FACE VERIFICATION
USING HIGH-DIMENSIONAL FEATURE

A Thesis presented to

the Faculty of the Graduate School

at the University of Missouri

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

---

by

Kai Huang

Dr. Tony X. Han, Thesis Supervisor

JULY 2014

The undersigned, appointed by the Dean of the Graduate School, have examined the dissertation entitled:

FACE VERIFICATION
USING HIGH DIMENSIONAL FEATURE

presented by Kai Huang,
a candidate for the degree of Master of Science and hereby certify that, in their opinion, it is worthy of acceptance.

Dr. Tony X. Han

Dr. Zhihai He

Dr. Yunxin Zhao

ACKNOWLEDGMENTS

Thank you to my advisor Prof. Tony X. Han for the continuous mentoring my study and research. Expecially he opened a door for me to enjoy the beautiful of the computer vision area.

Thank you to my committee members: Prof Zhihai He, Prof Yunxin Zhao. I learned a lot from you about the research attitude and give me the opportunity to present my research work.

Thank you to the senior students in our lab: Guobin Chen, Guang Chen, Miao Sun give me a lot of guide during my research.

Thank you to my friends I met in Mizzou, especially May Luo, Kevin Wang, Zong Supper, Chu Wu, Lei Guo, their supports I will keep them in mind.

Thank you to my friends in Donghua University, especially Yizhi Chen, Qiqi Shen, Jun Xue, Jie Zheng and Guifang Tang, they have been stayed with me when I feel alone.

Thank you for my parents Lifen Wu, Xianbin Huang and Changtian Zhang for their everlasting love.

Thank you to my girl friend Carrie Wu. She has been spiritually supporting my study for two years!

TABLE OF CONTENTS

LIST OF FIGURES

Figure Page

ABSTRACT

A face verification problem is very popular in computer vision area. In this thesis, we developed a face verification demo using high-dimensional feature. We first used Adaboost Cascade Classifier to detect face then using facial points detector get the points which we want to build the high-dimensional based on them. To the face verification problem, we used a "smart" algorithm Bayesian Face Revisited. Finally, we apply the face verification model into a face recognition without data training.

Chapter 1

Introduction

Face verification is very important application in computer vision. Different from face recognition, face verification focus on the match and mismatch problem. Moreover, the people for testing will not be appeared in the training part in face verification problem. In 2007, Gray B. Huang build a face verification dataset Laleled Faces in the Wild(LFW)[1] which collected face images from the Internet and most of face images in this dataset have different expressions and illuminations. And most popular face verification methods has been tested in this dataset.

A very famous method face recognition using eigenfaces came up by Matthew A. Turk and Alex P. Pentland[2]. However, before a reliable facial landmarks detector developed, people seldom use face verification based on facial points. With some reliable facial landmarks detector developing, for example, Active Shape Model[7] and facial landmarks learned by SVM [8], people began focused on face verification using facial points.

Based on the reliable facial landmarks and Bayesian face recognition[9]. Dong Chen use a Bayesian Face Revisited to get state-of-art performance in LFW dataset[10]. Further more, he pointed out the higher dimension can get a better result [11].

To achieve real-time face verification with sliding windows. We apply a face

detection [3] and facial landmarks detector [7]. After this process, we using Local Binary Feature and facial points to generate high-dimensional feature and finally apply them into Joint Bayesian Formulation. The process as showed in Figure 1.1.

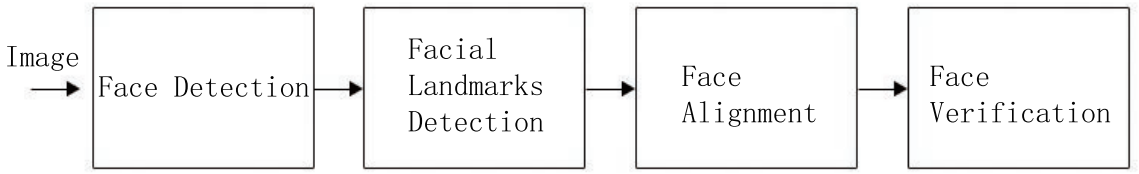

Figure 1.1: Face Verification System

Chapter 2

Face Detection

In the first step for a face verification system, we should build a face detection algorithm. Face detection is concerned with finding whether or not there are any faces in a given image. In this work, we used a famous Robust Real-time Object Detection by Paul Viola and Michael Jones[3]. We use a kind of feature called Haar feature to separate the face and non-face images. Moreover, in the purpose of speeding up the speed of extracting features, we use integral image to compute rectangle features. In the classification part, to get a faster classification speed, we use a cascade adaboost classifier.

2.1 Introduction

One of the main challenges of computer vision is detecting and classifying objects in an image efficiently. However, the main point for detection is to reduced the momory needed for machines.

An efficient face detection system presented by Viola and Jones in 2001[3]. They adopted Cascade AdaBoost to select a set of features and train a classifier. The

detector reduces the number of features by adopting a cascade structure.

The remainder of the work is organized as follows. Section 2.2 gives some related work. Section 2.3 describes the Haar feature. Section 2.4 discusses the implementation of cascade Adaboost for face detection.

2.2 Related Work

In 2001, Paul Viola and Micheal J.Jones[3] developed a window scanning method and Adaboost to detect object in real time. The most critical problem for object detection is increasing the speed. They implemented a face detection process at 15 frames per second.

Based on the way of window-Adaboost, Stan Z. Li further developed the detection system to ignore the influence of image rotation directions[4]. There are more examples of visual detectors using Adaboost.

Also, people use another way to detect face other than Haar-like feauture. In [5] Shin showed a more reliable way to develop face detection based on skin color. Qsuna used support vector machines to do the face detection [6].

2.3 Image Feature

In our work, we used a kind of simple feature to build weak classifiers. From [3], Viola and Jones came up with using features called Haar-like feature rather than using pixels directly. On the other hand, a feautre-based detector is much faster than a pixel-based detector.

There are three kinds of Haar-like features. The value of a two-rectangle feauture is the difference between the sum of pixels within two retangular regions. As seen in Figure 2.1, it can be built as horizontally and vertically adjacent. A three-rectangle

feature computes the difference of the sum within two outside rectangular regions and the sum in middle region. Finally, a four-rectangle feature computes the difference of two pairs of diagonal regions.

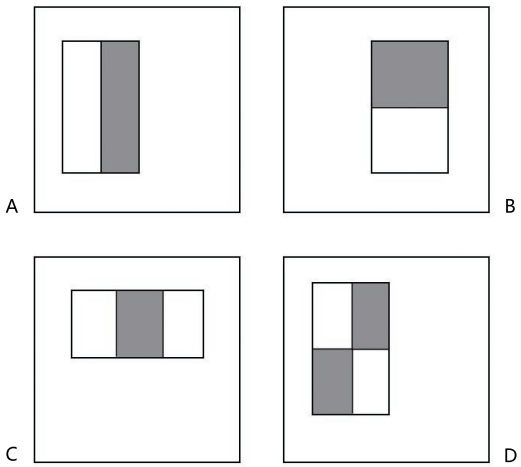

Figure 2.1: Haar-like feature for face detection. The feature extracted by the difference between sum of the pixels in white rectangle and in gray rectangle. A and B show the two-rectangle feature, C is the example of three-rectangle feature and D belongs to four-rectangle feature.

2.3.1 Integral Image

Rectangle features can be rapidly computed using an intermediate representation for the image called integral image. The value of integral image at location x, y is the sum of pixels in the region of the above and the left.

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y') \tag{2.1}$$

Where ii is the integral image and i is the pixel in the original gray image. Also, we can use the following equation to get the integral image:

$$s(x, y) = s(x, y - 1) + i(x, y) \tag{2.2}$$

$$ii(x, y) = ii(x - 1, y) + s(x, y) \tag{2.3}$$

Where $s(x, y)$ is the cumulative sum of row pixel. The above this integral image can be computed by one pass over the image.

After we get the integral image, we can get the sum of a certain region as shown in the figure 2.2.

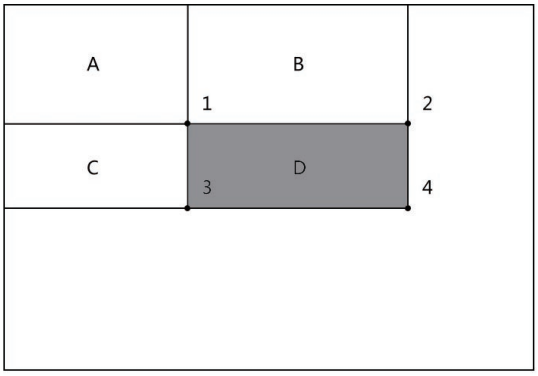

Figure 2.2: Integral Image.

In Figure 2.2, $1, 2, 3, 4$ means the value of integral image in current location. The value of region D can be computed as $(4 + 1) - (2 + 3)$. Therefore, the size of the retangle region has no relationship with the computation time using integral image.

2.4 Cascade Adaboost Classifier

Considering the huge number of Haar features that will be generated from a image, we should build a classifier which can choose the most particular feature that seperates positive and negative samples efficiently. Moreover, with the large number of features, even one of the features responses are very simple to compute, to apply all of the Haar feature would be too expensive both in memory and time. Therefore, we use Adaboost classifier to get a set of features which represents all possible faces. Also, if we wanted to speed up the time of classification, we would build a cascade adaboost classifier which can eliminate negative samples faster.

2.4.1 Learning with Adaboost

Weak Learners. The face detection system uses weak learners constrained to evaluating a feature. For each trial, one weak classifier is a feature evaluation followed by an optimal thresholding. This threshold means the number of misclassification samples is minimum. A weak classifier consists of a feautre f_i and a threshold Θ_i:

$$h = \begin{cases} 1 & f_i(x) < \Theta_i \\ 0 & \text{otherwise.} \end{cases} \tag{2.4}$$

Adaboost training. Adaboost is used to find the best weak learners and their corresponding weights. The boosting algorithm maximizes the margin between positive and negative samples. The algorithm first gives all features from both positive and negative examples and their labels $\{1, -1\}$. At the second step, it initializes the corresponding weights which depends on the number of positive and negative samples.

The boosting algorithm runs a series of trials T, each trial selects a new best weak learner. At the first round, it normalized the weights sum to 1. Next, the algorithm get the best learners with the minimum number of misclassifications number with

respect to the weight vector. After the best weak learner has been selected, the algorithm update the weight vectors and run the next trial.

Algorithm 1 Adaboost algotithm

• Given example images (x_1, y_1),(x_2, y_2),...,(x_n, y_n) where y is the label for positive and negative examples.

• Initialize weights $w_i = 1/2m$,$1/2l$ for different label, where m and l are the number of negatives and positives respectively.

for $t = 1$,...,T: **do**

1. Normalizie the weights

$$w_{t,i} \leftarrow \frac{w_{t,i}}{\Sigma_{j=1}^{n} w_{t,j}}$$

so that w_t is a probability distribution.

2. For each feautre, j, train a classifier h_j which using a single feautre. The error for each single feature classifier can be represented as: $\epsilon_j = \Sigma_i w_i |h_j(x_i) - y_i|$.
3. Choose a classifier which has the lowest error.
4. Updata the weight

$$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$$

where $e_i = 0$ is the example x_i classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$

end for

• The final strong classifier is

$$h = \begin{cases} 1 & \Sigma_{t=1}^{T} \alpha_t h_t(x) \geqslant \frac{1}{2} \Sigma_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise.} \end{cases}$$

where $\alpha_t = log \frac{1}{\beta_t}$.

Combined with normalization, this update leads to most of the weight being placed on hard to classifiy examples. Therefore, with the number of trials increasing, the error rate also increases. This results in smaller α values for weak learners selected later in the training process.

2.4.2 Cascade Adaboost

Scanning window scans all possible sub-windows in an image. Evaluating all sub-windows becomes intractable when the classifier classify all sub-windows' feauture. As Paul Viola and Michael J. Jones pointed out[3], the first two features selected by

adaboost can detect almost all faces with 40% false positive rate. Based on this, we build a "cascade" classifier.

The cascade adaboost classifier is construted such that the first layer evaluates a small number of features and the next layer adds more and more features. The threshold of each layer must satisfy the detect rate and false positive rate given before training. Once an example is rejected by one layer, it will not be output to next the classifier. The initial layer eliminates a large number of negative examples with very little processing. Subsequent layers eliminate additional negatives but require additional computation. A well constructed decision tree significantly reduces the number of features evaluated for each sub-window while maintaining accuracy close to the exhaustive approach. The structure of the cascade is shown in Figure 2.3.

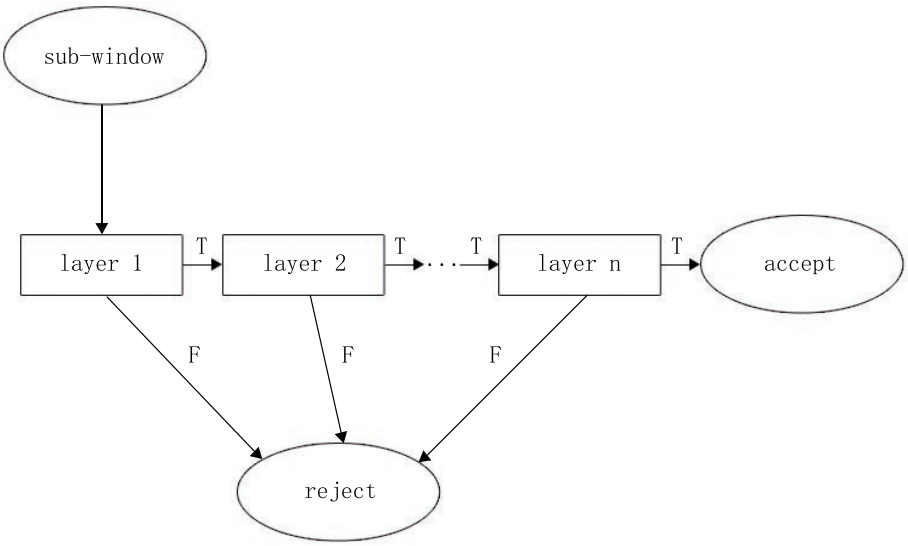

Figure 2.3: Cascade Adaboost Classifier

Chapter 3

Facial Landmarks Detection

3.1 Introduction

The detection of facial landmarks like nose, mouth coners, eyes is an essential part of face recognition or verification system. The accuracy of location for the detection significantly influences the result of face verification.

In this part of thesis, I will introduce the method we will in the facial landmarks detection which developed by Stephen Milborrow[7].

The remainder of the work is organized as follows. Section 3.2 gives some related work. Section 3.3 describes the active shape model. Section 3.4 discussed the implementation of feature descriptors.

3.2 Related work

To high dimensional feature face verification, the detection of facial landmarks is an essential part. The problem of detection facial landmark is largely considered to be a scientific problem. In 2012, M. Uricar, V. Franc develop an facial landmarks detection

using structured Output SVM[8]. In 2008, Stephen Milborrow develop an extended active shape model to locate facial feature.

In 2014, Stephen Milborrow further combined locate feature descriptor SIFT[12], Multivariate Adaptive Regression[13] and Active Shape Model to detect facial points which get a better result and faster speed.

3.3 Active shape model

3.3.1 Shape

Suppose we have n landmark points, $\{x_i, y_i\}$, for a 2-D image, we can form a vector X with $2n$ element vector.

$$X = (x_i, ..., x_n, y_1, ..., y_n)^T \tag{3.1}$$

These landmarks points are related to each other in some invariant sense. No mater how to rotate, expand or move it, it will be the same shape. The common way to measure the distance of two points is the euclidean distance.

A shape can be aligned to another shape by appling a linear transform which has the minimum distance between the shapes. From our purpose, we want this transform should be rotating, scaling and linear translation. Based on this, we use the follow transform:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x_{translate} \\ y_{translate} \end{pmatrix} + \begin{pmatrix} s * cos\Theta & s * sin\Theta \\ -s * sin\Theta & s * cos\Theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \tag{3.2}$$

Cootes and Taylor[14] gives methods to align two shape by using a least-squares procedure. Accurate alignment could be deemed more important for certain points

than others.

To a set of shapes can be aligned using a iterative algorithm as shown in algorithm 2.

Algorithm 2 Shape alignment

Require: Input set of unaligned shapes
1. Choose a reference shape
2. Translate each shape so that it is centered on the origin.
3. Scale the reference shape to the unite size.

repeat
Align all shapes to the mean shape
Recalculate the mean shape from the aligned shapes
Constrain the current mean shape
until
convergence

Ensure: output set of aligned shapes and mean shape

3.3.2 The Active Shape Model

The ASM starts the search for landmarks from the mean shape aligned from the position and size. The task of ASM is to convert the shape which given from the profile models to an allowable face shape. The shape model includes the mean shape $\overline{x}$, selected eigenvector ϕ of covariance matrix S_s and parameter vector b. Where covaraince matrix can be represented as:

$$S_s = \frac{1}{n_{shapes} - 1} \Sigma_{i=1}^{n_{shapes}} (x_i - \overline{x})(x_i - \overline{x})^T \tag{3.3}$$

And one shape vector define as:

$$\hat{x} = \overline{x} + \phi b \tag{3.4}$$

We generate different shapes with Equation 3.4 by modifying the vector parameter b. Given a suggested shape x, we can calculate the parameter b that allows Equation 3.4 best fit x with model shape $\hat{x}$. Cootes and Taylor[14] describe an iterative algorithm

that gives the b and T that minimizs

$$distance(x, T(\overline{x} + \phi b)) \tag{3.5}$$

Where T is the similarity transform that maps the model into the image.

3.4 SIFT Descriptors

In the ASM context, we can use a gradient descriptor so that each element of an array can represent the gradient at the corresponding pixel in the patch. We choose SIFT [15] descriptors because their well known good sensitivity and specificity properties.

The SIFT descriptors takes the form of an array of histograms. The feature generated from the patch we get around the landmark points. To ASM application, we use 15×15 patch and 4×5 histogram array. In common use, we separate 360 degree to 8 bins, that is, 45 degree per bin. The array of histograms is stored internally as a vector with $4 \times 5 \times 8 = 160$ elements.

To avoid the abrupt jump from one histogram bin to another caused by the small change in the orientation, we do the interpolation of orientations across histogram bins. For example, a gradient with a 45° orientation is shared both $0 - 45^\circ$ and $45^\circ - 90^\circ$ bins.

At the last step, we take the square root of each element and after that normalize the resulting vector to unit length. The effect of extreme can be reduced by these feature processes.

3.5 Descriptor Matching

Once we have the local feature descriptor, the next step is to build a feature matching measure. The most simple way is take the Euclidean distance detween two descriptors. However, it will lost the affect of some important pixel because this method gives every histogram bin weight equally. Also, we can take Mahalanobis and SVM to do the descriptor matching but both of them are slow.

From [7], the author give a traditional way to adopt Multivariate Adaptive Regression Spline(MARS) model can give ASM result almost as good as SVM and is mush faster.

3.5.1 Multivariate Adaptve Regression Spline

Multivariate adaptive regression splines(MARS) is a form of regression analysis introduce by Jerome H.Friedman[13]. It is a non-parametric regression technique automatically models non-linearities and interaction between variables.

To further explain the relationship between MARS and feature descriptors, we give an example of implement the descriptor match at the buttom left eyelid in the full scale image as follows:

$$\begin{aligned} match = 0.026 &+ 0.095max(0, 1.514 - b_5) \\ &+ 0.111max(0, 2.092 - b_{10}) \\ &+ 0.258max(0, b_{12} - 1.255) \\ &- 0.108max(0, 1.574 - b_{13}) \dots \end{aligned} \tag{3.6}$$

Where b_i means the ith bin from feature descriptor. The MARS model generate from the training data. And each facial point has the similar formula.

However, we just use some certain bins in the MARS formula. From Equation 6, we only use 17 bins information. Once the bin in the formular, we can ignore the

other bins because it involves little few information. Usually the final formula is quite short and the compute time is quicker than other descriptor matching such as SVM.

Chapter 4

High Dimensional Face Verification

In this work, we implemented a face verification application. Our face verification algorithm employs a high dimensional LBP feature and a joint bayesian formulation as classifier. We establish a benchmark dataset for face verification, named Labeled of Wild(LFW) dataset covers 13,233 images and 5749 different individuals. The face verification achieves good results on LFW dataset.

4.1 Introduction

Face verification is a sub-problem of face recognition, and face verification has been a hot topic in both the supervised and unsupervised field.

The remainder of the work is organized as follows. Section 4.2 details our face verification implementation. The face verification result for LFW dataset details are presented in Section 4.3.

4.2 Face Verification

4.2.1 High dimension feature

We built high dimension feature simply by extracting a small patch around dense facial landmarks in different image pyrimid sizes. Because our feature is based on the accurate and dense facial landmark, in this part of the experiment, we only focus on the high-dimensional feauture extracted by *accurate* facial landmarks. To avoid the wrong facial landmarks, we use the position of landmarks in LFW dataset offered by a recent face alignment method explicit shape regression [16].

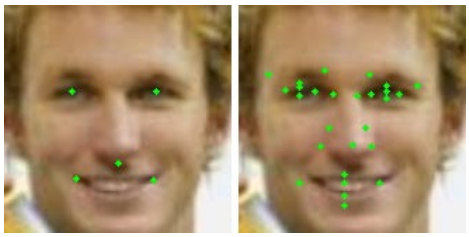

Figure 4.1: Dense facial landmarks

Multiple scales. After we have the position of facial landmarks, we use image pyramid to build a higher dimensional feature. As shown in Figure 4.1, we construct feature in three image pyramid. At each landmark we crop a fixed size patch around landmark and divied the patch into a 4x4 cell. Finally we using LBP feature descripter to represent the feature.

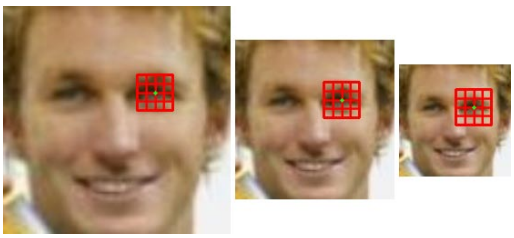

Figure 4.2: Image pyrimid for multiple scales

4.2.2 Local Binary Pattern

We use Local Binary Pattern(LBP) as a kind of feature that can be applied to computer vision research. LBP has applied to many applications in texture classification and segmentation, surface inspection and image retrieval, expecially for face recognition[17].

For the original LBP operator, we first get the patch in a fixed sized cell(e.g 40*40 pixel for each cell), and in each cell, label the image pixel by thresholding, the 3*3 neighborhood of each pixel with the center value. That is, compare the center value with each of its 8 neighborhoods, if the neighborhood is greater than center, set the binary value to 1, otherwise set to 0. Please see Fig 4.3 for illustration.

After calculating each pixel in the cell, the histogram of these 256 different labels can be used as a texture descriptor, however, instead of doing these, we use an extension of the original operator, so called uniform patterns, which reduce the length of the feature vector. In uniform LBP, only at most two bitwise transitions from 0 to 1 are allowed and 58 8-bit binary numbers satisfy this condition. So we separate this into 58 bins. If the descriptor has more than two bitwise transitions, we attribute it to a single bin.

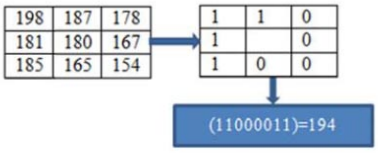

Figure 4.3: LBP calculation sample

4.2.3 Joint Bayesian Formulation

According to the main concept to do face verification [10], a face of a human consists of two factors μ and ξ . In Equation 4.1 μ stands for identity and ξ stands for intra-

personal features. Though these are two independent Gaussian variables, a human face would be verified.

$$x = \mu + \xi \tag{4.1}$$

Here, we can set the latent variable μ and ξ to follow two Gaussian distribution $N(0, S_\mu)$ and $N(0, S_\xi)$. Where μ represents its identity, ξ is the face variation(e.g. lightings, poses and expression) within the same identity. From Equation 4.1 only covariance matrix is needed to be learned by the EM-like algorithm. Representation and associated assumption above could be regarded as a prior face possibility.

Joint formulation with prior [10]. Based on the above knowledge, the joint distribution of x_1, x_2 is also Gaussian, with zero mean no matter under which hypothesis [10]. According to the linear from Equation 4.1 and the independent assumption between μ and ξ, the covariance of two faces is:

$$cov(x_i, x_j) = cov(\mu_i, \mu_j) + cov(\xi_i, \xi_j), i, j \in \{1, 2\} \tag{4.2}$$

H_I condition means that a pair of sample is from the same people, the identity μ_1,μ_2 of the pair is the same and intra-person variation ξ_1,ξ_2 is independent. In Equation 4.2, to the same pair, its covariance matrix can be represented as:

$$\Sigma_I = \begin{bmatrix} S_\mu + S_\xi & S_\mu \\ S_\mu & S_\mu + S_\xi \end{bmatrix} \tag{4.3}$$

Within H_E condition that means a pair of sample is from different people. Obviously identity and intra-personal feature are all independent under this assumption. Hence, the covariance matrix of the distribution $P(x_1, x_2|H_E)$ is:

$$\Sigma_E = \begin{bmatrix} S_\mu + S_\xi & 0 \\ 0 & S_\mu + S_\xi \end{bmatrix} \tag{4.4}$$

According to above two conditional joint probabilities, the log likelihood ratio $r(x_1, x_2)$ could be derived as:

$$r(x_1, x_2) = log \frac{P(x_1, x_2|H_I)}{P(x_1, x_2|H_E)} = x_1^T A x_1 + x_2^T A x_2 - 2x_1^T G x_2 \tag{4.5}$$

Where

$$A = (S_\mu + S_\xi)^{-1} - (F + G) \tag{4.6}$$

$$\begin{pmatrix} F+G & G \\ G & F_G \end{pmatrix} = \begin{pmatrix} S_\mu + S_\xi & S_\mu \\ S_\mu & S_\mu + S_\xi \end{pmatrix}^{-1} \tag{4.7}$$

In sum, the match and dis-match could be decided through Equation 4.5 to verify different face iamges for one person, to get the Equation 4.5, we can use an EM-like algorithm to compute it.

4.2.4 Joint Bayesian Model Learning

From the joint formulation described above, it is clear that if we want to get the log likelihood ratio $r(x_1, x_2)$, the only two unknown variables are covariance matrix S_μ and S_ξ. To solve these two varaibles, it can use an EM-like algorithm.

Initialization. At the first step, S_μ and S_ξ are computed by the between-class and within-class matrix.

E-step. Assume to one subject, it has n samples and each sample has d dimension. We develop a latent variable s to concatenate the identity variable μ and feature variation ξ, it can be represented as $s = [\mu_1; \xi_1; ...; \xi_n]$. With the training data $x =$

$[x_1; x_2; ...; x_n]$, the training data is equal to s times a matrix T.

$$x = Ts, \text{where} T = \begin{bmatrix} I & I & 0 & \cdots & 0 \\ I & 0 & I & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ I & 0 & 0 & \cdots & I \end{bmatrix} \tag{4.8}$$

Because the variables μ and ξ follow the Gaussian distribution $N(0, S_\mu)$ and $N(0, S_\xi)$. The distribution of latent variable s also follows Gaussian distribution $N(0, \Sigma_s)$, where $\Sigma_s = diag(S_\mu; S_\xi; \cdots; S_\xi)$. From Equation 4.8, the covariance matrix can be shown as:

$$\Sigma_x = \begin{bmatrix} S_\mu + S_\xi & S_\mu & \cdots & S_\mu \\ S_\mu & S_\mu + S_\xi & \cdots & S_\mu + S_\xi \\ \vdots & \vdots & \ddots & \vdots \\ S_\mu & S_m u & \cdots & S_\mu + S_\xi \end{bmatrix} \tag{4.9}$$

With the training data, we can have the expectation of latent variable s.

$$E(s|x) = \Sigma_s T^T \Sigma_x^{-1} x. \tag{4.10}$$

M-step. From the E-step, we can get the identity variable μ and face variation ξ. We update their covariance matrix as follows.

$$S_\mu = cov(\mu) \tag{4.11}$$

$$S_\xi = cov(\xi) \tag{4.12}$$

We implement E-step and M-step iteratively until the variables converge.

4.3 Experiment Result

At the experiment part, we use the one thousand pairs offered by Gray Huang[1] for development purposes. In the training part, we use images for identities which have more than four images except the identities in the testing pairs. We first reduce the dimension of the feature to 400 by PCA. The result is Figure 4.4, with the number of feature increasing, the accuracy becomes better. The highest accuracy is 89.3%.

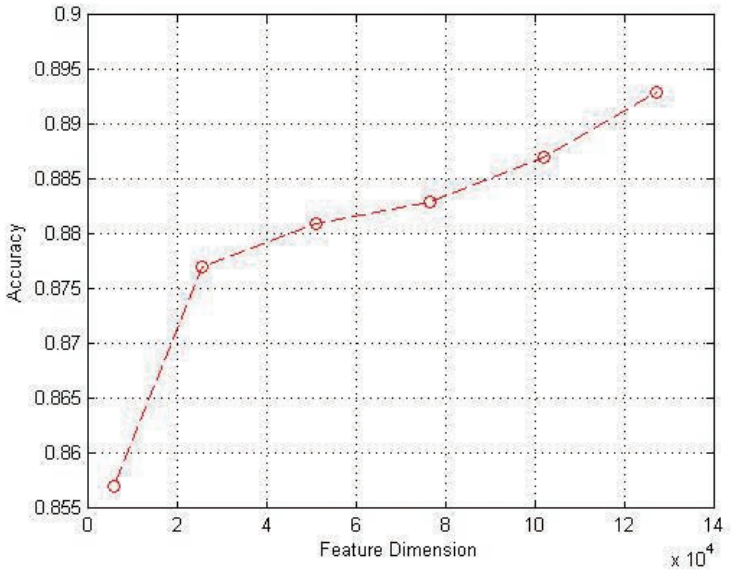

Figure 4.4: Accuracy as a function of the features dimensions

Chapter 5

Experiment

5.1 Face Verification Process

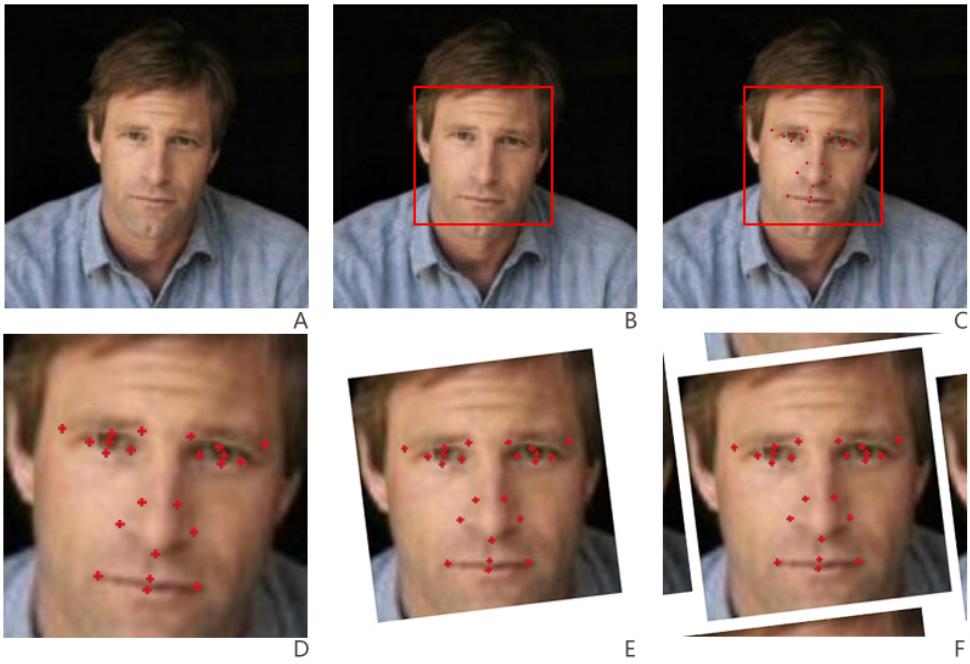

Figure 5.1: Face Verification Process

The process of our system is shown in Figure 5.1. A is the original image, the detection window in B means the detect result by the Adaboost Cascade classifier. C

is the result after facial landmarks detection. D is the process of resize and gray color normalization. E is the face alignment by making the positons of eyes horizontally. F is the parallel padding to contain the edge information.

5.2 Face Verification Result

The setting of the face verification experiment is the same as the experiment setting in chapter 4. However, we used three image pyrimids to get a better searching speed in our demo then we used PCA to reduce the dimension to 800, the result of the model we use is 86.7%.

Chapter 6

Summary and concluding remarks

In this thesis, we build a face verification system which including face detection and facial landmarks detection. Based on the reliable detection of face and facial landmarks, we build a high-dimensional feature with Local Binary Pattern. Also, based on the face recognitin with Bayesian, we use another "smart" way Joint Bayesian Formulation to get high accuracy in LFW dataset, the accuracy now is 89.3%. What's more, we develop a demo combined all technical in this thesis which can do the face recognition without data training.

Bibliography

[1] Gray B. Huang, Manu Ramesh, Tamara Bert, and Erik Learned-Miller. Labeled Faces in the Wild: A Database for studying Face Recognition in Unconstrained Environments.*University of Massachusetts, Amherst, Technical Report 07-49*, October 2007.

[2] Matthew A. Turk and Alex P. Pentland. Face Recognition Using Eigenfaces.*Computer Vision and Pattern Recognition* ,1991.

[3] Paul Viola, Michael J. Jones, Robust Real-time Object Detection.*International Journal of Computer Vision*, 2001.

[4] S.Z. Li, L. Zhu, Z.Q. ZHANG, A. Blake, H.J. Zhang and H. Shum, Statistical learning of multi-view face detection, ECCV, 2002.

[5] M.C.Shin, K. I. Chang, and L. V. Tsap., Does Colorspace Transformation Make Any Difference on Skin Detection?*IEEE Workshop on Application of Computer Vision*, pages 275-279. December 2002.

[6] E. Qsuna, R. Freund, and F. Girosi. Trainng support vector machines: an application to face detection. In *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition*, page 130. IEEE Computer Society, 1997

[7] S. Milborrow and F. Nicolls. Active Shaple Models with SIFT Descriptors and MARS. *VISAPP*, 2014.

[8] M. Uricar, V. Franc and V. Halvac. Detector of Facial Landmarks Learned by the Structured Output SVM. *VISAPP '12, Proceedings of the 7th International Conference on Computer Vision Theory and Applications*, 2012

[9] B. Moghaddam, T. Jebara and A. Pentland. Bayesian Face Recognition. *TR2000-42*, 2002

[10] D. Chen, X. Cao, L. Wang, F. Wen and J. Sun. Bayesian Face Revisited: A Joint Formulation. *ECCV 3, volume 7574 of Lecture Notes in Computer Science, page 566-579*, 2012

[11] D. Chen, X. Cao, F. Wen and J. Sun. Blessing of Dimensionality: High-Dimensional Feature and Its Efficient Compression for Face Verification. *CVPR, page 3025-3032. IEEE*, 2013

[12] S. Miborrow and F. Nicolls. Locating Facial Features with an Extended Active Shape Model *ECCV 4, volume 5305 of Lecture Notes in Computer Science, page 504-513*, 2008

[13] Jerome H. Friedman. Multivariate Adaptive Regression Splines *The Annals of Statistics, Vol. 19, page 1-67*, 1991

[14] T. F. Cootes and C. J. Taylor. Technical Report: Statistical Models of Appearance for Computer Vision. 2004

[15] David G. Lowe. Object Recognition from Local Scale-Invariant Features. *Proceeding ICCV Proceeding of the International Conference on Computer Vision Volume 2 Page 1150*, 1999

[16] X. Cao, Y. Wei, F. Wen and J. Sun. Face alignment by explicit shape regression. *Computer Vision and Pattern Recognition, page 2887-2894*, 2012

[17] Ahonen, Timo A. Hadid and M. Pietikanen. Face description with local binary patterns: Application to face recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 28*, 2006