Carnegie Mellon University

CARNEGIE INSTITUTE OF TECHNOLOGY

THESIS

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF Doctor of Philosophy

TITLE Pose Robust Video-Based Face Recognition

PRESENTED BY Xiaoming Liu

ACCEPTED BY THE DEPARTMENT OF

ADVISOR, MAJOR PROFESSOR — DATE

DEPARTMENT HEAD — DATE

APPROVED BY THE COLLEGE COUNCIL

DEAN — DATE

Abstract

Researchers have been working on human face recognition for decades. Face recognition is hard due to different types of variations in face images, such as pose, illumination and expression, among which pose variation is the hardest one to deal with. To improve face recognition, this thesis presents an integrated approach to performing *pose* robust *video-based* face tracking and recognition by using a face *mosaic* model. We approximate a human head with a *3D ellipsoid* model, where each face image is a projection of the 3D ellipsoid at a certain pose. In our approach, both training and test images are projected back to the surface of the 3D ellipsoid, according to their estimated poses, to form the *texture maps*. Thus the recognition can be conducted by comparing texture maps instead of the original images, as done in traditional face recognition. In addition, by representing the texture map as an array of local *patches*, we can train a *probabilistic model* for comparing corresponding patches. With multiple training images under different views, we are able to obtain a *statistical mosaic* model as well as a *geometric deviation model*, which not only reduces the blurring effect in the mosaic model, but also serves as an indication of how much the actual human faces geometry deviates from the 3D ellipsoid model. Furthermore, we apply the face mosaic model to video-based face recognition. The mosaic model is able to simultaneously track, register, and recognize human faces from video sequences. Finally, we also apply the *updating-during-recognition* scheme in using the mosaic model. This scheme allows the mosaic model to be updated during the test stage in order to enhance the modeling and recognition over time.

Declarations

Some part of the work presented in this thesis have been published in the following articles:

- Xiaoming Liu, Tsuhan Chen and Susan M. Thornton, "Eigenspace Updating for Non-Stationary Process and Its Application to Face Recognition", *Pattern Recognition*, special issue on Kernel and Subspace Methods for Computer Vision, Volume 36, Issue 9, pp.1945-1959, September 2003.
- Xiaoming Liu, Tsuhan Chen and B.V.K. Vijaya Kumar, "Face Authentication for Multiple Subjects Using Eigenflow", *Pattern Recognition*, special issue on Biometric, Volume 36, Issue 2, pp.313-328, February 2003.
- Xiaoming Liu and Tsuhan Chen, "Pose Robust Face Recognition Based on Mosaicing --An Example Usage of Face In Action (FIA) Database", Demo session of the IEEE International Conference on Computer Vision and Pattern Recognition 2004, Washington, DC, 27th June - 2nd July, 2004.
- Xiaoming Liu and Tsuhan Chen, "Geometry-assisted Statistical Modeling for Face Mosaicing", Proceeding of the IEEE International Conference on Image Processing 2003, Vol.2, pp.883-886, Barcelona, Spain, 2003.
- Xiaoming Liu and Tsuhan Chen, "Video-Based Face Recognition Using Adaptive Hidden Markov Models", Proceeding of the IEEE International Conference on Computer Vision and Pattern Recognition 2003, pp. 340-345, Madison, Wisconsin, June 16-22, 2003.
- Xiaoming Liu and Tsuhan Chen, "Shot Boundary Detection Using Temporal Statistics Modeling", Proceeding of the IEEE International Conference on Acoustics, Speech, and Signal Processing 2002, vol.4, pp.3389-3392, Orlando, Florida, May 13-17, 2002.
- Xiaoming Liu, Tsuhan Chen and B.V.K. Vijaya Kumar, "On Modeling Variations For Face Authentication", Proceedings of the International Conference on Automatic Face and Gesture Recognition 2002, pp. 384-389, Washinton D.C., May 20-21, 2002.
- Tsuhan Chen, Yu-Feng Hsu, Xiaoming Liu, and Wende Zhang, "Principle component analysis and its variants for biometrics", Proceedings of the IEEE International Conference on Image Processing 2002, Vol.1, pp. 61-64, Rochester, NY, Sep. 2002.

Acknowledgments

I am greatly indebted to my advisor, Prof. Tsuhan Chen, who guided me through my whole PhD study. I could not but deeply thankful for his continuous support, encouragement and great enthusiasm. His insights have been very helpful in guiding me to find the right direction and make progress in my research. He has always encouraged me to stay focused on my research and persevere with my ideas, especially when the going was tough. He himself is a great teacher and speaker. Through our group meetings and many interactions, I have learned many lessons on how to be a good speaker and presenter, which to me will be very valuable for my future career. Being his student, I also learn a lot from his enthusiasm for research, work, and life. I believe all of these will benefit not only my career but also my life. I have to say that I am really fortunate to have had him as my advisor.

I would like to thank my second advisor, Prof. B. V. K. Vijaya Kumar for his encouragement during my study. His insights have greatly enabled me to have a deep understanding about my research problems and to explore new ideas. I would also like to thank Dr. Jie Yang and Dr. Zhengyou Zhang for their interest in my work, their patience and their valuable time in being part of my thesis committee. I am grateful to them for their many insights and comments that have led to the improvement of this work.

I am thankful to all my group mates and friends during my study at Carnegie Mellon University. My group mates, Ta-chien Lin, Deepak Turaga, Fu Jie Huang, Trista Chen, Howard Leung, Jessie Hsu, Cha Zhang, Claire Fang Fang, Ed Lin, Sam Chen, Wende Zhang, Simon Lucey, Kate Shim, Jack Yu, Avinash Baliga, Michael Kaye, David Liu, Kubota Akira, and Todd Stephenson have been spending time and providing valuable feedbacks to my work. Thank my friends Depeng Wu, Jingfeng Liu, Hongwei Song, Yin Sun, Jin Lu, Haotian Zhang, Chanchai, and Liu Ren, for bringing the happy moments.

Finally, I would like to express my earnest gratitude to my wife and parents for their love and support. In particular, I am deeply indebted to my wife, Bing Yu. Without her sacrifice, support and encouragements, there would never have been any chance for this thesis to happen.

List of Figures

List of Tables

Table of Contents

1. Introduction

For decades human face recognition has been an active topic in the field of object recognition. A general statement of this problem can be formulated as follows: given still or video images of a scene, identify one or more persons in the scene using a stored database of faces [10]. A system that performs face recognition has many applications, such as nonintrusive identification and authentication for credit card usage, nonintrusive access control to buildings, and identification for law enforcement.

Comprehensive surveys of human and machine recognition techniques can be found in [10][1][20][79]. A lot of algorithms were proposed to deal with the image-to-image, or image-based, recognition where both the training and test sets consist of still face images. There are two basic kinds of face recognition algorithms: one is based on the feature matching, such as Elastic Graphic Matching [40]; the other is based on the template matching, such as the eigenface approach [72], and Linear Discriminate Analysis (LDA) [2]. In the latter, the eigenface approach, which applies Principal Component Analysis (PCA) in the pixel domain, plays a fundamental role. It is widely considered as the baseline of many face recognition algorithms. It has the advantage of fast computation, stable performance for the case of frontal face recognition with constraints on illumination, expression variations, etc. However, with existing approaches, the performance of face recognition systems in practice is affected by different types of variations, for example, expression, illumination, and pose.

At least two observations have been made from the previous extensive studies. First, face recognition is to deal with variations. Researchers have studied how face recognition is affected by different kinds of variations, such as expression [74][55][50], illumination [1], pose [78][58], aging [41], and sunglasses [80]. Among these, pose variation is the hardest one to model and therefore contributes most of the recognition error [20][61]. Because pose variation results not only in shape variation, but also in appearance variation due to the changing relation between the illumination source and the face. For example, as shown in Figure 1, one of the results from Face Recognition Vendor Test (FRVT) 2002, the recognition rate of pose variation is much lower than that of illumination variation. Second, face registration is the key of face recognition. This observation is a direct consequence of the first one. In dealing with different variations, if we could register face images into the canonical model, the recognition task would be simpler. In traditional image-based face recognition, the face area is normally cropped before feeding it into the recognition module. The importance of face registration has been overlooked in the literature. However, in video-based face recognition, the face portion has to be registered from the video frame before any recognition can take place.

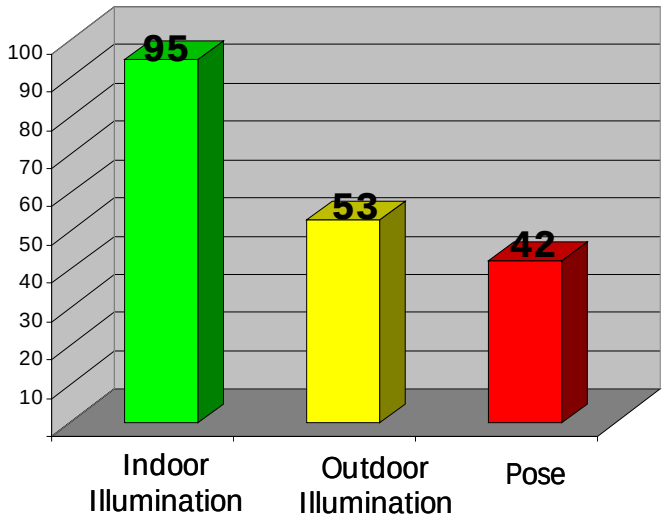

Figure 1 Experimental results of FRVT 2002 on the recognition rate of different variations.

1.1 Our approaches

In this thesis, we propose an integrated approach to performing video-based pose robust face tracking and recognition using a face mosaic model.

As motivated by the research on video mosaics [68] and fingerprint mosaicing [37], we propose to model the facial appearance by constructing a mosaic model from multiple faces at various poses. Traditionally, the pose variation is very difficult to model. We propose to use the geometry of a face to improve the mosaicing result. By approximating a human head with a 3D ellipsoid, each face image is the result of projecting the ellipsoid's certain portion on the image plane. Given a number of face images under various poses, as shown in Figure 2, we map the face portion of each frame onto the surface of the ellipsoid using the geometric mapping algorithm. Unwrapping the surface of the ellipsoid will result in a texture map, which has a α and β coordinate system. In the mean time, instead of one single texture map, a statistical model composed of a mean image and a number of eigen-images, is trained by using the *unwrapped* texture maps.

In this thesis, we first present how to perform pose robust face recognition using geometry-assisted probabilistic modeling. In our approach, all training and test images are projected to the surface of a 3D ellipsoid by estimating the optimal pose and position information, and represented as texture maps. The distance measure is calculated in the overlap area between texture maps of the training and test images. Also by representing a texture map as an array of local patches, it enables us to develop a probabilistic model for comparing corresponding patches from a face database with pose variation. We study how the discriminative power of corresponding patches varies for different poses. Eventually, we are able to utilize the Bayesian framework to evaluate the distance measure of corresponding patches.

Second, we combine multiple images with pose variation to build a statistical mosaic model, which is used for face recognition. In our mosaic method, combining multiple images is essentially combining multiple texture maps. Since the same facial feature, such as the mouth corner, found in multiple maps might not correspond to the same coordinate on the single texture map, the blurring effect would be observed when we combine multiple texture maps. To reduce such blurring, one key idea in our approach is to allow a patch to move locally toward better corresponding across multiple maps, use the flow representation for modeling the amount of movement, and train the flow representation to form a geometric deviation model via PCA. The benefit of this approach is that while we are obtaining a less-blurring facial appearance model from multiple views, we also form a geometric deviation model, which models how the actual geometry of each individual subject deviates to the 3D ellipsoid model. It is important to use two models: one for appearance variations and another for geometric deviation, especially when a rough 3D ellipsoid model is used as the face geometry. We show that both the appearance and the geometric model are useful for face recognition.

Third, since our face mosaic model is a simple statistical model combining both appearance and geometric information, we apply it for performing face tracking and recognition simultaneously. Given a test face sequence, we can track faces using the condensation method [28] or the Levenberg-Marquardt algorithm [63], based on a face mosaic model. Both algorithms are trying to estimate the optimal mapping parameter in order to minimize the distance measure between the test image and the model. Face tracking and recognition can be performed simultaneously by using the condensation framework.

Fourth, due to a limited number of training images, usually we cannot train a face mosaic model containing enough statistical information. To deal with this issue, we apply the updating-during-recognition scheme [51] in video-based face recognition. That is, by taking more images from a test sequence, which contains pose variation or expression that have not been seen, as the

training data, our mosaic model can be enhanced and eventually results in a better recognition system.

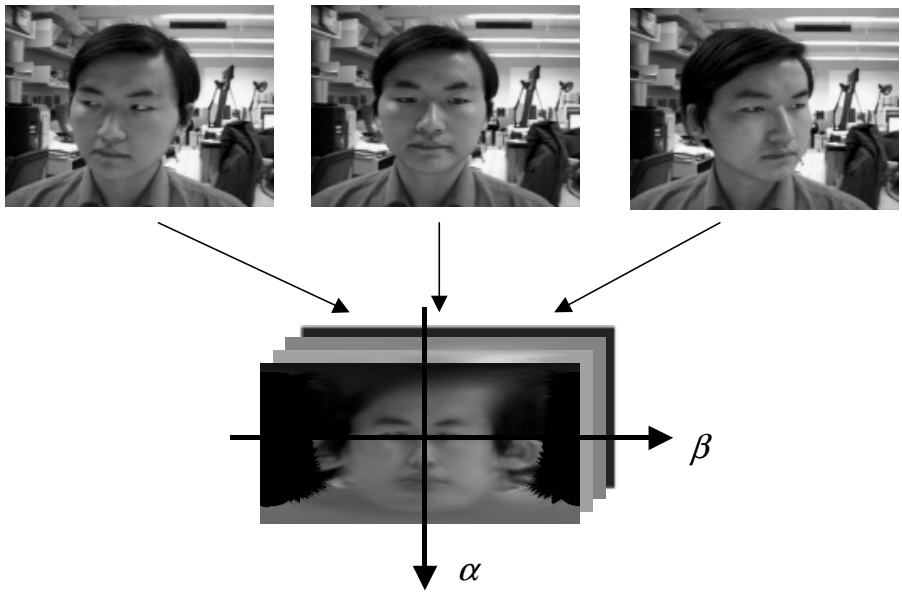

Figure 2 Generating a statistical face mosaic model from multiple images with different poses.

In literature, a few papers propose techniques similar to face mosaicing [11][42]. Compared to them, our method has a number of novelties. First is that instead of using the cylindrical projection, we use the ellipsoidal projection, which works more naturally with the head motion in both horizontal and vertical directions. Second is that while traditional mosaic algorithms usually result in only one texture map image, our method generates one statistical model with both the mean image and a number of eigen-images, which provides a more sophistical statistical model to represent different types of variations, compared to using only one template image. Third, we represent the mosaic as a set of patches and learn a probabilistic model for the similarity measure between the corresponding patches of the mosaic model and the test image. Fourth, while traditional mosaic algorithms assume a planar relation among multiple images, we use the ellipsoid model together with the deviation modeling, which results in better matching among multiple texture maps.

Comparing to other approaches in pose robust face recognition, we can imagine there is one dimension measuring how much the face recognition algorithm are relying on the geometric information. The approach of modeling dynamic and using mapping functions are on the extreme end of this dimension since they do *not* use any geometric information. While Blanz and Vetter's work [5] is on another extreme end of this dimension because they use a very sophistical shape model. Kanade and Yamada [32]'s work use a little geometric information because they register the face based on three facial features. Comparing to them, our approach uses less geometric information than Blanz and Vetter's work, but more than Kanade and Yamada's work. Researchers always assume that the better modeling leads to the better recognition performance. However, the price we have to pay for a more sophistical modeling is that the model fitting will become too difficult. For example, in [5], both the training and test images are manually labeled with 6 to 8 feature points. On the other hand, we believe that unlike the rendering applications in computer graphics, we might not need a very sophistical geometric model for face recognition applications. The benefit with a simpler geometric model is that the fitting will tend to be easier and automatic, which is the goal of our approach. Although our approach uses more geometric information than [32], which needs to track three feature points for any test images, we consider our approach can fit the model more reliable since we use the appearance information of all face portion to register the face according to the model. Compared to AAM [16][56], our mosaic model can model much large pose variations because in AAM none of the meshes can be occluded during the model fitting, which greatly constrains the possible pose variation.

Comparing to Zhou et. al.'s work [82], our approach uses a sophistical face mosaic model which can take care of pose variation better than the traditional PCA model used in [82]. In addition, we utilize the idea of statistical learning in updating the face mosaic model, which greatly improves the face recognition, especially when there are very few training images to begin with.

1.2 Thesis structure

The remaining of the thesis is organized as follows.

Chapter 2 introduces the background of human face recognition. We survey the previous work on three major problems in face recognition: template based face recognition, pose robust face recognition, and video based face recognition.

In Chapter 3, we introduce how to generate a texture map from a face image based on a known mapping parameter. We present a method of learning a probabilistic model for comparing corresponding patches from a face database with pose variation, and how to apply it for pose robust face recognition. In the experiments, we show that the probabilistic model can improve the pose robust face recognition. Comparing with the baseline algorithm, we observe a significant improvement when performing experiments on the CMU PIE database.

Chapter 4 presents a method of combining multiple training images for training a statistical mosaic model. A geometric deviation model is trained in order to have a better matching among patches from multiple texture maps. We show an improvement of using the deviation model from pose robust face recognition experiments.

Chapter 5 introduces the mosaic based face tracking and recognition from video sequences. Given a face mosaic model and a test sequence, we introduce two methods of performing face tracking: the condensation method and the Levenberg-Marquardt algorithm. We also present our effort in collecting a face video database. Experimental results of both tracking and recognition from video sequences are shown.

Chapter 6 presents how to apply an updating-during-recognition scheme in using the mosaic model. Different methods of subspace updating are presented. We show that by using the updating, pose robust face recognition can be greatly improved based on only one front training image.

Finally, Chapter 7 concludes this thesis and point out the contributions. Also we provide interesting extensions for the work described in this thesis.

2. Background

Human face has been an interesting research topic for decades. Many promising topics are explored based on the interaction between the face and the computer, such as face modeling [81], face animation [42], face rending [6], face detection [77][66][38], and face recognition [10][1][20][79].

As we mentioned before, comprehensive surveys of human and machine face recognition techniques can be found in [10][1][20][79]. Thus, this chapter does not intend to give a detailed survey of all previous work in the human face recognition. Rather, we would like to focus our attention on three major problems in face recognition: template based face recognition, pose robust face recognition, and video based face recognition.

2.1 Template based face recognition

Human face recognition has a long history in the vision community. The first major attempt is made in Kanade's Ph.D thesis in 1973 [31], which tries to recognize faces via the distribution of facial feature points. There are two basic kinds of face recognition algorithms: one is based on the feature matching, such as Elastic Graphic Matching [40]; the other is based on the template matching, such as the eigenface approach [72], Linear Discriminate Analysis (LDA) [2]. The eigenface approach has the benefit of fast computation, easy implementation and good performance in normal conditions. Since the birth of the eigenface approach, the template based

approach has become more dominant than the feature based approach. In the later chapters, we also use the eigenface approach as one of the baseline algorithms. The details of this algorithm are presented in this section.

Suppose there are M training face images for each of the K subjects. Let each face image $I(x, y)$ be a 2-dimensional N-by-N array of pixel values. An image may also be represented (after scanning) as a vector of dimension N^2, where each image corresponds to a single point in the N^2-dimensional image space. Let us denote each face image of the training set as $\mathbf{f}_{ij}$, a $N^2 \times 1$ vector, where i and j denote the subject index and the face image index respectively, and $0 \le i \le K-1$, $0 \le j \le M-1$. The average face vector $\mathbf{g}$ is defined by

$$\mathbf{g} = \frac{1}{M \times K} \sum_{i=0}^{K-1} \sum_{j=0}^{M-1} \mathbf{f}_{ij}$$

The difference between each training face and the average is denoted by the vector $\mathbf{s}_{ij} = \mathbf{f}_{ij} - \mathbf{g}$. These difference vectors form a $N^2 \times MK$ matrix, $\mathbf{A} = [\mathbf{s}_{00}, \mathbf{s}_{01}, ..., \mathbf{s}_{K-1,M-1}]$. We apply PCA to these difference vectors by finding a set of Q orthonormal eigenvectors, $\mathbf{u}_n$, corresponding to the largest eigenvalues of the matrix $\mathbf{A}\mathbf{A}^T$, i.e.,

$$\mathbf{A}\mathbf{A}^T \mathbf{u}_n = \lambda_n \mathbf{u}_n, \qquad \text{n=0,1,…, } Q-1, \tag{1}$$

where λ_0, $\lambda_1, \ldots, \lambda_{Q-1}$ are nonnegative and in a decreasing order. However, the matrix $\mathbf{A}\mathbf{A}^T$ is N^2 by N^2, and determining N^2 eigenvectors can be computationally intensive. Usually the number of training faces, $M \times K$, is much smaller than N^2. So we first determine the eigenvectors, $\mathbf{u}_n'$, of a $MK \times MK$ matrix $\mathbf{A}^T\mathbf{A}$, i.e.,

$$\mathbf{A}^T \mathbf{A} \mathbf{u}_n' = \lambda_n \mathbf{u}_n' \tag{2}$$

Pre-multiplying (2) by $\mathbf{A}$ and comparing to (1), we can see that $\mathbf{u}_n = \mathbf{A}\mathbf{u}_n' \lambda_n^{-1/2}$. These eigenvectors form an orthonormal basis set of a new feature space, called the eigenspace.

Essentially it is a subspace representation of all the faces. Thus, we can transform each face image, $\mathbf{f}_{ij}$, from the image space to the eigenspace as follows:

$$w_n = \mathbf{u}_n^T(\mathbf{f}_{ij} - \mathbf{g}) \qquad \text{n=0,1,…, } Q-1 \tag{3}$$

Each face image can be described as a vector $\mathbf{p}_{ij} = [w_0, w_1, ..., w_{Q-1}]^T$ in the eigenspace. During the test stage of the recognition system, a test face image $\mathbf{f}$ is projected to the eigenspace, as in (3), to obtain the projected vector $\mathbf{p}$. Then the nearest neighbor classifier is used to determine a subject whose $\mathbf{p}_{ij}$ has the minimal distance to $\mathbf{p}$.

The eigenvector determination can be computationally expensive when the number of training images is large. The power method [26] is one approach to efficiently determining the dominant eigenvectors. Instead of determining all the eigenvectors, the power method obtains only the dominant eigenvectors, i.e., the eigenvectors associated with the largest eigenvalues.

Essentially the eigenspace provides a low dimensional linear subspace for describing the facial appearance. All recognition tasks are performed in this subspace instead of the original pixel domain. However, the objective in the dimension reduction is to best represent the original data set in the mean squared sense, which might not be optimal in terms of classification. This observation leads to one direction of improving the role of PCA in face recognition: to treat classification as the criterion of constructing a subspace.

Fisherface [2] and its subsequent work [12][77] are one attempt along this direction. They basically combine PCA and LDA to generate a subspace that is optimal in terms of linear classification. Another attempt is to use kernel methods together with PCA [47][35] or LDA [48] to learn a subspace.

The second direction of extending PCA is to apply PCA on feature domains other than pixels. For example, we can apply PCA on the optical flow between two images, which results in an expression robust face recognition system [50]. Chung et. al. [15] apply PCA on the Gabor filter responses, and the new algorithm works better for illumination and pose variation.

2.2 Pose robust face recognition

As we mentioned before, among the variations that have been extensively studied, pose variation is the hardest. Let us review the previous work in dealing with the pose variation in face recognition.

There are different types of approaches for pose robust/invariant face recognition. The first type of approach is to learn the dynamics/trajectories from images with continuous pose variation. And then such trajectories can be used for recognizing faces from image sequences [43][3][52]. The trajectory is represented by either a curve or a surface. Notice this is also one typical approach in the literature of video-based face recognition. One drawback with these approaches is that certain application scenario, where the subject shows consistent motion in both training and test video sequences, has to be assumed, in order to make the *dynamic* to be meaningful. This assumption is not true in general, but might be true for specific tasks, which limits the popularity of this type of approaches.

The second type of approach is to treat the whole face image under a certain pose as one sample in a high dimension space, and learn the relation between a front pose image and non-front pose images by constructing a mapping function between them. Given a test image with an arbitrary pose, a recognition-by-synthesis approach can be applied. That is, we can either transform this test image into the front view [43], or transform each of the training images into the same pose as the test image [58], based on the learned mapping function. One potential problem with this type of approaches is that it is not clear whether the relation among multiple pose images could be approximated as a simple function, such as a linear transformation [43]. The eigen light-field method [23] can also be classified as this type of approach.

Since a face image is pretty complex and different parts of the face might transform in a different manner under varying poses, researchers start to look at faces as a set of parts/patches [54][32]. Kanade and Yamada [32] conduct a systematic analysis on how the discriminative

power of different parts on human faces changes according to different poses, and such analysis leads to a probabilistic approach to pose invariant face recognition. In this thesis, we propose to use the patch representation for texture maps. There are at least three benefits of using the patch representation instead of the original texture maps. First, the patches representation enables use to build a probabilistic model for the similarity between corresponding patches. Thus each patch could be treated differently according to its discriminative power. Second benefit is that the variation of the facial patch is simpler and thus more likely to be modeled with a certain function or a probabilistic model. Third, when multiple texture maps are combined together, the patches are allowed to move locally in order to have a better corresponding among multiple texture maps, which compensates the not-perfect assumption on the 3D ellipsoid geometry of the human head.

Since we are dealing with pose variation, which is a result of the human head's geometry projected differently, naturally researchers would rely on the geometric information to aid the pose invariant face recognition. If we imagine there is a dimension indicating how much geometric information is used for recognition, we can place many algorithms along this dimension. Many algorithms, such as the ones in the first and second type of approaches, do not use any geometric information. Others do make use of geometric information by assuming a particular head model. For example, a cylinder head model is widely used in face tracking [9][7]. However, the cylinder model does not act naturally for head nod. Thus we propose a spherical head model to enhancing the modeling [53]. In this thesis, we will use a 3D ellipsoid as the head model based on the consideration that the human head does have different width, height, and depth. Blanz and Vetter's approach [5] is in the extreme end of this dimension since they use perfect 3D geometric information of human heads. Based on a large set of face images, they train two subspace models for facial texture and shape respectively. Given a test image under any pose and lighting, they can fit the image with two models by tuning the coefficients in the models. Finally, the model coefficients are used for recognition.

Researchers also take care of pose variation via face registration. One common way in face registration is to detect the facial features, such as eyes, mouth, etc, and apply transforms based on the corresponding facial features in images under different poses [3]. Active Appearance Models (AAM) [16][56] has been used for face tracking and recognition, where the face is modeled by a triangulated mesh structure. Once the mesh could be fitted to a face image, this face image is registered with a canonical model. Of course, [5] is also a sophistical approach of registering face images.

2.3 Video-based face recognition

To improve face recognition, recently the researchers start to look at video-based face recognition [44][18][82][51], where the test images are video sequences containing faces. Recent psychophysical results show that a human makes use of facial motion information for face recognition [70]. Video-based face recognition has several advantages over image-based face recognition. First, the geometric information can be explored given continuous video sequences showing different poses of human faces, which helps to handle pose variation. Second, the motion information of faces can be utilized to facilitate the recognition task. For example, the subject-dependent dynamic characteristics can help face recognition [52]. Third, given the fact that in video sequences most of face variations are present in a continuous fashion, video-based recognition allows the learning or updating of the subject's model over time. For example, we propose an updating-during-recognition scheme, where the current and past frames in a video sequence can be used to update the subject's models to improve recognition results for future frames [51]. Furthermore, most practical face recognition systems actually take video sequences under certain scenario as the input. Thus, it is very natural to take advantage of the video information from the input, instead of just selecting certain frames and performing image-based recognition.

There are not much previous approaches can be considered as video-based recognition yet. Most of the previous video-based algorithms simply apply image-based recognition to each frame, and average the frame based scores to obtain the final similarity between the sequence and the model. We still consider this type of approaches as image-based recognition.

One type of video-based recognition is to model the dynamics in video sequences [43][3][52], where face tracking and recognition are two separate steps. Normally face recognition is performed after the tracking is finished, which still ignores the face registration issue. Recently, another trend in video-based face recognition is to simultaneously perform face tracking and recognition given a test sequence. For example, Zhou et. al. [82] apply the condensation method for video-based face recognition. In their case, since a simple PCA model is used in modeling facial appearance, the tracking performance in dealing with pose variation will be affected.

3. Face Recognition Using Geometry-Assisted Probabilistic Modeling

As we mentioned before, the most difficulty variation for face recognition is pose variation. The difficulty is that the intra-subject variations are as large as, or even larger than the inter-subject variations when pose variation is present. To improve face recognition under pose variation, we present a geometry-assisted probabilistic approach. We approximate a human head with a 3D ellipsoid model, where any face image is a projection of such a 3D ellipsoid at a certain pose. In this approach, both training and test images are projected back to the surface of the 3D ellipsoid, according to their estimated poses, to form texture maps. Thus the recognition can be conducted by comparing the texture maps instead of the original images, as done in traditional face recognition. The geometrical mapping could be treated as one way of compensating pose variation and reducing the intra-subject variations. In addition, we represent a texture map as an array of local patches, which enables us to train a probabilistic model for comparing corresponding patches.

In this chapter, we first introduce how to generate a texture map from a face image based on a known mapping parameter. Then we present a method of learning a probabilistic model for comparing corresponding patches from a face database with pose variation, and how to apply it for pose robust face recognition. In experimental results, we show that the probabilistic model can improve the performance of pose robust face recognition.

3.1 Geometrical mapping

If we compare two face images of the same subject captured at two different view angles, the pixel-by-pixel difference is relative big because these two images are not registered/aligned with respect to each other. This is also the reason why the traditional eigenface approach does *not* work well for face images with pose variation. Image registration is a way to fix this problem, i.e., the comparison should only be conducted *after* two images are registered. Considering the fact that a human head has the non-planar geometry, one way to register face images is to project them back to the surface of a 3D ellipsoid from each of the specific poses. This procedure of projection is called *geometrical mapping*.

Geometrical mapping is a key component in our proposed algorithm. In this section, we introduce how to generate a texture map $\mathbf{s}$ from a face image $\mathbf{f}$, given a known match parameter $\mathbf{x}$. In the following chapters, we will present how to estimate the mapping parameter $\mathbf{x}$ by various methods.

Three assumptions are made. First, a human head is a 3D ellipsoid with radius to be r_x, r_y, and r_z. Second, a face image is captured with a weak perspective camera model [20] and a camera focal length equals to one. Third, all images are captured under the ambient lighting environment. Under these assumptions, we use a mapping parameter $\mathbf{x}$ to describe the relation between a face image and its corresponding texture map. The mapping parameter $\mathbf{x}$ is a 6-dimensional vector $\mathbf{x} = [c_v \quad c_h \quad d \quad R_\alpha \quad R_\beta \quad R_\chi]^T$, where c_v and c_h indicate the center of the face area in the face image, d indicates the average distance between the face and the camera, and R_α, R_β and R_χ indicate the rotation of the human head with respect to the XYZ axis. As we can see, the mapping parameter $\mathbf{x}$ includes all the information for locating a face, as well as generating the texture map from the face image.

Let a human head centered at the origin of an XYZ coordinate system and the front face look at the positive Z axis. Thus different views of human faces can be obtained by fixing the camera and rotating the human head with certain degrees in various directions. To generate a texture map **s** from **f**, essentially for each pixel, $\mathbf{s}(\alpha,\beta)$, we need to find its corresponding coordinate, $\mathbf{f}(v,u)$, by knowing the mapping parameter **x**, which is followed by a bilinear interpolation [36] to fill in the intensity of pixel $\mathbf{s}(\alpha,\beta)$. The parameters v and u are the axis of the original image; α and β are the axis of the texture map. As shown in Figure 3 and Figure 4, there are basically two steps for this mapping.

First, a pixel $\mathbf{s}(\alpha,\beta)$ in the texture map corresponds to one point (p_x, p_y, p_z) on the surface of a sphere, whose radius is one:

$$\begin{aligned} p_x &= \sin(\alpha)\sin(\beta) \\ p_y &= \cos(\alpha) \\ p_z &= \sin(\alpha)\cos(\beta) \end{aligned} \tag{4}$$

As shown in the right illustration of Figure 3, the sphere is then converted into an ellipsoid by stretching each radius according to r_x, r_y, and r_z:

$$\begin{aligned} p_x &= r_x p_x \\ p_y &= r_y p_y \\ p_z &= r_z p_z \end{aligned}$$

Second, we can rotate the head ellipsoid by R_α, R_β and R_χ with respect to the XYZ axis. As shown in Figure 4, the point on coordinate (p_x, p_y, p_z) moves to a new coordinate (p'_x, p'_y, p'_z) by the following equation.

$$\begin{pmatrix} p'_x \\ p'_y \\ p'_z \end{pmatrix} = \begin{bmatrix} \cos(R_\chi) & \sin(R_\chi) & 0 \\ -\sin(R_\chi) & \cos(R_\chi) & \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(R_\alpha) & -\sin(R_\alpha) \\ 0 & \sin(R_\alpha) & \cos(R_\alpha) \end{bmatrix} \begin{bmatrix} \cos(R_\beta) & 0 & \sin(R_\beta) \\ 0 & 1 & 0 \\ -\sin(R_\beta) & 0 & \cos(R_\beta) \end{bmatrix} \begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix}$$

Then we project the coordinate (p'_x, p'_y, p'_z) onto the image plane by using the weak perspective camera model, and translate the resulting coordinate by c_v and c_h in both vertical and horizontal directions.

$$\begin{cases} v = \dfrac{p'_y}{d} + c_v \\ u = \dfrac{p'_x}{d} + c_h \end{cases} \tag{5}$$

Finally, we obtain the new coordinate (v, u) in the image coordinate. By judging whether (p'_x, p'_y, p'_z) is facing the positive Z axis or not, it can tell us whether (v, u) is a valid coordinate in the image plane. If it is, the bilinear interpolation result of (v, u) is filled in as the intensity of the pixel $\mathbf{s}(\alpha, \beta)$. Otherwise $\mathbf{s}(\alpha, \beta)$ is considered as a missing pixel and its intensity is set to be zero. To compensate the lighting variations, we also normalize the mean of the intensity of all non-missing pixels to be 128.

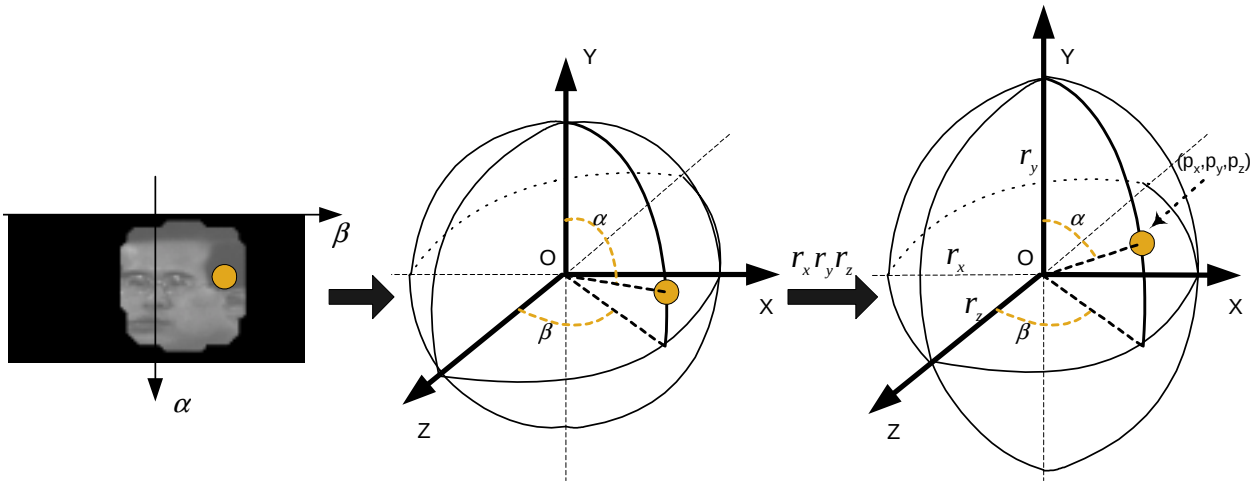

Figure 3 Geometric mapping: the corresponding between one pixel on the texture map and one point on the surface of the ellipsoid.

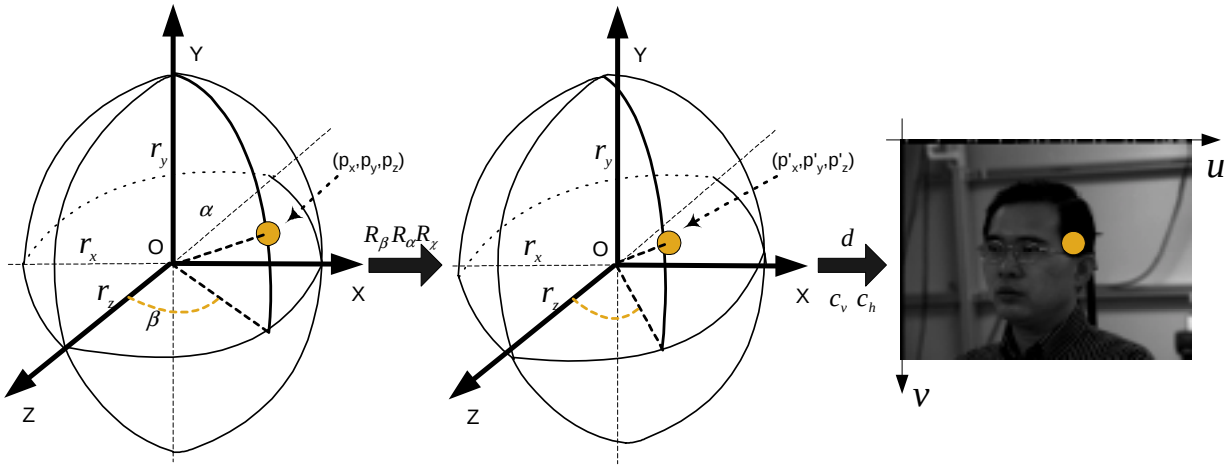

Figure 4 Geometric mapping: rotate the ellipsoid and obtain the corresponding pixel on the image plane.

One issue in the above mapping is how to determine the radius of a human head ellipsoid, r_x, r_y, and r_z, which is essentially the height, width and depth of the human head. Since we already include d in the mapping parameter, any one of the three radiuses, for example, the width r_x, can be set to be one. Thus we only need to determine the ratio between the width and the depth, and the ratio between the width and the height. In our algorithm, the former is set to be a fix constant 0.9 by considering that the head's depth is slightly larger than the head's width, while the latter is usually obtained from the external sources, such as a face detector or manual labeling of a front face image. Once we obtain these two ratios, they are assumed to be constant for the same subject. Of course, we can also treat these two ratios as two additional elements in the mapping parameter $\mathbf{x}$, and estimate them using the same framework of estimating $\mathbf{x}$, which will be introduced in future chapters.

Since the generation of the map is an essential step in our mosaicing algorithm, the efficiency of this step will affect the speed of face tracking/recognition. This step can be computationally intensive if every pixel in $\mathbf{s}$ needs to find its corresponding position in $\mathbf{f}$. To solve this problem, we approximate the mapping using a triangular mesh, as shown in Figure 5.

That is, the texture map **s** is represented as a set of triangles; for the vertexes of these triangles, we derive their corresponding coordinates in **f** using the above mapping equations. Then the mapping between two triangles can be approximated by an affine transformation, whose six parameters are estimated via three corresponding vertexes. For the pixels inside each triangle, the scan-line algorithm [73] can be used to quickly find the corresponding pixels. For example, in Figure 6, for each triangle in the destination texture map **s**, the corresponding pixels of a vertical column is lying on a line in the triangle of the source image **f**, whose slope, (d_x, d_y), is determined via the affine transformation. The goal of this approximation is to speed up the geometrical mapping while not noticeably affecting the recognition performance. The choice of triangle size is a trade-off between the mapping speed and the mapping precision. If the triangle is larger, the mapping is faster while the precision is also lower. If the triangle is too small, we do not gain much in speeding up the geometrical mapping. In our implementation, the triangle size is 4 by 4 pixels.

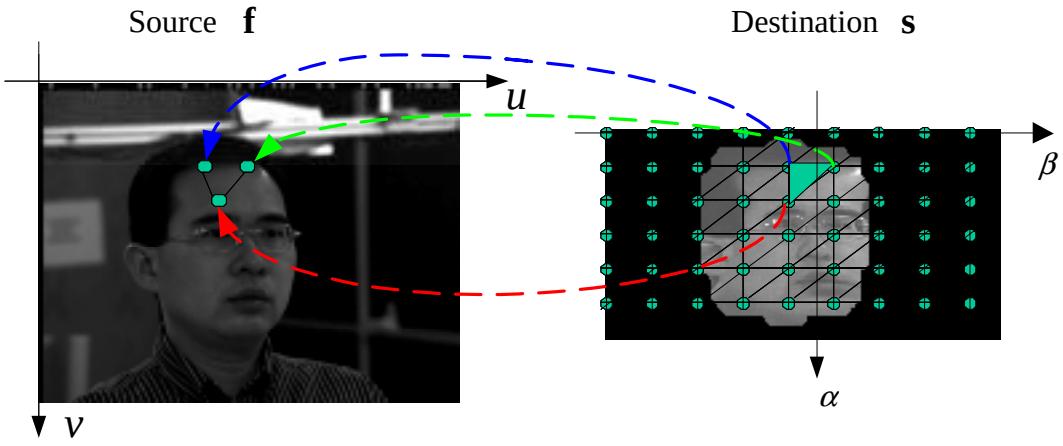

Figure 5 Triangle representation for speeding up the texture mapping

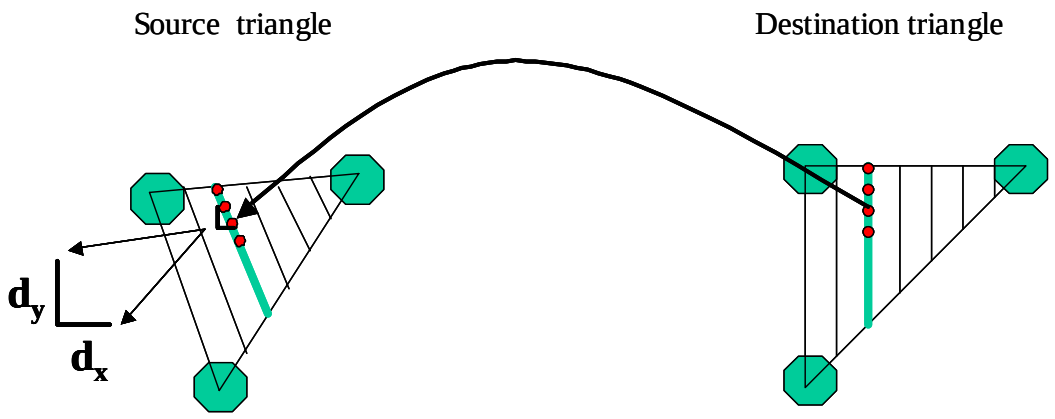

Figure 6 Scan-line algorithm: finding the corresponding pixel of one line in the destination triangle is equivalent to scan one line in the image place, whose slope is determined by the affine transformation parameters between these two triangles.

3.2 Geometry-assisted face recognition

In many face recognition systems, there is only one face image, normally the front view face image, during the training stage. However, in the test stage, there might be test images that correspond to different poses of human faces. This is a hard problem because the same subject looks very different under various poses. In this section, we present our geometry-assisted approach to deal with this case.

As shown in Figure 7, given a face database with L subjects, there is only one front view image, $\mathbf{f}_l\,(l=1,2,\cdots,L)$, for each subject that is available for training. During the training stage, we estimate the optimal mapping parameter $\mathbf{x}_l$ for each training image $\mathbf{f}_l$ based on a universal *mosaic* model, which will be described in the next chapter. Essentially this optimization process is trying to minimize the difference between the universal mosaic model and the texture map controlled by the mapping parameter, which provides the information about the position of the face, the distance of the face, and the pose. Notice that some of the parameters might be known from external sources. For example, if we know all training images are front view, the pose parameters, R_α, R_β and R_χ, are known to be zero. Once the optimization is done, the

corresponding texture map $\mathbf{s}_l$ is generated from each training image $\mathbf{f}_l$. It is obvious that in the texture map $\mathbf{s}_l$, only part of the pixels are valid information of the appearance, while the rest are missing pixels since each face image only corresponds to one portion of the 3D human ellipsoid's surface. To describe this missing pixel information, we also generate a mask map, $\mathbf{a}_l$, which has the same dimension as the texture map $\mathbf{s}_l$. For all missing pixels in $\mathbf{s}_l$, the corresponding pixel in $\mathbf{a}_l$ is zero and the others are one.

During the test stage, given one test image $\mathbf{f}_t$, first we estimate the optimal mapping parameter based on the universal mosaic model. Second, the resulting texture map $\mathbf{s}_t$ and mask map $\mathbf{a}_t$ are compared with each of the training texture maps as the following:

$$d_l = \frac{1}{\left\|\mathbf{a}_t \circ \mathbf{a}_l\right\|}\left\|(\mathbf{s}_t - \mathbf{s}_l) \circ \mathbf{a}_t \circ \mathbf{a}_l\right\|^2 \tag{6}$$

where $\circ$ refers to the element-wise multiplication. Basically d_l is the normalized mean-square-error between the overlap area of the test texture map $\mathbf{s}_t$ and the training texture map $\mathbf{s}_l$, and $\left\|\mathbf{a}_t \circ \mathbf{a}_l\right\|$ indicates the size of the overlap area between two texture maps. There is a degeneration case when the two texture maps have a very small overlapping area, which leads to small d_l. Because in our estimation algorithm, the mapping parameter changes slowly, there is a very low chance that we will fall into this degeneration case. Eventually, the test image is recognized as the subject with the minimal d_l.

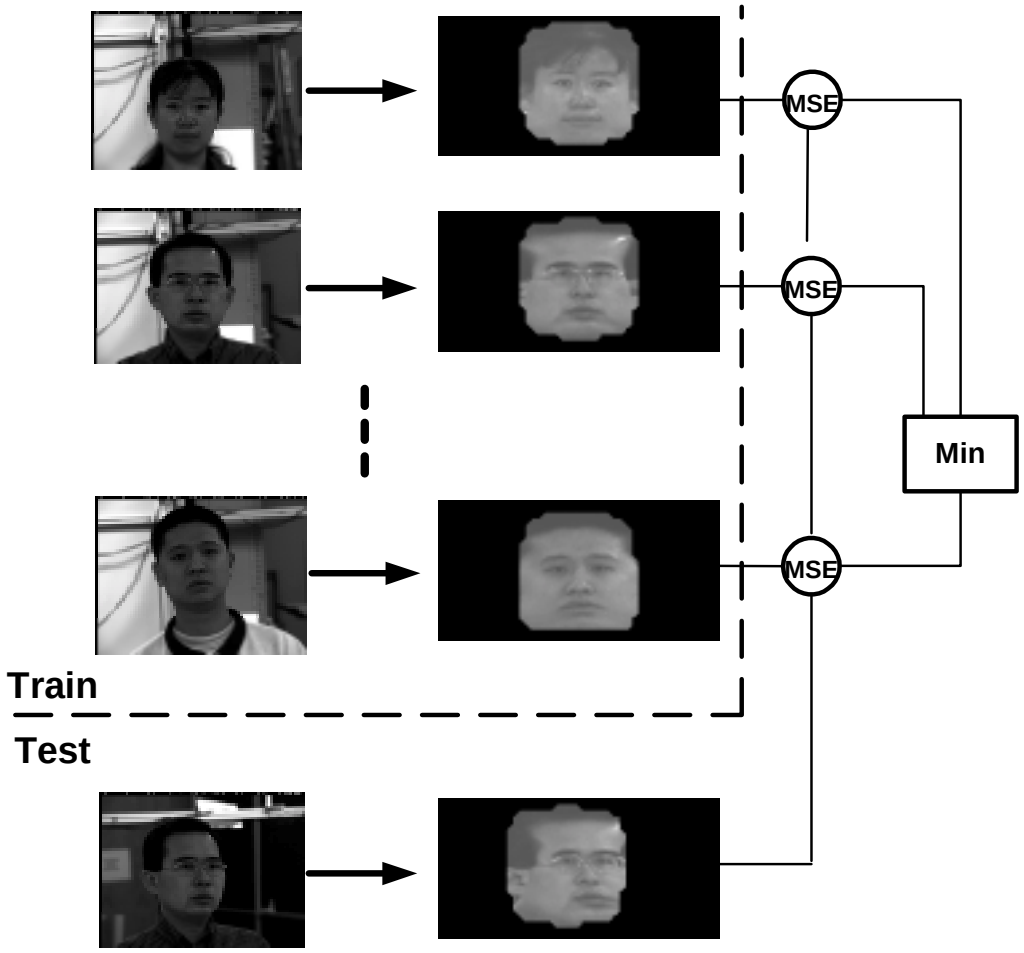

Figure 7 Geometry-assisted face recognition: all training and test images are converted into the texture map, and the distance measure is calculated based on the overlap area between two texture maps.

3.3 Probabilistic modeling for patches

Researchers have considered that different parts of a human face contribute differently to face recognition. For example, Pentland et. al. [58] propose to use modular eigenspaces to model the appearance of facial features, such as eyes, mouth, etc. Kanade and Yamada [32] perform discriminative analysis for all sub-regions in a face area and obtain a pose robust face recognition algorithm.

We extend the idea of sub-region analysis and applied it to the geometry-assisted approach. As shown in Figure 8, for each texture map $\mathbf{s}_l$, we represent it as an array of local patches $\mathbf{s}_{i,j}^{l}$. There are a number of benefits of using the patch representation instead of the original whole texture map. First, when combining different texture maps from multiple poses to generate a map that covers larger pose views, patches can be moved locally to find better matching with other poses. Hence the moving of local patches compensates when the assumption of the ellipsoid human head is not perfect. We will further utilize this benefit and propose new algorithms in the next chapter. Second, instead of treating each pixel equally by using (6), we can modify the similarity value of each patch according to the pose changes. In the meantime, a probabilistic model can be trained to model such changes and improve face recognition under pose variation.

Notice that after a texture map is decomposed into patches, in the boundary area of the face portion, there are some patches including partial missing data. For simplicity, we treat all these patches as missing data. Considering the fact that the patch size is not too big and also the boundary area is heavily up-sampled from the original image domain, the simplification is negligible since we only discard a very small amount of boundary pixels.

In our implementation, the patch size is 4 by 4 pixels and the texture map's size is 90 by 180 pixels. Thus there are 22 and 45 patches in the vertical and horizontal directions respectively. The selection of the patch size is a trade-off. If the patch size is too big, we lose the benefit of modeling local appearance and we could not model enough patch variation with respect to the varying pose. On the other hand, if the patch size is too small, it is harder to find corresponding among patches from multiple texture maps. From the experiments, we find 4 by 4 is a good choice for the patch size. Also we think it is not necessary to overlap patches since we will allow patches move around locally, which will be introduced in the next chapter.

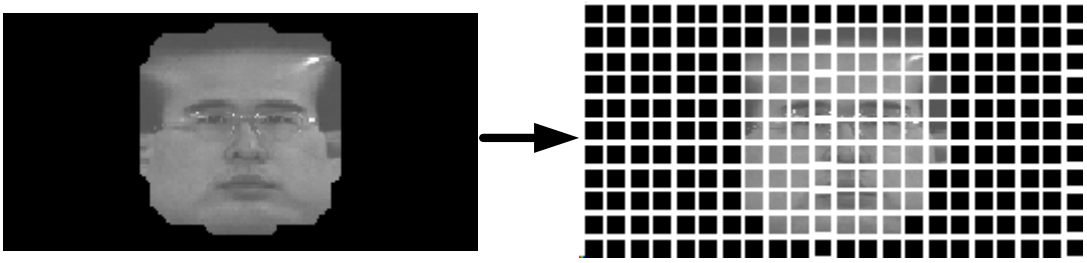

Figure 8 Patch representation for the texture map: a texture map is evenly decomposed into an array of local patches.

Let us introduce how to train a probabilistic model for the similarity value of patches from a face database with pose variation. In this thesis we train such a model on the CMU PIE database [24]. The PIE database consists of face images of 68 subjects under different combinations of poses and illuminations. We use part of this database in this thesis, which are 9 pose images from 68 subjects. These are the images with multiple poses under the neutral illumination. Sample images from one subject can be seen from Figure 9, where the number, c27, c34, c14, c11, c29, c22, c02, c37, c05, is the pose labels for each image. We choose c27 as the training pose and the other eight poses as the test poses.

We take 9 pose images of 34 subjects for training the probabilistic model. We denote each of the images as $\mathbf{f}(l,\phi_m)$, where ϕ_m is one of the eight pose labels. For the training process, the mapping parameters of all images are estimated based on the universal mosaic model. Thus we can obtain the texture maps of all images, and have the patch representation as $\mathbf{s}_{i,j}(l,\phi_m)$, where i and j are the index of patches vertically and horizontally.

Since we treat the front view, c27, as the training images, we need to study how the similarity values of corresponding patches between c27 and all other eight poses change. This is done by fixing one patch and one particular pose, and calculating the similarity value (mean-square-error) of one patch between all subjects in the pose c27 and all subjects in that particular pose. For example, Figure 10 is the result of such a calculation for one patch closer to the right eye and the pose c29. In this 2D map, the vertical axis represents all the training images, 34

subjects under the pose c27, while the horizontal axis represents all test images from 34 subjects under the pose c29. Each entry indicates the similarity value of the same patch between any pair of subjects. For each combination of all other patches and other eight test poses, we should generate one such 2D map.

Ideally we should expect that the diagonal elements of this 2D map are darker than the off-diagonal elements because the former is an indication of the intra-subject variations, while the latter is an indication of the inter-subject variations. In order to verify such expectation, we plot the histogram of the diagonal elements and off-diagonal elements separately. Also, for explicitly modeling these two types of variations, we approximate them as two Gaussian distributions. That is, we estimate the mean and stand deviation of intra-subject variations from the diagonal elements, and the mean of and stand deviation of inter-subject variations from off-diagonal elements. The resulting two distributions can be denoted as following:

$$P(d_{i,j}|same,\phi_m)=\frac{1}{\sqrt{2\pi}\sigma_{i,j}^{same}}\exp\left[-\frac{1}{2}(\frac{d_{i,j}-\mu_{i,j}^{same}}{\sigma_{i,j}^{same}})^2\right]$$

$$P(d_{i,j}|diff,\phi_m)=\frac{1}{\sqrt{2\pi}\sigma_{i,j}^{diff}}\exp\left[-\frac{1}{2}(\frac{d_{i,j}-\mu_{i,j}^{diff}}{\sigma_{i,j}^{diff}})^2\right] \quad (7)$$

where $\mu_{i,j}^{same}$, $\sigma_{i,j}^{same}$, $\mu_{i,j}^{diff}$, $\sigma_{i,j}^{diff}$ are the mean and stand deviation of intra-subject and inter-subject variations for the patch (i, j) under the test pose ϕ_m. Let us denote the probabilistic model as $\mathbf{P}_d=\{\{\mu_{i,j}^{same},\mu_{i,j}^{diff},\sigma_{i,j}^{same},\sigma_{i,j}^{diff}\}_{\phi_m}\}$. Notice that all four parameters depend on the test pose ϕ_m. For example, the first plot on the left of Figure 11 is the Gaussian approximation of two distributions in Figure 10. The solid and broken curves are the histograms of two distributions, and the dotted curves are the approximated two Gaussian distributions. The four figures in Figure 11 are from the two distributions of the same patch with four different test poses: slightly right (c29), more right (c11), further right (c14), profile (c34). We can see that as the pose changes

from the front view to the profile view, the discriminative power is getting less, which is an useful observation and should be taken into account during the recognition.

To illustrate the relation among these parameters for all test poses, we plot them in Figure 12. In total, there are five columns and eight rows, where each row corresponds to the statistical information of each test pose, namely c34, c14, c11, c29, c05, c37, c02, c22 from top to bottom. The first four columns are the plots of $\mu_{i,j}^{same}$, $\mu_{i,j}^{diff}$, $\sigma_{i,j}^{same}$, $\sigma_{i,j}^{diff}$ for all eight test poses. The intensity of each pixel indicates the value of parameter. The brighter the intensity it, the larger the value is. In order to compare the difference between these two distributions, we normalize the intensity of the first and second column, as well as the intensity of the third and fourth column. Naturally, we can observe that the second column, $\mu_{i,j}^{diff}$, is brighter than the first column, $\mu_{i,j}^{same}$, and the fourth column, $\sigma_{i,j}^{diff}$, is brighter than the third column, $\sigma_{i,j}^{same}$, which means the inter-subject variations have larger mean and stand deviation than those of the intra-subject variations.

The last column is the Fisher ratio [17] between two Gaussian distributions defined as following:

$$f_{i,j} = \frac{(\mu_{i,j}^{diff} - \mu_{i,j}^{same})^2}{\sigma_{i,j}^{same\,2} + \sigma_{i,j}^{diff\,2}}$$

Since the fisher ratio is a good indication of the discriminative power, we can study among all patches in the texture map, which patches provide more discriminative power than the others. From the last column of Figure 12, we observe that the nose and forehead seem to have more discriminative power. This observation might not be true in general. However, it seems to be a right conclusion for this particular dataset.

Figure 9 Sample Images of one subject from the PIE database: the image in the first row is the training image, while all the others are test images.

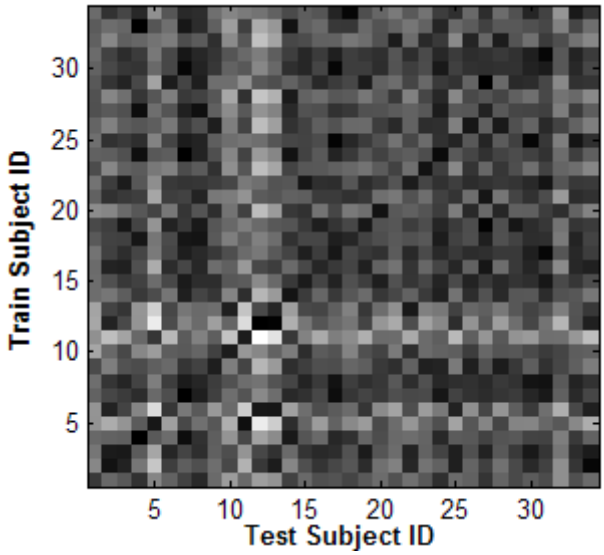

Figure 10 2D map of the similar values of one patch (around the right eye) between and pose c29 and c27

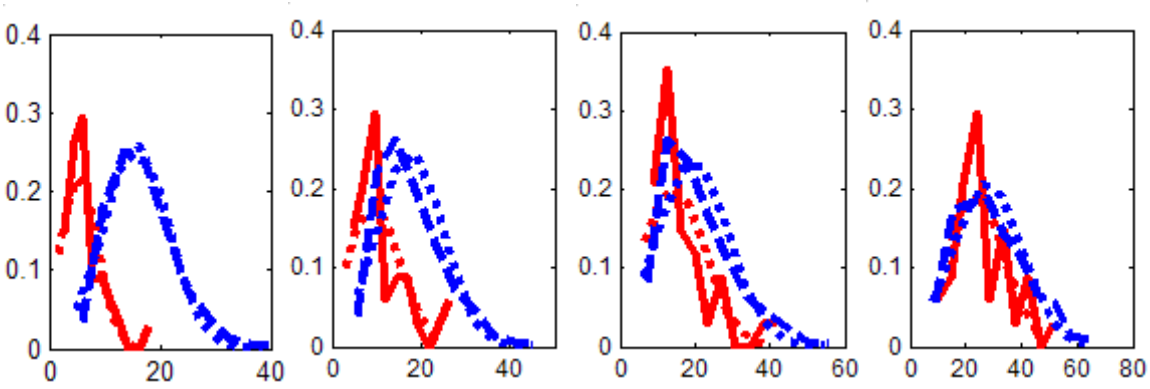

Figure 11 Gaussian approximation: each figure has two histograms (solid and broken curves) and two Gaussian approximations (dotted curves); four figures are from the two distributions of the same patch (around the right eye) with four different poses, c29, c11, c14, c34.

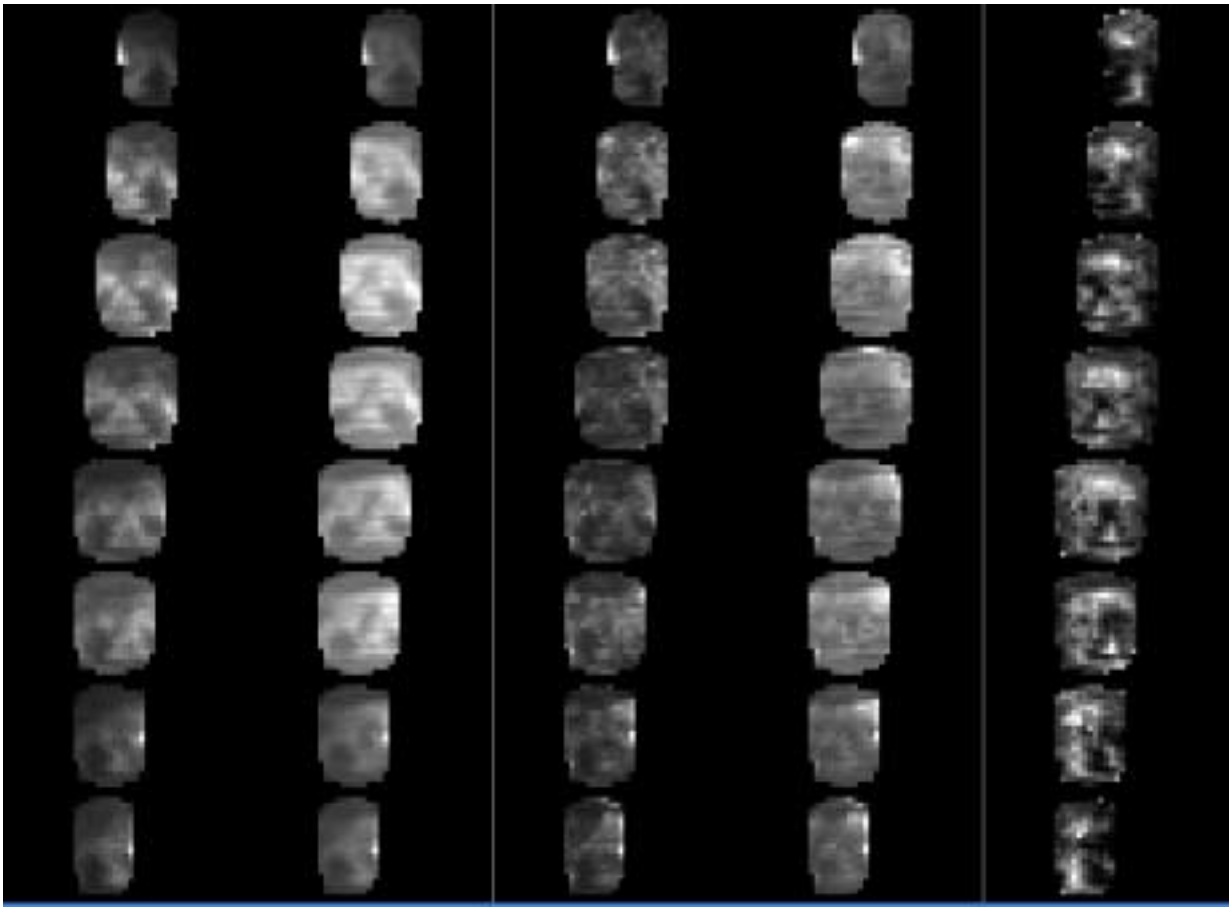

Figure 12 Probabilistic modeling for patches: the first four columns are plots of $\mu_{i,j}^{same}$, $\mu_{i,j}^{diff}$, $\sigma_{i,j}^{same}$, $\sigma_{i,j}^{diff}$ for all eight test poses; the last column is the fisher ratio of two distributions for all eight poses; each row corresponds to the statistical information of each test pose, namely c34, c14, c11, c29, c05, c37, c02, c22 from top to bottom.

3.4 Probabilistic geometry-assisted face recognition

After introducing how to train a probabilistic model, let us focus on how to utilize it for improving pose robust face recognition. Given a face database with L subjects, only one front view image, $\mathbf{f}_l$, of each subject is available for training. During the training stage, the mosaic algorithm estimates the optimal mapping parameter $\mathbf{x}_l$ for each training image $\mathbf{f}_l$ based on the universal mosaic model. The resulting texture map is represented as an array of local patches, $\mathbf{s}_{i,j}^l$.

Given a test image, we generate its texture map $\mathbf{s}_{i,j}^t$ based on the universal mosaic model. For the test texture map $\mathbf{s}_{i,j}^t$ and one of the training texture map $\mathbf{s}_{i,j}^l$, we compute the similarity values of all corresponding patches, $\{d_{i,j}\}$. Since we have developed the probabilistic models of similarity values of each local patch, it enables us to properly combine these similarity values, one computed for each patch, to reach to the local decision for recognizing whether the two texture maps/faces are from the same subject or not.

Given the similarity values and the pose of the test image, the posteriori probability that the test image and the training image belong to the same subject is:

$$P(same|d_{i,j},\phi_t) = \frac{p(d_{i,j}|same,\phi_t)P(same)}{p(d_{i,j}|same,\phi_t)P(same) + p(d_{i,j}|diff,\phi_t)P(diff)} \tag{8}$$

where ϕ_t is the pose of the test image, which can be obtained during the estimation of the mapping parameter, $P(same)$ and $P(diff)$ are a priority probability of being the same subject or not given any test image. For a database with L subjects, normally we can set $P(same) = \frac{1}{L}$ and

$P(diff) = \frac{L-1}{L}$. Notice that in order to calculate $p(d_{i,j}|same, \phi_t)$ using (7), ϕ_t needs to be equal to one of the test poses ϕ_m. This issue can be dealt with in two different ways.

First, if the pose of the test image ϕ_t is similar to one of the eight test poses ϕ_m, we can approximate ϕ_t using the most similar test pose. Second, if ϕ_t is not similar to any one of test poses ϕ_m, we can compute the marginal distributions of (8) over ϕ_m:

$$p(d_{i,j}|same) = \sum_m P(\phi_m)\, p(d_{i,j}|same, \phi_m)$$
$$p(d_{i,j}|diff) = \sum_m P(\phi_m)\, p(d_{i,j}|diff, \phi_m)$$
$$P(same|d_{i,j}) = \frac{p(d_{i,j}|same)P(same)}{p(d_{i,j}|same)P(same) + p(d_{i,j}|diff)P(diff)}$$

Here we assign a uniform distribution for $P(\phi_m)$. It could be non-uniform if we consider the probability of each pose presenting in the test set. Finally, the sum rule is applied. That is, the averaged probability measure of all patches $P(same|d_{i,j})$ will be the similarity measure between the test image and one of the training subjects. Basically different combination rules, such as the sum rule, the product rule, the max rule, etc, can be applied here. Kittler et. al. conclude that in general the sum rule outperforms other combination rules because the sum rule is more resilient to estimation errors [33]. The test image is recognized to be the subject that gives the highest similarity measure.

3.5 Experimental results

We evaluate our algorithm by comparing its performance on the CMU PIE database with a standard eigenface method [72]. We use half of the subjects (34 subjects) in the PIE database for training the probabilistic model as presented above. The 9 pose images per subject from remaining 34 subjects are used for the recognition experiments.

The front view image (c27) is used for the training, and the other 8 images are used for test. As shown in Figure 13, the horizontal axis represents the labels of 8 pose images,

34,14,11,29,05,37,03,22 from the right profile to the left profile. The vertical axis shows the recognition rate of four different algorithms for each specific pose. The first is the traditional eigenface approach [72], where the nearest neighbor classifier is applied. We have manually cropped the human face for both the training and test images, and normalized them to the size of 64 by 64 pixels. Since there are 34 training images in total, it is possible to use an eigenspace whose number of eigenvector varies from 1 to 33. We have tested all these possibilities and plotted the one with the best recognition performance, whose number of eigenvectors is 21. The second algorithm is our geometry-assisted method without probabilistic modeling, which is presented in Chapter 3.2. The third algorithm is the geometry-assisted method with probabilistic modeling.

A number of observations can be made from this result. First, when the pose of the test image is more toward the profile view, the recognition rate is getting lower. Second, both our algorithms perform much better than the baseline algorithm. Third, the geometry-assisted method with probabilistic modeling works better than the one without probabilistic modeling. We can see that with one training image, our algorithm presents satisfying recognition performance: it recognizes all face views with more than 90% correct rate except the two most extreme profile views. Even for the two profile views, around 70% and 60% recognition rates are obtained.

We also plot the results of the multi-subregion method reported in Figure 8(a) of [32]. We can see that the performance of our algorithm is comparable with the multi-subregion method for test images closer to the front view. For test images closer to profile views, our algorithm performs noticeably better. For example, in their report, the recognition rates of two profile views are both lower than 40%. There are a few reasons why our method works better for profile views. One is that we utilize more appearance information instead of only using the area bounded by facial features, such as eyes and the mouth, as done in [32]. Also, the geometrical mapping greatly compensates the pose variation and reduces the intra-subject variations.

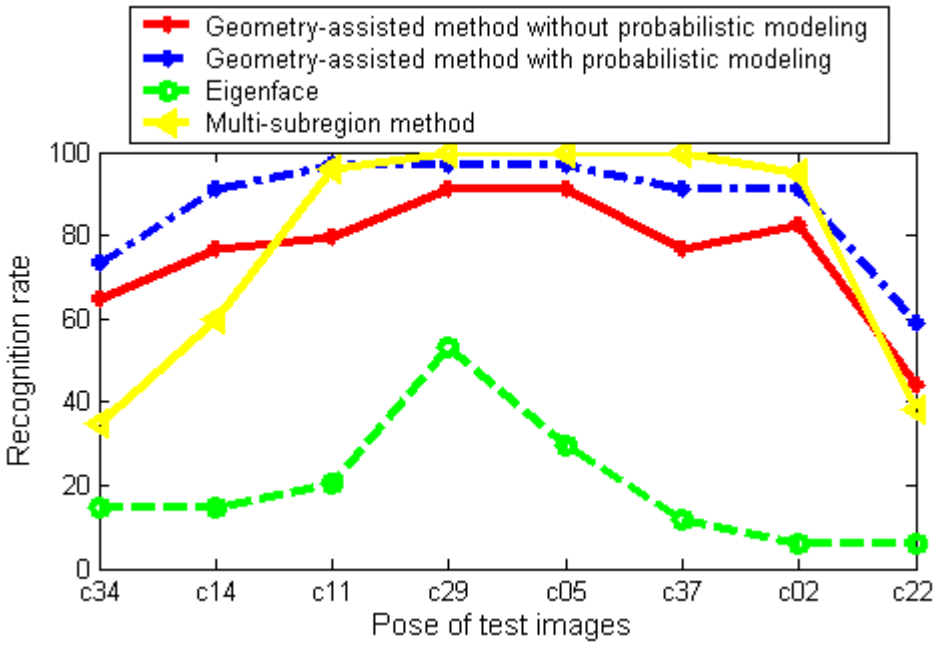

Figure 13 Recognition performances of four algorithms on the CMU PIE database based one front training image.

3.6 Conclusions

In this chapter we have introduced a probabilistic geometry-assisted approach and applied it to pose robust face recognition. All training and test images are projected onto the surface of a 3D ellipsoid by estimating the optimal pose and position, and represented as texture maps. The distance measure is calculated on the overlap area between any two texture maps. Also by representing a texture map as an array of local patches, it enables us to develop a probabilistic model for the similarity value of patches from a face database with pose variation. Eventually we are able to utilize the Bayesian framework to evaluate the similarity value of corresponding patches. Comparing with the baseline algorithm, we observe a significant improvement when performing experiments on the CMU PIE database.

The above proposed algorithms work well for the case where only one training image is available for each subject. However, if there is more than one training images, can we recognize faces better? The key issue is how to combine multiple training images and generate a unified

model that covers all pose variation in the training images. We will focus on this issue and propose new algorithms in the next chapter.

4. Face Mosaicing For Recognition

In the previous chapter, we propose that pose robust face recognition should be performed in the feature space of the texture map, instead of the original image space. Due to limited one training image per subject, there is only one *texture map* for each subject after training. In order to build a *statistical mosaic* model for each subject, we need multiple training images. This chapter will present our proposed algorithm on how to build such a statistical model from multiple images.

To be more specific, given $\mathbf{f}_k$, a set of images containing faces with different poses, we need to build a geometric deviation model $\Theta = \{\mathbf{g}, \mathbf{u}\}$ and a statistical appearance model $\prod = \{\mathbf{m}_{i,j}, \mathbf{V}_{i,j}\}$, which is an array of patches each of which is modeled by an eigenspace. The statistical mosaic model is composed of both these models together with the probabilistic model $\mathbf{P}_d$ whose training is presented in the previous chapter.

In our mosaic method, combining multiple images is essentially combining multiple texture maps since all images are converted to texture maps. When combining multiple texture maps, it is natural to observe that the same facial feature, such as the mouth corner, found in multiple maps might not correspond to the same coordinate on the single texture map. The blurring effect, which is normally not a good property for modeling, will therefore be observed when we combine many texture maps. To reduce such blurring, one key idea in our proposal is to allow a local patch to move toward better corresponding across multiple maps, use the flow

representation for modeling the amount of movement, and train the flow representation to obtain the geometric deviation model via PCA. Since the flow representation plays a key role in this process, we first present how we use it for face recognition. Next, we introduce our proposed algorithm for training both a geometric deviation model and an appearance model from multiple training images. Finally, we show the experimental results by using this new method.

4.1 Flow representation for face recognition

Flow representation (optical flow) [34] is generally used for motion analysis. Using two or more consecutive frames of an image sequence, a 2-dimensional vector field, called the optical flow, is computed to estimate the most likely displacement of image pixels from one frame to another. Some researchers use optical flow in the analysis of human expressions for the purpose of expression recognition [46][75][69]. Also Kruizinga and Petkov [34] propose to utilize optical flows in person identification. However, they only consider the optical flow residue as the criterion of classification, while we propose to make use of the eigenflow residue, which appears to exhibit better classification ability than the former.

Optical flow essentially is an approximation of the velocity field. It approximately characterizes the motion of each pixel between two images. If two face images, which show different expressions of the same subject, are fed into the optical flow algorithm, the resultant motion field will emphasize the regions of facial features, such as the eyes and the mouth. This is illustrated in Figure 14. The left half of the figure shows two face images from the same subject, but with different expressions. The resulting optical flow is shown below these figures. The second set shows the same figure except that the two input images are from two different subjects. Obviously, the optical flow looks more irregular in this case. This clue can help discriminating these two cases, which is the task of face recognition.

The same idea can be applied to images with registration errors. Because the traditional PCA approach is unacceptably sensitive to registration errors, even small shifts in input images can make the system performance degrade significantly. However, face images are usually difficult to register precisely, especially in a live authentication system. Therefore, we want to use the optical flow to build a system that is tolerant to different kinds of registration errors. In Figure 15, the second image in the left column is an up-shifted version of the first image. The optical flow shown below captures most of its motion around facial features. The right column shows images of different subjects leading to an optical flow that appears to be random.

Since the optical flow provides a useful pattern for classifying personal identity, we propose to use PCA to model this pattern. Suppose that in the training data set, there are a few images with different expressions for each subject, such as five images shown in Figure 16. Using these images, twenty optical flow images $\mathbf{o}_k (1 \le k \le K)$ (corresponding to twenty pairs) can be obtained through the optical flow estimation. PCA can be computed through the following:

$$\mathbf{g} = \frac{1}{K} \sum_{k=0}^{K} \mathbf{o}_k$$

$$\mathbf{C} = \frac{1}{K} \sum_{k=0}^{K} (\mathbf{o}_k - \mathbf{g})(\mathbf{o}_k - \mathbf{g})^T$$

By performing eigen-analysis for the covariance matrix $\mathbf{C}$, we can obtain a number of eigenvectors $\mathbf{u} = \{\mathbf{u}_1, \mathbf{u}_2, \cdots\}$. The three principal eigenflows of twenty optical flow images are shown in Figure 17. Obviously large motion can be observed in the region of facial features, such as mouth corners, eyebrows and nasolabial furrows. So all the expression variations occurring in a single subject can be represented by a space spanned by these eigenflows. The optical flow $\mathbf{o}$ between any two images of this subject should have small residue defined as:

$$e = \left\| \mathbf{o} - \mathbf{g} - \mathbf{u}\mathbf{u}^T (\mathbf{o} - \mathbf{g}) \right\|^2 \tag{9}$$

This is basically the error term that could not be modeled by a subspace. In contrast, the optical flow between this subject and other subjects cannot be represented well by this space, which results in a large residue. We call this the *eigenflow residue*. Thus, the eigenflow residue can be a useful feature for recognition. Similarly eigenflows can be used to model the optical flow caused by image registration errors.

We have applied the eigenflow approach for face recognition and authentication, and obtained satisfying results. Please refer to [50] for detail information about this approach. In the next section, we will introduce how to use the same idea for modeling the geometric deviation and serving for face recognition.

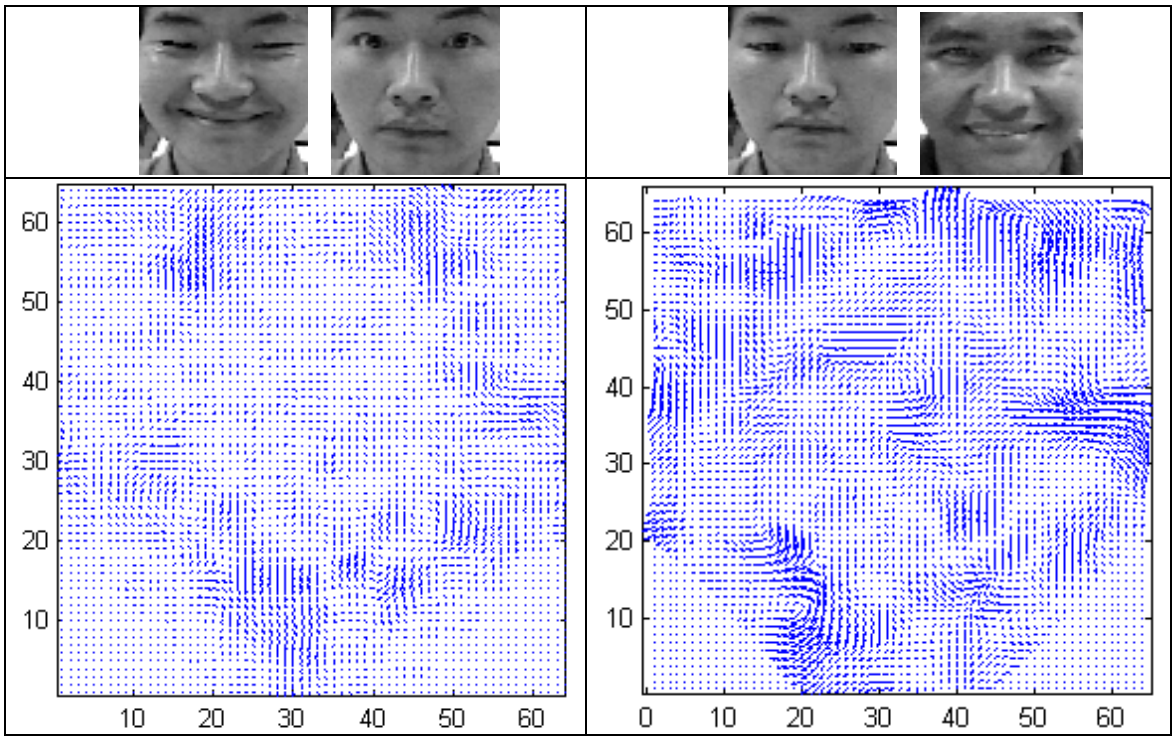

Figure 14 Applying optical flow on images with different expressions: left hand side are two images from the same subject, and right hand side are two images from different subject. Different randomness pattern can be observed from two resulting optical flows.

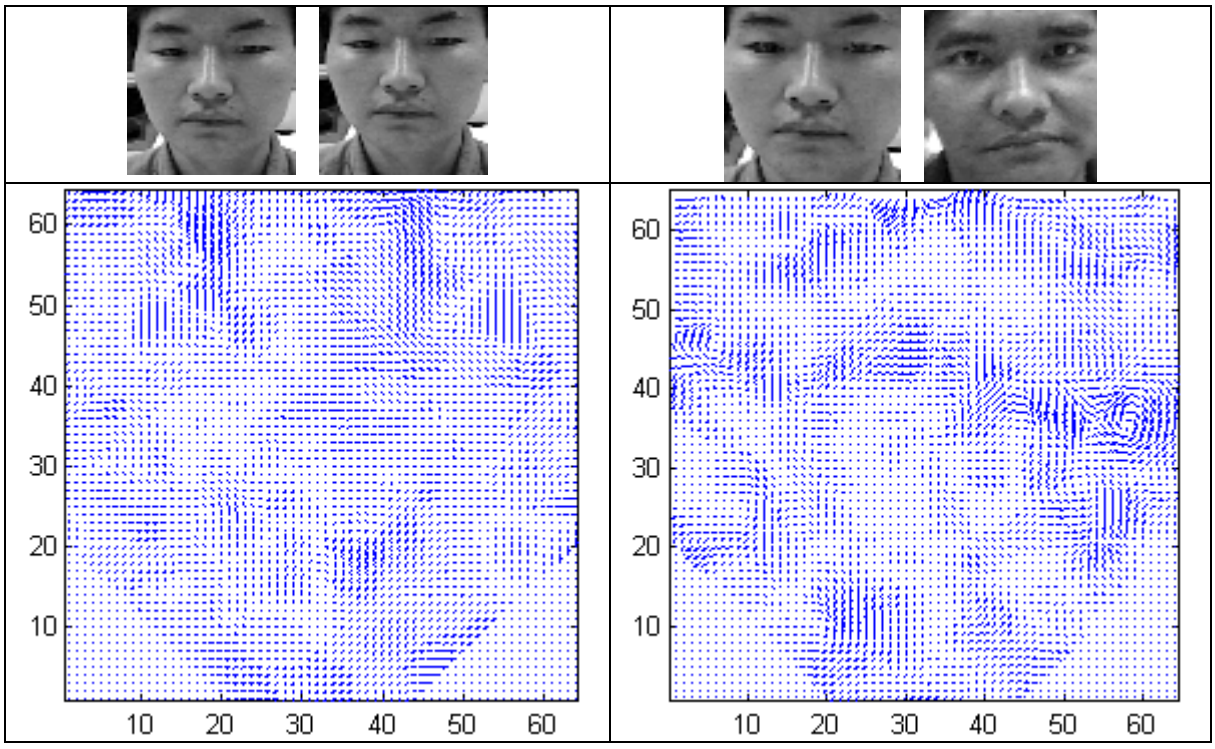

Figure 15 Applying optical flow on images with registration errors: left hand side are two images from the same subject, and right hand side are two images from different subject. Different randomness pattern can be observed from two resulting optical flows.

Figure 16 Five expression images used for training an individual eigenflow for this subject.

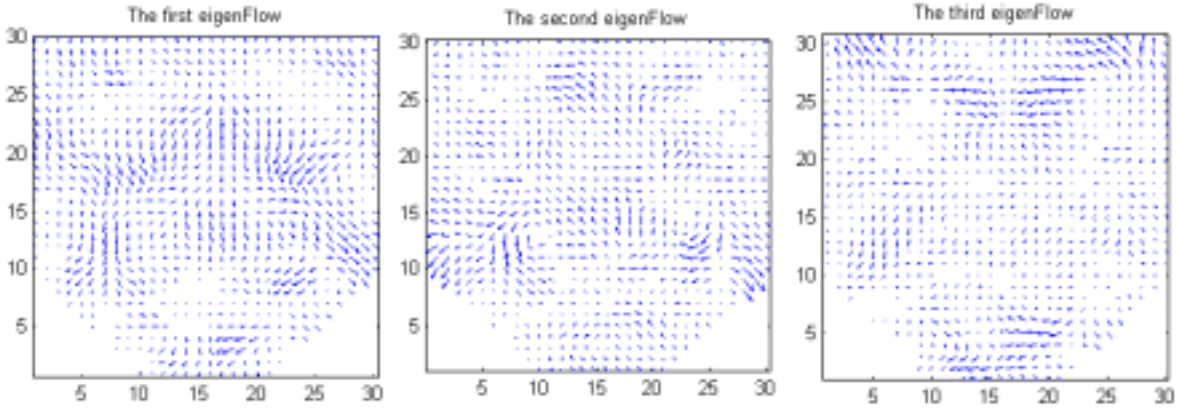

Figure 17 The first three eigenflows trained from expression images of one subject: Some prominent movements of facial features, such as mouth corners, eyebrows, nasolabial furrows, can be seen from them.

4.2 Modeling the geometric deviation

One potential problem of combining multiple texture maps is that the resulting averaged map might get blur due to the fact that facial features from multiple maps do not corresponding to the same coordinate in the texture map. To reduce such blurring, we might need to align the facial features better by relying on some landmark points. For the model training process, it is reasonable to obtain such landmark points by manual labeling.

Given K training images, $\mathbf{f}_k$, including different poses of human faces, in order to facilitate the modeling process, we label the position of facial feature points. As shown in Figure 18, 25 facial feature points are labeled. For each training image, only a subset of the 25 points will be marked according to their visibility. We call these points as *key points*.

Figure 18 Labeled facial features: up to 25 feature points are labeled on each training images

As usual, first we generate the texture maps $\mathbf{s}^k$ from each training images. Since we only label the facial key points on the training images, we need to find their corresponding coordinates $\mathbf{b}_k^i (i = 1,2,\cdots,25)$ in the texture map $\mathbf{s}^k$, as show in the first two rows of Figure 19. Essentially this is an inverse operation of the geometrical mapping described in the previous chapter.

After determining key points on all texture maps of the training images, we need to find the coordinate on the mosaic model where all corresponding key points deviate to. Ideally if the human head's geometry is a perfect 3D ellipsoid, the same key point $\mathbf{b}_k^i (1 \le k \le K)$ from multiple training texture maps should correspond to exactly the same coordinate, i.e., $\mathbf{b}_1^i = \mathbf{b}_2^i = \cdots = \mathbf{b}_K^i$, For example, if we look at the three texture maps in the second row of Figure 19, the coordinate of the left nose's corner should be the same. However, due to the fact that the human head is not a perfect ellipsoid, these key points will deviate to each other. The amount of deviation is an indication of how much geometrical difference between the actual head geometry and the 3D ellipsoid. We will model such deviation by applying PCA on the flow representation.

First, we compute the averaged position $\mathbf{b}^i$ of all key points $\mathbf{b}_k^i (1 \le k \le K)$ that correspond to the same facial feature and are also visible on the texture map. We treat this averaging as the target position in the final mosaic model where all corresponding key point should deviate. As shown in the third row of Figure 19, each white point is the averaged position computed from all training texture maps. Since our mosaic model is composed of an array of patches, each one of 25 averaged key points falls into one particular patch, which is called *key patch*. Notice that instead of averaging, we can also use weighting in generating $\mathbf{b}^i$. For example, the texture maps that are more reliable (mostly front view images) would have larger weights.

Second, for each texture map, we take the difference between the positions of key point $\mathbf{b}_k^i$ and that of the averaged key point $\mathbf{b}^i$ as the *key patch's deviation flow* (DF) that describes which patch from each texture map should move toward one key patch in the mosaic model. However, there are also non-key patches in the mosaic model. In order to model their deviation flows, as shown in Figure 20, we represent the mosaic model as a set of triangles, whose vertexes are the key patches. Thus for each non-key patch, it falls into at least one triangle. In the last step, the deviation flow for a non-key patch from each training texture map is interpolated by the key

patch's deviation flow of one triangle. The reason we assign a non-key patch to multiple triangles is that in case some key patch's deviation flows of one triangle are not available due to their invisibility, we can rely on other triangles to perform the interpolation. One might think why we compute deviation flows through the triangulation of key patches, rather than applying optical flow. The reason is that traditional optical flow computation starts with two images: a test image and a reference image. However, when we compute deviation flows, we do not have the reference texture map yet, which will be calculated *after* the deviation map is obtained. Thus we could not compute deviation flows using optical flow.

For each training texture map, its geometric deviation is a 2D vector map $\mathbf{v}_{i,j}^{k}$. Its dimension is the same as the number of patches in the vertical and horizontal directions, and each pixel is a vector indicating how far this patch is away from the averaged patch in the mosaic model. Notice that for any training texture map, some elements in the 2D $\mathbf{v}_{i,j}^{k}$ are considered as missing ones. We use $\mathbf{a}_{i,j}^{k}$ to denote the mask map of $\mathbf{v}_{i,j}^{k}$. If $\mathbf{v}_{i,j}^{k}$ is a miss element, $\mathbf{a}_{i,j}^{k}$ is zero, otherwise it is one.

In order to model the deviation, we train the geometric deviation $\mathbf{v}_{i,j}^{k}$ from all training texture maps using the PCA with missing data [71] as following:

$$\mathbf{g}_{i,j} = \frac{1}{\sum_{k=1}^{K} \mathbf{a}_{i,j}^{k}} \sum_{k=0}^{K} (\mathbf{a}_{i,j}^{k} \mathbf{v}_{i,j}^{k})$$

$$\mathbf{v}_{i,j}^{k} = \mathbf{g}_{i,j} \quad \text{if } \mathbf{a}_{i,j}^{k} = 0, \text{ for } 1 \le k \le K$$

$$\mathbf{C} = \frac{1}{K} \sum_{k=0}^{K} (\mathbf{v}^{k} - \mathbf{g})(\mathbf{v}^{k} - \mathbf{g})^{T}$$

By performing eigen-analysis for the covariance matrix $\mathbf{C}$, we obtain a number of eigenvectors $\mathbf{u} = \{\mathbf{u}_1, \mathbf{u}_2, \cdots\}$. Figure 21 shows the resulting deviation model $\Theta = \{\mathbf{g}, \mathbf{u}_1, \mathbf{u}_2\}$ based on the training images from one subject. Essentially the linear combination of these basis

vectors describes all the possible geometric deviation of any view angle for this particular subject's face.

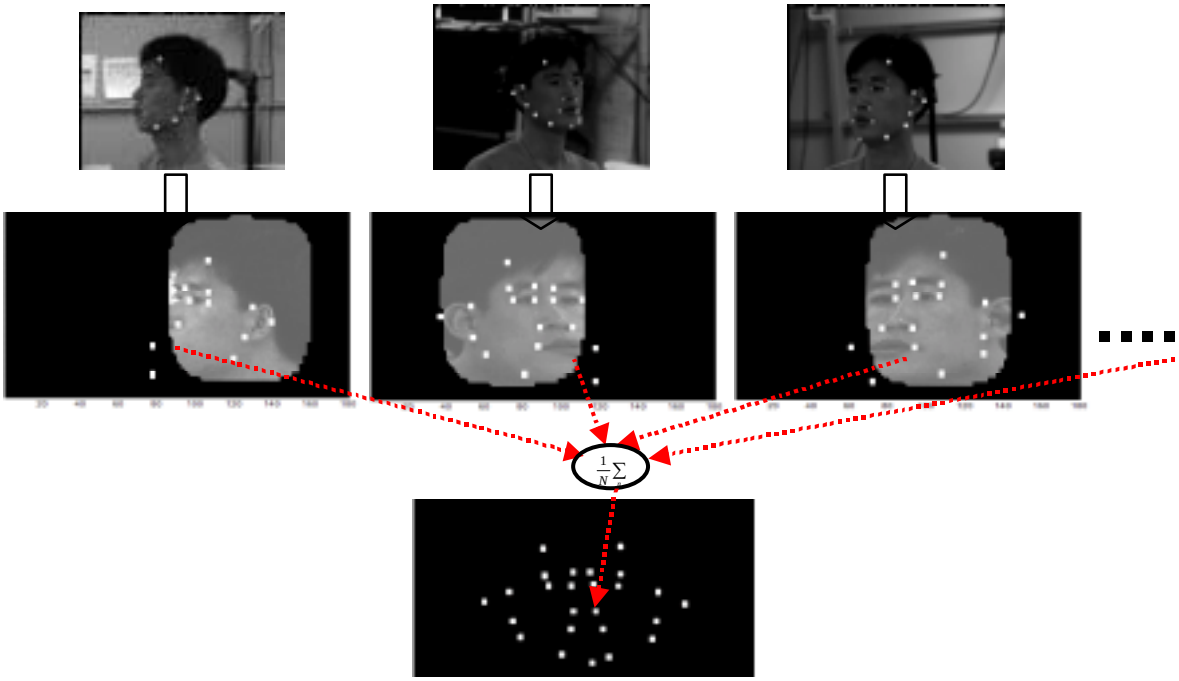

Figure 19 Mapping and averaging the position of key points: the position of all key points in the training texture maps (2nd row), which correspond to the same facial feature, such as the left eye corner, are averaged and result in the position in the final model (bottom row).

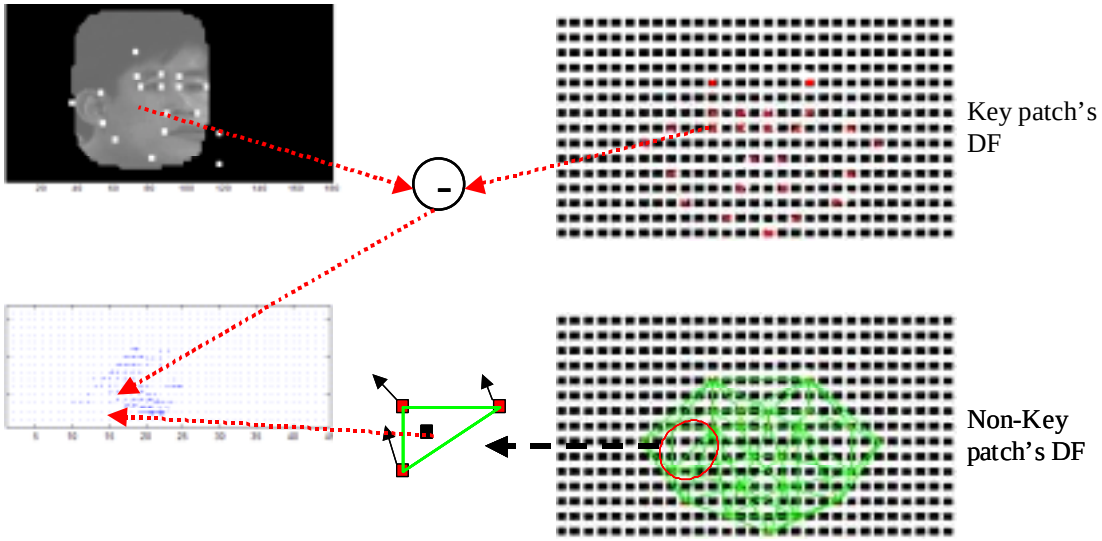

Figure 20 Computation of patch's deviation flow: each non-key patch falls into at least one triangle; the deviation of a non-key patch is interpolated by the key patch deviation of one triangle.

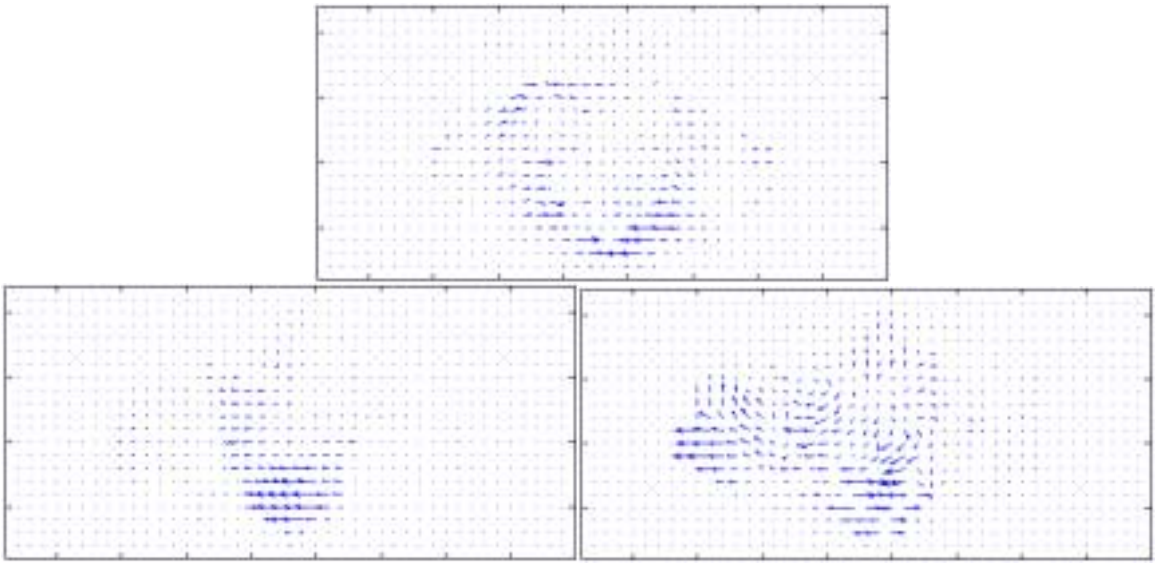

Figure 21 Trained geometric deviation model (Top: mean, left: 1st eigenvector, right: 2nd eigenvector)

4.3 Modeling the appearance

After modeling the geometric deviation, we also need to build an appearance model, which describes the facial appearance from all poses.

Figure 22 illustrates the process of building such an appearance model. On the left hand side, there are two pairs of training texture map $\mathbf{s}^k$ and its corresponding geometric deviation $\mathbf{v}_{i,j}^k$. The resulting appearance model $\prod = \{\mathbf{m}_{i,j}, \mathbf{V}_{i,j}\}$ with one mean and two eigenvectors are shown on the right hand side. This appearance model is composed of an array of eigenspaces, where each is devoted in modeling the appearance of the local patch indexed by (i, j). In order to train *one* eigenspace for one particular patch, the key issue is to *collect* one corresponding patch from each training texture maps $\mathbf{s}^k$, where the correspondence is specified by the geometric deviation $\mathbf{v}_{i,j}^k$. For example, to train an eigenspace $\Pi_{i,j}$ for a patch centered at (40,83), first we obtain the correspondence information from $\mathbf{v}_{i,j}^1$, which specifies how much deviation the corresponding patch in the texture map $\mathbf{s}^1$ with respect to the target location

(40,83). Hence the summation of $\mathbf{v}_{i,j}^{1}$ and (40,83) determines the center of corresponding patch, $\mathbf{s}_{i,j}^{1}$, in the texture map $\mathbf{s}^{1}$. Using the same way, we can find other corresponding patches $\mathbf{s}_{i,j}^{k}(k=2,3,\cdots K)$ from all other texture maps. Notice some of $\mathbf{s}_{i,j}^{k}$ might be considered as missing patches.

Once we collect corresponding patches $\mathbf{s}_{i,j}^{k}$ from all training texture maps, we are ready to take these patches $\mathbf{s}_{i,j}^{k}(1\le k\le K)$ as samples and train a statistical model $\Pi_{i,j}$ via PCA.

Figure 22 shows that a 2-dimensional eigenspace is obtained from the training patches. Finally, the appearance model is composed of an array of PCA models, where each PCA model describes the appearance of one patch. We call this the *patch-PCA mosaic*. Modeling via PCA is popular when the number of training samples is large, such as the training of a universal mosaic model based on many subjects, or of an individual mosaic model with many training images.

However, when the number of training samples is small, such as the training of an individual mosaic model with only a few training images, it might not be suitable for training a PCA model for each patch. Instead we would keep all the corresponding patches and use them directly as part of the model. One computational efficient way of doing this is to train a universal PCA model based on all corresponding patches $\mathbf{s}_{i,j}^{k}(1\le k\le K,1\le i\le I,1\le j\le J)$ of all training texture maps, and keep the coefficient of these patches in the universal PCA model as well. This is called as the *global-PCA mosaic*. Notice that the patch-PCA mosaic and the global-PCA mosaic only differ in how the corresponding patches across training texture maps are utilized to obtain a model, depending on the availability of training data in different application scenarios.

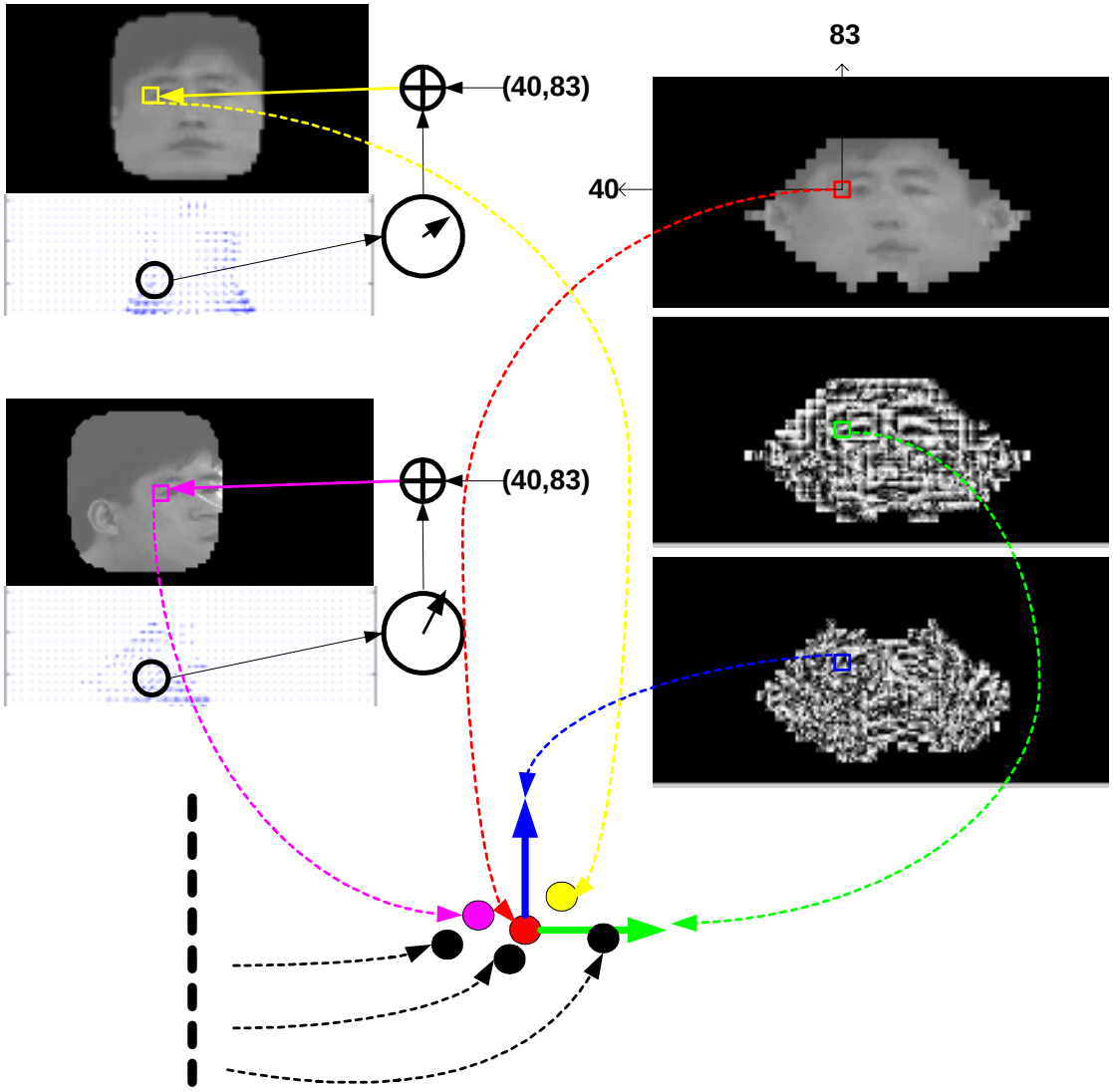

Figure 22 Training process of the appearance model for one patch: the deviation indicates where to find the corresponding patch from each of training texture maps; all corresponding patches are treated as samples for training a statistical model.

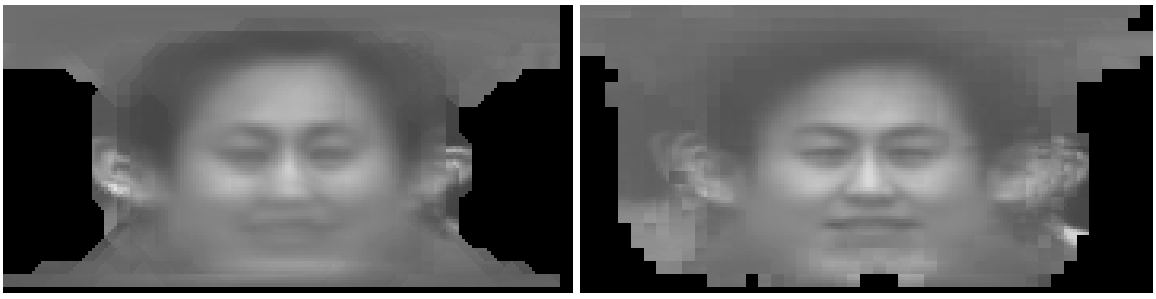

Figure 23 The mean of two universal mosaic models (left: without the modeling of geometric deviation, right: with the modeling of geometric deviation).

Eventually the statistical mosaic model includes the appearance model Π, the geometric deviation model Θ and the probabilistic model $\mathbf{P}_d$ trained as in the previous chapter. We consider that the geometric deviation model plays a key role in forming the mosaic model. For example, Figure 23 shows the mean appearance of two mosaic models trained by the same set of images from 10 subjects. The one on the left does not have the modeling of geometric deviation, while the right one has. It is obvious that the model on the right is much less blurring and captures more useful information of the facial appearance.

Looking at Figure 20, we notice that the modeling area of the mosaic model is bounded by the position of most outside key patches. In order to let the mosaic model cover larger pose variation, we can also do extrapolation while computing the deviation flow of non-key patches, so that more appearance information can be included in the final model. One example of using extrapolation is the right illustration of Figure 23, which covers much larger area on facial appearance comparing to the up-right illustration of Figure 22.

4.4 Face recognition using the statistical mosaic model

In the previous section, we present an approach to train a statistical mosaic model. Now let us see how this model can be used in pose robust face recognition.

Given L subjects with K training images for each subject, we use our approach to train an individual statistical mosaic model for each subject. For simplicity, let us assume we have enough training samples and obtain the patch-PCA mosaic for each subject. We will discuss the case of the global-PCA mosaic in the end of this section.

As shown in Figure 24, given one test image, we generate its texture map by using the universal mosaic model. Then we measure the distance between the test texture map and each of the individual mosaic. Thus the key issue here is to compute the *map-to-model* distance. Notice that the appearance model is composed of an array of patch models, which is called the *reference*

patch. Basically the map-to-model distance is the summation of *map-to-patch* distances. That is, for each reference patch, we need to find its corresponding patch from the test texture map, and compute its distance to the reference patch model. Figure 24 illustrates the calculation of the map-to-patch distance.

Since we deviate corresponding patches during the training stage, we should do the same while looking for the corresponding patch in the test stage, instead of picking up the patch from the text texture map that has the same coordinate as the reference patch. One simple approach is to search for the best corresponding patch for the reference patch inside a searching window, whose center is the coordinate of the reference patch. However, this approach does not impose any constraint on the deviation of neighboring reference patches. To solve this issue, we would like to make use of the deviation model that is trained before.

In the right hand size of Figure 24, there are three models, the deviation model $\Theta = \{\mathbf{g}, \mathbf{u}\}$, the appearance model $\prod = \{\mathbf{m}_{i,j}, \mathbf{V}_{i,j}\}$, and the probabilistic model $\mathbf{P}_d$, as the components of the statistical mosaic model. The deviation model describes all the possible geometric deviation of any view angle for one subject's faces. Because the geometries of different human heads are not the same, such deviation model contains useful information about individual subject's geometry. If we randomly sample one coefficient $\mathbf{c} = [c_1, c_2, \cdots]$ in this subspace model, the linear combination (or subspace reconstruction) of this coefficient describes the geometric deviation $\mathbf{v}^t$ for all reference patches.

$$\mathbf{v}^t = \mathbf{g} + \sum_k c_k \mathbf{u}_k$$

The benefit of this approach is that it enforces the geometric deviation of neighbor patches to follow certain constraint, which is described by the mean and eigenvectors of the deviation model. Based on this idea, the key is to find a coefficient in deviation subspace, which provides the optimal matching between the test texture map and the model. In our implementation, we adopt a simple searching scheme to find such a coefficient by determining

each dimension one by one. That is, in a K-dimensional deviation subspace, uniformly sample multiple coefficients along the first dimension while the coefficients for other dimensions are zero, and determine one of them which results in the maximal similarity between this text texture map and the model. The range of sampling is bounded by the coefficients of training deviation maps. Then we perform the same searching along the second dimension while fixing the optimal value for the first dimension and zero for all other dimensions. The searching is finished until the K^{th} dimension. Essentially this is a problem of motion estimation with a subspace constraint. In the future, we might also use the POCS idea to find a better solution [13].

For each sampled coefficient in the above searching scheme, the reconstructed 2D deviation map (in the bottom-left of Figure 24) indicates where to find the corresponding patch in the test texture map. Then the residue distance (9) between the corresponding patch and the reference patch model is computed, which is further feed into the probabilistic model. Finally, the probabilistic measurement provides how likely this corresponding patch belongs to the same subject as the reference patch. By doing the same operation for all other reference patches and averaging all patch-based probabilistic measurements, we obtain the similarity between this text texture map and the model based on the current sampled coefficient. Finally, the test image is recognized as the subject who provides the largest similarity.

Depending on how the individual mosaic model is trained (the patch-PCA mosaic or the global-PCA mosaic), there are different ways of calculating the distance between the corresponding patch and the reference patch model. As we presented before, for the patch-PCA mosaic, the residue with respect to the reference patch model is used as the distance measure. For the case of the global-PCA mosaic, since one reference patch model is represented by a number of coefficients, the distance measure is defined as the nearest neighbor of the corresponding patch among all these coefficients.

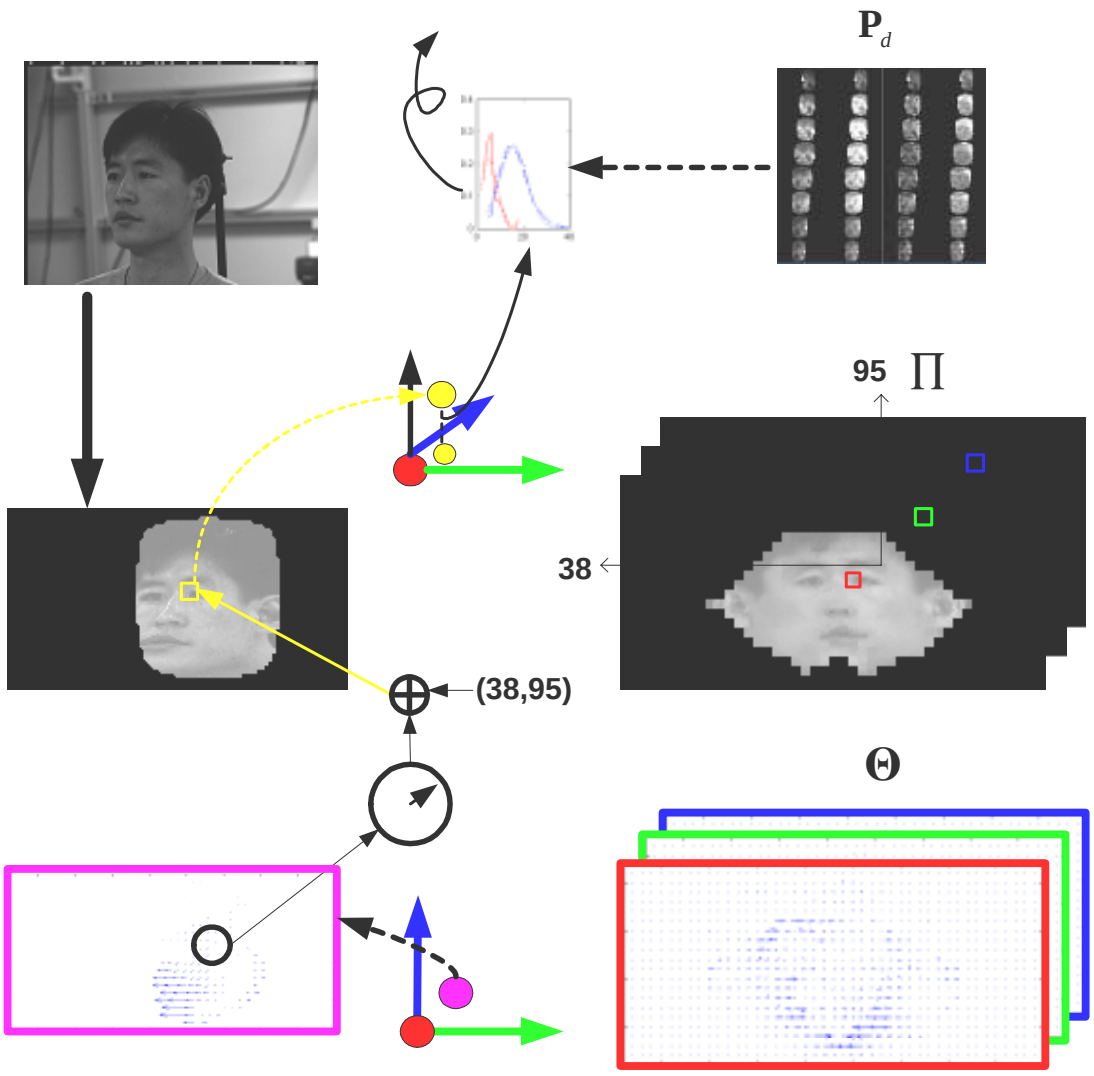

Figure 24 Computing the map-to-patch distance: the deviation map builds up the patch correspondence between the model and the test texture map; the distance measures from corresponding patches are feed into the Bayesian framework to generate a probabilistic distance measurement.

4.5 Determining the mapping parameters for training images

Given a set of training images for one subject, the first step in our mosaic algorithm is to generate the texture map for each training image. There are three ways of doing this. First, we can treat a pre-trained universal mosaic model as the reference and calculate the mapping parameter of all images refer to this universal model, by using the condensation method. Second, if one of the training images is the *front view*, we can generate its texture map, which will be

treated as the initial mosaic model, by labeling its boundary and assuming all rotation angles are zero. Then the mapping parameters of other training images can be found by minimizing their distances to the initial model. The third method is the same as the second one except that the rotation angles of the front view image are obtained from the 3D position of facial features, instead of assuming zero angles. This is to solve one potential problem with the second method, i.e., the front view face might not correspond to zero rotation angles. Actually this problem also exists for the first method when generating the initial texture map for the universal model. We will present the basic steps of the third method in this section.

The process of obtaining the 3D position of facial feature points is straightforward by using the stereo triangulation technique [20] in the vision community. We require that multiple view images of the initial front view image are available and all cameras are calibrated. In the following case, we have 3 views of the human face captured simultaneously, where the center view is the initial training image. First, as shown in Figure 25, we mark the common feature points among three views. We also mark the face vertical boundary on the 2D image. Second, by using the stereo triangulation, the 3D position of these feature points can be reconstructed. Third, based on the 3D position of feature points with respect to the center view, we can fit a 3D ellipsoid by minimizing the distance between the points to the ellipsoid surface, under the constraint that the 2D projection of the 3D ellipsoid at all three views should fit with the marked boundary. This constraint is important since if the 3D ellipsoid is larger than the actual head size, part of the background will be included in the texture mapping. Basically the 3D reconstructed vertical boundary points tell the vertical and horizontal center of the ellipsoid. Only the rotation angles and the center in the Z axis need to be determined during the fitting process. The fitted ellipsoid tells the optimal rotation angels that the front view training image could be approximated by an ellipsoid model. For example, in Figure 26, although the face in the center view looks like the front-view, the fitting results indicates that there is a slight tilt around the horizontal axis. We will use these rotation angles in generating the initial mosaic based on this

image. Later we will show the experimental results of comparing the second and third method in determining the mapping parameters for the training images.

What we have shown so far is one benefit of having calibrated multiple view images. Another benefit is that we may even build a 3D wire frame geometric model using the 3D coordinates of facial features. Thus the face image can be projected onto the wire frame, instead of the 3D ellipsoid model.

Figure 25 Marked feature points for three views

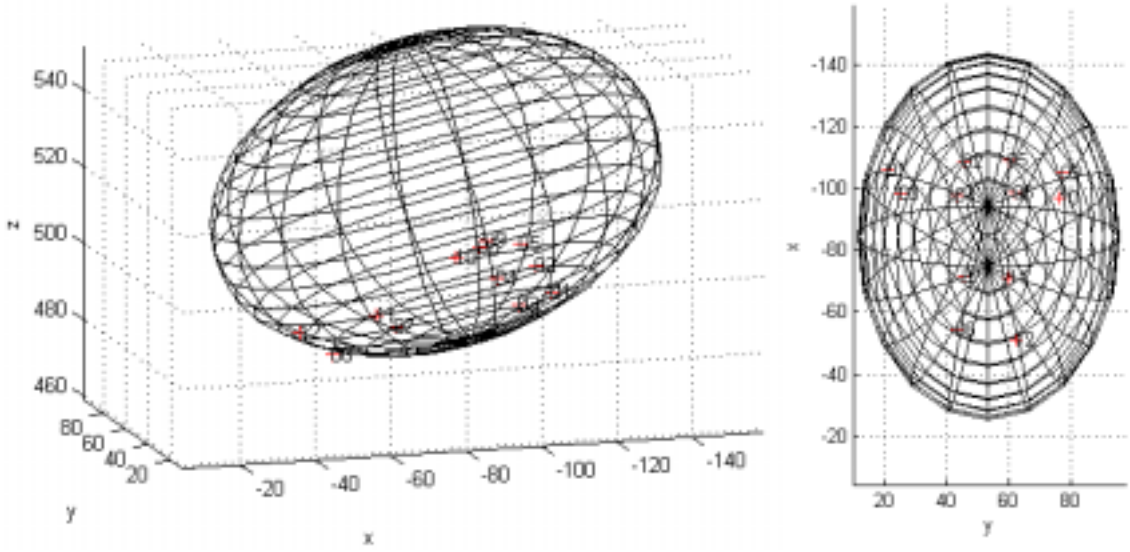

Figure 26 3D feature points and fitted ellipsoid

4.6 Experimental results

Similar to the previous chapter, we evaluate our algorithm by comparing its performance on the CMU PIE database with a baseline method. We use half of the subjects (34 subjects) in the PIE database for training the probabilistic. The 9 pose images per subject from remaining 34 subjects are used for the recognition experiments.

Three poses (c27, c14, c02) are used for the training, and the remaining 6 poses (c34, c11, c29, c05, c37, c22) are used for test. As shown in Figure 28, the horizontal axis represents the labels of 6 test poses. The vertical axis shows the recognition rate of three different algorithms for each specific pose. The first is the result of the eigen light-field algorithm from Figure 10 (a) in [23]. It is hard to find a previous method testing on the same scheme of the same database as us. We plot this result even it only uses one front view per subject as the training data. The second algorithm is our face mosaic method without the modeling of geometric deviation, which essentially let the mean of all eigenvectors of $\Theta = \{\mathbf{g}, \mathbf{u}\}$ to be zero. The third algorithm is the face mosaic method with the modeling of geometric deviation. Since the number of training images is small, we train the global-PCA mosaic for each subject. Three eigenvectors are used in building the global-PCA subspace. Thus each reference patch from the training stage is represented as a 3-dimentional vector. For the face mosaic method, the patch size is 4 by 4 pixels and the size of the texture map is 90 by 180 pixels. For illustration purpose, we plot the mean of three models in Figure 27. We can see that all mean images contain enough pose variation and do not blur much.

Comparing among these three algorithms, both of our algorithms works better than the baseline algorithm. Also, if we compare this result with the experimental results in the previous chapter (Figure 13), we can see the algorithm presented in this chapter works better since it has more training images and the individual mosaic model successfully combines the pose information from multiple images, while the algorithm in the previous chapter only takes one image for training. Obvious the mosaic approach provides a better way of registering multi-view images for an enhanced modeling, unlike the naïve training procedure of the traditional eigenface

approach. For our algorithms, the one with deviation modeling performs better than the one without deviation modeling. There are at least two benefits for the one with deviation modeling. One is that a geometric model can be used in the test stage. The other is that as a result of deviation modeling, the patch-based model also captures the personal characteristic of the multi-view facial appearance in a non-blurring manner.

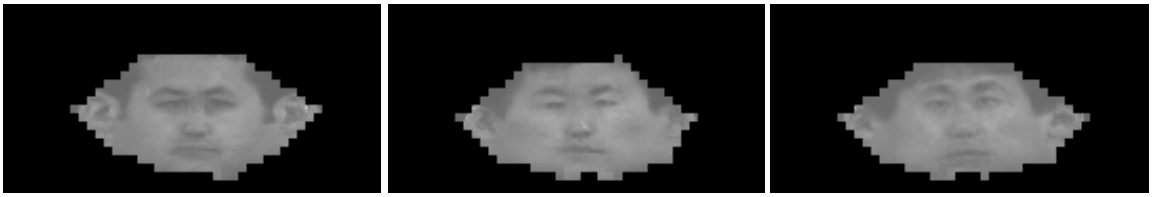

Figure 27 Mean images of three individual mosaic models.

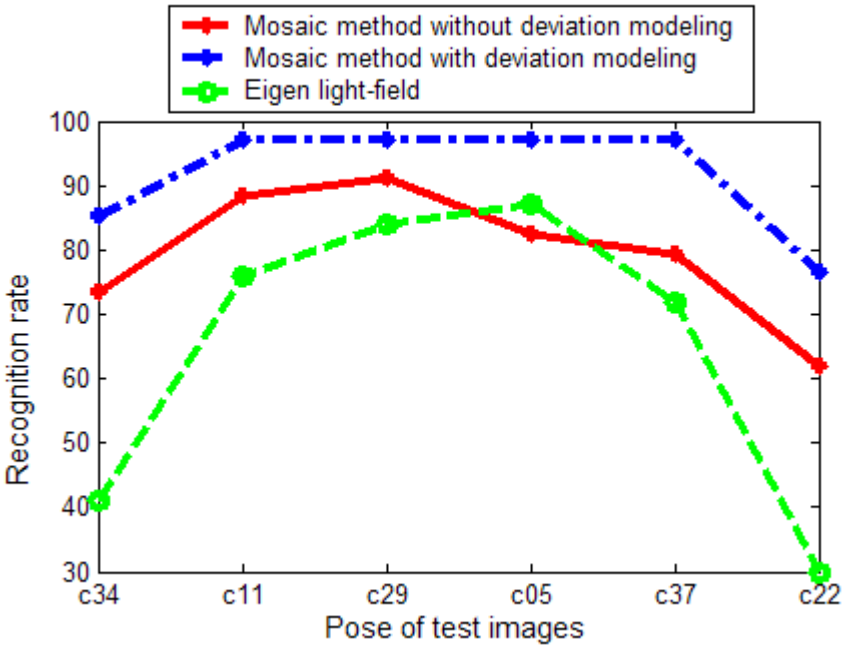

Figure 28 Recognition performances of three algorithms on the CMU PIE database based three training images.

In Section 4.5, we have mentioned that the 3D position of facial feature points could be used to determine the mapping parameters for the training images. We would like to see how could this help the mosaic based face recognition. We perform experiments based on the FIA

database, which we will introduce in detail in the next chapter. There are 20 subjects in the database, with 9 training images per subject, where one of them contains the front view face. There are 50 test images per subject. We have performed two algorithms on this database. Both of them use the individual mosaic method with deviation modeling. They only differ in that one uses zero angles in the mapping parameter for the initial front view image, the other uses the 3D position of facial feature points to determine the rotation angles. From the experimental results in the following table, we can see that the one use the 3D position works slightly better than the one assuming zero angles. This is reasonable since the 3D position provides better approximation to the true geometry. Also, due to the fact that *only* the initial mosaic is enhanced via 3D points fitting, the improvement is not dramatic.

Table 1 Comparison of methods in initializing the mosaic model

Mosaic method with zero initial angle	Mosaic method via 3D initialization
7.32%	6.71%

4.7 Conclusions

This chapter presents an approach to train a statistical mosaic model by combining multiple training images with pose variation. Also we propose to utilize the geometric deviation model for finding the corresponding patch during the test stage. We show improved performance for pose robust face recognition by using this new method.

Our face mosaic model is a quite sophistical statistical model because of the following. First, as the hardest variation, the pose variation is handled naturally by mapping images from different view-angles to form a mosaic mean image, which can be treated as a compact representation of faces under different view-angles. Second, all the other variations that could not be modeled by the mean image, for example, illumination and expression if they present in the training images, are taken care of by a number of eigenvectors. Therefore, instead of modeling only one type of variations as the conventional methods, our method is trying to model all

possible appearance variations in only one model. Third, as a simple geometrical assumption of the ellipsoid model, it has the problem of over-simplification since the human head is not truly an ellipsoid. This is taken care of by training a geometric deviation model, which results in better corresponding across multiple texture maps.

Having shown the application of pose robust face recognition, we would like to apply our mosaic model for video-based face recognition as well, which involves face tracking and recognition from video sequences. We will present it in the next chapter.

5. Video-based Face Recognition

In traditional image-based face recognition, usually the face area is cropped before feeding to a recognition system. However, in video-based face recognition, given a video sequence containing human faces, we have to *track* the face over sequences before any recognition task can proceed, which normally involves two different tasks: face tracking and face recognition. One computation efficient way is to combine these two tasks together. That is, by using the *same* model for both tracking and recognition, these two tasks can be performed simultaneously. Since this same model has to serve the purpose of both tracking, which requires a simple model for achieving real-time tracking efficiency, and recognition, which requires a specific model containing enough variations about the identity.

As we presented in the previous chapters, since our face mosaic model is a simple statistical model combining both appearance and geometric information, it is a good candidate for serving face tracking and recognition simultaneously. In this chapter, we will focus on how to use the mosaic model for video-based face tracking and recognition.

5.1 Face tracking using the mosaic model

Given one video frame, the most important task in all the tracking, recognition and online model training is to generate a texture map and compare it with the mosaic model, which results in the similarity between this frame and the model. Since the mapping parameter $\mathbf{x}$ contains all

the information for generating the texture map, the goal of face tracking is to estimate the optimal $\mathbf{x}$, which can result in the minimal distance (or maximal similarity) between the texture map and the mosaic model. In one word, the face tracking is equivalent to estimating $\mathbf{x}$.

There are two methods for estimating the mapping parameter $\mathbf{x}$: the condensation method [29] and the Levenberg-Marquardt algorithm [63], which is similar to the gradient decent method.

5.1.1 Face tracking via the condensation method

As we said before, the goal in face tracking is to estimate the mapping parameter $\mathbf{x}$ based on the current frame $\mathbf{f}_n$ and the mosaic model $\prod$. The basic idea of the condensation method is that instead of directly estimating $\mathbf{x}_n$ given each frame $\mathbf{f}_n$, it estimates the conditional *probability density function* (PDF) $p(\mathbf{x}_n \mid \mathbf{f}_n)$. The name "condensation" refers to "conditional density compensation", which means to compensate or propagate the conditional PDF $p(\mathbf{x}_{n-1} \mid \mathbf{f}_{n-1})$ by using the knowledge from $\mathbf{f}_n$, and to obtain an estimation of $p(\mathbf{x}_n \mid \mathbf{f}_n)$.

Because in general the conditional PDF $p(\mathbf{x}_n \mid \mathbf{f}_n)$ might not be Gaussian distribution, importance sampling is used to approximate the arbitrary non-Gaussian distribution, where a set of K samples together with their weights, $\{\mathbf{x}_n^{(k)}, w_n^{(k)}\}$ $(1 \le k \le K)$, is used. As shown in Figure 29, given a set of samples $\{\mathbf{x}^{(k)}, w^{(k)}\}$, a conditional PDF $\hat{p}(\mathbf{x} \mid \mathbf{f})$ could be synthesized to approximate the original conditional PDF $p(\mathbf{x} \mid \mathbf{f})$ as follows.

$$\hat{p}(\mathbf{x} \mid \mathbf{f}) = \sum_{k=1}^{K} w^{(k)} \delta(\mathbf{x} - \mathbf{x}^{(k)}) \tag{10}$$

Thus propagation of a conditional PDF becomes the updating of the sample set, i.e., given $\{\mathbf{x}_{n-1}^{(k)}, w_{n-1}^{(k)}\}$, and $\mathbf{f}_n$, generate $\{\mathbf{x}_n^{(k)}, w_n^{(k)}\}$. As shown in Figure 30, the propagation can be accomplished in two steps.

The first step "prediction" is essentially to answer the question: "if I have not seen the current observation $\mathbf{f}_n$, what would be the most likely place that each of the sample will sit on based on the best of my knowledge about the system?" This is answered by applying the knowledge of $p(\mathbf{x}_n \mid \mathbf{x}_{n-1})$ to predict a new sample set $\{\mathbf{x}_n^{(k)}\}$ from $\{\mathbf{x}_{n-1}^{(k)}\}$. The knowledge of $p(\mathbf{x}_n \mid \mathbf{x}_{n-1})$ can come from the domain knowledge or be trained from the training data. For example, if we know the human head is moving around in all possible directions, we can use the following equation as one way of applying the domain knowledge.

$$\mathbf{x}_n = \mathbf{x}_{n-1} + \mathbf{b}_n$$

where $\mathbf{b}_n$ is a white noise with certain variance. One potential problem with this multiple samples' propagation is that some of the samples might have too tiny weights, and propagating them would not contribute much to the modeling of a conditional PDF. For this reason, people have proposed to add a "re-sampling" step before the "prediction" step, where the Monte Carlo Method is used to generate a new set of samples $\{\mathbf{x}_{n-1}^{(k)'}\}$ from $\{\mathbf{x}_{n-1}^{(k)}, w_{n-1}^{(k)}\}$. Figure 31 illustrates the procedure of the Monte Carlo Method. Basically based on $\{\mathbf{x}_{n-1}^{(k)}, w_{n-1}^{(k)}\}$ and (10), we can obtain an estimated conditional PDF $\hat{p}(\mathbf{x} \mid \mathbf{f})$, from which a cumulative density function (CDF) is generated. Finally, by looking at which bin a random number is falling into, we can generate a set of samples $\{\mathbf{x}_{n-1}^{(k)'}\}$ that fit with this conditional PDF. For example, if a random number is in the range of the red bin, $\mathbf{x}_{n-1}^{(6)}$ will be a new sample in $\{\mathbf{x}_{n-1}^{(k)'}\}$.

In the second step "weight assignment", each sample $\mathbf{x}_n^{(k)}$ is assigned with a new weight $w_n^{(k)} = p(\mathbf{f}_n \mid \mathbf{x}_n^{(k)})$, which measures how likely the current frame $\mathbf{f}_n$ can be observed based on this particular sample $\mathbf{x}_n^{(k)}$. This likelihood measure is calculated by generating a texture map $\mathbf{s}_n$ based on $\mathbf{x}_n^{(k)}$ and $\mathbf{f}_n$, as presented in Section 3.1, and calculating the similarity measure between

$\mathbf{s}_n$ and the mosaic model $\prod$, as presented in Section 4.4. Eventually the new weights are normalized such that the total weights of all samples equals to one.

After the propagation, the weighted mean of the new sample set $\{\mathbf{x}_n^{(k)}, w_n^{(k)}\}$ becomes the current estimated $\mathbf{x}_n$. When the next frame $\mathbf{f}_{n+1}$ arrives, we will start the same estimation procedure based on the current sample set $\{\mathbf{x}_n^{(k)}, w_n^{(k)}\}$, which essentially carries all the statistical information of $\mathbf{x}_n$, and is propagated to future frames. We implement this algorithm for face tracking and observed reasonable good tracking results.

For face tracking in a video sequence, normally it is assumed that the tracking result of the first frame is available before the tracking starts. This result might come from the face detection or manual labeling. Notice in the condensation method, we also need to initialize a set of samples of their corresponding weights $\{\mathbf{x}_1^{(k)}, w_1^{(k)}\}$, which are obtained from the tracking result of the first frame, $\mathbf{x}_1$. Basically we generate random samples $\{\mathbf{x}_1^{(k)}\}$ around $\mathbf{x}_1$, and then assign weights according to the similarity measure between the texture maps from $\{\mathbf{x}_1^{(k)}\}$ and the mosaic model $\prod$.

Notice in previous chapters, for the training and test images in a recognition system, we generate their texture maps based on the universal texture map by using the condensation method. This procedure is actually the same as tracking *one* frame *without* a good sample set from the previous frames. Obviously, in this case we need to use more samples in order to have a good estimation of the mapping parameter of one face image. In our thesis proposal, we were using the Hidden Markov Model (HMM) to model the mapping parameters, which is not necessary anymore since the condensation method is an extension of the HMM. Because Gaussian assumption is made in the HMM, while the condensation method does not make such an assumption.

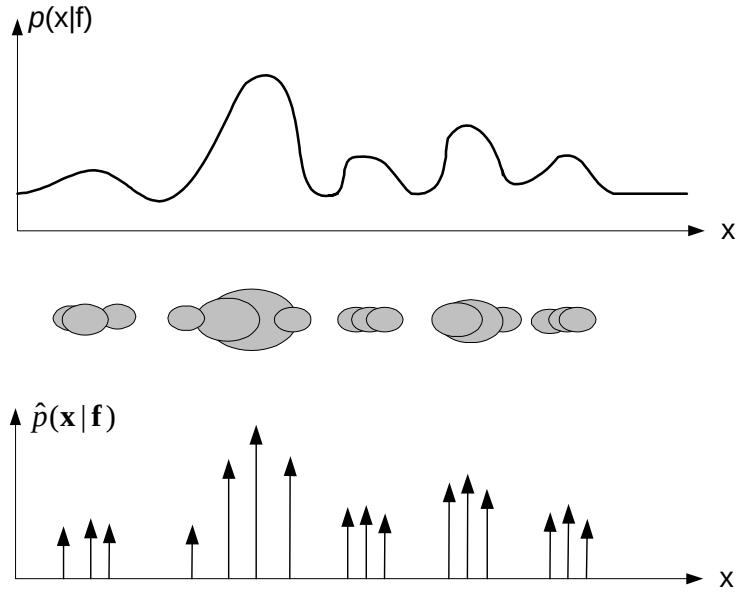

Figure 29 Importance sampling: represent a non-Gaussian PDF using a set of samples and corresponding weights (indicated by the size).

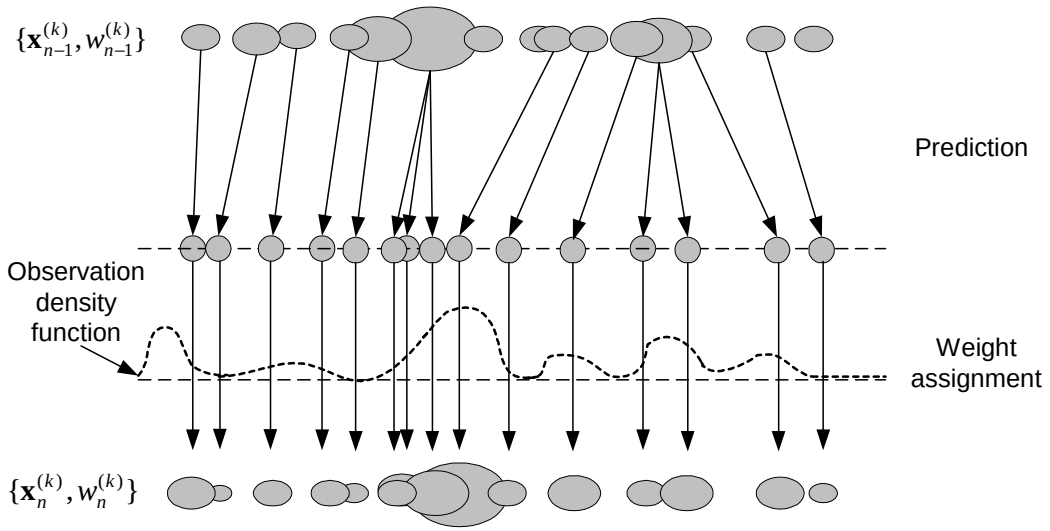

Figure 30 Two steps for density PDF propagation: prediction step estimates the new position of each sample and the weight assignment step assign weights for each sample based on the observation density function.

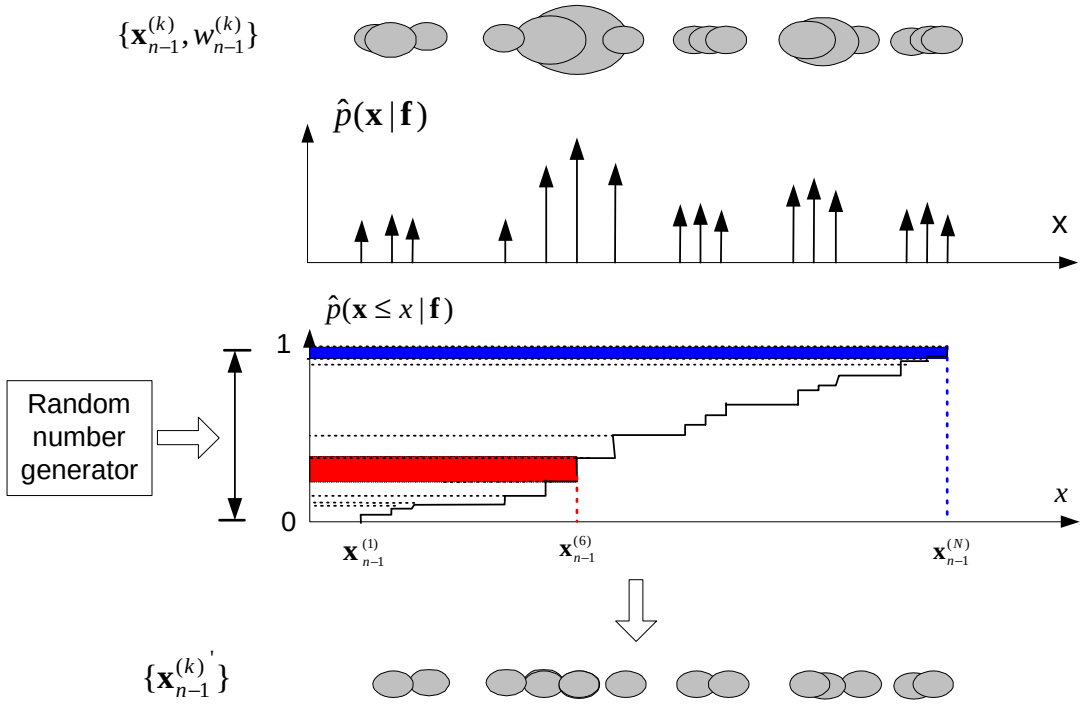

Figure 31 Monte Carlo method: a PDF is approximated by generating a set of samples with uniform weights.

5.1.2 Face tracking via the Levenberg-Marquardt algorithm

Having introduced the condensation method, let us look into another tracking algorithm, the Levenberg-Marquardt algorithm. This method is especially useful when the mosaic model is trained without the deviation model and the probabilistic model. In this case, since there is no notion of patch representation, the mosaic model can be simply represented as *one* eigenspace $\Pi = \{\mathbf{m}, \mathbf{V}\}$. When the patch representation is used, we need to add the coefficient of the geometrical deviation model into the minimization process as well.

Essentially the face tracking is a minimization procedure, which is illustrated by Figure 32. The objective is to iteratively minimize the difference between the texture map $\mathbf{s}_n(\alpha, \beta)$ and the statistical model $\Pi = \{\mathbf{m}, \mathbf{V}\}$, which consists of the mean $\mathbf{m}$ and q eigenvectors $\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_q\}$. Since the mapping parameter $\mathbf{x}_n$ controls the texture map $\mathbf{s}_n$, this

minimization is over the parameter $\mathbf{x}_n$. The minimization will stop if the following distance is small enough, otherwise it will keep updating the mapping parameter $\mathbf{x}_n$.

$$\min_{\mathbf{x}_n} J = \left\| \mathbf{w} \circ (\mathbf{s} - \mathbf{m} - \mathbf{vc}) \right\|^2 = \sum_{\alpha,\beta} e^2 \tag{11}$$

$$\mathbf{c} = (\mathbf{v}^T diag(\mathbf{w}^2)\mathbf{v})^{-1} (diag(\mathbf{w}^2)\mathbf{v})^T (\mathbf{s} - \mathbf{m})$$

where $\circ$ refers to the element-wise multiplication, *diag*() generates a matrix whose diagonal element is the input vector, and **c** is the eigen-coefficient of $\mathbf{s}_n$ with respect to the mosaic model. The parameter **w** is the mask map for $\mathbf{s}_n$, which combines the mask information from two sources. One is the mask map for the original input image $\mathbf{f}_n$. The other is the mask map from the mosaic model.

We adopt the Levenberg-Marquardt algorithm to find the optimal mapping parameter $\mathbf{x}_n$ that minimizes (11). This algorithm requires the computation of the partial derivatives of *e* with respect to all unknown parameters in $\mathbf{x}_n$, for example:

$$\frac{\partial e}{\partial R_\alpha} = \frac{\partial e}{\partial \mathbf{s}} \left(\frac{\partial \mathbf{s}}{\partial \alpha} \left(\frac{\partial \alpha}{\partial p_x^{'}} \frac{\partial p_x^{'}}{\partial R_\alpha} + \frac{\partial \alpha}{\partial p_y^{'}} \frac{\partial p_y^{'}}{\partial R_\alpha} \right) + \frac{\partial \mathbf{s}}{\partial \beta} \left(\frac{\partial \beta}{\partial p_x^{'}} \frac{\partial p_x^{'}}{\partial R_\alpha} + \frac{\partial \beta}{\partial p_y^{'}} \frac{\partial p_y^{'}}{\partial R_\alpha} \right) \right)$$

where

$$\frac{\partial e}{\partial \mathbf{s}} = diag(\mathbf{w})(\mathbf{I} - \mathbf{vv}^T),$$

$$\frac{\partial \alpha}{\partial p_x^{'}} = 0,$$

$$\frac{\partial \alpha}{\partial p_y^{'}} = -\frac{1}{\sqrt{r^2 - p_y^{'2}}},$$

$$\frac{\partial \beta}{\partial p_x^{'}} = \frac{\sqrt{r^2 - p_y^{'2}}}{p_z^{'2}},$$

$$\frac{\partial \beta}{\partial p_y^{'}} = \frac{p_x^{'}}{p_z^{'2}\sqrt{r^2 - p_y^{'2}}},$$

$$\frac{\partial p_x^{'}}{\partial R_\alpha} = p_y^{'} \cos(R_\alpha)\sin(R_\beta) - p_z^{'} \sin(R_\alpha)\sin(R_\beta),$$

$$\frac{\partial p_y^{'}}{\partial R_\alpha} = -p_y^{'} \sin(R_\alpha) - p_z^{'} \cos(R_\alpha),$$

and $\frac{\partial \mathbf{s}}{\partial \alpha}$ and $\frac{\partial \mathbf{s}}{\partial \beta}$ are the image intensity gradients of $\mathbf{s}_n$ at (α, β). With these partial derivatives, we can calculate an approximate Hessian matrix **A** and the weighted gradient vector **b** [63]. For simplicity, if there are two parameters, R_α and R_β, in $\mathbf{x}_n$,

$$\mathbf{A} = \begin{bmatrix} \sum \frac{\partial e}{\partial R_\alpha}\frac{\partial e}{\partial R_\alpha} & \sum \frac{\partial e}{\partial R_\alpha}\frac{\partial e}{\partial R_\beta} \\ \sum \frac{\partial e}{\partial R_\beta}\frac{\partial e}{\partial R_\alpha} & \sum \frac{\partial e}{\partial R_\beta}\frac{\partial e}{\partial R_\beta} \end{bmatrix}$$

$$\mathbf{b} = \begin{bmatrix} -\sum e\frac{\partial e}{\partial R_\alpha} & -\sum e\frac{\partial e}{\partial R_\beta} \end{bmatrix}^T$$

Then the parameters $\mathbf{x}_n$ can be linearly updated by

$$\Delta \mathbf{x}_n = (\mathbf{A} + \lambda \mathbf{I})^{-1}\mathbf{b} \tag{12}$$

This algorithm consists of the following steps:

1. Assign the initial value for $\mathbf{x}_n$.
2. Compute $\mathbf{s}_n$ and $\mathbf{w}$ according to Section 3.1**.**
3. Compute the error e as in (11) and the intensity gradient on $\mathbf{s}_n$, computer the partial derivative of e with respect to $\mathbf{x}_n$, and compute **A** and **b**.
4. Linearly update $[\Delta\alpha, \Delta\beta]^T$ by $\Delta \mathbf{x}_n$ calculated in (12).
5. Evaluate (11) using the updated parameters and check whether the error J decreases; if not, increase λ as described in [63], and compute a new $\Delta \mathbf{x}_n$.
6. Continue the iteration until the parameters converge or a fixed number of steps are finished.

If we compare the Levenberg-Marquardt algorithm with the condensation method, we can see that the former is similar to the gradient decent method, which tries to move toward the global minimal point on the error surface as fast as possible from a initial point, while the latter is a statistical method, which starts with many points (samples) on the error surface, moves each of them toward their best locations, and takes the averaged location as the tracking results.

Normally the Levenberg-Marquardt algorithm is more likely to be trapped into the local minimal since only one point is moving around on the error surface and it might starts with a *bad* initial point. On the other hand, due to its statistical nature, the condensation method is more robust in terms of tracking performance, because as long as some of the samples are closer to the true global minimal, they will be responded by high weights and the result would be pretty good already. However, the drawback of the condensation method is also due to its statistical nature. That is, since many samples are used for tracking, the computation load of the condensation method is usually higher than the Levenberg-Marquardt algorithm, which can converge in usually a few iterations. In summary, we can see that these two methods are complementary to each other in terms of tracking performance and computational efficiency.

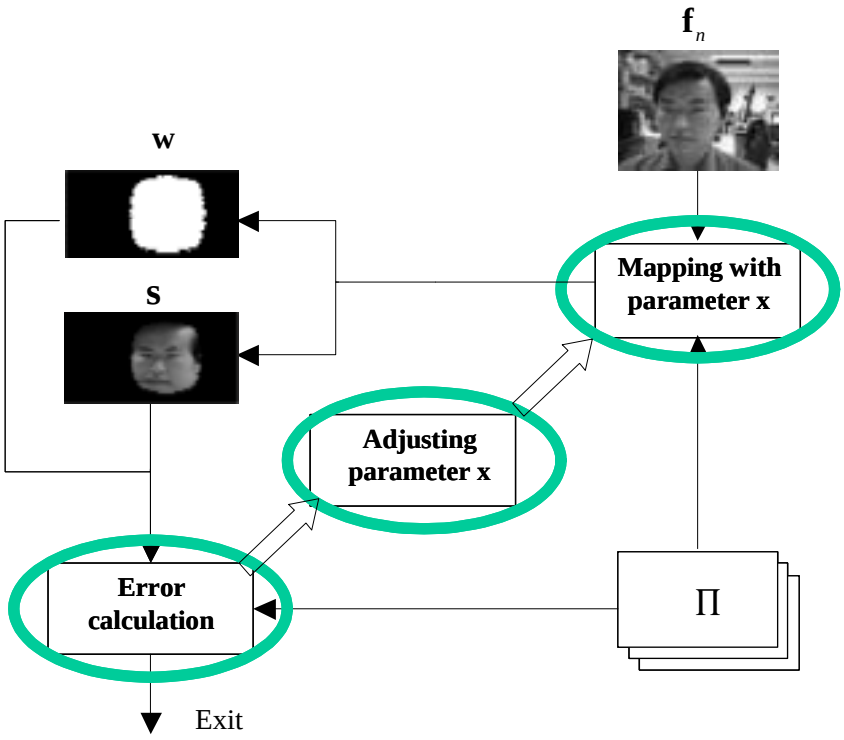

Figure 32 Tracking via the Levenberg-Marquardt algorithm: the mapping parameter is iteratively adjusted in order to minimize the distance between the texture map and the mosaic model.

5.2 Face recognition

There are two different schemes for performing face tracking and recognition from video sequences.

First is to use the image-based method. For a face database with L subjects, we build the individualized model for each subject, based on one or multiple training images. Given a test sequence and one specific model, a distance measurement can be calculated for each frame by face tracking. Averaging of the distance over all frames in the sequence provides the distance between the test sequence and one specific model. After the distances between the sequence and all models are calculated, we can obtain the recognition result for this sequence by comparing distances across subjects.

Second is to use the video-based method. Zhou et. al. [82] propose a framework to combine the face tracking and recognition using the condensation method. They basically propagate a set of samples governed by two parameters: the mapping parameter and the subject ID. Thus we call it as the 2D condensation method, as shown in Figure 33.

There are at least three benefits of using video-based recognition comparing to image-based recognition in using the condensation method. First, during the weighting normalization step, the conventional condensation method normalizes weights of all samples of one subject, while the 2D condensation method normalizes weights of all samples of *all* subjects. Thus the samples of the matched subject would have relative larger weights than samples of non-matched subjects. In the mean time, the weights of the samples of non-matched subjects are depressed, which is what we want.

Second, the set of samples with the same mapping parameters is assigned for all subjects. On one hand, it reduces the computation of evaluating the weights based on each sample because the geometrical mapping operation is the same. On the other hand, samples for non-matched subjects are not allowed to move freely, whose movement is mainly governed by samples of the matched subject.

Third, the 2D condensation method might be able to handle the open-set recognition problem. Due to the weight normalization, it is likely that no subject shows dominant weights if the test subject is not included in the training set. Otherwise the samples of the matched subject should have dominant weights comparing to samples of non-matched subjects. As shown in Figure 34, we perform a simple experiment to show this point. Given a face database with 29 subjects, if the test frame comes from one of the 29 subjects (i.e., this is a close-set recognition), the total probabilities of all samples from the matched subject is much larger than the ones from other subjects. However, in the close-set recognition, since the test frame does not match with any subjects in the database. No dominant probability is observed.

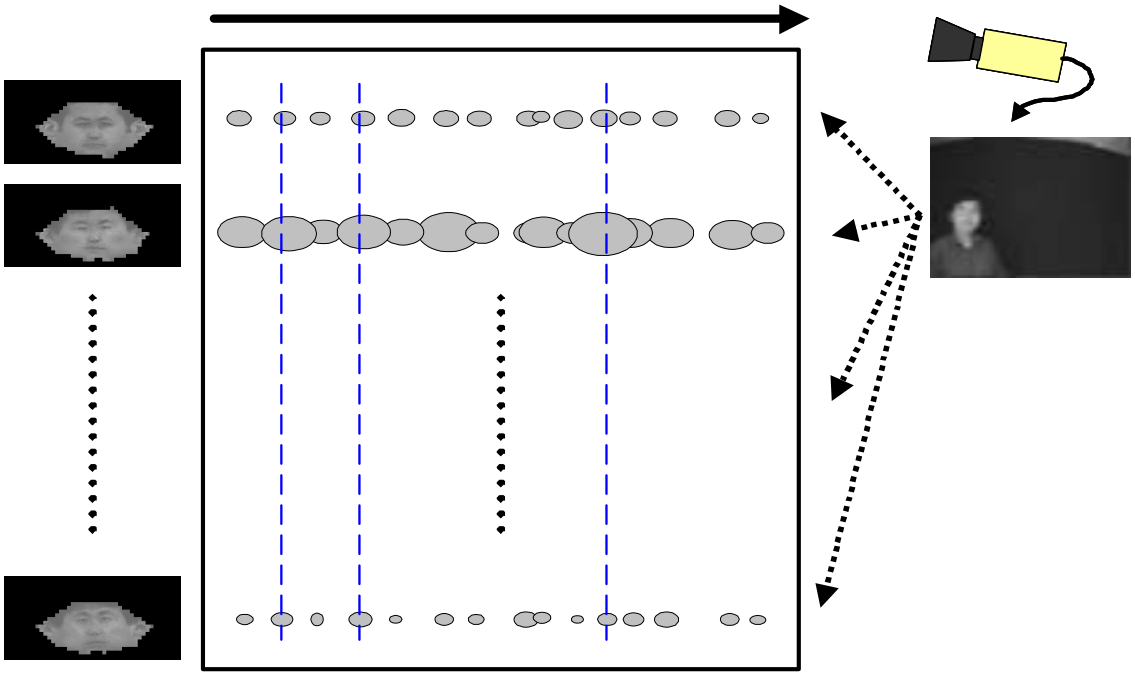

Figure 33 Basics of video-based recognition

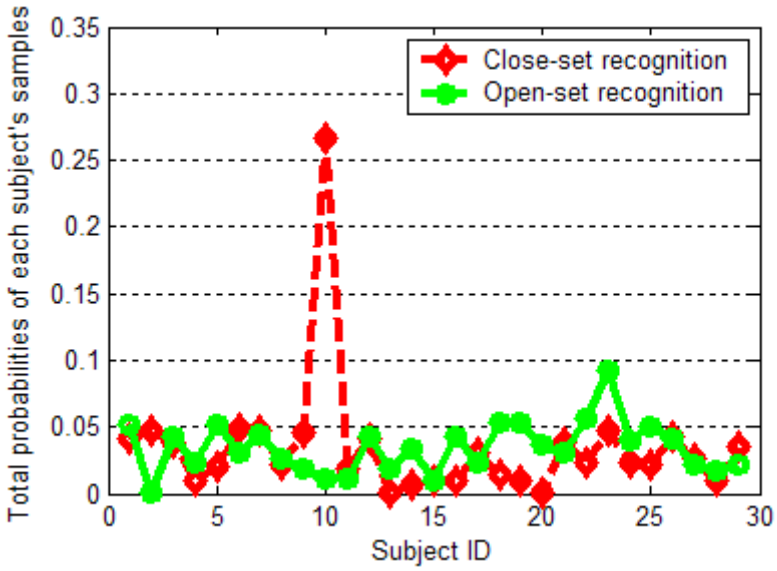

Figure 34 The different between close-set and open-set recognition using the 2D condensation method.

Let us introduce the basic step in the 2D condensation method using Figure 35. Given L subjects in the training set, the individualized model is built for each subject. Suppose we use a set of K samples for modeling the mapping parameter. In the initial status, there are $L \cdot K$ samples in the 2D space. The first step is to select the top K samples (red circles) that have the largest weights among all $L \cdot K$ samples. Then these K samples are predicted to a new location according to a certain model. Second, L samples are duplicated for all subjects based on each one of K samples. In other words, all L samples share the same set of mapping parameters. Third, the same mapping parameter from L samples would result in the same texture map, which greatly saves the computation load of geometrical mapping. The texture map will have different similarity with respect to L different models. Thus different weights are assigned to each sample. Finally, the subject who has the maximum total weights from K samples will be the recognition result.

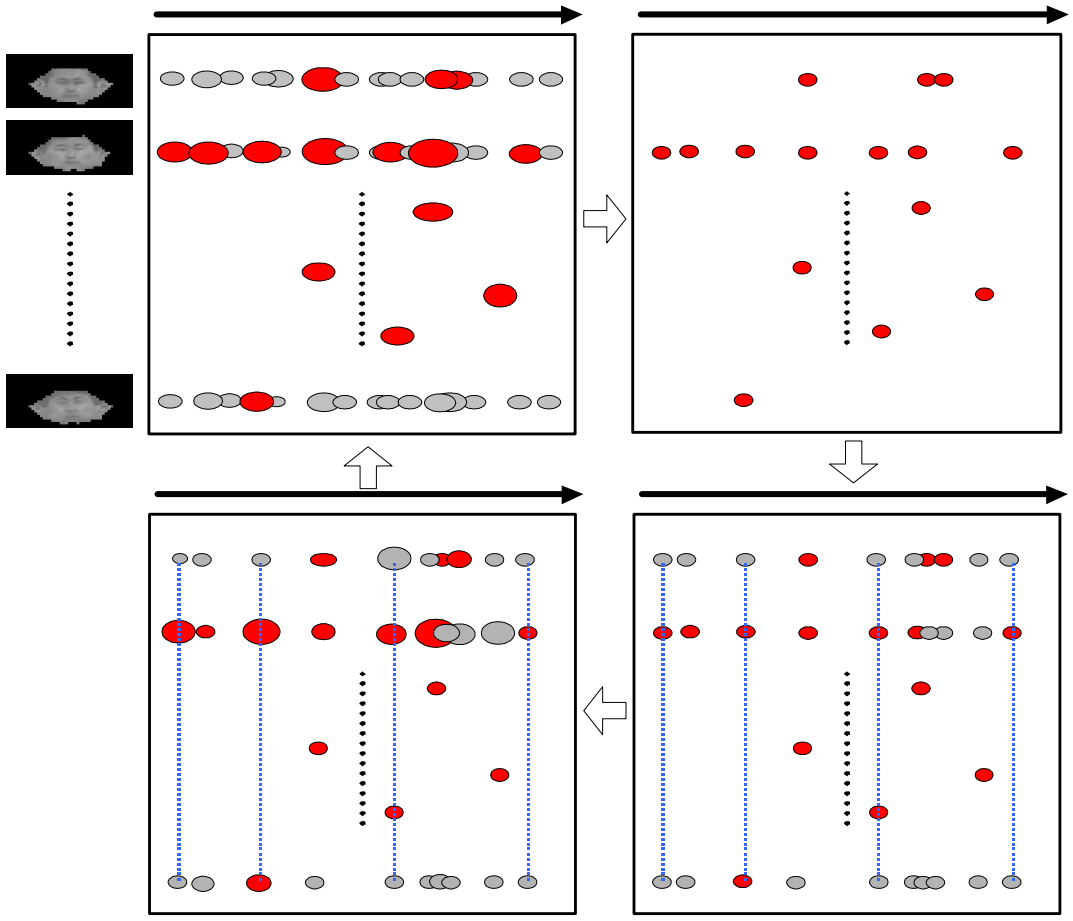

Figure 35 Propagation steps for video-based recognition

Before we present the experimental results, first we will introduce the Face-In-Action video database we are collecting.

5.3 Face-In-Action video database

As more and more researchers are starting to work on video-based face recognition, as opposed to traditional image-based face recognition, there is more demand for a database of video sequences containing human faces. With such a database, the benefits of video-based face recognition can be explored. We are making the effort to collect such a face video database, called Face In Action (FIA) database [49].

5.3.1 **Capturing scenario**

There are many existing databases containing face images under controlled conditions, such as FERET[60], PIE [67], ORL [64], Xm2vts [57], etc. However, when collecting a face database in videos, we have to bring in *motion*. Based on our study, we consider "passport checking" as the most popular motion scenario for real-world applications of face recognition techniques. As shown in Figure 36, in a controlled environment with the blue background, multiple cameras are pointing at the desk from three different angles. The cameras capture the whole process of the subject's walking approaching the desk, standing in front of the desk, making simple conversation, head motion that might happen during passport checking, and finally walking away from the desk. The resulting video hence contains the moving head while the subject is walking, user-dependent pose variation due to natural motion of the head, lip movements and expression variations during conversation.

Actually this capturing scenario is not only mimicking "passport checking", it is also highly representative for many other daily scenarios, such as checking in a hotel, visiting the hospital or governmental offices, etc.

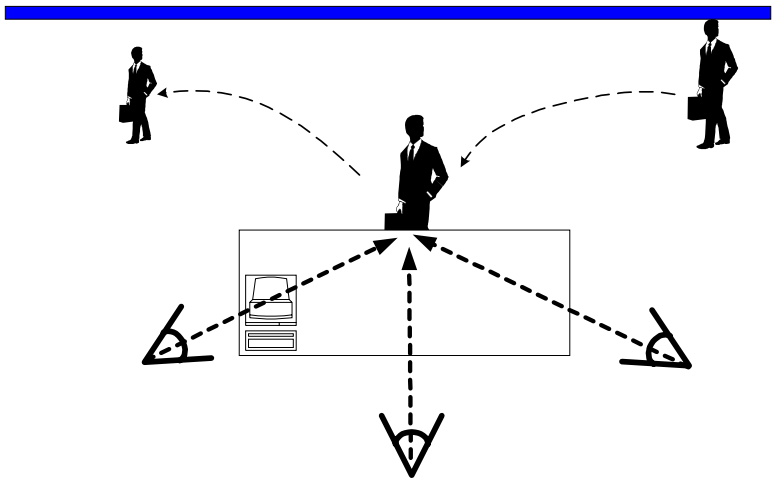

Figure 36 FIA capturing scenario: multiple cameras are capturing faces while the subject is mimicking in going through the airport "passport checking".

5.3.2 **Capturing system**

In face database collection, one samples the face in multiple dimensions, such as pose, illumination, expression, aging, etc. In our FIA capturing system, we sample in the following dimensions: motion, pose, image resolution, illumination and variations over time. Motion is sampled by continuous videos at 30 frames per second. Pose is sampled by capturing faces from three difference directions simultaneously. The image resolution is sampled by using cameras with two different focus lengths. Illumination is sampled by capturing faces in both indoor and outdoor scenarios. Variations over time are sampled by capturing three different sessions each spanning three months.

As shown in Figure 37, we built a cart for mounting the capturing system. On the C-shape arm, there are 6 cameras. All cameras are pointing to the same center spot and have the same distance (0.83M) to that spot. Each camera is able to capture video sequences with 640 by 480 frame size in 30 frames per second. Six cameras are arranged into three pairs. Since the C-shape arm can be adjusted vertically by the linear bearing according to the height of the subject, a face is essentially captured by three pairs of cameras with the same vertical angle but different horizontal angles (-60°, 0°, 60°) respectively. Within each pair of cameras, one has 4mm focal-length, which results in the face area with around 300 by 300 pixels, and the other has 8mm focal-length, which results in the face area with around 100 by 100 pixels. The video sequence with larger face area can be used for applications demanding high-resolution face images, such as 3D reconstruction, while the smaller one is closer to the face data in video surveillance applications. Figure 38 shows the picture of the camera cart. Three light bulbs are placed on the cart so as to create an ambient lighting environment for capturing.

Two carts are used for capturing human faces in the indoor and outdoor scenario respectively. There are three differences between the indoor and outdoor scenario. First is that

there is no controlled illumination in the outdoor scenario. Second is that no blue background will be placed for the outdoor scenario. Third is that neither color nor camera calibration is performed for the outdoor scenario. Thus the sequences from the outdoor scenario can be used to study how well the video-based face recognition performs in the natural illumination. To capture variations over time, we are planning to capture 200 subjects in three different sessions each spanning three months. For one session, both indoor and outdoor scenario will be captured. Six sequences are captured simultaneously for 20 seconds in each scenario.

Having introduced the camera cart, we now present the system configuration, as shown in Figure 39. We use the Dragonfly™ camera from Point Grey Research Inc.[62], which is an OEM-style IEEE-1394 board level camera. Based on the data rate we are capturing (640 by 480 by 30 frames by 20 seconds), one IEEE-1394 bus can only allow the data stream of three cameras. Although three cameras on the same bus are synchronized, we would have to synchronize two buses and thus all six cameras are synchronized. The SYNC Unit [62] plays the role of synchronizing two different IEEE-1394 buses. Eventually all six camera streams are saved onto the hard driver of one computer. Based on our experiences, the speed of the hard drive, rather than the CPU speed, is the bottleneck of the capturing system. Currently we use more memory as the cache to compensate the not-fast-enough hard driver.

For each subject, we collect the following data: six 20 seconds face sequences at 30 frames per second, for both indoor and outdoor scenarios, for 3 sessions in total. We store personal information for each subject, such as, age, gender, glasses, beard, mustache, etc. Also for each indoor scenario, we provide the color calibration data using the color checker, and the camera calibration data using the check board.

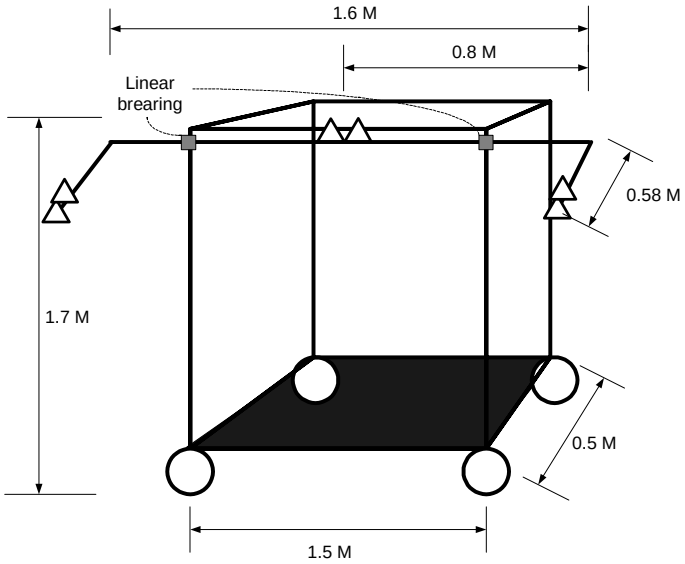

Figure 37 The design of the camera cart: six cameras are grouped into three pairs and mounted on a height-adjustable arm.

Figure 38 Camera cart and lights: 3 light bulbs are used to create an ambient lighting environment.

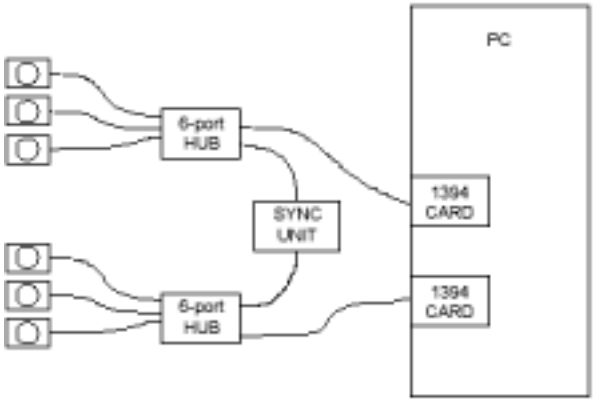

Figure 39 System configuration: six cameras are connected to two IEEE-1394 buses on the computer; the SYNC unit synchronizes two buses.

5.3.3 Specifications and samples

In summary, the specification of the FIA database are listed in the following:

- 200 subjects.
- 3 sessions per subject.
- 2 scenarios per session (indoor and outdoor).
- Color and camera calibration data for the indoor scenario.
- 6 sequences per scenario.
- 20 seconds per sequence.
- 30 frames per second.
- 640 by 480 24-bits color image per frame
- Storing mage data for each subject, such as age, gender, etc.
- Saving each image in JPEG format with 90% quality (100K).
- Total storage of the database: 100k*30*20*6*2*3*200=412G.

One sample snapshot from six cameras can be seen in Figure 40. Three images in the top row are captured by 8mm focal-length cameras. The others three images are captured by 4mm focal-length cameras. Figure 41 shows the sample images from one sequence in the FIA database. Substantial pose variation can be observed from this sequence.

Figure 40 A sample snapshot from 6 cameras: top images are from cameras with longer focal-length; bottom images are from cameras with short focal-length; each column are images from a pair of camera neighbor to each other.

Figure 41 Sample images of one sequence in the FIA database: substantial pose variation can be observed from this database.

5.4 Experimental results

5.4.1 Face tracking

Since a face mosaic model describes the facial appearance from multiple views, we can use it for performing pose robust face recognition. There are two methods for model-based face tracking. One is to use the condensation method. The other is to use the Levenberg-Marquardt

algorithm. The second option is faster, however might have slightly worse performance comparing to the first one. We use the patch-PCA or global-PCA mosaic model for the first option depending on the number of training images. In the second option, we can use the mosaic model without deviation modeling, for targeting at fast tracking speed.

The tracking result of one sequence using the individual patch-PCA mosaic is shown in Figure 42. The while circle shows the result of the face position, and two curves show the result of horizontal and vertical rotation angles. We can see these two curves always across the eyes and nose area across frames. We use 500 samples in the condensation method. The tracking can be performed at around 2 frames per second.

Figure 42 Tracking results based on the patch-PCA mosaic model: horizontal and vertical line indicates the estimated pose in two directions.

5.4.2 Face recognition

We have performed an experiment on the FIA database. There are 29 subjects in the database, with 10 sequences per subject as the test sequences. Each sequence has 50 frames, and the first frame is labeled with the ground truth data. We use the individual PCA algorithm with the image-based recognition and the individual PCA with the video-based recognition as the baseline algorithms. For both algorithms, 9 images are used for training and the best

performances are reported by trying different number of eigenvectors. The algorithms work best when the number of eigenvectors is 4. For example, Figure 43 shows the 9 training images for one subject in the FIA database. The face location of training images is from the manual labeling, while that of the test images is based on the tracking results using our mosaic model. All face images are cropped to be 64 by 64 pixels from the video frame.

For our algorithms, we tested three different options. First is to use the individual patch-PCA mosaic with image-based recognition, which uses the averaged distance between the frames to the mosaic model as the final distance measure. There are 9 images per subject as the training images. Second is to use the individual patch-PCA mosaic with video-based recognition, which uses the 2D condensation method to perform tracking and recognition. The same set of training images are used. The third is similar to the second option except that only one training image per subject is used. Thus only one texture map from each training image can be used for training. We illustrate the mean images of the individual models in three methods. We can observe dramatic blurring effect in the mean from the individual PCA model. On the other hand, the mean of our individual patch-PCA mosaic model covers large pose variation while still keeps enough individual facial characteristic. Since there is only one front training image in the third option of the mosaic method, the mean is only the texture map of that image.

The recognition performance of the baseline algorithms and our approaches are shown in the following table. Two observations can be made from here. First, given the same model, such as the PCA model or the mosaic model, video-based face recognition is better than the image-based recognition. Second, the mosaic model works much better than the PCA model for pose-robust recognition. The third option works worse than the first two options since it has only training image per subject.

Figure 43 9 training images from one subject in the FIA database

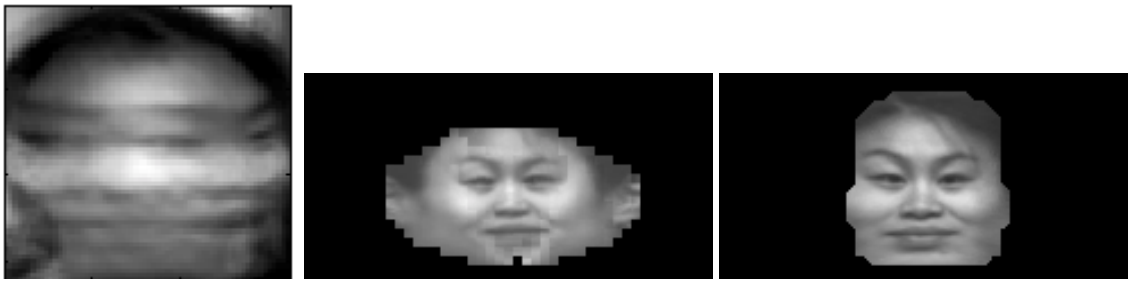

Figure 44 The means of individual model in three methods (left: Individual PCA trained from 9 images, middle: mosaic model trained from 9 images, right: mosaic model trained from 1 image).

Table 2 Recognition error rate of different algorithms

PCA with image-based method	PCA with video-based method	Mosaic with image-based method	Mosaic with video-based method	Mosaic with video-based method (1 training image)
17.24%	8.97%	6.90%	4.14%	9.66%

5.5 Conclusions

In this chapter, by using the face mosaic model, we are able to perform face tracking and recognition simultaneously even dramatic pose variation is present in the test sequences. We introduce the face tracking using two different algorithms: the condensation method and the Levenberg-Marquardt algorithm. We present two methods of integrated face tracking and recognition scheme: image-based method and video-based method. We also present the collection effort of the FIA face video database. We apply our algorithm on the FIA database and obtain satisfying tracking and recognition performance.

The strength of this approach is that simple geometric mapping is used to compensate the large pose variation, which makes texture maps have less intra-variations across poses comparing to the original image domain. Automatically face tracking and recognition can be performed from video sequences with large pose variation.

6. Face Recognition via Updating Mosaic Model

When applying the individual PCA approach for face recognition, we propose an eigenspace updating algorithm [51], which results in an updating-during-recognition scheme. That is, the eigenspace for each subject is updated by test images while each of them being recognized. There are two reasons for doing this. First, in many applications it is not feasible to capture many training images for each subject containing enough variations for statistical modeling of that subject. Usually only one or a few images under the normal condition are available for training. Thus, it would be better if more and more images of that subject are used to update its model during the testing stage. Secondly, people change their appearance over time. Even if there are many images available for training, the system may not recognize faces when a subject changes the appearance due to aging, expression, pose, and illumination changes. A recognition system that is able to learn the changing appearance of the subject and adapt to it can achieve better performance. From our experimental results [51], significant improvements are observed on recognizing face images with different variations, such as pose, expression and illumination variations.

We believe this idea of updating-during-recognition can also be applied to the face mosaic models. Due to the limitation of training images, our face mosaic model might only learn part of facial pose variation, or it only learns pose variation under the *same* expression. In this case, by taking more images from the test sequence, which contains pose variation or expression

that have not been seen, our mosaic model can be enhanced and eventually results in a better recognition system.

In this chapter, we first present the theory of updating eigenspaces. Then we present the experimental results of updating the mosaic model for face recognition application.

6.1 Eigenspace updating with decaying memory

We present three methods of performing eigenspace updating with decaying memory. When the dimension of the feature space is not high, we will use the updating based on the covariance matrix. Otherwise we will use the updating based on the inner-product matrix. Since the mosaic model is trained from the texture maps with missing data, the third algorithm will deal with the special case of updating the PCA model with missing data.

6.1.1 Updating based on the covariance matrix

Suppose there is a random process $\{\mathbf{x}_n\}$, where n is the time index, $\mathbf{x}_n$ is a column vector in a d-dimensional space, of which we want to find the eigenspace. Each sample will be available sequentially over time. If this random process is stationary, we can estimate its mean by the following equation:

$$\hat{\mathbf{m}} = \frac{\mathbf{x}_n + \mathbf{x}_{n-1} + \cdots + \mathbf{x}_1}{n}$$

If $\mathbf{x}_n$ is a non-stationary random process, which implies that it has a time-varying mean $\hat{\mathbf{m}}_n$, we propose to estimate the mean at time n as:

$$\hat{\mathbf{m}}_n = \frac{\mathbf{x}_n + \alpha_m \mathbf{x}_{n-1} + \alpha_m^2 \mathbf{x}_{n-2} + \cdots}{1 + \alpha_m + \alpha_m^2 + \cdots} \tag{13}$$

where α_m is the *decay parameter*. It controls how much the previous samples contribute to the estimation of the current mean. Since α_m is in the range of 0 to 1, we have:

$$1+\alpha_m+\alpha_m^2+\cdots=\frac{1}{1-\alpha_m} \tag{14}$$

Using (14) in (13), the resulting equation can be simplified to:

$$\hat{\mathbf{m}}_n=\alpha_m\hat{\mathbf{m}}_{n-1}+(1-\alpha_m)\mathbf{x}_n \tag{15}$$

This equation reveals that based on the current sample and the previously estimated mean, we can obtain the new estimated mean in a recursive manner. How to choose α_m mainly depends on the knowledge of the random process. Note that α_m controls how fast we want to forget about the old samples. Therefore, if the statistics of the random process change fast, we choose a small α_m. If the statistics change slowly, a large α_m may perform better.

After the mean of the random process is estimated, we can estimate the covariance matrix, $\hat{\mathbf{C}}_n$, at each time n by:

$$\hat{\mathbf{C}}_n=\frac{(\mathbf{x}_n-\hat{\mathbf{m}}_n)(\mathbf{x}_n-\hat{\mathbf{m}}_n)^T+\alpha_v(\mathbf{x}_{n-1}-\hat{\mathbf{m}}_{n-1})(\mathbf{x}_{n-1}-\hat{\mathbf{m}}_{n-1})^T+\alpha_v^2(\mathbf{x}_{n-2}-\hat{\mathbf{m}}_{n-2})(\mathbf{x}_{n-2}-\hat{\mathbf{m}}_{n-2})^T+\cdots}{1+\alpha_v+\alpha_v^2+\cdots}$$

where α_v is also a decay parameter, which is chosen based on how fast the covariance of a random process is changing. Now we can rewrite $\hat{\mathbf{C}}_n$ in a similar manner as $\hat{\mathbf{m}}_n$:

$$\hat{\mathbf{C}}_n=\alpha_v\hat{\mathbf{C}}_{n-1}+(1-\alpha_v)(\mathbf{x}_n-\hat{\mathbf{m}}_n)(\mathbf{x}_n-\hat{\mathbf{m}}_n)^T \tag{16}$$

Since we obtain $\hat{\mathbf{C}}_n$ at time n, we can perform PCA for $\hat{\mathbf{C}}_n$ and obtain the corresponding eigenvectors. We keep N eigenvectors corresponding to the N largest eigenvalues. In the recursive updating process, we only need to store the mean vector $\hat{\mathbf{m}}_n$ and the covariance matrix $\hat{\mathbf{C}}_n$. All the previous training samples can be discarded.

6.1.2 Updating based on the inner-product matrix

In many applications, PCA is applied directly in the image domain, such as face recognition. Suppose the face image has a size of 32 by 32, then the covariance matrix of an image set would be 1024 by 1024. It is very inefficient to store and update it using the algorithm introduced in Section 6.1.1. To solve this problem, we propose an updating algorithm based on the inner-product matrix.

Suppose at time n, we already have performed PCA for the random process at time $n-1$. Thus we have eigenvectors, $\boldsymbol{\varphi}_{n-1}^{(i)}$, and eigenvalues, $\lambda_{n-1}^{(i)}$, of the covariance matrix, $\hat{\mathbf{C}}_{n-1}$. We can write:

$$\hat{\mathbf{C}}_{n-1} = \lambda_{n-1}^{(1)}\boldsymbol{\varphi}_{n-1}^{(1)}\boldsymbol{\varphi}_{n-1}^{(1)\,T} + \lambda_{n-1}^{(2)}\boldsymbol{\varphi}_{n-1}^{(2)}\boldsymbol{\varphi}_{n-1}^{(2)\,T} + \cdots + \lambda_{n-1}^{(d)}\boldsymbol{\varphi}_{n-1}^{(d)}\boldsymbol{\varphi}_{n-1}^{(d)\,T}$$

where eigenvalues, $\lambda_{n-1}^{(i)}$, are sorted in the decreasing order and the superscript (i) indicates the order of eigenvalues. By retaining only the first Q eigenvectors (with the largest eiegnvalues), we can approximate $\hat{\mathbf{C}}_{n-1}$ as

$$\hat{\mathbf{C}}_{n-1} \approx \lambda_{n-1}^{(1)}\boldsymbol{\varphi}_{n-1}^{(1)}\boldsymbol{\varphi}_{n-1}^{(1)\,T} + \lambda_{n-1}^{(2)}\boldsymbol{\varphi}_{n-1}^{(2)}\boldsymbol{\varphi}_{n-1}^{(2)\,T} + \cdots + \lambda_{n-1}^{(Q)}\boldsymbol{\varphi}_{n-1}^{(Q)}\boldsymbol{\varphi}_{n-1}^{(Q)T} \tag{17}$$

The criteria for choosing Q vary, and depend on practical applications. We have tried three methods: (a) Fix Q to be a constant value; (b) Set a minimum threshold, and keep the first Q eigenvectors whose eigenvalues are larger than this threshold; (c) Keep the eigenvectors corresponding to the largest eigenvalues, such that a specific fraction of energy in the eigenvalue spectrum is retained. These methods will result in different computational complexity for the updating algorithm.

Now we can use (15) to estimate the mean at time n. By replacing $\hat{\mathbf{C}}_{n-1}$ in (16) with (17), we can obtain

$$\hat{\mathbf{C}}_n \approx \alpha_v\lambda_{n-1}^{(1)}\boldsymbol{\varphi}_{n-1}^{(1)}\boldsymbol{\varphi}_{n-1}^{(1)\,T} + \alpha_v\lambda_{n-1}^{(2)}\boldsymbol{\varphi}_{n-1}^{(2)}\boldsymbol{\varphi}_{n-1}^{(2)\,T} + \cdots + \alpha_v\lambda_{n-1}^{(Q)}\boldsymbol{\varphi}_{n-1}^{(Q)}\boldsymbol{\varphi}_{n-1}^{(Q)T} + (1-\alpha_v)(\mathbf{x}_n - \hat{\mathbf{m}}_n)(\mathbf{x}_n - \hat{\mathbf{m}}_n)^T$$

An equivalent formulation as above is that

$$\hat{\mathbf{C}}_n \approx \mathbf{B}_n \mathbf{B}_n^T$$

where

$$\mathbf{B}_n = \left[\sqrt{\alpha_v \lambda_{n-1}^{(1)}} \boldsymbol{\varphi}_{n-1}^{(1)} \quad \sqrt{\alpha_v \lambda_{n-1}^{(2)}} \boldsymbol{\varphi}_{n-1}^{(2)} \quad \cdots \quad \sqrt{\alpha_v \lambda_{n-1}^{(Q)}} \boldsymbol{\varphi}_{n-1}^{(Q)} \quad \sqrt{1-\alpha_v}\,(\mathbf{x}_n - \hat{\mathbf{m}}_n) \right] \tag{18}$$

Based on the $\mathbf{B}_n$ matrix, an inner-product matrix can be formulated as

$$\mathbf{A}_n = \mathbf{B}_n^T \mathbf{B}_n$$

Furthermore, $\mathbf{A}_n$ can be described by the following equations:

$$(A_n)_{i,j} = \alpha_v \sqrt{\lambda_{n-1}^{(i)} \lambda_{n-1}^{(j)}}\, \delta_{ij}; \quad i, j = 1,2,...,Q$$

$$(\mathbf{A}_n)_{i,Q+1} = (\mathbf{A}_n)_{Q+1,i} = \sqrt{\alpha_v (1-\alpha_v) \lambda_{n-1}^{(i)}}\, (\mathbf{x}_n - \hat{\mathbf{m}}_n); \quad i = 1,2,...,Q$$

$$(\mathbf{A}_n)_{Q+1,Q+1} = (1-\alpha_v)(\mathbf{x}_n - \hat{\mathbf{m}}_n)^T (\mathbf{x}_n - \hat{\mathbf{m}}_n). \tag{19}$$

Since the matrix $\mathbf{A}_n$ is usually a small matrix with the size of $Q+1$ by $Q+1$, we can determine its eigenvectors $\boldsymbol{\psi}_n$ by a direct method, which satisfies

$$\mathbf{A}_n \boldsymbol{\psi}_n^{(i)} = \mathbf{B}_n^T \mathbf{B}_n \boldsymbol{\psi}_n^{(i)} = \lambda_n^{(i)} \boldsymbol{\psi}_n^{(i)} \quad i = 1,2,...,Q+1 \tag{20}$$

By pre-multiplying (20) with $\mathbf{B}_n$, we can obtain the eigenvectors of matrix $\hat{\mathbf{C}}_n$ as follows:

$$\boldsymbol{\varphi}_n^{(i)} = \lambda_n^{(i)^{-\frac{1}{2}}} \mathbf{B}_n \boldsymbol{\psi}_n^{(i)} \quad i = 1,2,...,Q+1 \tag{21}$$

where the term $\lambda_n^{(i)^{-\frac{1}{2}}}$ is to make the resulting eigenvector to be a unit vector. Now we summarize the iterative updating algorithm outlined in this section:

Initialization:

1. Given the first two samples $\mathbf{x}_0, \mathbf{x}_1$, estimate the mean, $\mathbf{m}_1$, by (15), and construct the matrix

$$\mathbf{B}_1 = \left[\sqrt{\alpha_v}\,(\mathbf{x}_0 - \hat{\mathbf{m}}_1) \quad (\mathbf{x}_1 - \hat{\mathbf{m}}_1) \right]$$

2. Based on (20) and (21), we can get the eigenvector, $\boldsymbol{\varphi}_1$, and the eigenvalue, λ_1.

Iterative updating:

1. Get a new sample $\mathbf{x}_n$.
2. Estimate the mean, $\hat{\mathbf{m}}_n$, at time n by (15), and get the $\mathbf{B}_n$ matrix from (18).
3. Form the matrix $\mathbf{A}_n$ by (19) and calculate its eigenvectors, $\boldsymbol{\psi}_n$, and eigenvalues, λ_n, by a direct method.
4. Sort the eigenvalues λ_n, and retain Q corresponding eigenvectors.
5. Obtain the eigenvectors, $\boldsymbol{\varphi}_n$, at time n by (21).

We have mentioned three methods of choosing Q. If we use the second and the third methods, Q will increase as more and more training samples arrive till it reaches the intrinsic dimensionality of previous training samples. Due to the approximation in (17), among the Q eigenvectors, typically the first few eigenvectors are more precise than the others. Therefore, in practice if we need N eigenvectors for building an eigenspace, we would keep Q to be a number larger than N.

6.1.3 Updating eigenspace with missing data

In some applications, the training vector $\mathbf{x}_n$ contains the missing data. In this case, the updating of the eigenspace needs to be re-stated as following: Given a new sample $\mathbf{x}_n$ and its corresponding mask $\mathbf{a}_n$, update the existing eigenspace $\hat{\mathbf{m}}_{n-1}$, $\hat{\boldsymbol{\varphi}}_{n-1}$, and $\mathbf{w}_{n-1}$ to be $\hat{\mathbf{m}}_n$, $\hat{\boldsymbol{\varphi}}_n$, and $\mathbf{w}_n$.

Similar to what we present in Section 6.1.1, this updating can be formulated in the following three steps. First, the mean $\hat{\mathbf{m}}_{n-1}$ is updated based on the current weighting vector

and the masked test sample. Notice the division is element-wise. Also the weighting vector $\mathbf{w}_{n-1}$ is updated by the new mask vector.

$$\hat{\mathbf{m}}_n = \frac{1}{\mathbf{w}_{n-1} + \mathbf{a}_n}(\mathbf{w}_{m-1}\hat{\mathbf{m}}_{n-1} + \mathbf{a}_n\mathbf{x}_n) \tag{22}$$

$$\mathbf{w}_n = \mathbf{w}_{n-1} + \mathbf{a}_n$$

Then the covariance matrix is updated by combining the previous covariance and the current new masked sample. The weighting matrix is updated by the out-product matrix of the new mask vector.

$$\hat{\mathbf{C}}_n = \frac{1}{\mathbf{W}_{n-1} + \mathbf{a}_n\mathbf{a}_n^T}(\mathbf{W}_{v-1}\hat{\mathbf{C}}_{n-1} + \mathbf{a}_n(\mathbf{x}_n - \hat{\mathbf{m}}_n)(\mathbf{x}_n - \hat{\mathbf{m}}_n)^T\mathbf{a}_n^T) \tag{23}$$

$$\mathbf{W}_n = \mathbf{W}_{n-1} + \mathbf{a}_n\mathbf{a}_n^T$$

Finally, by performing eigen-analysis for the estimated covariance matrix $\hat{\mathbf{C}}_n$, we can obtain the new eigenvectors $\hat{\boldsymbol{\varphi}}_n$.

Here we might meet the same problem as the updating of the normal eigenspace (without missing data). That is, for some applications, we are dealing with high dimension training samples and thus the covariance $\hat{\mathbf{C}}_n$ is expensive to compute and update. Due to the fact that we need to perform the element-wise operation in (23), we could not borrow the same idea as in Section 6.1.2, which updates the inner-product matrix, instead of the covariance matrix. Thus in practice, we might keep all training sample $\mathbf{x}_n$ and their corresponding masks $\mathbf{a}_n$.

6.2 Experimental results of updating the mosaic model

Given a set of face images from *L* subjects for training, each subject has one individual mosaic model trained from his/her own training images. When a test image arrives, it is

compared with every individual mosaic model and assigned to the one that gives the minimal similarity measure.

Now we need to decide whether to update the mosaic model of the recognized subject, using the test image. We use the twin-threshold scheme for making the decision. First, by comparing the minimal similarity with a pre-defined threshold, we can see whether the current model can represent the test image well. If it does, we do not perform updating since this test image does not bring enough new statistical information for the mosaic model. Second, we calculate the confidence measure as the difference between the similarity measure of the second candidate and the similarity measure of the top candidate. Then the confidence measure is compared with another pre-defined threshold. If the confidence measure is larger than that threshold, this test image will be utilized to update the assigned mosaic model using our updating method, as described in Section 6.1.3. Basically the larger the confidence measure, the more confidence we have about the current recognition result. Thus as time goes on, the mosaic model will include the most recent statistics of the subject's appearance, and be able to recognize more "new looking" images from that subject.

Notice that there might be chances we would do a wrong updating, i.e., the test image and the model being updated are not from the same subject. The wrong updating corrupts the model, which is very bad for recognition. However, given the fact the most of the recognition performance without updating can achieve 90% recognition rate, the chance of wrong updating is very small.

As we mentioned before, the value of the decaying parameter α_m should be determined according to the statistics of the random process. If the statistics of the random process change fast, we choose a small α_m. If the statistics change slowly, a large α_m may perform better. In [51], we provide an explicit formulation on how to choose α_m dynamically. In the implementation of updating with missing data, we do not incorporate this dynamic scheme.

We have tested our proposed algorithm on the FIA database. We use the same experimental setup as the option 3 of Section 5.4.2. There are 29 subjects in the database, with 10 sequences per subject as the test sequences. Each sequence has 50 frames, and the first frame is labeled with the ground truth data. Only one image per subject is available for training a mosaic model without deviation. During the test stage, if any video frame from the test sequence passes the twin-threshold test, we will use it to update the mosaic model of the recognized subject. Figure 45 shows the mean of one individual mosaic model being updated during the test stage. We show the recognition performance of our proposed updating algorithm comparing with the one without updating. We can see that the mosaic model with updating greatly improves the face recognition from video sequences.

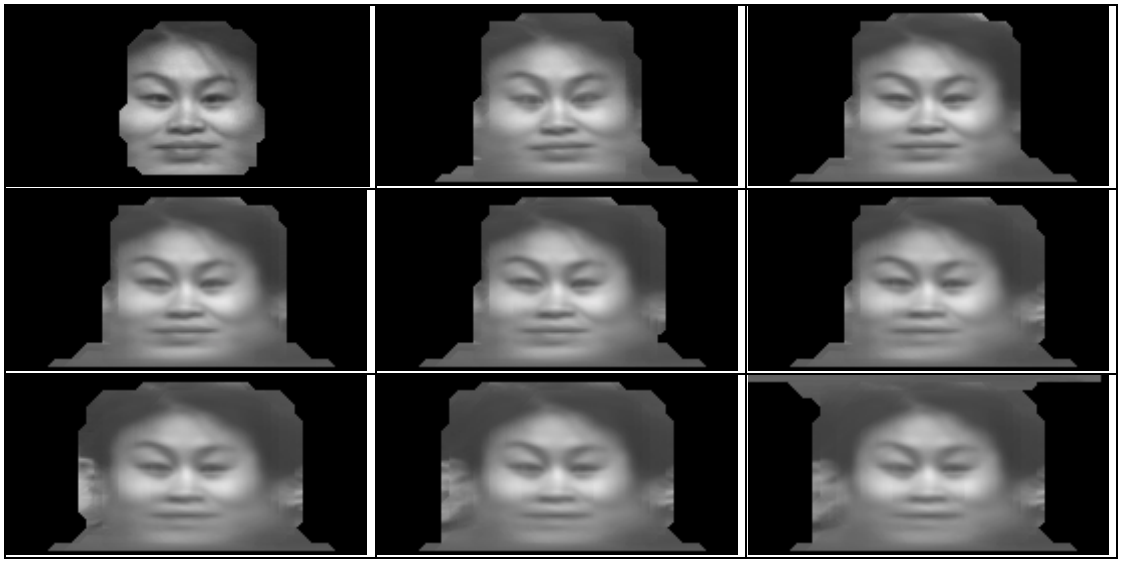

Figure 45 The mean of the mosaic model being updated during the test stage of one subject

Table 3 Recognition performance of updating the mosaic model

Mosaic model without updating	Mosaic model with updating
12.07%	6.90%

6.3 Conclusions

In this chapter we apply the updating-during-recognition scheme to using the face mosaic models on face recognition applications. We first present the theory for updating a normal eigenspace, which can be performed via the covariance matrix or the inner-product matrix. Then we show the similar covariance updating can also be applied in updating an eigenspace with missing data. In the updating-during-recognition scheme, any video frame that satisfies the twin-threshold will be utilized to update the individual mosaic model of the recognized subject.

In the experimental part, we show that by using the updating scheme, the face mosaic model can be enhanced during the test stage, and eventually improve the face recognition from video sequences.

7. Summary and Future Directions

All the work in this thesis is to improve the face recognition performance in dealing with variations. We propose the idea of geometrical mapping in taking care of the pose variation. The resulting texture map from the geometrical mapping is represented as an array of local patches, which enables us to study the discriminative power of local patches according to different patches. Also, we propose to build a statistical mosaic model from multiple training images with the help from a geometric deviation model. Furthermore, we apply the face mosaic model in video-based face tracking and recognition. And also we use the updating-during-recognition scheme to enhance the model and recognition over time.

The contribution of this thesis lies in the following:

The first contribution is to use a 3D ellipsoid model to compensate the pose variation. As the hardest variations to deal with, previously pose variation is either modeled purely from a 2D representation, or via a very sophistical 3D shape model. We attempt to model the pose variation by relying on a simple 3D ellipsoid, which provides the benefit of automatic and efficient modeling comparing to the sophistical 3D shape model, and of better approximation to the true geometry comparing to the 2D representation.

The second contribution is to learn a probabilistic model for comparing similarity measures of corresponding patches. By modeling the intra-subject and inter-subject variations

explicitly, we can study how the discriminative power changes according the pose variation. Also the Bayesian framework can be used to evaluate the similarity value of corresponding patches.

The third contribution is to represent the mosaic model as an array of patches and as a subspace model instead of *one* plane texture map. Traditionally the mosaic is generated by sticking multiple images together to form one larger texture map. We propose to use a statistical subspace for modeling the mosaic, which certainly models much more variations comparing to using only one texture map. Also the patch representation for the texture map and the mosaic model is better than the original texture maps because the low-dimensional feature (patch) is easier to be modeled using the statistical tool than the high-dimensional feature (map).

The fourth contribution is to propose the geometric deviation model in combining multiple images for training a mosaic model. There are two benefits of using this deviation models. First, it alleviates the blurring problem raised by the week geometric assumption of the 3D ellipsoid model. Much clear and informative mosaic model can be obtained by utilizing the deviation model. Second, this deviation model is an indication of how much the true geometry of the human face deviates from the 3D ellipsoid model. For training images from one subject, this deviation model contains the individualized geometric information, which will certainly help the face recognition together with our mosaic model on facial appearances.

The fifth contribution is to utilize the face mosaic model in pose robust video-based face recognition. The traditional problem with video-based face recognition is face registration, i.e., we should not only track the face area, but also register the face before the recognition can take place. Since the face mosaic is a good model for pose variation, we use it in performing face tracking and recognition simultaneously.

The sixth contribution is to build a capturing system for collecting a multi-view face video database. In the face recognition community, there are a number of public available face image databases, such as FERET, ORL, PIE, etc, which have been the great sources for different algorithms to compare face recognition performance. However, there are not much face video

databases, especially the multi-view database. We have built a capturing system on both hardware and software. This system is able to capture six different video sequences of human faces simultaneously and save into the hard disk. Three different views of face videos are captured synchronously.

The final contribution is to apply the updating-during-recognition scheme for video-based face recognition. Due the limited amount of training data, the face model normally does not contain enough statistical information after the training stage. By updating the mosaic model using part of the test data, we can enhance the modeling and recognition over time. This scheme is general in the sense that it can be applied to video-based face recognition based on any updateable models.

There are some interesting extensions for the work described in this thesis:

First, we have mentioned there is one dimension on how much the recognition algorithms are relying on the geometric model. Our approach is in the middle along this dimension. If we move toward the direction of using more geometric information, hopefully the recognition performance is better, while the difficult of model fitting is also getting harder. Now the question is that where would be the best *trade-off* along this dimension, in terms of efficient and automatic recognition.

Second, suppose we only have one front training image for one subject, can we estimate/anticipate what would be the profile view looks like for this subject? I think we can approach this problem by studying the relationship between the patches from the front view and the patches from the profile view. Our face mosaic has already modeled the statistical of the within-patch appearance, but we did not model statistical of the between-patch case.

Third, an interesting problem in pose robust face recognition is that, if we only allow one training image per subject, which pose image should we use for training, in order to have the best recognition performance? Is that purely front view? Or 45 degree in horizontal direction? We wish that the answer to this question is approach-independent.

Fourth, more and more interests are moving toward the 3D face recognition, where both the training and test data are 3D range face images. In this case, we can still apply the mosaicing idea. That is, to register different parts of range images and stick them together to form a larger model. The difference between this and our application is that the geometric deviation model is not needed since we have a perfect 3D geometric model.

Bibliography

[1] Adini, Y., Moses, Y., Ullman, S., Face recognition: the problem of compensating for changes in illumination direction, Pattern Analysis and Machine Intelligence, IEEE Transactions on, Volume: 19, Issue: 7 , July 1997, pp.721 – 732.

[2] P.N. Belhumeur, J.P. Hespanha, D.J. Kriegman, Eiegnfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection, IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol.19, No.7, 1997, pp.711-720.

[3] David J. Beymer, Face Recognition Under Varying Pose, In Proceedings of 2nd Int'l Symposium on Image and Signal Processing and Analysis, 2001.

[4] Zoran Biuk, Sven Loncaric, Face recognition from Multi-Pose Image Sequence, Proceedings of 2nd Int'l Symposium on Image and Signal Processing and Analysis, Pula, Croatia, 2001, pp. 319-324.

[5] Blanz, V., Vetter, T., Face Recognition Based on Fitting a 3D Morphable Model, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 25, No. 9, 2003, pp.1063-1074.

[6] George Borshukov, Measured BRDF in film production: realistic cloth appearance for "The Matrix Reloaded", SIGGRAPH 2003.

[7] Brown, L.M., 3D head tracking using motion adaptive texture-mapping Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on, Volume: 1, 8-14 Dec. 2001, pp. 998 - 1003.

[8] R. Brunelli, T. Poggio, Face Recognition: Features versus Templates. IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol.15, No.10, 1993, pp.1042-1052.

[9] La Cascia, M., Sclaroff, S., Athitsos, V., Fast, reliable head tracking under varying illumination: an approach based on registration of texture-mapped 3D models, Pattern Analysis and Machine Intelligence, IEEE Transactions on, Volume: 22, Issue: 4, April 2000, pp.322 – 336.

[10] R. Chellappa, C.L. Wilson, S. Sirohey, Human and machine recognition of faces: a survey, Proceedings of the IEEE, Vol.83, No.5, 1995, pp.705–741.

[11] Chen D., State A., Banks D., Interactive shape metamorphosis. In 1995 symposium on interactive 3D graphics, ACM SIGGRAPH, 1995, pp. 43-44.

[12] L.F. Chen, H.Y.M. Liao, M.T. Ko, J.C. Lin, G.J. Yu, A new LDA-based face recognitions system which can solve the small sample size problem, Pattern Recognition, Vol. 33, 2000, pp.1713-1726.

[13] T. P.-C. Chen, T. Chen, Second-Generation Error Concealment for Video Transport over Error Prone Channels, Wireless Communications and Mobile Computing, Special Issue on Multimedia over Mobile IP, October 2002.

[14] A. Roy Chowdhury, R. Chellappa, R. Krishnamurthy, T.Vo, 3D Face Recostruction from Video Using A Generic Model, Int. Conf. on Multimedia and Expo, 2002.

[15] Ki-Chung Chung, Seok Cheol Kee, Sang Ryong Kim, Face recognition using principal component analysis of Gabor filter responses, Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems, 1999. Proceedings. International Workshop on, 26-27 Sept. 1999, pp.53 – 57.

[16] T.F.Cootes, G.J.Edwards, C.J.Taylor, Active Appearance Models, ECCV'98.

[17] R.O. Duda, P.E. Hart, D.G. Stork, Pattern Classification, Second edition. John Wiley & Sons. Inc., New York, 2001.

[18] G. J. Edwards, C.J. Taylor, T.F. Cootes, Improving Identification Performance by Integrating Evidence from Sequences, In Proc. Of 1999 IEEE Conference on Computer Vision and Pattern Recognition, June 23-25, 1999 Fort Collins, Colorado, pp.486-491.

[19] G.J.Edwards, C.J.Taylor, T.F.Cootes, Learning to Identify and Track Faces in Image Sequences, Int. Conf. on Face and Gesture Recognition 1998, pp.260-265.

[20] O. D. Faugeras, Three-Dimensional Computer Vision, MIT Press, Cambridge, 1993.

[21] C.L. Fennema, W.B. Thompson, Velocity determination in scenes containing several moving objects. Computer Graphics and Image Processing, Vol. 9, 1979, pp.301-315.

[22] T. Fromherz, P. Stucki, M. Bichsel, A Survey of Face Recognition, MML Technical Report, No 97.01, Dept. of Computer Science, University of Zurich, Zurich, 1997.

[23] R. Gross, I. Matthews, S. Baker, Appearance-Based Face Recognition and Light-Fields, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 26, No. 4, April, 2004, pp. 449 - 465.

[24] R. Gross and J. Shi, The CMU Motion of Body (MoBo) Database, tech. report CMU-RI-TR-01-18, Robotics Institute, Carnegie Mellon University, June, 2001.

[25] R. Gross, J. Shi, J. Cohn, Quo Vadis Face Recognition, Third Workshop on Empirical Evaluation Methods in Computer Vision, 2001.

[26] R.A. Horn, C.R. Johnson, Matrix Analysis. Cambridge University Press 1985.

[27] F. J. Huang, T. Chen, Tracking of Multiple Faces for Human-Computer Interfaces and Virtual Environments. IEEE Intl. Conf. on Multimedia and Expo, New York, July 2000.

[28] Introduction to Monte Carlo Methods, Copyright (C) 1991, 1992, 1993, 1994, 1995 by the Computational Science Education Project.

[29] Michael Isard, Andrew Blake, CONDENSATION -- conditional density propagation for visual tracking, Int. J. Computer Vision, 29, 1, 1998, pp.5-28.

[30] Michael Isard, Andrew Blake, Active Contours, Springer-Verlag, 1998.

[31] Kanade T., Picture processing system by computer complex and recognition of human faces. Doctoral dissertation, Department of Information Science, Kyoto University, November, 1973.

[32] Kanade, T., Yamada, A., 2003. Multi-subregion Based Probabilistic Approach toward Pose-invariant Face Recognition. Proceedings of 2003 IEEE International Symposium on Computational Intelligence in Robotics Automation, Kobe, Japan, Vol. 2, pp.954-959.

[33] J Kittler, M Hatef, R P W Duin and J Matas, On combining classifiers, IEEE Transactions on Pattern Analysis and Machine Intelligence, 1998 Mar, Vol. 20, pp. 226-239.

[34] P. Kruizinga, N. Petkov, Optical flow applied to person identification. Proceeding of the 1994 EUROSIM Conference on Massively Parallel Processing Applications and Development, Delft, The Netherlands, 21-23 June 1994, Elsevier, Amsterdam, pp.871-878.

[35] Kwang In Kim, Keechul Jung, Hang Joon Kim, Face recognition using kernel principal component analysis, Signal Processing Letters, IEEE, Volume: 9, Issue: 2 , Feb. 2002, pp.40 – 42.

[36] A. K. Jain, Fundamentals of Digital Image Processing, Prentice Hall, Englewood Cliffs, NJ, 1989.

[37] A. K. Jain, A. Ross, Fingerprint Mosaicking, Proc. of ICASSP, Orlando, Florida, May 13-17, 2002.

[38] Jones, M.J., Viola, P., Fast Multi-view Face Detection, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2003.

[39] T. Sim, T. Kanade, Combining Models and Exemplars for Face Recognition: An Illuminating Example, Proceedings of the CVPR 2001 Workshop on Models versus Exemplars in Computer Vision, December 2001.

[40] M. Lades, J.C. Vorbruggen, J. Buhmann, J. Lange, C. von der Malsburg, R.P. Wurtz, and W. Konen, Distortion Invariant Object Recognition in the Dynamic Link Architecture, IEEE Transactions on Computers, Vol.42, No.3, 1992, pp.300-311.

[41] Lanitis, A., Taylor, C.J., Cootes, T.F., Toward automatic simulation of aging effects on face images, Pattern Analysis and Machine Intelligence, IEEE Transactions on, Volume: 24, Issue: 4, April 2002, pp.442 – 455.

[42] Lee, Y., Terzopoulos, D., Waters, K. Realistic Modeling for Facial Animation, in SIGGRAPH 95 conference proceedings, ACM SIGGRAPH, 1995, pp. 55-62.

[43] Hyung-Soo Lee, Daijin Kim, Pose Invariant Face Recognition Using Linear Pose Transformation in Feature Space, ECCV 2004 Workshop on Computer Vision in Human-Computer Interaction, Prague, Czech Republic, May 16, 2004.

[44] Yongmin Li, Dynamic face models: construction and applications, PhD Thesis, Queen Mary, University of London, 2001.

[45] Yongmin Li, Shaogang Gong and Heather Liddell, Recognizing the Dynamics of Faces Across Multiple Views, In Proc. British Machine Vision Conference (BMVC2000), Bristol, England, September 2000, pp.242-251.

[46] J.J. Lien, A. Zlochower, J.F. Cohn, T. Kanade, Automated Facial Expression Recognition. Proceedings of the Third IEEE International Conference on Automatic Face and Gesture Recognition. Nara, Japan. April 1998, pp. 390-395.

[47] Chengjun Liu, Gabor-based kernel PCA with fractional power polynomial models for face recognition, Pattern Analysis and Machine Intelligence, IEEE Transactions on , Volume: 26 , Issue: 5 , May 2004, pp.572 – 581.

[48] Qingshan Liu, Rui Huang, Hanqing Lu, Songde Ma, Face recognition using Kernel-based Fisher Discriminant Analysis, Automatic Face and Gesture Recognition, 2002. Proceedings. Fifth IEEE International Conference on, 20-21 May 2002, pp.205 – 211.

[49] X. Liu, T. Chen, CMU Face In Action (FIA) Database Collection, Carnegie Mellon Technical Report AMP 04-01.

[50] X. Liu, T. Chen, B.V.K. Vijaya Kumar, Face Authentication for Multiple Subjects Using Eigenflow, Pattern Recognition, Volume 36, Issue 2, February 2003, pp.313-328.

[51] X. Liu, T. Chen, S. M. Thornton, Eigenspace Updating for Non-Stationary Process and Its Application to Face Recognition. Pattern Recognition, special issue on Kernel and Subspace Methods for Computer Vision, Volume 36, Issue 9, September 2003, pp.1945-1959.

[52] X. Liu, T. Chen, Video-Based Face Recognition Using Adaptive Hidden Markov Models, In the Proceeding of the IEEE International Conference on Computer Vision and Pattern Recognition 2003, Madison, Wisconsin, June 16-22, 2003.

[53] X. Liu, T. Chen, Geometry-assisted Statistical Modeling for Face Mosaicing, Proceeding of the IEEE International Conference on Image Processing 2003, Vol.2, Barcelona, Spain, 2003, pp.883-886.

[54] S. Lucey, T. Chen, A GMM Parts Based Face Representation for Improved Verification through Relevance Adaptation, In Proc. Of IEEE Conference on Computer Vision and Pattern Recognition 2004, 27th June - 2nd July, 2004, Washington, DC.

[55] Martinez, A.M., Recognizing expression variant faces from a single sample image per class, Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on, Volume: 1, 18-20 June 2003, pp.353-358.

[56] I. Matthews, S. Baker, Active Appearance Models Revisited, tech. report CMU-RI-TR-03-02, Robotics Institute, Carnegie Mellon University, April, 2003.

[57] K Messer, J Matas, J Kittler, J Luettin, G maitre. Xm2vtsdb: The extended m2vts database. In Second International Conference of Audio and Video-based Biometric Person Authentication, March 1999.

[58] K. Okada, C. von der Malsburg, Pose-Invariant Face Recognition with Parametric Linear Subspaces, In proceedings of the Fifth International Conference on Automatic Face and Gesture Recognition, pp. 64-69, Washington DC, May 2002.

[59] Alex Pentland, Baback Moghaddam, Thad Starner, View-Based and Modular Eigenspaces for Face Recognition, Proc. of IEEE Conf. on Computer Vision and Pattern Recognition 1994.

[60] P. J. Phillips, P. J. Rauss, S. Z. Der, FERET (Face Recognition Technology) Recognition Algorithm Development and Test Results, October 1996. Army Research Lab technical report.

[61] P.J. Phillips, P. Grother, R.J Micheals, D.M. Blackburn, E Tabassi, and J.M. Bone. FRVT 2002: Evaluation Report, March 2003. (http://www.frvt.org/FRVT2002/documents.htm)

[62] Point Grey Research Inc. http://www.ptgrey.com/products/index.html

[63] W.H. Press et al., Numerical Recipes in C: The Art of Scientific Computing, 2nd edition, Cambridge Univeristy Press, Cambridge, England, 1992.

[64] Ferdinando Samaria, Andy Harter. Parameterisation of a Stochastic Model for Human Face Identification. Proceedings of 2nd IEEE Workshop on Applications of Computer Vision, Sarasota FL, December 1994.

[65] C. Schmid, R. Mohr, C. Bauckhage, Comparing and Evaluating Interest Points, IEEE Proc. Int'l Conf. Computer Vision, 1998, pp. 230-235.

[66] H. Schneiderman and T. Kanade, Object Detection Using the Statistics of Parts, International Journal of Computer Vision, 2002.

[67] T. Sim, S. Baker, M. Bsat, The CMU Pose, Illumination, and Expression Database, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 25, No. 12, December 2003, pp.1615 - 1618.

[68] Szeliski R., Video Mosaics for Virtual Environments, IEEE Computer Graphics and Applications, Vol.16, No.2, March 1996, pp.22-30.

[69] Yingli Tian, T. Kanade and J. F. Cohn, Recognizing Action Units for Facial Expression Analysis, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 23, No. 2, February, 2001.

[70] A.J.O.' Toole, D.A. Roark, H. Abdi, Recognizing moving faces: A psychological and neural synthesis, Trends in Cognitive Science, 6(6), 2002, pp.261-266.

[71] De la Torre, F., Black, M. J., Robust principal component analysis for computer vision, Int. Conf. on Computer Vision, ICCV-2001, Vancouver, BC, Vol. I, pp. 362-369.

[72] M. Turk, A. Pentland, Eigenfaces for Recognition, Journal of Cognitive Neuroscience, Vol.3, No.1, 1991, pp.71-86.

[73] George Wolberg, Digital image warping, published by IEEE Computer Society Press, 1990.

[74] Y. Yacoob, L. Davis, Smiling faces are better for face recognition Automatic Face and Gesture Recognition, 2002. Proceedings. Fifth IEEE International Conference on, 20-21 May 2002, pp.52 – 57.

[75] Y. Yacoob, L.S. Davis, Recognizing Human Facial Expressions From Long Image Sequences Using Optical Flow. IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol. 18, No. 6, 1996, pp.632-646.

[76] Ming-Hsuan Yang, David Kriegman, Narendra Ahuja, Detecting Faces in Images: A Survey, IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), vol. 24, no. 1, pp. 34-58, 2002.

[77] H. Yu and J. Yang, A Direct LDA Algorithm for High-dimensional Data - with Application to Face Recognition, Pattern Recognition, Vol. 34, 2001, pp.2067-2070.

[78] De Vel, O., Aeberhard, S., Line-based face recognition under varying pose, Pattern Analysis and Machine Intelligence, IEEE Transactions on, Volume: 21, Issue: 10, Oct. 1999.

[79] Wen-Yi Zhao, Rama Chellappa, P.J. Jonathon Phillips, Azriel Rosenfeld, Face Recognition: A Literature Survey, ACM Computing Survey, Vol. 35, No 4, 2003, pp.399-458.

[80] Ming Zhang, Fulcher, J., Face recognition using artificial neural network group-based adaptive tolerance (GAT) trees, Neural Networks, IEEE Transactions on, Volume: 7, Issue: 3, May 1996, pp.555 – 567.

[81] Z. Zhang, Z. Liu, D. Adler, M. F. Cohen, E. Hanson, Y. Shan, Robust and Rapid Generation of Animated Faces from Video Images: A Model-Based Modeling Approach, International Journal of Computer Vision, Vol.58, No.1, pages 93-119, June 2004.

[82] S. Zhou, V. Krueger, R. Chellappa, Face Recognition from Video: A CONDENSATION Approach, In Proc. of Fifth IEEE International Conference on Automatic Face and Gesture Recognition, Washington D.C., May 20-21, 2002, pp.221-228.

[83] S. Zhou, V. Krueger, R. Chellappa. Probabilistic recognition of human faces from video. Computer Vision and Image Understanding (CVIU) (special issue on Face Recognition), Vol. 91, 2003, pp.214-245.