Correlation filters and feature extraction for facial image analysis

Zhou, Lubing

2014

Zhou, L. (2014). Correlation filters and feature extraction for facial image analysis. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/59211>

<https://doi.org/10.32657/10356/59211>

Correlation Filters and Feature Extraction for Facial Image Analysis

ZHOU LUBING

School of Electrical and Electronic Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfilment of the requirement for the degree of
Doctor of Philosophy

2014

Acknowledgments

I would like to give my sincere thankfulness to Associate Professor Wang Han, for his continuous supervision and patience throughout my PhD life. He provided me the opportunity to experience a memorable life in NTU, and gradually led me to an independent researcher.

I must thank my seniors, Ying Ying and Wang Sisong, for the help since my first day in NTU. I also appreciate the assistance from Mr Tan Chai Soon, a very kind technician in autonomous robotics research lab. It is my pleasure to thank all the colleagues including Mou Wei, Tang Yue, Wang Tao, Akshay Rao and others for the friendship.

Most important, I would like to thank my parents and wife for their understanding and support in my life.

Contents

Acknowledgements	i
Summary	vi
List of Figures	viii
List of Tables	xi
List of Acronyms	xiii
1 Introduction	1
1.1 Motivation	1
1.2 Objectives	4
1.3 Major Contributions of the Thesis	5
1.4 Organization of the Thesis	8
2 Literature Review	10
2.1 Correlation Filtering for Pattern Detection	10
2.1.1 Correlation and Fourier Transform	10
2.1.2 Simple Templates for Object Detection	13
2.1.3 Overview of Correlation Filters and Notations	14
2.1.4 Synthetic Discriminant Functions (SDF), Minimum Average Correlation Energy (MACE), and Optimal Tradeoff Filters (OTF)	17

2.1.5	Unconstrained MACE (UMACE), Maximum Average Correlation Height (MACH), and Unconstrained Optimal Tradeoff SDF (UOTSDF)	19
2.1.6	Average of Synthetic Exact Filters (ASEF) and Minimum Output Sum of Squared Error (MOSSE)	21
2.1.7	An Overview of Common Object Detection Methods	22
2.2	Feature Extraction Methods	28
2.2.1	Intensity, Colors	29
2.2.2	Gradients, Edges	29
2.2.3	Keypoint Descriptors	31
2.2.4	Holistic Feature Representation	32
2.2.5	Local Feature Extraction	34
3	Correlation Filters	39
3.1	Introduction	39
3.2	Generalized Unconstrained Filters (GUF)	40
3.3	Analysis of GUF Filters	47
3.4	Relations to Prior Filters	51
3.4.1	GUF vs AVG	52
3.4.2	GUF vs UMACE	54
3.4.3	GUF vs MACH, UOTSDF	55
3.4.4	GUF vs ASEF	55
3.4.5	GUF vs MOSSE	56
3.5	Summary	57

4	Facial Image Analysis via Correlation Filters	58
4.1	Facial Landmark Localization	58
4.1.1	Introduction	58
4.1.2	Error Compensation and Filter Updating	60
4.1.3	Experiments	62
4.2	Face Detection and Tracking	73
4.2.1	Introduction	73
4.2.2	Detection and Online Tracking	76
4.2.3	Experiments	83
4.3	Summary	87
5	Image Representation by Local Features	89
5.1	Local Intensity Increasing Patterns (LIIP)	89
5.1.1	Extraction of LIIP	90
5.1.2	Properties of LIIP	92
5.2	Local Gradient Increasing Patterns (LGIP)	94
5.3	Multiscale Cell LIIP (MC-LIIP)	96
5.4	Summary	100
6	Facial Image Analysis Using Proposed Features	102
6.1	Facial Expression Recognition	102
6.1.1	Introduction	102
6.1.2	Multiscale Feature Fusion and Classifier Design	105
6.1.3	Experiments	108
6.2	Gender Recognition	117
6.2.1	Introduction	117
6.2.2	Face Representation and Classifier Design	119

6.2.3	Experiments	121
6.3	Eye State Recognition	125
6.3.1	Introduction	125
6.3.2	Extraction of Eye Descriptor	127
6.3.3	Open/Closed State Classifier	131
6.3.4	Experiments	132
6.4	Summary	136
7	Conclusion and Recommendations	139
7.1	Conclusion	139
7.2	Recommendation for Future Research	142
	Author's Publications	144
	References	146

Summary

Facial image analysis contains many applications including facial landmark localization, face detection, facial expression recognition, gender recognition, eye open/close state recognition and others. Novel algorithms within two types of techniques: correlation filtering and feature extraction have been developed in this work based on these applications.

Correlation filtering is one attractive technique to detect patterns of interest from images. This work presents an introduction of design theories for correlation filters, and proposes a new class of filters called Generalized Unconstrained Filters (GUF). Compared to existing filters, merits of GUF include: free of over-fitting problem, and relief from the restriction that targets should be placed in the center of training images. More importantly, GUF provides an insight to unify many existing filters including Average Filter (AVG), Unconstrained Minimum Average Correlation Energy (UMACE), Maximum Average Correlation Height (MACH), Unconstrained Optimal Tradeoff Synthetic Discriminant Functions (UOTSDF), Average of Synthetic Exact Filters (ASEF) and Minimum Output Sum of Squared Error (MOSSE). These filters are just special cases of GUF with appropriate parameters. Performance evaluation of GUF is conducted under two different scenarios. In one scenario, GUF filters are offline trained using thousands of face images, and then applied to find facial landmarks. In the other scenario, GUF is used as an online visual tracker, and provides a real-time solution to face detection under dif-

difficult conditions by incorporating with the well known Viola Jones detector. For both scenarios, extensive experiments have been carried out to show the promising performance of GUF filters.

On the other hand, feature extraction is an essential component of facial image analysis. Several feature extraction methods are presented to improve classification accuracy. First, a texture feature named Local Intensity Increasing Patterns (LIIP) is proposed to represent the intensity increasing trend within local intensity region. In theory, LIIP inherits the strengths from many existing approaches including Local Binary Patterns (LBP), uniform patterns and gradient information, and it represents local texture micro-structures in images. In addition, a variant of LIIP, the Local Gradient Increasing Patterns (LGIP) has been introduced to enhance the feature stability in case of white noise or non-monotonic intensity variations. Moreover, Multiscale Cell LIIP (MC-LIIP) extends original LIIP to scalable levels, so both texture micro-structures and macro-structures could be well described. Using these features, different strategies are adopted to extract feature vectors from face (big) and eye (small) images. Experiments show the proposed methods can achieve satisfactory performance in facial expression recognition, gender recognition and eye state recognition.

List of Figures

2.1	This figure shows the common use of correlation filtering	11
2.2	An example of simple template matching	14
2.3	Architecture of Cascade detector	26
2.4	Common architecture of pattern recognition system	28
2.5	Diagram for PCA and LDA	33
2.6	An example of LBP operator	36
2.7	Circular supporting neighborhood patterns with different radius and pixel numbers	37
2.8	Kirsch edge masks in eight directions	38
2.9	Generating process of LDP code	38
3.1	General architecture of facial landmark localization, right eye center is used for illustration	41
3.2	This figure shows many examples of 1D single-peaked function. . . .	42
3.3	Training images and synthetic correlation output for GUF training .	43
3.4	This figure shows the example of a training image and the correlation output being shifted to different peak locations	49
3.5	This figure illustrates how the GUF filter for right eye is constructed	52
3.6	This figure shows how AVG filter is trained using GUF method . . .	54
4.1	Illustration of image preprocessing procedure.	64

4.2	The nose filters with different training sizes.	67
4.3	This figure shows the correct localization rate of GUF using different size of training data.	68
4.4	GUF filters for detecting right eye center.	69
4.5	GUF filters for detecting nose.	70
4.6	Architecture of GUF tracking algorithm	78
4.7	Architecture of Cascade detector + GUF tracking algorithm	82
4.8	Initial frames along with the initial tracking window	84
4.9	This figure shows the detection results for each frame	86
4.10	This figure shows many challenging frames from the videos	87
5.1	This figure illustrates the concept of intensity increasing trend	90
5.2	An example of LIIP pattern generation.	91
5.3	Look up table for interpreting OTV to LIIP patterns.	92
5.4	Four examples of LIIP patterns.	94
5.5	Local gradient masks in eight directions.	95
5.6	An example of LGIP generation.	95
5.7	Stability of LBP, LDP, LIIP and LGIP.	97
5.8	Spatial support areas of MC-LIIP operator at different scales.	98
5.9	This figure shows the resulting average images and corresponding MC-LIIP pattern images at different scales.	98
5.10	This figure shows the MC-LIIP images of two emotional faces.	100
6.1	This figure illustrates the generation of face descriptor for expression recognition based on MC-LIIP feature.	106
6.2	The process to extract face descriptor based on LGIP features.	121

6.3	This figure illustrates the process to extract eye descriptor based on LIIP features.	128
6.4	Fast calculation of LIIP-based histogram bins from block ABCD. . .	130
6.5	Architecture of Discrete AdaBoost algorithm	132
6.6	Illustration of some positive and negative samples used for training	133
6.7	Illustration of corresponding regions of six most discriminative fea- tures.	134
6.8	Some examples of eye state recognition results.	135

List of Tables

3.1	Relations between GUF and other filters	53
4.1	Detection results of right eye center(%)	71
4.2	Detection results of nose (%)	72
4.3	Filter construction time with a size of 128×128 using 8192 training images	72
4.4	Categorization of frontal face detection methods	74
4.5	Comparison of Cascade detector and GUF tracker for face detection	81
4.6	Description of test videos	83
4.7	Face detection results of algorithms including Cascade detector, GUF tracker, and Cascade + GUF (%).	85
6.1	Recognition rates of MC-LIIP ₁ based face descriptor with different grid partitions	111
6.2	Recognition results of MC-LIIP _s features at various scales	111
6.3	Confusion matrix of 6-class facial expression recognition using MC- LIIP ₁₂₃₄	112
6.4	Confusion matrix of 7-class facial expression recognition using MC- LIIP ₁₂₃₄	113
6.5	Parameter settings in facial expression recognition	113

6.6	Algorithm comparisons between different approaches using CK/CK+ database.	114
6.7	Facial expression recognition results on KDEF database.	117
6.8	Parameter settings in gender recognition	121
6.9	Classification accuracy on CAS-PEAL-R1 database with different image resolutions and SVM kernels (%)	123
6.10	Comparisons of different methods for gender classification with FERET database (DT: decision trees)	124
6.11	Parameter settings in eye state recognition	129
6.12	Eye state recognition results on static images	135

List of Abbreviations

ASEF	Average of Synthetic Exact Filters
AVG	Average Filter
CCF	Constrained Correlation Filters
DFT	Discrete Fourier Transform
EOC	Equal Output Correlation-Peak
ESR	Eye State Recognition
FD	Face Detection
FER	Facial Expression Recognition
GR	Gender Recognition
IDFT	Inverse Discrete Fourier Transform
LBP	Local Binary Patterns
LDP	Local Directional Patterns
LGIP	Local Gradient Increasing Patterns
LIIP	Local Intensity Increasing Patterns
MACE	Minimum Average Correlation Energy
MACH	Maximum Average Correlation Height
MC-LIIP	Multiscale Cell Local Intensity Increasing Patterns
MOSSE	Minimum Output Sum of Squared Error
MVSDF	Minimum Variance Synthetic Discriminant Functions
OTF	Optimal Tradeoff Filters
SDF	Synthetic Discriminant Functions
SIFT	Scale-Invariant Feature Transform
UCF	Unconstrained Correlation Filters
UMACE	Unconstrained Minimum Average Correlation Energy
UOTSDF	Unconstrained Optimal Tradeoff Synthetic Discriminant Functions

Chapter 1

Introduction

Facial image analysis is a traditional but still very active area in image processing and computer vision. Its scope is vast and diversified. In terms of applications, relevant subtopics include facial landmark localization (FLL), face detection (FD), facial expression recognition (FER), gender recognition (GR), eye state recognition (ESR, to identify whether the eye is open or closed) and others [1]. During the past several decades, although tremendous work has been done, there are still substantial challenging problems such as large intraclass variations, lighting distortions, occlusions and cluttered backgrounds. This chapter provides an introduction of this work.

1.1 Motivation

This work focuses on solving two broad kinds of problems: object detection or localization, and pattern recognition or classification based on face applications. Object detection aims to search for a reference target from a cluttered background, and output the center location. On the other hand, pattern recognition requires to correctly identify the label of an image. In facial image analysis, object detection problems covered in this work contain FLL and FD; while pattern recognition problems contain FER, GR and ESR. The motivation of this work is to find some

alternative algorithms to better resolve these object detection and pattern recognition issues.

Finding objects in images is a fundamental and challenging computer vision problem. Simply, it is to determine whether a target is presented in an image, or where in the image the target is located. The concept is simple, but object detection tasks can be very difficult to solve because the appearance of object may change substantially when it is observed under different views. Such changes are jointly caused by factors including lighting, pose, viewpoint, occlusion and intraclass variation. Consequently, robust and accurate object localization remains an open problem. So finding an effective solution of object localization is one motivation of this thesis.

FLL and FD are frequently used as preceding steps before other tasks such as automatic face recognition. High accuracy and efficiency are two important aspects. Commonly, localization or detection accuracy directly affects the performance of face alignment or normalization, and further affects subsequent tasks such as face recognition. Meanwhile, runtime speed is a critical consideration in automatic detection system. One attractive approach in pattern detection is through correlation filtering, which searches for the peak in the output by correlating input image with the trained filter. Especially, implementation simplicity and time efficiency are two outstanding advantages of correlation filters. The simplest example is template matching [2]. Typically, a template is just a reference image that has been carefully cropped and centered on the target, while a filter generally refers to an advanced template trained from a number of samples. In other words, template can be viewed as simplest filter. However, template matching highly relies on the quality of the template image, and has poor tolerance of appearance distortions. In literature, many advanced filters have been proposed with examples including

Synthetic Discriminant Functions (SDF) [3], Minimum Average Correlation Energy (MACE) [4], Unconstrained MACE (UMACE) [5], Average of Synthetic Exact Filters (ASEF) [6] and so on.

Peak sharpness and distortion tolerance are two critical and somewhat conflicting performances of filters. Generally, different traditional filters use different objective functions during filter construction. Many previous filters suffer from following shortcomings. First, training images should be cropped carefully with target located in the center. This will limit the flexibility of training images. Taking FLL for example, it is hard to crop training images for these facial points that are located near face boundaries. Second, these filters that apply hard constraints will easily over-fit the training images, and greatly reduce the performance. In this work, a unified correlation filter called Generalized Unconstrained Correlation Filters (GUF) is proposed to handle above problems, and many state-of-the-art filters could be considered as special cases of GUF.

Pattern classification constitutes another key component of this thesis. Broadly, a pattern classification problem can be further divided into two parts: image feature extraction and classifier design. Both parts have significant influence on the final classification accuracy and can separately be expanded to broad areas, and this work concentrates on the former part. Undoubtedly, without a discriminative feature representation, classification is only a “garbage-in-garbage-out” process. Specifically, a robust face descriptor benefits facial tasks including FER and GR, and a robust eye descriptor benefits eye state recognition task. Even though a large amount of work has been reported to extract features of face and eye images, it is still a far from closed research field. A promising descriptor should be able to reduce within-class variation and increase between-class variation. Another motivation of this thesis is to develop new feature extraction approaches for facial image analysis.

Various features have been proposed for image representation ranging from raw pixels to advanced *integral image* [7] and Gabor features [8]. Recently, Local Binary Patterns (LBP), which was originally proposed for texture classification [9], has demonstrated outstanding capability for facial image analysis [10–14]. Implementation simplicity and fast speed are the most attractive properties of LBP. One drawback is that the LBP operator has low resistance to white noise [15]. This thesis proposes several local features that inherit the strengths of LBP and gradient information, and have less noise sensitivity.

1.2 Objectives

In literature, many correlation filters and feature extraction methods have been developed for facial image analysis. More or less, many drawbacks exist and the performance could be further improved. Generally, the work presented in the thesis is to explore new ways to achieve two primary objectives. The first objective is to develop fast and effective object detection approaches to locate facial features and detect faces under harsh conditions. The other objective is to work out new face descriptor and eye descriptor for FER, GR and ESR. Details are given as follows.

Correlation filters are commonly trained in frequency domain. The reason is that correlation or convolution in spatial domain can be equivalently computed very quickly as element-wise multiplication after the Fourier Transform (FT). Correlation filtering has been successfully applied in optical pattern recognition system [3, 4]. This work focuses on face related applications, and a critical summary can provide fine perspective and better guide the research. Our goal is to design filters that maintain high peak sharpness, distortion as well as noise tolerance. Moreover, the generality capability is also tested by training the filters with large-scale

data and a single sample. Additionally, theory analysis is conducted to describe the strengths and weakness of state-of-the-art filters and reveal their internal relations.

Local features [16] provide a new insight for facial image representation. Hence part of my research is devoted to the design of new local features. Originally LBP features encode the eight pixel-wise comparisons between center pixel and eight neighboring pixels. Alternatively, we find the intensity increasing trend is a good identification of local patches. Consequently, the Local Intensity Increasing Patterns (LIIP) is proposed to encode local intensity images based on such trend. In addition, LIIP can evolve to more stable variant and multiscale version. Compared to LBP, the feature length can be greatly reduced to 37 from 256 (or 59 for Uniform Local Binary Patterns). In practice, big face images and small eye images differ in terms of feature extraction based on local features. Therefore, one problem pending for solving is on how to apply these local features to represent faces and eyes.

1.3 Major Contributions of the Thesis

The major contributions of this thesis are listed as follows:

- A family of correlation filters, namely Generalized Unconstrained Filters (GUF), is introduced. Comparing to prior filters such as SDF, MACE and Optimal Tradeoff Filters (OTF), GUF no longer needs to restrict to the equal output correlation-peak (EOC) constrains that correlation response at the training images' center must equal to a fixed constant. This helps to prevent GUF filter from over-fit the training data that is faced by SDF, MACE and OTF. Besides over-fitting problem, these traditional filters also suffer from target-in-center (TIC) issue. That is, the target point must be located in the center of training images. Many advanced filters require to obey the

TIC requirement including the Average Filter (AVE), UMACE, Maximum Average Correlation Height (MACH) [5], and Unconstrained Optimal Trade-off Synthetic Discriminant Function filters (UOTSDF) [17]. GUF avoids the TIC requirement and allows greater flexibility with respect to training images. This point is important in detecting facial landmarks, especially for those points near the face boundaries. Furthermore, GUF is shown to be a general form of AVE, UMACE, MACH, UOTSDF, ASEF, and Minimum Sum of Squared Output Error (MOSSE) [18]. In other words, these filters are special cases of GUF when the parameters are set properly.

- FLL and FD tasks respectively show the capability of GUF under two kinds of scenarios. In FLL, a large number of training image are utilized to separately train offline filters for different facial points. The resultant filters normally appear as the blurred portrait of keypoint template with the keypoint in the center. In addition, an iterative error compensation phase is added to correct these inaccurate detections and accordingly update the filter. Extensive experiments show the GUF filters can achieve promising localization accuracy in facial point finding. Differently, face detection is accomplished by detection-by-tracking strategy. In detail, the GUF filter is initialized using only the first frame of a video sequence, and then updated using the detected face in next frame. One preferable characteristic of GUF is that it is convenient to add new image to update the filter without recalculation for previous images. The GUF tracker and Viola Jones cascade detector [7] are combined to complement each other. GUF tracking can detect face under very harsh conditions such as big in-plane rotation, out-of-plane rotation, occlusion, and poor lighting, which are beyond the capability of viola Jones detector. On the other hand, adaptive appearance trackers often suffer from drift, a gradual

adaptation of trackers to background. Adding Viola Jones detector provides a way of tracker rectification and recovery from drift.

- The second part of this thesis presents a novel appearance-based feature called Local Intensity Increasing Patterns (LIIP), which is capable of describing the intensity distribution or texture structure surrounding each pixel. Theoretically, LIIP uses eight trend vectors to separately express the intensity increasing conditions along eight directions based on pixel-wise comparisons. Afterwards, a global vector by adding these individual trend vectors is obtained to indicate intensity increasing status of the whole region. Eventually, decimal numbers are utilized to encode such global vectors. LIIP has many good properties: simple, fast, illumination invariant, and small pattern number (37 in total). To enhance the stability, a variant of LIIP named Local Gradient Increasing Patterns (LGIP) is developed. Instead of pixel-wise comparisons, LGIP adopts directional Sobel operators to get individual trend vectors. Additively, LIIP is further extended from pixel level to cell level, leading to the Multiscale Cell LIIP (MC-LIIP) feature. In MC-LIIP, trend vectors are calculated using average values of scalable cells rather than pixels. Certainly, MC-LIIP possesses higher stability and noise invariance. Meanwhile, it can encode not only local textural micro-structures but also macro-structures.
- Based on the proposed local features, different strategies are used to extract feature vectors from face and eye images for relevant tasks including facial expression analysis, gender recognition and eye state recognition. To obtain face representation, the images are first partitioned into grid regions where sub-histograms of aforementioned local features are extracted, normalized and then concatenated to a single feature vector. Expression and gender

recognition are carried out by incorporating the face descriptor with SVM classifier [19, 20]. However, grid partition is not applicable to eye images due to their small size. Given an eye image, a large number of blocks are obtained by scanning the image at various scales and locations, and then normalized histograms from these blocks are concatenated to a feature vector. AdaBoost method [21] is adopted to selecting a small number of important features and classifying open/closed state. Extensive experiments have been conducted to show the validity of the proposed methods in aforementioned tasks.

1.4 Organization of the Thesis

The thesis is organized into seven chapters summarized as follows. The motivation and objective of the research work are introduced in this chapter, and major contributions are summarized.

Chapter 2 provides an overview on related techniques and applications. Two core techniques documented in this thesis are: correlation filtering and feature extraction methods. First, a short introduction to correlation and major developments of correlation filters are provided. The construction theories for several leading filters are described such as MACE, UFACE, ASEF and MOSSE. Second, a short survey is given on state-of-the-art feature extraction methods, especially for those appearance-based local features.

Chapter 3 introduces the theories of constructing GUF filters. This includes the optimization process behind GUF and deep investigation to the properties of GUF. More important, this chapter also bridges the gaps between GUFs and other state-of-the-art filters including AVE, UFACE, MACH, UOTSDF, ASEF and MOSSE. Elaborate explanation and proofs are presented to show GUF is a common version of these filters.

Chapter 4 investigates the capability of GUFs in two different applications: FLL and FD. For the former one, offline training of GUFs with large-scale data is conducted to detect facial features such as eye centers and nose. Additionally, the performance of GUF filters is improved by introducing an extra error compensation and filter updating step. Experimental results with FERET data set [22] show the performance improvement of GUFs over prior filters and other complex methods. For face detection, this chapter presents the methodology of combining GUF tracker and cascade detector, and shows these two methods are mutually complementary. GUF tracker is able to track the face under very difficult scenarios, while cascade detector helps the tracker in filter recovery and alleviating drift problem.

Chapter 5 presents the generation process of the LIIP feature, and analysis of its characteristics. In addition, LIIP operator is expanded to a variant: the LGIP feature. This chapter also investigates the significance of introducing scalable LIIP, i.e. MC-LIIP feature.

Chapter 6 conducts tests of the proposed features on three face related tasks: FER, GR and ESR. It also introduces different feature vector extraction ways for face and eye images due to image size reason. Respectively, multi-class and binary SVM are adopted to perform FER and GR. In FER, various approaches are compared with the Extended Cohn-Kanade (CK+, [23]) and KDEF [24] databases, and the proposed methods show satisfactory performance. GR evaluation use FERET and CAS-PEAL databases to examine the recognition ability. Moreover, fast calculation of eye descriptor is proposed based on *integral image*, and AdaBoost classifier is applied to recognize open/closed eye state with a large-scale combo dataset as well as video sequences.

Finally, Chapter 7 summarizes the major conclusions of the thesis, and many remaining issues are listed. It also provides an insight of some directions for future research.

Chapter 2

Literature Review

This chapter presents tutorial level introduction of correlation filtering and feature extraction techniques which constitute the main topics of this thesis.

2.1 Correlation Filtering for Pattern Detection

2.1.1 Correlation and Fourier Transform

In mathematics, the term correlation means a dot product or inner product between two functions. It also refers to a similarity measure of two waveforms in signal processing. Commonly, correlation may be better known as the cross-correlation in mathematics and signal processing.

For two functions or signals, f and h , their correlation is typically denoted by the operator $\star$, and the mathematical definition is expressed as:

$$y(x) = f(x) \star h(x) = \int_{-\infty}^{\infty} f(\tau)h(x + \tau)d\tau \quad (2.1)$$

In practice, f and h usually refer to an original signal and a filter. Accordingly, y is the correlation output, which can be viewed as a modified version of f . Similarly, for one-dimensional discrete functions, the correlation is defined as:

$$(f \star h)[x] = \sum_{dx=-\infty}^{\infty} f[dx]h[x + dx] \quad (2.2)$$

Since this document concentrates on image analysis, the most frequently used two-dimensional correlation is expressed as:

$$(f \star h)[x, y] = \sum_{dx=-\infty}^{\infty} \sum_{dy=-\infty}^{\infty} f[dx, dy]h[x + dx, y + dy] \quad (2.3)$$

When talking about correlation, people often think of another similar concept called convolution. In nature, correlation and convolution are interchangeable. The only difference is that in convolution the filter should be first reversed.

Figure 2.1: This figure shows the common use of correlation filtering

Figure 2.1 shows the block diagram for common filtering process. Most frequently, design of correlation filters is carried out in frequency domain or Fourier domain. The main reason is that the computational complexity of correlation calculation in spatial domain could be greatly reduced though Correlation Theorem in Fourier domain. Given an image $f[x, y]$, its Discrete Fourier Transform (DFT) is defined as:

$$F[u, v] = \frac{1}{\sqrt{MN}} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f[x, y] e^{-j2\pi(\frac{ux}{M} + \frac{vy}{N})} \quad (2.4)$$

Reversely, original image can be obtained by Inverse Discrete Fourier Transform (IDFT):

$$f[x, y] = \frac{1}{\sqrt{MN}} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F[u, v] e^{j2\pi(\frac{ux}{M} + \frac{vy}{N})} \quad (2.5)$$

Correlation Theorem states the correlation of two functions in spatial domain can be computed by element-wise multiplication between Fourier Transform of one function and complex conjugate of Fourier Transform of another function. Formally, the correlation of function f and filter h can be computed as follows:

$$f \star h = \mathcal{F}^{-1}(\mathcal{F}(f) \odot \mathcal{F}^*(h)) = \mathcal{F}^{-1}(F \odot H^*) \quad (2.6)$$

where $\mathcal{F}$ and $\mathcal{F}^{-1}$ denote DFT and IDFT, and $\odot$ denotes element-wise multiplication. In similar manner, convolution ($*$) can also be computed via Convolution Theorem which is expressed as:

$$f * h = \mathcal{F}^{-1}(\mathcal{F}(f) \odot \mathcal{F}(h)) = \mathcal{F}^{-1}(F \odot H) \quad (2.7)$$

It is worthwhile to clarify the difference in terms of speed by using spatial and DFT-based ways to compute correlation or convolution. David Bolme has conducted detailed comparisons in his PhD dissertation [25]. The major conclusions are listed as follows. First, DFT is the fastest way to compute correlation when the filter size is large, and preprocessing is required to smooth image edges to produce good results because DFT method actually computes circular cross-correlation where images are supposed to be periodically repeated. Second, computational complexity of using two-dimensional (2D) spatial correlation, separable one-dimensional (1D) spatial correlation, and two-dimensional DFT are $O(MNPQ)$, $O(MN(P + Q))$, and $O(MN(\log(MN)))$ respectively, where $M \times N$ and $P \times Q$ are the sizes of image and filter. To be specific, DFT-based way runs faster than 2D spatial correlation when filter size exceeds 8×8 , and faster than 1D separable correlation when filter size exceeds around 40×40 . In runtime, the detection process on an input image of 128×128 pixels costs only a few milliseconds on a common computer.

Besides the fast speed, another major advantage of correlation based detection is shift invariance [26]. That is, the correlation operation can smartly project the input image vector onto the filter vector at every shift. As a result, the input image has no need to be centered before correlation. For other non-linear classifiers such as neural network and SVMs, this property is not applicable, so usually preprocessing must be implemented to roughly locate the center.

2.1.2 Simple Templates for Object Detection

Matching with a predefined template is very earliest and simplest way to detect objects. This technique is widely recognized as *template matching*. Template matching searches for matches between an image patch (template) and the input image. Figure 2.2 shows a simple example of template matching. Many matching methods are applicable to measure the similarity between two patches such as Sum of Absolute Difference and Normalized Cross-correlation. The best match can be found at global minimum or maximum in the correlation plane (output). Although template matching is simple to implement and understand, it is computationally intensive because the matching process must run once on every pixel in spatial domain.

Simple templates can perform well when the objects in input images have small difference or variations to the templates. One successful application is optical flow technique to estimate local motions in consecutive frames [27], in which the templates are small local patches from last frame. However, in more general cases, template matching is introduced as a theoretical example for the purpose of better understanding. In object detection, simple templates are rarely used in real tasks where object appearance often has large variations, and a specific template can

Figure 2.2: An example of simple template matching

only handle relatively ideal conditions. To enhance generalization capability of the template, computing the average of many training images is a possible solution, and the resultant template is called Average Filter (AVG). Compared to simple template, AVG can be applied to broader conditions, and meantime peak sharpness in the output may also decrease.

2.1.3 Overview of Correlation Filters and Notations

Instead of simple templates, there are a great many advanced methods to design correlation filters. This document focuses on their use for object detection. In general, a desired filter should have three good characteristics: (i) sharp correlation peak to easily distinguish target object from the background, (ii) effectively suppress clutter and noise, and (iii) nice distortion tolerance. Different filters utilize different optimization methods to achieve these three goals. Commonly, filters are constructed with a set of training images, and named with the terminology *composite filters*.

Broadly, composite filters can be classified into constrained and unconstrained categories [5, 28, 29]. Constrained correlation filters (CCFs) are constructed by imposing hard constraints to the correlation output of training images: the value

at the origin (normally image center) must be equal to a predefined value [28]. Such restrictions are known as the equal output correlation-peak (EOC) constraint [29]. This also implies the target must be placed in the center of training images or patches. As a result, the flexibility with respect to the training images is limited, especially when it comes to detect these facial landmarks that lie far away from face center. Chronologically, earliest representative of CCFs is Synthetic Discriminant Functions (SDFs) [3, 30, 31]. Other notable CCFs include Minimum Average Correlation Energy (MACE) [4], Minimum Variance Synthetic Discriminant Functions (MVSDF) [32], Gaussian-MACE [33], and Optimal Tradeoff Filters (OTF) [34]. One common shortcoming of CCFs is that the filters easily over-fit the training data due to the hard EOC restriction [5, 6]. Consequently, CCFs are only trained using at most dozens of images, and the results are not very satisfactory for facial landmark localization as reported in [6, 25].

In contrast, more advanced filters remove the EOC constraints, and they are known as unconstrained correlation filters (UCFs). Indeed, such hard constraints have been reported to be unnecessary or even harmful [5]. Instead of imposing EOC constraint to each image, UCFs are designed by optimizing certain statistical measures over the entire training set. For this reason, they can avoid suffering from over-fitting issue encountered by constrained correlation filters. In literature, one traditional class of UCFs includes Unconstrained Minimum Average Correlation Energy (UMACE) [5], Maximum Average Correlation Height (MACH) [5], and Unconstrained Optimal Tradeoff Synthetic Discriminant Functions (UOTSDF) [17]. UMACE, MACH, and UOTSDF no longer have over-fitting problem, but they still require the targets to be placed in image center.

In summary, CCFs face with both over-fitting problem and target-in-center problem, while aforementioned UCFs also encounter the second problem. Recently,

Bolme et al. proposed a new UCF named Average of Synthetic Exact Filters (ASEF) [6] that is applicable to use non-centered training images, which provides greater flexibility with regard to training data. ASEF computes so-called exact filters for each training image, and the average is calculated as the global filter. Soon after, Bolme et al. proposed an extension of ASEF: Minimum Output Sum of Squared Error (MOSSE) [18]. The difference in MOSSE is that filter optimization uses sum of squared error criterion instead of averaging.

Correlation filtering is still an ongoing research field and there are many notable research directions. In [35], Principal Component Analysis was applied to the synthetic exact filters of training images. Compared to a single filter, subspace based technique can represent larger target variations at the sacrifice of about n times of computational cost (n is the number of principals). Another research direction is to develop nonlinear filters, which aim to increase the representational capability of linear filters. One realization way is to apply nonlinear transformations to the input data before learning the linear filters [36]. Another way is using nonlinear kernel function to replace the inner product (correlation) [37], but this will lose the speed advantage of computing correlation in frequency domain. In [38], the authors proposed the Action MACH spatio-temporal domain for gesture recognition which extends filters from 2D to 3D. In 2013, Hamed et al. proposed a framework to learn correlation filters in multi-channels and combine the output to a single response map [39]. In addition, correlation filters were applied to other applications such as object alignment in [40], which proposed a landmark appearance model where landmarks were detected via correlation filters.

The following contents will introduce the design theories of typical CCFs and UCFs. For consistency, many notations should be declared beforehand. A training set has N training images with each image containing $d = d_1 \times d_2$ pixels. The i th

image is denoted by x_i in spatial domain. Let X_i (or $X_i(u, v)$) be the FT of x_i , i.e. $X_i = \mathcal{F}(x_i)$. A $d \times 1$ complex vector $\mathbf{x}_i$ is obtained by lexicographically reordering the rows of X_i . Besides, $\mathbf{X}_i$ denotes the $d \times d$ matrix with $\mathbf{x}_i$ being the diagonal. The size of filter is always equal to the size of training image. Accordingly, in frequency domain the desired filter can be represented by a $d_1 \times d_2$ complex image $H(u, v)$, or corresponding reshaped column vector $\mathbf{h}$. The matrix notation of the training set $\mathbf{X}$ is obtained by putting each vectorized image $\mathbf{x}_i$ as a column: $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N]$. Note matrices and vectors are notated by upper-case and lower-case characters. By the way, let $\mathbf{m}$ be the frequency vector of average image as in Eq. (2.8), and $\mathbf{M}$ be the $d \times d$ diagonal form of $\mathbf{m}$. It is also the AVG filter mentioned above.

$$\mathbf{m} = \frac{1}{N} \sum_{i=1}^N \mathbf{x}_i \quad (2.8)$$

2.1.4 Synthetic Discriminant Functions (SDF), Minimum Average Correlation Energy (MACE), and Optimal Tradeoff Filters (OTF)

As mentioned, using EOC constraint is one distinguishing quality of CCFs. Specifically, correlation peak should locate at image origin (center) and is equal to a constant c (commonly $c = 1$), that is, $c = \mathbf{x}_i \mathbf{h}^*$. In matrix-vector form, EOC constraint can be written as:

$$\mathbf{c}^* = \mathbf{X}^+ \mathbf{h} \quad (2.9)$$

where “ $*$ ” and “ $+$ ” represent complex conjugate and conjugate transpose, $\mathbf{c}$ is an $N \times 1$ vector. This line of research has led to many well-known filters such as SDF, MVSDf, MACE, and OTF. In some sense, SDF can be considered as the starting point of correlation filtering. Basically, it is based on the assumption that the filter is composed as a linear combination of training images as expressed by:

$$\mathbf{h} = \sum_{i=1}^N a_i \mathbf{x}_i = \mathbf{X} \mathbf{a} \quad (2.10)$$

Combing Eqs. (2.9) and (2.10), SDF filter is solved as:

$$\mathbf{h}_{\text{SDF}} = \mathbf{X}(\mathbf{X}^+ \mathbf{X})^{-1} \mathbf{c}^* \quad (2.11)$$

SDF can respond well to positive images, but SDF is not optimized for noise tolerance since it does not consider any noise. Meanwhile, because it controls only the correlation value at the origin, it is hard to guarantee that the value will be the global maximum. Hence, the peak sharpness at the origin is weakened.

MVSDF removes the conventional assumption that the filter must be a linear combination of training images. It satisfies the EOC constraint and simultaneously minimizes the output noise variance (ONV) resulted from input noise [32]. Under this optimization, the MVSDF filter is defined as:

$$\mathbf{h}_{\text{MVSDF}} = \mathbf{C}^{-1} \mathbf{X}(\mathbf{X}^+ \mathbf{C}^{-1} \mathbf{X})^{-1} \mathbf{c}^* \quad (2.12)$$

where $\mathbf{C}$ is a diagonal matrix containing the power spectrum of input noise. MVSDF typically enhances low-frequencies that suppress noise, but at the same time the peak sharpness is weakened. Interestingly, traditional SDF is identical to MVSDF when the input noise is white, i.e. $\mathbf{C} = \mathbf{I}$ ($\mathbf{I}$ - identity matrix). However, MVSDF is mainly used for theory study purpose, because it is hard to estimate how *colored* the input noise is.

MACE is capable of producing sharp correlation peak at the origin by minimizing the average correlation energy (ACE) while satisfying the EOC constraints. On one hand, EOC restriction can guarantee correlation peak at origin has a fixed value, and on the other, minimizing ACE is to yield near zero values all over the correlation plane. Together with the two aspects, MACE is able to produce a sharp peak at the output origin and very small values elsewhere. Formally, the objective

function of MACE is to minimize ACE expressed in Eq. (2.13), subject to EOC constraint in Eq. (2.9).

$$\text{ACE} = \mathbf{h}^+ \mathbf{D}_x \mathbf{h}, \quad (2.13)$$

Here $\mathbf{D}_x$ is the average energy plane of training images:

$$\mathbf{D}_x = \frac{1}{N} \sum_{i=1}^N \mathbf{X}_i^+ \mathbf{X}_i \quad (2.14)$$

Eventually, the filter is given by:

$$\mathbf{h}_{\text{MACE}} = \mathbf{D}_x^{-1} \mathbf{X} (\mathbf{X}^+ \mathbf{D}_x^{-1} \mathbf{X})^{-1} \mathbf{c}^* \quad (2.15)$$

In summary, SDF can minimize ONV and hence maximize noise tolerance if input noise is white, whereas MACE maximizes peak sharpness [28]. In fact, both aspects are important, so a compromise filter, OTF, was designed in [28].

$$\mathbf{h}_{\text{OTF}} = \mathbf{D}^{-1} \mathbf{X} (\mathbf{X}^+ \mathbf{D}^{-1} \mathbf{X})^{-1} \mathbf{c}^* \quad (2.16)$$

where $\mathbf{D} = \alpha \mathbf{D}_x + \sqrt{1 - \alpha^2} \mathbf{I}$. Comparing Eqs. (2.11), (2.15) with Eq. (2.16), it is shown that $\mathbf{h}_{\text{SDF}}$ and $\mathbf{h}_{\text{MACE}}$ are special cases of $\mathbf{h}_{\text{OTF}}$, with $\alpha = 0$ and $\alpha = 1$ respectively.

2.1.5 Unconstrained MACE (UMACE), Maximum Average Correlation Height (MACH), and Unconstrained Optimal Tradeoff SDF (UOTSDF)

All CCFs rely on EOC constraint to training images, and this determines the resulting filters are likely to over-fit the training data. UCFs discard such hard constraint, and only require statistical and average response from training images.

For instance, UMACE filter endeavors to maximize the ratio of the average correlation height (ACH) at the origin to the ACE value. Maximizing ACH is designed for the purpose of making big correlation value at the origin, while minimizing ACE

lowers the output at other places as discussed in MACE filter. ACH is defined in Eq. (2.17), and it also requires target-in-center training images. Formally, UMACe is optimized by maximizing the objective function $J_{\text{UMACE}}(h)$ shown in Eq. (2.18), with resulting filter expressed in Eq. (2.19).

$$\text{ACH} = \frac{1}{N} \sum_{i=1}^N \mathbf{h}^+ \mathbf{x}_i = \mathbf{h}^+ \mathbf{m} \quad (2.17)$$

$$J_{\text{UMACE}}(h) = \frac{|\text{ACH}|^2}{\text{ACE}} = \frac{\mathbf{h}^+ \mathbf{m} \mathbf{m}^+ \mathbf{h}}{\mathbf{h}^+ \mathbf{D}_x \mathbf{h}} \quad (2.18)$$

$$\mathbf{h}_{\text{UMACE}} = \mathbf{D}_x^{-1} \mathbf{m} \quad (2.19)$$

Instead of using ACE, MACH filter uses Average Similarity Measure (ASM) as an alternative. Equations (2.20) and (2.21) present the formal expression of ASM. Combing these two equations and putting $\mathbf{h}^+$ and $\mathbf{h}$ in Eq. (2.20) into the summation of Eq. (2.21), the whole item inside the summation operator represents the sum of squared distance over all frequency components between the i th image's correlation output and average correlation output. It would be preferable that by passing the filter, the training images' correlation outputs have lower disparities. Basically, ASM measures the variance of correlation output, or in other words, it tells how divergent the correlation planes are. In summary, $\mathbf{S}_x$ describes the variance of training images at individual frequency component before passing the filtering, and ASM describes the overall variance of all frequencies after the filtering.

$$\text{ASM} = \mathbf{h}^+ \mathbf{S}_x \mathbf{h} \quad (2.20)$$

$$\mathbf{S}_x = \frac{1}{N} \sum_{i=1}^N (\mathbf{X}_i - \mathbf{M})^+ (\mathbf{X}_i - \mathbf{M}) \quad (2.21)$$

Over all, MACH is optimized by maximizing the objective function $J_{\text{MACH}}(h)$ shown in Eq. (2.22), with resulting filter expressed in Eq. (2.23).

$$J_{\text{MACH}}(h) = \frac{|\text{ACH}|^2}{\text{ASM}} = \frac{\mathbf{h}^+ \mathbf{m} \mathbf{m}^+ \mathbf{h}}{\mathbf{h}^+ \mathbf{S}_x \mathbf{h}} \quad (2.22)$$

$$\mathbf{h}_{\text{MACH}} = \mathbf{S}_x^{-1} \mathbf{m} \quad (2.23)$$

Through Eq. (2.19) and (2.23), it can be observed that the average image $\mathbf{m}$ constitutes the basis of both UMAC and MACH. With the basis, UMAC emphasizes on high frequencies because normally their power spectrum are smaller, resulting in higher weights in $\mathbf{D}_x^{-1}$. On the other hand, MACH suppresses these frequencies that have bigger variance caused by noise or other factors. Similar to OTF, UOTSDF filter is a compromise filter between UMAC and MACH. Equation (2.24) gives the definition of UOTSDF.

$$\mathbf{h}_{\text{UOTSDF}} = (\alpha \mathbf{D}_x + \sqrt{1 - \alpha^2} \mathbf{S}_x)^{-1} \mathbf{m} \quad (2.24)$$

2.1.6 Average of Synthetic Exact Filters (ASEF) and Minimum Output Sum of Squared Error (MOSSE)

ASEF and MOSSE are two recently proposed UCFs. They suffer from neither the over-fitting problem nor the problem of non-centered training images. Differently, the correlation plane g_i of the i th image (x_i) is synthesized as a 2D Gaussian with peak at target point (r_i, c_i) , that is,

$$g_i = e^{-\frac{(r-r_i)^2 + (c-c_i)^2}{\sigma^2}} \quad (2.25)$$

where σ is the standard deviation, r and c are row and column indexes of the target. Then the goal is to get a filter h_i that can exactly map the image x_i to g_i . Such a filter is so-called exact filter, and can be easily computed in Fourier domain. In frequency domain, we have $\mathbf{g}_i = \mathbf{x}_i \odot \mathbf{h}_i^*$, so the filter can be inversely computed as:

$$\mathbf{h}_i^* = \frac{\mathbf{g}_i}{\mathbf{x}_i} = \frac{\mathbf{g}_i \odot \mathbf{x}_i^*}{\mathbf{x}_i \odot \mathbf{x}_i^*} \approx \frac{\mathbf{g}_i \odot \mathbf{x}_i^*}{\mathbf{x}_i \odot \mathbf{x}_i^* + \epsilon} \quad (2.26)$$

where ϵ is a small constant to avoid zero divisions. In principle, each training image corresponds to a specific exact filter which has no generality ability to other images.

Theoretically, ASEF is gained by averaging the exact filters of all training images to retain common characteristics. The formal formula of ASEF is:

$$\mathbf{h}_{\text{ASEF}} = \frac{1}{N} \sum_{i=1}^N \mathbf{h}_i = \frac{1}{N} \sum_{i=1}^N \frac{\mathbf{x}_i \odot \mathbf{g}_i^*}{\mathbf{x}_i \odot \mathbf{x}_i^* + \epsilon} \quad (2.27)$$

On the other hand, MOSSE is optimized by minimizing the sum of squared error between the real correlation output and synthesized output. The objective function takes the following form:

$$J_{\text{MOSSE}}(h) = \sum_{i=1}^N |\mathbf{x}_i \odot \mathbf{h}^* - \mathbf{g}_i|^2 \quad (2.28)$$

Taking zero-derivative optimization process, the resultant MOSSE filter is defined as:

$$\mathbf{h}_{\text{MOSSE}} = \frac{\sum_{i=1}^N \mathbf{x}_i \odot \mathbf{g}_i^*}{\sum_{i=1}^N \mathbf{x}_i \odot \mathbf{x}_i^*} \approx \frac{\sum_{i=1}^N \mathbf{x}_i \odot \mathbf{g}_i^*}{\sum_{i=1}^N \mathbf{x}_i \odot \mathbf{x}_i^* + \epsilon} \quad (2.29)$$

In summary, the design theories of leading CCFs and UCFs have been described. Generally, the shortcomings of CCFs mainly include three points: (i) computationally intensive and even risky to compute matrix inverse as shown in Eqs. (2.11), (2.15) and (2.16), (ii) the filters are likely to over-fit training images due to EOC constraint, (iii) the training images are required to have the target points in the center of images. UCFs optimize statistical measures, so they relax over-fitting problem. However, traditional UCFs such as UMACE, MACH and UOTSDF still face with the issue described in item (iii). Recent UCFs including ASEF and MOSSE remove these limitations by generating a Gaussian output for each training image.

2.1.7 An Overview of Common Object Detection Methods

In general, object detection has a wide variety of methods. Such diversity is resulted from two aspects. On one hand, it is due to the diversity of the word “object” itself.

On the other hand, the “detection” approaches for the same kind of objects can differ greatly. Because of the diversity, so far there is no generalized method that is able to deal with all kinds of circumstances. Different objects usually come with particularly tailored object detection methods. This subsection will introduce the common strategies in object detection.

2.1.7.1 Simple Features for Specified Objects

Some objects can be easily distinguished from the background by exploiting their particular characteristics such as colors and shapes. For instance, it is not difficult to detect the white football and players wearing in black and red out of the big green field. For another example, human eyeballs can be easily located at the darkest pixel in the face images under controlled condition. In another example, to detect a red traffic cone, the red regions are first segmented out based on color thresholding, and then the triangular contour is examined to confirm the detection.

Often, people naturally tend to think of such simple features at the early stage of developing detection algorithm for a new object. This kind of method is simple and fast to implement, which allows us to estimate the difficulty of the detection and accordingly plan for more complicated algorithms. In practice, methods in this type have many limitations and often the performance is not that satisfactory in real-world conditions. First, sometimes it is hard to find such a distinct and unique characteristic for most objects under uncontrolled environment. In addition, those simple features are usually susceptible to illumination, motion blur, and noises. Continuing with the example of eyeball detection, it may often happen that the eyebrows are detected as the darkest point. In a better way, such specified features are often adopted as the early processing steps before other sophisticated algorithms so as to greatly boost the speed and reduce false alarms.

2.1.7.2 Search by Sliding Window

A very common way for object detection is to exhaustively search the input image, and predict the location of target from the response map. The template based method described in Section 2.1.2 is one simple example. Generally, the input image is scanned by a sliding window at different scales and locations, and each patch covered under the sliding window is investigated by a filter, predictor or classifier to get a response or probability of being the target. Finally, the object is found by applying certain threshold or criteria to the response image.

Similar to simple templates, correlation filters can be thought of a particular type of sliding window algorithms. The filters in spatial domain and frequency domain are interchangeable, hence we can think about them in spatial domain. Different correlation filters differ on how to construct the filter itself. Upon using it to detect targets, it could be viewed as a sliding window method: the response at each pixel is obtained by correlating or convolving its neighborhood (same size to the filter) with the trained filter.

Many other algorithms focus on using more advanced feature extraction and classification techniques in order to increase the detection performance. At the same time, the computational burden is higher and it needs sophisticated optimization to enhance the speed. The following several paragraphs will introduce several very successful examples in this scope.

The first example is Viola and Jones Cascade face detector [7], which is an appearance-based detector and can be well extended to many other objects. Basically, the success of Cascade detector depends on a combination of three key components: (i) Haar features, and *integral image* which speeds up the calculation of Haar features, (ii) AdaBoost algorithm, which selects the most discriminative features and constructs a bunch of weak classifiers; and (iii) a cascaded structure of

boosted classifiers, which discards the majority of background quickly while maintains most possible candidate regions. In addition, small decision trees constitute the building blocks of the classifiers in the implementation.

Haar features are defined as the difference between intensity sums of neighboring areas. The values of Haar features indicate certain local characteristic of the object. For instance, a Haar feature can provide an indication of where the border between a bright and dark region lies. As Haar features operate on arbitrary rectangles, it can yield a huge redundant set of around 160,000 instances for a 24×24 resolution. To boost the speed, *integral image* concept was proposed in [7], and it enables the quick calculation of the sum of pixels in arbitrary rectangular region.

Although integral image considerably alleviates the computing loads, completely computing of the vast number of Haar features is still prohibitively expensive and unnecessary. Therefore, AdaBoost algorithm is adopted to select the most discriminative features form the feature pool, and each selected feature corresponds to a weak face/non-face classifier. Also a strong classifier is constructed by combining a few of these weak classifiers. Generally, the more number of weak classifiers are used, the more powerful the strong classifier is, and naturally more time is required. Based on this hypothesis, a cascaded structure is employed to quickly discard the majority of irrelevant regions and focus more attention on the possible candidate regions. Figure 2.3 shows the cascaded architecture. In each stage, a boosting classifier is trained to remove most outliers and maintain the positive targets. At last, only the real face patches can reach the final stage. Such strategy could achieve good detection speed as well as accuracy.

The second example is Dalal and Triggs's Histogram of Oriented Gradients (HOG) for human detection [41]. The major contributions focus on building robust feature sets (HOG features) over the sliding window. To extract HOG features over

Figure 2.3: Architecture of Cascade detector

windows, the main steps include: gamma normalization, gradients computation, weighted vote in spatial and oriented cells, contrast normalization over overlapping cells and concatenation of HOGs over detection window. Based on the HOG features, linear Support Vector Machines is applied to classify the window as human or non-human. Similar to the Cascade method, HOG based algorithm is also appearance based, and requires the human body to be upright fully visible.

Another example is Random Forests for head pose estimation from depth data [42]. Each non-leaf node in a single decision tree represents a binary test on the difference between the average values of two arbitrary rectangular areas. For each regression tree, the non-leaf nodes direct an input sample towards a leaf, where a real-valued multi-variant distribution of the pose parameters is stored. Given an input depth image, patches are densely sampled and fed into all trees in the forest. The forest combines the results from all leaves to discriminate whether the patch belongs to a head as well as produce a pose estimation.

In summary, similar methods as aforementioned three examples commonly require fairly high computational costs. It will save much efforts if simple features as described in previous subsection could help to focus on these candidate regions with high possibilities of being the target.

2.1.7.3 Keypoint Matching via Local Descriptors

Another research line of object detection is through keypoint matching between input image to stored database. This normally requires the image contains rich non-periodic texture so that enough number of stable keypoints can be detected to extract point descriptors. Often times two phases are included: offline training and online detection. Offline training learns these outstanding keypoints from a initial frame or many images, and stores their feature vectors into the database. In online detection phase, keypoints are detected from input image and their feature vectors are extracted to match with the database. Enough number of matched correspondences enables to declare a confident object detection as well as estimate the corresponding pose.

The pipeline of keypoint matching consists of two modules: local detectors [43] and local descriptors [44]. The former refers to the techniques to find the keypoints, while the latter refers to how to extract the feature vector from the patch around the detected keypoint. Typical local detectors include Harris corner detector, Laplacian of Gaussian (LoG), and Difference of Gaussians (DoG). Local descriptors will be further introduced in Subsection 2.2.3.

Over the past decade, the most popular keypoint algorithms are the Scale Invariant Feature Transform (SIFT) [45] and its variant Speeded Up Robust Features (SURF) [46]. SIFT and SURF can be used as both a keypoint detector and a keypoint descriptor. Besides object detection, they have a broader range of applications such as scene recognition, pose estimation, and visual tracking.

Another wise algorithm is to learn randomized trees to find discriminative points and simultaneously act as the descriptor [47]. Binary tests in nodes of the decision trees use the intensity comparison between randomly selected pixels from the neighborhood of the keypoint. Original authors have incorporated the ideas into

a real-time system that could detect planar, non-planar and deformable objects. In addition, it also estimates the pose of rigid objects and the deformations of the others.

2.2 Feature Extraction Methods

The “feature” might be one of the most commonly used term in computer vision. What is a feature and what constitutes a feature? This is very hard to give a clear definition. Generally, it refers to what information we can extract from an image to represent the image or particular object. What constitutes a feature varies greatly depending on specific applications. Typically, a pattern recognition system contains three stages as displayed in Figure 2.4. First, preprocessing algorithms such as geometric or photometric normalization are often applied to the raw data according to specific tasks. Second, feature extraction techniques are adopted to derive a discriminative representation from the preprocessed images, so as to making is easier to distinguish between different classes. The final stage involves using machine learning techniques to label the images based on the image representation of previous stage. Feature extraction is a critical part to realize successful pattern classification.

Figure 2.4: Common architecture of pattern recognition system

This part presents a brief review on the common image feature representations in computer vision field, giving a particular emphasis on the applications of facial image analysis covered in this thesis. Many particular approaches may be only applicable to particular tasks. For instance, spatial-temporal features can be extracted to describe the muscle movements for facial expression recognition [11].

Literature surveys to these specific applications will be further introduced in the application chapters: Chapter 4 and Chapter 6. Due to the breadth and voluminous publications on features, this work could not cover all aspects. Besides, it is hard to classify existing methods to strict categories, as many methods are mixed of various aspects. The following review is organized by abstracting many primary keywords in the feature world.

2.2.1 Intensity, Colors

Raw intensity (brightness) might be the earliest feature for image representation, and is still commonly used. In facial analysis, a typical example is to exploit intensity characteristics to find face components such as eyebrows, eyes and mouth. This often requires the input of geometrical and intensity properties of the specific components. For instance, eye center is often searched at the location with lowest intensity. In addition, image histogram is another frequently applied representation, and it extracts the global statistical distribution of the raw pixels.

Color is another important feature. It is particularly feasible in describing these images or objects with distinguished colors. One typical example is segmentation of an object or region of interest. Face detection aims at segmenting the face regions out of the background. Many early methods are often based on the skin color to find valid face region [48]. There are various color modules, such as RGB, HSV and YCbCr.

2.2.2 Gradients, Edges

While intensity and color are very susceptible to lighting and viewpoint changes, gradients provide a more stable feature. Given an intensity image $I(x, y)$, we form the gradient vector by computing the partial derivatives at every pixel in the x

direction and y direction, that is,

$$\Delta I = \left\{ \frac{\partial I}{\partial x}, \frac{\partial I}{\partial y} \right\} \quad (2.30)$$

Another way is to calculate the magnitude (M) and orientation (θ):

$$\begin{aligned} M &= \sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2} \\ \theta &= \arctan\left(\frac{\partial I}{\partial y} / \frac{\partial I}{\partial x}\right) \end{aligned} \quad (2.31)$$

On top of gradients, many complicated high-level features have been proposed. One famous representation is the Histogram of Oriented Gradients (HOG), which accumulates the gradient orientation bins that are weighted by corresponding magnitudes over cells. The most successful application of HOG is people detection by combining a SVM learning architecture [41]. Another well-known feature is the Scale Invariant Feature Transform (SIFT) [45], which belongs to the keypoint descriptor. SIFT is computationally intensive, and usually used to describe interest points for finding correspondences between two images.

The concept of edges is closely relative to gradients. An edge is defined as these points which have sharp intensity changes to the neighborhood or large gradients. Various algorithms could be employed to detect edges in an image, including:

- Gradient operators
- Sobel operators
- Canny edge detector
- Krish operators
- Laplacian operators

- Prewitt operators

Commonly, edges can well describe the contours of natural objects that normally have distinguished intensity distributions to the background. For example, it would be possible to extract the edges that represent the face contour and eye lids, and use these information for subsequent processing. Taking ESR for example, to distinguish open eyes with closed eyes, Hough Transform could be applied to detect the presence of iris circle for open-eye images after extracting the contours [49].

2.2.3 Keypoint Descriptors

Local keypoint descriptor is one hottest type of feature extraction techniques in the past decades, and much progress has been made. Even though most applications of facial analysis seldomly adopt this type of technique, it is worth introducing it as an important part of feature extraction literature. These local descriptors tend to find the most distinctive keypoints among different objects. But for FER and GR, most of the salient keypoints appear in all the facial images even with different expressions or genders, so local descriptors are not very suitable for these applications.

Keypoints in an image are sometimes referred to as interest point or corner points. Keypoint involves two aspects: detection and description, which correspondingly refer to keypoint detector and keypoint descriptor. The local descriptors are obtained by extract a feature vector from a local patch centered at the detected keypoints. Many typical keypoint descriptors are:

- Scale-Invariant Feature Transform (SIFT) [45]
- Speeded Up Robust Features (SURF) [46]

- PCA-SIFT [50]
- Gradient Location-Orientation Histogram (GLOH) [44]
- Fast Retina Keypoint (FREAK) [51]
- Oriented BRIEF (ORB) [52]

Among them, SIFT is undoubtedly the most famous local descriptor that has profoundly influenced computer vision community for the past decade, particularly on object detection and recognition.

On top of keypoint descriptors, bag of keypoints [53] has been proposed to build a visual words vocabulary using a set of extracted descriptors from image patches. It has gained great success in object categorization. This method requires to combine machine learning techniques such as clustering to train the vocabulary.

2.2.4 Holistic Feature Representation

Holistic method is frequently used in facial representation. A holistic representation of an image is formed by lexicographically reordering all the pixels to a vector. In this way, the image could be considered as a point in high-dimensional feature space. For 100×100 images or patches, the corresponding dimensionality reaches 10,000, within which lots of redundancies. The redundancy is mainly due to the adjoining pixels are highly correlated.

The large dimensionality of raw feature space prohibits the use of learning process to be carried out. One common way to handle it is to first employ a dimensionality reduction techniques such as Principal Component Analysis (PCA) [54] to project the raw data to low-dimensional feature subspaces. Another notable subspace method is Linear Discriminant Analysis (LDA) [55]. PCA and LDA are

separately well recognized as eigenfaces and fishfaces in facial image analysis, and have been immensely studied in the past two decades.

As displayed in Figure 2.5, the goal of PCA is to find a subspace whose basis vectors correspond to the maximum-variance directions in the original space (Figure 2.5). On the other hand, LDA finds a set of basis vectors that best split samples from different classes in original feature space. It aims to maximize the between-class distance while minimize the within-class distance.

Figure 2.5: Diagram for PCA and LDA

When talking about feature space, it has to mention another very hot research topic in machine learning and computer vision in recent years: unsupervised feature learning and deep learning. Instead of finding feature space by linear transforms, this type of method uses deep networks to learn a hierarchical feature representation that is highly nonlinear to original raw data. Recent works show deep learning methods have demonstrated state-of-the-art performance in many applications such as speech recognition, visual recognition. Such feature representations mainly rely on machine learning techniques. It forms a big research area, interested readers can refer to the works of the pioneers such as Geoffrey Hinton [56, 57], Yann Lecun [58] and Yoshua Bengio [59].

2.2.5 Local Feature Extraction

Gottumukkal [60] argued that some of local facial features could keep consistent even with varying poses, lighting directions and facial expressions, and advised to partition the face into smaller sub-images. Accordingly, local feature extraction is motivated to effectively represent these local regions. This part mainly introduces two families of most commonly used local feature techniques in face area: gabor wavelets based features, and Local Binary Patterns and its variants.

2.2.5.1 Gabor Wavelets

In 1978, Granlaund first introduced the 2D Gabor elementary function [61]. 2D Gabor wavelets were first applied for face recognition by Lades et al. [62]. Two distinct characteristics of Gabor wavelets are spatial locality and orientation selectivity. It has been indicated that Gabor wavelets based face representation should be robust to variations because of lighting and expression changes [63, 64]. Its outstanding performance in recognition tasks has been repeatedly proved [65–67].

A 2D Gabor kernel is defined by a sinusoidal plane wave multiplied by a 2D Gaussian function. These two components can be expressed in a complex form:

$$g(x, y; \lambda, \theta, \psi, \sigma, \gamma) = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) \exp\left(i\left(\frac{2\pi x'}{\lambda} + \psi\right)\right) \quad (2.32)$$

where

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad (2.33)$$

In the equation, λ is the wavelength of the sinusoidal signal, θ is the orientation perpendicular to the parallel stripes of the sinusoidal plane, ψ is the phase offset, σ is the standard deviation of the Gaussian function and γ is the spatial aspect ratio that decides the ellipticity of the Gabor wave.

In practice, a bank of Gabor filters are used to convolve with the image, each filter resulting in a response map of the same size as image. These Gabor filters are tuned to various scales and orientations. Therefore, one disadvantage is the intensive computational time and memory to convolve face images with such a bank of Gabor filters. In addition, each pixel will be represented by a set of response values as many as the number of filters.

2.2.5.2 Local Binary Patterns and Its Variants

Local Binary Patterns is a more recently proposed local feature technique. Since first introduced for texture classification in 1996 [68], the Local Binary Patterns (LBP) has gradually grown to a widely recognized feature in various applications including texture classification [9], face recognition [10], and facial expression analysis [12]. Since the proposed features in this work are closely related to LBP family, the following contents expand the details of relevant techniques.

The LBP operator was originally designed for texture classification and has proved to be highly discriminative [9]. In personal viewpoint, LBP operator can be considered as a novel 3×3 nonlinear filter. Similarly, each pixel in an image will be assigned with a response value. But unlike linear filters such as average, Gaussian and Sobel, the LBP output is no longer a weighted sum of the values of neighboring pixels. Instead, the response of LBP filter depends on intensity comparison results between center pixel and eight surrounding pixels. In details, given a center pixel (x_c, y_c) , first eight circular bits are obtained by thresholding eight surrounding pixels' intensity with the center pixel as shown in Figure 2.6. This ordered bit string is then transformed to a decimal number by multiplying the p th bit with weight 2^p and adding all together. The mapping function of LBP

is defined as:

$$LBP(x_c, y_c) = \sum_{p=0}^7 \text{sign}(i_p - i_c)2^p, \quad \text{sign}(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases} \quad (2.34)$$

where i_p denotes the intensity of the p th pixel, and i_c is the value of center pixel.

Clearly, the number of LBP patterns (labels) is $2^8 = 256$.

Figure 2.6: An example of LBP operator

LBP is simple and fast to implement. After all, compared to raw pixel intensity, LBP is invariant to monotonic lighting changes, making it suitable for demanding image analysis tasks. The idea of applying LBP to face representation is motivated by the fact that face image can be considered as a composition of micro-structures that can be well described by LBP patterns.

In [9], LBP was further extended to circularly symmetric neighborhood with different scales and neighbors. The notation (P, R) denotes the supporting neighborhood with P neighbors at radius R . Figure 2.7 gives three examples. Naturally, adding one more neighbor will double the number of patterns. Ojala et al. also proposed another extension in [9]: *uniform patterns*. In their work, they noticed that a small subset of LBP patterns contains most of the important textures. These patterns are called the uniform patterns which depict majority of primitive micro-structures such as edges, corners, and highlight spots. Basically, uniform patterns are these patterns that contain at most two $1-0$ or $0-1$ transitions in the circular bit string. Each uniform pattern is designated with a separate label and all other

non-uniform patterns share an extra label. In total, there are $58 + 1 = 59$ patterns. Therefore, the modified feature is named Uniform Local Binary Patterns (ULBP). Since LBP earned big reputation for successful use in facial representation, a great number of works have been done to extend its scope in terms of feature variants, integration with different classifiers or new applications. A comprehensive research progress of LBP was presented in [16] including face analysis.

Figure 2.7: Circular supporting neighborhood patterns with different radius and pixel numbers

One drawback of LBP is that the operator is very sensitive to white noise and non-monotonic illumination changes [15]. One possible solution is to adopt another texture feature: Local Directional Patterns (LDP) [15, 69]. The LDP also assigns an eight-bit string to each pixel of input image. Differently, these binary bits are calculated by ordering the edge responses of the pixel at different directions. For such purpose, these edge responses are evaluated through convolving the local intensity pattern with Kirsh masks in eight orientations ($M_0 - M_7$) as shown in Figure 2.8. The resulting eight responses represent edge significance in eight directions. The authors believe the k most prominent directions are the k directions with highest responses, having higher probability of presence of corner or edge. LDP operator sets the binary bits corresponding to three most prominent directions as one, and

the rest as zero. Totally, there are ${}^8C_3 = 56$ patterns. Figure 2.9 illustrates the generating process of LDP code. LDP is more stable to non-monotonic intensity changes and white noise. One shortcoming is the increasing computational cost from calculating and ranking the eight Kirsch edge responses.

Figure 2.8: Kirsch edge masks in eight directions

Figure 2.9: Generating process of LDP code

Chapter 3

Correlation Filters

This chapter describes a new class of correlation filters called Generalized Unconstrained Filters (GUF). It contains the construction process of GUF, and an analysis of the properties for GUF. Afterwards, proofs will be presented to show GUF is the universal version of many prior filters.

3.1 Introduction

Without loss of generality, filtering process can be divided into two stages: training (synthesis) stage, and testing (use) stage. Obviously, training stage is the core technical part, and also constitutes the motivations of various filter design methodologies. To detect a particular type of object, training samples with certain distortions are provided to learn the composite filters. Clearly, high detection accuracy on the testing images is the ultimate criterion of filter design. However, such information is unknown during training. One feasible way in design is to optimize performance criteria such as high peak sharpness, high output similarity score, and low output noise variance on the training images. The following content is going to describe these criteria and corresponding measuring metrics.

It is desirable that a sharp peak is produced at the target point in the correlation output. This not only requires a large response value at the target, but also needs

to suppress strong responses at other places to prevent imposters. This is called peak sharpness measure, *i.e.*, relative height of peak compared to surrounding area. Peak sharpness can be used to judge the detection quality, or infer the confidence level of the peak. Peak to sidelobe ratio (PSR) is a frequently adopted metric to measure peak sharpness, with the definition:

$$\text{PSR} = \frac{\text{peak} - \mu}{\sigma} \quad (3.1)$$

where μ and σ represent the mean and standard deviation of sidelobe region, and “peak” is the value of peak point. Sidelobe refers to these areas in the output excluding an 11×11 patch centered at the peak.

Output similarity reflects the filter’s capability to converge the distorted training images to a solid centroid in the output space. The higher the output similarity score is, the closer the training data distributes after projection by the filter, therefore the less the filter will be influenced by distortions in the training images. The average similarity measure expressed in Eq. 2.20 can be used to measure the output divergence, and accordingly indicates the output similarity.

Another performance criterion is output noise variance, which defines how the filter deals with additive noise in the training images. According to MVSDF [32], assume the input noise is white noise, the resulting output noise variance can be expressed as:

$$\text{ONV} = \mathbf{h}^T \mathbf{I} \mathbf{h} = \mathbf{h}^T \mathbf{h} \quad (3.2)$$

By minimizing ONV, the filter can reduce the effects of noise at utmost.

3.2 Generalized Unconstrained Filters (GUF)

To present the derivation of GUF, facial landmark localization on right eye center is used as the study example. Figure 3.1 shows the general architecture of method-

ology to detect the right eye center in near frontal faces. The full face images are utilized to train the GUF filters rather than target-in-center face regions.

Figure 3.1: General architecture of facial landmark localization, right eye center is used for illustration

Essentially, the optimization process of GUFs takes three performance criteria into consideration, and adopts relevant constraints to control the filter. In the first criterion, the GUF filter's output correlation planes are required to have a specified shape function f in spatial domain. In principle, this synthetic shape function can be any two-dimensional single-peaked functions such as Dirac delta function, Triangular function and Gaussian function. Figure 3.2 gives many examples in 1D form. It needs to mention that the peak of the synthesized plane for each training image must be shifted to the location where the target point is located. Suppose f_i denotes the designed output for the i th image, and $\mathbf{f}_i$ denotes the vectorized form of its Fourier Transform. Partially, GUF filter is to minimize the mean squared error (MSE) term in frequency domain:

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^N \|\mathbf{X}_i \mathbf{h}^* - \mathbf{f}_i\|^2 \quad (3.3)$$

Basically, minimizing MSE endeavors to drive the filter to produce an ideal single-peaked output from the training image. Historically, Gaussian-MACE [33] is optimized by minimizing the MSE item subject to EOC constraints, while MOSSE filter is directly resulted from minimizing the MSE [18].

Figure 3.2: This figure shows many examples of 1D single-peaked function. Their two-dimensional forms can be used as the synthetic correlation plane.

The second criterion is to control the divergence of correlation planes. In mathematics, this can be described as variance of correlation output (VCO):

$$\text{VCO} = \frac{1}{N} \sum_{i=1}^N \|\mathbf{X}_i \mathbf{h}^* - \mathbf{M} \mathbf{h}^*\|^2 \quad (3.4)$$

Actually, VCO is identical to the ASM in Eq. (2.20). Minimizing this means the correlation outputs of training images resemble each other the most. One thing worth mentioning is that the target points of training images should be circularly shifted to any common location $p(r_p, c_p)$ as shown in Figure 3.3. The intention is to roughly align the training images with common reference point so as to compute the mean image $\mathbf{M}$ in Eq. (3.4). Considering the border issue during image shifting,

images are supposed to be circularly repeated and windowing process should be applied to produce approaching-zero values at image edges beforehand. Feasible window functions include Cosine window, Hann window and Hamming window. In fact, periodic repetition assumption and windowing are also highly required to reduce edge distortions during Fourier Transform [6, 70]. Accordingly, peaks of synthesized correlation planes should also be placed at the common pixel $p(r_p, c_p)$, so all $\mathbf{f}_i$'s are identical, and use common symbol $\mathbf{f}_p$ to denote it, that is $\mathbf{f}_i = \mathbf{f}_p$.

	(1) Original image	(2) Preprocessed image	(3) Keypoints can be shifted to any fixed pixel p	(4) Ideal output f_i ($f_i = f_p$)
x_1
x_2

Figure 3.3: Training images and synthetic correlation output for GUF training. (1) original images; (2) normalized images by cosine windowing and other preprocessing steps. Windowing smooths image edges, which reduces edge distortions during Fourier transform and prevents abrupt intensity changes in images after shifting (the images are assumed to periodically repeated in space); (3) resultant images by circularly shifting target points to any arbitrarily common location $p(r_p, c_p)$; (4) desired correlation planes f_i 's that correspond to the images in (3). These planes are identical because their peaks commonly locate at pixel $p(r_p, c_p)$. f_p denotes the common output.

The last part of GUF minimizes the filter energy $\mathbf{E}_h$, the sum of squares of filter's pixel values. That is,

$$\mathbf{E}_h = \|\mathbf{h}\|^2 \quad (3.5)$$

For one reason, this term can achieve good performance against white noise as described in [17, 32, 33]. Suppose the input image is corrupted by additive zero-mean white Gaussian noise n with power spectral density $P_n(u, v)$. Using the linearity property of DFT, the resulting correlation output will have variance $\mathbf{h}^+ \mathbf{P} \mathbf{h}$ due to the extra noise, where $\mathbf{P}$ is the diagonal matrix obtained from $P_n(u, v)$. It depicts output noise variance at all pixels [17]. For white Gaussian noise, the spectral power $\mathbf{P}$ becomes $\sigma_n^2 \mathbf{I}$, where σ_n is the standard deviation of noise. Therefore, a low filter energy $\mathbf{E}_h$ is preferred. For another important reason, adding this part could avoid division by zero for those frequency components that have very small spectral power ($\mathbf{D}_x$) and variance ($\mathbf{S}_x$) in training images as shown in the formula of the final filter in Eqs. (3.14) and (3.15) (the role of term $\mathbf{I}$). This will be further described in Section 3.3.

Overall, the objective function is expressed in Eq. (3.6), which is a weighted sum of the three aforementioned performance measures.

$$\begin{aligned} J_{\text{GUF}}(\mathbf{h}^*) &= \alpha \text{MSE} + \beta \text{VCO} + \gamma \mathbf{E}_h \\ &= \alpha \frac{1}{N} \sum_{i=1}^N \|\mathbf{X}_i \mathbf{h}^* - \mathbf{f}_i\|^2 + \beta \frac{1}{N} \sum_{i=1}^N \|\mathbf{X}_i \mathbf{h}^* - \mathbf{M} \mathbf{h}^*\|^2 + \gamma \|\mathbf{h}\|^2 \end{aligned} \quad (3.6)$$

The weights α , β and γ are non-negative, ranging from 0 to 1. In fact, the filter's performance relies on only the relative scales of these weights rather than their absolute values. The GUF filter is the solution to minimize this objective function. The function is a real-valued function of a vector of complex variables. The solution can be obtained by getting its stationary point.

To get the solution of minimizing Eq. (3.6), first set $\tilde{\mathbf{h}} = \mathbf{h}^*$. Then, J_{GUF} becomes:

$$J'_{\text{GUF}}(\tilde{\mathbf{h}}) = \alpha \frac{1}{N} \sum_{i=1}^N \|\mathbf{X}_i \tilde{\mathbf{h}} - \mathbf{f}_i\|^2 + \beta \frac{1}{N} \sum_{i=1}^N \|\mathbf{X}_i \tilde{\mathbf{h}} - \mathbf{M} \tilde{\mathbf{h}}\|^2 + \gamma \|\tilde{\mathbf{h}}\|^2 \quad (3.7)$$

Expand the equation, we have:

$$\begin{aligned}
 J' &= \alpha \frac{1}{N} \sum_{i=1}^N (\mathbf{X}_i \tilde{\mathbf{h}} - \mathbf{f}_i)^+ (\mathbf{X}_i \tilde{\mathbf{h}} - \mathbf{f}_i) + \beta \frac{1}{N} \sum_{i=1}^N (\mathbf{X}_i \tilde{\mathbf{h}} - \mathbf{M} \tilde{\mathbf{h}})^+ (\mathbf{X}_i \tilde{\mathbf{h}} - \mathbf{M} \tilde{\mathbf{h}}) + \gamma \tilde{\mathbf{h}}^+ \tilde{\mathbf{h}} \\
 &= \alpha \frac{1}{N} \sum_{i=1}^N (\tilde{\mathbf{h}}^+ \mathbf{X}_i^+ \mathbf{X}_i \tilde{\mathbf{h}} - \tilde{\mathbf{h}}^+ \mathbf{X}_i^+ \mathbf{f}_i - \mathbf{f}_i^+ \mathbf{X}_i \tilde{\mathbf{h}} + \mathbf{f}_i^+ \mathbf{f}_i) + \beta \tilde{\mathbf{h}}^+ \left(\frac{1}{N} \sum_{i=1}^N (\mathbf{X}_i - \mathbf{M})^+ (\mathbf{X}_i \right. \\
 &\quad \left. - \mathbf{M}) \right) \tilde{\mathbf{h}} + \gamma N \tilde{\mathbf{h}}^+ \tilde{\mathbf{h}} \\
 &= \alpha \frac{1}{N} \sum_{i=1}^N (\tilde{\mathbf{h}}^+ \mathbf{X}_i^+ \mathbf{X}_i \tilde{\mathbf{h}} - \tilde{\mathbf{h}}^+ \mathbf{X}_i^+ \mathbf{f}_i - \mathbf{f}_i^+ \mathbf{X}_i \tilde{\mathbf{h}} + \mathbf{f}_i^+ \mathbf{f}_i) + \beta \tilde{\mathbf{h}}^+ \mathbf{S}_x \tilde{\mathbf{h}} + \gamma \tilde{\mathbf{h}}^+ \tilde{\mathbf{h}}
 \end{aligned} \tag{3.8}$$

Finding the stationary point of function J' can be accomplished by setting its first-order derivative to zero, and then solving the equation. However, it is somewhat difficult in our case, because J' is a real-valued function of a vector of complex variables: $\tilde{\mathbf{h}}$. To summarize, the feasible solution is to consider the function as an analytic function of two independent complex vectors: $\tilde{\mathbf{h}}$ and its complex transpose $\tilde{\mathbf{h}}^+$ [71]. Then the conditions for a stationary point are (i) $\nabla_{\tilde{\mathbf{h}}^+} J' = 0$ and (ii) $\nabla_{\tilde{\mathbf{h}}} J' = 0$, where ∇ is the partial derivative operator. In other words, the solution must satisfy both equalities. First evaluate equality (i) as example:

$$\begin{aligned}
 0 &= \nabla_{\tilde{\mathbf{h}}^+} J' \\
 &= \alpha \frac{1}{N} \sum_{i=1}^N (\mathbf{X}_i^+ \mathbf{X}_i \tilde{\mathbf{h}} - \mathbf{X}_i^+ \mathbf{f}_i) + \beta \mathbf{S}_x \tilde{\mathbf{h}} + \gamma \tilde{\mathbf{h}} \\
 &= \alpha \left(\frac{1}{N} \sum_{i=1}^N \mathbf{X}_i^+ \mathbf{X}_i \right) \tilde{\mathbf{h}} - \alpha \frac{1}{N} \sum_{i=1}^N \mathbf{X}_i^+ \mathbf{f}_p + \beta \mathbf{S}_x \tilde{\mathbf{h}} + \gamma \tilde{\mathbf{h}} \\
 &= \alpha \mathbf{D}_x \tilde{\mathbf{h}} - \alpha \mathbf{M}^+ \mathbf{f}_p + \beta \mathbf{S}_x \tilde{\mathbf{h}} + \gamma \tilde{\mathbf{h}} \\
 &= (\alpha \mathbf{D}_x + \beta \mathbf{S}_x + \gamma \mathbf{I}) \tilde{\mathbf{h}} - \alpha \mathbf{M}^+ \mathbf{f}_p
 \end{aligned} \tag{3.9}$$

Note in step 3 of Eq. (3.9), $\mathbf{f}_i = \mathbf{f}_p$ has been used as peaks of all $\mathbf{f}_i$'s are at a common location. Move the item with $\tilde{\mathbf{h}}$ to left hand side, rewrite Eq. (3.9) as:

$$(\alpha \mathbf{D}_x + \beta \mathbf{S}_x + \gamma \mathbf{I}) \tilde{\mathbf{h}} = \alpha \mathbf{M}^+ \mathbf{f}_p \tag{3.10}$$

Thus, the solution for $\nabla_{\tilde{\mathbf{h}}^+} J' = 0$ is:

$$\tilde{\mathbf{h}} = \alpha(\alpha\mathbf{D}_x + \beta\mathbf{S}_x + \gamma\mathbf{I})^{-1}\mathbf{M}^+\mathbf{f}_p \quad (3.11)$$

In similar manner, solve the second equality $\nabla_{\tilde{\mathbf{h}}} J' = 0$ as follows:

$$\begin{aligned} 0 &= \nabla_{\tilde{\mathbf{h}}} J' \\ &= \alpha \frac{1}{N} \sum_{i=1}^N (\tilde{\mathbf{h}}^+ \mathbf{X}_i^+ \mathbf{X}_i - \mathbf{f}_i^+ \mathbf{X}_i) + \beta \tilde{\mathbf{h}}^+ \mathbf{S}_x + \gamma \tilde{\mathbf{h}}^+ \\ &= \alpha \tilde{\mathbf{h}}^+ \left(\frac{1}{N} \sum_{i=1}^N \mathbf{X}_i^+ \mathbf{X}_i \right) - \alpha \frac{1}{N} \sum_{i=1}^N \mathbf{f}_i^+ \mathbf{X}_i + \beta \tilde{\mathbf{h}}^+ \mathbf{S}_x + \gamma \tilde{\mathbf{h}}^+ \\ &= \alpha \tilde{\mathbf{h}}^+ \mathbf{D}_x - \alpha \mathbf{f}_p^+ \mathbf{M} + \beta \tilde{\mathbf{h}}^+ \mathbf{S}_x + \gamma \tilde{\mathbf{h}}^+ \\ &= \tilde{\mathbf{h}}^+ (\alpha \mathbf{D}_x + \beta \mathbf{S}_x + \gamma \mathbf{I}) - \alpha \mathbf{f}_p^+ \mathbf{M} \end{aligned} \quad (3.12)$$

Consequently, the solution for the second equality is:

$$\tilde{\mathbf{h}}^+ = \alpha \mathbf{f}_p^+ \mathbf{M} (\alpha \mathbf{D}_x + \beta \mathbf{S}_x + \gamma \mathbf{I})^{-1} \quad (3.13)$$

It is not difficult to show that $\tilde{\mathbf{h}}$ in Eq. (3.11) and $\tilde{\mathbf{h}}^+$ in Eq. (3.13) are actually the same solution displayed in different forms: one in original form and another in conjugate transpose form. This means the solution can satisfy both zero-derivative conditions in equalities (i) and (ii), so it is the stationary point of function J' . Furthermore, it is observed that the first α in Eqs. (3.11) and (3.13) is only a scalar to the filter and will not affect the performance. Therefore, it is discarded in the filter's final form. Actually, this operation can also help to prevent the filter from being zero even if α is equal to zero. Eventually, replace $\tilde{\mathbf{h}}$ back to $\mathbf{h}^*$, and it leads to the formal formula of GUF filter in Eqs. (3.14) and (3.15).

$$\mathbf{h}_{\text{GUF}}^* = (\alpha \mathbf{D}_x + \beta \mathbf{S}_x + \gamma \mathbf{I})^{-1} \mathbf{M}^+ \mathbf{f}_p \quad (3.14)$$

$$\mathbf{h}_{\text{GUF}} = (\alpha \mathbf{D}_x + \beta \mathbf{S}_x + \gamma \mathbf{I})^{-1} \mathbf{M} \mathbf{f}_p^* \quad (3.15)$$

3.3 Analysis of GUF Filters

Based on Eq. (3.14) or (3.15), GUF filter is composed by five components: average energy plane $\mathbf{D}_x$, variance of training data $\mathbf{S}_x$, identity matrix $\mathbf{I}$, average of training data $\mathbf{M}$, and an optimal correlation output $\mathbf{f}_p$. It is noticed that the first four items are diagonal matrices, therefore the inverse of weighted sum matrix $(\alpha\mathbf{D}_x + \beta\mathbf{S}_x + \gamma\mathbf{I})^{-1}$ can be easily obtained by computing the reciprocals of diagonal elements. In addition, calculation of matrix $\mathbf{S}_x$ can be simplified through computing a combination of $\mathbf{D}_x$ and $\mathbf{M}$ by expanding $\mathbf{S}_x$:

$$\begin{aligned}
 \mathbf{S}_x &= \frac{1}{N} \sum_{i=1}^N (\mathbf{X}_i - \mathbf{M})^+ (\mathbf{X}_i - \mathbf{M}) \\
 &= \frac{1}{N} \sum_{i=1}^N \mathbf{X}_i^+ \mathbf{X}_i - \frac{1}{N} \sum_{i=1}^N \mathbf{X}_i^+ \mathbf{M} - \frac{1}{N} \sum_{i=1}^N \mathbf{M}^+ \mathbf{X}_i + \frac{1}{N} \sum_{i=1}^N \mathbf{M}^+ \mathbf{M} \\
 &= \mathbf{D}_x - \mathbf{M}^+ \mathbf{M} - \mathbf{M}^+ \mathbf{M} + \mathbf{M}^+ \mathbf{M} \\
 &= \mathbf{D}_x - \mathbf{M}^+ \mathbf{M}
 \end{aligned} \tag{3.16}$$

Consequently, GUF filter in Eq. (3.15) can be alternatively rewritten as:

$$\begin{aligned}
 \mathbf{h}_{\text{GUF}} &= [\alpha\mathbf{D}_x + \beta(\mathbf{D}_x - \mathbf{M}^+ \mathbf{M}) + \gamma\mathbf{I}]^{-1} \mathbf{M} \mathbf{f}_p^* \\
 &= [(\alpha + \beta)\mathbf{D}_x - \beta\mathbf{M}^+ \mathbf{M} + \gamma\mathbf{I}]^{-1} \mathbf{M} \mathbf{f}_p^*
 \end{aligned} \tag{3.17}$$

Computing GUF filter using Eq. (3.17) has two advantages. For one thing, it can save extra computational cost to get $\mathbf{S}_x$. For another thing, with the new formula, it is realizable to update the filter with newly added images because $\mathbf{D}_x$ and $\mathbf{M}$ can be effortlessly renewed with new images. The significance of the second advantage will be further explained in Chapter 4.

Another property of GUF filter is shift invariance of peaks, i.e., the filter maintains unchanged no matter where the common location $p(r_p, c_p)$ is set for all peaks (target points). From Eq. (3.17), this statement is supported if we can prove

that $\mathbf{D}_x$, $\mathbf{M}^+\mathbf{M}$, and $\mathbf{M}\mathbf{f}_p^*$ will stay unchanged at different common peak locations. For illustration, Figure 3.4 presents the training images and corresponding desired correlation planes at two peak locations. When target points are shifted to a common pixel $p_1(r_1, c_1)$, the i th image, its Fourier Transform image, and the vectorized form of FT are respectively denoted by $x_{i1}(r, c)$, $X_{i1}(u, v)$, and $\mathbf{x}_{i1}$. Likewise, symbols $x_{i2}(r, c)$, $X_{i2}(u, v)$, and $\mathbf{x}_{i2}$ denote relevant entities when the common pixel is set at $p_2(r_2, c_2)$. As described in aforesaid contents, images are circularly shifted, so the image $x_{i2}(r, c)$ can be viewed as shifting $x_{i1}(r, c)$ by offset $(\Delta r, \Delta c) = (r_2 - r_1, c_2 - c_1)$. That is,

$$x_{i2}(r, c) = x_{i1}(r - \Delta r, c - \Delta c) \quad (3.18)$$

In similar manner, the i th desired correlation planes of these two cases are denoted by $f_{i1}(r, c)$, $f_{i2}(r, c)$ (spatial domain), $F_{i1}(u, v)$ and $F_{i2}(u, v)$ (frequency domain). Also the relation between $f_{i1}(r, c)$ and $f_{i2}(r, c)$ is:

$$f_{i2}(r, c) = f_{i1}(r - \Delta r, c - \Delta c) \quad (3.19)$$

To continue the proof, first recap the translation property or time shifting theorem of Fourier Transform:

Theorem 3.1 *If there exists a Fourier pair: $f(x, y) \leftrightarrow F(u, v)$, for any $x_0 \in \mathbf{C}$, $y_0 \in \mathbf{C}$, we have another Fourier pair: $f(x - x_0, y - y_0) \leftrightarrow F(u, v) \exp^{-2\pi i(\frac{ux_0}{N} + \frac{vy_0}{M})}$*

Apply this theorem to Eqs. (3.18) and (3.19), we can get the following two equalities:

$$X_{i2}(u, v) = X_{i1}(u, v) \exp^{-2\pi i(\frac{u\Delta r}{N} + \frac{v\Delta c}{M})} \quad (3.20)$$

$$F_{i2}(u, v) = F_{i1}(u, v) \exp^{-2\pi i(\frac{u\Delta r}{N} + \frac{v\Delta c}{M})} \quad (3.21)$$

Figure 3.4: This figure shows the example of an training image and the correlation output being shifted to different peak locations. The image in first column shows the original image, and the middle column shows the preprocessed images with the right eye center shifted to two different locations. The last column shows the ideal correlation outputs corresponding to the images in the middle column (the peaks are located at the same locations with the right eye centers).

In Eq. (3.20), the exponential operator with an imaginary power will not change the magnitude, thus it can be concluded that the spectrum power (energy) stays unchanged for each image even if the target point is shifted to different location. In mathematical form, the following equality holds:

$$X_{i2}^*(u, v) \odot X_{i2}(u, v) = X_{i1}^*(u, v) \odot X_{i1}(u, v) \quad (3.22)$$

Add the LHS and RHS of Eq. (3.22) across all images, and reshape the results to diagonal matrices, then we can see the average correlation energy $\mathbf{D}_x$ keeps

invariant. Simultaneously, if we add the LHS and RHS of Eq. (3.20) across all images, the average images $M_1(u, v)$ and $M_2(u, v)$ satisfy:

$$M_2(u, v) = M_1(u, v) \exp^{-2\pi i(\frac{u\Delta r}{N} + \frac{v\Delta c}{M})} \quad (3.23)$$

Based on Eq. (3.23), we have:

$$M_2^*(u, v) \odot M_2(u, v) = M_1^*(u, v) \odot M_1(u, v) \quad (3.24)$$

Rewrite Eq. (3.24) to diagonal matrix and vector form, then it proves $\mathbf{M}^+\mathbf{M}$ stays unchanged. Similarly, combining Eqs.(3.21) and (3.23), it obtains:

$$M_2(u, v) \odot F_2^*(u, v) = M_1(u, v) \odot F_1^*(u, v) \quad (3.25)$$

Rewrite above equation to matrix and vector form, we can prove $\mathbf{M}\mathbf{f}_p^*$ stays unchanged. In summary, by shifting target points and correlation peaks to different locations, the items $\mathbf{D}_x$, $\mathbf{M}^+\mathbf{M}$, and $\mathbf{M}\mathbf{f}_p^*$ maintain unchanged, so the GUF filter also keeps invariant.

Finally, look back to the filter formula in Eq. (3.15), and I will give the interpretation of GUF filters. Figure 3.5 presents the internal composition of GUF construction. Essentially, the basic module of GUF filter is the average image $\mathbf{M}$ (in diagonal form) computed from a set of samples that have been preprocessed and aligned. Next, the product $\mathbf{M}\mathbf{f}_p^*$ can be interpreted as correlating the average image with an optimal correlation output. Particularly, if this optimal output is the Dirac delta function in spatial domain, that is, only the target point location has non-zero value and all other places are zeros. Specifically, delta function is:

$$f_p = \delta_p(r - r_p, c - c_p) = \begin{cases} 1, & \text{at pixel } (r_p, c_p) \\ 0, & \text{elsewhere} \end{cases} \quad (3.26)$$

Recall that Fourier Transform of the delta function turns to be a flat planes with all ones, i.e., the vectorized frequency representation is: $\boldsymbol{\delta}_p = \mathbf{1}$, where $\mathbf{1}$ is a $d \times 1$

complex vector of ones. In that case, $\mathbf{M}\mathbf{f}_p^*$ turns to be the average image $\mathbf{m}$ (in vector form). The last step, the role of the inverse of matrix can be viewed as frequency filtering or reweighting on previous result: $\mathbf{M}\mathbf{f}_p^*$.

Separately, the three components contained in the matrix inverse $(\alpha\mathbf{D}_x + \beta\mathbf{S}_x + \gamma\mathbf{I})^{-1}$ have different preferences in the frequency selection. First, $\mathbf{D}_x$ generally acts as a high-pass filter because normally the average energy of high frequencies is relatively small and accordingly the reciprocals are big. Second, $\mathbf{S}_x$ emphasizes these frequencies with small variance or divergence across training images, and suppress those frequencies that vary greatly across the entire training set. The last item $\mathbf{I}$, equally treats all frequency components, and it is more like a low-pass filter. Adding this item can avoid zero division during calculating the reciprocals, and also increase the white noise tolerance as described before. At the same time, $\mathbf{I}$ will affect the peak sharpness. After all, the effects of frequency filtering are controlled by the weights (α , β , and γ) of these three performances. Different weights will produce filters emphasizing on different aspects. Next part will present the discussions and proofs to show the inherent relations between GUFs and prior filters.

3.4 Relations to Prior Filters

As described in previous part, the formula of GUF filters contains three parameters. By adjusting their values, GUF filters could turn to be previous unconstrained filters including AVG, UMACE, MACH, UOTSDF, ASEF and MOSSE. In other words, these prior filters can be viewed as special cases of GUF filters when the parameters are set properly as indicated in Table (3.1). Involved parameters include α , β , γ and the optimal output f_p .

Figure 3.5: This figure illustrates how the GUF filter for right eye is constructed. First training images are normalized and shifted to a common peak location. Second, the average image is obtained. Third, correlate the average image with an optimal correlation plane. Finally, the resulting frequencies are reweighted to achieve different filters by setting parameters α , β and γ .

3.4.1 GUF vs AVG

Conventionally, a set of target-in-center training images should be prepared to calculate the AVG filter. Still take right eye detection for example, only a small portion of full face images can be cropped and utilized for average computing. Purposely we can release that limitation for AVG: full face images are first pre-processed to smooth edges, and circularly shift right eyes to image center, then a full-size AVG filter is gained by averaging operator. Such a filter is also reachable using the general GUF formula in Eq. (3.15). Substitute $\alpha = 0$, $\beta = 0$, $\gamma = 1$ and $f_p = \delta_p$ into the formula, where δ_p represents the 2D Dirac delta function expressed in Eq. (3.26). GUF filter becomes $\mathbf{I}^{-1} \mathbf{M} \delta_p = \mathbf{M} \mathbf{1} = \mathbf{m}$, which is the average

Table 3.1: Relations between GUF and other filters

General filter form: $\mathbf{h} = (\alpha \mathbf{D}_x + \beta \mathbf{S}_x + \gamma \mathbf{I})^{-1} \mathbf{M} \mathbf{f}_p^*$					
Filters	Parameter settings				Formula
	α	β	γ	$f_p(r - r_p, c - c_p)$	$\mathbf{h}_{\text{GUF}} = (\alpha \mathbf{D}_x + \beta \mathbf{S}_x + \gamma \mathbf{I})^{-1} \mathbf{M} \mathbf{f}_p^*$
AVG	0	0	1	$\delta_p(r - r_p, c - c_p)$	$\mathbf{h}_{\text{AVG}} = \mathbf{M} \delta_p^* = \mathbf{m}$
UMACE	1	0	0	$\delta_p(r - r_p, c - c_p)$	$\mathbf{h}_{\text{UMACE}} = \mathbf{D}_x^{-1} \mathbf{M} \delta_p^* = \mathbf{D}_x^{-1} \mathbf{m}$
MACH	0	1	0	$\delta_p(r - r_p, c - c_p)$	$\mathbf{h}_{\text{MACH}} = \mathbf{S}_x^{-1} \mathbf{M} \delta_p^* = \mathbf{S}_x^{-1} \mathbf{m}$
UOTSDF	α	$\sqrt{1 - \alpha^2}$	0	$\delta_p(r - r_p, c - c_p)$	$\mathbf{h}_{\text{UOTSDF}} = (\alpha \mathbf{D}_x + \sqrt{1 - \alpha^2} \mathbf{S}_x)^{-1}$ $\mathbf{M} \delta_p^* = (\alpha \mathbf{D}_x + \sqrt{1 - \alpha^2} \mathbf{S}_x)^{-1} \mathbf{m}$
ASEF	1	0	ϵ	$g_p(r - r_p, c - c_p)$	$\mathbf{h}_{\text{ASEF}} = \frac{1}{N} \sum_i \mathbf{h}_{\text{GUF}_i}$ $= \frac{1}{N} \sum_i [(\mathbf{X}_i^+ \mathbf{X}_i + \epsilon \mathbf{I})^{-1} \mathbf{X}_i \mathbf{g}_p^*]$
MOSSE	1	0	ϵ	$g_p(r - r_p, c - c_p)$	$\mathbf{h}_{\text{MOSSE}} = (\mathbf{D}_x + \epsilon \mathbf{I})^{-1} \mathbf{M} \mathbf{g}_p^*$
GUF ₁	1	0	0	$g_p(r - r_p, c - c_p)$	$\mathbf{h}_{\text{GUF}_1} = \mathbf{D}_x^{-1} \mathbf{M} \mathbf{g}_p^*$
GUF ₂	0	1	0	$g_p(r - r_p, c - c_p)$	$\mathbf{h}_{\text{GUF}_2} = \mathbf{S}_x^{-1} \mathbf{M} \mathbf{g}_p^*$
GUF ₃	0	1	ϵ	$g_p(r - r_p, c - c_p)$	$\mathbf{h}_{\text{GUF}_3} = (\mathbf{S}_x + \epsilon \mathbf{I})^{-1} \mathbf{M} \mathbf{g}_p^*$

image. No matter where the common peak location $p(r_p, c_p)$ is set, the resulting AVG filter displays an average face with right eye circularly shifted to the center as shown in Figure 3.6. This is due to shifting invariance property of correlation $\mathbf{M} \delta_p$ as described in Section 3.3. From the figure, it is observed that the right eye in computed mean image (spatial form of $\mathbf{M}$) lies at $p(r_p, c_p)$, while correlating with an optimal function f_p peaked at $p(r_p, c_p)$ shifts the right eye to image center. The smoothed edges in the filter image result from the windowing preprocessing.

Figure 3.6: This figure shows how AVG filter is trained using GUF method.

3.4.2 GUF vs UMACE

It is not difficult to generalize GUF filter to UMACE. Substitute $\alpha = 1$, $\beta = 0$, $\gamma = 0$ and $f_p = \delta_p$ to the GUF formula, it leads to UMACE filter: $\mathbf{h} = \mathbf{D}_x^{-1} \mathbf{M} \delta_p = \mathbf{D}_x^{-1} \mathbf{M} \mathbf{1} = \mathbf{D}_x^{-1} \mathbf{m}$. It is worth mentioning that the UMACE filter computed in this way slightly differs from the traditional UMACE filter. Similar to AVG, conventionally UMACE filter requires target-in-center training images and does not contain edge smoothing and image shifting steps. As a result, the filter size of UMACE filter by GUF method can be as large as full image, while the size by traditional way is equal to the size of cropped regions. The following MACH and UOTSDF are similar to UMACE on these points.

3.4.3 GUF vs MACH, UOTSDF

Theoretically, MACH and UOTSDF are also special cases of GUF filter. For the former one, the parameter settings are: $\alpha = 0$, $\beta = 1$, $\gamma = 0$ and $f_p = \delta_p$, leading to filter: $\mathbf{h} = \mathbf{S}_x^{-1} \mathbf{M} \delta_p = \mathbf{S}_x^{-1} \mathbf{M} \mathbf{1} = \mathbf{S}_x^{-1} \mathbf{m}$. For UOTSDF filter, α and β range from 0 to 1, and $\gamma = 0$ and $f_p = \delta_p$. UOTSDF is the tradeoff of UMACF and MACH.

3.4.4 GUF vs ASEF

Unlike aforesaid filters, the relation between GUF and ASEF is not explicitly exhibited. Nevertheless, ASEF filter inherently belongs to a particular condition of GUF. Considering an individual image in the training set, the target point is located at pixel $t_i(r_i, c_i)$. After normalization and windowing, the i th image is denoted by $\hat{x}_i(r - r_i, c - c_i)$. Accordingly, the optimal correlation output is synthesized as a 2D Gaussian $\hat{g}_i = g(r - r_i, c - c_i)$. After shifting the target point or peak to the common pixel $p(r_p, c_p)$, their counterparts can conventionally denoted by $x_i(r - r_p, c - c_p)$ and $g_p = g(r - r_p, c - c_p)$. Using the time shifting theorem (Theorem 3.1), it is easy to prove that correlation between two planes keeps unchanged if both components have been shifted together with same amount of offsets. To be specific, the following two equalities stand: (i) $\hat{\mathbf{X}}_i \hat{\mathbf{g}}_i^* = \mathbf{X}_i \mathbf{g}_p^*$, and (ii) $\hat{\mathbf{X}}_i^+ \hat{\mathbf{X}}_i = \mathbf{X}_i^+ \mathbf{X}_i$. The proofs can refer to Section 3.3 for details. Based on the i th image, set the parameters as: $\alpha = 1$, $\beta = 0$, $\gamma = \epsilon$, and $f_p = g_p$, then the outcome of GUF filter for this individual image is:

$$\begin{aligned}
 \mathbf{h}_{\text{GUF}_i} &= (\mathbf{X}_i^+ \mathbf{X}_i + \epsilon \mathbf{I})^{-1} \mathbf{X}_i \mathbf{g}_p^* \\
 &= (\hat{\mathbf{X}}_i^+ \hat{\mathbf{X}}_i + \epsilon \mathbf{I})^{-1} \hat{\mathbf{X}}_i \mathbf{g}_i^* \\
 &= \frac{\hat{\mathbf{x}}_i \odot \hat{\mathbf{g}}_i^*}{\hat{\mathbf{x}}_i^* \odot \hat{\mathbf{x}}_i + \epsilon} \\
 &= \mathbf{h}_{\text{ASEF}_i}
 \end{aligned} \tag{3.27}$$

In the third step of Eq. (3.27), multiplication of diagonal matrices $(\mathbf{X}_i, \widehat{\mathbf{X}}_i)$ can be equivalently replaced by element-wise multiplication of relevant vectorized entities $(\mathbf{x}_i, \widehat{\mathbf{x}}_i)$. After all, the derivation process in Eq. (3.27) shows the GUF filter is equal to the exact filter with each individual image. Naturally, averaging the GUF filter for each image $\mathbf{h}_{\text{GUF}_i}$ over the training data results in ASEF filter. Under this condition, computing the ASEF filter through either the original way or GUF way will output exactly the same filter because original ASEF is also trained with full face images.

3.4.5 GUF vs MOSSE

The link between GUF and MOSSE can be revealed directly. In Eq. (3.15), substitute in parameters $\alpha = 1$, $\beta = 0$, $\gamma = \epsilon$, and $f_p = g_p$, and deduce the filter as shown in Eq. (3.28).

$$\begin{aligned}
 \mathbf{h} &= (\mathbf{D}_x + \epsilon \mathbf{I})^{-1} \mathbf{M} \mathbf{g}_p^* \\
 &= \left(\frac{1}{N} \sum_i \mathbf{X}_i^+ \mathbf{X}_i + \epsilon \mathbf{I} \right)^{-1} \left(\frac{1}{N} \sum_i \mathbf{X}_i \right) \mathbf{g}_p^* \\
 &= \left[\frac{1}{N} \left(\sum_i \mathbf{X}_i^+ \mathbf{X}_i + N \epsilon \mathbf{I} \right) \right]^{-1} \left(\frac{1}{N} \sum_i \mathbf{X}_i \mathbf{g}_p^* \right) \\
 &= \left(\sum_i \widehat{\mathbf{X}}_i^+ \widehat{\mathbf{X}}_i + N \epsilon \mathbf{I} \right)^{-1} \left(\sum_i \widehat{\mathbf{X}}_i \widehat{\mathbf{g}}_i^* \right) \\
 &= \frac{\sum_{i=1}^N \widehat{\mathbf{x}}_i \odot \widehat{\mathbf{g}}_i^*}{\sum_{i=1}^N \widehat{\mathbf{x}}_i^* \odot \widehat{\mathbf{x}}_i + N \epsilon} \\
 &= \mathbf{h}_{\text{MOSSE}}
 \end{aligned} \tag{3.28}$$

Obviously, the MOSSE filter turns to be a special case of GUF if ϵ is set properly. But using GUF form to compute MOSSE (or ASEF) only requires about half computational time, because it avoids computing the correlations $\widehat{\mathbf{x}}_i \odot \widehat{\mathbf{g}}_i^*$ for every image as shown in Eq. (3.28) (or Eq. (3.27)). Meanwhile, it no longer requires to separately produce an optimal output for every image depending on the target

location, and only a common correlation output g_p should be predefined. This property is helpful to analyze the influence of different optimal output because it is no need to recompute the remaining part $(\alpha\mathbf{D}_x + \beta\mathbf{S}_x + \gamma\mathbf{I})^{-1}\mathbf{M}$ when f_p is changed.

3.5 Summary

Basically, the objective of GUF is to minimize the weighted sum of three performance measures: mean squared error to optimal output, variance of correlation output, and filter energy. As these measures all belong to statistical and general metrics over entire training data, the GUF filter can overcome the problem of over-fitting training images encountered by constrained correlation filters. Taking advantage of the shifting invariance of correlation, GUF is capable of carrying out training on full-size non-centered images. This can greatly improve the flexibility with respect to training images for many traditional filters. In addition, a new perspective has been presented to decompose GUF filter to three parts: *average image + correlation with optimal output + frequency filtering*. This chapter has also shown that many state-of-the-art filters can be expressed as a special form of the GUF filter with specific parameter settings.

Chapter 4

Facial Image Analysis via Correlation Filters

This chapter looks at two applications of Generalized Unconstrained Filters (GUF). The first one is the facial landmark localization, in which the filters for facial points are offline trained with a large number of face images. The second application is the combined face detection and tracking, where face filter is online trained using the initial frame of a video sequence and incrementally updated with newly coming frames.

4.1 Facial Landmark Localization

4.1.1 Introduction

In general, facial landmark localization (FLL) techniques find the accurate positions of facial feature points in frontal or near frontal face images. Typical facial features include eye centers, nose, mouth corners, etc. FLL is a fundamental and invaluable work in face-related research areas such as face recognition, facial expression recognition and face pose estimation, etc [1].

There are a number of works to detect facial landmarks. Early works tend to find facial landmarks like eye centers and corners, mouth corners based on

geometrical knowledge and their particular distinctiveness. Commonly a search region is established within the face for each landmark. For example, horizontal and vertical pupil boundaries are extracted to detect eye centers by applying *projection functions* such as Integral Projection Function and Variance Projection Functions [72, 73]. Such approaches have fast speed, but usually they are sensitive to handle face images with large variations due to changes in lighting, rotation and background. Meanwhile, methods of this type rely on the distinct characteristics of particular points, so their generality capability is limited. Model-dependent approaches have attracted much research attention with the successful research of Active Shape Model (ASM) [74] and Active Appearance Model (AAM) [75]. The face shape is modeled by a set of points, controlled by the shape model, and these algorithms aim to match the model to a new image. To detect the position for each landmark point, the original methodology suggests using Mahalanobis distance to look in the new image around each point.

Another way of point detection is to embrace the fruits of classification techniques including Support Vector Machines [19, 20], Bayesian classifier [76], Neural Networks [77], Decision Trees [78, 79], and Boosting [79, 80]. Generally, detectors are trained using local images around the facial landmarks, and then a sliding window is applied to an input image to perform classification at different places and scales. For speed consideration, the window size is normally kept as small as possible.

In recent years, most real-time systems adopt the Viola Jones Cascade detector [7, 79] for object detection. This is one of the milestones in computer vision. The Cascade detector is capable of running fast detection by using Haar based features with integral image, and cascaded decision trees.

Correlation filtering is not a newly emerging technique, but it has not obtained sufficient attention in facial landmark detection. For a long time, the impression

of correlation filters appears in the form of template matching, which has very limited detection ability. Mostly, correlation filters have been proposed in optical pattern recognition or synthetic aperture radar detection. For face biometrics, simple matched spatial filters have been applied to eye finding before [81]. In [6], Bolme et al. proposed Average of Synthetic Exact Filter (ASEF) for eye localization, and conducted experimental comparison with other leading filters including Synthetic Discriminant Functions (SDF), Minimum Average Correlation Energy (MACE), Optimal Tradeoff Filters (OTF), and Unconstrained Minimum Average Correlation Energy (UMACE) filters. In Bolme’s doctoral dissertation [25], an extension of ASEF, Minimum Output Sum of Squared Error (MOSSE) filters was also evaluated with facial feature localization. This work also gets inspiration and benefits from these pioneering works. This part evaluates the GUF filter’s ability to detect facial points, and demonstrates its improvements over prior filters.

4.1.2 Error Compensation and Filter Updating

To further improve the performance, an error compensation phase was introduced to reduce inaccurate detections caused by the GUF filter. In general, unconstrained filters capture the global or averaged characteristics of training images. According to discussions in Section 3.3, the inherent composition of GUF filter originates from the average image, and follows correlating with an optimal output as well as frequency filtering. As a result, every image contributes equally to the GUF filter. However, from an overall viewpoint, the contributions of those images with large variations are weakened because normally their occurrence probabilities in the training data are pretty small. On the other hand, naturally such images account for more inaccurate detections. Therefore, the basic principle of updating GUF filter is to increase the weights of these inaccurate detected images in constructing

the filter. Similar handling methods can be inspired from many machine learning methods. For example, weighted least square regression includes an additional weight, that determines how much each observation in the dataset influences the final parameter, on top of the original least square model. In boosting algorithm [80], larger weights should also be placed on these training samples that are not well classified. In addition, SVM classifier imposes different penalties to the samples according to their distances to the decision boundary. Therefore, it is rational to update the filter with varying sample weights.

Initially, the GUF filter is constructed from N images by Eq. (3.15). For description convenience, $\mathbf{h}_0$ denotes the initial filter, and $\mathbf{D}_{x,0}$ and $\mathbf{M}_0$ respectively represents the initial average correlation energy and the mean image. Subsequently, these images whose facial points failed to be accurately located by previous filter are reused to augment the filter. Hence, the filter updating is an iterative process. In iteration $k - 1$, the filter h_{k-1} is learnt. Using this filter, suppose there are N_k out of N training images that are not successfully located. To increase their contributions in the filter, these N_k images are utilized again to update the filter in the k th iteration. To be specific, the newly added average energy plane $\Delta\mathbf{D}_{x,k-1}$, and average image $\Delta\mathbf{M}_k$ are calculated based on these N_k inaccurately detected images. Suppose the indexes of these images are gathered in set $\mathcal{S}$. Mathematically, these two entities are expressed by Eqs. (4.1) and (4.2).

$$\Delta\mathbf{D}_{x,k} = \frac{1}{N_k} \sum_{i \in \mathcal{S}} \mathbf{X}_i^\dagger \mathbf{X}_i \quad (4.1)$$

$$\Delta\mathbf{M}_k = \frac{1}{N_k} \sum_{i \in \mathcal{S}} \mathbf{X}_i \quad (4.2)$$

To sum up, the augmented average energy plane $\mathbf{D}_{x,k}$ and average image $\mathbf{M}_k$ for iteration k are computed as:

$$\mathbf{D}_{x,k} = \eta \mathbf{D}_{x,k-1} + (1 - \eta) \Delta\mathbf{D}_{x,k} \quad (4.3)$$

and

$$\mathbf{M}_k = \eta \mathbf{M}_{k-1} + (1 - \eta) \Delta \mathbf{M}_k \quad (4.4)$$

where η indicates the ratio of image numbers (inclusive of repetitive use) used to train filters $\mathbf{h}_{k-1}$ and $\mathbf{h}_k$, that is,

$$\eta = \frac{N + \sum_{i=1}^{k-1} N_i}{N + \sum_{i=1}^k N_i} \quad (4.5)$$

Using the formula in Eq. (3.17), the filter $\mathbf{h}_k$ is formally expressed as:

$$\mathbf{h}_k = [(\alpha + \beta) \mathbf{D}_{x,k} - \beta \mathbf{M}_k^+ \mathbf{M}_k + \gamma \mathbf{I}]^{-1} \mathbf{M}_k \mathbf{f}_p^* \quad (4.6)$$

Gradually, it puts more weights on those images that are difficult to correctly locate the landmarks. In addition, another significance is that it is possible to apply GUF filters for online object tracking.

4.1.3 Experiments

4.1.3.1 Experimental Setup and Evaluation Methodology

Experimental evaluation for facial landmark localization was conducted with the Facial Recognition Technology (FERET) database [22]. The used data set contains 3368 images of 1204 persons. The ground truth positions of facial landmarks including eye centers, nose and mouse were manually marked and publicly available at <http://www.cs.colostate.edu/evalfacerec/data/normalization.html> (including the name list of images). This data was first randomly halved into two groups with 1684 images of 602 persons each. One group was further divided into a training set of 1024 images and a validation set of 660 images, while the other group was purely used as the testing set. In the experiments, filters were constructed using the training set, and relevant parameters were tuned with the validation set if necessary, then final detection results were reported with the testing set.

To make it a person-independent issue, all images of the same person were distributed to the same set. Generally, face images are often detected using Haar based Cascade detector implemented in OpenCV library. Following the experimental protocols described in [6], face detection was simulated by first generating 128×128 images with eye coordinates at $(40, 50)$ and $(88, 50)$, and then applying random geometric transform up to $\pm\pi/12$ rotations, 1 ± 0.15 scales and $(\pm 5, \pm 5)$ pixels translations. All the three sets were enlarged 8 times by generating 8 perturbations from each image. It should be clarified that the major concern here is on how to better detect the facial landmarks over previous algorithm assuming the faces have been detected and roughly normalized by preceding face detection and normalization steps. After all, only depending on a linear filter can not well handle too big variations in rotations and scaling.

Preprocessing algorithm must be implemented to all images in spatial domain beforehand. The preprocessing consists of many steps as shown in Figure 4.1. First, each image was normalized by applying $\log(\log(I + 1))$ operator to every pixel; Second, the resulting image was further normalized to have zero mean and unit squared sum in order to reduce lighting effects. Third, a Cosine window was applied to reduce edge distortions during Fourier transform. Simultaneously, windowing operation also helps to avoid abrupt intensity changes after image shifting in next step. Lastly, circularly shift the image to place the target point to a common reference point. This is to assume the image is periodically repeated in spatial domain, that is, the image top is connected to the bottom, and the left is connected to the right. At the same time, the periodic repetition is also a default processing in Fourier transform.

As described in Section 3.4, GUF filters are identical to many previous filters including AVG, UMACE, MACH and UOTSDF in principle. The difference is that

Figure 4.1: Illustration of image preprocessing procedure. The face image is first cropped out of the background. Each face will generate 8 perturbations to enlarge the dataset by applying geometric transforms. Subsequently, intensity at each pixel is normalized by applying a logarithm operation ($\log(I)$) to reduce lighting effects. Then all pixels in the image are further normalized to have a sum of one. A Cosine windowing operation is followed to reduce the edge distortions during DFT. Lastly, the keypoints in all training images should be shifted to a common reference location.

previously these filters could only use target-in-center images, while GUF based method enables the use of full-sized images. In the experiments, AVG, UMACE, MACH and UOTSDF filters were trained and evaluated under two scenarios: first one was to use traditional way in which filter design utilized 64×64 images centered on the specified landmarks; For the other, these filters were trained using full 128×128 images by means of GUF general formula. Particularly, ASEF and MOSSE also could be trained using original way and GUF way, but both ways lead to exactly the same filter because originally these two filters are trained with full images.

As a study case, experimental evaluation was conducted to detect two facial landmarks including right eye center and nose. To determine the position of a

landmark, each input image is fed into the preprocessing phase and transformed to frequency domain, then correlating with a filter is implemented via element-wise multiplication, resulting in the output plane from whose spatial counterpart the landmark is found at the place with highest value. The landmarks are searched across whole face images. To evaluate the detection accuracy, normalized error (distance) denoted by N_e is adopted as the measuring criterion. As introduced in [82], it is defined as the ratio of absolute localization error to the interocular distance:

$$N_e = \frac{\| \tilde{P} - P \|}{\| M_l - M_r \|} \quad (4.7)$$

where $\tilde{P}$ is the predicted position of facial point, and P , M_l and M_r respectively represent the ground truth positions of the landmark, left and right eyes. A successful localization is counted if the normalized error is below a specified threshold. For example, $N_e < 0.10$ roughly corresponds to the absolute detection error within about the size of iris. In the experiments, performance evaluation was performed under various criteria with different thresholds: 0.02, 0.04, 0.06, 0.08, 0.10. Meanwhile, algorithms were tuned with the criterion of $N_e < 0.10$.

4.1.3.2 Experimental Results and Discussions

In this section, the landmark detection ability of the class of GUF filters is compared, other competing correlation filters and commonly adopted Viola Jones Cascade detector. The experiments utilized the implementation of OpenCV library with already trained Cascade classifiers for right eye (classifier file *haarcascade_right_eye_2splits.xml*) and nose (classifier file *haarcascade_mcs_nose.xml* [83]).

As reported in [25], one shortcoming of SDF, MACE and OTF is that they easily over-fit the training data. That is to say, the more images the filters are trained on, the worse they perform on the validation or testing data. Unconstrained filters

such as UMACe, ASEF, etc. have no such problem. The proposed GUF filters are also free of this issue. To better understand it, Figure 4.2 shows how different nose filters change with increasing number of training images. SDF and MACE gradually run into a mess, while other constrained filters are trending to show up the appearance of nose. In addition, Figure 4.3 reports the correct detection rates of many nose filters with different number of training images. Obviously, the successful rates of SDF and MACE will decline in the end with increasing number of images, while unconstrained filters normally exhibit better results with more training samples.

The resultant filters for right eye and nose are presented in Figures 4.4 and 4.5. Again it is observed the central part of the filters shows the appearance of right eye or nose. Particularly, the AVG filter of size 64×64 , obtained by averaging 8192 target-in-center subimages, appears as a blurred and average image across the whole image, while in the 128×128 sized AVG filter, we can see the clues of smoothed face edges due to image preprocessing. Interestingly, other 128×128 filter images could be generated from this AVG (128) image by post-processing procedure: adding either a correlation phase with optimal output or frequency filtering or both. For example, UMACe (128) is resulted from AVG (128) image followed by dividing with average frequency energy ($\mathbf{D}_x$). Besides, MOSSE (128) could be viewed as AVG (128) followed by correlating with a Gaussian and dividing by frequency energy.

Tables 4.1 and 4.2 separately record the localization results on testing set for right eye and nose. In the tables, many methods such as AVG, UMACe, MACH and UOTSDF include two rows: one corresponds to the testing results using conventional way with filter size of 64×64 , and the other corresponds to the results using GUF based approach, with filter size of 128×128 . As mentioned before,

Figure 4.2: The nose filters with different training sizes.

computing ASEF or MOSSE by two means turn to the same filter (128×128), so their localization results are presented with a single row each. Specially, the

Figure 4.3: This figure shows the correct localization rate of GUF using different size of training data.

MOSSE filter sized 64×64 was deliberately examined. GUF_1 , GUF_2 and GUF_3 are three special cases of GUF filters with parameters given in Table 3.1.

Through these two tables, one obvious observation is that the performance of traditional filters including AVG, UMACE, MACH and UOTSDF could be improved after relaxing the limitation of filter size due to requirement of target-in-center images. In other words, these 128×128 sized filters constructed by GUF formula are superior to corresponding 64×64 sized ones constructed in conventional way. This makes some sense: filter design using full images performs better than that using partial images in finding facial points over the whole face. Moreover, I also examined the MOSSE filter trained using 64×64 target-in-center images, and the outcomes in Tables 4.1 and 4.2 show MOSSE filter with size of 64×64 no longer exhibits persuasively higher accuracy than conventional MACH. Conclusively, GUF based method could improve the performance of aforementioned traditional filters. Furthermore, advanced filters perform better than commonly used Cascade detector.

Figure 4.4: GUF filters for detecting right eye center.

In addition, the influence of synthetic output f_p could be observed by comparing UMACE with GUF₁, and MACH with GUF₂ at a size of 128×128 . As indicated in Table 3.1, the only difference is that UMACE and MACH use delta function δ_p while GUF₁ and GUF₂ use Gaussian function g_p as the optimal output. In principle, δ_p is more ideal than g_p because the former has perfect peak sharpness. But the reality is that GUF₁ and GUF₂ have exhibited better localization ability.

AVG (64)	UMACE (64)	MACH (64)	UOTSDF (64)

AVG (128)	UMACE (128)	MACH (128)	UOTSDF (128)

MOSSE (64)	ASEF (128)	GUF ₁ (128)	GUF ₂ (128)

MOSSE (128)	GUF ₃ (128)		

Figure 4.5: GUF filters for detecting nose.

One rational explanation is that it is difficult to place the peak of δ_p to a clear landmark since the ground truth positions are manually picked, while g_p allows bigger tolerance to picking error.

Eventually, the validity of adding error compensation phase is shown in the last two rows of Tables 4.1 and 4.2. As an example, MOSSE' and GUF_3' represent the filters by applying filter updating phase to original GUF filters with parameter

Table 4.1: Detection results of right eye center(%)

Methods	Filter size	Criterion				
		0.02	0.04	0.06	0.08	0.10
Cascade	-	19.7	60.7	84.0	90.7	92.2
AVG	64×64	4.0	13.0	23.8	30.27	35.1
	128×128	10.8	37.3	62.1	72.5	77.6
UMACE	64×64	19.1	58.0	74.3	77.8	78.6
	128×128	20.9	62.4	81.0	85.4	86.3
MACH	64×64	20.6	62.9	80.5	84.3	85.2
	128×128	21.7	65.1	84.6	89.2	90.1
UOTSDF	64×64	19.5	59.5	76.4	80.0	80.8
	128×128	21.3	63.6	82.5	87.0	87.9
ASEF	128×128	24.5	70.5	90.4	94.6	95.5
MOSSE	64×64	22.2	63.5	80.1	83.8	84.5
	128×128	25.3	71.0	90.3	94.5	95.2
GUF ₁	128×128	24.9	70.9	90.0	94.3	94.9
GUF ₂	128×128	25.4	72.0	91.4	95.6	96.3
GUF ₃	128×128	25.4	71.8	91.6	95.7	96.5
MOSSE'	128×128	25.3	73.6	92.3	96.2	97.0
GUF' ₃	128×128	26.6	74.0	92.6	96.1	97.1

settings corresponding to MOSSE and GUF₃. Based on the validation result, 29 and 10 iterations were used to update the filters for detecting right eye center and nose. The testing results in the tables indicate the error compensation process is beneficial to reduce localization error. The improvement is slight but noteworthy for old and important correlating filtering technique.

4.1.3.3 Runtime Performance

Runtime efficiency is one outstanding advantage of correlation filters. It could be considered from two aspects: training time and testing time. For comparison, training a Viola Jones Cascade classifier empirically requires many hours or even days. As indicated in Eq. (3.17), the training time of GUF filters are mainly

Table 4.2: Detection results of nose (%)

Methods	Filter size	Criterion				
		0.02	0.04	0.06	0.08	0.10
Cascade	-	5.1	24.6	51.8	70.0	78.1
AVG	64×64	1.9	6.7	13.6	19.0	22.0
	128×128	5.7	22.3	43.7	59.2	69.5
UMACE	64×64	10.0	39.6	62.8	73.5	77.7
	128×128	11.5	45.2	73.1	87.2	92.8
MACH	64×64	11.8	44.9	69.6	81.0	85.2
	128×128	12.0	45.9	74.0	87.7	93.2
UOTSDF	64×64	10.6	42.4	66.4	77.3	81.5
	128×128	11.8	45.5	73.6	87.6	93.1
ASEF	128×128	14.2	49.3	75.6	88.4	93.3
MOSSE	64×64	11.7	45.4	69.9	80.7	84.7
	128×128	14.4	50.3	77.3	89.4	93.9
GUF ₁	128×128	14.3	50.2	77.1	89.2	93.7
GUF ₂	128×128	14.8	50.5	77.2	89.7	94.2
GUF ₃	128×128	15.2	51.2	77.7	90.3	94.7
MOSSE'	128×128	15.8	51.8	78.7	91.0	95.3
GUF' ₃	128×128	15.8	52.0	78.6	91.2	95.6

consumed in computing the average spectrum energy $\mathbf{D}_x$ and average image $\mathbf{M}$. Though the formula, UMACE and MACH related filters cost approximately the same amount of training time to GUFs. Table 4.3 records the training time of different filters. Notably, GUF needs only 8.23 seconds to construct the filter from 8192 images.

Table 4.3: Filter construction time with a size of 128×128 using 8192 training images

Filters	ASEF	MOSSE	GUFs/UMACE/MACH	AVG
Training time	19.67s	16.55s	8.23s	7.14s

On the other hand, filters of the same size theoretically require the same amount

of testing time. In the experiments, it took only 20.45 seconds to complete the detection of entire testing set with $1648 \times 8 = 13,472$ images (128×128) using FFTW, Eigen and OpenCV libraries on an i5 3.2GHz CPU. In other words, the testing speed of correlation filters is around 658 frames per second. Comparatively, Viola Jones Cascade method consumed 485.02 seconds or a speed of about 27.8 frames per second. Overall, GUF is really fast in training and testing.

4.2 Face Detection and Tracking

4.2.1 Introduction

Since early 1970s, face detection and tracking has attracted extensive research interests. Robust face detection is quite an essential matter in many applications such as human-computer interfaces, robotics, multimedia systems, etc. Roughly, face detection methods could be classified into two categories: feature-based method and appearance-based method. Table 4.4 gives a summary and more details will be presented below.

Feature-based type often detects facial features including eyebrows, eyes, nose and mouse, by means of color, edge or texture information to infer the presence of face. There are many representative works on feature-based face detection. In [84], many edge feature based techniques were presented for facial feature extraction. As the foremost step, the second derivative Gaussian filter has been applied to derive edge map. Alternative edge extraction techniques include Sobel operator and Canny operator. Skin color is another effective feature in face detection as the color of face is distinct from background. The popular color spaces comprise RGB, CIE, HSV and YCrCb. After the color space is chosen, the skin pixels and non-skin pixels are classified by using clustering or other classifiers. For example, HSV color and Gaussian mixture model were adopted in [85]. Gabor feature, which

Table 4.4: Categorization of frontal face detection methods

Category	Methods	
Feature-Based	Edge [84], Color [85], Gabor Feature [86]	
	Multiple Features [87]	
Appearance-Based	Template-Based	Snake [88]
		Deformable Template [89]
		Active Shape Model [74]
	Manifold Embedding	Eigenface (PCA) [54]
		Fisherface (LDA) [55]
		ICA [90]
	Neural Network [91]	
	Support Vector Machines [92]	
	Hidden Markov Model [93]	
	Naive Bayes Classifier [94]	
	Viola Jones Cascade detector [7, 79]	

is spatially localized and selective to orientation and scale, has also been utilized in face detection [86]. Naturally, it is believed that algorithms based on multiple features perform better than the ones using single feature when encountering variations in illumination, background, or pose. The majority of them utilize global features such as skin color, size, or shape to find face candidates, and then verify them using local features (eyebrows, nose, eyes, mouth, etc.). For example, Chiang et al. [87] first performed color segmentation in normalized RGB space to locate skin-like regions, lip, and eyes. Afterwards, the false candidates were removed by rules derived from the spatial and geometrical relationships of facial components. Accordingly, the precise face regions were determined.

Appearance-based methods usually use the whole face as the input, rather than some distinct local features like eyebrows, eyes, nose and mouth. Statistical analysis or machine learning techniques are applied to find the internal characteristics of face and non-face images. Generally, the learnt characteristics are in the form of

distribution model or classification function, which is used to detect face.

Early works in this category mainly belong to *template-based methods*. These methods use a standard face pattern to compute the correlation value with the input image, and determine the existence of face. One representative is the active contour, or called snake [88], which is commonly used to locate a head boundary. Active contour initializes at the proximity around the head boundary, then gradually locks onto nearby edge using an energy function, and finally shapes the head. The function consists of three parts describing the internal energy, the image forces, and the external energy respectively. In [89], deformable template was used to detect face. It works under similar principle like “snake”. The deformable energy function involves the steepest gradient descent minimization of a combination of external energy due to the valley, edge, peak, and image brightness. A third representative is the active shape model (ASM) proposed by Cootes et al [74]. ASM employs shape and intensity parameters together for classification. In general, template based face detection is very simple to implement, but it is inadequate to deal with variations in scale, pose and shape.

Another sub-category is based on subspace methods, which reinforce computation efficiency and detection efficacy by dimensionality reduction. Among this method, *eigenfaces* [54] and *fisherfaces* [95] are absolutely the most popular ones. Both of them can linearly encode a new face using a number of basis images. Eigenfaces and fisherfaces correspond to famous PCA (*Principal Component Analysis*) and LDA (*Linear Discriminant Analysis*) respectively. It worth mentioning that Belhumeur et al. [95] adopted PCA as a preliminary step for dimension reduction of the input image, and then used LDA to conduct the face or non-face classification. Another manifold embedding method is *Independent Component Analysis (ICA)* [90], which is considered as a generalization of PCA.

Numerous learning methods have been proposed to learn the difference between face and non-face patterns. Table 4.4 only gives several typical ones. In [91], *Neural Network* was utilized to solve face/non-face classification. Their algorithm comprises two components: distribution-based models for face/non-face patterns, and a multilayer perception (MLP) classifier. Osuna et al. [92] first applied *Support Vector Machines* (SVM) in face detection. The scheme is similar to that of [91], and the difference is that SVM is used to classify face/non-face instead of Neural Network classifier. Moreover, *Hidden Markov model*(HMM) was applied to extract facial feature and recognize face in [93]. And in [94], a classifier based on *Naive Bayes* was proposed to estimate the joint probability of local appearance and position of face patterns at multiple resolutions. A face is believed to be existing if the likelihood ratio is larger than the ratio of prior probability.

4.2.2 Detection and Online Tracking

So far, the Viola Jones Cascade method is the most frequently used real-time face detector. It has demonstrated great capability to detect frontal faces. Nevertheless, such a method still has limitations and could be improved. The biggest issue is that the detector performs badly when the face images encounter in-plane rotation, partial occlusion, and poor lighting conditions. In addition, face images captured in profile are also hard to detect with the trained frontal face detector. Online incremental tracking is competent for the aforementioned scenarios, but it also needs to handle drift issue, i.e., gradual adaptation of tracker to background. Combining the Cascade detector and online tracking should be able to complement each individual technique in detecting faces from video sequences. The follows will show the promising properties of GUF filters as being a tracker, and strategy to perfectly fuse with Cascade detector for the purpose of face detection and tracking.

4.2.2.1 Tracking with GUF Filters

The Cascade face detector encounters difficulties when dealing with changing rotation, pose, or moving through harsh lighting conditions. Face tracking could quickly get adapted to the foreseeing conditions by updating previous detection model. Correlation filtering is a promising technique for online tracking. Historically, simple template matching has widely been used to find targets in consecutive frames. In addition, the MOSSE filter has demonstrated good performance in visual object tracking [18]. Here I will evaluate the possibility of extending GUF filters to visual tracking. As this thesis focuses on face analysis, the scope of visual tracking is intentionally narrowed to face tracking. However, the technique could also be extended to general object tracking.

During tracker initialization, filters are constructed using Eq. (3.17). The training set consists of nine images: the initial tracking window in the first frame and its eight perturbations by applying affine transforms. Specifically, these nine images are generated from affine transforms with three scales: $\{0.95, 1, 1.05\}$ and three rotations: $\{-0.1, 0, 0.1\}$. As described in facial landmark localization, the same preprocessing algorithms are utilized to these images, and an optimal output f_p is also generated with peak corresponding to the face center. During tracking, the appearance of face would change continuously by changing the rotations (yaw, roll and pitch), scaling, moving across different lighting conditions, or being partially occluded. Thus the filter must be able to quickly adapt to the new appearance.

Given a new frame, the first step is to correlate with subimage in the tracking window with existing GUF filter, and then the new location of face center is found at the peak of correlation output. The new face image is then utilized to update the filter. The architecture of GUF tracking is illustrated in block diagram as in Figure 4.6.

Figure 4.6: Architecture of GUF tracking algorithm

In theory, the filter $\mathbf{h}(i)$ learnt from i th frame could be updated using two ways. In one way, the GUF filter can be computed as:

$$\mathbf{h}(i) = (1 - \eta)\mathbf{h}(i - 1) + \eta\mathbf{h}_i \quad (4.8)$$

where $\mathbf{h}(i - 1)$ is the GUF filter learnt before (including) frame $i - 1$, and $\mathbf{h}_i$ is the filter learnt only using frame i , or the exact filter. η is updating rate that controls the adapting speed to frame i . Following Eq. 3.14, $\mathbf{h}_i$ is computed as:

$$\begin{aligned} \mathbf{h}_i &= (\alpha\mathbf{D}_i + \beta\mathbf{S}_i + \gamma\mathbf{I})^{-1}\mathbf{M}_i\mathbf{f}_p^* \\ &= (\alpha\mathbf{X}_i^+\mathbf{X}_i + \gamma\mathbf{I})^{-1}\mathbf{X}_i\mathbf{f}_p^* \end{aligned} \quad (4.9)$$

where $\mathbf{D}_i$, $\mathbf{S}_i$ and $\mathbf{M}_i$ correspond to the spectrum power, variance and average image for the i th image. Obviously, using a single image, $\mathbf{S}_i$ is equal to zero and $\mathbf{M}_i$ is the i th image. By this updating way, one drawback is that the role of $\mathbf{S}_x$ has

no effect. An alternative handling method is to generate nine perturbations from frame i as in the initialization, but this will add extra computational load even it is not a big problem for GUF filters.

Another better updating way is to separately update filter components $\mathbf{D}_x$ and $\mathbf{M}_i$ in Eq. (3.17). For frame i , the constructed GUF filter is computed as:

$$\mathbf{h}(i) = [(\alpha + \beta)\mathbf{D}_x(i) - \beta\mathbf{M}^+(i)\mathbf{M}(i) + \gamma\mathbf{I}]^{-1}\mathbf{M}(i)\mathbf{f}_p^* \quad (4.10)$$

where $\mathbf{D}_x(i)$ represents the adaptive average spectrum power, and $\mathbf{M}(i)$ represents the adaptive mean image from frame i . They are computed as:

$$\begin{aligned} \mathbf{D}_x(i) &= (1 - \eta)\mathbf{D}_x(i - 1) + \eta\mathbf{D}_i \\ &= (1 - \eta)\mathbf{D}_x(i - 1) + \eta\mathbf{X}_i^+\mathbf{X}_i \end{aligned} \quad (4.11)$$

and

$$\mathbf{M}(i) = (1 - \eta)\mathbf{M}(i - 1) + \eta\mathbf{X}_i \quad (4.12)$$

The tracker could be quickly computed to adapt the new appearance using above three equations. Again, I would like to mention the updating rate η which is the weight for new image. Large value enables the tracker to quickly get adapted to appearance changes, and simultaneously increases the risk of adaptation to background. In contrast, too small value may not lead to a robust filter as it fails to follow the appearance changes. In the experiments, η is set to $\frac{1}{16}$ without fine tuning.

4.2.2.2 Combination of Cascade Detector and GUF Tracker

In practice, face tracker by purely using the filter may suffer from drifting to background sometimes especially under such conditions as moving into or moving out of partial occlusion, or switching between frontal and profile face. Once the filter gets adapted to the background, the tracker will stick to that wrong target for the rest

sequence. Besides, scaling information is lacking in most tracking algorithms, and so is GUF based tracker. The size of tracking window will keep unchanged all over the video sequence. This often leads to the situation that the tracked rectangle does not fit the face well.

Fusion of Cascade detector and GUF tracker could bring great benefits to detect faces. Table 4.5 summarizes the strengths and weaknesses of using individual method. Clearly, these two methods could nicely complement each other. Figure 4.7 shows the overall framework to fuse these two techniques for face detection. In the initial frame, GUF filter is constructed using the subimage from initial rectangle or the face detected by the Cascade detector. The face is tracked using GUF filter as described above. To prevent drift, the Cascade detector is activated every ten frames to rectify the tracking. If no face is detected, the GUF tracking continues. Otherwise suppose the face is detected at rectangle S and the tracking rectangle is denoted by T in a frame. Three criteria could be used to measure the similarity score of these two rectangles. One is the relative distance (RD) which is defined as the ratio of the Euclidean distance between their centers to half the sum of width and height of T . This item indicates the offset between results of GUF tracker and Cascade detector. The second criterion is the overlapping ratio (OR) of T and S that is defined as:

$$\text{OR} = \frac{S \cap T}{S \cup T} \quad (4.13)$$

This criterion implicitly reflects both the offset of centers and scale changes of S and T . The last item is purely the scaling which is defined as the ratio between area of S to area of T . The goal is to find a confident face by Cascade detector to reinitialize the GUF filter so as to stop error propagation or drift during tracking. Empirically, corresponding three requirements are used to confirm a confident face detection: (i) $\text{RD} \in [0.05, 0.90]$; (ii) $\text{OR} \in [0.05, 0.90]$; and (iii) $\text{scaling} \in [0.5,$

1.5]. If [(i) || (ii)] && (iii) is a true statement, then rectangle S is considered as a confident detection and used to reinitialize the GUF filter, otherwise this detected face is ignored and tracking continues. Basically, item (iii) can filter out these false detections or inaccurate detections by Cascade detector because naturally the face size will not change beyond that range in continuous frames. In addition, either item (i) or (ii) helps to guarantee the detected face is reasonably around the tracked position, which removes detections from background. Generally, it is unnecessary to reconstruct the GUF tracker too frequently, so the lower bound of RD and upper bound of OR are set to avoid filter reinitialization if S and T differ slightly which happens at most cases. In fact, it takes less than 10 milliseconds to reinitialize the GUF filter by using nine images perturbed from the detected face subimage.

Table 4.5: Comparison of Cascade detector and GUF tracker for face detection

Cascade Detector	Positive	1. Detect frontal faces well for most natural conditions. 2. The detected window fits the face well. 3. Frame-independent, incorrect detections have no influence to following frames.
	Negative	1. Speed is marginally sufficient for real time requirement (10-30 fps). 2. Perform poorly when the face is partially off the image. 3. Perform poorly under harsh lighting conditions. 4. Can not detect face with large rotations (yaw, pitch, roll).
GUF Tracker	Positive	1. Detect faces well for smoothly changing sequences. 2. Detection speed is very fast (over several hundreds fps). 3. Can detect faces with large rotations. 4. Can detect faces with occlusions. 5. Can detect faces under extreme lighting conditions.
	Negative	1. Frame-dependent, detection error will propagate to subsequent frames. 2. Easy drift to background, no self-recovery. 3. Size of tracking window is fixed, and it cannot reflect the face size.

Figure 4.7: Architecture of Cascade detector + GUF tracking algorithm

The combined Cascade detector and GUF filters would help to make up their individual weaknesses. For example, Cascade detector performs badly with large rotations, partial occlusions, or harsh lighting conditions that occasionally occur in natural videos, GUF tracker is competent to these scenarios. Simultaneously, the detection speed could be increased to over 80 frames per seconds. On the other hand, Cascade detector provides extra scale information to GUF filters and the tracking window better fits the face. Moreover, inaccurate tracking will be quickly recovered after reinitialization.

4.2.3 Experiments

The face detection and tracking system was evaluated on live videos from webcam. Qualitative analysis was conducted to show the validity. Five publicly available face videos were used to test the system. Two videos include occlusion scenarios: *occluded face* (*faceocc*) and *occluded face 2* (*faceocc2*), which can be downloaded from http://vision.ucsd.edu/~bbabenko/project_miltrack.shtml. The rest three videos are: *davidin300*, *dudek*, and *trellis70*, which can be downloaded from <http://www.cs.toronto.edu/~dross/ivt/>. Detailed description of these five videos can refer to Table 4.6. The first frames along with the initial tracking window are presented in Figure 4.8.

Table 4.6: Description of test videos

	faceocc	faceocc2	davidin300	dudek	trellis70
Camera is moving	✓	×	✓	✓	✓
Occlusion	✓	✓	×	×	×
Off Frame	×	×	×	✓	✓
In-plane rotation	×	✓	×	×	×
Out-of-plane rotation	×	×	✓	✓	✓
Extreme lighting	×	×	×	×	✓
Put on/remove glasses/hat	×	✓	✓	✓	×

In this work, algorithms including Viola Jones Cascade detector, pure GUF tracker and combination of Cascade detector with GUF tracker were compared to qualitatively demonstrate the performance of proposed methods on videos. Detection outputs of frames are manually labeled as correct detection, no detection, off-center detection and false detection. Off-center detection refers to the case that most part of the face is located within the detected rectangle, but the face center can be clearly observed with certain offset from the rectangle center. False detection means the detected rectangle is obviously not on the face. Normally, no detection mainly occurs on Cascade detector, and off-center detection occurs more

Figure 4.8: Initial frames along with the initial tracking window

on tracking-based algorithms. As the major goal of this application is to show the usefulness of the unified detection and tracking algorithm, I used the parameters $\alpha = \beta = 1$, $\gamma = 0.01$, and $f_p = g_p(\sigma = 2)$ for GUF filter without special consideration. Meanwhile, it seemingly has no big changes with different settings because the most primary factor is that the filter should be augmented with latest frames.

Table 4.7 records the test results. It is observed that Cascade method could achieve good detection results under conditions without big rotations, off frame, occlusion or harsh lighting. On the other hand, GUF tracker is likely to yield off-center or even false detections under two cases: (i) the face is turning to and then turning back from profile, and (ii) the covering object (book, etc.) is slowly moving into or removing from the face. For case (i), the tracking window is partially occupied by background when the face is in profile, consequently the GUF filter is also updated and contaminated by the background information. After turning the face back to frontal viewpoint, it is often hard to accurately detect the face center

Table 4.7: Face detection results of algorithms including Cascade detector, GUF tracker, and Cascade + GUF (%).

Videos (# frames)	Algorithms	Correct Detection	No Detection	Off-center Detection	False Detection
faceocc (886 frames)	Cascade	83.0	17.0	0	0
	GUF	6.5	0	0	93.5
	Cascade+GUF	91.5	0	8.5	0
faceocc2 (815 frames)	Cascade	48.1	51.9	0	0
	GUF	100	0	0	0
	Cascade+GUF	100	0	0	0
davidin300 (462 frames)	Cascade	69.7	30.3	0	0
	GUF	32.0	0	68.0	0
	Cascade+GUF	96.3	0	3.7	0
dudek (573 frames)	Cascade	96.7	3.1	0	0.2
	GUF	79.2	0	20.8	0
	Cascade+GUF	100	0	0	0
trellis70 (501 frames)	Cascade	47.3	47.5	0	5.2
	GUF	100	0	0	0
	Cascade+GUF	100	0	0	0

again. This reason accounts for the off-center detections in videos *davidin300* and *dudek*. For the second case, a covering object first partially occludes the face, and then slowly slides away. This may cheat the filter to follow and track such fake target, which is so-called drift issue. This is also the reason for the false detections on video *faceocc2* with GUF tracker, and off-center detections with Cascade + GUF method.

Furthermore, Figure 4.9 shows the detection output for each frame. It explicitly demonstrates the superiority of the Cascade + GUF method. Moreover, it is worth mentioning that the pure GUF tracker additionally faces with the issue of fixed tracking size that is not reflected in the results. This relies on the video itself: if distance of face changes slightly from the initial frame, then the tracking window can fit the face well for subsequent frames, otherwise the detected rectangle may not fit the face well. Through the experiments, Cascade detector and the Cascade

+ GUF do not suffer from this problem. In addition, Figure 4.10 presents the detection results of many challenging frames with Cascade + GUF method. Full results for all frames could refer to my youtube website: <https://www.youtube.com/channel/UCwwzA0VY-vof7jSQFeJQM7A>.

Figure 4.9: This figure shows the detection results for each frame. Each resulting frame was manually labeled, where Green indicates a correct detection, Blue indicates no detection, Yellow indicates a detection with drifted off center, and Red indicates a false detection.

Figure 4.10: This figure shows many challenging frames from the videos, the rectangles give the results from Cascade + GUF method.

4.3 Summary

This chapter has introduced two applications to investigate two different aspects of GUF filters. First one is facial landmark localization, in which GUF is offline trained with a number of existing samples. Experiments on FERET database have shown GUF filters could obtain better localization accuracy over prior correlation filters as well as Viola Jones method. The performance of GUF filters can be further improved by adding an error compensation step. On the other hand, it should be pointed out that the experiments are limited to controlled frontal faces

without large rotation and scale difference. After all, GUF filters are linear filters that are insufficient to cover all complex appearance in one single filter.

Another application, face detection and tracking, was assessed to show the dynamic aspect of GUF filters. Starting with a single frame, GUF filters are initialized and continuously updated online to adapt to appearance changes of the face. By incorporating with Viola Jones face detector, scalability of GUF filters becomes possible, and automatic recovery from drifted tracking or lost tracking is realized. Experiments with five videos have been performed to support the conclusion that the proposed method, Cascade detector + GUF tracker, has promising performance in detecting faces with large rotations, occlusions, partially off frame, and harsh lighting conditions.

Chapter 5

Image Representation by Local Features

Novel feature extraction methods and related applications will be presented in this and next chapters. The contents summarize and extend the works in [96–98]. This chapter mainly introduces theoretical analyses: derivation of a new texture feature, the Local Intensity Increasing Patterns (LIIP), and followed by the introduction of one variant of LIIP, the Local Gradient Increasing Patterns (LGIP). Furthermore, Multiscale Cell LIIP (MC-LIIP) is proposed to extend the scope of LIIP.

5.1 Local Intensity Increasing Patterns (LIIP)

As described in Section 2.2, local features such as LBP are capable of representing micro-level textures such as edges, spots and corners. On the other hand, local gradient information has been proved to be a very good discriminative tool, either appearing in sparse form as the Scale Invariant Feature Transform (SIFT) [45] keypoint descriptor or in dense form as the Histogram of Oriented Gradients (HOG) [41] features. LIIP aims to inherit the simplicity and efficiency from LBP, and simultaneously encode certain gradient information to improve the discriminating ability.

5.1.1 Extraction of LIIP

Basically, the proposed LIIP feature is based on a concept called intensity increasing trend. Roughly speaking, it tells toward which direction and how much the local intensity increases. As shown in Figure 5.1, such trend always points from low-intensity area to high-intensity area.

Figure 5.1: This figure illustrates the concept of intensity increasing trend. Dark areas corresponds to low intensity, and bright areas corresponds to high intensity. Trend vectors always point from low-intensity side to high-intensity side.

Similar to LBP and LDP, LIIP encodes every pixel in an image with a decimal label to express the local structure or intensity distribution. Suppose a 3×3 intensity patch is given around one pixel, first the eight surrounding pixels are binarized to 1 or 0 according to intensity comparison results with the center pixel: these pixels whose intensity are greater than the center one will be encoded as 1, and otherwise encoded as 0. For example, Figure 5.2 gives an example. Consequently, an 8-bit bit word is obtained from eight directions. In a certain direction, code 1 means the intensity along that direction is increasing from center location toward corresponding surrounding pixel, while code 0 means the opposite trend. Formally, the intensity increasing trends with respect to these surrounding pixels

are expressed by eight so-called trend vectors (TVs) as indicated by solid arrows in Figure 5.2. Note the origin is located at the center pixel.

Bit string: 1110 0010

OTV = $(1,0) + (1,1) + (0,1) + (0,-1) = (2,1)$

LIIP label = 11

Figure 5.2: An example of LIIP pattern generation.

An individual trend vector can only tell the intensity increasing trend in that particular direction of a local patch. To obtain an overall trend, the overall trend vector (OTV) is defined as the sum of eight directional TVs. Via Figure 5.2, it is easy to conclude that the sum of 1-coded TVs (black solid arrows) is equal to the sum of 0-coded TVs (gray solid arrows) for symmetrically distributed neighboring pixels. For simplicity, the OTV is equivalently computed by adding only the 1-coded TVs. Hence, the TVs of 0-coded pixels are masked. In global viewpoint, direction of OTV is from low-intensity side to high intensity side in the local patch. It is capable of indicating local intensity increasing direction and even telling the rough magnitude. Eventually, the OTV is encoded to a decimal pattern label using a predefined two-dimensional pattern look up table (LUT) as displayed in Figure 5.3. The meaning of arranging the LUT in such way would be explained later. Let us look back to the example in Figure 5.2, the 8-bit string is extracted as:

1110 0010. Adding these 1-coded TVs, the resulting OTV is (2, 1). Then referring to the LUT, the corresponding pattern label is 11. In total, there are 37 LIIP patterns for a 3×3 neighborhood.

3			30	29	28		
2		16	15	14	13	12	
1	31	17	4	3	2	11	27
0	32	18	5	0	1	10	26
-1	33	19	6	7	8	9	25
-2		20	21	22	23	24	
-3			34	35	36		
	-3	-2	-1	0	1	2	3

Figure 5.3: Look up table for interpreting OTV to LIIP patterns.

5.1.2 Properties of LIIP

LIIP has many good properties that make it discriminative for image representation. Overall, it incorporates the strengths from classic LBP, uniform patterns, and gradient information.

First, it has inherited the main advantages from classic LBP features. For one thing, LIIP also possesses implementation simplicity and runtime efficiency because only eight pixel-wise comparisons and several addition operations are involved. For

another thing, LIIP is invariant to monotonic illumination changes as the results of pixel-wise comparisons are hardly influenced.

Second, the pattern size (37) is favorable for commonly adopted regional histogram based image representation. In contrast, ULBP and LDP respectively contain 59 and 56 patterns, and original LBP contains 256 patterns. Larger pattern size means the final feature size is larger. Meanwhile, it often happens that number of pixels in partitioned regions is less than number of patterns or histogram bins. Therefore, smaller pattern size leads to fewer redundant, informationless zero histogram bins.

Third, LIIP is able to make use of uniform patterns in ULBP. This is related to the answer why I define the pattern LUT as in Figure 5.3. In general, the peripheral patterns in the LUT belong to uniform patterns that carry more texture information, while these patterns that are located near the origin mostly belong to non-uniform patterns that are carrying less texture information. Figure 5.4 gives four samples. The left two patterns fall into the edge cells in the pattern LUT and belong to uniform patterns. Separately they could be viewed as informative micro-structures like edge and corner. The rest two patterns are located close to origin in the LUT, and belong to textureless non-uniform patterns.

Finally, LIIP provides rough implication of local gradients. Basically, larger pattern values in the LUT correspond to the local patches with more biased intensity distribution or higher gradient toward to OTV direction, while smaller values correspond to such patches that have no dominant gradient toward a specific direction. Still look at the four samples in Figure 5.4, qualitatively the first two patterns should have stronger gradient and the other two patterns have less biased intensity distribution.

Figure 5.4: Four examples of LIIP patterns. A bright dot indicates the pixel's intensity is higher than center pixel, and a dark dot indicates the pixel's intensity is lower than center pixel. Gray arrows represent 1-codes TVs, and red arrow represents the OTV.

5.2 Local Gradient Increasing Patterns (LGIP)

LGIP is considered as an extension from LIIP. In principle, it is also designed to qualitatively express the direction and magnitude of local intensity increasing trend. Even though LBP is robust to monotonic intensity changes, but it is very sensitive to non-monotonic illumination variations and random noise [69]. LIIP also faces with this problem. One major reason is that the resulting 8-bit string will be affected with non-monotonic intensity changes or white noise. To enhance the stability of LIIP operator, LGIP uses eight directional gradient masks to calculate the bit string instead of pixel-wise comparisons.

For each pixel in an image, first eight directional Sobel kernels $M_0, M_1, \dots, M_7$ as shown in Figure 5.5 are utilized to calculate gradient responses in eight orientations. Then the pixel is assigned with an 8-bit string according to the signs of these responses. These orientations with positive responses are set to one, and those with negative responses are set to zero. For instance, applying these kernels to an intensity pattern in Figure 5.6, eight neighboring pixels are encoded to 0011 1010. Similarly, 1-valued bits indicate positive gradients or the intensity of off-center side in corresponding orientation is greater than the center side. Subse-

Figure 5.5: Local gradient masks in eight directions.

quently, the OTV is computed to reflect the overall trend of local intensity. And the rest of LGIP follows the steps of LIIP operator.

Figure 5.6: An example of LGIP generation.

From the extraction procedure, LGIP operator is similar to LIIP operator except in the ways of generating the bit string. With regard to the properties of LGIP, it also inherits majority of the strengths from LIIP. First, LGIP is still easy and fast to implement even though it is more complex compared with LIIP. Meantime, it is invariant to monotonic intensity changes because signs of the gradient

responses are rarely influenced. Second, it also takes advantages of the idea of uniform patterns. Even though the surrounding eight bits are no longer generated by pixel-wise comparisons, these patterns with at most two $1 - 0$ or $0 - 1$ transitions could still be viewed as uniform patterns because they also imply higher intensity in consecutive angles of a local neighborhood and lower intensity in remaining angles. Third, each individual TV of LGIP qualitatively tells the gradient in certain direction, and OTV is able to qualitatively depict the global gradient of a local patch. Same as in LIIP, bigger LGIP patterns represent those intensity patches that have more biased intensity distribution or bigger gradients, and verse vice. Besides, dimensionality of LGIP is also 37.

In addition, LGIP possesses better stability in the presence of non-monotonic intensity changes and white noise like LDP [69]. Taking an example from [69], the intensity patch shown in Figure 5.7 is generated by adding white noise (circled pixel) and non-monotonic intensity changes to the original patch displayed in Figure 5.6. To simulate non-monotonic lighting changes, intensities of pixels lying in the southeast are artificially increased and intensities of those pixels lying in the northwest are reduced. As a result, the 5th bit of LBP and LIIP changes from 1 to 0 due to the adding white noise, because it results from pixel-to-pixel comparison. In contrast, LDP and LGIP adopt more stable gradient operators, hence the bit string or pattern label could maintain the same in the presence of the adding noise and intensity changes. Figure 5.7 also presents the resulting bit string of LBP, LDP, LIIP and LGIP.

5.3 Multiscale Cell LIIP (MC-LIIP)

One prominent limitation of the original LIIP is the small spatial support area. Features calculated in a 3×3 neighborhood can only capture textural micro-structures

Figure 5.7: Stability of LBP, LDP, LIIP and LGIP.

and are incapable of providing information on large-scale macro-structures, which are more stable and may be the dominant textures. Hence, the motivation of MC-LIIP is to enlarge the support area of LIIP, and simultaneously enhance the resistance ability of LIIP against noise.

In MC-LIIP, the smallest operating unit is based on small cells instead of pixels. At a scale of s , the size of a cell is $s \times s$, so the spatial support area of MC-LIIP operator has a size of $3s \times 3s$ as shown in Figure 5.8. To facilitate the implementation, a set of average images are first obtained by computing the mean values of non-overlapping $s \times s$ cells from an original image. Note that the original image could be viewed as the average image at scale of $s = 1$, where the cell size is 1×1 . For the rest scales, the size of resulting average image at scale s is $\frac{1}{s}$ of

the original image. In Figure 5.9, the left image in each pair shows the cell-based average image.

Figure 5.8: Spatial support areas of MC-LIIP operator at different scales.

Figure 5.9: This figure shows the resulting average images by applying scalable cells and corresponding MC-LIIP pattern images at different scales. Note the pattern images are scaled up by six times for display purpose.

Based on these multiscale cell-based average images, LIIP pattern images are calculated using standard LIIP algorithm. And that is why I name it Multiscale Cell LIIP (MC-LIIP) features. For notation convenience, MC-LIIP features at

scale s is denoted by MC-LIIP_s . To recap, each pixel in pattern image MC-LIIP_s could equivalently be viewed as being evaluated from nine $s \times s$ cells in original image. In Figure 5.9, the right images in the pairs represent MC-LIIP pattern images for the first four scales $s = 1, 2, 3, 4$. It is observed from these images that the forehead and cheeks below the eyes are mostly encoded to textureless MC-LIIP patterns with label 0 (darkest), because these areas have been deteriorated due to strong highlights. It is worth mentioning that the cell-based averaging can be efficiently computed by direct summing for small s and by integral image for large s . Consequently, extraction of MC-LIIP features is still very fast.

Obviously, one advantage of MC-LIIP is that the pattern generation process becomes more robust for bigger cells despite the presence of random noise or lighting changes. Intuitively, sum of a bigger cell will be less affected by such negative factors, hence the cell-to-cell comparison results are more stable. Figure 5.10 illustrates the MC-LIIP_s ($s = 1 - 4$) images of two emotional face images. For better illustration, the average images are computed from overlapping cells rather than non-overlapping cells, so the resulting average and pattern images are of the same size as original images. Clearly, it is observed that MC-LIIP_s images with small s present small-scale, noisier and micro patterns of face structures, which is more suitable to describe local detailed face features. On the other hand, MC-LIIP_s images with large s demonstrate better noise resistance, and depict relatively large-scale, regional-level, and macro patterns of face structures, but more local detailed features are dropped. To summarize, MC-LIIP features at different scales could provide complementary information for each other, hence together they get a more complete description of face images. In addition, through the pattern images in Figure 5.10, we can easily observe the differences between the joy and surprise faces in many areas including the forehead, eyes, and mouth. Overall, MC-LIIP features would perform well for face representation.

Figure 5.10: This figure shows the MC-LIIP images of two emotional faces. The left column contains the original images. Rows 1 and 4 represent the average images computed from overlapping cells of face regions in the original images. Rows 2 and 3 are the corresponding MC-LIIP_s images (scaled up by six times for display). Totally four scales are illustrated.

5.4 Summary

In this chapter, a class of local texture features has been introduced. Inspired by LBP, the proposed Local Intensity Increasing Patterns (LIIP) uses decimal numbers to express local intensity structures. Interestingly, LIIP patterns qualitatively

reflect to which direction and to what extent the intensity of local patches increase. In addition, LIIP features make good use of the merits from many existing concepts including LBP, uniform patterns, and gradients. Another proposed feature is the Local Gradient Increasing Patterns (LGIP), which utilizes eight Sobel kernels to generate binary bits for the central pixel's neighbors compared to LIIP. The design of LGIP aims to provide extra stability against random noise and certain non-monotonic intensity variations. Lastly, LIIP is extended from pixel-level to multiscale cell-level, forming the Multiscale Cell LIIP (MC-LIIP). Theoretically, the MC-LIIP features could represent both local micro-patterns and macro-patterns of an image by using various scales.

Chapter 6

Facial Image Analysis Using Proposed Features

This chapter presents three face related applications of the proposed features described in previous chapter, namely facial expression recognition (FER), gender recognition (GR) and eye state recognition (ESR). The reasons to choose these three applications as listed as follows. First, the central topic of the thesis is on facial image analysis. Second, the proposed feature representations are closely related to LBP, and LBP has gained widespread recognition because of its success in facial representation. Hence, FER and GR are the most suitable applications. In addition, FER and GR separately represent multi-class classification and binary classification cases. Finally, ESR shows the possible diversity of using the proposed features. Instead of grid-based representation, eye images are much smaller and feature extraction scenario differs from FER and ER. Accordingly, this chapter is composed of three parts.

6.1 Facial Expression Recognition

6.1.1 Introduction

Facial expression recognition provides a natural and immediate way to determine what is saying on his/her face. In 1978, Ekman and Friesen defined six basic types of

human expressions: anger, disgust, fear, joy, sadness, surprise [99]. Often, neutral expression is considered as the seventh basic expression. In general, these basic expressions could be generalized over all human beings regardless of the regions and races. Although numerous efforts have been contributed to the developing of better algorithms for expression recognition, but facial expression recognition with a high accuracy still remains an open task in computer vision field. On the other hand, FER has invaluable significance in many applications such as human computer interaction, lie detection, data-driven animation and video indexing [1].

Generally, facial expression analysis involves two aspects: facial feature representation and classifier design. This thesis focuses on the former aspect. Broadly, existing methods for face representation could be classified into two categories: geometric feature based method, and appearance feature based method. Meanwhile, according to whether temporal information is used, FER could also be divided to frame-based and sequence-based methods.

Basically, successful FER relies greatly on discrimination ability of face descriptor. A robust face descriptor should be able to aggregate faces with identical expressions and split those faces from different expressions. Geometric features make use of either the shape of face components or coordinates of facial points and their relations, and then construct a shape or structure model to represent face images. For example, a constrained local model was used to represent expression faces based on a number of tracked facial points in [100]. In [99], a face image was partitioned into dozens of *action units*, where movements of certain muscles were encoded to identify particular expressions.

Comparatively, FER based on appearance features depends on intensity distribution or texture information across entire face or face regions. It can avoid the step to accurately locate facial points. In literature, Gabor wavelets have been proved

to perform well in recognizing facial expressions [66, 67]. However, one shortcoming is that it is time-consuming to convolve the input image with a bank of Gabor filters. In recent years, Local Binary Patterns (LBP) has been repeatedly adopted for facial image analysis. Shan et al. presented a comprehensive study on FER based on LBP feature [12]. To boost feature stability of LBP, the paper [15] introduced face descriptor based on Local Directional Patterns (LDP) to identify face expressions. In addition, favorable performance has been achieved by combining two features including similarity-normalized shape (SPTS) and canonical appearance (CAPP) features [23]. Once the feature representation is decided for expression faces, advanced pattern recognition techniques are commonly adopted to determine the expression labels. Such techniques include but not limited to template matching, linear programming, subspace methods like eigenfaces and fisherfaces, SVM, neural network (NN), AdaBoost and hidden Markov models (HMMs).

FER consists of two practical scenarios: frame-based and sequence-based recognition. Intuitively, frame-based methods independently treat every static image and output a separate class label for it [12, 15, 23, 66, 67], while sequence-based methods deal with a short video sequence at one time and assign a single label to the entire sequence. Either geometric features or appearance features may be applied to represent individual frames. Currently, the majority of existing approaches belong to frame-based type, because it is relatively simpler to separately process individual frame than work out a spatial-temporal model to link all frames. On the other hand, sequence-based methods are able to utilize dynamic facial motion information, which can help to perform more robust expression recognition [11]. The shortcoming is that methods of this type often require to accurately track particular facial features, and the tracking accuracy will directly affect recognition rate in FER. In [101], horizontal and vertical components of the flow were used as

features, and discrete HMMs were trained to dynamically recognize expressions for video sequences. And in [102], automatic FER was accomplished by using facial animation parameters and multi-stream HMMs. Another paper [103] incorporated Gabor wavelets and genetic algorithm (GA) for dynamic recognition of facial expressions. Additionally, Zhao et al. [11] proposed two local spatial-temporal features for face representation: the Volume Local Binary Patterns (VLBP) and LBP from three orthogonal planes (LBP-TOP), and combined together with SVM classifier for FER.

6.1.2 Multiscale Feature Fusion and Classifier Design

The contents for FER are mainly extracted from the work in [98]. This section introduces the algorithms to extract face descriptor and train expression classifier.

6.1.2.1 Face Representation via MC-LIIP

The adopted face descriptor is based on histograms of local patterns expressed by MC-LIIP feature. Intuitively, direct use of raw pixels in the pattern images would be able to fully keep the spatial information of these local textures. However, such a descriptor tends to be very susceptible to noises and intra-expression variations. Moreover, all face images should be well aligned. In one word, face representation by raw pattern pixels is too local to being discriminative. On the other hand, global representation by a single pattern histogram would give up all the information about the locations of the local patterns. Therefore, often a compromise is obtained between above two extreme cases. As the size of face images is sufficiently large, grid-partition is applied to the images, and regional histograms are extracted from the grid cells.

Formally, Figure 6.1 illustrates the overall procedure to fuse multiscale local features to represent faces for FER. Given a normalized face image, non-overlapping

cell-based MC-LIIP_s images are first obtained at L scales $s = 1, 2 \dots L$. Then, at each level s , the resulting image MC-LIIP_s is partitioned to M_s regions denoted by $R_{s0}, R_{s1} \dots R_{sM_s}$. In each region, histogram of local patterns is extracted and normalized to unit length. Mathematically, the i th histogram bin before normalization from the j th region in image MC-LIIP_s is defined as:

$$H(i, j, s) = \sum_{(x,y) \in R_j} 1\{\text{MC-LIIP}_s(x, y) = i\} \quad (6.1)$$

where $1\{*\}$ is the zero/one indicator function. After normalizing each regional histogram, all the bins from various regions and scales are finally concatenated into a single feature vector, which forms the face descriptor.

Figure 6.1: This figure illustrates the generation of face descriptor for expression recognition based on MC-LIIP feature.

Naturally, the proposed descriptor is more comprehensive because textures at various scaling levels are encoded. That is to say, such descriptor could reflect textural and statistical features in many levels. Specifically, the MC-LIIP operator

operates in pixel and scalable cell levels, regional histograms represent information in block level, and the united feature vector represent faces at a global level. Overall, the dimensionality of the face descriptor is $37 \sum_{s=1}^L M_s$.

6.1.2.2 Classifier for Expression Recognition

Based on the face descriptor, multi-class SVM classifier was utilized to split different facial expressions. In practice, SVM has been repeatedly proved to be a powerful tool in FER [12, 15, 66]. Basically, in two-class case, SVM maps input data into a higher or infinite dimensional feature space, and then searches for a linear separating hyperplane with maximal margin to separate the data in this new space.

Given a training set $\{(x_i, y_i)\}$, SVM endeavors to optimize the following objective function:

$$\begin{aligned} \min_{w, b, \xi} : \quad & \frac{1}{2} w^T w + C \sum_{i=1}^N \xi_i \\ \text{subject to:} \quad & y_i(w^T \Phi(x_i) + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0 \end{aligned}$$

Here the training vectors x_i are mapped into the higher dimensional space by the function $\Phi(x_i)$, and $C(C > 0)$ is the penalty for a sample's error ξ_i . Additionally, $K(x_i, x_j) = \Phi(x_i)^T \Phi(x_j)$ is called a kernel function, which is an important concept in SVM theory. Three classic types of kernel functions are:

- linear: $K(x_i, x_j) = x_i^T x_j$
- polynomial: $K(x_i, x_j) = (\gamma x_i^T x_j + r)^d, \gamma > 0$
- radial basis function (RBF): $K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2), \gamma > 0$

where γ , r and d are kernel parameters.

For a new sample x , the label is determined by the decision function:

$$f(x) = \text{sign}(\sum_i \alpha_i y_i K(x_i, x) + b) \quad (6.2)$$

where (x_i, y_i) are support vectors, α_i are Lagrange multipliers, b is the hyper-plane parameter, and K is the kernel function. A tutorial introduction is presented in [104]. As linear kernel is very fast, and evaluation on different SVM kernels is not the main concern in this work, hence the experiments only use linear SVM for classification. Original SVM is applied for binary classification issue, and one-against-rest strategy is adopted to perform multi-class expression recognition. Following the tutorial in [104], SVM parameter settings are optimized through grid-search during 10-fold cross-validation. These values that can achieve best validation accuracy are selected. The OpenCV (<http://opencv.willowgarage.com/wiki/>) implementation of SVM was used in the experiments.

6.1.3 Experiments

The experiments are performed to evaluate the proposed methods in recognizing six basic facial expressions as well as seven expressions by adding the neutral one. Evaluation was mainly conducted with the most frequently used database: the extended Cohn-Kanade (CK+) database [23]. The influence of grid-partition and scales for MC-LIIP based face descriptor will be also investigated. Besides, algorithm comparisons are carried out between the proposed method and many other state-of-the-art methods. At last, recognition results would also be provided on the Karolinska directed emotional faces (KDEF) database [24].

6.1.3.1 Experimental Data and Setup

Since year 2000, the Cohn-Kanade (CK) database [105] is the most frequently used test-bed for FER. The CK dataset contains 486 captured frame sequences

from 97 persons. However, it is often difficult to perform fair comparisons with existing algorithms because ground truths of expression labels are not available. Commonly, expression labels of many sequences are really hard to distinguish. As a result, different authors may select and manually label different subsets of these sequences for algorithm evaluation. The newer distribution, CK+ database, is an augmented version of CK database, and newly adds a percent of 22% sequences and 27% persons. Besides, CK+ practically provides the ground truth labels for 309 sequences from 106 persons. Obviously, experiments using CK+ would be repeatable, so CK+ was chosen as the main testing dataset.

In detail, the CK+ dataset consists of 123 subjects that are aged from 18 to 50 years. Statistically, it contains about 69% females, and 81% Euro-American, 13% Afro-American, and 6% other groups. In each sequence, the face starts from a neutral expression, and gradually grows to the peak of one of remaining six basic expressions. The frames of each sequence are with a size of 490×640 or 480×640 . Using conventional way [12, 15], the first neutral frame and last three frames with peak expressions were selected from the 309 labeled sequences. Consequently, the dataset has 1236 images including 135 in anger expression, 177 in disgust, 75 in fear, 207 in joy, 84 in sadness, 249 in surprise and 309 in neutral. As suggested in [12, 15, 67], face images are first manually cropped out of background and then normalized to a standard size of 110×150 by keeping a fixed interocular distance. There are no further preprocessing steps like lighting normalization to be applied. Indeed, the MC-LIIP operator possesses certain resistance against illumination issue as described before.

To guarantee recognition generality ability for new persons, person-independent 10-fold cross-validation was implemented. At first, all the 106 persons were randomly split to ten groups with roughly identical numbers of persons in each group.

The expression classifier was trained on nine groups and tested with the rest one group. Every group got one-time chance of being the test group, hence the process was repeated ten times. Finally, the average results were calculated.

6.1.3.2 Determination of Face Partitions

As discussed before, the pattern images MC-LIIP_s should be divided into non-overlapping regions, where sub-histograms would be extracted. Here I would like to investigate the impact of grid partition on the recognition results. Take scale $s = 1$ for example, Table 6.1 records the 6-class and 7-class FER results of using different image partition schemes. Through the table, we can observe that either too few or too many cells will downgrade the recognition accuracies. Undoubtedly, the more regions the partition generates, the larger the feature size is because histogram length of each region is fixed at 37. In addition, finer partition would retain richer spatial location information of local texture structures, and at the same time suffer a higher risk of being less reliable. In contrast, large-scale cells possess steady description ability at a sacrifice of losing more spatial information. From Table 6.1, the partition with a grid of 8×9 is optimal for MC-LIIP_1 . In a similar manner, the optimal grid partition schemes for MC-LIIP_2 , MC-LIIP_3 and MC-LIIP_4 are 7×8 , 5×6 and 5×6 , respectively.

6.1.3.3 Evaluation of MC-LIIP Features

In this part, evaluation experiments will be first presented to examine performance of the face representation based on individual scale s as well as combination of several scales. Notably, the notation MC-LIIP_{abc} represents a fusion of MC-LIIP features in scales $s = a, b, c$. Eventually, Table 6.2 records the recognition accuracies of FER with MC-LIIP features at various scales. Obviously, MC-LIIP_{1234} features that concatenate all four scales have demonstrated the highest accuracies. On

Table 6.1: Recognition rates of MC-LIIP₁ based face descriptor with different grid partitions

Partitions	Feature size	6-class (%)	7-class (%)
5×6	1110	91.9 ± 5.1	89.0 ± 3.9
6×7	1554	91.9 ± 4.5	89.2 ± 4.1
7×8	2072	93.6 ± 4.4	89.9 ± 4.3
8×9	2664	93.7 ± 4.0	91.8 ± 3.7
9×11	3663	93.1 ± 3.9	90.7 ± 3.8

average, it could reach an accuracy of 95.3% and 93.2% separately for 6-class and 7-class expression recognition. Intrinsically, MC-LIIP₁₂₃₄ fuses texture features in terms of local micro-structures and macro-structures at four different scales. For MC-LIIP features of an individual scale, it is observed MC-LIIP₂ roughly exhibits best performance. The possible reason is that it could obtain a better balance between descriptor stability and local details.

Table 6.2: Recognition results of MC-LIIP_s features at various scales

Features	Feature size	6-class (%)	7-class (%)
MC-LIIP ₁	2664	93.7 ± 4.0	91.8 ± 3.7
MC-LIIP ₂	2072	94.7 ± 3.7	91.7 ± 3.8
MC-LIIP ₃	1110	94.1 ± 3.9	90.7 ± 3.8
MC-LIIP ₄	1110	93.7 ± 3.3	88.4 ± 3.5
MC-LIIP ₁₂	4736	94.1 ± 4.0	92.1 ± 3.7
MC-LIIP ₁₂₃	5846	94.6 ± 3.8	92.8 ± 3.7
MC-LIIP ₁₂₃₄	6956	95.3 ± 3.6	93.2 ± 3.4

Furthermore, we look into the recognition rates for specific facial expressions. Tables 6.3 and 6.4 provide the confusion matrices for the method of using MC-LIIP₁₂₃₄ and linear SVM. Comparing with the reported accuracies in work [12] (LBP + RBF

SVM), the proposed method could achieve higher or at least competitive recognition rates in every individual expression. Moreover, from the results, seemingly many expressions such as disgust, joy, surprise and neutral can be nicely distinguished, while fear and sadness expressions tend to get confused. This observation is consistent with what was claimed in [12]. Interestingly, it is found that happiness or joy turns to be the simplest expression, while sadness is the hardest expression.

Table 6.3: Confusion matrix of 6-class facial expression recognition using MC-LIIP₁₂₃₄

	Anger (%)	Disgust (%)	Fear (%)	Joy (%)	Sadness (%)	Surprise (%)
Anger	93.2	1.7	0	0	4.3	0.8
Disgust	0.9	99.1	0	0	0	0
Fear	1.3	0	87.2	4.1	0	7.4
Joy	0	0	0	100	0	0
Sadness	15.2	1.1	2.4	0	78.1	3.2
Surprise	0.1	0.4	0.2	0.1	1	98.2

6.1.3.4 Comparisons with Other Methods

In literature, most of existing FER approaches belong to frame-based type, and a few others belong to sequence-based type. In principle, the test-beds or evaluation data for FER methods of these two types are different: frame-based type can use independent frames, while sequence-based method must use consecutive video sequences. With regard to evaluation database, existing methods of both types mostly like to use the CK/CK+ database [1, 11, 12, 67, 69, 103, 106]. This data is suitable to evaluate algorithms of either frame-based or sequence-based type, so it is also adopted to perform algorithm comparisons with state-of-the-art approaches.

Table 6.4: Confusion matrix of 7-class facial expression recognition using MC-LIIP₁₂₃₄

	Anger (%)	Disgust (%)	Fear (%)	Joy (%)	Sad (%)	Surprise (%)	Neutral (%)
Anger	84.9	1.2	0	0	0.2	0	13.7
Disgust	0	96.4	0	0	0	0	3.6
Fear	0.4	0	78.6	4.1	0	4.4	12.5
Joy	0	0	0	100	0	0	0
Sadness	5.6	0	0	0	68.7	0.6	25.1
Surprise	0	0	0.1	0	0	96.6	3.3
Neutral	0.8	0	0.1	0	0.7	0.4	98.0

For the competing frame-based methods, commonly different face representation algorithms are utilized, and then expression recognition is accomplished by a SVM classifier unless otherwise stated. To make it clear, Table 6.5 summarizes the parameters for extracting the MC-LIIP₁₂₃₄ descriptor used in this experiment. Details on selection of these parameters can refer to previous experiments.

Table 6.5: Parameter settings in facial expression recognition

Image resolution	Scale (s)	Cell size	Grid partitions	Total block number
110×150	$s = 1$	1×1	8×9	188
	$s = 2$	2×2	7×8	
	$s = 3$	3×3	5×6	
	$s = 4$	4×4	5×6	

In Table 6.6, classification rates are compared between MC-LIIP based method and other reported results of FER algorithms with CK/CK+ database. Specifically, the table lists experiment setups of different FER algorithms in terms of number of persons (NP), number of sequences (NS), number of expression classes (NC), whether person-independent (PI) or not, dynamic (sequence-based) or static (frame-

based), and evaluation metrics. The last column provides the overall recognition rates for all expressions.

Table 6.6: Algorithm comparisons between different approaches using CK/CK+ database. Note: loso is the abbreviation for leave-one-subject-out, and number of classes (NC) for SPTS+CAPP method is 8 by adding an extra contempt expression.

Method	NP	NS	NC	PI	Dynamic	Metric	Accuracy (%)
Gabor (Neural Network) [67]	97	375	6	N	N	-	92.2
Gabor [66]	90	313	7	Y	N	losa	88.0
LBP [12]	96	320	7(6)	Y	N	10-fold	88.1(91.5)
LDP [15]	96	408	7(6)	N	N	7-fold	92.8(94.9)
SPTS+CAPP [23]	-	327	8	Y	N	losa	88.3
Optical flow + HMM [101]	97	-	6	Y	Y	5-fold	90.9
Multi-stream HMM [102]	90	284	6	Y	Y	-	93.66
Spatio-temporal LBP [11]	97	374	6	Y	Y	10-fold	96.26
Gabor + GA [103]	115	321	6	Y	Y	losa	92.97
MC-LIIP ₁₂₃₄	106	309	7(6)	Y	N	10-fold	93.2(95.3)

Specially, it is necessary to clarify the reason that different approaches used varying NP and NS as indicated in the table. As discussed above, before release of the new distribution CK+ in 2010, there were no standard expression labels for the sequences included in CK database. What's worse, many sequences are very difficult to assign with a confident label by human perception. Consequently, authors commonly selected these sequences that they were confident in labeling. In CK+, totally 309 sequences with seven expressions (including neutral) are clearly labeled, hence the proposed method is evaluated with all these sequences. Indeed, this could facilitate the repetition of experiments or algorithm comparisons in the future.

Look back to the results in Table 6.6, we can observe that the MC-LIIP₁₂₃₄ method can achieve higher accuracies than other frame-based methods (including Gabor-based [66, 67], LBP-based [12], and LDP-based [15] methods) for both 6-class and 7-class facial expression recognition. Moreover, the proposed method also outperforms most dynamic or sequence-based methods. Exceptionally, in 6-class recognition, the spatial-temporal LBP method [11] is slightly superior to MC-LIIP₁₂₃₄ based method which extends the original LBP by adding the temporal plane. Taking into account that the accuracy difference is marginal and many different experimental setups such as NP and NS, still it could be concluded that the performance of MC-LIIP based approach is reasonably satisfactory.

In addition, particular emphasis was placed on the comparisons with LBP and LDP based methods, because they have adopted similar architecture with MC-LIIP method. Overall, MC-LIIP exhibits better performance over other two methods. Specially, LDP-based approach [15] reported competing results, however, they were not based on person-independent FER experiments. To remove uncertain factors, experiments were implemented on LBP-based and LDP-based approaches using the same setup as MC-LIIP₁₂₃₄ (see Table 6.6). Consequently, the successful recognition rates for 7(6)-class FER by LBP-based and LDP-based methods are 89.4% (91.1%) and 90.4% (92.2%). Clearly, these figures are inferior to that of MC-LIIP based method. With regard to the speed, averagely LBP, LDP, MC-LIIP₁ and MC-LIIP₁₂₃₄ respectively consume 0.574 milliseconds, 2.594 milliseconds, 0.518 milliseconds and 0.890 milliseconds to process one frame on a 3.2 GHz i5 CPU.

6.1.3.5 Experiments on other Database

The MC-LIIP based SVM approach was further evaluated on another publicly available database: the Karolinska Directed Emotional Faces (KDEF) [24]. In

FER field, it seems the KDEF has not attracted much attention, the possible reason is that the material was originally developed to be used for psychological and medical research purposes. However, it is believed the KDEF dataset is really valuable for FER research. Currently, there are only a few facial expression databases free for download. For example, the Japanese Female Facial Expression (JAFFE) database [107] was frequently used to assess FER algorithms in early years. However, JAFFE contains only 213 images posed by 10 Japanese female models. Relatively, the KDEF set is a large-scale and more balanced facial expression database. It contains 70 subjects, each displaying seven basic expressions, and each expression being captured (twice) from five different angles ($0, \pm 45, \pm 90$ degrees). The experiments used all the frontal-view (0 degree) set which consists of 980 faces with a size of 562×762 . The set contains 70 persons including 35 females, and each person has 14 images with two for every expression. Compared with CK/CK+, the data has no bias among different expressions, genders and individuals.

Similar experimental setup and preprocessing were adopted, and 10-fold cross-validation was implemented to evaluate the performance of person-independent FER with KDEF dataset. Consequently, the hit rates for 6-class (7-class) FER by MC-LIIP based SVM classifier are: anger - 82.5% (81.5%), disgust - 86.4% (86.4%), fear - 69.7% (66.0%), joy - 97.8% (97.8%), sadness - 79.0% (77.2%), surprise - 90.9% (90.7%), neutral - (92.2)%. Table 6.7 shows the experimental results of many FER methods. Particularly, only frame-based methods are included because the KDEF is composed by a collection of isolated frames. Moreover, the original motivation is to mainly compare MC-LIIP with LBP and LDP based methods as they are closely related. From the results, it can be seen that the proposed method outperforms the others. Furthermore, the performance difference between 6-class and 7-class

recognition is very subtle, which differs from that in CK/CK+ database. The reasonable explanation is that the numbers of images for different expressions vary greatly (especially for neutral expression) in CK/CK+, while maintain identical in KDEF.

Table 6.7: Facial expression recognition results on KDEF database.

Method	6-class accuracy(%)	7-class accuracy(%)
LBP [12]	81.6 $\pm$ 4.1	82.0 $\pm$ 3.8
LDP [15]	81.7 $\pm$ 4.0	81.7 $\pm$ 4.2
SPTS+CAPP [23]	-	82.8
MC-LIIP ₁₂₃₄	84.4 $\pm$ 3.9	84.5 $\pm$ 3.7

6.2 Gender Recognition

6.2.1 Introduction

Intuitively, gender recognition via face images is to determine whether the subject is male or female based on facial image analysis. It quickly becomes one heated subtopic with the growth of facial analysis techniques. In practice, face based gender recognition has already demonstrated its invaluable significance in many applications such as human computer interaction, visual surveillance systems, effective image and video search, and consumer behavior analysis systems. Particularly, automatic gender recognition could also be used as a pre-filtering step for face recognition, because it may reduce half efforts by searching matched faces from one gender. Despite a good number of applications, gender recognition with high accuracy still requires to deal with many challenges. The major difficulty is due to the high variability of human faces because of many uncertain factors including lighting influence, make-ups, occlusions, etc. In fact, these negative factors also exist in other face related issues like face recognition, facial expression recognition.

As a pattern classification issue, gender recognition also involves with two key points: feature extraction and pattern classification. For the former point, the simplest way is directly using the raw pixels [108]. One type of alternative methods is based on subspace theories such as Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA) and Independent Component Analysis (ICA), which map faces to a low-dimensional space and perform gender recognition [109]. On the other side, gender classification techniques include but not limited to nearest neighbor, support vector machines (SVM), Bayesian classifier, neural networks and AdaBoost classifiers.

Since 1990s, many efforts have been made to developing more successful gender recognition algorithms. In the work [110], Gutta et al. presented one of the earliest gender recognition proposals that were conducted with a large dataset. They adopted an ensemble of RBF networks and decision trees, and evaluation was performed on 3006 face images of 1009 persons from FERET database [22]. But the images from same person could appear in both training and testing sets, which may drive the learning process to recognizing faces rather than genders. In one word, the work conducted a person-dependent gender recognition. Moghadam and Yang [108] also utilized the FERET database at a low resolution, and gender recognition was implemented by combining raw pixels with SVM classifier. Similarly, person-dependent evaluation was carried out and it achieved a high accuracy. In paper [111], a systematic study was presented to find out how the gender recognition accuracy was affected by face alignment.

Instead of working out new classifiers, there is a rapid growth in developing advanced feature extraction techniques in recent years. Rather than using raw pixels of whole face as input, it is more preferred that high-level features are extracted from pre-defined face regions and then concatenated together. This aims

to reach a compromise between spatial information and feature stability as discussed in contents of facial expression recognition. Face related research with local features becomes one hottest branch. Particularly, one outstanding representative is the LBP features, which was first applied to recognize faces in [10]. Later, Sun et al. extended the applications of LBP-based descriptor to gender recognition using AdaBoost classifier on FERET database [112]. Shan [113] also adopted similar LBP-based face descriptor to perform gender recognition on real-world face images. Differently, AdaBoost was first used to select the discriminative LBP features, and followed with the SVM classifier to make final decision. Under similar framework of LBP, Jabid et al. proposed Local Directional Pattern (LDP) to perform gender recognition with SVM. In addition, contrast information and LBP were combined together to derive the feature representation, and then classify face genders in [114].

However, it is lack of standard evaluation platform. Plenty of works often have one or two of the following issues during experimental evaluation. First, non-replicable image sets are used in performance evaluation, which makes it hard for future experiment repetition as in the papers [69, 112]. Second, several images from same person appear in both training and testing data, which leads to person-dependent recognition [69, 110, 114]. Therefore, this work tends to conduct a person-independent gender recognition, and guarantees the used image lists could be exactly accessible for future algorithm comparison. The contents mainly come from my conference paper [97].

6.2.2 Face Representation and Classifier Design

This part is going to introduce the face descriptor as well as classifier used to classify genders. Basically, the performance of LGIP features is examined in this application, and the binary classifier still relies on SVM.

Given a face image, preprocessing is first applied before extracting features. The image is first transformed to a standard size via keeping two eyes in a horizontal level and with a fixed interocular distance. Then illumination normalization is implemented by histogram equalization. Actually, this is not a must step as LGIP operator owns certain resistance against lighting changes. In order to investigate performance under different resolutions, this thesis has adopted image resolutions at three distinct levels: 128×128 , 72×72 and 36×36 .

Normalized face images are converted to feature vectors according to the flowchart displayed in Figure 6.2. First, the LGIP pattern image $f_L(x, y)$ is extracted from the normalized image by applying LGIP operator to each pixel. Similarly, the face descriptor depends on regional LGIP histograms, which describe statistical distribution of local structures with certain spatial information. Alternatively, this time the LGIP image is partitioned to overlapping blocks denoted by $R_0, R_1, \dots, R_{M-1}$. In each block R_i , the j th histogram bin is computed as:

$$H(i, j) = \sum_{(x,y) \in R_i} 1(f_L(x, y) = j), \quad 1(X) = \begin{cases} 1, & X \text{ is true} \\ 0, & X \text{ is false} \end{cases} \quad (6.3)$$

where the region index i and bin index j satisfy: $i \in [0, M - 1], j \in [0, 36]$. Every regional histogram is normalized to have a unit length and all of them are concatenated to the final feature vector. Overall, the feature size is $37M$, depending on the number of regions. Look back to region partition, Table 6.8 summarizes the parameter settings. The three resolutions 128×128 , 72×72 and 36×36 respectively correspond to a block size of 16×16 , 12×12 and 6×6 . Besides, each block is half overlapped by four nearby blocks (top, left, right and bottom regions if exist), which means to shift the block window horizontally or vertically by half of the block size. Totally, the above three resolutions separately include 255, 121 and 121 regions, meaning a feature size of 8325, 4477 and 4477.

Figure 6.2: The process to extract face descriptor based on LGIP features. The left image represents the normalized face image, and middle image represents LGIP pattern image, which is scaled up by six times for display. The pattern image is partitioned into grid-based regions (50% overlapping ratio), where LGIP histograms are extracted and combined.

Table 6.8: Parameter settings in gender recognition

Image resolution	Block size	Block overlapping ratio
128×128	16×16	50%
72×72	12×12	
36×36	6×6	

According to the introduction, SVM classifier is the most frequently used technique in gender recognition. Hence, experiments here also adopt it to classify male and female face images. Regarding to the kernels, linear and radial basic function (RBF) are examined. As suggested in [104], grid-search and 5-fold cross-validation are utilized to determine the SVM parameters. Again, this work directly uses the SVM implementation in OpenCV library.

6.2.3 Experiments

6.2.3.1 Experimental Data and Setup

Tests are carried out on two data sets which are respectively collected from CAS-PEAL-R1 database [115] and gray FERET database [22].

CAS-PEAL-R1 database contains 30,900 images of 1040 individual with varying pose, expression, accessory, and lighting (PEAL). The images are with a size of

480×360 , and publicly available for download. For the purpose of gender recognition, the experiments use a subset named *normal* which is composed by taking one frontal image for each person. Overall, this set contains 1040 images with 595 males and 445 females. Obviously, the list of used images is easily obtainable.

The second set is collected as a subset of FERET database. Similar to CAS-PEAR-R1, FERET is also a large-scale face database that consists of 14,051 grayscale images of 1199 individuals. Here the thumbnails with a size of 384×256 are utilized. The evaluation set comes from the combination of two subsets in the database. One is the *fa* gallery set, which contains 1196 images from 1196 subjects (including 698 males and 498 females). The name list of these images can be found in the file named *gallery.names* in FERET database CDs. The other subset is *fb* probe set, which contains 1196 images from 1195 individuals (including 687 males and 498 females). Similarly, the name list can be found in the file named *probe_fafb_expression.names*. Generally, *fb* images were captured from the same persons as *fa* by adding extra facial expressions. To summarize, this evaluation set totally contains 2391 images of 1196 individuals with 698 males and 498 females. Lastly, it worth mentioning that the motivation of providing the image lists here is to guarantee that the evaluation sets are replicable for comparative experiments, which is usually lacked in many existing works such as [69, 112].

In run time, repeated 5-folder cross-validation is implemented to achieve an average recognition rate. In detail, each evaluation set is first equally separated into five groups. Every time, four groups are used to train the SVM classifier, and the remaining group is used as the testing set. The procedure runs five times to make sure that every image could appear exactly once in the testing data. Furthermore, above 5-folder cross-validation is repeated ten times to obtain a final average result, and each time the whole set will be reshuffled.

For person-independent evaluation, images of a same person must be placed in the same divided group. Conversely, person-dependent evaluation does not have that constraint, and all images distribute randomly. Regarding the two evaluation sets, clearly the CAS-PEAL-R1 set is free of person-dependent issue since all the 1040 images come from different persons. While for the FERET set, evaluation experiments are separated to person-independent and person-dependent cases.

6.2.3.2 Experimental Results

Generally, the two evaluation sets play different roles in the experiments. The goal of CAS-PEAR-R1 set is to analyze how image resolution and SVM kernel affect the recognition results. And performance comparisons with other methods were implemented on the FERET set because it is the most frequently used database in gender recognition research.

Table 6.9 presents the average recognition rates on CAS-PEAL-R1 data. The experiments investigated the three aforementioned image resolutions and two kernel types (linear and RBF). From the table, we can find two observations: (i) the performance of gender recognition with higher-resolution images seems to be slightly better than that with lower-resolution images; and (ii) SVM classifier with RBF kernel demonstrates trivial superiority over linear kernel.

Table 6.9: Classification accuracy on CAS-PEAL-R1 database with different image resolutions and SVM kernels (%)

Gender	128×128		72×72		36×36	
	Linear	RBF	Linear	RBF	Linear	RBF
Male	97.93	97.68	97.50	97.88	96.29	97.11
Female	95.66	96.22	93.55	94.34	92.47	93.08
Overall	96.96	97.06	95.81	96.37	94.65	95.38

Table 6.10: Comparisons of different methods for gender classification with FERET database (DT: decision trees)

Method	Image size	Person-independent	Accuracy(%)		
			Male	Female	Overall
LGIP+SVM Exp. #1	128×128	yes	96.75	94.58	96.29
LGIP+SVM Exp. #2	72×72	yes	95.84	93.78	95.55
LGIP+SVM Exp. #3	36×36	yes	91.46	96.20	95.74
LGIP+SVM Exp. #4	128×128	no	98.58	97.47	98.12
LGIP+SVM Exp. #5	72×72	no	97.96	97.19	97.64
LGIP+SVM Exp. #6	36×36	no	98.15	94.90	96.80
RBF network + DT [110]	72×64	no	-	-	96
LBP+ADaBoost [112]	144×120	no	-	-	95.75
LDP+SVM [69]	100×100	no	94.81	95.33	95.05
LBP/variance + SVM [114]	24×24	no	87.69	95.38	91.54

Table 6.10 records the experimental results of the proposed LGIP+SVM method as well as many other methods which were also tested with FERET database. Since this test set contains more than one images for a person, so the recognition should be divided to person-independent and person-dependent cases. As listed in Table 6.10, the six experiments Exp. #1 – #6 cover three image resolutions and person-independent/dependent conditions. Viewing from the last column, it is found that successful gender recognition rates of person-independent cases are approximately 2% lower than corresponding person-dependent cases. Besides, results in Table 6.10 support the observation on image resolution drawn from CAS-PEAL-R1 set, i.e., larger image resolution corresponds to slightly higher recognition rates. At last, LGIP+SVM method has demonstrated superior performance than other methods with similar setup as shown in last four rows.

6.3 Eye State Recognition

6.3.1 Introduction

Eye state recognition (ESR) is to determine the eye closure states (open or closed) by processing eye images. It should be mentioned beforehand that this thesis only discusses detecting eye states for independent frames. In fact, it closely relates to another technique, namely eye blink detection which emphasizes on detecting the sudden changes of closure states in a short video sequence. In principle, blink detection can be converted to detect eye state for individual frames.

Many attempts have been made to apply ESR to various practical matters. In [116], Grauman et al. developed a system which automatically detected eye blinks and measured their durations, providing an alternative way for disable people to access computer. For example, voluntary long blinks were detected to mimic mouse-clickers. More frequently, ESR gives a clue to monitor the fatigue condition of drivers in driver fatigue awareness systems [117–120]. Especially it has been proven that the fatigue condition is highly related to the PERCLOS (percentage of eye closure) [121]. Specially, the open/closed eye states are continuously monitored by processing input frames, and driver’s fatigue status can be inferred from closure frequency and duration. Liu and Ai also examined eye states to judge photo quality and correct those photos with closed eyes [122].

In literature, there exist many ESR methods. Broadly, they can be classified into three categories: template-based [117, 123, 124], feature-based [49, 125, 126] and appearance-based [118, 122, 127, 128] methods.

In template-based ESR, templates are designed for open or closed eyes, and similarity scores to input eye images are calculated to identify their states. In [123], an open eye template was used to get the correlation score with input eye image, and

a threshold was set to determine its state. In [117], open and closed eye templates were defined as two characteristic curves generated from plotting vertical integral projection. Then the characteristic curve of a new image was to match these two template curves. Tian et al. [124] developed a parametric dual-state eye model based system.

Feature-based methods find distinct features to distinguish open and closed eyes. Such features include dark pupil, white sclera, circular iris and so on. For example, Wang et al. [125] used three criteria to judge eye states: (i) circle similarity, which characterizes to what degree the low-intensity region is circle-shaped, (ii) upper eyelid bending direction, which is different for open (bending upward) and closed (bending downward) eyes, and (iii) width height ratio, which captures the shape difference. In [49], Hough transform algorithm was applied to detect iris circle to check the presence of open eyes. If non-detection happens, the eye is considered as in closed state. Tan and Zhang [126] tracked intensity-based iris and contour-based eyelids to recognize eye states. In general, methods of this type require less computational costs, but simultaneously they are often sensitive to appearance changes due to uncertain conditions like lighting etc.

The last type is appearance-based methods which apply on photometric appearance. Methods of this type need a big collection of training samples representing different lighting conditions and eye appearances to learn a dual-state classifier. Various classification techniques could be considered. Tian et al. [129] proposed an automatic system to detect eye-state action unit by combining Gabor coefficients and neural networks. In [122], AdaBoost Cascade open-eye detectors were trained at different scales to find open eyes in face region. A failure detection indicates the eyes are closed. In similar manner, Park et al. [118] adopted separate SVM-based detectors for open and closed eyes, to simultaneously detect eyes and recognize the

states. Xu et al. [127] learnt a cascaded dual-state classifier based on LBP features. Later in [128], Sun and Ma improved Xu's method, and applied SVM+LBP method to classify open/closed eyes.

In summary, feature-based methods are less time-consuming. But they are not able to deal with complex conditions with large variations due to lighting, orientation, occlusion, etc. Particular features to distinguish open and closed states may get obscure or even disappear. Template-based methods require to find a suitable model to generate the templates for eyes. The template could appear in the form of intensity pattern, a curve, or feature representation. Generally, this type has better stability compared to feature-based type. Finally, it is believed that appearance-based methods can best copy with complex appearance changes of eyes. And the prerequisite is that a large set of training images should be collected to capture different conditions as much as possible. Meanwhile, the training process is often slow, but commonly methods of this type exhibit better performance.

6.3.2 Extraction of Eye Descriptor

Before classifying eye states, feature vectors should be extracted from eye images. The proposed eye descriptor is based on histograms of LIIP features, and it was originally published in a conference paper [96].

Obviously, the size of eyes is much smaller than that of faces. Specifically, all eye images should be normalized to have a size of 40×20 in the experiments. Because of the small size, grid-based partitions for eye images become infeasible. Figure 6.3 illustrates the generation of LIIP-based eye descriptor. First, corresponding LIIP pattern image is gained by applying LIIP operator to every pixel in the normalized eye image. Afterwards, a sliding window is applied to scan the LIIP image at different locations and scales, resulting in a set of blocks. Within each block, the

LIIP histogram is extracted and normalized to have an unit length. Finally, all histogram bins from every block are combined together as the feature pool. Totally, the feature size is $37M$ for all M blocks. Similar feature extraction strategy is not suitable for face images due to the computational burden. For one thing, the number of blocks will drastically increase due to the larger size of face images. For the other thing, average block size of face images becomes much larger, resulting in more time cost to compute the histogram of each block. In contrast, block size of eye images is at most 40×20 .

Figure 6.3: This figure shows the process to extract eye descriptor based on LIIP features. The top left image represents the normalized eye image, and top right image represents LIIP pattern image, which is scaled up by six times for display. The bottom represents the feature vector by concatenating LIIP histograms from various blocks.

It is worth to explain more about generation of blocks. Corresponding parameters are provided in Table 6.11. Given a 40×20 image, the sliding window

Table 6.11: Parameter settings in eye state recognition

Image resolution	Initial window	window expanding step	Window sliding stride (current window = $w \times h$)
40×20	6×3	4×2	$\max(\frac{w}{4}, 4) \times \max(\frac{h}{4}, 2)$

initializes at a size of 6×3 , and gradually increases by expanding 4×2 pixels in width and height. When the current window has a size of $w \times h$, it slides over the whole image in a manner of left to right and top to bottom, at a stride of $(\max(w/4, 4), \max(h/4, 2))$ in width and height. At each location, a histogram is produced with 37 dimensions. Ultimately, this process generates 1722 blocks or equivalently $37 \times 1722 = 63714$ features.

In practice, computing all histogram bins is tremendous and unnecessary. In next subsection, we will see only a few discriminative features are really used. Therefore, it is valuable to propose a fast way to just calculate any specified histogram bin. Basically, the idea is to make use of *integral image* introduced in [7]. Figure 6.4 shows the procedure to quickly compute any individual feature. At first, the LIIP pattern image I_{liip} is obtain from eye image. Subsequently, this pattern image is split to 37 individual channels $B_0, \dots, B_{36}$. In detail, those pixels of B_i will be set to one if the corresponding values in I_{liip} are equal to i , and the rest pixels of B_i are set to zero. As shown in Figure 6.4, these 37 binary images can be obtained using indicator function $1\{*\}$, which takes value one if the statement in the bracket is true, and zero otherwise. Based on these binary images, their integral images are computed and denoted by $I_0, \dots, I_{36}$.

Assume we want to get the j th histogram bin from any rectangle ABCD, only the four vertex pixels in image I_j should be used as expressed by:

$$\text{LIIPH}_{ABCD}(j) = I_j(A) + I_j(D) - I_j(B) - I_j(C) \quad (6.4)$$

Figure 6.4: Fast calculation of LIIP-based histogram bins from block ABCD. I_{org} and I_{liip} separately denote the normalized eye image and corresponding LIIP pattern image.

where $LIIPH_{ABCD}(j)$ denotes the j th bin from block ABCD, and $I_j(A)$, $I_j(B)$, $I_j(C)$ and $I_j(D)$ are the vertices' values in image I_j . Furthermore, the individual bin could be normalized via dividing by the area of ABCD. In this way, it is easy to compute any individual feature without computing the full feature pool, which helps to save much computational burdens.

6.3.3 Open/Closed State Classifier

As mentioned above, the feature pool of eye descriptor has a huge size, and only a small portion of features will be used in ESR. Thus, effective dimensionality reduction and classifier are very important. In this work, AdaBoost algorithm [21] is adopted to select the most discriminant ones as well as perform open/closed state classification.

In theory, the principle of AdaBoost algorithm is to unify a set of weak classifiers to form a strong classifier. Usually, these weak classifiers are only required to have better-than-chance accuracies, but they must have promising time efficiency for real time running. Commonly, each feature from the huge pool could be considered as a weak classifier by simply setting an optimal threshold. There are four variants of AdaBoost algorithms [80]: Discrete AdaBoost, Real AdaBoost, LogitBoost and Gentle AdaBoost. All of them are very similar in the overall structure, and only the two-class Discrete AdaBoost algorithm is explored here.

Formally, Figure 6.5 presents the training procedure of Discrete AdaBoost algorithm. The binary classifier is learnt from a set of labeled samples. At first, identical weights are assigned to every sample as in Step 2. In each iteration, every variable in the feature vector constitutes a weak classifier by optimizing a threshold, and the best weak classifier is chosen from the one with least weighted classification error (Step 3a, 3b). Subsequently, the weights of those misclassified samples will be artificially increased (Step 3c), and all samples' weights are normalized to have a sum of one. Finally, T weak classifiers would be selected after T iterations, and a final strong classifier is constructed as a weighted sum of the responses from these T weak classifiers (Step 4). The AdaBoost algorithm in the experiments adopts the realization of OpenCV library. The only difference comes from the weak classifiers, which use standalone binary decision trees [130] with small depth (less than 4).

-
1. Given n samples (x_i, y_i) , with $x_i \in R^m, y_i = \pm 1, i=1 \dots n$.
 2. Every sample x_i starts with the same weight: $w_{1,i} = 1/n$.
 3. Iterate $t = 1 \dots T$:
 - a. For each variable j , train a weak classifier $h_j()$. Evaluate the weighted error of the classifier $\epsilon = \sum_i w_{t,i} b_i$. $b_i = 0$ if x_i is correctly classified, $b_i = 1$ otherwise.
 - b. Select the weak classifier with lowest weighted classification error ϵ_t .
 - c. Update weights: $w_{t+1,i} = w_{t,i} \beta_t^{1-b_i}$, with $\beta_t = \epsilon_t / (1 - \epsilon_t)$.
 4. The objective function of the strong classifier is:
$$F(x) = \text{sign}(\sum_{t=1}^T \alpha_t h_t(x)), \text{ with } \alpha_t = \lg(1/\beta_t).$$
-

Figure 6.5: Architecture of Discrete AdaBoost algorithm

6.3.4 Experiments

6.3.4.1 Static Test

First, tests on static images are implemented to check the performance of ESR. Currently, it is lack of publicly available open/closed eye databases. The common handling way is to capture homemade open and closed eye images. In this experiment, eye images were collected from many public face databases. Obviously, open eyes could be easily obtained from any face database, since normally the subjects were captured with eyes open. Especially, closed eye images occasionally appear in the database, so I collected a combo set from several databases. In total, the combo set contains 7458 samples including 4979 open eyes and 2479 closed eyes. These images, especially the closed eyes, were gathered from four databases that have a few face images with closed eyes. These databases are CAS-PEAL-R1 face database [115], AR face database [131], BioID face database [82], and RPI ISL Eye Training database [132]. On the whole, these samples represent various eye appear-

ances in terms of different scales, skin colors, orientations, lighting conditions and so on. All eye images were cropped and normalized to a standard size of 40×20 , without other preprocessing. Many samples can be seen in Figure 6.6.

Figure 6.6: Illustration of some positive and negative samples used for training

The combo data was further divided into training and testing sets. In the experiment, 2000 positive (open eyes) and 1000 negative (closed eyes) samples were arbitrarily picked to train the open/closed state classifier. Eventually, the strong classifier was made up of 35 standalone small decision trees which were constructed by selecting a few most discriminative features from the feature pool. Naturally, these features are expected to split positive and negative samples well. For instance, it is found the first decision tree (weak classifier) learnt in iteration 1 has used only 11 features, and could surprisingly achieve a recognition rate of 99.13% for positive samples and 98.26% for negative samples. This reflects the high discrimination ability of LIIP-based features. To illustrate, Figure 6.7 shows the

corresponding blocks where first six features come from. Mainly they distribute around eye pupils and corners where the two eye states differ greatly.

Figure 6.7: Illustration of corresponding regions of six most discriminative features. Each region is marked by two samples: one positive and one negative.

Table 6.12 gives the recognition results. Note the table includes results for training data and testing data. Unexpectedly, the classifier can correctly recognize all the training samples. More important, recognition rates for positive and negative samples in testing data are 99.87% and 99.86%. Clearly, the performance is very encouraging. Some recognition samples are displayed in Figure 6.8. We can see the classifier is able to handle variations in eye appearances. Moreover, it is observed that these misclassified samples mostly stay in a middle or half-open condition, which tends to happen during transitions of open and closed states.

To measure the time efficiency, the proposed method was programmed using Visual Studio software in a laptop with 2.53GHz CPU and 3G RAM. In the test, it just took 4.15 milliseconds to extract all 63,714 features. Only several hundreds of these features will be utilized in the end. As discussed, any particular feature can be independently computed using the concept of integral image. Thus, consumed

		Recognized state		Recognition rate
		Open	Closed	
Training	Open	2000	0	100%
	Closed	0	1000	100%
Testing	Open	2975	4	99.87%
	Closed	2	1477	99.86%

Table 6.12: Eye state recognition results on static images

Figure 6.8: Some examples of eye state recognition results. Top left - some true positives, top right - some true positives, bottom left - all false negatives, bottom right - all false positives.

time for feature extraction can be further reduced. Moreover, predicting the state of an image costs only $16 \mu s$, so the whole process of ESR requires no more than 5 milliseconds. Such a high speed makes it possible for real time running.

6.3.4.2 Video Test

This part examines the performance of the proposed eye state recognizer with live video. There exist very few public videos that are suitable for ESR as the eye blinks rarely appear. Fortunately, the Talking Face Video (TFV) provides a test-bed for ESR evaluation. The video can be freely downloaded at http://www-prima.inrialpes.fr/FGnet/data/01-TalkingFace/talking_face.html. The TFV consists of 5000 frames taken from a video of a person engaged in conversation, as part of an experiment to model the behavior of face in natural conversation. More details about the data could check with the provided website.

An automatic system is created to recognize eye states. The system consists of face detection, eye detection and open/closed state recognition. Since only frontal faces are included in TFV data, the first two modules mainly utilize the Viola Jones Cascade method as well as geometric location of eyes. And open/closed states of eye images are classified using the classifier trained during static tests. Consequently, the successful rates for open and closed eyes are 97.68% (9165/9383) and 99.83% (582/583) respectively. Again it is found these incorrectly recognized open eyes occur more frequently during transitions between the two states. Moreover, the closed eyes are hardly classified as open eyes. In runtime, the speed of the system is about 28 frames per second. More detailed results on the TFV video could check my Youtube channel at <https://www.youtube.com/channel/UCwwzA0VY-vof7jSQFeJQM7A>.

6.4 Summary

This chapter has introduced three applications that make use of the proposed feature extraction methods. Separately, local features appear in different forms

to get adapted to facial expression recognition, gender recognition and eye state recognition.

Firstly, facial expression recognition has been performed based on MC-LIIP features and multi-class SVM classifier. To represent face images, MC-LIIP images at various scales are partitioned to grid-based regions, where histograms of local patterns are extracted and unified to a single vector. The experiments first investigate the influence of grid partitions and feature scales to the face descriptor. Person-independent evaluation show the proposed MC-LIIP+SVM method has achieved a successful rate of 93.2% and 95.3% for 7-class and 6-class facial expression recognition with CK+ database. Furthermore, experimental results on both CK+ and KDEF databases support the statement that the new method outperforms many other algorithms.

Secondly, application on gender recognition has examined the performance of LGIP features for face representation. Differently, LGIP histograms are extracted from overlapping regions at a single scale and concatenated together. Binary SVM is used to identify face genders, and the results on FERET demonstrates that LGIP+SVM method is superior to some other methods.

Lastly, eye state recognition exhibits the discrimination ability of LIIP features in distinguishing open and closed eyes. Unlike face images, eye images have a small size, so a sliding window is used to scan them at different locations and scales to generate hundreds of regions, where LIIP histograms are extracted and then combined together to form a huge feature pool. For classification, Discrete AdaBoost iteratively selects dozens of small decision trees as weak classifiers, and these weak classifiers are combined together to form a strong classifier. In the end, only a small portion of features will be used in the classifier. Under this consideration, a fast way to compute any particular feature has been proposed.

Evaluation Experiments on static images and real-world video show the proposed method possesses not only very high recognition rate (over 97%) at very high speed (28 fps).

Chapter 7

Conclusion and Recommendations

7.1 Conclusion

The topic of this thesis is on developing novel algorithms for facial image analysis. Concretely, it covers two kinds of techniques (correlation filters and feature extraction) and five relevant applications (facial landmark localization, face detection, facial expression recognition, gender recognition and eye state recognition).

In the research a new way to construct correlation filters named Generalized Unconstrained Filters (GUF) is firstly introduced in Chapter 3. Basically, the objective of GUF is to minimize a weighted sum of three criteria to the filter: mean squared error of correlation output to a desired output, variance of correlation output, and filter energy. Compared to previous filters, the class of GUF filters has a number of advantages. Many existing methods such as Average Filter (AVG), Unconstrained Minimum Average Correlation Energy (UMACE), Maximum Average Correlation Height (MACH) and Unconstrained Optimal Tradeoff Synthetic Discriminant Functions (UOTSDF) require cropped and target-in-center images during filter training. Through Fourier transform and convolution, GUF-based method could remove the above limitations on training images. Another two methods, the Average of Synthetic Exact Filters (ASEF) and Minimum Sum Squared Output Error (MOSSE), are able to work on non-centered training images,

but every image must be assigned with a specially designed correlation-output image. Last but not least, these aforementioned filters have been proved to be special conditions of GUF filters in theory.

This research has investigated two face-related applications of GUF filters: facial landmark localization, and face detection and tracking (Chapter 4). Separately, these application domains have different challenges, and they can well examine different aspects of GUF filters.

In facial landmark localization, thousands of normalized face images are first used to train GUF filters offline for specific facial points. This mainly looks at the static aspect of GUF filters with large-scale training data. As discussed in Subsection 4.1.3, the resulting filters in spatial domain could characterize the appearance of center templates of the targets. Changing the weights of aforesaid three criteria would result in various filters showing different appearances. In addition, an error correction and filter updating module has been proposed to further correct these inaccurately located samples. In the experiments, detection of two facial points including right eye center and nose was tested on FERET database. The outcomes demonstrated that GUF filters could achieve higher accuracy than prior filters, and they also outperformed the Viola Jones Cascade detector. Finally, GUF filters have exhibited amazingly high speed in both training and testing. At a glance, the running speed is around 658 frames per second with image resolution of 128×128 .

Contrastively, face detection and tracking uses a single frame to initialize the GUF filter, which is subsequently online updated with newly coming frames. Mathematically, GUF method has been justified to be very suitable for online tracking. GUF tracker is capable of dealing with changing appearances due to lighting, pose, etc, but it may drift to background sometimes and has no self-recovery function. On the other hand, Viola Jones face detector performs very successfully in natural frontal faces, but it lacks the ability to handle complex conditions. This work

proposed to fuse the Cascade face detector and GUF tracker to complement their drawbacks. Tests on five challenging videos showed the proposal could obtain very promising performance.

In second part, this research has presented many local features in Chapter 5. Originally, Local Intensity Increasing Patterns (LIIP) was proposed as a local texture descriptor. Simply, LIIP uses 37 pattern labels to represent different kinds of intensity structures in the neighborhood of each pixel. These patterns are generated via pixel-wise intensity comparisons and feature mapping with a look up table. Intrinsically, LIIP describes toward which direction and to what degree the intensity of a local patch increases. LIIP has inherited a number of strengths from Local Binary Patterns (LBP), uniform patterns, and gradient feature. Later, the Local Gradient Increasing Patterns (LGIP) was developed as a variant of LIIP feature. LGIP maintains most features from LIIP, and the difference comes from the fact that pattern generation relies on eight directional gradient responses rather than pixel-wise comparisons. Furthermore, Multiscale Cell LIIP (MC-LIIP) extends LIIP feature to multiple scale levels, in which both texture micro-structures and macro-structures are represented.

Three face-related applications were adopted to evaluate these new local features: facial expression recognition, gender recognition, and eye state recognition (Chapter 6). In multi-class facial expression classification, MC-LIIP pattern images were partitioned into non-overlapping regions, where regional histograms were extracted and concatenated to represent face images. By combining with SVM classifier, experimental results on CK+ database have shown that successful rates of 95.3% and 93.2% for 6-class and 7-class person-independent expression recognition could be obtained. In addition, MC-LIIP based method had higher or comparable accuracy than other state-of-the-art algorithms. Performance of LGIP feature

was checked in the binary classification task: gender recognition. Alternatively, regional histograms of LGIP patterns were extracted from overlapping regions, and combined to form the face descriptor. Based on it, binary SVM classifier was used, and the new method exceeded the accuracy of many other algorithms with FERET database. Finally, application on eye state recognition has demonstrated the capability of LIIP feature in representing small images. Instead of grid-based partitions, a series of regions are generated at different locations and scales all over the eye images. Still regional histograms are adopted. This work has introduced a fast way to calculate any individual histogram bin. By incorporating with AdaBoost algorithm, the eye descriptor has demonstrated very high accuracy (over 97%) as well as fast speed (28 fps) in static images and video tests.

7.2 Recommendation for Future Research

Although the presented works have gained satisfactory performance, they still have the potentials for further improvements. Several possible directions for future investigation are listed as follows.

- The proposed GUF filters belong to linear filters. For facial landmark localization, even good results have been obtained in FERET database, the performance of GUF declines when face images contain large rotations (in-plane and out-of-plane) and scaling. After all, using a single linear filter cannot deal with diversified conditions. One feasible way to improve the generality of GUF is to construct many filters especially for different rotations and scaling. For example, K-Means clustering may be applied to the exact filters of all samples. Besides, non-linear filters may provide another line of research.

- In face detection and tracking, the GUF filter keeps changing with the input frame. Occasionally, the tracking window may be misled to the background or somewhat off the center. This thesis has proposed to get recovery with the help of Viola Jones face detector. The drawback is the detector should be trained beforehand with a large training data. This is infeasible for common object detection. Therefore, it may be a good option to train a global filter-based detector along with current GUF tracker. The global detector should be equally learnt from all previous frames, so it depends slightly on the newly incoming frame. It aims to provide an alternative solution for tracking recovery. If this proposal is feasible, the detection and tracking scheme will be applicable to common object detection without prior offline training.
- Finally, it would be a good blueprint to possibly combine these five applications together to construct a fully automatic facial analysis system.

Author's Publications

1. **L.B. Zhou**, and H. Wang, "Iterative Adaptive Synthetic Correlation Output Filters," *Electronics Letters*, vol. 49, no. 14, 2013.
2. **L.B. Zhou**, and H. Wang, "Feature Extraction and Filter Design for Eye Pattern Analysis," *IFAC International Conference on Intelligent Control and Automatic Science (ICONS)*, Chengdu, China, 2013.
3. **L.B. Zhou**, and H. Wang, "Facial landmark localization via boosted and adaptive filters," *IEEE International Conference on Image Processing (ICIP)*, Melbourne, Australia, 2013.
4. **L.B. Zhou**, and H. Wang, "Person-independent facial expression analysis by fusing multiscale cell features," *Optical Engineering*, vol. 52, no. 3, 037201, 2013.
5. **L.B. Zhou**, and H. Wang, "Local gradient increasing pattern for facial expression recognition," in *IEEE International Conference on Image Processing (ICIP)*, Orlando, 2012, pp. 2601-2604.
6. **L.B. Zhou**, and H. Wang, "Local gradient increasing pattern (LGIP) for facial representation and gender recognition," in *International Conference on Image Analysis and Recognition (ICIAR)*, Aveiro, Portugal, 2012, pp. 46-53.

7. **L.B. Zhou**, and H. Wang, “Fast and effective system to locate eye and recognize open/closed state,” *Far East Journal of Electronics and Communications*, vol. 7, no. 2, pp. 91-114, 2011.
8. **L.B. Zhou**, and H. Wang, “Local features by intensity increasing tendency for face recognition,” *International Journal of Computational Vision and Robotics*, vol. 2, no. 2, pp. 127-140, 2011.
9. **L.B. Zhou**, and H. Wang, “Open/closed eye recognition by local binary increasing patterns,” in *IEEE International Conference on Robotics, Automation and Mechatronics (RAM)*, Qingdao, China, 2011, pp. 7-11.
10. H. Wang, **L.B. Zhou**, and Y. Ying, “A novel approach for real time eye state detection in fatigue awareness system,” in *IEEE International Conference on Robotics, Automation and Mechatronics (RAM)*, Singapore, 2010, pp. 528-532.

Bibliography

- [1] S. Z. Li and A. K. Jain, *Handbook of face recognition*. Springer, 2011.
- [2] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification and Scene Analysis*. Wiley-Interscience, 1995.
- [3] C. F. Hester and D. Casasent, “Multivariant technique for multiclass pattern recognition,” *Applied Optics*, vol. 19, no. 11, pp. 1758–1761, 1980.
- [4] A. Mahalanobis, B. V. K. Vijaya Kumar, and D. Casasent, “Minimum average correlation energy filters,” *Applied Optics*, vol. 26, no. 17, pp. 3633–3640, 1987.
- [5] A. Mahalanobis, B. V. K. Vijaya Kumar, S. Song, S. Sims, and J. Epperson, “Unconstrained correlation filters,” *Applied Optics*, vol. 33, no. 17, pp. 3751–3759, 1994.
- [6] D. S. Bolme, B. A. Draper, and J. R. Beveridge, “Average of synthetic exact filters,” in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2105–2112, 2009.
- [7] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 511–518, 2001.

- [8] L. Wiskott, J. Fellous, N. Kuiger, and C. von der Malsburg, “Face recognition by elastic bunch graph matching,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 775–779, 1997.
- [9] T. Ojala, M. Pietikäinen, and T. Maenpaa, “Multiresolution gray-scale and rotation invariant texture classification with local binary patterns,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 971–987, 2002.
- [10] T. Ahonen, A. Hadid, and M. Pietikäinen, “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, 2006.
- [11] G. Zhao and M. Pietikäinen, “Dynamic texture recognition using local binary patterns with an application to facial expressions,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 915–928, 2007.
- [12] C. Shan, S. Gong, and P. W. McOwan, “Facial expression recognition based on local binary patterns: a comprehensive study,” *Image and Vision Computing*, vol. 27, no. 6, pp. 803–816, 2009.
- [13] G. Heusch, Y. Rodriguez, and S. Marcel, “Local binary patterns as an image preprocessing for face authentication,” in *International Conference on Automatic Face and Gesture Recognition*, pp. 9–14, 2006.
- [14] H. C. Lian and B. L. Lu, “Multi-view gender classification using local binary patterns and support vector machines,” *Advances in Neural Networks-ISNN*, pp. 202–209, 2006.
- [15] T. Jabid, M. H. Kabir, and O. Chae, “Robust facial expression recognition based on local directional pattern,” *ETRI Journal*, vol. 32, pp. 784–794, 2010.

- [16] M. Pietikäinen, A. Hadid, G. Zhao, and T. Ahonen, *Computer vision using local binary patterns*, vol. 40. Springer, 2011.
- [17] B. V. K. Vijaya Kumar, D. W. Carlson, and A. Mahalanobis, “Optimal trade-off synthetic discriminant function filters for arbitrary devices,” *Optics Letters*, vol. 19, no. 19, pp. 1556–1558, 1994.
- [18] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, “Visual object tracking using adaptive correlation filters,” in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2544–2550, 2010.
- [19] M. A. Hearst, S. Dumais, E. Osman, J. Platt, and B. Scholkopf, “Support vector machines,” *Intelligent Systems and their Applications*, vol. 13, no. 4, pp. 18–28, 1998.
- [20] C. J. Burges, “A tutorial on support vector machines for pattern recognition,” *Data mining and knowledge discovery*, vol. 2, no. 2, pp. 121–167, 1998.
- [21] Y. Freund and R. E. Schapire, “A decision-theoretic generalization of on-line learning and an application to boosting,” *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997.
- [22] P. J. Phillips, H. Moon, S. A. Rizvi, and P. J. Rauss, “The FERET evaluation methodology for face recognition algorithms,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 10, pp. 1090–1104, 2000.
- [23] P. Lucey, J. F. Cohn, T. Kanade, J. Saragih, Z. Ambadar, and I. Matthews, “The extended cohn-kanade dataset (CK+): a complete dataset for action unit and emotion-specified expression,” in *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 94–101, 2010.

- [24] D. Lundqvist and J. Litton, “The averaged karolinska directed emotional faces,” *Department of Clinical Neuroscience, Psychology Section, Karolinska Institutet. Stockholm, Sweden*, 1998.
- [25] D. Bolme, “Theory and applications of optimized correlation output filters,” *Colorado State University*, 2011.
- [26] R. A. Kerekes and B. V. K. Vijaya Kumar, “Selecting a composite correlation filter design: a survey and comparative study,” *Optical Engineering*, vol. 47, no. 6, pp. 067202–067202, 2008.
- [27] B. K. Horn and B. G. Schunck, “Determining optical flow,” *Artificial intelligence*, vol. 17, no. 1, pp. 185–203, 1981.
- [28] B. V. K. Vijaya Kumar, “Tutorial survey of composite filter designs for optical correlators,” *Applied Optics*, vol. 31, no. 23, pp. 4773–4801, 1992.
- [29] V. Diaz-Ramirez, L. Trujillo, and S. Pinto-Fernandez, *Advances in object recognition systems: Advances in adaptive composite filters for object recognition*. InTech, 2012.
- [30] D. Casasent, “Unified synthetic discriminant function computational formulation,” *Applied Optics*, vol. 23, no. 10, pp. 1620–1627, 1984.
- [31] D. Casasent and W. T. Chang, “Correlation synthetic discriminant functions,” *Applied Optics*, vol. 25, no. 14, pp. 2343–2350, 1986.
- [32] B. V. K. Vijaya Kumar, “Minimum-variance synthetic discriminant functions,” *JOSA A*, vol. 3, no. 10, pp. 1579–1584, 1986.

- [33] D. Casasent, G. Ravichandran, and S. Bollapragada, “Gaussian–minimum average correlation energy filters,” *Applied optics*, vol. 30, no. 35, pp. 5176–5181, 1991.
- [34] P. Réfrégier, “Optimal trade-off filters for noise robustness, sharpness of the correlation peak, and horner efficiency,” *Optics Letters*, vol. 16, no. 11, pp. 829–831, 1991.
- [35] V. Štruc, J. Ž. Gros, and N. Pavešić, “Principal directions of synthetic exact filters for robust real-time eye localization,” in *Biometrics and ID Management*, pp. 180–192, 2011.
- [36] A. Mahalanobis and B. V. K. Vijaya Kumar, “Polynomial filters for higher order correlation and multi-input information fusion,” in *Optoelectronic Information Processing Workshop*, pp. 221–232, 1997.
- [37] B. V. K. Vijaya Kumar, M. Savvides, and C. Xie, “Correlation pattern recognition for face recognition,” *Proceedings of the IEEE*, vol. 94, no. 11, pp. 1963–1976, 2006.
- [38] M. Rodriguez, J. Ahmed, and M. Shah, “Action MACH a spatio-temporal maximum average correlation height filter for action recognition,” in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, 2008.
- [39] H. K. Galoogahi, T. Sim, and S. Lucey, “Multi-channel correlation filters,” in *International Conference on Computer Vision*, 2013.
- [40] V. N. Boddeti, T. Kanade, and B. V. K. Vijaya Kumar, “Correlation filters for object alignment,” in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2291–2298, 2013.

- [41] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 886–893, 2005.
- [42] G. Fanelli, J. Gall, and L. Van Gool, “Real time head pose estimation with random regression forests,” in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 617–624, 2011.
- [43] T. Tuytelaars and K. Mikolajczyk, “Local invariant feature detectors: a survey,” *Foundations and Trends in Computer Graphics and Vision*, vol. 3, no. 3, pp. 177–280, 2008.
- [44] K. Mikolajczyk and C. Schmid, “A performance evaluation of local descriptors,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 10, pp. 1615–1630, 2005.
- [45] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.
- [46] H. Bay, T. Tuytelaars, and L. Van Gool, “Surf: Speeded up robust features,” in *European Conference on Computer Vision*, pp. 404–417, 2006.
- [47] V. Lepetit and P. Fua, “Keypoint recognition using randomized trees,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 9, pp. 1465–1479, 2006.
- [48] V. Vezhnevets, V. Sazonov, and A. Andreeva, “A survey on pixel-based skin color detection techniques,” in *Proc. Graphicon*, vol. 3, pp. 85–92, 2003.
- [49] Q. Wang and J. Yang, “Eye location and eye state detection in facial images with unconstrained background,” *Journal of Information and Computing Science*, vol. 1, no. 5, pp. 284–289, 2006.

- [50] Y. Ke and R. Sukthankar, “PCA-SIFT: A more distinctive representation for local image descriptors,” in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 506–513, 2004.
- [51] A. Alahi, R. Ortiz, and P. Vandergheynst, “Freak: Fast retina keypoint,” in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 510–517, 2012.
- [52] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “ORB: an efficient alternative to SIFT or SURF,” in *IEEE International Conference on Computer Vision*, pp. 2564–2571, 2011.
- [53] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray, “Visual categorization with bags of keypoints,” in *ECCV Workshop on Statistical Learning in Computer Vision*, vol. 1, p. 22, 2004.
- [54] M. Turk and A. Pentland, “Eigenfaces for recognition,” *Journal of Cognitive Neuroscience*, vol. 3, no. 1, pp. 71–86, 1991.
- [55] K. Etemad and R. Chellappa, “Discriminant analysis for recognition of human face images,” *JOSA A*, vol. 14, no. 8, pp. 1724–1733, 1997.
- [56] G. E. Hinton and R. R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” *Science*, vol. 313, no. 5786, pp. 504–507, 2006.
- [57] G. E. Hinton, S. Osindero, and Y. W. Teh, “A fast learning algorithm for deep belief nets,” *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.
- [58] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

- [59] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, “Greedy layer-wise training of deep networks,” *Advances in Neural Information Processing Systems*, vol. 19, p. 153, 2007.
- [60] R. Gottumukkal and V. K. Asari, “An improved face recognition technique based on modular PCA approach,” *Pattern Recognition Letters*, vol. 25, no. 4, pp. 429–436, 2004.
- [61] G. H. Granlund, “In search of a general picture processing operator,” *Computer Graphics and Image Processing*, vol. 8, no. 2, pp. 155–173, 1978.
- [62] M. Lades, J. C. Vorbruggen, J. Buhmann, J. Lange, C. von der Malsburg, R. P. Wurtz, and W. Konen, “Distortion invariant object recognition in the dynamic link architecture,” *IEEE Transactions on Computers*, vol. 42, no. 3, pp. 300–311, 1993.
- [63] B. A. Olshausen and D. J. Field, “Emergence of simple-cell receptive field properties by learning a sparse code for natural images,” *Nature*, vol. 381, no. 6583, pp. 607–609, 1996.
- [64] B. Schiele and J. L. Crowley, “Recognition without correspondence using multidimensional receptive field histograms,” *International Journal of Computer Vision*, vol. 36, no. 1, pp. 31–50, 2000.
- [65] J. G. Daugman, “Complete discrete 2-D gabor transforms by neural networks for image analysis and compression,” *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 36, no. 7, pp. 1169–1179, 1988.
- [66] M. S. Bartlett, G. Littlewort, M. Frank, C. Lainscsek, I. Fasel, and J. Movellan, “Recognizing facial expression: machine learning and application to

- spontaneous behavior,” in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 568–573, 2005.
- [67] Y. L. Tian, “Evaluation of face resolution for expression analysis,” in *IEEE Conference on Computer Vision and Pattern Recognition Workshop, CVPRW’04*, pp. 82–88, 2004.
- [68] T. Ojala, M. Pietikäinen, and D. Harwood, “A comparative study of texture measures with classification based on featured distributions,” *Pattern recognition*, vol. 29, no. 1, pp. 51–59, 1996.
- [69] T. Jabid, M. Hasanul Kabir, and O. Chae, “Gender classification using local directional pattern (LDP),” in *International Conference on Pattern Recognition (ICPR)*, pp. 2162–2165, 2010.
- [70] B. Jähne, *Digital Image Processing*. Springer, Berlin, 2005.
- [71] D. Messerschmitt, “Stationary points of a real-valued function of a complex variable,” tech. rep., EECS, UC Berkeley, 2006.
- [72] G. Feng and P. Yuen, “Variance projection function and its application to eye detection for human face recognition,” *Pattern Recognition Letters*, vol. 19, no. 9, pp. 899–906, 1998.
- [73] Z. Zhou and X. Geng, “Projection functions for eye detection,” *Pattern Recognition*, vol. 37, no. 5, pp. 1049–1056, 2004.
- [74] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham, “Active shape models-their training and application,” *Computer Vision and Image Understanding*, vol. 61, no. 1, pp. 38–59, 1995.

- [75] T. F. Cootes, G. J. Edwards, and C. J. Taylor, “Active appearance models,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 6, pp. 681–685, 2001.
- [76] K. Fukunaga, *Introduction to statistical pattern recognition*. Academic press, 1990.
- [77] H. A. Rowley, S. Baluja, and T. Kanade, “Neural network-based face detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 1, pp. 23–38, 1998.
- [78] J. R. Quinlan, “Induction of decision trees,” *Machine Learning*, vol. 1, no. 1, pp. 81–106, 1986.
- [79] P. Viola and M. J. Jones, “Robust real-time face detection,” *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004.
- [80] J. Friedman, T. Hastie, and R. Tibshirani, “Additive logistic regression: a statistical view of boosting,” *The Annals of Statistics*, vol. 28, no. 2, pp. 337–407, 2000.
- [81] R. Brunelli and T. Poggio, “Template matching: Matched spatial filters and beyond,” *Pattern Recognition*, vol. 30, no. 5, pp. 751–768, 1997.
- [82] O. Jesorsky, K. J. Kirchberg, and R. W. Frischholz, “Robust face detection using the hausdorff distance,” in *Audio-and Video-based Biometric Person Authentication*, pp. 90–95, Springer, 2001.
- [83] M. Castrillón, O. Déniz, C. Guerra, and M. Hernández, “Real-time detection of multiple faces at different resolutions in video streams,” *Journal of Visual Communication and Image Representation*, vol. 18, no. 2, pp. 130–140, 2007.

- [84] K. C. Yow and R. Cipolla, "Feature-based human face detection," *Image and Vision Computing*, vol. 15, no. 9, pp. 713–735, 1997.
- [85] S. McKenna, S. Gong, and Y. Raja, "Modelling facial colour and identity with gaussian mixtures," *Pattern Recognition*, vol. 31, no. 12, pp. 1883–1892, 1998.
- [86] L. Huang, A. Shimizu, and H. Kobatake, "Robust face detection using Gabor filter features," *Pattern Recognition Letters*, vol. 26, no. 11, pp. 1641–1649, 2005.
- [87] C. Chiang and C. Huang, "A robust method for detecting arbitrarily tilted human faces in color images," *Pattern Recognition Letters*, vol. 26, no. 16, pp. 2518–2536, 2005.
- [88] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *International Journal of Computer Vision*, vol. 1, no. 4, pp. 321–331, 1988.
- [89] A. Yuille, P. Hallinan, and D. Cohen, "Feature extraction from faces using deformable templates," *International Journal of Computer Vision*, vol. 8, no. 2, pp. 99–111, 1992.
- [90] M. Bartlett, J. Movellan, and T. Sejnowski, "Face recognition by independent component analysis," *IEEE Transactions on Neural Networks*, vol. 13, no. 6, pp. 1450–1464, 2002.
- [91] K. Sung and T. Poggio, "Example-based learning for view-based human face detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 1, pp. 39–51, 1998.

- [92] E. Osuna, R. Freund, and F. Girosit, “Training support vector machines: an application to face detection,” in *IEEE International Conference on Computer Vision and Pattern Recognition*, pp. 130–136, 1997.
- [93] F. Samaria and S. Young, “HMM-based architecture for face identification,” *Image and Vision Computing*, vol. 12, pp. 537–543, Oct 1994.
- [94] H. Schneiderman and T. Kanade, “Probabilistic modeling of local appearance and spatial relationships for object recognition,” in *IEEE International Conference on Computer Vision and Pattern Recognition*, pp. 45–51, 1998.
- [95] P. Belhumeur, J. Hespanha, and D. Kriegman, “Eigenfaces vs. fisherfaces: recognition using class specific linear projection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 711–720, 1997.
- [96] L. B. Zhou and H. Wang, “Open/closed eye recognition by local binary increasing intensity patterns,” in *IEEE Conference on Robotics, Automation and Mechatronics*, pp. 7–11, 2011.
- [97] L. B. Zhou and H. Wang, “Local gradient increasing pattern (lgip) for facial representation and gender recognition,” in *International Conference on Image Analysis and Recognition*, pp. 46–53, Springer, 2012.
- [98] L. B. Zhou and H. Wang, “Person-independent facial expression analysis by fusing multiscale cell features,” *Optical Engineering*, vol. 52, no. 3, pp. 037201–037201, 2013.
- [99] P. Ekman and W. Friesen, “Facial action coding system: a technique for the measurement of facial movement,” *Consulting Psychologists Press, Palo Alto, CA*, vol. 12, pp. 271–302, 1978.

- [100] S. Chew, P. Lucey, S. Lucey, J. Saragih, J. Cohn, S. Sridharan, *et al.*, “Person-independent facial expression detection using constrained local models,” in *International Conference on Automatic Face and Gesture Recognition*, pp. 915–920, 2011.
- [101] M. Yeasin, B. Bullot, and R. Sharma, “From facial expression to level of interest: a spatio-temporal approach,” in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 922–927, 2004.
- [102] P. S. Aleksic and A. K. Katsaggelos, “Automatic facial expression recognition using facial animation parameters and multistream hmms,” *IEEE Transactions on Information Forensics and Security*, vol. 1, no. 1, pp. 3–11, 2006.
- [103] S. Shojaeilangari, Y. W. Yun, and T. E. Khwang, “Person independent facial expression analysis using gabor features and genetic algorithm,” in *International Conference on Information, Communications and Signal Processing*, pp. 1–5, 2011.
- [104] C. Hsu, C. Chang, and C. Lin, “A practical guide to support vector classification,” tech. rep., 2003. available at <http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf>.
- [105] T. Kanade, J. Cohn, and Y. Tian, “Comprehensive database for facial expression analysis,” in *International Conference on Automatic Face and Gesture Recognition*, pp. 46–53, 2000.
- [106] M. Pantic and L. Rothkrantz, “Automatic analysis of facial expressions: The state of the art,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 12, pp. 1424–1445, 2000.

- [107] M. Lyons, S. Akamatsu, M. Kamachi, and J. Gyoba, “Coding facial expressions with gabor wavelets,” in *IEEE International Conference on Automatic Face and Gesture Recognition*, pp. 200–205, 1998.
- [108] B. Moghaddam and M.-H. Yang, “Learning gender with support faces,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 707–711, 2002.
- [109] K. Balci and V. Atalay, “PCA for gender estimation: which eigenvectors contribute,” in *International Conference on Pattern Recognition*, vol. 3, pp. 363–366, 2002.
- [110] S. Gutta, H. Wechsler, and P. J. Phillips, “Gender and ethnic classification of face images,” in *International Conference on Automatic Face and Gesture Recognition*, pp. 194–199, 1998.
- [111] E. Makinen and R. Raisamo, “Evaluation of gender classification methods with automatically detected and aligned faces,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 3, pp. 541–547, 2008.
- [112] N. Sun, W. Zheng, C. Sun, C. Zou, and L. Zhao, “Gender classification based on boosting local binary pattern,” in *Advances in Neural Networks-ISNN*, pp. 194–201, Springer, 2006.
- [113] C. Shan, “Learning local binary patterns for gender classification on real-world face images,” *Pattern Recognition Letters*, vol. 33, no. 4, pp. 431–437, 2012.
- [114] J. Ylioinas, A. Hadid, and M. Pietikäinen, “Combining contrast information and local binary patterns for gender classification,” in *Proceedings of the 17th Scandinavian Conference on Image Analysis*, pp. 676–686, 2011.

- [115] W. Gao, B. Cao, S. Shan, X. Chen, D. Zhou, X. Zhang, and D. Zhao, “The cas-peal large-scale chinese face database and baseline evaluations,” *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 38, no. 1, pp. 149–161, 2008.
- [116] K. Grauman, M. Betke, J. Lombardi, J. Gips, and G. Bradski, “Communication via eye blinks and eyebrow raises: video-based human-computer interfaces,” *Universal Access in the Information Society*, vol. 2, no. 4, pp. 359–373, 2003.
- [117] Z. Zhang and J. Zhang, “Driver fatigue detection based intelligent vehicle control,” in *International Conference on Pattern Recognition*, pp. 1262–1265, 2006.
- [118] I. Park, J. Ahn, and H. Byun, “Efficient measurement of eye blinking under various illumination conditions for drowsiness detection systems,” *Pattern Recognition*, vol. 1, pp. 383–386, 2006.
- [119] P. Tabrizi and R. Zoroofi, “Open/closed eye analysis for drowsiness detection,” in *1st Workshop on Image Processing Theory, Tools and Applications*, pp. 1–7, 2008.
- [120] C. Xu, Y. Zheng, and Z. Wang, “Efficient eye states detection in real-time for drowsy driving monitoring system,” in *International Conference on Information and Automation*, pp. 170–174, 2008.
- [121] D. F. Dinges and R. Grace, “Perclos: A valid psychophysiological measure of alertness as assessed by psychomotor vigilance,” *Tech. Rep. MCRT-98-006, Federal Highway Administration*, 1998.

- [122] Z. Liu and H. Ai, “Automatic eye state recognition and closed-eye photo correction,” in *International Conference on Pattern Recognition*, pp. 1–4, 2008.
- [123] A. Martinez and R. Benavente, “Real time eye tracking and blink detection with USB cameras,” tech. rep., Computer Science Department, Boston University, 2005.
- [124] Y. Tian, T. Kanade, and J. Cohn, “Dual-state parametric eye tracking,” in *International Conference on Automatic Face and Gesture Recognition*, pp. 110–115, 2000.
- [125] F. Wang, M. Zhou, and B. Zhu, “A novel feature based rapid eye state detection method,” in *IEEE International Conference on Robotics and Biomimetics*, pp. 1236–1240, 2009.
- [126] H. Tan and Y. Zhang, “Detecting eye blink states by tracking iris and eyelids,” *Pattern Recognition Letters*, vol. 27, no. 6, pp. 667–675, 2006.
- [127] C. Xu, Y. Zheng, and Z. Wang, “Eye states detection by boosting local binary pattern histogram features,” in *IEEE International Conference on Image Processing*, pp. 1480–1483, 2008.
- [128] R. Sun and Z. Ma, “Robust and efficient eye location and its state detection,” *Advances in Computation and Intelligence*, pp. 318–326, 2009.
- [129] Y. Tian, T. Kanade, and J. F. Cohn, “Eye-state action unit detection by gabor wavelets,” in *Advances in Multimodal Interfaces ICMI*, pp. 143–150, Springer, 2000.
- [130] L. Breiman, *Classification and regression trees*. CRC press, 1993.

BIBLIOGRAPHY

- [131] A. M. Martinez, “The AR face database,” *CVC Technical Report*, vol. 24, 1998.
- [132] P. Wang and Q. Ji, “Learning discriminant features for multi-view face and eye detection,” in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 373–379, 2005.