Grouping Photos by Face

Huang-wei Chang Lei Huang Xiangrui Meng

Abstract

Image organizers such as imgSeek and GIFT allow query by image content (QBIC). QBIC help users find duplicate and similar photos. However, the similarity is measured based on color, texture and shape, not in terms of people.

We implemented a "query by face" feature for organizing personal digital photos, using face detection and recognition techniques. We used the OpenCV library to extract faces and combined Principal Components Analysis (PCA) and Locality Preserving Projections (LPP) for face matching. We observed acceptable accuracy in our experiment.

1 Introduction

Digital cameras have successfully dominated the consumer camera market. Compared to its film counterpart, the cost of taking and storing a digital photo is virtually zero. As a result, the common size of a personal photo collection has increased remarkably and tools for organizing digital photos become extremely popular. Current generation of photo organizers, such as ACDSee, Picasa and F-Spot, can automatically group photos by folder, album, tag, date, etc. Some organizers, e.g., ImgSeek and GIFT, have"query by content" function to find photos having similar style of color, shape and texture.

Our goal is to implement an interesting feature for photo organizing, that is, "query by face". We show a list of faces extracted from the photo being viewed. And for each face, we try to find photos containing similar faces within the collection. It looks like the offline version of Facebook's "Photos of ..." application. The key difference is that we don't have user generated tags. Some photo organizers do provide name tagging functionality, which is, however, rarely used. If we can achieve acceptable accuracy, this feature will certainly help people rediscover their memories.

In section 2, we give a brief survey of related work. Then we discuss face detection in section 3 and two face recognition methods, PCA and LPP, in section 4, as well as some practical issues. In section 5, we demonstrate our GUI and some result.

2 Related Work

Several photo organizers, such as imgSeek and The GNU Image Finding Tool (GIFT), use content-based image retrieval systems (CBIR). They allow query by image content (QBIC) and the similarity is commonly measured based on color, texture and shape. Using QBIC, it is easy to find duplicate and similar photos. However, it doesn't read and understand photos as a human does. In our opinion, the most desired feature for personal use is finding photos containing the same person. This is how we distinguish our project from existing CBIR-based photo organizers.

The most closely related to our work is the work by Zhang *et al.* In [4], they allow users to multi-select a group of photos and assign names to these photos. Then they try to propagate names from photo level to face level by solving an optimization problem on face similarity measures. However, our experience is that people do not do offline tagging very often. Our unsupervised "group by face" feature should be more practical than semi-supervised face annotation.

Just before we finished this report, we found that Google recently added face recognition in Picasa Web Albums [1]. They provide a friendly interface that groups similar faces together for quick name tagging. The precision of their "query by face" is very high. However, for efficiency of tagging, they only return very limited results with high similarity at the cost of losing recall rate.

3 Face Detection

Our first step is extracting faces from photos. Here we assume frontal face images only due the majority of frontal faces in personal photo collections and the high accuracy of frontal face detection. We use the OpenCV library in our project.

The function cvHaarDetectObjects() in OpenCV supports objects detection. It will return a sequence of detected objects in one image. We use the Haar classifier "haarcascade_frontalface_alt.xml" for frontal face detection, which comes with the library. The classifier computes Haar-like features which includes edge features, line features and center-surround features. After some classifying stages, it decides if the region is likely to show the object. To search for the object in the whole image one can move the search window across the image and check every location using the classifier. We set scale_factor as 1.1 so that the window will increase by 10% after each scan and set min_size as 100×100. The initial window size is a lower bound of face sizes in our collection and it is also the optimal scale for face recognition. Moreover, we use flag CV_HAAR_DO_CANNY_PRUNING to allow that canny edge detector to reject some image regions that contain too few or too much edges and thus can not contain the face. Since Haar-like features consider the shape of the image region, it can get higher accuracy than most color-based classifiers.

The code for this part is provided as sample code by OpenCV. It is easy to integrate in our photo organizer, which is a modified version of Eye of GNOME (EOG). We save the detected faces when photos are imported and build an index for searching.

4 Query by Face

4.1 Preprocessing

After cutting out the face images, we still need to do several preprocessing steps before applying the learning algorithm. First, the face images are mostly not of the same size. In order to perform dimension reduction, we resize all face images to 100×100 scale, which is also the suggested scale for face recognition.

Secondly, face images from a personal collection are quite different to those from a face database such as FERET. In FERET the background is generally very clean while in a personal collection the background varies a lot, which will lower the precision of face recognition. So we mask out the background and only use the center part of a face (Figure 1).

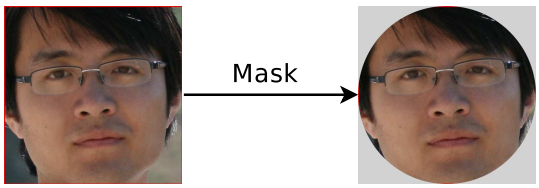

Figure 1: Mask out the background.

Finally, to alleviate the problem of light condition and light color, we transfer the face images from the RGB space to the gray-level space and normalize all face images to zero mean and unit variance.

4.2 Feature Space Learning — PCA and LPP

Since our goal is "query by face", we need to define some similarity or distance measure between face images. A naive mathod is just to vectorize the images and then compute the Euclidean distances. However, that will be inefficient and not robust. Therefore, we consider learning some feature space from face images. We explore two different projections from the image space to the feature space: principal components analysis (the eigenface space) [3] and locality preserving projection (LPP) [2].

The eigenface approach is a standard way to analysis face images, so here we only show the eigenfaces we learned (Figure 2) and in the following we mainly describe the locality preserving projection, which is one kind of manifold learning algorithms. Since it is usually believed that the manifold learning algorithms are more suitable than PCA for data that is possibly nonlinear distributed (such as face images), we are very interested in applying them to our application and comparing the result.

Figure 2: Eigenfaces

Briefly speaking, if the input space is high dimensional but the data are actually distributed over a low dimensional nonlinear manifold, the distance in the input space between two data points is realiable only when they are close to each other. Therefore, the LPP algorithm tries to learn a mapping into a lower dimensional space and simultaneously maintain neighborhood structures. The algorithm we use can be summarized as the following steps:

1. Constructing the adjacency graph: We build a graph in which each node represents one face image, and we add an edge between node i and node j if and only if i is among k nearest neighbors of j or j is among k nearest neighbors i.

2. Computing the edge weight: If there is one edge between i and j, we define the edge weight as $W_{ij} = \exp\left(-\frac{\|x^i - x^j\|^2}{2}\right)$.

3. Eigenmap: Define D to be a diagonal matrix whose entries are column sums of W, that is $D_{ii} = \sum_j W_{ji}$. Then we solve the generalized eigenvector problem

$$X(D - W)X^T v = \lambda XDX^T v,$$

and use the eigenvectors corresponding to the smallest l eigenvalues as our projection directions.

Please see the paper [2] for justification of the above algorithm.

4.3 Additional Rules

After mapping the face images into one feature space, we can use the Euclidean distance between two feature vectors to find the nearest neighbors as the results for the query. However, although we handle the photo retrieval part as an unsupervised problem, we still can take advantage of some basic facts. First, face images from the same photo can not be from the same person. Second, denote x^{ij} to be the j-th face in the i-th photo. If we want to include x^{kl} into the query result of x^{ij}, then x^{kl} shouldn't be nearer to any other face in the i-th photo than its distance to x^{ij}. In fact, this can be thought as an extension of the first rule. We use those two facts as additional rules for our photo retrieval routine.

5 Result

Our image viewer is a modified version of EOG, where we added a face browser. After user chooses one face as query, we will show query results in the photo collection box.

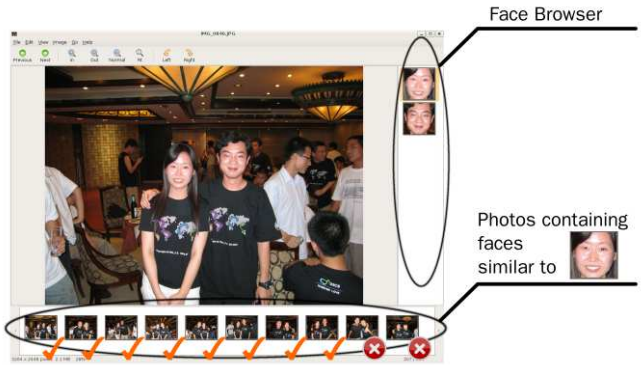

Figure 3: EOG-Face

We found the precision of finding nearest neighbors in the PCA space and in the LPP space are close. We also manually labeled all the face images and found if the nearest neighbors are outputed the error rate using the PCA features is about 29% and error rate using the LPP features is about 28%.

Furthermore, suppose we ask to retrieve k photos that contain the query person. Since in a personal photo collection only few people may appear many times, the presion will degenerate quickly as k increases. Figure 4 shows some sample queries from our experiment.

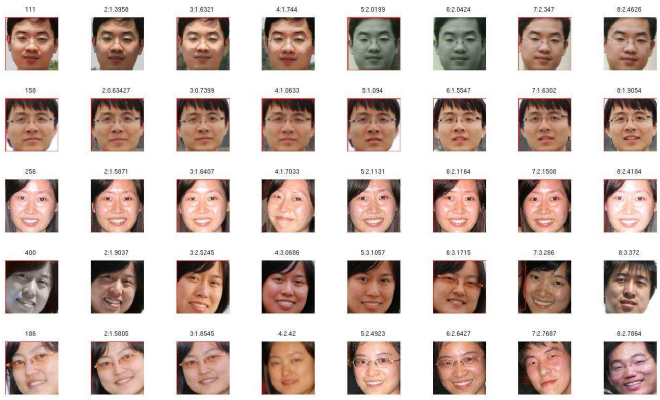

Figure 4: Sample Queries. Faces in the left column are the queries.

Moreover, in a personal collection the photos can have very different illumination conditions; not only the angles of light, which are usually taken into account in standard face databases, but also the intensity and the light color. If the light condition is controlled, the intensity (or color) of skin can be a distinguishing feature for face retrieval. However, the real situation is the lighting conditions are far more complicated than those of the standard face database. Simple preprocessing can not solve this problem well. Therefore, this may be one direction we can explore further.

Finally, not surprisingly, because our feature spaces are learned for whole faces and no side information about the faces is given, our system can not handle occlusion, poses, rotations with large angle very well. That is also an interesting direction for us to investigate.

References

[1] Picasa name tags. http://picasa.google.com/support/bin/topic.py?topic=14605.

[2] X. He and P. Niyogi. Locality Preserving Projections. In *Advances in Neural Information Processing Systems 16: Proceedings of the 2003 Conference*. Bradford Book, 2004.

[3] M. Turk and A. Pentland. Eigenfaces for Recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.

[4] L. Zhang, Y. Hu, M. Li, W. Ma, and H. Zhang. Efficient propagation for face annotation in family albums. In *Proceedings of the 12th annual ACM international conference on Multimedia*, pages 716–723. ACM New York, NY, USA, 2004.