On-line Face Recognition Using SIFT Features
– Pattern Recognition Course Project Report

Zhu Tianhua
09300240004

Abstract

The project has implemented an automatic face recognition program, which can learn on-line new comers and identify different people. It performs well for single person sceneries, and can be used under multi-personal circumstances. In this project, I have also surveyed several classical statistical learning methods concerning face recognition or tracking, including Eigenface, Fisherface, CAMShift, SIFT, etc.

The report first reviewed the background and previous work done on face recognition. Then, after the introduction of Scale Invariant Feature Transformation (SIFT), the report showed its application in on-line face recognition, as well as the design and implementation for a practical program. At last, the report concludes that further works are still to be done on improving accuracy and speed.

1 Introduction

As robotics prospers, it becomes more and more likely for ordinary families to own a household robot for security monitoring, for simple housework, and for entertainment. Thus a friendly interface between people and robots is required. In this project, I intend to go through a research on on-line face recognition, to cope with situations where very limited samples for training is available and we need to track multiple faces.

In the following sections, we will first have a quick look at existing solutions for this objective, and propose our solution with its implementation.

2 Related Work

2.1 Face Detection

Literature [4] surveyed algorithms ranging from simple edge-based algorithms to composite high-level approaches utilizing advanced pattern recognition methods. The survey traced back the history of face detection. Literature [9] describes a face detection framework that is capable of processing images extremely rapidly while achieving high detection rates. Cascade classifiers combination is also proposed, and is contributing to speeding up processing by quickly discarding background regions of images.

Nowadays, face detection using *OpenCV*'s *Haar Feature-based Cascade Classifier* has been quite common. Haar feature and the classifier was proposed in [8] and [5].

However, due to its training set, the default Haar feature data does not perform well when frontal face rotates. Thus it requires a more robust way for detection. Also, it may output false alarms. Later we will see how to reduce false alarms using SIFT.

2.2 Face Recognition

Eigenface[7] and Fisherface[1], using PCA (Principal Component Analysis) and FLD (Fisher Linear Discriminant), are involved in face recognition phase. A coarse comparison is to be conducted.

Paper [11] overviewed the history of face recognition and cited many related works. The paper observed a critical survey of still- and video-based face recognition research. In addition, relevant topics such as psychophysical studies, system evaluation, and issues of illumination and pose variation are covered.

Zhang *et al*. made comparison among Eigenface, elastic matching and neural nets, concluding with the fact that elastic matching works best with relatively simple implementation comparing with neural nets, as well as eigenface is fit for stable illumination and fast programming. [10] The paper also proposed some disadvantage of igenface.

2.2.1 Eigenface

Eigenface learn 'eigenfaces' from a closure set of samples. Informally, eigenfaces can be considered a set of "standardized face ingredients", derived from statistical analysis of many pictures of faces.[1]

With the help of Haar Classifier and preprocessing in face detection phase, one of major disadvantages of Eigenface, that is, sensitive to hair style, face position and background, can be eliminated to some extent. With the assistance of OpenCV, problems caused by illumination condition can be reduced.

2.2.2 Fisherface

Fisherface, on the other hand, differs from Eigenface. It emphasizes variance between classes as well as coherence within classes, rather than solely seeking for most 'principal' components in the meaning of within-class variance. It was first proposed in [**?**], and in a comparative performance analysis carried out in [1], Fisherface method performed significantly better than correlation-based method, Eigenface, and a variant of the linear subspace method. [11] However, no claim was made about the relative performance.

2.2.3 Comparison and Possible Problems

It is clear that both Eigenface and Fisherface are not online learning, and consequently, it will inevitably occur that the robot needs to 'refresh' its knowledge about someone under circumstances where environmental conditions or the person's appearance changed. There are, as far as I have thought, two ways to avoid such cons: a) make the process nature and user-friendly, and thus acceptable, by interacting via voice; b) store a definite number of past correctly recognized images as training samples, and the robot re-train itself frequently. Both the two, however, are linked with human-machine interface construction.

2.3 CAMShift

CAMShift stands for "Continuously Adaptive Mean Shift."[2] It is a histogram-based algorithm used for fast face tracking. Given the initial position of a face, CAMShift uses its colour statistic histogram as a pattern, search in the adjacent area around last face position to match for a face. It requires manually set up at first, and can hardly be extended to situations where more than one face may occur.

This method is fast and appears on initial testing to be moderately accurate. It may be possible to improve accuracy by using a different colour representation. It inspired us to find only faces in their last positions for fast matching. We can therefore make use of information provided by nearby frames.

[1] http://en.wikipedia.org/wiki/Eigenface

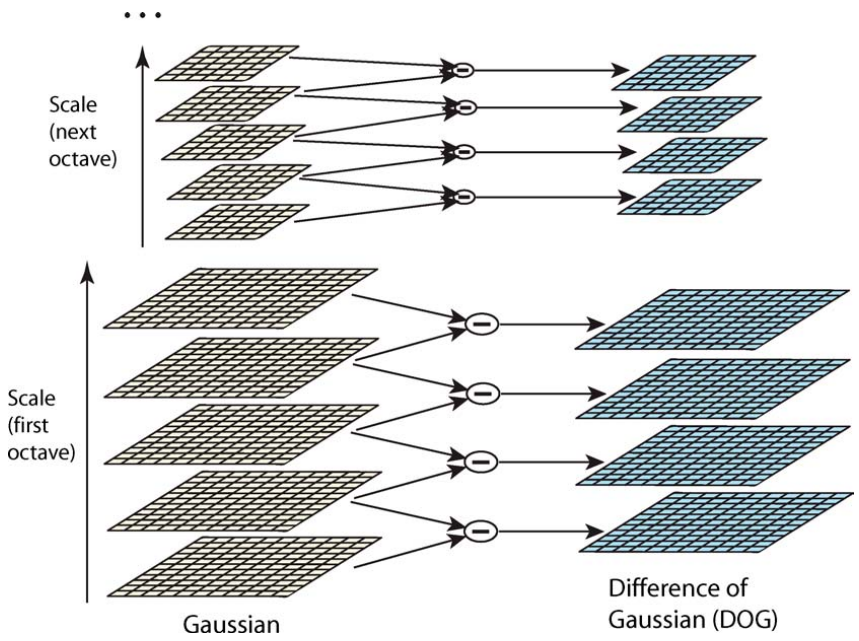

Figure 1: DoG operation

2.4 SIFT

Scale Invariant Features Transformation (SIFT) was first proposed by Lowe in 1999 and accomplished in 2004 [6]. For any object in an image, interesting points on the object can be extracted to provide a "feature description" of the object. This description, extracted from a training image, can then be used to identify the object when attempting to locate the object in a test image containing many other objects. [2]

Following are the major stages of computation used to generate the set of image features:

1. Scale-space extrema detection: it searches over all scales and image locations, implemented efficiently by using a difference-of-Gaussian function (as is shown in Figure 1 and 2) to identify potential interest points that are invariant to scale and orientation.
2. Key-point localization: At each candidate location, a detailed model is fit to determine location and scale.
3. Orientation assignment: One or more orientations are assigned to each key-point location based on local image gradient directions. All future operations are performed on image data that has been transformed relative to the assigned orientation, scale, and location for each feature, thereby providing invariance to these transformations.
4. Key-point descriptor: The local image gradients are measured at the selected scale in the region around each key-point. These are transformed into a representation that allows for significant levels of local shape distortion and change in illumination.

SIFT transforms image data into scale-invariant coordinates relative to local features. For image matching and recognition, SIFT features are first extracted from a set of reference images and stored in a database. A new image is matched by individually comparing each feature from the new image to this previous database and finding candidate matching features based on Euclidean distance of their feature vectors.

Due to these reasons, we can imagine using SIFT for face recognition. However, we find it insufficient to rely on SIFT. As is seen in figure 3, picture (a) showed a successful example, while (b) and (c) showed that SIFT may fail in distinguishing two faces. A test running on the whole ORL set

[2] http://en.wikipedia.org/wiki/Scale-invariant_feature_transform

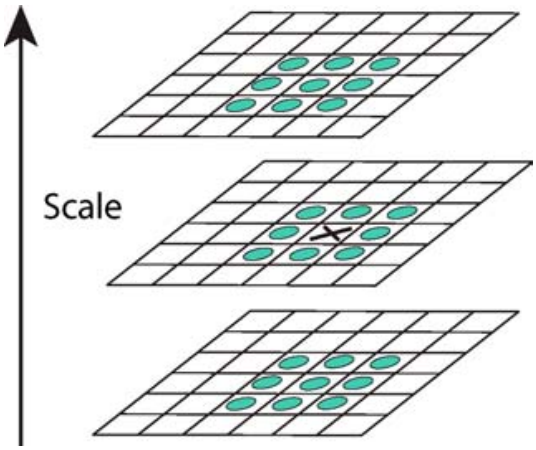

Figure 2: Scale-space extrema detection

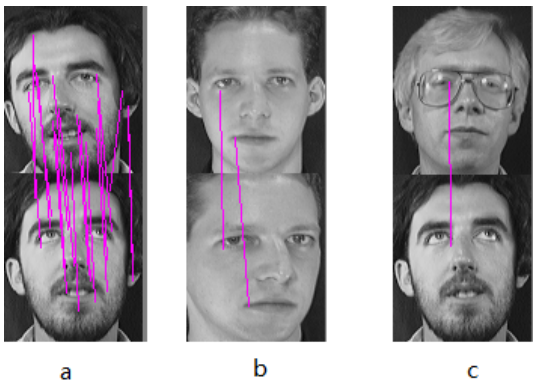

Figure 3: Tests on examples from the ORL dataset

indicated that pure SIFT method can reach 87.8% accuracy (49 errors in all 400 tests) when trained with 2 pictures per person and the least matches threshold set to zero, which is just a little better than face recognition using Eigenface. An example of face recognition failure is given in figure 4.

In this project, I use the implementation of SIFT algorithm by Rob Hess [3] [3]. His open-source SIFT library is implemented in C using OpenCV and includes functions for computing SIFT features in images, and matching SIFT features between images using kd-trees.

3 Previous Work

Tests on ORL face dataset revealed the fact that as training objectives increases, the error rate gets increasing. On the other hand, another test on ORL shows that Fisherface enjoys approximately 96% accuracy on the 40-person ORL dataset.

Previous work also includes the implementation of Eigenface and a real-time detection & recognition program, as is shown in figure 6.

3.1 An Improvement on Eigenface for Recognition

There are several disadvantages in the original Eigenface recognition process. First, if we want to add a new person for recognition, we need not only newly sampled face images, but also those for

[3] http://web.engr.oregonstate.edu/ hess/

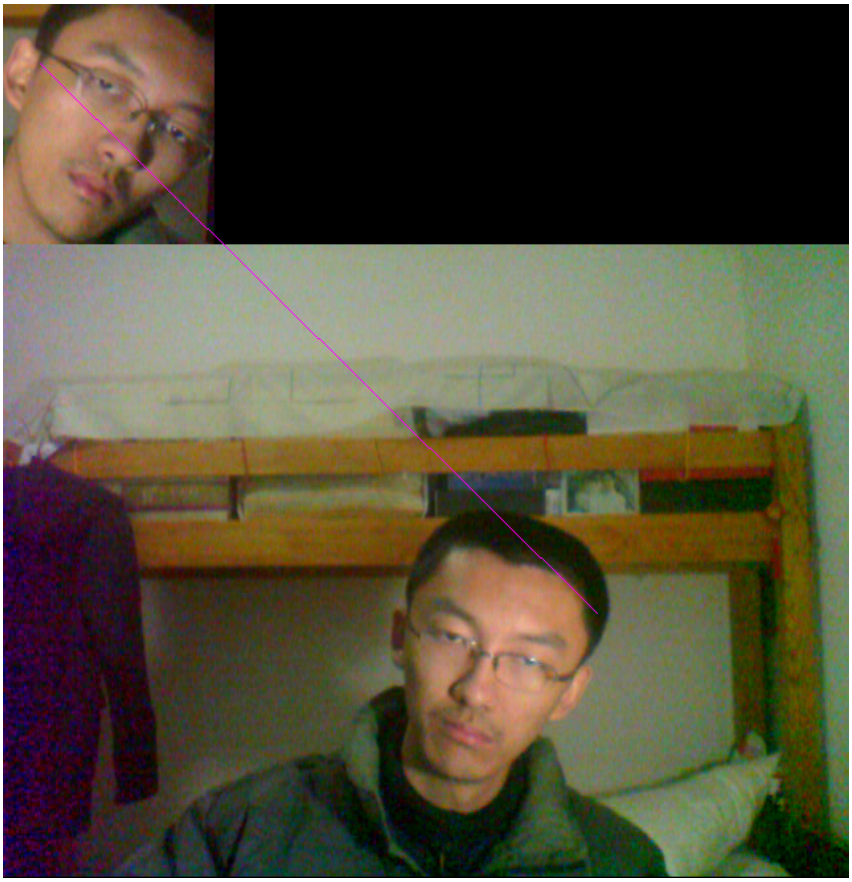

Figure 4: Face recognition failure

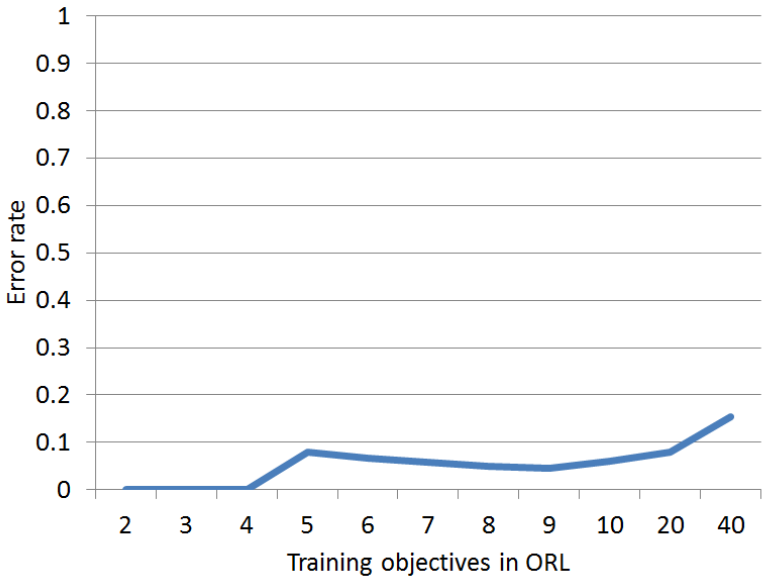

Figure 5: Face recognition result on ORL using eigenface

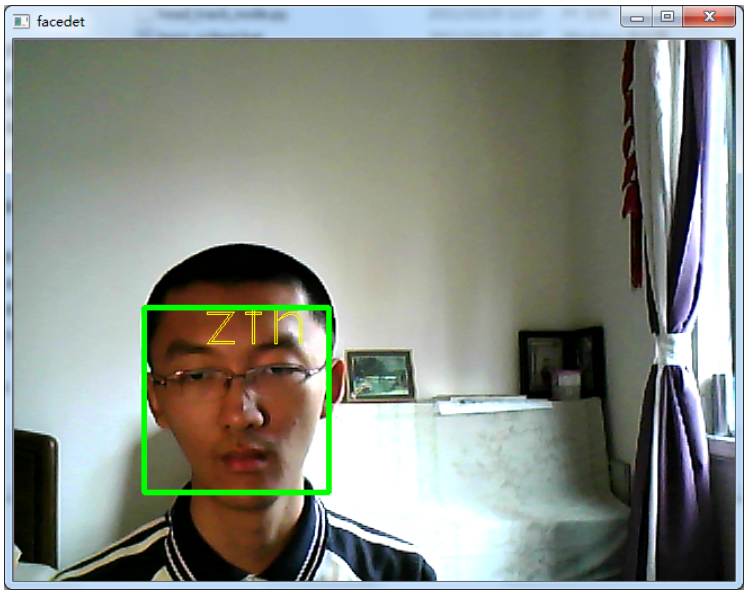

Figure 6: Face detection and recognition using Eigenface, runtime snapspot

Figure 7: Eigenface decomposition for a single person

people we already trained for. Secondly, as the number of the training samples grows, we need to spend more time for training.

To solve these problems, we may consider a face space spanned by M' Eigenfaces. Suppose that the reconstructed image is $\hat{\Omega} = \sum_{i=1}^{M'} \omega_i \vec{v}_i = \sum_{i=1}^{M'} \vec{v}_i \vec{v}_i^T (\Gamma - \Psi) + \Psi$ with remainder $r = \|\Omega - \hat{\Omega}\|$ representing the distance from the original image space to its projection in face space.

Now, we build a face space respectively for each recognition object, so that we convert the problem of classifying new input image Ω to finding out $k* = \arg\min_k r_k$, where r_k is the distance between Ω to the k-th face space. Figure 7 has shown an example for Eigenface decomposition.

Assemble the matrix $W_k = [\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_{M'_k}]$, and we will have that $r_k = \|W_k W_k^T (\Gamma - \Omega_k) - (\Gamma - \Omega_k)\| = \|(W_k W_k^T - I)(\Gamma - \Omega_k)\|$.

Let $n_k \equiv M/K$, $M'_k \equiv M/2K$, so we need to perform $KN^2M/2K = N^2M/2$ times of multiplication in order to distinguish among K objects. Namely, computation complexity for recognition remains unchanged. When training, we had to calculate eigenvectors for a $M \times M$ symmetric matrix, but now we need only to compute those for at most K $M/K \times M/K$ matrices. Complexity for the former is $O(M^3)$, and the latter is $O(M^3/K^2)$, which gets optimized. At the same time, the latter enables add and remove objects or re-train quickly, without keeping the training set for other objects. Experiments also showed that it gets higher accuracy than original method.

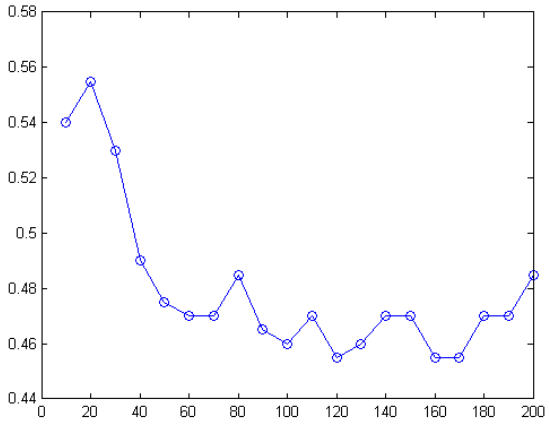

Figure 8: Error rate vs. Eigenfaces used, original method

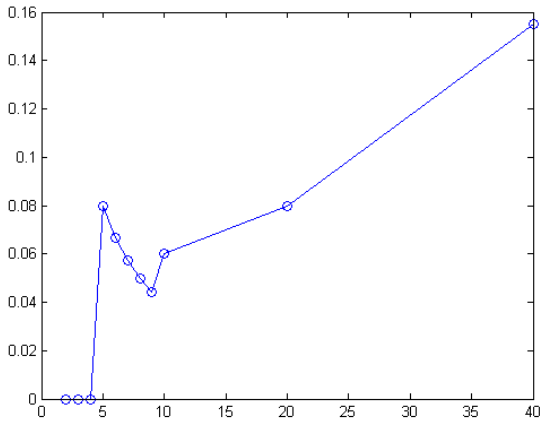

Figure 9: Error rate vs. objects, improved method

3.2 Experiment Result

Figure 8 has shown the relationship between the number of Eigenfaces and error rate. We can see that it is not monotonous. This indicated that too many Eigenfaces cannot improve accuracy.

Figure 9 has shown the performance of improved method on ORL face dataset. The X-axis is the number of objects, and Y-axis is error rate. We can see that our improved methods enjoys higher precision.

4 Face Detection and Recognition Using SIFT

4.1 Single Person Sceneries

As is introduced in the last section, we have SIFT to extract features from given images and can use them for recognition. Given an evaluation function $f(a, b)$ and a corresponding threshold θ, where a, b are two series of Scale-invariant features (SIFs), we can perform face recognition using SIFT. The value of f increases as a and b differs more. The procedure, shown below, is on-line.

1. Use Cascade Haar Classifiers to detect faces in an image.

Figure 10: The two people are both labelled as yellow and same id

2. For each face detected, perform SIFT in its rectangle area, resulted in a series of SIFs, notated by a.
3. If $\|a\| < N_0$, where N_0 is a priorly given minimum of SIFs, we will ignore this face image for it contains too little information for our determination.
4. For each series of SIFs s stored in list L, we evaluate their similarity using f. If $min_{s \in L} f(a, s) > \theta$, we regard the input face image as a new face, and thus add a to L, and give it a new id. Otherwise, we just return the id of s.

The experiments on ORL will evaluate its precision.

The procedure above may encounter two typical kinds of errors: (a) recognizing a person as another, and (b) regard a preliminarily acquainted face as unknown. We may guess that the more input images we use for training, the higher the accuracy will be.

Inspired by CAMShift, we may assume that, it is highly possible that in continuous two frames of video, the person located at (x, y) in the first frame will be located in an adjacent area of that point in the next. So we introduce a probability distribution function $p(d)$, where d is the distance of two faces in a frame, to score this probability.

4.2 Extending to Multi-face Sceneries

Cascade Haar Classifiers usually identify an area of blank as face. To reduce such errors, we have introduced N_0 to ignore those areas containing too few details. On the other hand, if we combine multiple approaches of face detections together, we may have overlapping areas as well as rectangle areas too close. During the detection phase, such overlapping areas will be merged or ignored, so that increase detection accuracy. As a disadvantage of it, we may get fewer-than-actuality face areas. However, due to our aim, we need enough resolution to identify people, so there won't be many faces in an image.

After such pre-process in detection phase, we just start up recognition phase and check similarities between the SIFs a of input and those of stored acquaintances, one by one. We may simply use the following equation to find the result:

$$\arg\min_{x \in L} f(a, x) \tag{1}$$

Such judgement will result in that a person occurs in more than one place in the image, which is impossible in reality (see figure 10). And also, it may cause chaos when multiple samples trained and recorded for one person.

So we may change our strategy to assigning each face a person, rather than processing them independently. In the recognition phase, we have scored $f(a, x)$ for each $x \in L$, so that we can build

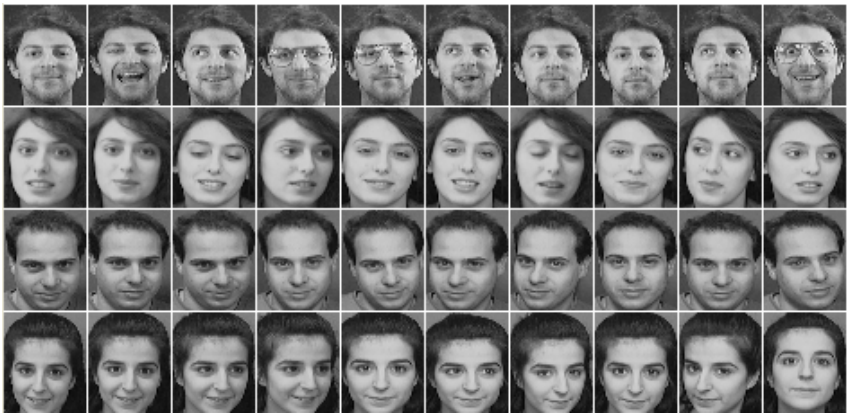

Figure 11: Four examples from the ORL dataset

a matrix $A = (f(a_i, l_j))$, where a_i is the i-th face and l_j is the j-th SIFs in database L. For those scores lower than threshold $theta$, we assign the id of l_j to face a_i if $l_j = \arg\min A_i$, otherwise we recognize it as unknown and leave it for acquainting phase.

In acquainting phase, we first find out the nearest face in the last frame, and compare it with that of current frame. If the score is lower than a looser bound $\hat{\theta} > \theta$, we regard the two faces as the same person, and then add the new sample SIFs to L. Otherwise, we will allocate a new id for it.

4.3 Determining function f

In Rob Hess's implementation, he used kd-tree and 2-NN method to find matches. If the two nearest neighbours in SIF space are near enough comparing to their norms, we admit it as a match. Our function f simply returns $1 - m/n$, where n is the number of SIFs in the current series.

An alternative is to use $\sum_{b_i} d_0/d_1$, where d_0 and d_1 are distances between the given b_i SIF in series b and its two NNs.

As a consideration of using probability distribution p, we may score it as $\hat{f} = f(\cdot) \cdot p(d)$ for valid distance d, or $\hat{f} = f \cdot p_0$. In our program, we use log to improve float calculation precision, $\hat{f} = \log(f(\cdot) + \epsilon) + \log(p(d))$, where ϵ is very small to avoid log on zero, and $p(d)$ is (not really a probability distribution function) adapted from Gaussian distribution: $p(d) = \exp(-d^2/(2\sigma^2))$, so that we can also bypass log.

5 Experiments and Results

5.1 Dataset

In order to examine recognition availability and precision, some widely-used datasets are introduced for testing proposes. The ORL Database of Faces[4] is used for face recognition trails.

In order to examine recognition on video, I used two short video clips recorded in class. One is a teacher's monologue, another is a side-frontal view of students. The snapshots is shown in figure 12.

The two video clips are all sampled at 29 fps, with height of 480 px and width of 720 px, lengthened 1' 42" and 20", respectively. They are converted from an Advanced Streaming Format (ASF) video of lecture to AVI format, and contains some codec errata, typically are scan lines and noises.

[4]http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html

Figure 12: Snapshots for the two video clips

5.2 Results for Static Images

Running on the ORL face dataset help diagnose errata while programming and optimizing results by rounds of tests. An improved Eigenface implementation was adopted for Eigenface, versus SIFT and KNN strategy for recognition.

Results for Eigenface (use 5 samples per person for training) is shown in figure 5. A test running on the whole ORL set indicated that pure SIFT method can reach 87.8% accuracy (49 errors in all 400 tests) when trained with 2 pictures per person and the least matches threshold set to zero. We can find out that SIFT is more competitive when training samples are highly limited, and we are inconvenient to re-train the model on-line for Eigenface.

5.3 Results for Videos

Face recognition test shows that the program works well on single person samples. When facing many people, it may recognize some but not all (due to face detection limitations), and tolerate rotation to some degree. If we just use the whole image for SIFT, we may get better results when facing rotation problems, but we must sacrifice speed performance. The program is yet not so robust or stable, as illumination conditions differs and colour space is sensitive to it.

The processing speed is approximately 1.66 fps, and may get even slower as time grows, and so the number of features it acquainted keep increasing. To avoid this problem, we may set up some thresholds for the program, so that it may keep only features occurring frequently and/or recently and eliminate those seldom come around or out of sight for long. It still leaves large room for speed improvement.

The following snapshots figures out the output of this recognition program. Notice that it can only identify frontal faces, and also that these faces are actually very small and somehow vague. The forth picture in figure 13 tagged two person with green, for that they have id 1 and 9, respectively, and the program reused its 8-colour representation. The white circles in pictures are filtered false face regions.

6 Conclusion and Discussions

The project has implemented an automatic face recognition program, which can learn on-line new comers and identify different people. It performs well for single person sceneries, and can be used under multi-personal circumstances. Its speed is slow, and far from real-time recognition, but its simplified version can get higher speed, yet lower correctness.

In this project, I have surveyed several classical statistical learning methods concerning face recognition, including Eigenface, Fisherface, CAMShift, SIFT, etc., and improved my academic ability as well as command of English. I have also learned LaTeX, OpenCV, Python, and a little Lisp when working on this project. I would like to appreciate Prof. Chi for her guidance on this project.

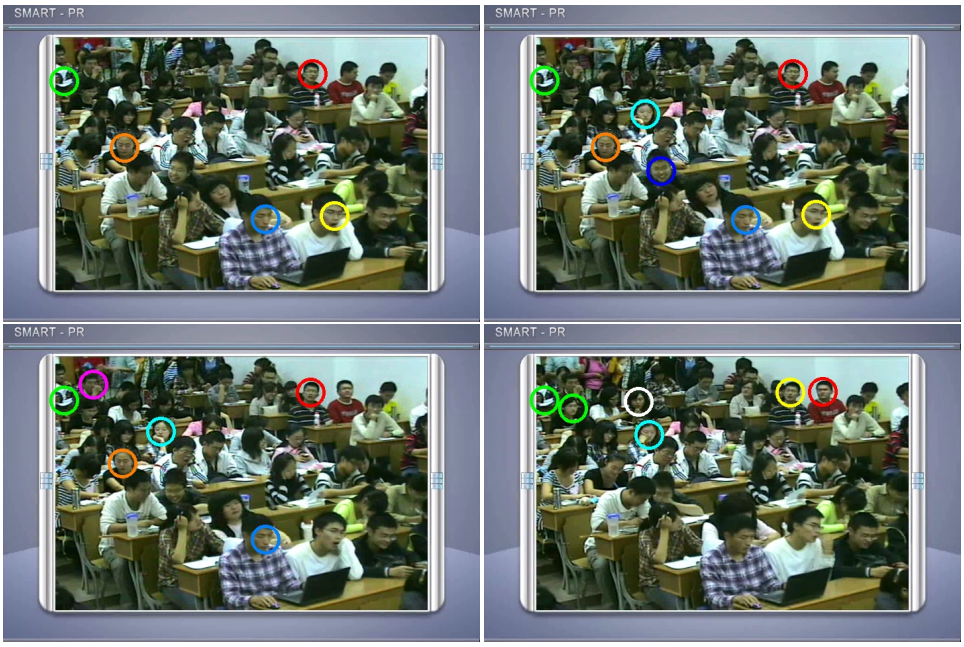

Figure 13: Snapshots for multi-person scenery

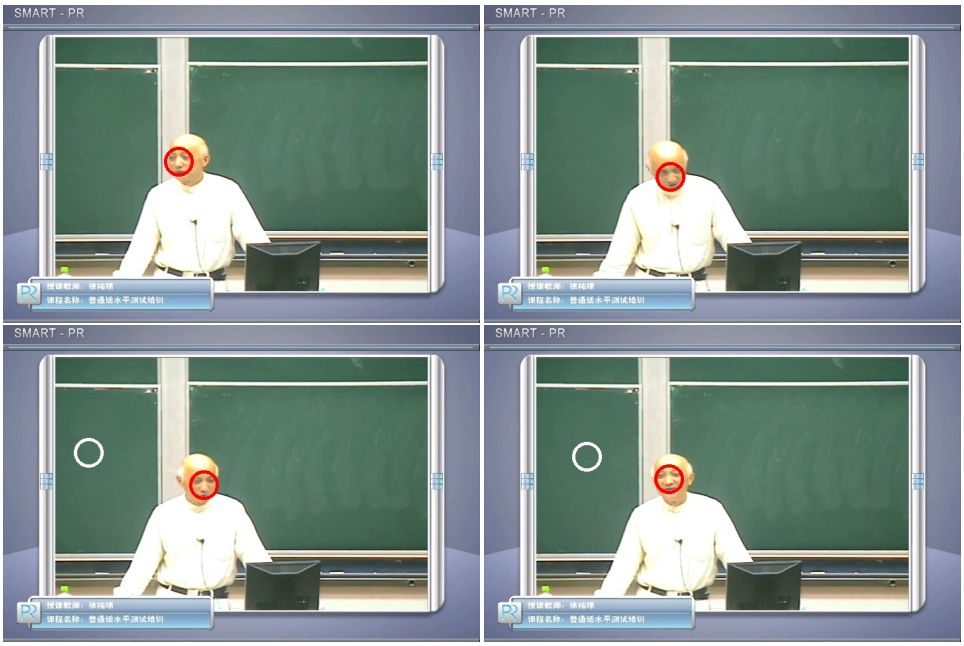

Figure 14: Snapshots for multi-person scenery

References

[1] P. Belhumeur, J. Hespanha, and D. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Trans. Patt. Anal. Mach. Intell.*, (7):711.

[2] Yizong Cheng. Mean shift, mode seeking, and clustering. *IEEE Trans. Pattern Anal. Mach. Intell.*, 17:790–799, August 1995.

[3] Rob Hess. An open-source siftlibrary. In *Proceedings of the international conference on Multimedia*, MM '10, pages 1493–1496, New York, NY, USA, 2010. ACM.

[4] Erik Hjelmas and Boon Kee Low. Face detection: A survey. *Computer Vision and Image Understanding*, 83:236–274, 2001.

[5] Rainer Lienhart and Jochen Maydt. An extended set of haar-like features for rapid object detection. In *IEEE ICIP*, pages 900–903, 2002.

[6] D Lowe. Distinct image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91 – 110, 2004.

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

[8] Paul Viola and Michael J. Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE CVPR*, 2001.

[9] Paul Viola and Michael J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, (2):137–154, 2004.

[10] Jun Zhang, Yong Yan, and Martin Lades. Face recognition: Eigenface, elastic matching, and neural nets. *Proceedings of the IEEE*, 85(9), 1997.

[11] W. Zhao, R. Chellappa, P. J. Phillips, and A. Rosenfeld. Face recognition: A literature survey. *ACM Comput. Surv.*, 35:399–458, December 2003.