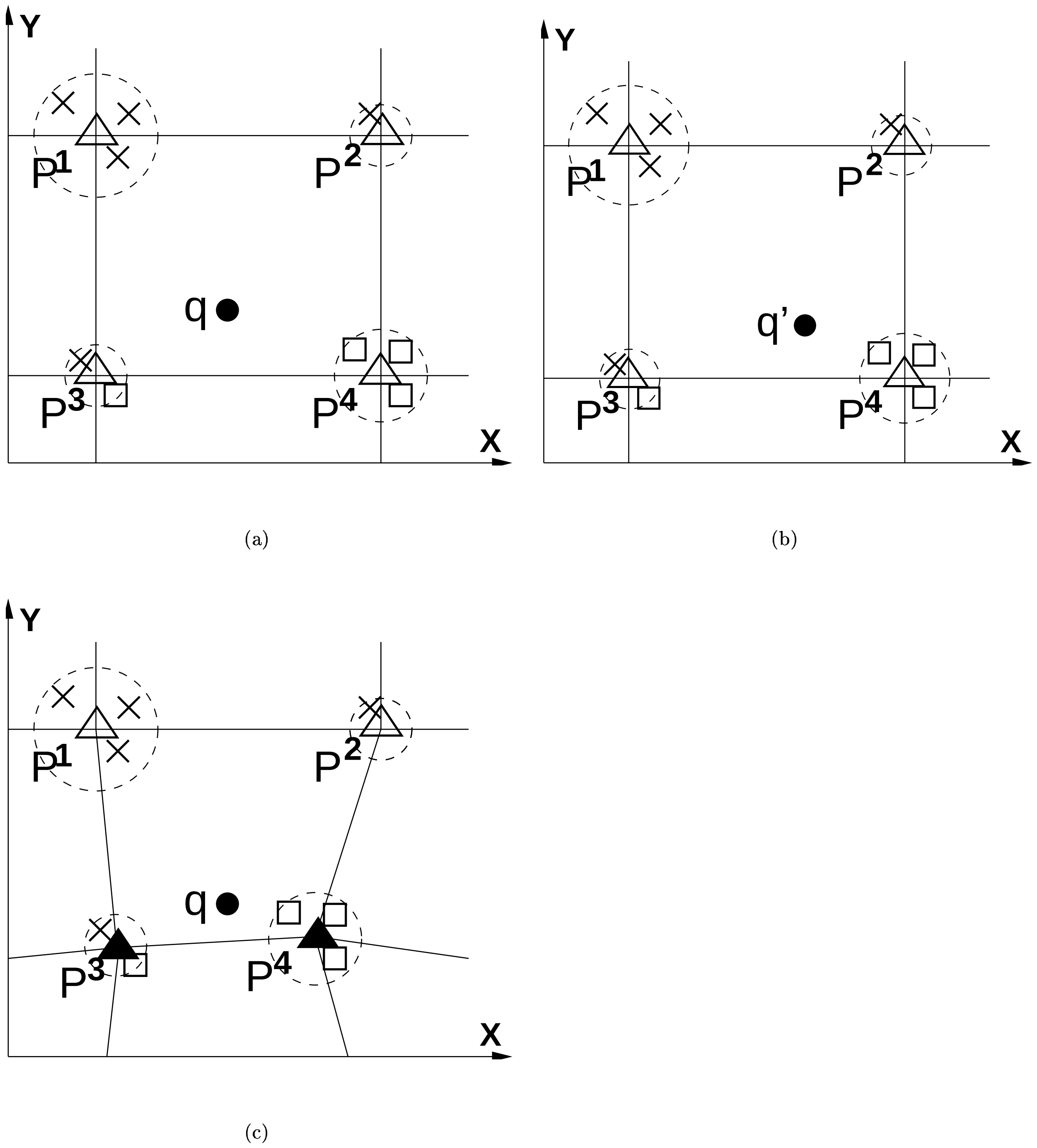


Fig. 6. Relevance feedback gives more control over the search criteria. In this and the following figures, a black circle means a query **q**, a triangle a prototype of the corresponding cluster (dotted lines), a black triangle an updated prototype, a square a relevant result, and a cross an irrelevant result. (a) Query and 4 prototypes $\mathbf{P}^1$, $\mathbf{P}^2$, $\mathbf{P}^3$ and $\mathbf{P}^4$, (b) Query refinement where **q**' is a refined query, and (c) Prototype refinement where $\mathbf{P}^3$ and $\mathbf{P}^4$ are updated based on user feedback.

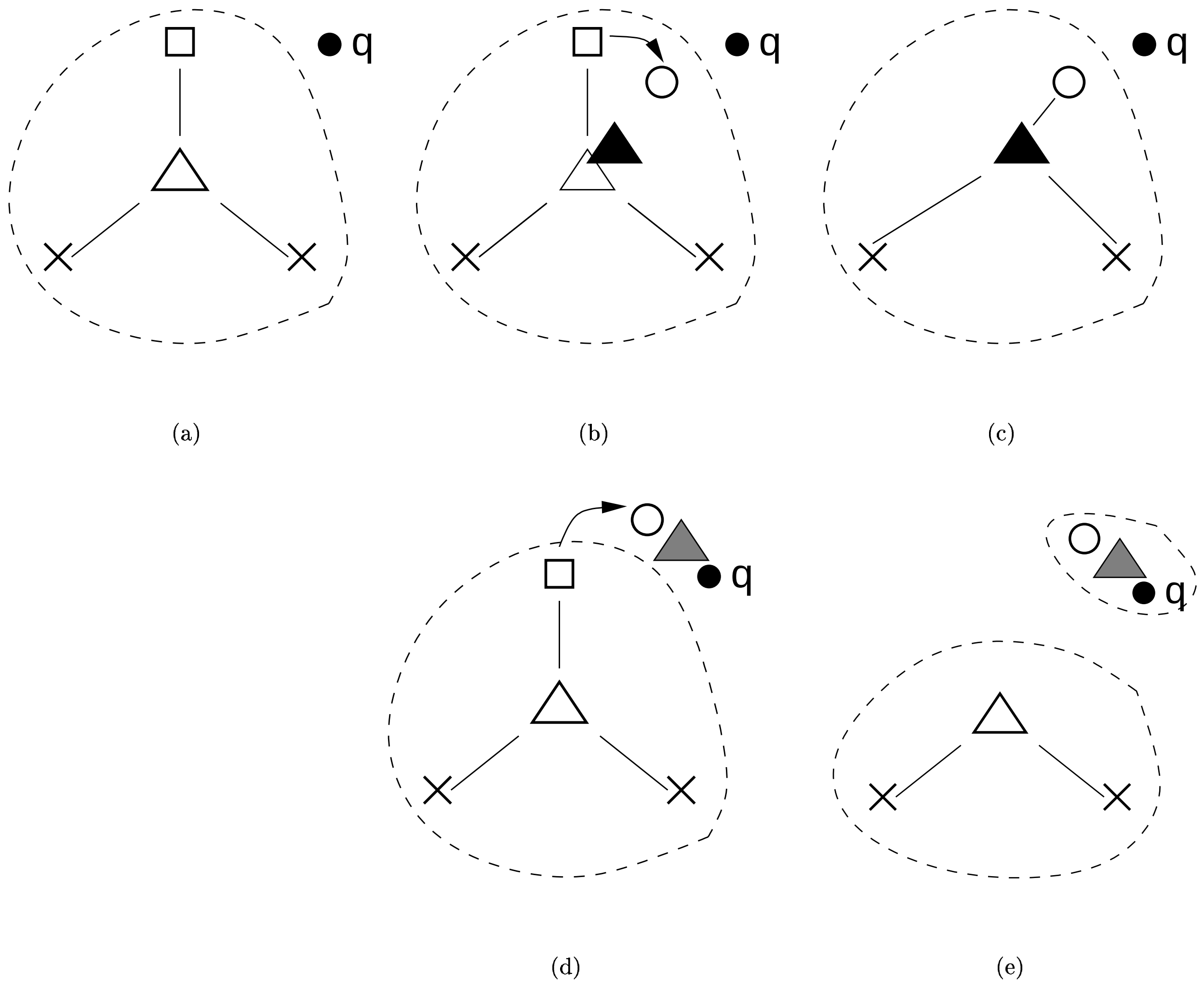
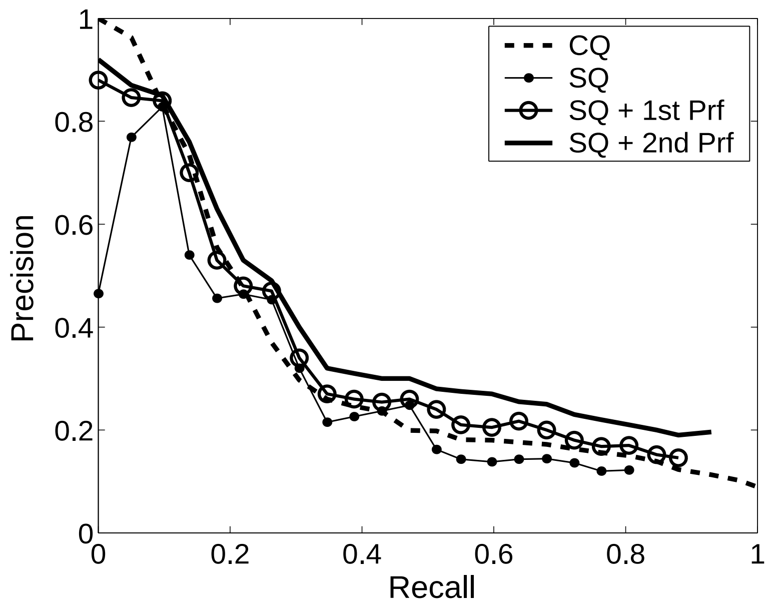
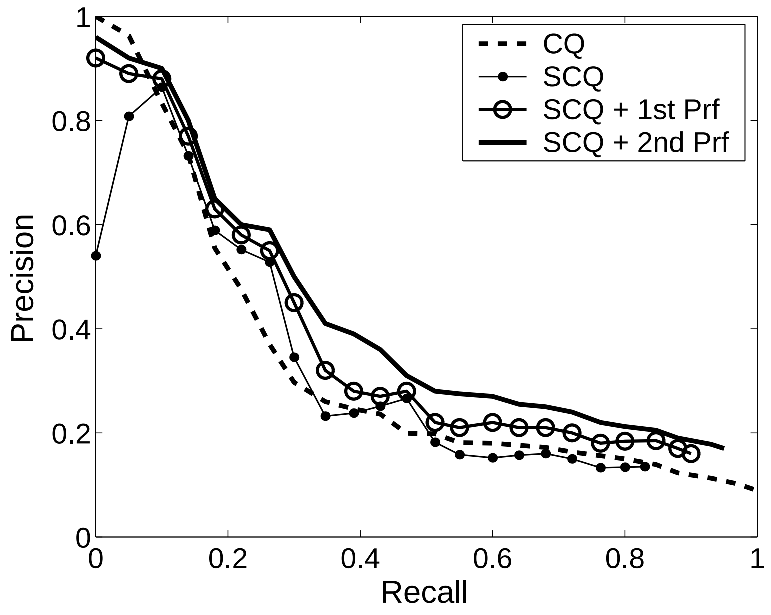


Fig. 7. Prototype refinement is performed on-line and off-line. In this figure, where a white circle means a manually drawn boundary from a relevant result, a gray triangle a newly created prototype, and a line a link between a prototype and an object: (a) An initial cluster includes one relevant object and two irrelevant objects with a given query **q**, (b) On-line refinement: a user is requested to draw a boundary of the relevant object and if the drawn boundary (white triangle) is still similar the corresponding prototype, the prototype is adjusted with the boundary, but the relationship (line) between the prototype and the relevant object is unchanged, (c) Off-line refinement: the system recomputes the distance between the adjusted prototype and relevant objects and periodically re-indexes with the updated prototypes, (d) On-line refinement: if the drawn boundary is not sufficiently similar, a new prototype (gray triangle) is created, and (e) Off-line refinement: the system computes a distance between the boundary and the new prototype and re-indexes the database with the new prototypes.

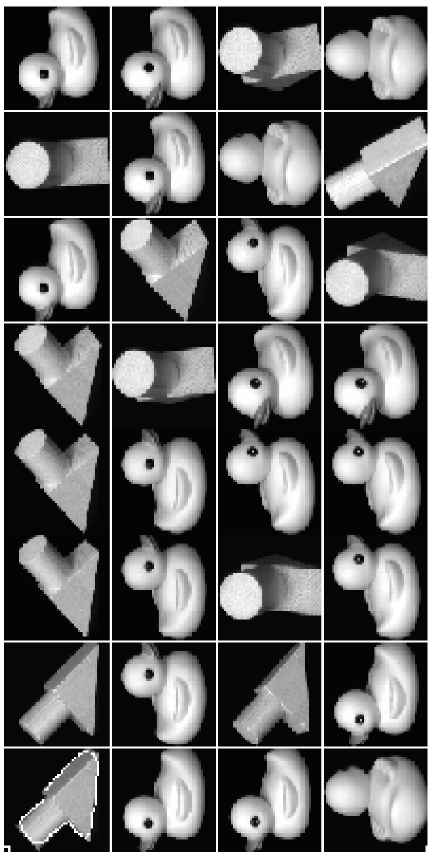


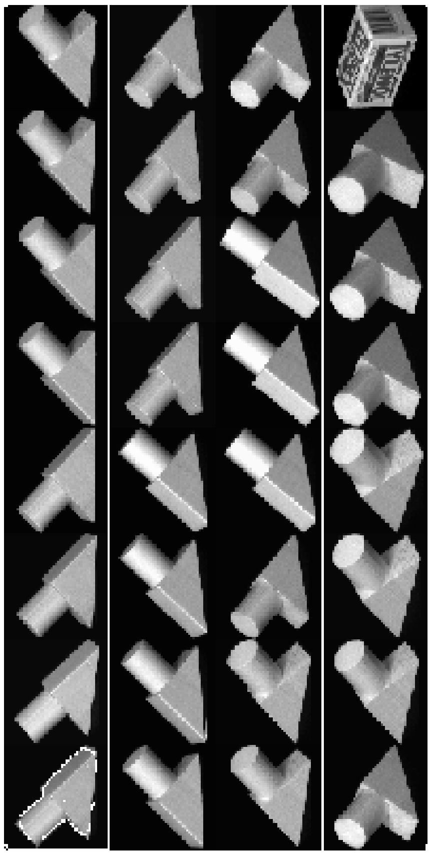
(a)



(b)

Fig. 8. Results on Columbia database: CQ indicates the color-based query, SQ the shape-based query, SCQ the shape-color-based query, and Prf the prototype refinement.


(a) Color-based query


(b) Shape-based query

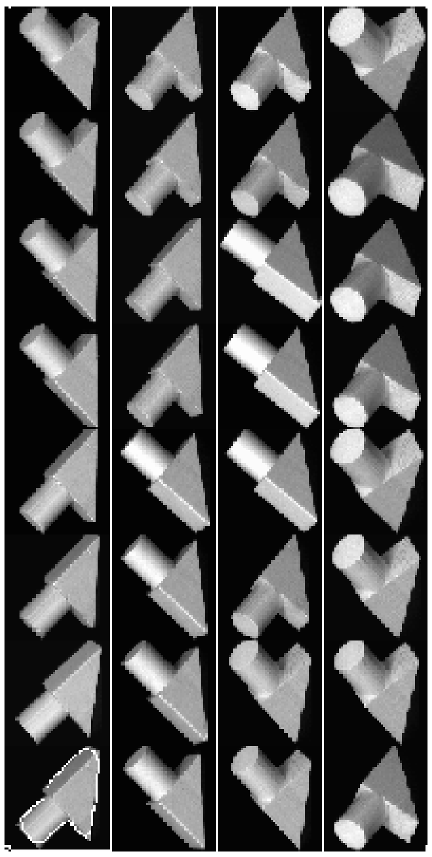
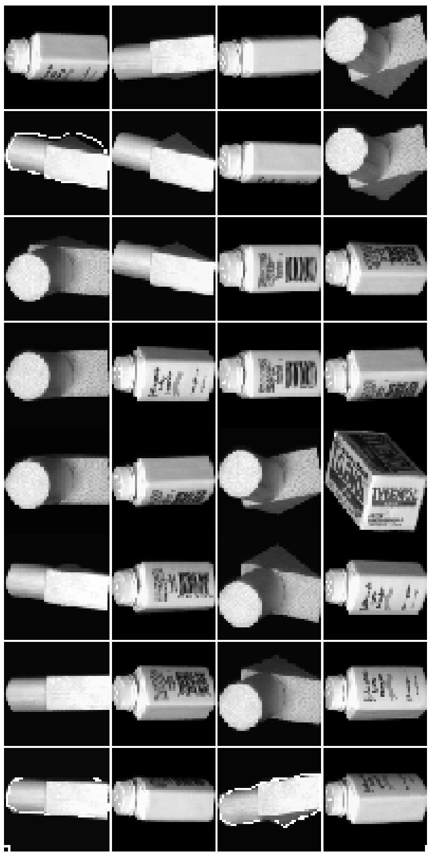
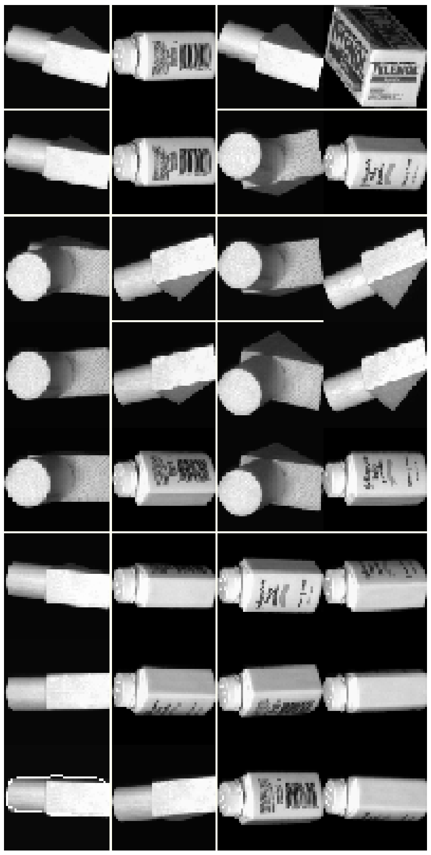

(c) Shape-color based query

Fig. 9. Sample retrieval result from the Columbia database. Images are ranked from left to right, top to bottom by increasing distance


(a) Shape-based query


(b) On-line refinement

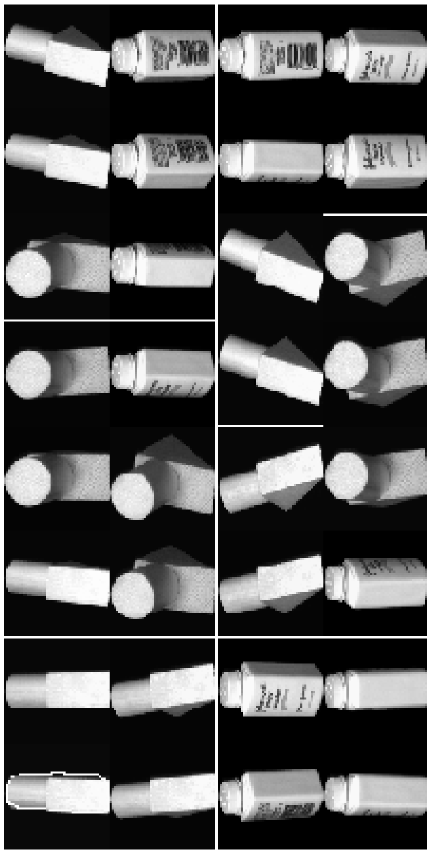
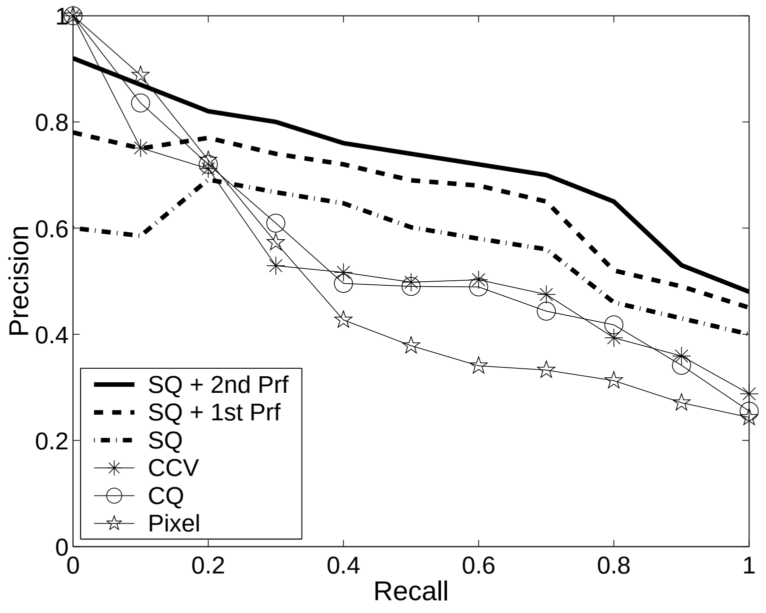
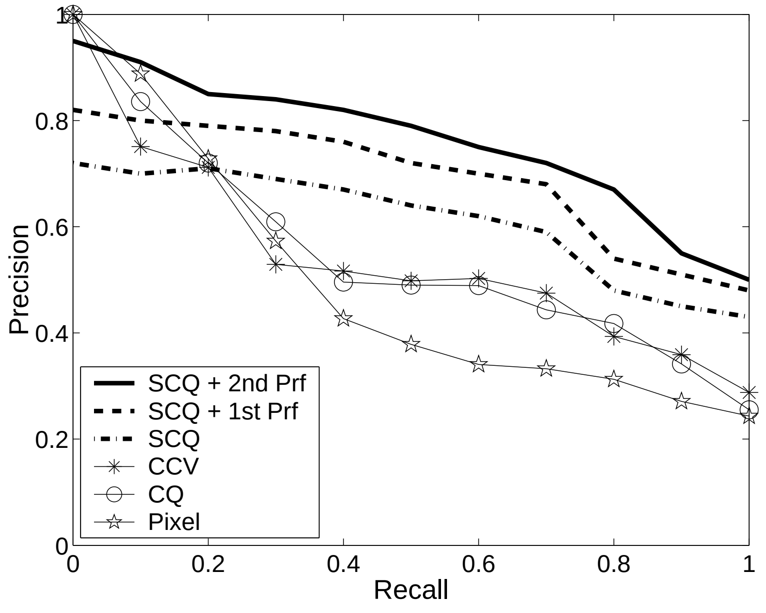

(c) Off-line refinement

Fig. 10. Prototype refinement increases precision and recall.

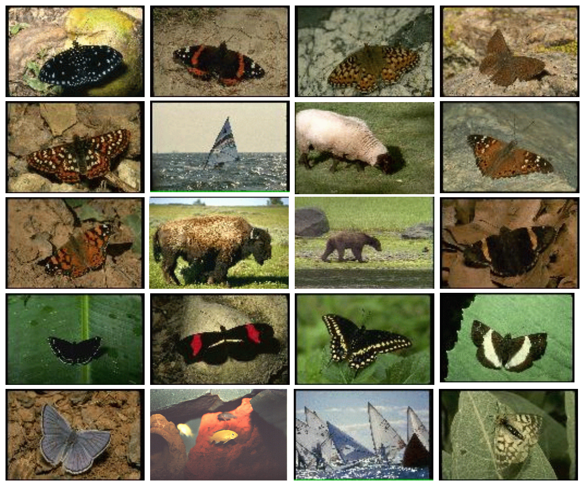


(a)

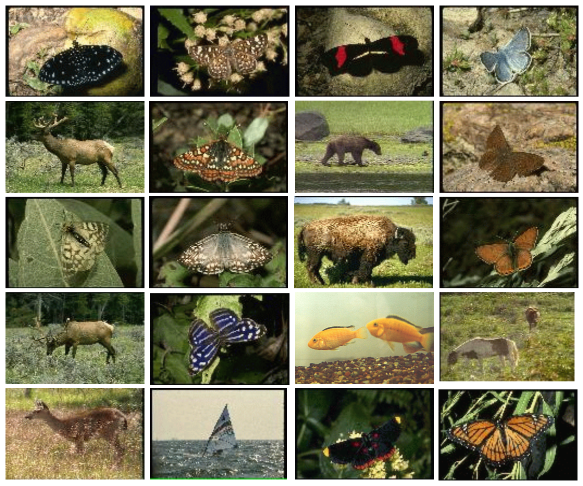


(b)

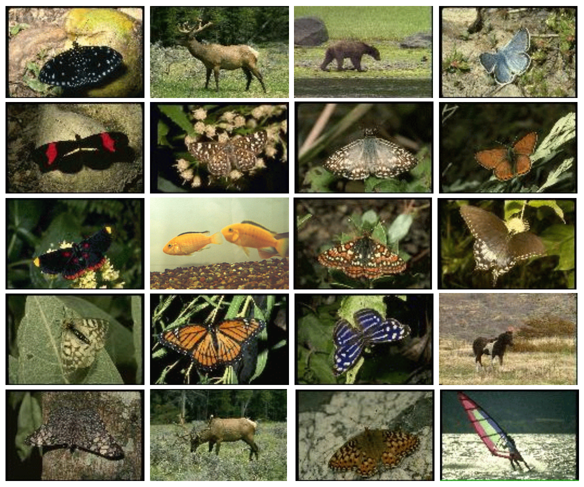
Fig. 11. Results on the color image collection: Pixel indicates the pixel-by-pixel comparison, CQ the color histogram, and CCV the color coherence vector.



(a) Pixel-by-pixel comparison

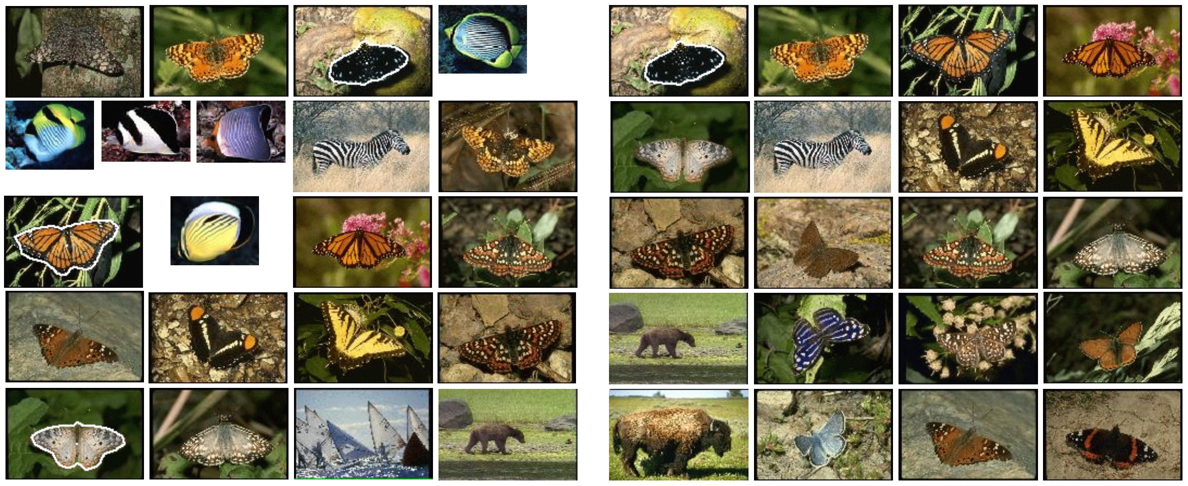


(b) Color histogram



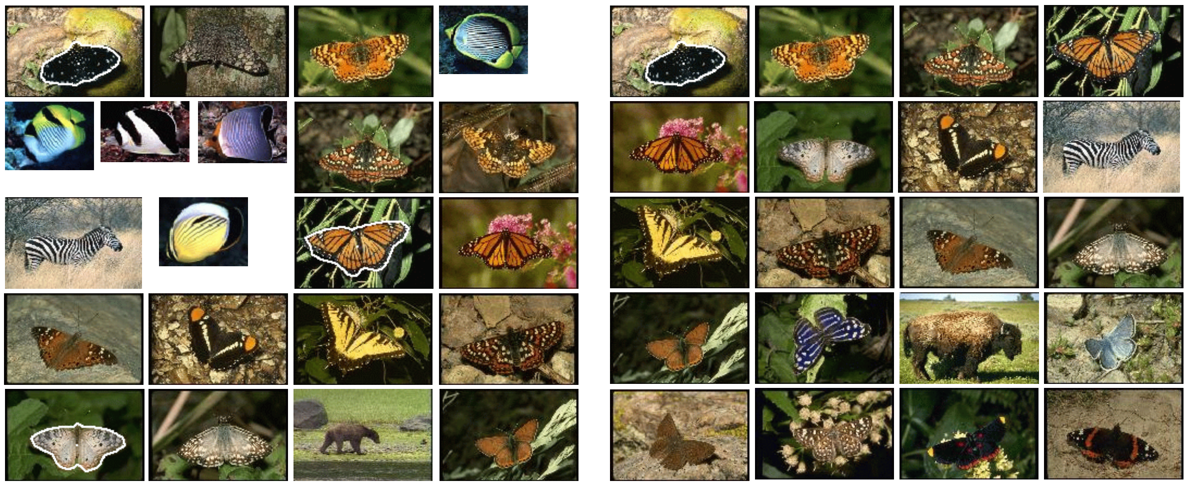
(c) Color coherence vector

Fig. 12. Color-based retrieval results of the Color image collection



(a) Shape query (b) Shape query with refinement

Fig. 13. Shape-based retrieval results of the Color image collection



(a) Shape-color query (b) Shape-color query with refinement

Fig. 14. Shape-color-based retrieval results of the Color image collection