The Hong Kong Polytechnic University

Department of Electronic and Information Engineering

Face Tracking and Verification in Video

Choi Wing Pong

A thesis submitted in partial fulfilment of the requirements for the Degree of Master of Philosophy

January 2008

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in text.

__________________________(Signed)

___CHOI WING PONG_____(Name of Student)

Abstract

The objective of this research is to develop novel algorithms for face tracking and verification in video. Our research focuses on the areas of facial feature representations and feature matching algorithms, both of which are efficient and effective for tracking and verification. The outputs from this research can be integrated to build a real-time face tracking and verification system.

The automatic detection and tracking of human faces has many valuable applications, such as human computer interaction, visual surveillance, access control in special areas, etc. An accurate face tracker will definitely improve the performance of face recognition and other human activity analysis applications that are currently beyond the capabilities of current face tracking technology. We have proposed an effective face tracking algorithm based on the combination of shape and texture information. The edge map is used to represent the shape of a face, while the texture information is characterized by the local binary pattern (LBP). As the face patterns to be tracked in consecutive frames are highly correlated, accurate tracking can be achieved by searching for the shortest weighted feature distance between the face pattern and the possible face candidates. The weights of the shape and the texture can be adapted for real-time tracking. Both the edge map and the LBP can, to a certain extent, alleviate the illumination effect. Moreover, skin-color-like objects will not be falsely tracked as a face. Our proposed algorithm complements the AdaBoost face detection algorithm to form a multi-view face-tracking system. Experimental results show that our algorithm can track faces in varying poses (tilted or rotated) in real time.

Beyond the detection or tracking of faces, recognition is performed to verify the identity of the tracked faces for visual surveillance. In order to make a practical surveillance system, the face recognition algorithm must be both accurate and efficient. We have proposed that simplified Gabor wavelets (SGWs) be applied to face recognition. Gabor wavelets (GWs) are commonly used to extract texture features for various applications, such as object detection, recognition and tracking. However, extracting Gabor feature is very computationally intensive, so Gabor-related methods are impractical for most real-time applications. This has inspired us to investigate a simplified version of Gabor wavelets and an integral image based algorithm for extracting Gabor-like feature efficiently. We have evaluated the performance of the SGW feature for face recognition. Experimental results show that using SGWs can achieve a performance level similar to that of using GWs, while the runtime for feature extraction by using SGWs is, at most, 4.39

times faster than that using GWs implemented by using the fast Fourier transform (FFT). Extracting the SGW features of 5 different scales and 4 different orientations from a 64×64 image takes only 16.09ms. This is a very encouraging result which allows the use of SGW features for real-time applications.

List of Publications

The following technical papers have been published or accepted for publication based on the result generated from this work.

Journal Paper

1. Wing-Pong Choi, Siu-Hong Tse, Kwok-Wai Wong, and Kin-Man Lam, "Simplified Gabor Wavelets for Human Face Recognition," *Pattern Recognition*, vol. 41, no. 3, pp. 1186-1199, March 2008.

Conference Papers

1. Wing-Pong Choi, Siu-Hong Tse, Kwok-Wai Wong, and Kin-Man Lam, "Simplified Gabor Wavelets for Efficient Feature Extraction," *Proceedings, TENCON 2007 IEEE Region 10 Conference, Taiwan,* pp. 1-4. 30 Oct. 2007-2 Nov. 2007.
2. Wing-Pong Choi, Kin-Man Lam, "An Effective Shape-Texture Weighted Algorithm for Multi-view Face Tracking in Videos," *Proceedings, International Congress on Image and Signal Processing 2008*, Sanya, Hainan, China., vol.4, pp.156-160, 27-30 May 2008.

Acknowledgement

I would like to take this opportunity to express my sincere gratitude to my Supervisor, Dr. K. M. Lam, from the Department of Electronic and Information Engineering of The Hong Kong Polytechnic University, for his patient and guidance throughout my research studies. My MPhil. study would have never been completed without his supervision. His immense enthusiasm for research is greatly appreciated. His professional advice and plentiful experience help me to further improve myself.

I am also thankful to all the members of the DSP Research Laboratory, especially for Mr. K. W. Wong, Mr. Spider K.H. Chung, Mr. H.S. Koo and Mr. Thomas S.H. Tse for their supportive and constructive comments that will never be forgotten. Their expert knowledge and friendly encouragement have helped me to overcome a lot of difficulties during my research studies. I would also like to thank the Centre for Multimedia Signal Processing of the Department of Electronic and Information Engineering for generous support over the past two years.

Last but not least, without the patience and forbearance of my family, the preparation of this research work would have been impossible. I appreciate their constant and continuous support and understanding.

List of Abbreviations

AAM	Active Appearance Model
DARPA	Defense Advanced Research Project Agency
DLA	Dynamic Link Architecture
DMSGW	Demeaned Simplified Gabor Wavelet
FA	False Acceptance Rate
FAE	False Acceptance on the Evaluation Set
FFT	Fast Fourier Transform
FR	False Rejection Rate
FRE	False Rejection Error
GWs	Gabor Wavelets
HID	Human Identification at a Distance
IFFT	Inverse Fast Fourier Transform
LBP	Local Binary Pattern
LDA	Linear Discriminant Analysis
NR-SGW	Non-Rotated Simplified Gabor Wavelet
PCA	Principle Component Analysis
R-SGW	The Rotated Simplified Gabor Wavelet
SGWs	Simplified Gabor Wavelets

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter outlines the motivation for the research on face tracking and verification in video. The problem of face tracking and verification will then be stated. An overview of the techniques for face tracking and verification, and the organization of this thesis will be presented at the end of this chapter.

1.1. Motivation for face tracking and verification in video

The automatic detection, tracking and verification of human faces is currently one of the most active and challenging research topics in computer vision. It is the broad range of applications, including human computer interaction, interactive visual surveillance, access control in special areas, and detection of anomalous behaviors, etc, that motivates the interests of researchers worldwide. For example, the IEEE has sponsored the IEEE International Workshop on Visual Surveillance [1] on three occasions, in India (1998), the U.S. (1999), and Ireland (2000). In [2] and [3], a special section on visual surveillance was published in June and August of 2000, respectively. In [4], a special issue on visual analysis of human motion was published in March 2001. In [5], a special issue on third-generation surveillance systems was published in October 2002. Furthermore, the amount of attention paid to visual surveillance has been noticeably increased since the '9/11' event. This attention comes not only from the academic community, but also from industry and governments.

Many large research projects on this topic have been under taken. For instance, the teleservices of the European Project Banca [6] aims for multimodal biometric access control with face verification as the core of the system. Moreover, a significant percentage of the basic technologies for video-based detection and tracking have been developed under a U.S. government-funded program called Video Surveillance and Monitoring (VSAM) [7]. All these are indications which reflect the fact that human face tracking and verification, or visual surveillance, is an active, challenging and significant research topic.

1.2. Introduction to face tracking and verification in video

Automatic face tracking and verification in video involves different facial image analysis techniques, which mainly include face detection and tracking, human face representation, and human face verification. In addition, different image pre-processing and enhancement techniques must also be applied so as to improve the detection and verification performance.

Detecting human faces in a scene is the first crucial stage in a fully automatic human face tracking and verification system. The detection scheme can be classified according to a cluttered or an uncluttered background in the digital video scene. For example, crowd surveillance is associated with a cluttered or complex background, while passport identification has an uncluttered background. Finding human faces automatically in a cluttered background and under different poses is still a challenging and significant problem.

Face detection is a computational process. In order to maintain the real-time performance, face tracking can be employed to identify the position of the detected face in subsequent frames. However, the tracking algorithm must be able to cope with the changes in poses and occlusion of the tracked face and variations in environment. These problems are still unsolved and under investigation. Facial feature modeling, template matching and estimators are the existing approaches for tracking faces. Some of these can be incorporated to improve the tracking accuracy. However, the tracking results are still not satisfactory yet, due to the unstable environment of the tracking process. For example, lighting variations remain a problem in tracking. In such a situation, facial feature modeling and template matching methods will probably not work when the model (template) and the candidate faces are very dissimilar. In addition, camera motion further increases the difficulty in tracking as it degrades the performance of the approaches that utilize motion for tracking. To handle these difficulties in tracking, adaptive modeling methods that are able to cope with the changes in environment must be studied.

Face verification in a visual surveillance system is performed after a face has being tracked. The identity of the tracked face can be determined. This kind of system is useful for access control in special areas and person-specific identification in certain scenes. When a face is being tracked, a number of its features can be extracted for recognition or verification. Linear subspaces analysis like PCA, LDA and Gabor wavelet features can be employed for face recognition, but these algorithms are relatively slow in processing and, therefore, not suitable for real-time visual surveillance system. As a result, face variation approaches that are both accurate and efficient must be developed to use in real-time visual surveillance system.

1.3. Our methods on face tracking and verification in video

The objectives of this research are to develop efficient techniques for analyzing and verifying human faces in videos. Face detector which is trained by the Adaboost algorithm will be utilized. This is because the face detector is highly accurate, efficient and reliable for detecting frontal faces in still images. Our tracking algorithm incorporates the Adaboost face detector to track non-frontal faces in videos. Edges and texture will be used as features to track faces. These features complement to each other. For instance, poor and variations in lighting make edge detection difficult. In this situation, texture feature can make up the deficiency of the other feature as it is relatively less sensitive to lighting variations. The matching score of the features is in the form of a weighted sum, such that the importance of the respective features in different environments can be adjusted. The features in the current frame will be incorporated with the feature in the previous frame and served as a template for the next frame. In this way, the template feature is adapted to the new appearance of the tracked face.

For face verification in videos, we use a simplified version of the Gabor wavelets to extract features from a tracked face. Gabor wavelets have been shown to have high performance in face recognition, but it is rather computationally intensive for feature extraction. The simplified Gabor wavelets are used to extract features which have a comparable performance to the original Gabor wavelets, but with a much higher efficiency in feature extraction. Such an efficient feature extraction tool is absolutely useful for real-time demanding visual surveillance systems.

1.4. Organization of the thesis

This thesis is organized into 5 chapters, and each of the chapters is outlined as follows:

Chapter 2 is a literature review. The definitions and the challenges of face tracking and face recognition/verification are described. We review some important techniques and current advances on face tracking and face recognition/verification. The problems associated with the techniques will be stated.

Chapter 3 describes our face tracking algorithm. An effective face tracking algorithm based on a combination of shape and texture information is proposed. The edge map is used to represent the shape of a face, while the texture information is characterized by the local binary pattern (LBP). As the face patterns to be tracked in consecutive frames are highly correlated, an accurate tracking can be achieved by searching for the shortest weighted feature distance between the face pattern and the possible face candidates. The weights of the shape and texture can be adapted for real-time tracking. Both the edge map and the LBP can, to a certain extent, alleviate the illumination effect. Moreover, skin-color-like objects will not be falsely tracked as a face. Our proposed algorithm complements the AdaBoost face detection algorithm to form a multi-view face-tracking system. Experimental results show that our algorithm can track faces in varying poses (tilted or rotated) in real time.

Chapter 4 introduces a simplified version of Gabor wavelets (SGWs) and an efficient algorithm for extracting the features based on an integral image. Gabor wavelets (GWs) are commonly used for extracting local features for various applications such as object detection, recognition and tracking. However, extracting Gabor features is computationally intensive, so the features are impractical for real-time applications. This has inspired us to develop the SGW and an efficient feature extraction algorithm. As SGW is effective and efficient for face recognition, it is suggested that SGW can be the core component for face verification in a visual surveillance system.

The last chapter concludes this thesis and suggests further development for this research.

Chapter 2

Literature Review

Face tracking and recognition may be an easy task for human beings, but computerized algorithms are still not mature enough to achieve completely reliable face tracking and a high recognition performance. In this chapter, we will firstly define face tracking and face recognition/verification. The challenges or difficulties will then be discussed. A review of some important algorithms and of progress in made algorithms for face tracking and recognition/verification will also be given.

2.1. Review of Face Tracking

2.1.1. Problem Definition

The goal of face tracking is to localize a face and return the extent of the face from a video or an image [8]. Face tracking can be divided into two types, head tracking and facial feature tracking. Head tracking means that the head of a human is localized from an image in different situations, such as heads with different poses and facial expressions, under occlusion, and with different lighting conditions in a complex background. Facial feature tracking localizes the positions of facial features, such as the eyes, nose and mouth of a face, in an image and correlates the features to track a face. Most of the tracking algorithms have two assumptions that the inter-frame motion of the target object is almost constant, and that the inter-frame changes are either small or predictable. A target face is said to be "loss of lock" [9] or "out of tracked" if a tracking algorithm fails to localize it. Most of the papers present their results in a subjective way. A rectangle or an ellipse which encloses a face is said to be a correct tracking. Figure 2-1 shows some examples of correctly tracked faces. Some papers also present their results in an objective way. The deviations between the tracked target's x-y position and that of the ground truth data against the frame index are plotted. This type of graph can show the tracking accuracy along a video sequence.

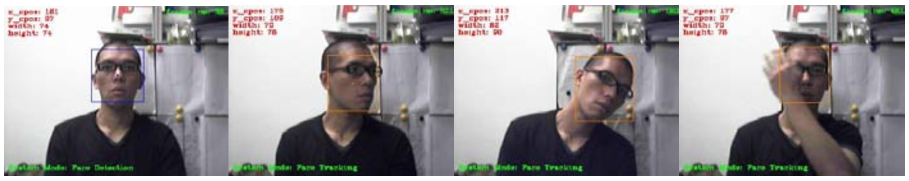


Figure 2-1 Examples of correctly tracked faces.

2.1.2. Challenges

A computerized algorithm interprets a human face as a set of numbers called the face pattern. This face pattern can have a large variation in its appearance. For example, different poses and facial expressions, the presence or absence of structural components, and non-constant environmental lighting, make the distributions of face patterns highly complex [10-12]. Specifically, uncontrolled environmental lighting is the utmost problem for face tracking in practice. Adini *et al.* [13] have made a detailed study of the effect of illumination on face pattern. They drew several conclusions: 1) lighting conditions, in particular the lighting angles, greatly affect the appearances of a face pattern. 2) The distance between the same face with different lighting conditions is even larger than the distance between two different faces under the same lighting condition. These findings suggest that a face tracking algorithm should adapt to the human face and to environmental changes in order to achieve a reliable tracking. Apart from tracking accuracy, the real-time performance is also an important factor in judging the effectiveness of a tracking algorithm. This is because most of the tracking applications online processes.

Face tracking can basically be seen as motion estimation. However, some regions of faces are too smooth to estimate accurately, and some changes in the local appearances of faces are too large to give a reliable result. Sections 2.1.3.1 to 2.1.3.5 will review the different approaches for face tracking, and will comment on their benefits and deficiencies concerning face tracking.

2.1.3. Approaches to Face Tracking

As mentioned at the beginning of this chapter, human beings can track faces relatively effortlessly in different poses and lighting conditions. This tells us that there must exist some primary properties or low-level features in faces that are, to a certain extent, invariant over different conditions for tracking. The face tracking approaches based on different features will be reviewed below.

2.1.3.1. Skin-Color Based Approaches

Among the various low-level features, skin color is the most intuitive for tracking faces. This is because skin color is a prominent feature for identifying human faces. The normalized RGB, HSV and YCbCr are the common choices for representing the skin color. These color spaces are less sensitive to lighting conditions, ethnicity and camera characteristics when compared to other color spaces [14-17]. This is because the luminance component is explicitly separated. The Bayesian classifier, with the histogram technique and the Gaussian classifier, is effective, and is commonly used for modeling and classifying skin color pixels [18]. The conditional pdfs of the skin color and the non-skin color pixels in the classifiers are estimated using histograms or parametric density estimation techniques [19-22].

After skin color-pixel classification, the connected components are identified, and small components are treated as noise, and deleted. For each connected component, an ellipse or a rectangle which best fits the connected component is computed. The ellipse or rectangle, whose major to minor axis ratio is closest to the golden ratio of an ideal face according to the anthropometric recommendation [23], is chosen to represent a face. Faces can be tracked by applying this method in successive frames. Figure 2-2 shows the results of skin-color segmentation and face identification.

The robustness against rotations, scaling and partial occlusions are high for face tracking via skin color. In addition, it is computationally efficient for processing skin color as a feature. However, this face tracking approach will fail if the illumination condition varies, or if the background image has a skin-like color. Several papers have tried to tackle this problem. In [24], a Gaussian mixture model to represent the skin-color model and update the model's parameters over time was proposed. In [25], an adaptive color model was proposed, which uses the basic structure of condensation for model updating. In [26], a color model switching strategy has been proposed, which tracks faces under different illumination by using different color models. Nevertheless, all these algorithms will fail when the lighting condition changes drastically or the background contains skin-color-like objects.

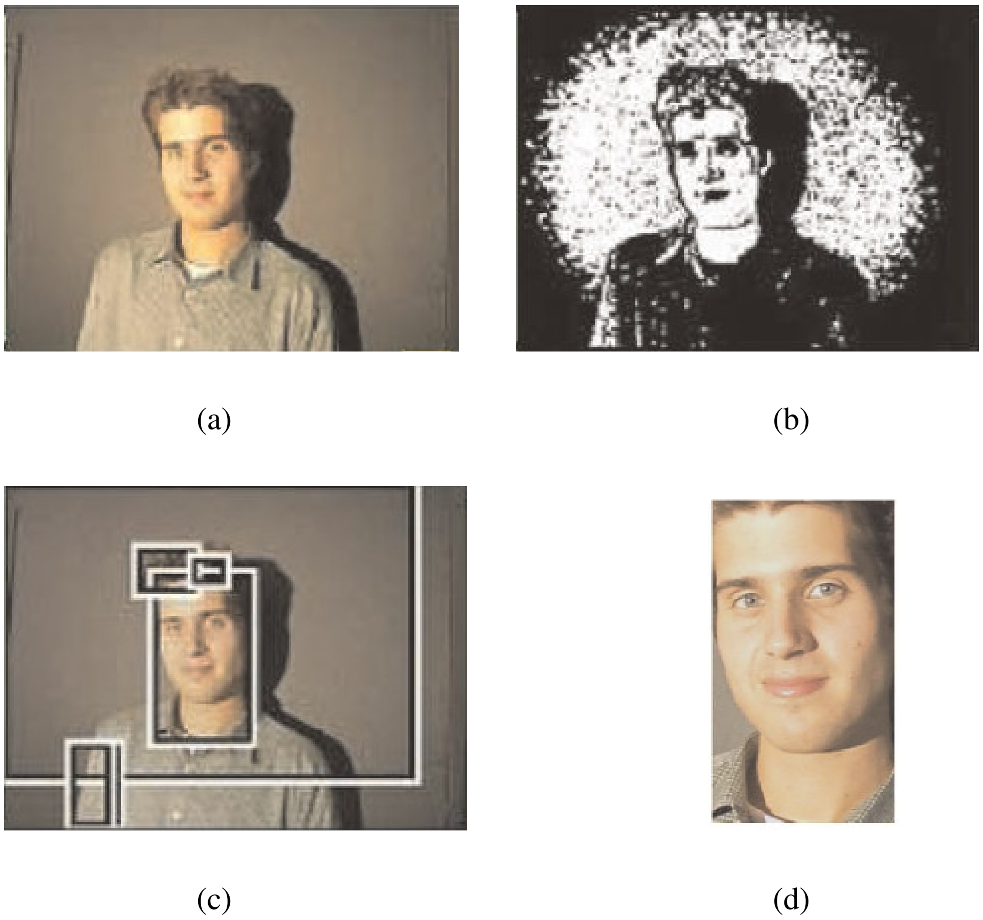


Figure 2-2 The results of skin-color segmentation and face identification. (a) The original image, (b) the skin-color pixel segmentation results, (c) the connected regions which are the face candidates, and (d) the identified face region. (Figure reprinted from [17])

2.1.3.2. Facial Feature Based Approaches

Each face consists of different facial features by which we can identify faces. Facial features include eyes, nose, mouth, hair and a face's contour shape. There are several different representations methods for facial features. Maure and Malsburg [27] and Mckenna *et al* [28] used the texture feature extracted by Gabor filters as the cues for face tracking (a more detailed review on Gabor wavelets is given in Section 2.2.3.2). Only the selected positions on a face shape template or a face-mesh model are considered. This approach is usually insensitive to global intensity changes and, to a certain extent, insensitive to illumination changes. However, texture feature extraction is computationally expensive, and is therefore hard to implement for a real-time application. In addition, the error in tracking will cause drifting in the template face model, without error recovery. Figure 2-3 shows some tracking results, based on tracking Gabor features, for the selected facial features.

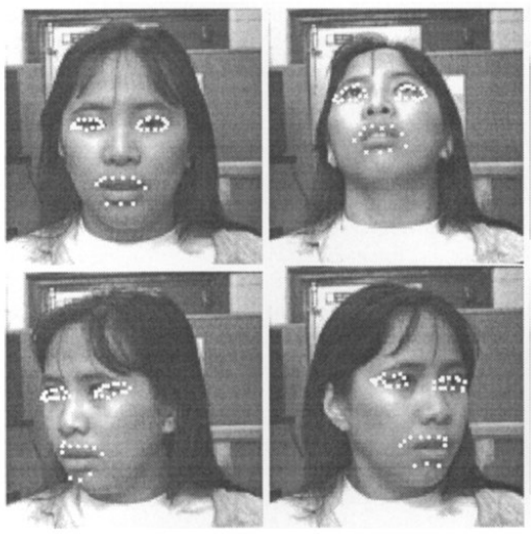


Figure 2-3 Tracking based on matching the Gabor features of the selected facial features. (Figure reprinted from [28])

A face can also be tracked using elliptical contour tracking [29-31]. This method is not sensitive to background color and illumination changes, but a highly cluttered background fails this tracking approach. Birchfield [32] used gradient intensity to measure an object's boundary shape, and a color histogram to measure the color of an object's interior. In this way, only the elliptical contours with a skin-color interior will be extracted in a cluttered background. However, the shape and the color features are not integrated properly, so it occasionally fails to extract the face contour. Figure 2-4 shows the results of the tracking method proposed in [32].

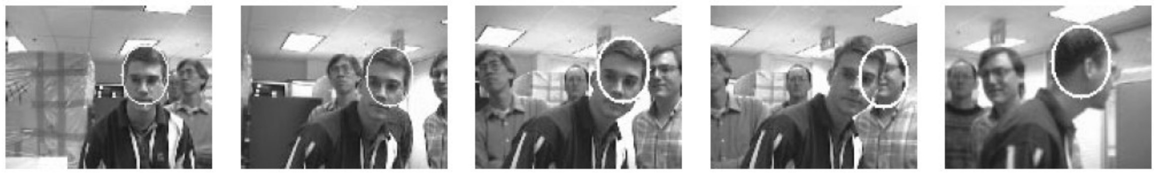


Figure 2-4 Tracking based on using gradient intensity to measure an object's boundary shape and a color histogram to measure the color of an object's interior. (Figure reprinted from [32])

2.1.3.3. Rapid Continuous Face Detection Based Approaches

As mentioned at the beginning of Section 2.1.2, the most primitive approach for tracking is by motion estimation or block matching [37, 38]. However, this approach is computationally inefficient. Recently, face pattern classifiers have received a great deal of attention from researchers. A learning-based approach has so far been the most effective for constructing face and non-face classifiers. Learning-based approaches attempt to learn a boundary between the face and non-face patterns which is highly non-linear due to the fact that the face manifold, formed by variations in facial appearance, lighting, head pose and expression, is highly complex [39, 40]. Among the different learning algorithms, the AdaBoost learning algorithm for learning a frontal face classifier is the most impressive. AdaBoost solves the following three fundamental problems of learning a highly non-linear decision boundary in one boosting procedure:

(1) learning effective features from a large feature set,

(2) constructing weak classifiers each of which is based on one of the selected feature and

(3) boosting the weak classifiers into a stronger classifier.

The classifier processes all the sub-windows of an image to detect faces. This process is speeded-up via cascading. Figure 2-5 shows the structure of a cascade classifier. Simple, boosted classifiers can reject many negative candidates while detecting all positive instances. A series of such simple classifiers can achieve good detection performance while also eliminating the need for any further processing of those negative candidates. This cascaded classifier can achieve a real-time performance for detecting frontal faces (14 frames per second) in 320×240 images [41]. A detection rate of 82.7% with 10 false alarms on the CMU+MIT test set has been reported. Figure 2-6 shows some detection results for the AdaBoost trained classifier. Its long learning time and its dedication to frontal-face detection are the weaknesses of the Adaboost learned face classifier.

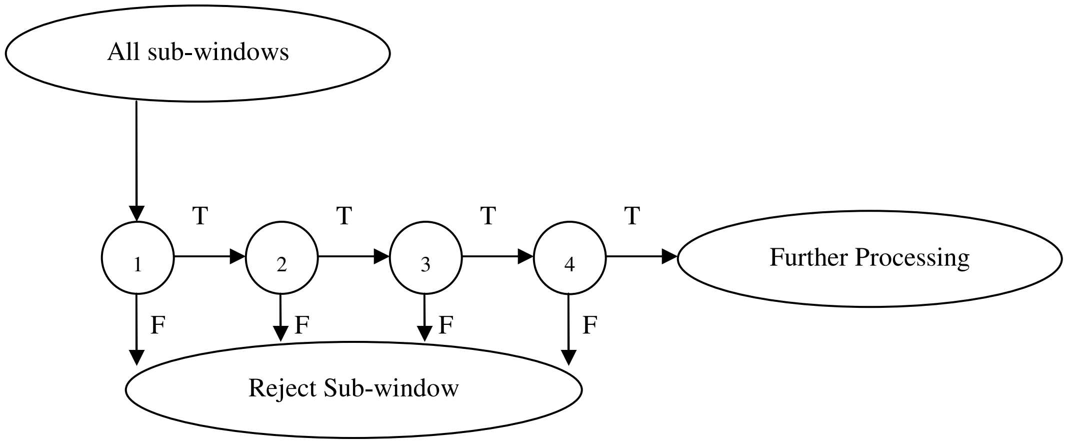


Figure 2-5 The structure of cascade classifier. A series of classifiers are applied to every candidate. The initial classifier eliminates a large number of negative candidates examples with very little processing. Subsequent layers eliminate additional negative but require additional computation. After several stages of processing, the number of candidates has been reduced radically. Further processing can take other forms, such as additional stages of the cascade, or another face detection algorithm.

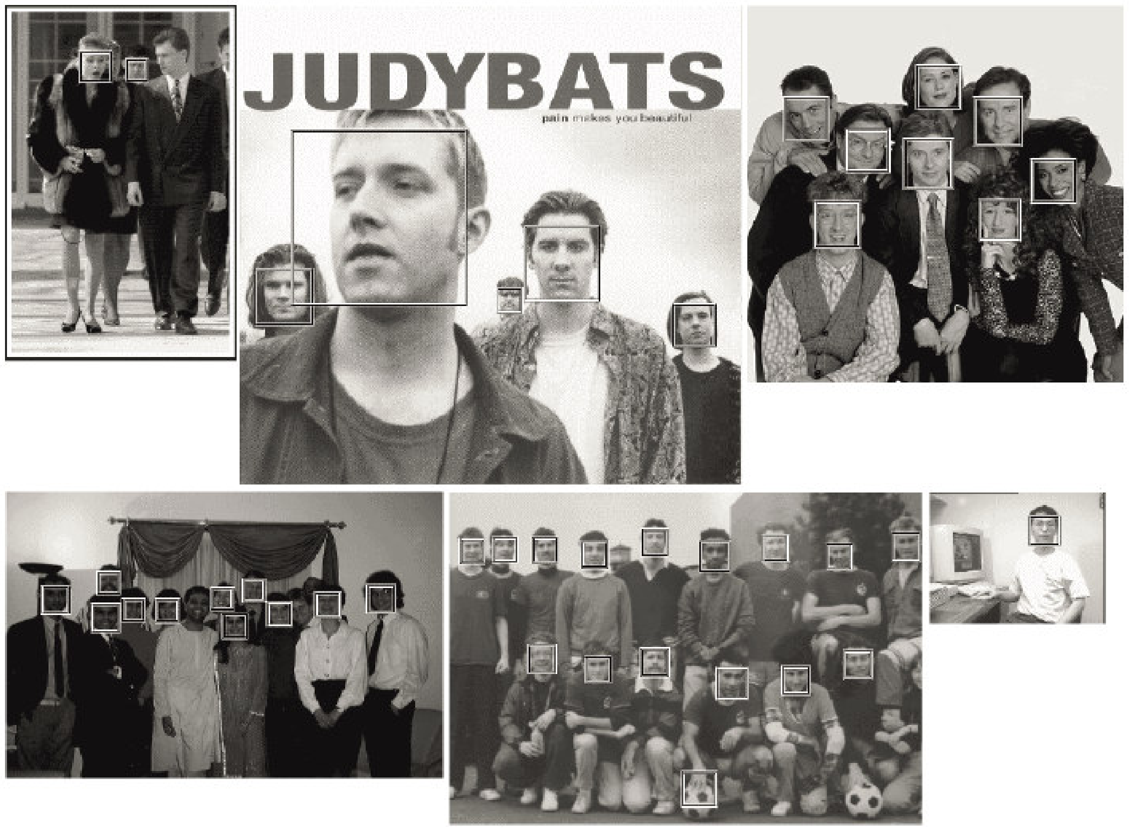


Figure 2-6 The face detection results of the AdaBoost-trained classifier. (Figure reprinted from [41])

S. Li *et al*. developed the FloatBoost learning method and a detector pyramid architecture [42], which improve the Adaboost learned classifier for face detection. It uses a backtrack mechanism after each iteration of AdaBoost learning to minimize the error rate directly, rather than minimizing an exponential function of the margin, as in the AdaBoost algorithm. It requires fewer weak classifiers and can achieve lower error rates in both training and testing for multi-view face detection. A detection rate of 83.6% with 10 false alarm on the CMU+MIT test set has been reported. However, when comparing the number of weak classifiers used, the FloatBoost requires approximately 5 times longer in training than the AdaBoost algorithm does, and the real-time performance is lowered to about 5 frames per second. Figure 2-7 shows some face detection results of the FloatBoost trained classifier.

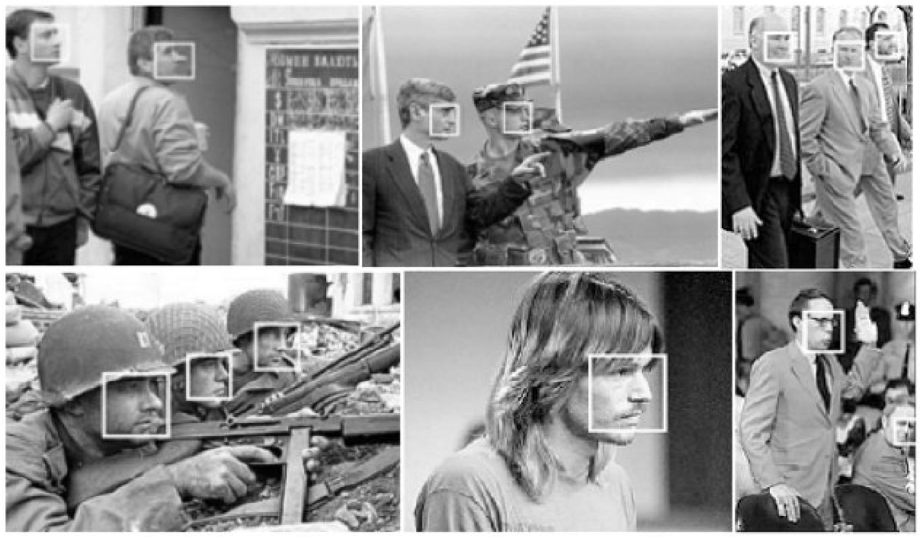


Figure 2-7 Multi-view face detection results for the classifier trained by the FloatBoost algorithm. (Figure reprinted from [42])

2.1.3.4. Active Appearance Model (AAM) Based Approaches

Active appearance model (AAM) based approaches can also be treated as template matching approaches. The AAMs for a face are generative, parametric models that show both shape and appearance variations of a face [33]. These variations are represented via subspace methods such as principal component analysis (PCA), which finds a subspace representing the maximum variance of a given set of data. An AAM face model can be constructed from training data, and face tracking is achieved by fitting the trained model to a face candidate. Figure 2-8 shows an example of the model fitting using the AAM-based approach.

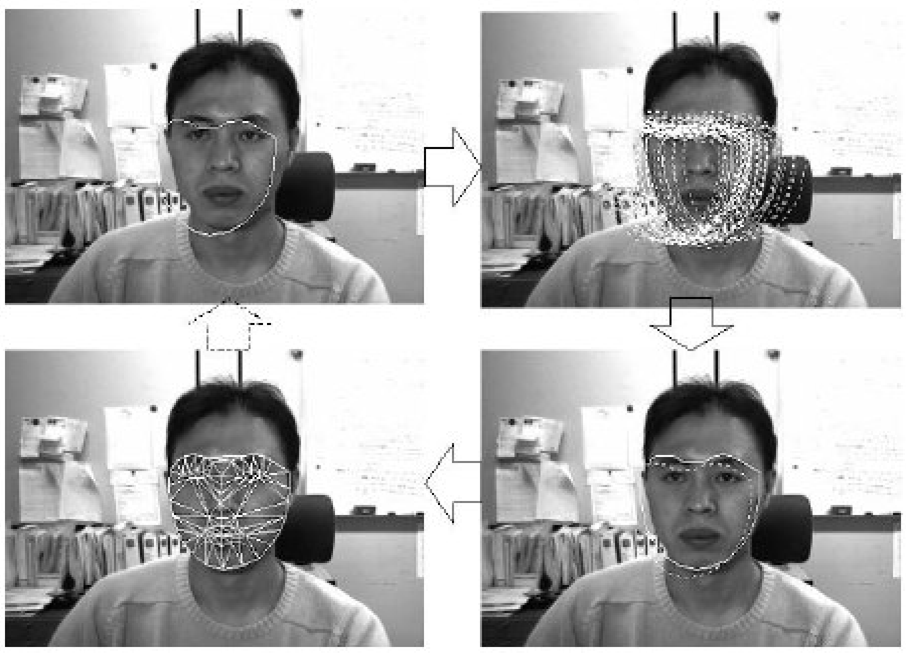


Figure 2-8 Model fitting using the AAM-based approach. (Figure reprinted from [34])

In [34-36], the view-based AAM and the background robust AAM, respectively, were proposed. The former handles the problem of face rotation, while the later can solve the problem of cluttered backgrounds. However, the fitting process using AAM fails occasionally, and an error will be propagated into subsequent frames, which causes drifting of the AAM. Moreover, this approach cannot be used a real-time applications.

2.1.3.5. Statistical Estimation Based Approaches

This type of face tracking approach is also known as analysis-by-synthesis, and it is used in a predict-match-update style. Firstly, the face status for the next frame is predicted according to prior knowledge and the tracking history. Then, the predicted face status is synthesized and projected onto the image plane for comparison with the image data. An evaluation function is needed to measure the similarity between the projected face status and the image data. According to different search strategies, this is done either recursively or using sampling techniques until the correct status is finally found and is used to update the face status.

A Kalman filter is a classical computational mechanism for incorporating predictions from an autoregressive model of the face status's dynamics [9, 44] into a stream of observations for updating the face status. In terms of face tracking, the Kalman filter addresses the problem of estimating a face state (position and size) in a dynamic process. The prediction of the next state of a tracked face is done by a state-transition matrix which determines the dynamic of the tracked face. The predicted state is refined by a measurement, and this refined state is used to predict the next state of the face. Figure 2-9 shows the flow of the Kalman filter cycle.

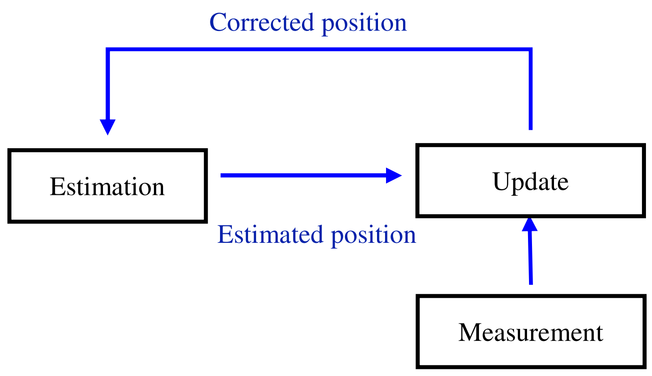


Figure 2-9 The Kalman filter cycle

The most straightforward setting is for face contour tracking by using the idea of normal displacement as measurement [44], as shown in Figure 2-10. This setting of the Kalman filter for face tracking suffers from the problem of clutter, as shown in Figure 2-11, which tends to generate multiple observations at each location, and from the problem of dynamics, which assumes the linear dynamic of a target face is assumed. The extended Kalman filter [45] has been designed to deal with non-linear dynamics, but only a mild non-linearity can be handled by local linearization of motion dynamics and noise co-variances.

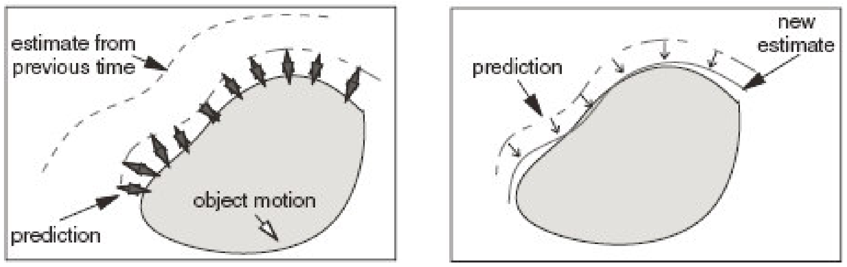


Figure 2-10 Tracking a moving object by using the Kalman filter. The prediction and measurement phases for contours, with observations (double arrows) of normal displacement are shown. (Figure reprinted from [9])

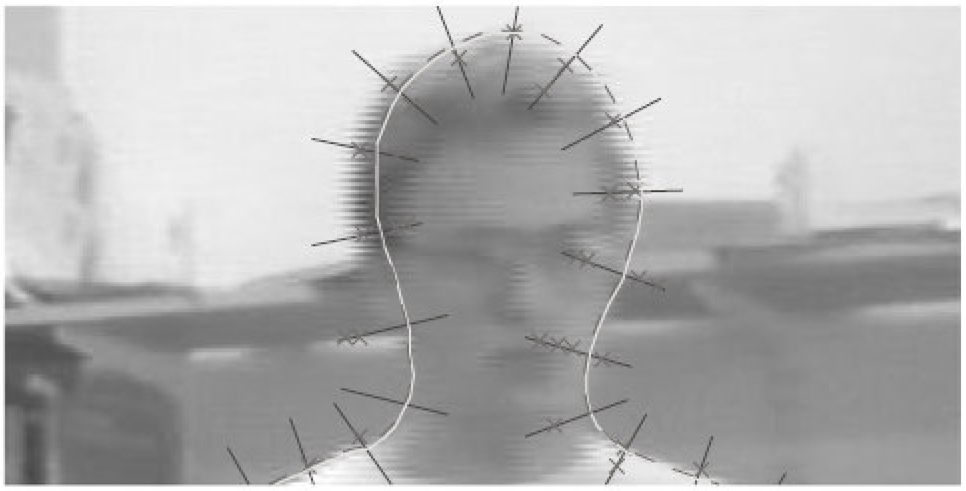
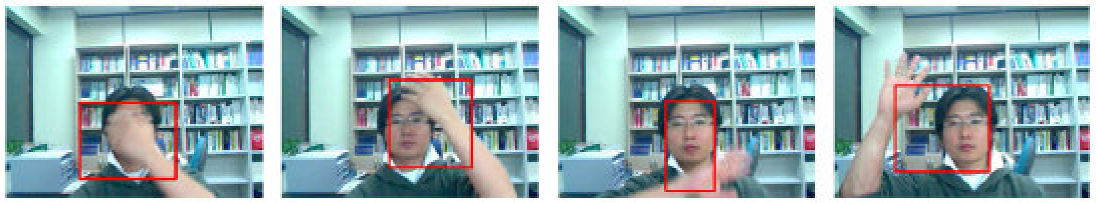
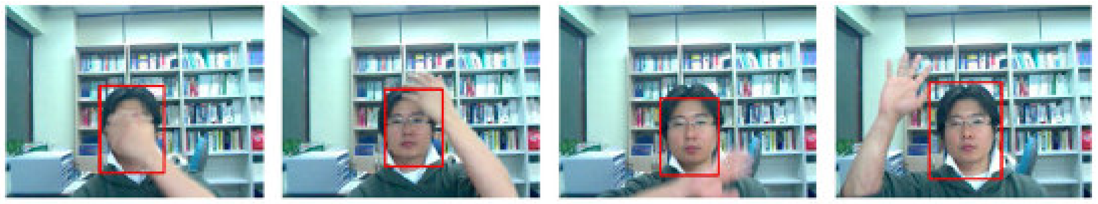


Figure 2-11 An active contour and the normals are shown. Crosses mark observations of high contrast features, some of which are triggered by a true object outline, while others are responding to clutter, both inside and outside the object. (Figure reprinted from [46])

Recently, the particle filter has been developed to solve the problems of the Kalman filter. The pioneers of using the particle filter for visual tracking are M. Isard and A. Blake [47]. The particle filter is a kind of conditional density propagation method for visual tracking. Based upon sampling the posterior distribution estimated in the previous frame, it is extended to propagate these samples iteratively to successive images. By combining a tractable dynamic model with visual observations, it can accomplish a highly robust tracking of face motion. One important advantage of particle filter is that it allows the information from different measurement sources to be fused in a principled manner [48-51]. Data fusion with particle filters has been mostly confined to skin color and edge cues inside and around simple face shapes [52]. Figure 2-12 shows the face tracking results obtained using the skin-color based particle filter and the skin color and shape feature fused particle filter. We can see that the tracking result from the multi-features fused particle filter is not affected by skin-color like objects (the hand), while the skin-color based particle filter cannot track accurately.



(a)



(b)

Figure 2-12 The face tracking results obtained by (a) the skin-color-based particle filter, and (b) the skin color and shape feature fused particle filter. (Figure reprinted from [52])

However, the particle filter usually requires a relatively large number of samples to ensure a fair estimation of a face's current state. In other words, the number of samples, or particles, used is an unknown, and is tracking scenario dependent. Moreover, the larger the number of particles used, the slower the tracking is.

2.2. Review of Face Recognition/Verification

2.2.1. Definition of Face Recognition/Verification

Face recognition is understood as automatically recognizing the identity of a person from all facial images in a database. In terms of computing, it is defined as presenting a probe sample, and a face recognition algorithm returns all the similarities between the probe sample and all gallery samples in a database. Feature extraction and classification are the main steps in all face recognition algorithms. Employing different features and classification algorithms will certainly produce different face recognition results. The efficiency of a face recognition algorithm is measured by its face recognition rate, which is the ratio between the number of testing face images that are being correctly recognized and the total number of testing face images.

The definition of face verification is quite similar to face recognition. Face verification can be divided into two types. The first type involves confirming or denying the identity claimed by a person. The question, *"Is this person whom he/she claims to be?"* is asked. An algorithm verifies the provided identity of a person by computing the similarities or distances between the person and the gallery images of the claimed identity in a database. This is a one-to-one matching, and is often called verification or authentication. The second type involves establishing the identity of a given person out of *N* people in a database. The question, *"Who is this?"* is asked. An algorithm identifies the person by comparing the similarities or distances between the person and all gallery images in a database. This is a one-to-*N* matching ,and is called face identification.

People often confuse the definitions of face verification and face recognition. In fact, they are different in three fundamental aspects [53]:

(1) An authorized user of a personal identification system, who is called a client, is assumed to be cooperative and makes an identity claim. It is not necessary to perform matching between the face of a client and the entire gallery in a database. The face image of a client is thus compared to a subset of the gallery in a database only, while for face recognition, the potentially large gallery in a database is matched with the face image of a client.

(2) The processing performance of an automatic authentication system must operate in near real-time to be acceptable to users.

(3) The case of a previously unseen person, who is called an imposter, is presented to the system. This is of the utmost importance for

authentication, whereas in face recognition, only images of people from the training database are presented to the system.

The algorithms for face verification are often directly transplanted from those used for face recognition. For the task of personal verification, a standard protocol for performance assessment has been defined. The Lausanne protocol randomly splits all subjects into a client and an imposter group. Three different sets of images are built as follows [53]:

Set 1. *Training set:*

It is used to construct client models.

Set 2. *Evaluation set:*

It is selected to produce client and impostor access scores, which are used to find a threshold that determines if a person is accepted or not. This threshold is often called the client-specific threshold, or global threshold. According to the Lausanne protocol, the threshold is set to satisfy certain performance levels (error rates) in the evaluation set.

Set 3. *Testing set:*

It is selected to simulate realistic authentication tests where an impostor's identity is unknown to the system.

The False Acceptance rate (*FA*) and the False Rejection rate (*FR*) are used to measure the performance of a verification algorithm. *FA* is for cases where an impostor is accepted, while *FR* is for cases where a client is rejected. *FA* and *FR* are computed by:

$$FA = \frac{EI}{I} \times 100\% \qquad FR = \frac{EC}{C} \times 100\% \tag{2-1}$$

where *EI* is the number of impostor acceptances, *I* is the number of impostor claims, *EC* is the number of client rejections, and *C* is the number of client claims. Both *FA* and *FR* are influenced by an acceptance threshold. To simulate a real application, a threshold is set based on the evaluation set such that false acceptance error set (*FAE*) and false rejection error (*FRE*) are obtained. This threshold is then applied to measure the *FA*, *FR* based on the test data. Three thresholds are defined in the evaluation set:

$$\begin{aligned} &(1) \quad T_{FAE=0} = \arg\min_T (FRE \mid FAE = 0) \\ &(2) \quad T_{FAE=FRE} = (T \mid FRE = FAE) \qquad (2\text{-}2) \\ &(3) \quad T_{FRE=0} = \arg\min_T (FAE \mid FRE = 0) \end{aligned}$$

Consequently, performance in the test set is characterized by six error rates:

(1) $FA_{FAE=0}$

(2) $FA_{FAE=FRE}$

(3) $FA_{FRE=0}$ (2-3)

(4) $FR_{FAE=0}$

(5) $FR_{FAE=FRE}$

(6) $FR_{FRE=0}$

In general, the lower the values of FA and FR, the better an algorithm is.

2.2.2. Challenges of Face Recognition/Verification

The large variation in face patterns is the main challenge for all face processing algorithms. Face recognition and verification also suffer from face manifolds caused by different factors:

(1) Pose: The images of a face vary due to the relative camera-face pose, and some facial features, such as the eyes or the nose may become partially or wholly occluded.

(2) Facial expression: The appearance of a face is directly affected by a person's facial expression.

(3) Occlusion: A face may be partially occluded by other objects. In an image with a group of people, some faces may partially occlude each other.

(4) Face orientation: A Face varies with different rotations about the camera's

optical axis.

(5) Presence or absence of structural components: Structural components, such as glasses, beards and different hair styles, vary the face patterns to a large extent. This is because these structural components vary in shape, color, size and texture, which further increase the variability of a face pattern.

(6) Lighting conditions: As mentioned in Section 2.1.2, the distance between the same face with different lighting conditions is even larger than the distance between two different faces under the same lighting condition. Moreover, camera characteristics (sensor response, lenses) affect the appearance of faces in an image.

In addition, real-time performance is important for practical face verification systems. Therefore, feature extraction and classification of a face recognition/verification algorithm should be computationally efficient to fulfill the real-time requirement.

Most of the literature has assumed that each face in an image is normalized in terms of position, orientation, size and lighting condition. These pre-processing step are called face alignment and normalization. The next section will discuss different approaches to tackling the problems of face recognition/verification.

2.2.3. Approaches for Face Recognition/Verification

Research on the automatic machine recognition/verification of faces has been carried out over 30 years, and is on going. Researchers have been concerned with the issue of whether face perception is done holistically or by local feature analysis. This concern is combined with different techniques, such as image processing, pattern recognition, computer vision, computer graphics, etc., to develop a sequence of algorithms, as well as systems for automatic machine recognition/verification of human faces. In sections 2.2.3.1 to 2.2.3.3, the most significant advances in face recognition/verification, which include linear subspace analysis, textural analysis and graph matching techniques, will be reviewed.

2.2.3.1. Linear Subspace Analysis (Holistic approach)

The most popular technique for face recognition/verification is the subspace methods. This is mainly due to its effectiveness and computational efficiency for feature extraction and representation. The subspace algorithms represent a facial image by a feature vector. With a set of feature vectors, projections or bases that optimize some criteria defined over the feature vectors that correspond to different classes are found. Afterwards, the original high-dimensional image space is projected into a low-dimensional one. So, the subspace methods reduce the dimensionality and represent the global feature of facial images. Feature classification is usually performed according to a simple similarity measure (usually employing the Euclidean distance) in the projected multi-dimensional space. Figure 2-13 shows the general scheme of the linear subspace methods.

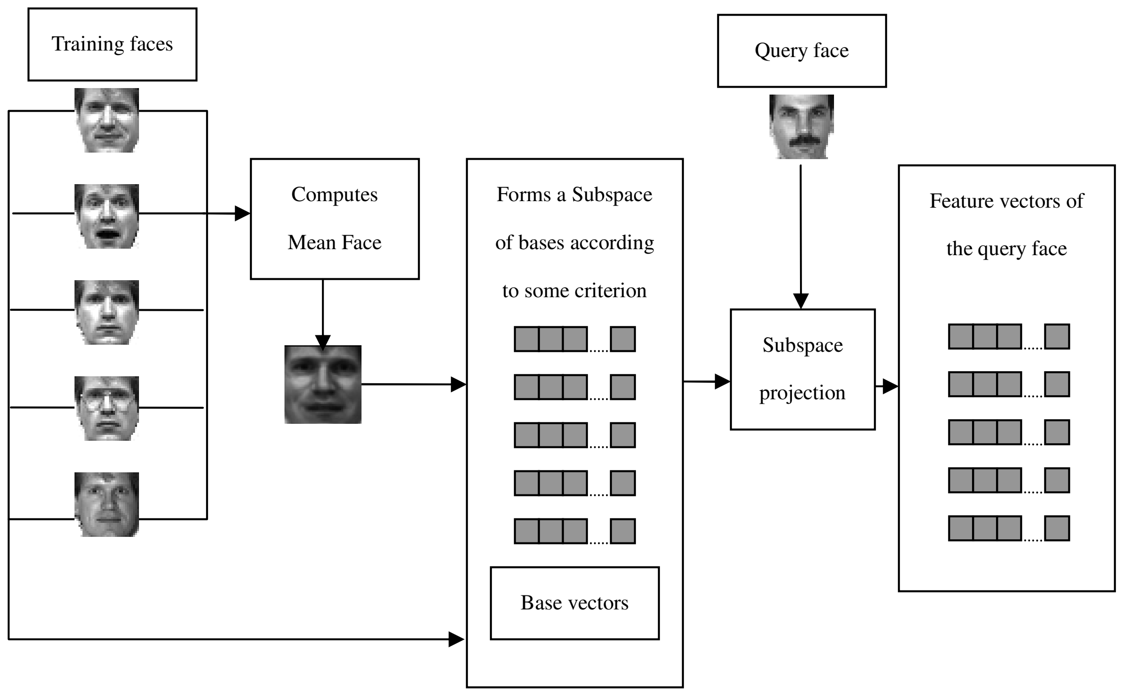


Figure 2-13 The general scheme of the linear subspace methods

Different criterion will produce different bases and, as a result, the projected subspace will also have different properties. Principal Component Analysis (PCA)

[54-56] is a classical method and is widely used for face recognition and face reconstruction. The major idea of PCA is to decompose an input image into a linear combination of a small collection of basis images, which is called the eigenfaces. The eigenfaces are orthogonal and capture the directions of maximum variance in the training face images. Therefore, the eigenfaces capture the global feature of the training face images and the projected input image will have a reduction in dimension while the main components are maintained. For face recognition, when the testing images have variations caused by global components such as lighting or perspective variations, the performance of PCA will be greatly reduced [49].

A better alternative of PCA is the Linear Discriminant Analysis (LDA) [57, 58]. LDA often outperform PCA because it expressly provides discrimination among the classes, while PCA deals with the input data in their entirely and without considering the underlying structure. The main objective of LDA is to find a base of vectors providing the best discrimination among the classes. This base of vectors maximizes the between-class difference and minimizes the within-class difference. The between- and within-class differences are represented by the corresponding scatter matrices. Although LDA is often considered to outperform PCA, LDA provides better performance only when a large training set is available [59]. In addition, LDA is more sensitive to different training data sets when compared to PCA.

2.2.3.2. Textural Analysis

The intensity of an image is the only source from a camera used for face recognition. However, a lot of variations, such as albedo and shape of the face, lighting, etc., are all encoded as intensity. To eliminate these extrinsic factors, invariance textural feature analysis for face recognition is widely studied. By invariance, not only are features meant which are invariant to a set of geometric transforms, but also methods to perform face recognition regardless of pose and image conditions using features which are not invariant. Textural features refer to the spatial arrangement of intensity in an image. In other words, it is a local feature descriptor. In face recognition, the most typical invariance requirements are the illumination, orientation, scale and translation. General textural descriptor such as moment invariants do exist, but have problems in practice since they require precise segmentation and uniform lighting. Among different textural descriptors, Gabor wavelet (GW) and local binary pattern (LBP) have recently received more attention.

2.2.3.2.1. Gabor Wavelets

The Gabor wavelets, whose kernels are similar to the response of the two-dimensional receptive field profiles of mammalian simple cortical cell [60], exhibit the desirable characteristics of capturing salient visual properties such as spatial localization, orientation selectivity, and spatial frequency [61]. The Gabor wavelets can effectively abstract local and discriminating features, which are useful for face recognition [62-64]. In the spatial domain, a Gabor wavelet is a complex exponential modulated by a Gaussian function [60, 62, 65]. By selecting different frequencies and orientation of the complex exponential, a family of Gabor kernels can be obtained which can be used to extract features from an image by convolution. The convolution can be computed efficiently by performing the fast Fourier transform (FFT), then point-by-point multiplications, and finally the inverse fast Fourier transform (IFFT). This IFFT output is the Gabor representation of an image. The magnitude of the IFFT is considered to be the local properties of an image [62] and is less sensitive to the lighting conditions [66]. Euclidean distance is usually chosen to compare the similarity between the Gabor representations of two different face images. High computational complexity is the major disadvantage of using Gabor wavelets for face representation. Figure 2-14 shows the Gabor representation of a face image at 4 different frequencies and 8 different orientations.

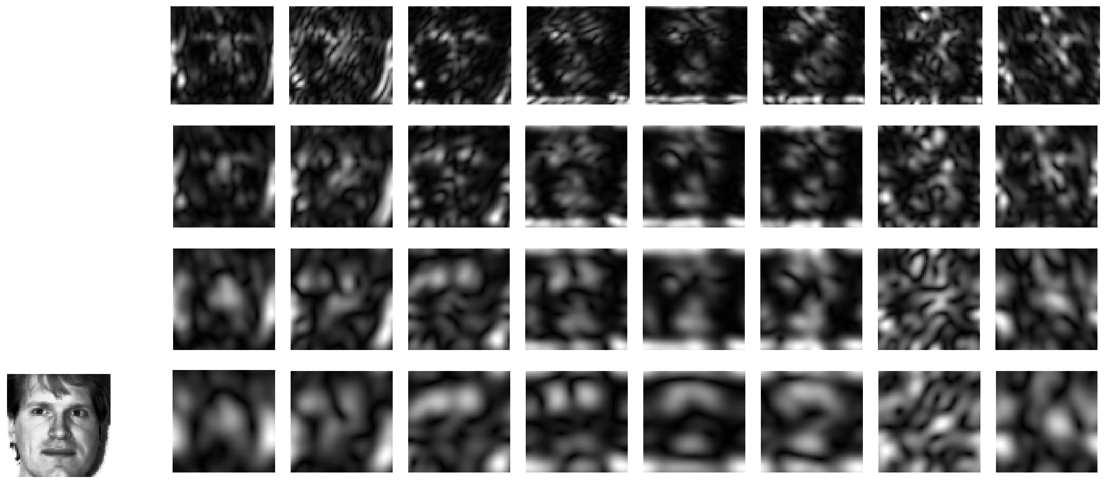


Figure 2-14 Gabor representations of a human face. (a) The original face, (b) The magnitudes of Gabor representations with 4 different frequencies and 8 orientations.

2.2.3.2.2. Local Binary Pattern (LBP)

The local binary pattern (LBP) [67] is one of the best performing texture descriptors and it has been widely used in various applications. It has proven to be highly discriminative and its key advantages are its invariance to monotonic gray-level changes and computational efficiency. These advantages make it suitable for demanding image analysis tasks. The first use of LBP features for face representation is in the ECCV 2004 conference [68]. After this, several research groups [69-75] have adopted LBP for face recognition.

The LBP operator was originally designed for texture description. The operator assigns a label to every pixel of an image by thresholding the 3×3 neighborhood of each pixel with the center pixel value and considering the result as a binary number. Then, the histogram of the labels can be used as a texture descriptor. Figure 2-15 shows an illustration of the basic LBP operator.

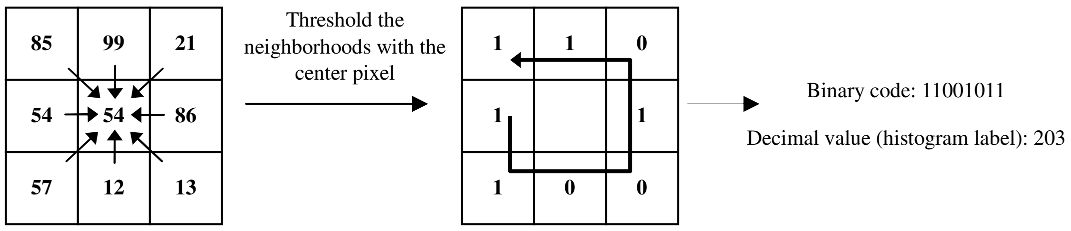

85
99
21
54
54
86
57
12
13
Threshold the neighborhoods with the center pixel
1
1
0
1
1
1
0
0
Binary code: 11001011
Decimal value (histogram label): 203

Figure 2-15 An illustration of the basic LBP operator.

To deal with textures at different scales the LBP operator can be extended to use neighborhoods of different sizes [76]. Defining the local neighborhood as a set of sampling points evenly spaced on a circle centered at the pixel to be labeled allows any radius and number of sampling points. Bilinear interpolation is used when a sampling point does not fall in the center of a pixel. Figure 2-16 shows examples of these advanced versions of LBP operator.

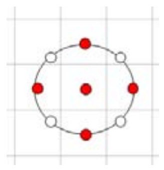
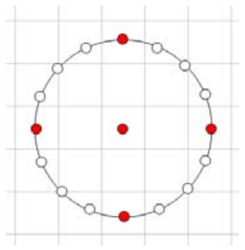

(a)

(b)

Figure 2-16 The advanced versions of the LBP operators. (a) a 8 circular sampling points LBP operator with radius equal to 1 pixel, (b) a 16 circular sampling points LBP operator with radius equal to 2 pixels

The invariance to monotonic lighting variation of the LBP is illustrated in Figure 2-17. Monotonic lighting variation will either shift-up or shift-down the gray-levels in an image. Although the lighting varies considerably in Figure 2-17(b) when compared to Figure 2-17(a), the two LBP are the same. This is because LBP at a pixel is computed by thresholding with its neighborhoods. Shifting-up or shifting-down the gray-levels of an image has no effect on the thresholding.

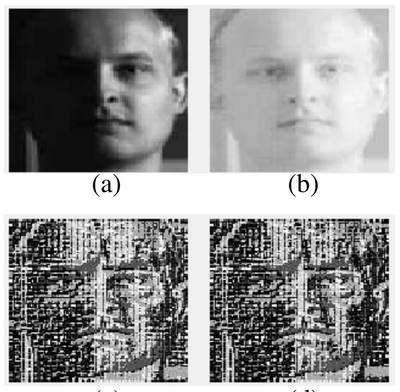


Figure 2-17 An illustration of the invariance to monotonic illumination variations of the LBP. (a) an face image; (b) the face image of (a) with brightness increased monotonically; (c) the LBP of (a); (d) the LBP of (b).

The idea of using LBP for face description is motivated by the fact that faces can be seem as a composition of micro-patterns which are well described by such local feature-based method. In contrast with the holistic texture descriptors, holistic texture descriptors tend to average over a large image area and have invariance on translation and large scale rotation. This is a desirable property only for ordinary textures but not appropriate for faces. This is because holistic texture descriptors fail to retain the information about spatial relations of the micro-patterns of a face, due to its invariance feature properties. In addition, holistic texture description methods seem to be less robust against variations in pose or illumination than the local-feature based methods [76-78], which is understandable given the limitation of the holistic representations. This reasoning leads to the use of LBP for face representation.

The face recognition is performed by comparing the normalized LBP labeled histograms of two faces. The comparisons can be a correlation distance or Chi square distance which are both commonly used for comparing two histograms. As a face is composed of several facial features with different importance, the comparison can be divided into several weighted regions. In [79], a face is divided into 49 regions and weighting is assigned to each of the regions. Salient facial regions like eyes and mouth are assigned with a higher weighting. The comparison of two faces is in the form of summing the weighted Chi square distances of the regions. Through this configuration, the face recognition result is improved by 8% on average when compared with the approach without region dividing.

2.2.3.3. Graph Matching

As mentioned in Section 2.2.2, one of the difficulties of face recognition is to recognize faces with different variations such as facial expressions, poses and uneven lighting conditions. The dynamic link architecture (DLA) [62] is an effective face recognition approach to handle the above mentioned difficulties. An object is recognized by using sparse graphs, whose vertices are labeled by a multi-resolution description in terms of a local power spectrum, and whose edges are labeled by geometric distance vectors. The local features are extracted by using Gabor wavelet. Object recognition can be formulated as an elastic graph matching, which is performed by minimizing a matching cost function. There are two stages within a graph matching process. The first stage one is to position the grid over a face, and the second stage is to allow the grid structure to deform until each node has achieved a minimum of the cost function. The difference between the edge labels of the image graph and the model graph, and the Gabor features' similarity of the corresponding vertex labels between the image graph and the model graph, are weighted and summed to form a matching cost. As the grid structure is allowed to deform, this graph matching approach can handle face recognition with facial expression and pose variations. However, it is time-consuming to let define grid structure to deform and to compute the matching cost for each of the nodes of the grid.

2.2.4. Conclusion

In this chapter, the definitions and the challenges for face tracking and face recognition/verification have been described. We have reviewed some well-known techniques and current advances on face tracking and face recognition/verification. The approaches to face tracking can be divided into five major categories: (1) skin-color based, (2) facial feature based, (3) rapid continuous face detection based, (4) active appearance based, and (5) statistical estimation based. For face recognition/verification approaches, three major approaches have been reviewed: (1) linear subspace analysis, (2) textural analysis, and (3) graph matching techniques. In the following chapters, we will present our proposed algorithms on face tracking and face recognition/verification.

Chapter 3

An Effective Shape-Texture Weighted Algorithm for Multi-view Face Tracking in Videos

In this chapter, an effective face tracking algorithm based on the combination of shape and texture features is presented. As the face patterns to be tracked in consecutive frames are highly correlated, an accurate tracking can be achieved by searching for the shortest weighted feature distance between the face pattern and the possible face candidates. The weights of the shape and texture can be adapted for real-time tracking. This tracking algorithm employs the Adaboost face detection to form an accurate multi-view face tracking system.

3.1. Introduction

The automatic detection and tracking of human faces has many valuable applications, such as human-computer interaction, visual surveillance, access control in special areas, etc. An accurate face tracker will definitely improve the performance of face recognition and other human activity analysis applications that are currently beyond face tracking.

In recent years, many researchers have been motivated to develop efficient face detection algorithms. The most successful one was proposed by Viola and Jones [80]. The AdaBoost algorithm is used to train a cascade classifier for the rapid detection of faces. Some advanced versions [81, 82] of the AdaBoost-trained face detector have been proposed to improve the accuracy and achieve real-time performance. However, these algorithms only work for detecting frontal faces, which cannot fulfill the varying head poses (multi-view faces) tracking requirement.

Multi-view face tracking is critical in many practical applications. Cootes et al. [33] have proposed an adaptive active appearance model, and Batur et al. [83] improved that model. Wiskott et al. [84] used an elastic bunch graph template for multi-view face detection and tracking. These methods use a labeled graph to represent a face. The nodes of the graph represent the texture information about local face regions, while the links of the nodes represent the shape of the face. Tracking is achieved by matching the labeled graph with those candidate positions. The tracking accuracy level is satisfactory, but it is too slow for real-time applications.

Skin-color models [14, 17, 18] have also been widely studied and applied to face tracking [26, 85]. The color distribution of a face candidate is compared with the skin-color model for tracking. Color-model-based methods are computationally efficient, but their performances will be degraded significantly if the lighting conditions are unstable. Objects which are skin color-like will be falsely tracked as a face.

In this chapter, an effective shape-texture weighted algorithm for multi-view face tracking is proposed. Both the edge map and the local binary pattern (LBP) [87] of a face region are used as features for tracking. Having observed that the target face patterns in consecutive frames are highly correlated, a face can be tracked by searching for the least weighted feature distance between a face region and those possible face candidates. The weights of the shape and texture feature distances can adapt to the tracking in an unstable lighting situation. The edge map and the LBP can, to a certain extent, alleviate the effect of changing illumination during tracking. Moreover, skin-color-like objects will not be falsely tracked as a face. The proposed algorithm complements the AdaBoost face detection algorithm to form a multi-view face-tracking system.

The rest of this chapter is organized as follows. Section 3.2 gives an overview of our tracking system. Section 3.3 introduces the features used in the tracking algorithm. Section 3.4 describes the shape-texture weighted face-tracking algorithm.

Section 3.5 presents the experimental results, and the conclusion will be drawn in Section 3.6.

3.2. Tracking system overview

Our face tracking system is composed of two modes of detection. The first mode is to use the AdaBoost face detection algorithm to detect the first appearance of frontal faces. If a face is continuously detected, the features (which will be discussed in Section 3.3) of the detected face will be extracted. If no face is detected, the system will switch to a tracking mode, which tracks faces using our shape-texture weighted algorithm. The status of the face being tracked will then be updated. Figure 3-1 shows the flow of our tracking system.

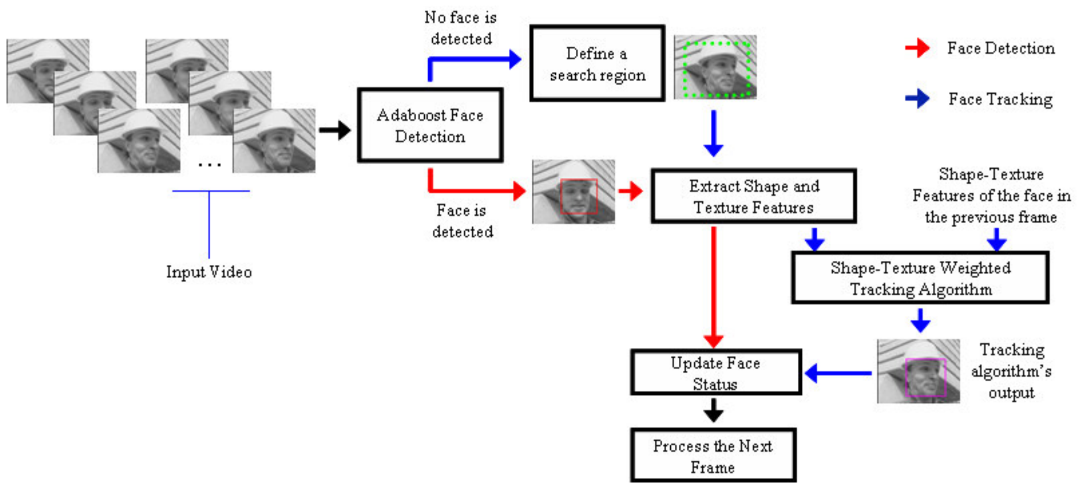

No face is detected
Define a search region
Face Detection
Face Tracking
Adaboost Face Detection
Extract Shape and Texture Features
Shape-Texture Features of the face in the previous frame
Face is detected
Input Video
Shape-Texture Weighted Tracking Algorithm
Update Face Status
Tracking algorithm's output
Process the Next Frame

Figure 3-1 The flow of the tracking system.

3.3. Features for tracking

Face shape and texture are the prominent features for representing and tracking faces. This is because the shape of a face, or a head, is ellipse-like under different poses, and the texture feature selected for representing faces can be insensitive to lighting variations [86]. In addition, the variations of these two features are usually small in successive frames, and are therefore suitable for accurate and efficient tracking. In order to alleviate the lighting effect, our algorithm does not utilize skin color for tracking. Moreover, skin-color-like objects are often incorrectly tracked as faces.

3.3.1. Shape feature

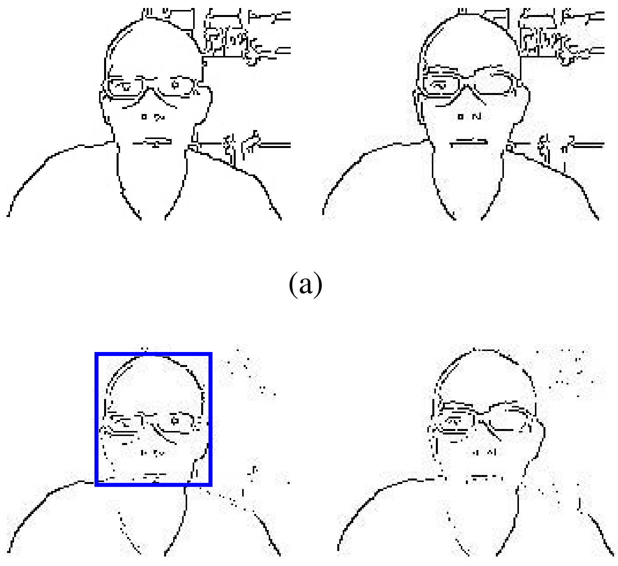
The shape feature used includes the shape of a face and its facial features. It is represented by the binary edge map. We employ the Canny edge detector to detect edges. In order to obtain a good representation of a face shape in a video frame, the background of a video frame should first be removed. Let $E_{t-1}(x, y)$ and $E_t(x, y)$ be the binary edge maps of two successive frames. Then, the binary edge map of the current frame with the background removed, $E'_t(x, y)$, can be computed as follows:

$$E_t^{'}(x,y) = T(E_t(x,y) - E_{t-1}(x,y)), \quad (3\text{-}1)$$

$$\text{where } T(x) = \begin{cases} 0, & \text{if } x \neq 1, \\ 1, & \text{otherwise,} \end{cases}$$

$T(x)$ is a threshold function used to remove the edge points of the objects from frame t−1.

Figure 3-2 shows the binary edge maps of two successive frames with and without background removal. The edge points inside the blue-colored rectangle are used to represent the shape of a face.



(a)

(b)

Figure 3-2 (a) The binary edge maps of two successive frames, and (b) the corresponding edge maps with background removal.

3.3.2. Texture feature

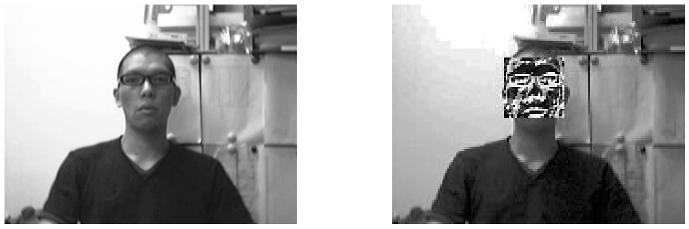
The shape feature alone is not sufficient to track a face in a situation with unstable lighting. This is because edge points used to represent a face shape will change rapidly under poor lighting conditions. This problem can be alleviated, to some extent, by using the texture feature, which is less sensitive to lighting variations.

Most of the texture descriptors, e.g. Gabor wavelets [62] and autocorrelation, are computationally intensive for feature extraction. Recently, the LBP [87] has received much attention in terms of texture analysis. This feature can describe an object's local structural properties effectively. The main characteristics of the LBP are its invariance to monotonic changes in grayscale and its simple computation. This feature has also been applied to face recognition with an outstanding result [79].

The LBP has several variants. Our tracking algorithm adopts the basic LBP operator, in which a pixel is compared to its eight neighborhoods to form a binary number as follows:

$$LBP(x_c, y_c) = \sum_{n=0}^{7} s(g_n - g_c) \cdot 2^n, \qquad \text{where} \quad s(x) = \begin{cases} 1, & x \geq 0, \\ 0, & x < 0, \end{cases} \qquad (3\text{-}2)$$

g_c is the gray value of the center pixel (x_c, y_c) of a local region, and g_n represents the gray values of its neighborhoods, respectively. Figure 3-3 shows an example of the LBP representation of a face. The texture of a face region will then be represented by a histogram of the LBPs within the face region. We refer to this histogram as HLBP().



(a) (b)

Figure 3-3 (a) The original image, and (b) its LBP representation at the face region.

3.4. Shape-texture weighted distance

Having described the shape and texture features, this section will describe our shape-texture weighted algorithm for face tracking.

Videos provide temporal properties. Successive frames are highly correlated in a single shot. Having observed this important property, we should be able to track faces in consecutive frames by searching for the shortest feature distance between a face, *f*, and the possible face candidates, *fc*, within a search region in the next frame.

The feature distance, *d*(*f*, *fc*), can be effectively described using a weighted sum of the shape distance, $d_S()$, and the texture distance, $d_T()$ as follows:

$$d(f, fc) = \alpha \cdot d_S\left(E^{'}(f), E^{'}(fc)\right) + \beta \cdot d_T\left(HLBP(f), HLBP(fc)\right), \qquad (3\text{-}3)$$

where α and β are the respective weights of the two distance measures. The subscripts *S* and *T* represent the shape and the texture features, respectively. The shape distance, $d_S()$, and the texture distance, $d_T()$ are computed as follows:

$$\begin{gathered} d_S\left(E^{'}(f), E^{'}(fc)\right) = 1 - \frac{O\left(E^{'}(f), E^{'}(fc)\right)}{N(E^{'}(f))}, \\ d_T\left(HLBP(f), HLBP(fc)\right) = 1 - corr\left(HLBP(f), HLBP(fc)\right), \end{gathered} \qquad (3\text{-}4)$$

where *N*(*E'*(*f*) is the total number of edge points in the face region; *O*(*E'*(*f*), *E'*(*fc*)) is the number of edge points from the face region and a face candidate which are spatially overlapped. The more similar the shapes of the face and a face candidate is, the larger the value of *O*(). *corr*() computes the correlation of the LBP histograms of the face and the candidate.

The values of α and β can be changed according to the tracking environments. For instance, if the lighting becomes poor, the number of detected edge points will be too small to represent a face shape. In this case, the shape feature is unreliable. Our tracking system can decrease the importance of the shape feature by decreasing

α, and it relies more on the texture feature more by increasing β. Through this adaptation, the multi-view of a face can be tracked under environmental changes.

3.5. Experimental Results

Our face-tracking system has been implemented on a standard PC (Pentium4, 3GHz) with C++ and the OpenCV library [88]. Three different test sequences were used in the experiments; two of them are from the MPEG-7 video databases (Akiyo and foreman), while the last one is a sequence captured using an ordinary webcam. All the tracking results are available on web [89]. Examples of faces being correctly tracked are shown in Figure 3-4

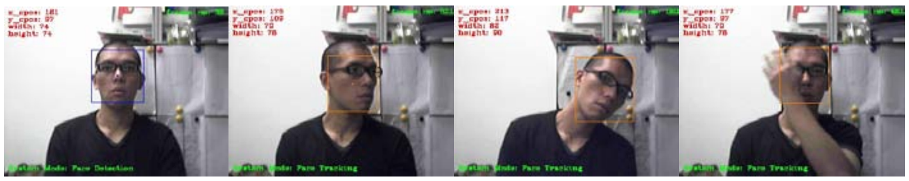
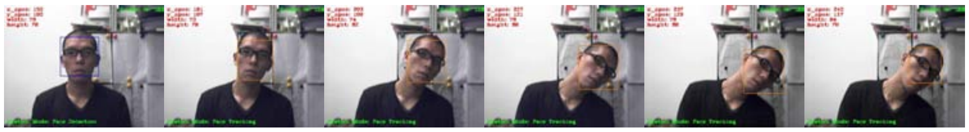


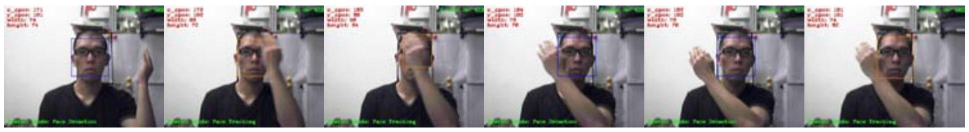
Figure 3-4 Examples of faces that are correctly tracked.

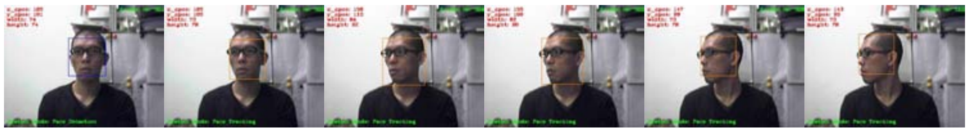
The tracking results based on the two MPEG-7 test sequences are very impressive. Almost all the frames with faces are correctly tracked (Akiyo: 295 out of 297, Foreman: 176 out of 186). This high correctly tracked rate is due to the fact that the motions of the faces in these two sequences are not intensive. Therefore, the features extracted from consecutive frames are highly correlated and can be tracked easily.

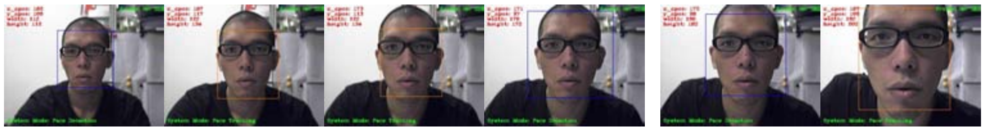
In our self-captured video sequence, we have intentionally included those difficult tracking situations that usually cause face tracking to fail. Figure 3-5 shows some of the frames in our sequence. Our tracking system can track a face when it is tilted (frames 163-190), occluded by the hand (frames 469-480), under out-of-plane rotation (frames 501-510), and under-going scale changes (frames 572-587). Many face-tracking algorithms fail to track faces in these situations [1, 2, 3]. The tracking system runs at 17 frames/sec on average.

We have also evaluated the accuracy of our algorithm by comparing the tracked-face positions to the ground-truth face positions measured manually, as shown in Figure 3-6. The positions of the tracked face are very close to the ground-truth positions, except for the frames from 213 to 264 only. This is because the face in these frames is being under a heavy out-of-plane rotation. When the face rotates too heavy, it is not able to detect the edges of facial features accurately. This degrades the effectiveness of the shape matching in our tracking algorithm, and therefore causes tracking errors.


(Tilted)


(Occluded by hand)


(Out-of-plane rotation)


(Large scale changes)

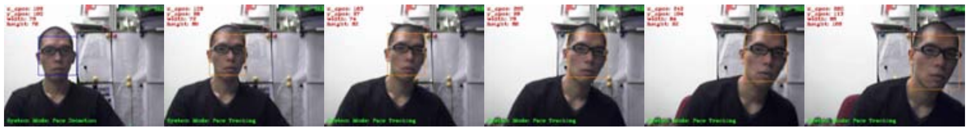
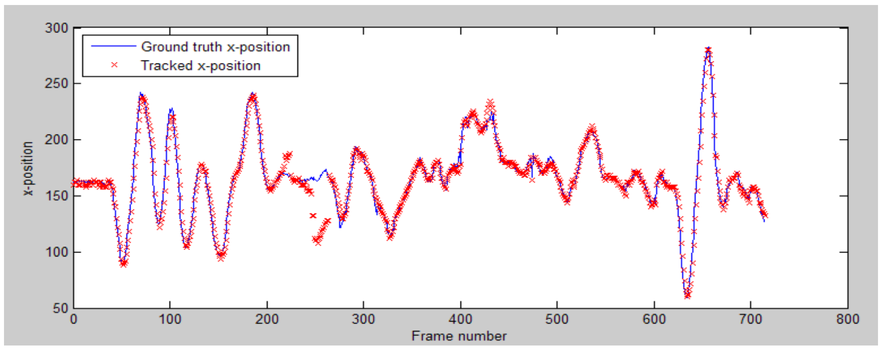
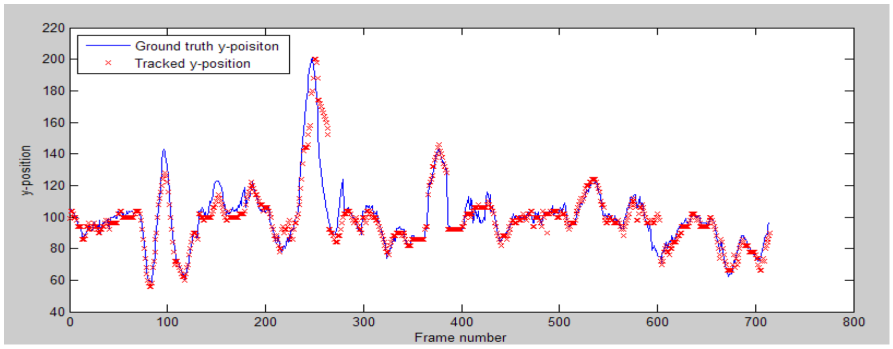

(Motion)

Figure 3-5 Results of multi-view face tracking. These show that our algorithm can track a face when it is tilted (frames 163-190), occluded by hand (frames 469-480), under out-of-plane rotation (frames 501-510), under-going scale changes (frames 572-587), and has motions (frames 641-654).



(a)



(b)

Figure 3-6 The ground-truth positions and the tracked positions of the face: (a) *x*-coordinate, and (b) *y*-coordinate.

3.6. Conclusion

In this chapter, we have proposed an effective face tracking algorithm based on a combination of the shape (edge-map) and the texture (LBP) information. The algorithm utilizes the observation that the face patterns to be tracked in consecutive frames are highly correlated. A face region is tracked by searching for the shortest weighted feature distance between the face pattern and the possible face candidates. The weights of the features can be adapted during tracking, so that an accurate tracking can be achieved. Our proposed algorithm complements the AdaBoost face-detection algorithm to form a multi-view face-tracking system. Experimental results show that our algorithm can track faces in varying poses (tilted or rotated) with 17 fps on average.

In visual surveillance applications, the identities of the tracked faces are recognized for higher level events like private area access-control. Accurate and efficient face recognition algorithms are required to compose a reliable and usable surveillance application. In the next chapter, we will introduce the simplified Gabor wavelets which are highly efficiently and effective on face recognition.

Chapter 4

Simplified Gabor Wavelets for Human Face Recognition

In chapter 3, we have presented a face tracking algorithm which can track multi-view faces in real-time. The succeeding task a surveillance system is to recognize or to verify the identities of the tracked faces. Apart from accuracy, computations involved in this task should also be low such that real time performance can be achieved. In this chapter, we will present a simplified version of the Gabor wavelet which is a highly efficient feature extraction tool. Moreover, the feature extracted by simplified Gabor wavelet has a high representative power in representing human faces. Therefore, it is favorable to use simplified Gabor wavelet for recognizing or verifying the tracked faces.

4.1. Introduction

The Gabor wavelet (GW) [65, 90] is well known for its effectiveness as a feature for image processing and pattern recognition. Its kernels are similar to the response of the two-dimensional receptive field profiles of the mammalian simple cortical cell [60], and exhibit the desirable characteristics of capturing salient visual properties such as spatial localization, orientation selectivity, and spatial frequency selectivity [61]. In the spatial domain, a GW is a complex exponential modulated by a Gaussian function, which is defined as follows [64]:

$$\psi_{\omega,\theta}(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x\cos\theta + y\sin\theta)^2 + (-x\sin\theta + y\cos\theta)^2}{2\sigma^2}} \cdot \left[e^{i(\omega x\cos\theta + \omega y\sin\theta)} - e^{-\frac{\omega\sigma^2}{2}} \right], \tag{4.1}$$

where x, y denote the pixel position in the spatial domain, ω is the radial center frequency of the complex exponential, θ is the orientation of the GW, and σ is the standard deviation of the Gaussian function. By selecting different center frequencies and orientations, we can obtain a family of Gabor kernels from (4.1), which can then be used to extract features from an image.

GWs can effectively abstract local and discriminating features. In textural analysis [91, 92] and image segmentation [93], GW features have achieved outstanding results, while in machine vision, they are found to be effective in object detection [94, 95], recognition [63, 92, 95] and tracking [96-98]. The most successful application of the GWs is for face recognition. In [84, 99-102], GWs are employed for face recognition, and achieve very high performance levels. As the dimension of the feature vectors using GWs is very large, linear subspace methods such as PCA and LDA are used to reduce the dimension. To further improve the performance, kernel methods are also used with the Gabor features. The improvement of both the linear methods and the kernel methods is due to the fact that the GW features are robust to illumination, rotation, and scale [90].

In spite of its superior performance, extracting GW features is highly computational. Given an image $f(x,y)$, GW features are extracted by convolving $f(x, y)$ with $\Psi_{\omega,\theta}(x, y)$ as follows:

$$Y_{\omega,\theta}(x, y) = f(x, y) * \psi_{\omega,\theta}(x, y), \tag{4.2}$$

where * denotes the convolution operator. Usually, convolution is implemented by the fast Fourier transform (FFT) to reduce the computation required for feature extraction. However, the computation required is still very intensive; this, in turn, creates a bottleneck for real-time processing. Hence, an efficient method for extracting Gabor features is important for many practical applications.

The main purpose of this chapter is to introduce a simplified version of Gabor wavelets, whose features can be computed efficiently and can achieve a similar performance level for face recognition. These simplified Gabor wavelets (SGWs) can be viewed as an approximation of the original Gabor wavelets (GWs). A SGW is generated by quantizing a corresponding GW into a certain number of levels. With SGWs, features can be computed efficiently using an integral image. Our proposed

SGWs can replace the GWs for the purpose of real-time processing and applications. The rest of this chapter will describe the structure and the properties of SGWs. Fast algorithms for extracting features by using SGWs will be described, and their corresponding computational complexity will be analyzed. Finally, we will compare the performances of the SGW features and the GW features for face recognition, and discuss the discriminative power of these features.

4.2. Simplified Gabor Wavelets

In this section, we will describe the structure of our proposed SGW. This includes the shape of the SGW, the number of quantization levels, and the methods which determine the respective quantization values.

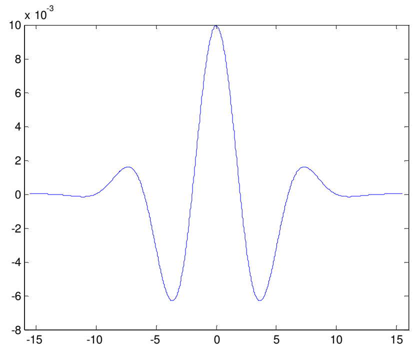
4.2.1. Shape of a SGW

To simplify our discussion, a one-dimensional GW is first considered, whose equation is shown as follows:

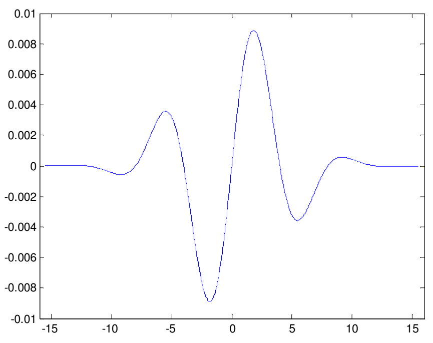
$$\psi_{\omega,\theta}(x) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2}{2\sigma^2}\right) \cdot \exp(i\omega x), \qquad (4.3)$$

where the term $\exp(-\omega\sigma^2/2)$ in (4.1) is ignored. Fig.4-1(a) shows the real part of this

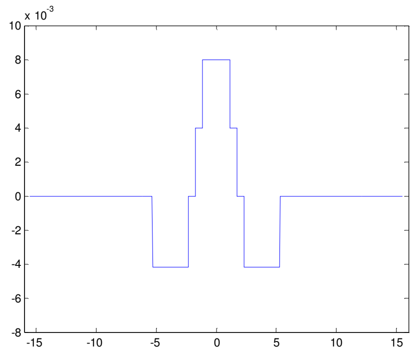
GW, whose values are continuous. To simplify the GW, its values are quantized to a certain number of levels. Fig.4-1(b) illustrates a quantized SGW with 2 quantization levels for the positive values and 1 quantization level for the negative values. Including a level of zero value, the wavelet is said to be quantized into 4 levels. Fig.4-1(c) and Fig.4-1(d) illustrate the corresponding imaginary part of the GW and its simplified version, respectively. The same number of quantization levels is used for the positive and the negative values of the wavelet, because their magnitudes are the same. In Fig.4-1(d), the total number of quantization levels used is 5. For 2-dimensional cases, Fig.4-2(a) and Fig.4-2(d) show the real and imaginary parts of the original 2-D GWs with the gray-level intensities representing the magnitudes of the wavelet. The contours of $\Psi_{\omega,\theta}(x,y)$ whose values equal those quantization levels in Fig.4-1(b) and Fig.4-1(d) are illustrated in Fig.4-2(b) and Fig.4-2(e), respectively. In SGWs, the contours are approximated by rectangles. We have derived two approximation methods for forming the rectangles, as shown in Fig.4-3(a) and Fig.4-3(b), respectively. The first method is to use a rectangle of a size just large enough to contain the corresponding contour of the quantized GW. The second method is to choose a rectangle such that the squared error between the elliptical contour of the GW and the corresponding rectangle is a minimum. To simplify the approximation, we adopt the first method in our algorithm. Fig.4-2(c) and Fig.4-2(f) illustrates the corresponding quantized GWs in Fig.4-2(b) and Fig.4-2(e), respectively, approximated by rectangles.



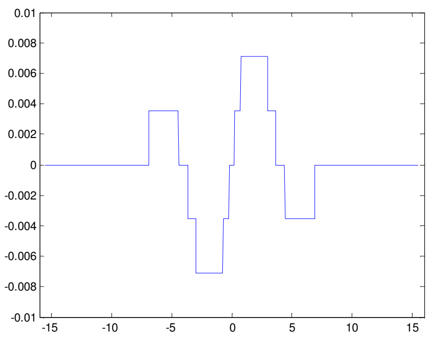
(a)



(b)



(c)



(d)

Fig.4-1. (a) The real part of a one-dimensional GW; (b) the imaginary part of the wavelet; (c) the simplified version of (a); and (d) the simplified version of (c).

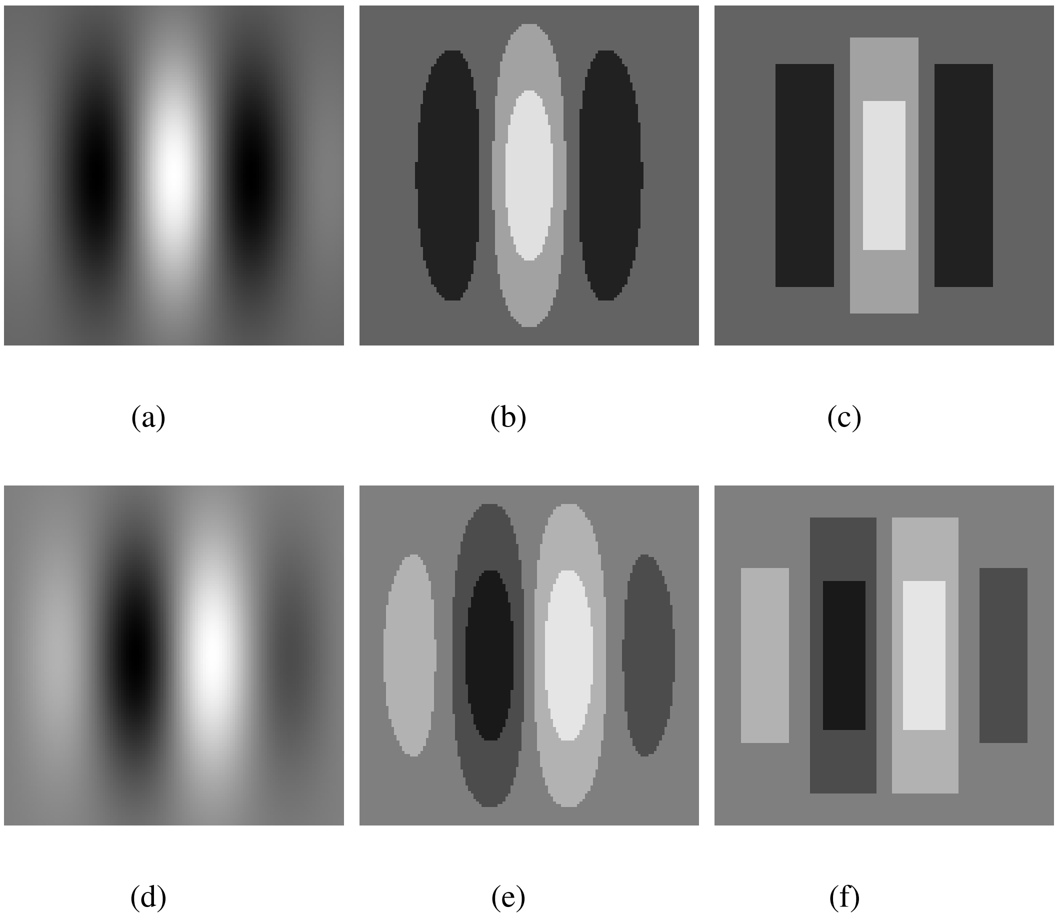


Fig.4-2. (a) The real part of a 2-D GW; (b) the contours of the quantized GW of (a); (c) the approximation of the contours in (b) by rectangles; (d) the imaginary part of the 2-D GW; (e) the contours of the quantized GW of (d); and (f) the approximation of the contours in (e) by rectangles.

4.2.2. Number of Quantization Levels

The number of rectangles in a SGW depends on the number of quantization levels used to quantize the GW. If more quantization levels are employed, the SGWs will be more similar to the original GW, but more computation will then be involved for feature extraction. In other words, there is a trade-off between computation and approximation accuracy. In Section 4.4, the computational analysis of using SGWs and GWs for feature extraction will be performed, and the experiments to evaluate the relative performances of SGWs and GWs with different numbers of quantization levels for face recognition will be conducted in Section 4.5.

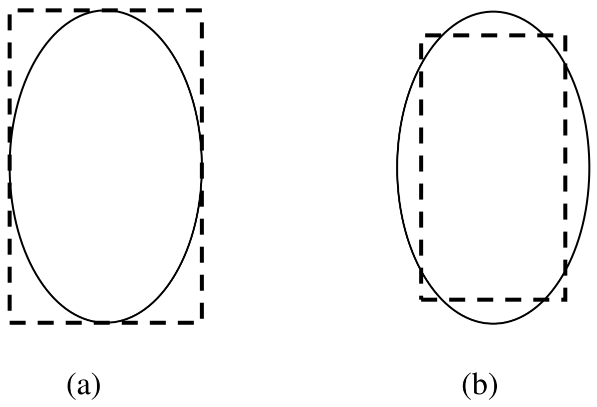


Fig.4-3. (a) Approximation of an elliptical contour using a rectangle just large enough to enclose it; and (b) approximation of the elliptical contour using a rectangle such that the squared error between the rectangle and the contour is a minimum.

4.2.3. Determination of Quantization Levels

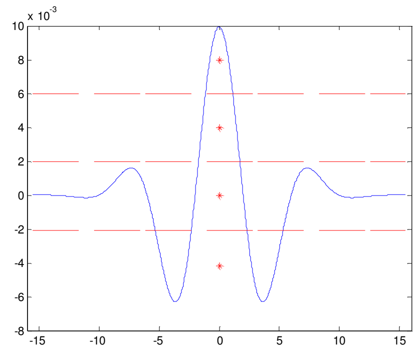
We describe two methods for determining the quantization levels to be used in constructing the SGWs. One of the quantization levels of the SGW is set to zero. Assume that the number of quantization levels for the positive and negative values are n_p and n_n, respectively. Then, the total number of quantization levels is n_p+n_n+1.

Uniform Quantization: In this method, the positive and negative parts of a GW are quantized uniformly according to the corresponding number of levels, as shown in Fig.4-4(a) and Fig.4-4(b). Suppose the most positive and negative values of a GW are A_+ and A_-, respectively, the corresponding quantization levels for positive levels $q_+(k)$ and negative levels $q_-(k)$ are as follows:

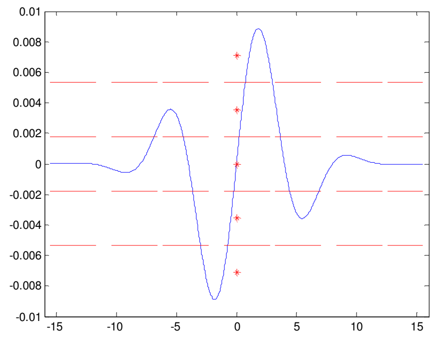
$$q_+(k) = \frac{A_+}{2n_p+1} \cdot 2k, \text{ where } k = 1, \ldots, n_p,$$
$$\text{and} \qquad q_-(k) = \frac{A_-}{2n_n+1} \cdot 2k, \text{ where } k = 1, \ldots, n_n. \tag{4.4}$$

***k*-means Clustering:** As the GWs are not evenly distributed, so the *k*-means algorithm is used to determine the respective optimal quantization levels. The positive values and the negative values are sampled, and are then partitioned into n_p+1 and n_n+1 clusters, respectively. However, after each iteration, the cluster whose centroid is the closest to zero will be set at zero.

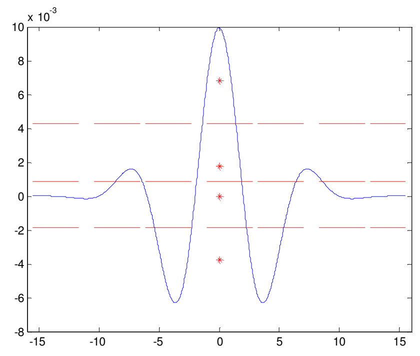
Fig.4-5 illustrates the real part and the imaginary part of a GW and their corresponding simplified versions. These SGWs are then convolved with an image to extract the SGW features at different center frequencies and orientations, which then form a simplified Gabor jet.



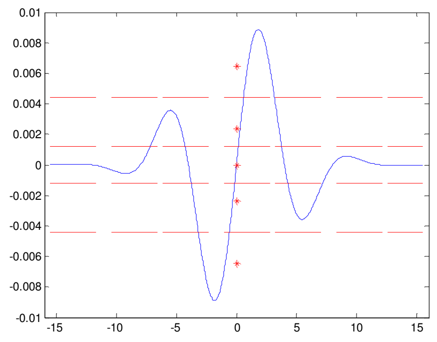
(a)



(b)



(c)



(d)

Fig.4-4. (a) The quantization levels for the real part of a GW based on uniform quantization with n_p = 2 and n_n = 1, (b) the quantization levels for the imaginary part of the GW based on uniform quantization with n_p = 2 and n_n = 2, (c) the quantization levels for the real part of the GW based on *k*-means clustering with n_p = 2 and n_n = 1, and (d) the quantization levels for the imaginary part of the GW based on *k*-means clustering with n_p = 2 and n_n = 2.

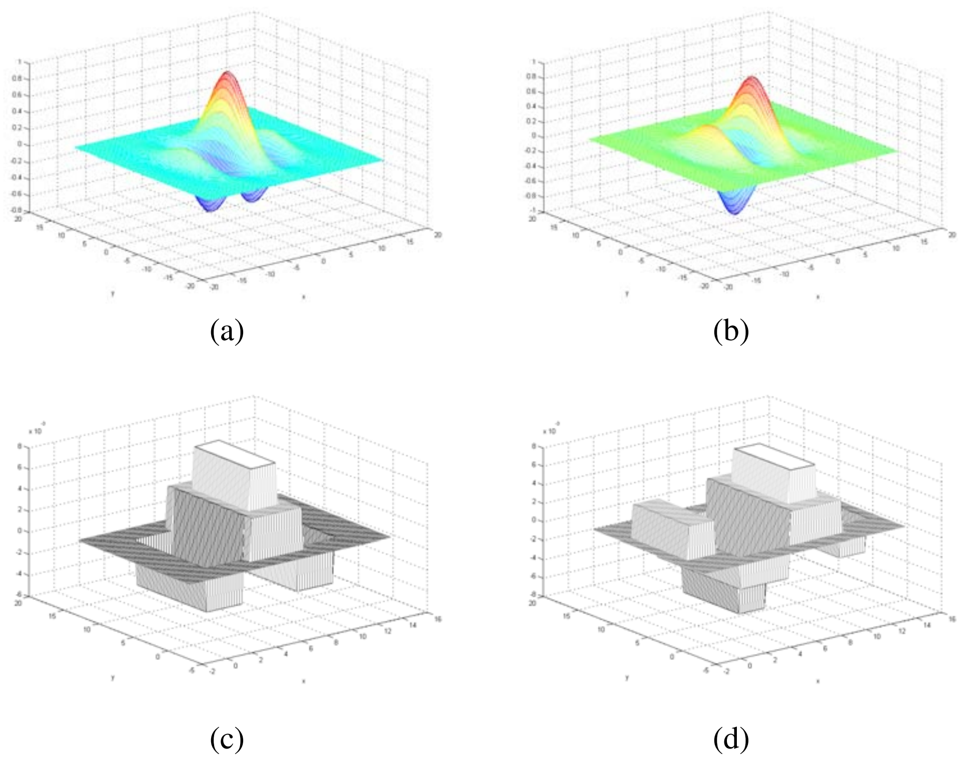


Fig.4-5. The 3-D structures of (a) the real part and (b) the imaginary part of a 2-D GW, and (c) the real part and (d) the imaginary part of the corresponding SGW.

4.2.4. Demeaned SGW (DMSGW)

The term $e^{-\frac{\omega\sigma^2}{2}}$ in (4.1) makes the GW have a zero mean. A SGW formed by quantizing a GW has a non-zero mean; this makes the SGW features sensitive to the lighting conditions of an image. Hence, each of the SGWs has to be demeaned. The mean of a SGW is computed by summing all of its values, and then dividing this sum by the size of the filter. A demeaned simplified Gabor wavelet (DMSGW) is obtained by subtracting the SGW from its mean value. In the rest of this chapter, we will use SGW to refer to a demeaned SGW, and the mean of a SGW is denoted as q_m. The next section will describe an efficient algorithm for computing the SGW features using our proposed SGWs.

4.3. Fast Algorithm for Feature Extraction

The feature extraction process with a SGW is far more efficient than that with a GW. This section will, firstly, describe the extraction of GW features using the FFT, and then devise the fast algorithms for extracting features using the SGWs. The computational complexities of using the GW and the proposed SGW for different orientations will be analyzed in Section 4.4, and their respective runtimes will be measured in Section 4.5. In addition to requiring less computation, the SGW features for any pixel position can be extracted. This is particularly an advantage if the SGW features are used for object tracking. To use the FFT, the size of the image must be a power of 2.

4.3.1. Feature Extraction Using the Original GWs

By selecting different center frequencies and orientations, we can obtain a family of GW kernels from (4.1), which can be used for extracting features from images. Given a gray-level image $f(x,y)$, the convolution of $f(x,y)$ and $\psi_{\omega,\theta}(x,y)$ is given by (4.2). The convolution can be computed efficiently by performing the FFT, then point-by-point multiplications, and finally the inverse FFT (IFFT). By concatenating the convolution output, we can obtain a GW feature vector $\boldsymbol{Y}_{\omega,\theta}$ of dimension $N_w \cdot N_H$.

$$\boldsymbol{Y}_{\omega,\theta} = \left[Y_{\omega,\theta}(0,0), Y_{\omega,\theta}(0,1), \ldots, Y_{\omega,\theta}(0,N_H-1), Y_{\omega,\theta}(1,0), \ldots, Y_{\omega,\theta}(N_W-1,N_H-1)\right]^T, \quad (4.5)$$

where T represents the transpose operation, and N_w and N_H are the width and height of the image, respectively. We consider only the magnitude of the GW representations, which can provide a measure of the local properties of an image [62] and is less sensitive to the lighting conditions [66] (for convenience, we denote it as $\boldsymbol{Y}_{\omega,\theta}$). $\boldsymbol{Y}_{\omega,\theta}$ is normalized to have zero mean and unit variance distribution; and then the Gabor representations with different ω and θ are concatenated to form a high-dimensional vector, as shown in (4.6), and are used for face recognition,

$$\boldsymbol{Y} = \left[\boldsymbol{Y}_{\omega_1,\theta_1}^T \ \boldsymbol{Y}_{\omega_1,\theta_2}^T \ \cdots \ \boldsymbol{Y}_{\omega_1,\theta_n}^T \ \boldsymbol{Y}_{\omega_2,\theta_1}^T \ \cdots \ \boldsymbol{Y}_{\omega_l,\theta_n}^T\right]^T, \quad (4.6)$$

where l and n are the number of center frequencies and the number of orientations used, respectively. Although the FFT is employed so as to reduce the computational complexity, it is still very computationally intensive because a total of $l \times n$ GWs are involved. In addition, the size of the image must be a power of 2, so that the FFT can be used to implement the convolution for saving the computation.

4.3.2. Fast Algorithms for Feature Extraction based on SGWs

In this section, we will present fast algorithms for feature extraction with the SGW at different orientations. Consider a SGW that is convolved with an image *f*(*x*,*y*), and the SGW is shifted to the pixel position (x_c, y_c), as shown in Fig.4-6. The convolution output at this point is given as follows:

$$Y(x_c, y_c) = \sum_{k=1}^{NR_p} q_+(k) S_+(k) + \sum_{k=1}^{NR_n} q_-(k) S_-(k) + q_m S_F, \qquad (4.7)$$

where $S_+(k)$, $S_-(k)$ and S_F are the sum of the gray-level intensities of those pixels covered by the rectangles with quantization values $q_+(k)$, $q_-(k)$, and the rectangular region of the filter, respectively. NR_p and NR_n are the numbers of rectangles with positive quantization values and negative quantization values, respectively. As an example in Fig.4-2(c), $n_p = 2$ and $n_n = 1$, then $NR_p = 2$ and $NR_n = 2$.

$S_+(k)$, $S_-(k)$ and S_F are computed based on the idea of an integral image [80], which can calculate the sum of pixel values within a rectangle efficiently. In addition, a fast algorithm for rectangles rotated by 45° or 135° is also available [81]. Consequently, our SGW considers 4 orientations only, which are 0°, 45°, 90°, and

135°. Denote *ii*(*x*, *y*) as the integral image, then its value at location (*x*, *y*) is the sum of the pixel values above and to the left of (*x*, *y*) inclusive, i.e.

$$ii(x, y) = \sum_{x' \le x, y' \le y} f(x', y'). \tag{4.8}$$

The following pair of recursive equations is used to compute the integral image in one pass over the image:

$$\begin{aligned} s(x, y) &= s(x, y-1) + f(x, y) \text{ and} \\ ii(x, y) &= ii(x-1, y) + s(x, y), \end{aligned} \tag{4.9}$$

where $s(x,-1) = ii(-1, y) = 0$. Let us denote (x_k^1, y_k^1), (x_k^2, y_k^2), (x_k^3, y_k^3), and (x_k^4, y_k^4) as the respective coordinates of the four corners of the rectangle for the k^{th} quantization level. Fig.4-6 shows the four corners for $k = n_p$. Hence, we have

$$S_+(k) = \begin{cases} ii(x_{n_p}^4, y_{n_p}^4) + ii(x_{n_p}^1 - 1, y_{n_p}^1 - 1) - ii(x_{n_p}^2, y_{n_p}^2 - 1) - ii(x_{n_p}^3 - 1, y_{n_p}^3), & k = n_p, \\ ii(x_k^4, y_k^4) + ii(x_k^1 - 1, y_k^1 - 1) - ii(x_k^2, y_k^2 - 1) \ - ii(x_k^3 - 1, y_k^3) - S_+(k+1), & k < n_p. \end{cases} \tag{4.10}$$

Similarly, if $k = n_n$, we have

$$S\_(k) = \begin{cases} ii(x_{n_n}^4, y_{n_n}^4) + ii(x_{n_n}^1 - 1, y_{n_n}^1 - 1) - ii(x_{n_n}^2, y_{n_n}^2 - 1) - ii(x_{n_n}^3 - 1, y_{n_n}^3), & k = n_n, \\ ii(x_k^4, y_k^4) + ii(x_k^1 - 1, y_k^1 - 1) - ii(x_k^2, y_k^2 - 1) \ - ii(x_k^3 - 1, y_k^3) - S\_(k+1), & k < n_n. \end{cases} \tag{4.11}$$

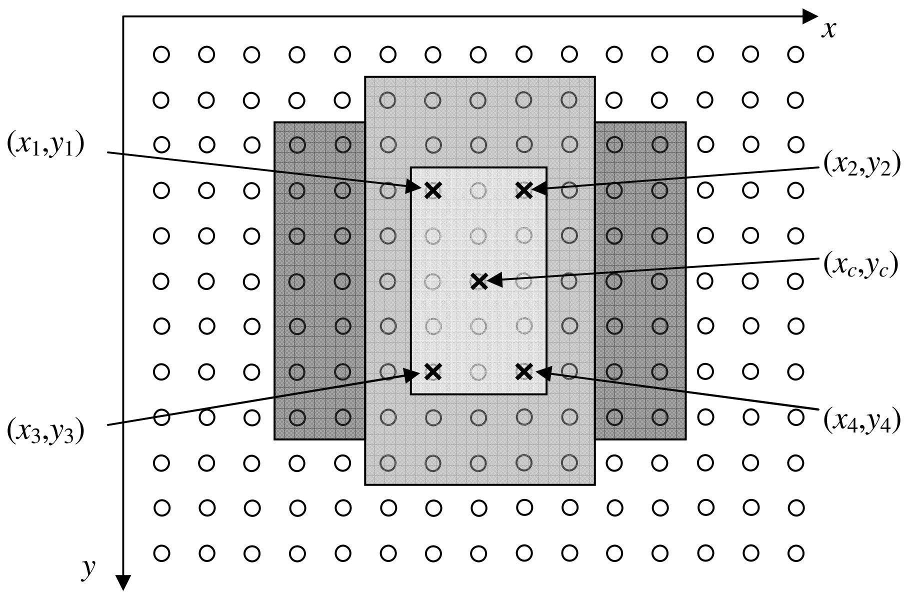


Fig.4-6. Image $f(x,y)$ is convolved with a SGW whose center is shifted to the pixel position (x_c, y_c).

For a rectangle at an orientation of 45°, the rotated integral image, $rii(x, y)$ at location (x, y) contains the sum of the pixel values of the rectangle rotated by 45°, with the rightmost corner at (x, y) and extended to the boundaries of the image, as shown in Fig.4-7, i.e.

$$rii(x, y) = \sum_{x' \le x, x' \le x - |y - y'|} f(x', y'). \tag{4.12}$$

Two passes over an image are required to compute the rotated integral image. The first pass is performed from left to right and top to bottom as follows:

$$rii(x, y) = rii(x-1, y-1) + rii(x-1, y) + f(x, y) - rii(x-2, y-1), \tag{4.13}$$

where $rii(x,-1) = rii(-1, y) = rii(-2, y) = 0$. The second pass is performed from right to left and bottom to top as follows:

$$rii(x, y) = rii(x, y) + rii(x-1, y+1) - rii(x-2, y). \tag{4.14}$$

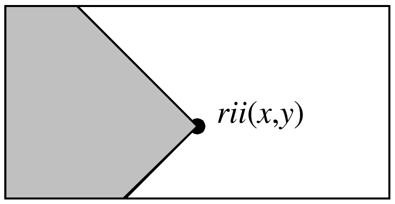


Fig.4-7. Rotated integral image *rii*(*x*,*y*), which is equal to the sum of pixel intensities inside the shaded and rotated rectangle.

Let us denote (x_k, y_k, w_k, h_k) as the *x*-coordinate, *y*-coordinate, width, and height, respectively, of the rotated rectangle in Fig.4-8. Then, we have

$$S_+(k) = \begin{cases} rii(x_{n_p} + w_{n_p} - 1, y_{n_p} + w_{n_p} - 1) + rii(x_{n_p} - h_{n_p} - 1, y_{n_p} + h_{n_p} - 1) \\ -rii(x_{n_p} - 1, y_{n_p} - 1) - rii(x_{n_p} + w_{n_p} - h_{n_p} - 1, y_{n_p} + w_{n_p} + h_{n_p} - 1), & k = n_p, \\ rii(x_k + w_k - 1, y_k + w_k - 1) + rii(x_k - h_k - 1, y_k + h_k - 1) \\ -rii(x_k - 1, y_k - 1) - rii(x_k + w_k - h_k - 1, y_k + w_k + h_k - 1) - S_+(k+1), & k < n_p. \end{cases} \tag{4.15}$$

Similar formulation can be derived for the computation of $S_-(k)$, as well as for the case when a rectangle is at an orientation of 135°.

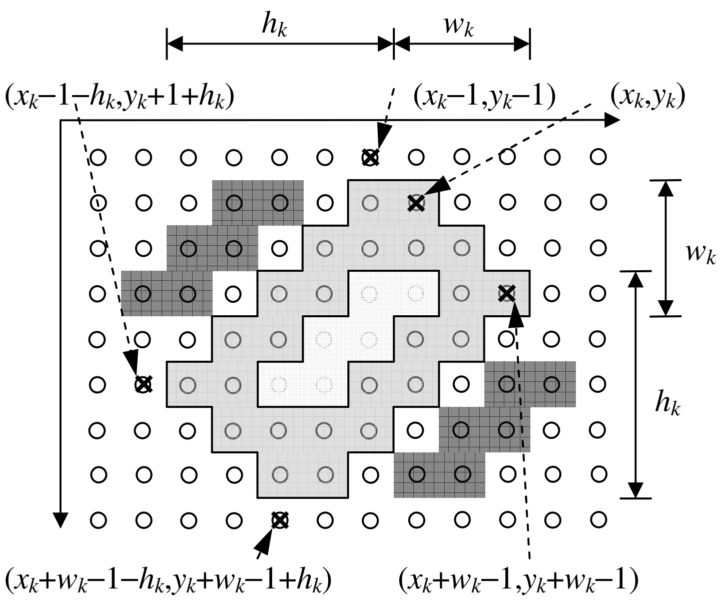


Fig.4-8. The computation scheme for a rotated rectangle.

To further speed up feature extraction, let us denote $RS(k)$ as the sum of pixel intensities inside a rectangle with the coordinates of its four corners being (x_k^1, y_k^1), (x_k^2, y_k^2), (x_k^3, y_k^3), and (x_k^4, y_k^4), respectively. Thus

$$RS(k) = ii\left(x_k^4, y_k^4\right) + ii\left(x_k^1 - 1, y_k^1 - 1\right) - ii\left(x_k^2, y_k^2 - 1\right) - ii\left(x_k^3 - 1, y_k^3\right). \quad (4.16)$$

Let $RS_+(k)$, $RS_-(k)$ and RS_F be the sum of the gray-level intensities of those pixels inside the rectangles with quantization values $q_+(k)$, $q_-(k)$ and the rectangular region covered by the SGW, respectively. Fig.4-9 shows the real part of a SGW with $n_n = n_p = 2$ or $NR_n = 4$ and $NR_p = 2$. Then, the convolution output at the pixel position (x_c, y_c) is:

$$
\begin{aligned}
Y(x_c, y_c) &= \sum_{k=1}^{NR_n}\left[q_-(k)\cdot S_-(k)\right] + \sum_{k=1}^{NR_p}\left[q_+(k)\cdot S_+(k)\right] + q_m\cdot S_F \\
&= q_-(1)\cdot S_-(1) + q_-(2)\cdot S_-(2) + q_-(3)\cdot S_-(3) + q_-(4)\cdot S_-(4) + q_+(1)\cdot S_+(1) + q_+(2)\cdot S_+(2) + q_m\cdot S_F \\
&= q_-(1)\cdot\left[RS_-(1) - RS_-(2)\right] + q_-(2)\cdot RS_-(2) \\
&\quad + q_-(3)\cdot\left[RS_-(3) - RS_-(4)\right] + q_-(4)\cdot RS_-(4) \\
&\quad + q_+(1)\cdot\left[RS_+(1) - RS_+(2)\right] + q_+(2)\cdot RS_+(2) \\
&\quad + q_m\cdot\left(RS_F - RS_-(1) - RS_-(3) - RS_+(1)\right) \\
&= \left[q_-(1) - q_m\right]\cdot RS_-(1) + \left[q_-(2) - q_-(1)\right]\cdot RS_-(2) \\
&\quad + \left[q_-(3) - q_m\right]\cdot RS_-(3) + \left[q_-(4) - q_-(3)\right]\cdot RS_-(4) \\
&\quad + \left[q_+(1) - q_m\right]\cdot RS_+(1) + \left[q_+(2) - q_+(1)\right]\cdot RS_+(2) \\
&\quad + q_m\cdot RS_F \\
&= \sum_{k=1}^{NR_n}\left[m_-(k)\cdot RS_-(k)\right] + \sum_{k=1}^{NR_p}\left[m_+(k)\cdot RS_+(k)\right] + m_F RS_F
\end{aligned}
\tag{4.17}
$$

where $m_+(k) = \begin{cases} q_+(k) - q_m, & k\text{, refer to the outermost rectangles} \\ q_+(k) - q_+(k-1), & k\text{, refer to the inner rectangles.} \end{cases}$

$$m_-(k) = \begin{cases} q_-(k) - q_m, & k\text{, refer to the outermost rectangles,} \\ q_-(k) - q_-(k-1), & k\text{, refer to the inner rectangles.} \end{cases}$$

$$m_F = q_m.$$

Hence, instead of using $q(k)$ directly, the $m(k)$s are employed in the computation.

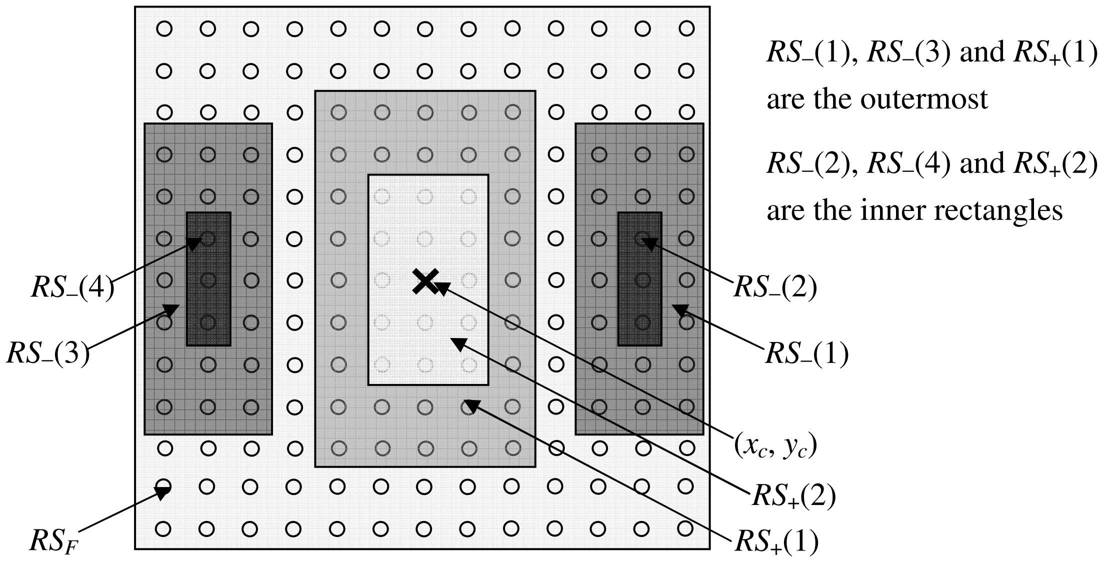
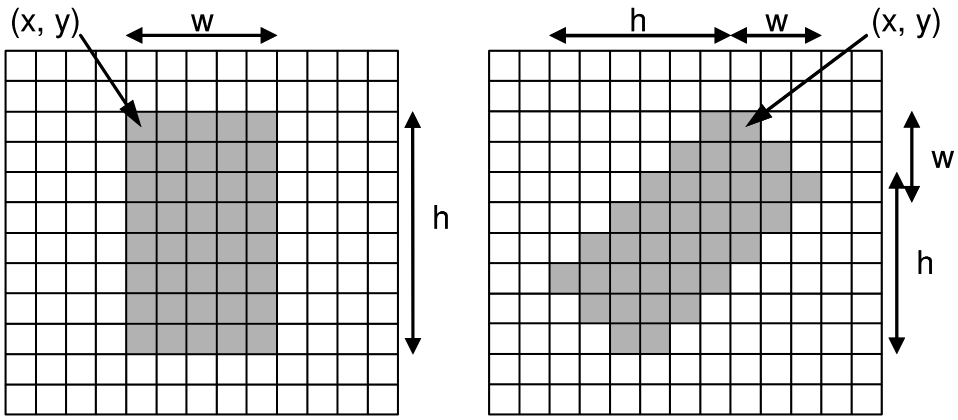

$RS_{-}(1)$, $RS_{-}(3)$ and $RS_{+}(1)$ are the outermost
$RS_{-}(2)$, $RS_{-}(4)$ and $RS_{+}(2)$ are the inner rectangles
$RS_{-}(4)$
$RS_{-}(3)$
$RS_{-}(2)$
$RS_{-}(1)$
(x_c, y_c)
$RS_{+}(2)$
$RS_{+}(1)$
RS_F

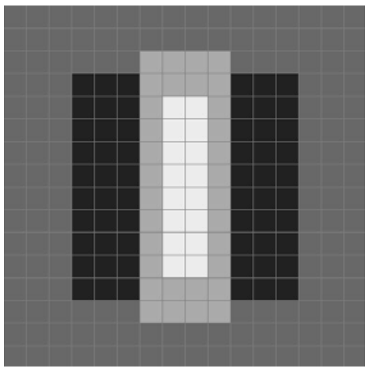
Fig.4-9. The rectangles in a SGW.

For implementation, a number of parameters are required to describe a rectangle, which govern the computation of $RS_{+}(k)$, $RS_{-}(k)$ and RS_F. These parameters include the orientation, $m_{+}(k)$, $m_{-}(k)$, m_F, (x, y) coordinates, and the width and height of each rectangle. Fig.4-10 defines the (x, y) coordinates, and the width and height of an upright rectangle and a rotated rectangle, which is similar to that in [80] and [81]. Fig.4-11(a) shows a SGW, while Fig.4-11(b) describes the parameters of this SGW.



(a) (b)

Fig.4-10. Definition of the (*x*, *y*)-coordinates, width and height of a rectangle in a SGW at an orientation of (a) 0°, and (b) 45°.



```
16 16// width & height of the SGW
0.785398    // scale of the SGW
0           // orientation of the SGW (0 = 0°, 1 = 45°, 2 = 90°, 3 = 135°)
0           // real or imaginary part (0=real, 1=imaginary)
-0.000082   // mean of SGW
2           // number of quantization levels in the positive side
1           // number of quantization levels in the negative side
-0.004159 0.000082 0.004032 0.007982   // quantization values
4           // number of rectangles
// Quantization value      m(k)       x   y   w   h
      -0.004159         -0.004241    3   3   3   10
      -0.004159         -0.004241   10   3   3   10
       0.004032          0.003950    6   2   4   12
       0.007982          0.003950    7   4   2    8
```

(a) (b)

Fig.4-11. (a) A SGW and (b) the corresponding parameters of this wavelet.

4.4. Computational Analysis for Feature Extraction

In this section, we will analyze and compare the computations required for extracting features using GW and SGW, respectively. Within our context, computations refer to the number of real additions and real multiplications required for extracting the GW features of an image using a GW. In our analysis, we assume that the image size is a power of 2 so that the FFT can be applied when using GWs for faster feature extraction. Actually, for the use of SGW, the image may be any size and the features at any individual pixel position can be computed efficiently.

4.4.1. Feature Extraction with GW

Given an $N \times N$ image, f, and a GW, g, with an arbitrary scale and orientation, GW features can be extracted by convolution, i.e. $f * g$. The convolution is implemented by using the FFT, then point-by-point multiplications, and finally the IFFT. In our analysis, we assume that the FFTs of the GWs are pre-computed.

The FFT of an $N \times N$ image requires $N^2 \log_2 N^2$ complex additions and $0.5N^2 \log_2 N^2$ complex multiplications. The IFFT requires the same amount of computation as the FFT. The point-by-point multiplications involve N^2 complex multiplications. Performing one complex addition requires 2 real additions, while one complex multiplication requires 2 real additions and 4 real multiplications. Therefore, feature extraction based on a GW requires a total of $2N^2 \log_2 N^2$ complex additions and $N^2 log_2 N^2 + N^2$ complex multiplications; this is equivalent to a total of $6N^2 \log_2 N^2 + 2N^2$

real additions and $4N^2\log_2 N^2+4N^2$ real multiplications.

4.4.2. Feature Extraction with SGW

As described in Section 4.3, fast algorithms are available for extracting SGW features using SGWs at 4 different orientations. These fast algorithms are based on the use of integral images and rotated integral images, such that features at any position in an image can be computed efficiently. Our algorithm will first perform a table look-up operation to compute the sum of pixel values for the respective rectangles of the SGW. Then, each of the pixel sums is multiplied by the quantization value of the corresponding rectangle. The sum of these products is the SGW feature at a given pixel position.

The computation for extracting features using a SGW at orientation 0° or 90° (a non-rotated SGW) is different from that when using a SGW at orientation 45° or 135° (a rotated SGW). This is because, for feature extraction, the non-rotated SGW uses the integral image, while the rotated SGW uses the rotated integral image. The computations involved are different for different orientations. Consequently, we separate our analysis into two parts: the non-rotated SGW (NR-SGW) and the rotated SGW (R-SGW).

4.4.2.1. The non-rotated SGW (NR-SGW)

Before extracting features using a NR-SGW, the integral image must be computed. From (4.9), 4 real additions are required to compute an entry of the integral image. For an image of size $N \times N$, $4N^2$ real additions are required for the whole integral image. Suppose that the SGW contains a total of N^t_{rect} rectangles. From (4.16) and (4.17), $3N^t_{rect}$ real additions are required to compute all the rectangular pixel sums, and N^t_{rect} real multiplications and $(N^t_{rect}-1)$ real additions are required to compute the SGW feature for a given pixel position. The coordinates of the four corners in (4.16) can be generated by a table look-up operation. Consequently, a total of $4N^2N^t_{rect} + 3N^2$ real additions and $N^2N^t_{rect}$ real multiplications are required to extract the SGW feature.

4.4.2.2. The rotated SGW (R-SGW)

The rotated integral image is computed for extracting feature with a rotated SGW. From (4.13) and (4.14), 9 real additions are required to compute an entry in the rotated integral image. For an image of size $N \times N$, $9N^2$ real additions are required to compute the whole rotated integral image.

Feature extraction with a R-SGW is computed in a similar way to that with the

NR-SGW. The rotated pixel sums covered by the rotated rectangles of the R-SGW are computed. Suppose that the R-SGW contains N^t_{rect} rectangles, then from (4.16) and (4.17), $3N^t_{rect}$ real additions are required to compute all the rotated rectangular pixel sums, and N^t_{rect} real multiplications and $(N^t_{rect}-1)$ real additions are required to compute the R-SGW feature at a pixel position. Therefore, a total of $4N^2N^t_{rect}+8N^2$ real additions and $N^2N^t_{rect}$ real multiplications is required to extract the feature from the whole image. Table 4-1 shows the summarization of the computational complexities of feature extraction using GW and SGW.

		+	×
GW	**A**: Compute FFT of image (floating point operations)	$3N^2\log_2 N^2$	$2N^2\log_2 N^2$
	B: Compute feature by multiplying FFT image and FFT GW (floating point operations)	$2N^2$	$4N^2$
	C: Compute IFFT of feature (floating point operations)	$3N^2\log_2 N^2$	$2N^2\log_2 N^2$
	Total	$\mathbf{6N^2\log_2 N^2+2N^2}$	$\mathbf{4N^2\log_2 N^2+4N^2}$
NR-SGW	**D**: Compute SAT (integer additions)	$4N^2$	0
	E: Compute rectangular pixel sums (integer additions)	$3N^2N^t_{rect}$	0
	F: Compute feature by multiplying rectangular pixel sums and quantization value of rectangles (floating point multiplications)	0	$N^t_{rect}N^2$
	G: Add all products in F (floating point additions)	$N^2(N^t_{rect}-1)$	0
	Total	$\mathbf{4N^2N^t_{rect} + 3N^2}$	$\mathbf{N^2N^t_{rect}}$
R-SGW	**H**: Compute RSAT (integer additions)	$9N^2$	0
	I: Compute rotated rectangular pixel sums (integer additions)	$3N^2(N^t_{rect}-1)$	0
	J: Compute SGW background pixel sums (integer additions)	$3N^2$	0
	K: Compute feature by multiplying rectangular pixel sums and quantization value of rectangles (floating point multiplications)	0	$N^t_{rect}N^2$
	L: Add all products in K (floating point additions)	$N^2(N^t_{rect}-1)$	0
	Total	$\mathbf{8N^2+4N^2N^t_{rect}}$	$\mathbf{N^t_{rect}N^2}$

Table 4-1(a) Computational complexities of feature extraction using GW and SGW.

*1: Image dimension = $N\times N$, where N must be to the power of 2 in order to speed up the GW feature extraction process.

*2: N^t_{rect} is the total number of rectangles in a SGW, which is listed in Table 4-3.

To illustrate the computational advantage of using SGWs over GWs, Table 4-2 tabulates the respective numbers of arithmetic operations required for extracting GW features and SGW features, and Table 4-3 shows the respective numbers of rectangles used to represent the different level quantized SGWs. It is found that about 2.85 times and 2.44 times the arithmetic operations are saved if a 3-level quantized NR-SGW and R-SGW, respectively, are used. Moreover, the number of multiplications required for SGW feature extraction is reduced significantly when compared to that for GW. In general, the runtime required for multiplication is longer than that for addition. Furthermore, the runtime consumed by a floating point arithmetic operation is longer than that for an integer arithmetic operation. Feature extraction with SGW involves fewer floating point operations than does GW, therefore, the runtime for SGW feature extraction should in practice have a speed-up rate higher than 2.85 times.

			+	×	Total
GW			303,104	212,992	516,096
NR-SGW	**No. of quantization levels used**	**3 levels**	147,844	32,768	180,612
		5 levels	229,764	53,248	283,012
		7 levels	344,452	81,920	426,372
R-SGW	**No. of quantization levels used**	**3 levels**	179,049	32,768	211,817
		5 levels	260,969	53,248	314,217
		7 levels	375,657	81,920	457,577

Table 4-2 Number of arithmetic operations required for extracting GW features from a 64×64 pixel image using a GW and a SGW with different numbers of quantization levels.

Number of quantization levels ($n_n + n_p + 1$)	**Number of rectangles in the *real part* of a SGW (N^r_{rect})**	**Number of rectangles in the *imaginary part* of a SGW (N^i_{rect})**	**Total number of rectangles in a SGW, including the background of SGW (N^t_{rect}) = (N^r_{rect}) + (N^i_{rect}) + 1**
3 ($n_n = 1, n_p = 1$)	$(N^r_{rect}) = (n_n \times 2 + n_p) = 3$	$(N^i_{rect}) = ((n_n + 1) + (n_p + 1)) = 4$	8
5 ($n_n = 2, n_p = 2$)	$(N^r_{rect}) = (n_n \times 2 + n_p) = 6$	$(N^i_{rect}) = ((n_n + 1) + (n_p + 1)) = 6$	13
7 ($n_n = 3, n_p = 3$)	$(N^r_{rect}) = (n_n \times 2 + (n_p + 2)) = 11$	$(N^i_{rect}) = ((n_n + 1) + (n_p + 1)) = 8$	20

Table 4-3 The number of rectangles of a SGW with different numbers of quantization levels, where n_n and n_p are the number of negative quantization levels and the number of negative quantization levels in a SGW.

4.5. Experimental Results

In this section, we will evaluate the respective performances of the proposed

SGWs with different numbers of quantization levels. The two different methods for determining the quantization values of a SGW will also be evaluated. Then, we will compare the performances of the SGW features and the GW features for face recognition. Finally, we will compare the runtimes for extracting the SGW features and the GW features.

4.5.1. Face Databases and Experimental Set-up

The standard face databases used include the Yale database, YaleB database and AR database. The number of distinct subjects, the number of testing images and the characteristics of the databases are tabulated in Table 4-4.

Databases	**Characteristics**	**Number of distinct subjects**	**Number of images**	**Number of images per subject**
Yale	Variations in facial expression	15	150	10
YaleB	Large variations in lighting	10	640	64
AR	Variations in facial expression	121	605	5
Overall		146	1395	

Table 4-4 The number of distinct subjects, the number of images and the characteristics of the face databases.

For face recognition, a frontal-view image of each subject in the databases is selected as a training image, and the remaining faces are used for testing. Each face image is normalized to a size of 64×64, and is aligned based on the position of the two eyes for matching. In order to enhance the global contrast of the images and reduce the effect of uneven illuminations, histogram equalization is applied to all images. As described in Section 4.2.3, we have two different ways to determine the quantization levels of SGWs. The SGWs derived based on uniform quantization and on *k*-means clustering are denoted as SGW1 and SGW2, respectively. The GW and SGW adopt 3 to 5 center frequencies with 4 orientations. In other words, 12 to 20 GWs and SGWs are used for feature extraction. The extracted features with each Gabor filter are concatenated to form a feature vector, which is then normalized to have zero mean and unit variance. These Gabor jets are then used directly to compute the distance between two images, pixel position by pixel position.

4.5.2. Relative Performances of SGW1 and SGW2

Table 4-5 shows the recognition rates based on SGW1 and SGW2 with different numbers of quantization levels for the different databases. For the real part of a GW, the dynamic range of the positive values is usually larger than that of the negative values. Hence, n_p should be set larger than n_n. However, for the imaginary part of the GW, the dynamic ranges of the positive values and negative values are the same, so n_p should be equal to n_n. To simplify the experiment, we set n_p equal to n_n for both the real and imaginary parts. Consequently, including the level for zero, the numbers of quantization levels considered in the experiments are 3, 5, and 7.

From Table 4-5, the relative performances of SGW1 and SGW2 are very similar. The face recognition rate increases slightly with an increase in the number of quantization levels. If more quantization levels are used, the SGW can better approximate the GW, and its performance will then be closer to that of the GW. However, using the SGW with more quantization levels will involve more computations.

We have also investigated the effect of using more scales of the SGW with a fixed number of quantization levels. Experimental results show that using 4 scales of SGW results in the best recognition rate. Theoretically, using 5 scales should produce a better performance than using 4 scales only. However, the error in representing a GW is large when its scale is large. As discussed in Sections 4.2.1 and 4.3, in order to utilize fast algorithms to extract the features, the SGWs must be approximated with rectangles after quantizing the GWs. This constraint will alter the effective regions of the SGWs. Fig.4-12 shows a GW, a GW after quantization, and a SGW approximated by rectangles. We can observe that part of the effective regions of the quantized GW is removed or extended in order to form a rectangular shape, which will therefore introduce quantization errors. As the size of a SGW is 16×16 pixels only, large rectangles cannot be formed. As a result, the quantization errors in forming the rectangles are significant for those large-scale SGWs. On the contrary, for small-scale SGWs, small rectangles will be formed without requiring much of the original shape of the quantized GW to be changed. This will introduce fewer quantization errors. For SGWs with 5 scales, the approximation of some of the large-scale GWs is not accurate. This, in turn, will degrade the overall recognition performance.

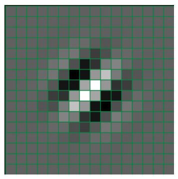
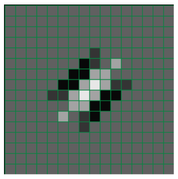
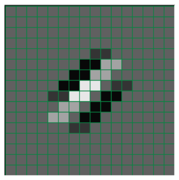
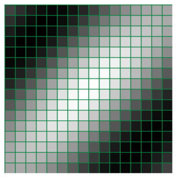
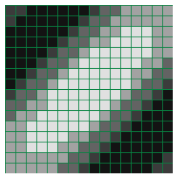
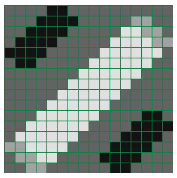
	GW	**Quantized GW**	**SGW with rectangular shape**
Small scale (ω=π/2)			
Large scale (ω=π/8)			

Fig.4-12. The first column is the GW, the second column is the quantized form of GW, and the third column is the SGW with a rectangular shape. The top row is the small-scale (ω=π/2) GW being quantized and formed into a rectangular-shaped SGW. The bottom row is the large-scale (ω=π/8) GW being quantized and formed into a rectangular-shaped SGW.

	Different combinations of scales-orientations-quantization levels	**Recognition rate**		
		Yale	**YaleB**	**AR**
SGW1	**5** scales **4** orientations **3** quantization levels	82.00%	90.16%	92.40%
	5 scales **4** orientations **5** quantization levels	84.67%	92.19%	92.40%
	5 scales **4** orientations **7** quantization levels	82.67%	92.66%	92.89%
	4 scales **4** orientations **3** quantization levels	82.67%	93.13%	92.07%
	4 scales **4** orientations **5** quantization levels	82.00%	94.69%	91.74%
	4 scales **4** orientations **7** quantization levels	82.67%	94.84%	92.07%
	3 scales **4** orientations **3** quantization levels	82.67%	92.97%	92.23%
	3 scales **4** orientations **5** quantization levels	82.67%	93.91%	92.23%
	3 scales **4** orientations **7** quantization levels	83.33%	94.69%	92.23%
SGW2	**5** scales **4** orientations **3** quantization levels	82.67%	91.09%	91.90%
	5 scales **4** orientations **5** quantization levels	82.67%	92.50%	92.23%
	5 scales **4** orientations **7** quantization levels	82.67%	92.50%	92.56%
	4 scales **4** orientations **3** quantization levels	82.67%	93.91%	91.74%
	4 scales **4** orientations **5** quantization levels	82.67%	95.00%	91.74%
	4 scales **4** orientations **7** quantization levels	83.33%	95.47%	91.90%
	3 scales **4** orientations **3** quantization levels	82.00%	93.59%	92.40%
	3 scales **4** orientations **5** quantization levels	82.67%	95.00%	91.90%
	3 scales **4** orientations **7** quantization levels	83.33%	94.53%	92.23%
GW	**5** scales **4** orientations	80.00%	94.69%	92.73%
	4 scales **4** orientations	78.00%	97.50%	92.23%
	3 scales **4** orientations	74.00%	99.22%	89.92%

Table 4-5 Face recognition performances of SGW1, SGW2 and GW with different scales, orientations, and quantization levels (SGW1: uniformly quantized SGWs, SGW2: *k*-means quantized SGWs, GW: Gabor wavelets).

4.5.3. Performances of the SGW and the GW

The use of the SGW can save a lot of computation when compared to the GW, while maintaining a comparable performance to the GW. Table 4-5 tabulates the performances using SGW1, SGW2 and GW for face recognition with different numbers of center frequencies and orientations. The face recognition results show that, with the same number of center frequencies and orientations, the relative performances of the SGW and the GW are very similar; and in some cases, the SGW outperforms the GW. Actually, the center frequency of a SGW should be very similar to its original GW. A SGW is a quantized version of its GW; their rates of variation should be maintained. Hence, in the frequency domain, the center frequencies of the SGW and the GW should be very close, while the shape of their spectra will differ. The features extracted by a GW and the corresponding SGW should be similar. Fig.4-13 shows the magnitudes of the GW features and the SGW features at 3 scales and 4 orientations. We can observe that the general shapes of SGW features and GW features are similar; however, SGWs introduce a directional pattern on the features, which is a drawback with quantizing GWs coefficients to a certain number of levels.

From Table 4-5, the performance of the SGW is slightly worse than that of the

GW with the YaleB database, while the SGW has a very similar performance to the GW with the other databases. The reason for this is that the images in the YaleB database have a wide variation in lighting conditions. As we discussed in Section 4.2.4, a SGW is the quantized version of a GW, so the values of the SGWs are changed in step. Therefore, when two images of the same person have a significant difference in lighting conditions, the features extracted by GWs and SGWs will also differ greatly. Hence, the performance of the SGW will be degraded in this circumstance.

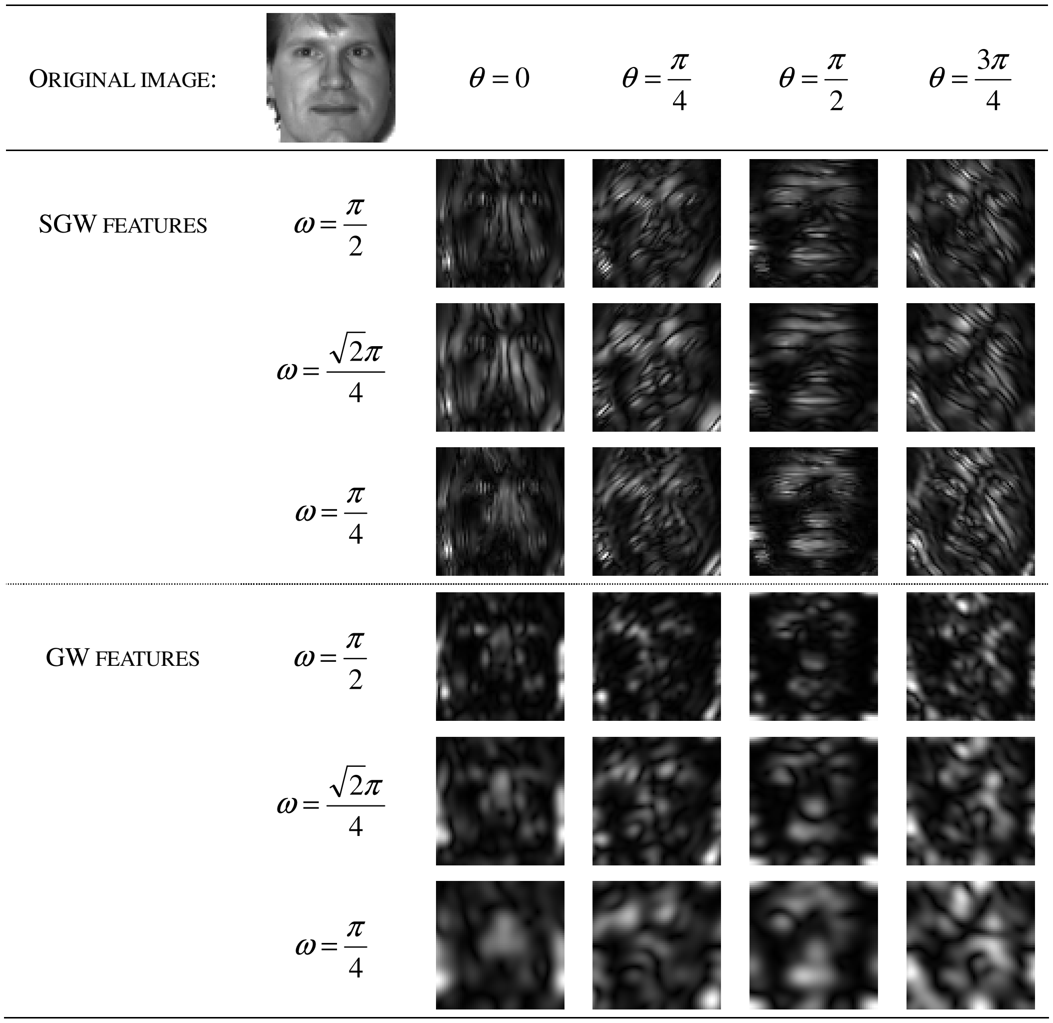


Fig.4-13. The magnitudes of SGW features and GW features at 3 scales and 4 orientations.

4.5.4. Runtimes for Feature Extraction with the SGW and the GW

In our experiments, we also measure the runtimes required for feature extraction using the SGW and the GW. One of the images from the Yale database was used, and the size of each face region is 64×64 pixels. Feature extractions using the SGW and the GW at 5 scales and 4 orientations were performed for 100 times, and the respective total runtimes were measured. Table 4-6 tabulates the runtimes for extracting features using the SGW and the GW. With a 3-level quantized SGW, the speed-up rate for feature extraction is 4.39 times that of a GW. The reduction in runtime will decrease if the SGW uses more quantization levels. For SGWs with 5 and 7 quantization levels, the runtimes for feature extraction are 27.5ms and 37.97ms, respectively, and the corresponding speed-up rates are 2.57 and 1.86, respectively.

To conclude our experiment results, the performance of the SGW is comparable to that of the GW, while the computation required by the SGW is significantly less than that for the GW. GWs can extract features which are discriminative and useful for many applications, but they are impractical for real-time applications due to their high complexity in feature extraction. Consequently, SGWs can be propelled to replace GWs for real-time applications and processing.

	5 scales, 4orientations			**4 scales, 4 orientations**			**3 scales, 4 orientations**		
SGW	**3-Lv**	**5-Lv**	**7-Lv**	**3-Lv**	**5-Lv**	**7-Lv**	**3-Lv**	**5-Lv**	**7-Lv**
	16.09 ms	27.50 ms	37.97 ms	12.81 ms	22.50 ms	30.94 ms	9.37 ms	17.50 ms	23.44 ms
GW	70.64 ms			56.73 ms			42.67 ms		
Speed-up rate	4.39	2.57	1.86	4.43	2.52	1.83	4.55	2.44	1.82

Table 4-6 The average runtimes for feature extraction using the GW and the SGW with different scales, orientations, and numbers of quantization levels. The speed-up rate is equal to the runtime required by the GW divided by that of the SGW.

4.6. Conclusion

In this chapter, we have proposed a simplified version of GWs, which can achieve a performance level similar to the original GWs for face recognition. We have also described fast algorithms for feature extraction based on SGWs at different orientations. In addition, we have presented how to construct these SGWs and their performance with different numbers of quantization levels, center frequencies and orientations. When 5 center frequencies and 4 orientations are employed, the relative performances of the SGWs and the GWs are very similar, while, at most, a speed-up rate of 4.39 times can be achieved if 3-level quantized SGWs are used. The runtimes required for feature extraction in a 64×64 image, based on a SGW with 3 quantization levels and a GW, are 16.09 ms and 70.64 ms, respectively. These results can propel SGWs to replace GWs for realizing real-time applications and processing. However, the simplified Gabor features are slightly more sensitive to lighting variations than the original Gabor features are.

In Chapters 3 and 4, we have presented our face tracking algorithm and face recognition/verification algorithm, respectively. These algorithms are highly efficient and have desirable performances which are favorable for building a real-time visual surveillance system.

Chapter 5

Conclusion and Future Work

5.1. Conclusion

In this thesis, we have firstly stated the motivation and the problems of research on face tracking and verification in videos. Our research focuses on the area of facial feature representations and feature matching algorithms, which are both efficient and effective methods for face tracking and verification. In Chapter 2, we have reviewed some well-known techniques and the current advances in face tracking and recognition/verification.

A good face tracking algorithm must be able to track faces varying poses and scales, under occlusion, and in a non-constant environment. In Chapter 3, we have proposed an effective face tracking algorithm based on the combination of shape and texture information. As the face patterns to be tracked in consecutive frames are highly correlated, an accurate tracking can be achieved by searching for the shortest weighted feature distance between the face pattern and the possible face candidates. The features used to represent a face include the edge map, which describes the shape of a face, and the local binary pattern (LBP), which describes the texture information about a face. The weights of the shape and texture feature distances can be adapted for real-time tracking. Both the edge map and the LBP can, to a certain extent, alleviate changes in environmental lighting. Moreover, skin-color-like objects will not be falsely tracked as a face. Our proposed algorithm complements the AdaBoost face detection algorithm to form a multi-view face-tracking system. Experimental results show that our algorithm can track faces in varying poses and scales, under occlusion and in a non-constant environment in real time.

In most of the current visual surveillance systems, face recognition/verification is a task beyond the capability of face tracking. In this scenario, accuracy is not the only criterion for the face recognition algorithms being used. Efficiency of the algorithms is also a critical issue so that real time processing in the surveillance systems can be ensured. The Gabor wavelet has been shown to have a high performance in face recognition, but it is rather computationally intensive in feature extraction. In Chapter 4, we have proposed a simplified version of GWs, which can achieve a performance level similar to the original GWs for face recognition, but with a much higher efficiency in feature extraction. We have also described fast algorithms for feature extraction based on SGWs at different orientations. In addition, we have outlined how to construct these SGWs and their performance with different numbers of quantization levels, center frequencies and orientations. When 5 center frequencies and 4 orientations are employed, the relative performances of the SGWs and the GWs are very similar, while, at most, a speed-up rate of 4.39 times can be achieved if 3-level quantized SGWs are used. The runtimes required for feature extraction in a 64×64 image, based on a SGW with 3 quantization levels and a GW, are 16.09 ms and 70.64 ms, respectively. These encouraging results imply that SGWs can replace GWs for realizing real-time applications and processing. However, the simplified Gabor features are slightly more sensitive to lighting variations than the original Gabor features are.

5.2. Future works

Outdoor tracking is much more difficult than indoor tracking. This is because the changes in lighting and the complexness of the background are far more complicated in an outdoor environment. In an outdoor environment, lighting can vary randomly by clouds moving in the sky. The background of an outdoor-tracking scene consists of many moving objects, which confuse most of the existing indoor tracking algorithms. A possible future work of this research is the tracking of faces in outdoor environments. One feasible way to approach this is to model the features of a detected face in a time-series sense. A time-series of feature models can tolerate to a certain extent, the effect of outdoor environmental changes. Models in the time-series can be weighted such that the individual importance of the models at different times can be reflected. Our shape-texture weighted algorithm can be modified to a time-series of weighted shape-texture models to track faces in outdoor environments.

The simplified Gabor wavelet (SGW) that we have proposed in Chapter 4 is not optimized in terms of quantization and reshaping. In our proposed method, we employ uniform quantization and the K-mean algorithm to quantize a Gabor function. The shape of a SGW is fixed such that a rectangle with dimension just large enough to enclose the quantized GW is used. However, these approaches are not optimal. We can obtain, through training, a set of optimal quantization values and rectangular shape. The bootstrap method [103] can be included to increase the reliability of the training. One drawback of this approach is that the training time can be very lengthy, as a large set of quantization values and rectangular shapes will be tested during the training. We believe that the SGW obtained from training can further increase its applicability to face recognition.

References

[1] W.M. Hu, T. Tan, L. Wang and S. Maybank, "A Survey On Visual Surveillance Of Object Motion And Behaviors," *IEEE Transaction on System, Man, and Cybernetics – Part C: Applications and Reviews*, vol.34, no.3, pp.334-352, August 2004.

[2] S. J. Maybank and T. N. Tan, "Special Section On Visual Surveillance – Introduction," *International Journal on Computer Vision*, vol.37, no. 2, pp 173-174, 2000.

[3] R. T. Collins, A. J. Lipton, and T. Kanade, "Introduction to the Special Section on Video Surveillance," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 745-746, August 2000.

[4] A. Hilton and P. Fua, "Modeling People Toward Vision-Based Understanding of A Person's Shape, Appearance, and Movement," *Computer Vision and Image Understanding*, vol. 81, no. 3, pp. 227-230, March 2001.

[5] C. Regazzoni and V. Ramesh, "Special Issue on Video Communications, Processing, and Understanding for Third Generation Surveillance Systems," *Proceeding of the IEEE*, vol. 89, pp. 1355-1367, Oct. 2001.

[6] M. Sadeghi, J. Kittler, A. Kostin and K. Messer, "*A Comparative Study of Automatic Face Verification Algorithms on the BANCA Database,*" Proceeding of Audio and Video Based Biometric Person Authentication, vol.2688, pp.1058-1067, 2003.

[7] R. Collins et al. "*A system for video surveillance and monitoring,*" VSAM Final Report, Carnegie Mellon Univ., Pittsburgh, PA, Tech. Rep. CMU-RI-TR-00-12, May 2000.

[8] M.H. Yang, D.J. Kriegman and N.Ahuja, "Detecting Faces in Images: A Survey," IEEE Transactions Pattern Analysis and Machine Intelligence, vol. 24, no. 1, January 2002.

[9] A. Blake and M. Isard, "Active Contours," *Springer*, 1998.

[10] M. Bichsel and A.P. Pentland, "Human Face Recognition and the Face Image Set's Topology," *CVGIP: Image Understanding*, vol. 59, no.2, pp. 254-261, March 1994.

[11] P.Y. Simard, Y.A.L. Cun, J.S. Denker, and B. Victorri, "Transformation Invariance in Pattern Recognition—Tangent Distance and Tangent Propagation," *Neural Networks: Tricks of the Trade*, pp. 239-274, 1998.

[12] M. Turk, "A Random Walk through Eigenspace," *IEICE Transaction on*

Information and Systems, vol. 84, no. 12, pp. 1586-1695, Dec. 2001

[13] Y. Adini, Y. Moses, and S. Ullman, “Face Recognition: The Problem of Compensating for Changes in Illumination Direction,” IEEE Transaction on Pattern Analysis and Machine Intelligence, vol. 19, no. 7, pp. 721-732, July 1997.

[14] P. Kakumanu, S. Makrogiannis, N. Bourbakis, “A Survey of Skin-Color Modeling and Detection Methods,” *Pattern Recognition*, vol.40, no.3, pp.1106-1122, March 2007.

[15] J. Yang, W. Lu, and A. Waibel, “Skin-Color Modeling and Adaptation,” Asian Conference on Computer Vision 1998, vol.1352, pp.687-694, 1998.

[16] M.H. Yang, and N. Ahuja, “Gaussian Mixture model for human skin color and its application in image and video databases,” *Proceedings of SPIE: Conference on Storage and Retrieval for Image and Video Databases*, vol. 3656, , pp. 458-466, 1999

[17] R.L. Hsu, M. Abdel-Mottaleb and A.K. Jain, “Face detection in color images,” *IEEE Transactions Pattern Analysis and Machine Intelligence*, vol.24, no.5, pp. 696–706, May 2002.

[18] S.L. Phung, A. Bouzerdoum, and D. Chai, “Skin Segmentation Using Color

Pixel Classification: Analysis and Comparison," IEEE Transactions Pattern Analysis and Machine Intelligence, vol.27, no.1, pp. 148-154, January 2005.

[19] M.J. Jones and J.M. Rehg, "Statistical Color Models with Application to Skin Detection," *International Journal on Computer Vision*, vol.46, no.1, pp.81-96, January 2002.

[20] K.Sobottka and I.Pitas, "A Novel Method for Automatic Face Segmentation, Facial Feature Extraction and Tracking," *Signal Processing: Image Communication*, vol.12, no.3, pp.263-281, 1998.

[21] H. Greenspan, J. Goldberger and I. Eshet, "Mixture Model for Face Color Modeling and Segmentation," *Pattern Recognition Letters*, vol. 22, pp. 1525-1536, Sept. 2001.

[22] M.-H. Yang and N. Ahuja, "Gaussian Mixture Model for Human Skin Color and Its Applications in Image and Video Databases," *SPIE Storage and Retrieval for Image and Video Databases*, vol. 3656, pp. 45-466, Jan. 1999.

[23] L.G. Farkas and I.R. Munro, "Anthropometric Facial Proportions in Medicine," *Springfield*, 1987.

[24] Y. Raja, S. McKenna and S. Gong, "Color model selection and adaptation in dynamic scenes," *Proceedings of the European Conference on Computer Vision*, vol.1406, pp. 460–474, 1998

[25] G. Jang, I. Kweon, "Robust real-time face tracking using adaptive color model," *International Conference on Robotics and Automation*, 2001.

[26] H. Stern and B. Efros, "Adaptive Color Space Switching For Tracking Under Varying Illumination," *International Journal of Image and Vision Computing*, vol.23, pp. 353–364, March 2005.

[27] T. Maurer and C. Malsburg, "Tracking and Learning Graphs And Pose on Image Sequences of Faces," *Proceedings of the International Conference on Automatic Face and Gesture Recognition*, pp. 176–181 1996.

[28] S. McKenna, S. Gong, R. Würtz, J. Tanner and D. Banin, "Tracking Facial Feature Points with Gabor Wavelets and Shape Models," *International Conference on Audio and Video-based Biometric Person Authentication*, vol.1206, pp. 35-42, 1997.

[29] S. Birchfield, "An Elliptical Head Tracker," *Conference Record of the Thirty-First Asilomar Conference on Signals, Systems & Computers*, vol.2, pp. 1710–1714, 1997.

[30] M. Pardàs and E. Sayrol, "A New Approach to Tracking with Active Contours," *International Conference on Image Processing*, vol.2, pp. 259–262, September 2000.

[31] X. Bing, Y. Wei and C. Charoensak, "Face Contour Tracking in Video using Active Contour Model," *International Conference on Image Processing*, vol.2, pp. 1021–1024, October 2004.

[32] S. Birchfield, "Elliptical Head Tracking using Intensity Gradients and Color Histograms," *International Conference on Computer Vision and Pattern Recognition*, pp. 232–237, 1998.

[33] T. Cootes, G. Edwards and C. Taylor, "Active Appearance Models," *European Conference on Computer Vision*, pp. 484–498, 1998.

[34] T. Cootes, G. Wheeler, K. Walker and C. Taylor, "View-Based Active Appearance Models," *International Journal of Image and Vision Computing*, vol.20, pp. 657–664, Sep 2002.

[35] J.-W. Sung and D. Kim, "A Background Robust Face Tracking using Active Contour Technique Combined Active Appearance Model," *International Conference on Biometrics*, vol.3832, pp. 159–165, 2005.

[36] J.-W. Sung and D. Kim, "Large Motion Object Tracking using Active

Contour Combined Active Appearance Model," *International Conference on Computer Vision Systems*, pp. 31, January 2006.

[37] S. Baker and I.Matthews, "Lucas-kanade 20 years on: A Unifying Framework," *International Journal of Computer Vision*, vol.56, no.3, pp.221-255, March 2004.

[38] G.D. Hager and P.N. Belhumeur, "Efficient Region Tracking with Parametric Models of Geometry and Illumination," *IEEE Transactions Pattern Analysis and Machine Intelligence*, vol.20, no.10, pp.1025-1039, October 1998.

[39] M. Bichsel and A.P. Pentland, "Human Face Recognition and the Face Image Set's Topology," *CVGIP: Image Understanding*, vol. 59, pp. 254-261, 1994.

[40] P.Y. Simard, Y.A.L. Cun, J.S. Denker and B. Victorri, "Transformation Invariance in Pattern Recognition Tangent Distance and Tangent Propagation," *Neural Networks: Tricks of the Trade, G.B.Orr and K.-R. Muller, eds., Springer*, 1998

[41] P. Viola and M. Jones, "Robust Real Time Object Detection," *International Journal of Computer Vision*, vol.57, no.2, pp.137-154, May 2004.

[42] S.Z. Li and Z.Q. Zhang , "Floatboost Learning and Statistical Face Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.26, no.9, pp.1112-1123, September 2004.

[43] G.Welch and G. Bishop, "An Introduction to the Kalman Filter," *UNC-Chapel Hill, TR 95-041*, July, 2006.

[44] A. Blake, R. Curwen and A. Zisserman, "A Framework of Spatio-Temporal Control in the Tracking of Visual Contours," *International Journal of Computer Vision*, vol.11, no.2, pp.127-145, 1993.

[45] G. Calvagno, F. Fantozzi, R. Rinaldo and A. Viareggio, "Model-Based Global and Local Motion Estimation for Videoconference Sequences," *IEEE Transaction on Circuits and Systems for Video Technology*, vol.14, no.9, pp.1156-1161, Sep 2004.

[46] M. Isard, A. Blake, "Visual Tracking by Stochastic Propagation of Conditional Density," *Proceeding of the 4th European Conference on Computer Vision*, pp. 343-356, 1996.

[47] M. Isard and A. Blake, "CONDENSATION – Conditional Density Propagation for Visual Tracking," *International Journal of Computer Vision*, vol.29, no.1, pp.5-28, 1998.

[48] M. Isard and A. Blake, “ICONDENSATION: Unifying Low-Level and High-Level Tracking in a Stochastic Framework,” *European Conference on Computer Vision*, vol.1406, pp. 893-908, 1998.

[49] M. Spengler and B. Schiele, “Toward Robust Multi-Cue Integration for Visual Tracking,” *Machine Vision and Applications*, vol.14, no.1, April 2003.

[50] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, “Pfinder: Real-time Tracking of the Human Body,” *IEEE Transactions Pattern Analysis and Machine Intelligence*, vol. 19, pp. 780–785, July 1997.

[51] Y. Wu and T. S. Huang, “A Co-Inference Approach to Robust Visual Tracking,” *International Conference on Computer Vision*, vol.2, pp.26-33, July 2001.

[52] H.S. Lee, D. Kim, “Robust Face Tracking by Integration of Two Separate Trackers: Skin Color and Facial Shape,” *Pattern Recognition*, vol.40, no.11, pp. 3225-3235, Nov 2007.

[53] J.Matas et al., “Comparison of face verification results on the XM2VTS database,” *International Conference on Pattern Recognition*, vol.4, pp.858-863, Sept. 2000.

[54] M. Kirby and L. Sirovich, "Application of the KL Procedure for the Characterization of Human Faces," *IEEE Transactions Pattern Analysis and Machine Intelligence*, vol. 12, no. 1, pp. 103-108, 1990.

[55] M. Turk and A. Pentland, "Eigenfaces for Recognition," *Journal of Cognitive Neuroscience*, vol. 3, pp. 71-86, 1991.

[56] A. Pentland, B. Moghaddam, T. Stamer, and M. Turk, "View-Based and Modular Eigenspaces for Face Recognition," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 84-91, 1994.

[57] J. Lu, N. Plataniotis Kostantinos and N. Venetsanopoulos Anastasios, "Face Recognition Using LDA-Based Algorithms," *IEEE Transaction on Neural Networks*, vol.14, no.1, pp.195–200, 2003.

[58] A.M. Martinez and A.C. Kak, "PCA Versus LDA," *IEEE Transactions Pattern Analysis and Machine Intelligence*, vol.23, no.2, pp.228–233, 2001.

[59] K.C. Yow and R. Cipolla, "Feature-Based Human Face Detection," *Image and Vision Computing*, vol. 15, no. 9, pp. 713-735, 1997.

[60] C.K. Chui, An Introduction to Wavelets, *Boston: Academic Press*, 1992.

[61] C. Liu and H. Wechsler, "Independent Component Analysis of Gabor

Features for Face Recognition," *IEEE Transactions Neural Networks*, vol. 14, no. 4, pp. 919-928, 2003.

[62] M. Lades, J.C. Vorbruggen, J. Buhmann, J. Lange, C.V.D. Malsburg, R.P. Wurtz, and W. Konen, "Distortion Invariant Object Recognition in the Dynamic Link Architecture," *IEEE Transactions Computers*, vol. 42, no. 3, pp. 300-311, 1993.

[63] C. Liu and H. Wechsler, "Gabor Feature Based Classification Using the Enhanced Fisher Linear Discriminant Model for Face Recognition," *IEEE Transactions on Image Processing*, vol. 11, no. 4, pp. 467-476, 2002.

[64] D. Liu, K.M. Lam, and L.S. Shen, "Optimal Sampling of Gabor Features for Face Recognition," *Pattern Recognition Letters*, vol. 25, no. 2, pp. 267-276, 2004.

[65] T.S. Lee, "Image Representation Using 2D Gabor Wavelets," *IEEE Transactions Pattern Analysis and Machine Intelligence*, vol. 18, no. 10, pp. 959-971, 1996.

[66] L. Shams and C. Malsburg, "The Role of Complex Cells in Object Recognition," *Vision Research*, vol. 42, no. 22, pp. 2547-2554, 2002.

[67] T. Ojala, M. Pietika¨inen, and D. Harwood, "A Comparative Study of

Texture Measures with Classification Based on Feature Distributions," *Pattern Recognition*, vol. 29, no. 1, pp. 51-59, 1996.

[68] T. Ahonen, A. Hadid, and M. Pietika¨inen, "Face Recognition with Local Binary Patterns," European Conference on Computer Vision, vol.3021, pp. 469-481, 2004.

[69] A. Hadid, M. Pietika¨inen, and T. Ahonen, "A Discriminative Feature Space for Detecting and Recognizing Faces," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 797-804, June 2004.

[70] X. Feng, M. Pietika¨inen, and A. Hadid, "Facial Expression Recognition with Local Binary Patterns and Linear Programming," *Pattern Recognition and Image Analysis*, vol. 15, no. 2, pp. 546-548, 2005.

[71] C. Shan, S. Gong, and P.W. McOwan, "Robust Facial Expression Recognition Using Local Binary Patterns," *IEEE International Conference on Image Processing*, vol.2, pp. II-370-3, September 2005.

[72] G. Zhang, X. Huang, S.Z. Li, Y. Wang, and X. Wu, "Boosting Local Binary Pattern (LBP)-Based Face Recognition," *Proceeding of Advances in Biometric Person Authentication*, vol.3338, pp. 179-186, 2004.

[73] S.Z. Li, R. Chu, M. Ao, L. Zhang, and R. He, “Highly Accurate and Fast Face Recognition Using Near Infrared Images,” International Conference on Advances in Biometrics, vol.3832, pp. 151-158, 2006.

[74] W. Zhang, S. Shan, W. Gao, X. Chen, and H. Zhang, “Local Gabor Binary Pattern Histogram Sequence (LGBPHS): A Novel Non-Statistical Model for Face Representation and Recognition,” IEEE International Conference on Computer Vision, pp. I: 786-791, 2005.

[75] Y. Rodriguez and S. Marcel, “Face Authentication Using Adapted Local Binary Pattern Histograms,” European Conference on Computer Vision, vol.3954, pp. IV: 321-332, 2006.

[76] P.S. Penev and J.J. Atick, “Local Feature Analysis: A General Statistical Theory for Object Representation,” *Network-Computation in Neural Systems*, vol. 7, no. 3, pp. 477-500, Aug. 1996.

[77] B. Heisele, P. Ho, J. Wu, and T. Poggio, “Face Recognition: Component-Based versus Global Approaches,” *Compter Vision and Image Understanding*, vol. 91, nos. 1-2, pp. 6-21, 2003.

[78] R. Gottumukkal and V.K. Asari, “An Improved Face Recognition Technique Based on Modular PCA Approach,” *Pattern Recognition Letters*,

vol. 25, pp. 429-436, March 2004.

[79] T. Ahonen, M.Pietikainen, “Face Description with Local Binary Patterns: Application to Face Recognition,” *IEEE Transactions on Pattern Analysis And Machine Intelligence*, vol.28, no.12, pp.2037-2041, Dec 2006.

[80] Paul Viola and Michael J. Jones, “Rapid Object Detection Using A Boosted Cascade of Simple Features,” *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. I-511 – I-518, 2001

[81] R. Lienhart and J. Maydt, “An Extended Set of Haar-like Features for Rapid Object Detection,” *IEEE Conference on Image Processing*, vol.1, pp. I-900 - I-903, September 2002.

[82] Bernhard Froba and Christian Kublbeck, “Face Tracking by Means of Continuous Detection,” *IEEE Conference on Computer Vision and Pattern Recognition Workshop*, pp. 65-65, June 2004

[83] A.U. Batur and M.H. Hayes, “Adaptive active appearance model,” *IEEE Transaction on Image Processing*, vol. 14, no. 11, pp. 1707-1721, November 2005.

[84] L. Wiskott, J. Fellous, N. Kruger, and C.V. Malsburg, “Face Recognition by Elastic Bunch Graph Matching,” *IEEE Transactions Pattern Analysis and*

Machine Intelligence, vol. 19, no. 7, pp. 775-779, July 1997.

[85] T.W. Yoo and I.S Oh, "A Fast Algorithm for Tracking Human Faces Based on Chromatic Histograms," *Pattern Recognition Letters*, vol.20, no. 10, pp. 967-978, 1999.

[86] M. Tuceryan and A. Jain. Texture analysis. In C. Chen, L. Pau, and P.Wang, editors, Handbook of Pattern Recognition and Computer Vision, pp.207-248, World Scientific, second edition, 1999.

[87] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Transactions Pattern Analysis and Machine Intelligence*, no.24, pp. 971-987, 2002.

[88] OpenCV, http://www.intel.com/technology/computing/opencv/index.htm

[89] Multi-view face tracking results in this chapter, http://myweb.polyu.edu.hk/~05900814r/index.html

[90] J.K. Kamarainen, V. Kyrki and H. Kalviainen, "Invariance properties of Gabor filter-based features-overview and applications," *IEEE Transaction on Image Processing*, vol. 15, no. 5, pp. 1088-1099, May 2006.

[91] A.C. Bovik, M. Clark, and W.S. Geisler, "Multichannel texture analysis

using localized spatial filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 12, no. 1, pp. 55-73, January 1990.

[92] B.S. Manjunath and W.Y. Ma, "Texture Feature for Browing and Retrieval of Image Data," I*EEE Transactions Pattern Analysis and Machine Intelligence*, vol. 18, no. 8, pp. 837-842, 1996.

[93] Y. Chen and R.S. Wang, "Texture Segmentation Using Independent Component Analysis of Gabor Features," *International Conference on Pattern Recognition*, vol. 2, pp. 20-24, August 2006.

[94] H. Cheng, N. Zheng, and C. Sun, "Boosted Gabor Features Applied to Vehicle Detection," *International Conference on Pattern Recognition*, vol.1, pp. 662-666, 2006.

[95] M. Valstar, and M. Pantic, "Fully Automatic Facial Action Unit Detection and Temporal Analysis," *IEEE Conference on Computer Vision and Pattern Recognition Workshop*, pp. 149, June 2006.

[96] E. Painkras, and C. Charoensak, "A Framework for the Design and Implementation of a Dynamic Face Tracking System," *IEEE Region 10 TENCON*, pp. 1-6, November 2005.

[97] S. Dubuisson, "An adaptive clustering for multiple object tracking in

sequences in and beyond the visible spectrum," *IEEE Conference on Computer Vision and Pattern Recognition Workshop*, pp. 142, June 2006.

[98] D. Tao, X. Li, S.J. Maybank, and X. Wu, "Human Carrying Status in Visual Surveillance," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 1670-1677, 2006.

[99] C. Liu, "Gabor-based kernel PCA with fractional power polynomial models for face recognition," *IEEE Transactions Pattern Analysis and Machine Intelligence*, vol. 26, no. 5, pp. 572–781, May 2004.

[100] C. Liu, "Capitalize on dimensionality increasing techniques for improving face recognition grand challenge performance," *IEEE Transactions Pattern Analysis and Machine Intelligence*, vol. 28, no. 5, pp. 725–727, May 2006.

[101] X. Xie, and K.M. Lam, "Gabor-based kernel PCA with doubly nonlinear mapping for face recognition with a single face image," *IEEE Transaction on Image Processing*, vol. 15, no. 9, pp. 2481-2492, September 2006.

[102] L. Shen, L. Bai and M. Fairhurst, "Gabor wavelets and General Discriminant Analysis for face identification and verification," *Image and Vision Computing*, vol. 25, no. 5, pp. 553–563, May 2007.

[103] S. Shan, P. Yang, X Chen, and W. Gao, "AdaBoost Gabor Fisher Classifier

for Face Recognition," *IEEE International Workshop on Analysis and Modeling of Faces and Gestures*, LNSC 3723, pp.279-292, 2005