Intelligent Object-based Image Retrieval using Cluster-driven Personal Preference Learning

Kyoung-Mi Lee
Department of Computer Science
The University of Iowa
Iowa City, IA 52242
klee1@cs.uiowa.edu

W. Nick Street
Department of Management Sciences
The University of Iowa
Iowa City, IA 52242
nick-street@uiowa.edu

ABSTRACT

This paper introduces a personalization method for image retrieval based on the learning of personal preferences. The proposed system indexes objects based on shape and groups them into a set of clusters, or prototypes. Our personalization method refines corresponding prototypes from objects provided by the user in the foreground, and simultaneously adapts the database index in the background.

1. INTRODUCTION

The target in this paper is an intelligent image retrieval system with personalization services. When different users have the same query but the level and depth of knowledge they have and what they expect differ, the system should return the user-preferred results to each of them. Further, 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.

We introduce the mechanisms of prototype refinement using cluster-driven personal preference learning. For indexing, the proposed system can cluster objects extracted by template matching. Each cluster is represented by *prototypes.* Users specify their particular preferences by providing feedback, and then this feedback is added into the corresponding prototype. 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.

2. SHAPE-BASED INDEXING

For automatic shape-based indexing, the proposed system uses the Generalized Hough Transform (GHT) to detect the boundaries of objects in an image. GHT requires a set of predefined templates (see example at right), and an array of points of the same size as the image, the accumulator, in which each point has a value, $h(x, y)$, specifying the possibility that the reference point of a shape to be detected is located at the point. When the value of the point in the accumulator exceeds a certain threshold, then an object with the desired shape is said to be detected at the location of the point. Each detected object is saved into the object database, along with the information on the image. All this processing is done in the background, creating an index of objects that is used for retrieval.

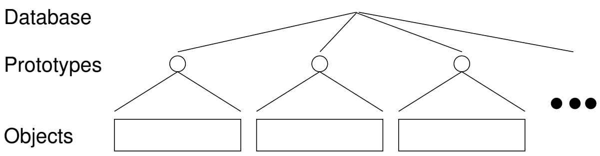


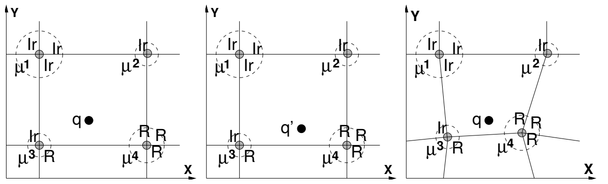
Figure 1: Index tree of the object-based database

Since only a small number of objects is likely to match any single user query, a large number of unnecessary comparisons are being performed. We use a two-level index tree structure to provide an efficient means to reduce the number of comparisons. Figure 1 consists of a prototype layer and an object layer. The prototype layer of the tree is a simple filter that reduces the search space quickly, discriminating the objects based on shape. This structure is easily extensible by adding a new layer to the index structure for a new feature, such as color. More importantly, using a smaller number of dimensions can avoid performance degeneration that arises for high-dimensional data.

3. CLUSTER-DRIVEN PERSONAL PREFERENCE LEARNING

The motivation is to store user feedback and avoid redundant and unnecessary steps in the personalization service. Figure 2 (a) illustrates a case where there are four prototypes in a two-dimensional feature space. Many

SIGIR'01, September 9-12, New Orleans, Louisiana, USA..



(a) (b) (c)

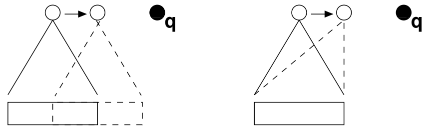
Figure 2: Relevance feedback gives more control on the search criteria. In figures, q means a query, $\mathbf{P}^1$, $\mathbf{P}^2$, $\mathbf{P}^3$ and $\mathbf{P}^4$ the centers of clusters (dotted lines), R a relevant result and Ir an irrelevant result. (a) Query and clusters, (b) Query refinement, and (c) Prototype refinement

existing systems use user feedback to adjust the desired query. This process of query refinement is illustrated in Figure 2 (b). For query $\mathbf{q}$, query refinement modifies the query utilizing user feedback. Without prototype refinement, when the user enters the same query again, the system would need to again perform query refinement. Instead, Figure 2 (c) shows how the prototypes are adapted to $\mathbf{q}$ and thus the system can start to search with those adapted prototypes.

An incremental learning method enables the system retrieve an ever-increasing volume of data. Each cluster is represented by a shape prototype. To measure the spread of a set of data around the center of the data in the cluster, we use the standard deviation (σ). Incrementally, whenever a new object (a query or a user-defined relevant object) is assigned to an existing cluster $\mathbf{c}$, the prototype P^c, can be updated by averaging the features that belong to the cluster $\mathbf{c}$. We propose the incremental learning method for the user to refine prototypes by specifying over time a set of relevant results. P_i^c and σ_i^c are updated by clustering the new object corresponding to the prototype P^c into the cluster P^c.

In retrieval, given a query $\mathbf{q}$ created by drawing an object on a sketch or on an image, selection of objects sharing similar characteristic features starts by comparing $\mathbf{q}$ to the set of prototypes. After computing the distance between prototypes and $\mathbf{q}$, the system sorts the prototypes based on this distance. Then, using the value $h(x, y)$ of each object as the similarity, the system can compute the final distance between $\mathbf{q}$ and each object matching the prototypes with high similarity.

At each step, the user may label retrieved objects as relevant or irrelevant to $\mathbf{q}$. The user also has to draw boundaries of the relevant objects. If the boundary is similar to one of the existing prototypes, the prototype is updated. On such an updated prototype, distances between $\mathbf{P}^c$ and $\mathbf{q}$ and between $\mathbf{P}^c$ and each object are changed. However, it is not appropriate to perform the indexing algorithm to calculate distances between $\mathbf{P}^c$ and objects for an on-line retrieval system. Therefore we run the indexing algorithm in the background and compute only the distance between $\mathbf{P}^c$ and $\mathbf{q}$ for on-line retrieval. Figure 3 illustrates the two methods for prototype refinement: on-line and off-line.

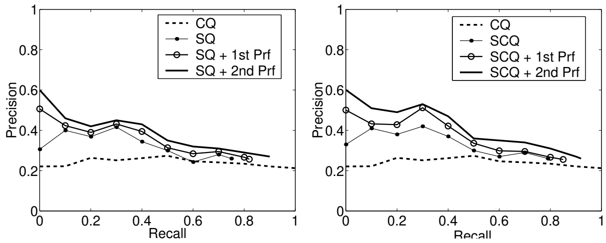


(a) (b)

Figure 3: Prototype refinement in the index tree: (a) On-line refinement (prototype adjusted to match query) and (b) Off-line refinement (match scores between prototype and objects recomputed)

4. EXPERIMENTAL RESULTS

We present experiments consisting of 127 color images collected from the Internet, divided into five different object classes. Each object is represented by a 36-dimensional shape feature and a 64 color histogram of the whole image, clustered into 64 colors in LUV space. For shape-based indexing, the system requires an initial definition of shapes we want to find. In this paper, we made 10 prototypes for the color image collection by drawing the boundary from 10 images. After getting the prototypes, the system performs shape-based indexing in the background. Each image is first preprocessed to get a good edge image, and then analyzed using GHT.



(a) (b)

Figure 4: Results: CQ indicates the color-based query, SQ the shape-based query, SCQ the shape-color-based query, and Prf the prototype refinement.

Figure 4 shows the average result of 10 queries. Even though the collection suffers from the difficulty of finding a clear edge boundary for objects in natural images, the indexing and the result of the shape-based query are better than that of the color-based query (Figure 4). These results continue to increase when we add the prototype refinements (Figure 4(a)) or the color features (Figure 4(b)) to find objects.

5. CONCLUSIONS

The proposed learning algorithm is based on an incremental clustering. This paper contributes to refine prototypes (both on-line and off-line learning) and to personalize the database system. We are currently extending our work to incorporate irrelevance feedback into prototype refinement.