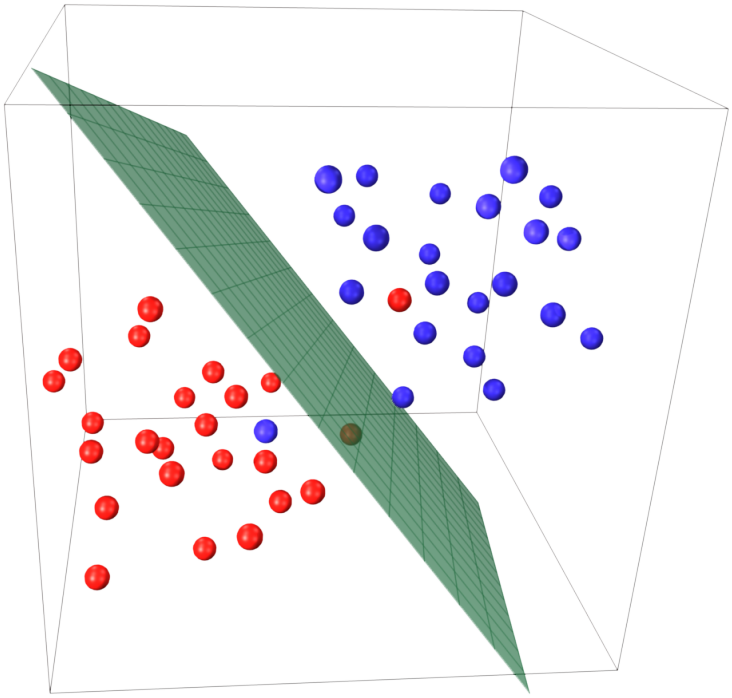
CHALMERS



Analysis and Classification of Object Poses:
Using Visual / Infrared Images and Feature Fusion

Master of Science Thesis

YIXIAO YUN

Department of Signals and Systems
Signal Processing Group
Chalmers University of Technology
Gothenburg, Sweden 2011
Report No. Ex 040/2011

MASTER OF SCIENCE THESIS 2011

Analysis and Classification of Object Poses:

Using Visual / Infrared Images and Feature Fusion

Yixiao Yun

Supervisor and Examiner: Prof. Irene Gu

Department of Signals and Systems
Signal Processing Group
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2011

Analysis and Classification of Object Poses:
Using Visual / Infrared Images and Feature Fusion

Master of Science Thesis 2011

Department of Signals and Systems
Signal Processing Group
Chalmers University of Technology
SE-41296 Gothenburg
Sweden

Tel. +46-(0)31 772 1000

Department of Signals and Systems
Gothenburg, Sweden 2011

Abstract

This master thesis addresses issues in computer vision and pattern classification. More specifically, we are interested in classification of various object poses from images, for examples, poses of human faces or cars. Analysis and classification of visual object poses are important steps towards different applications, e.g., surveillance and traffic safety. In this thesis work, several feature extraction methods are implemented, including HOG (histogram of oriented gradients) and Gabor features. A multi-class object classifier based on multi-class AdaBoost is implemented. Experiments have been conducted on large numbers of face images and car images with different poses. For human faces, the classifier contains 5 classes of poses (frontal, left, right, upward and downward), while for car images the classifier contains 4 classes of poses (frontal, rear, left and right).

Two types of images are tested: one is from the visual band and another is from the thermal infrared band. Due to different properties and characteristics in these two types of images, different types of features are extracted. For visual band images, HOG is used as the main feature descriptor. For thermal IR images, Gabor features are used.

For classification of object poses, classifiers are tested separately by using visual band images only, and thermal IR images only. Performance is then evaluated for these two types of classifiers. Attempts are also made on classifiers through fusing these two types of features in visual and IR images. Results will be presented and future work will be discussed.

Acknowledgements

I would like to thank those people who made this master thesis possible.
First of all, I greatly appreciate the help of my supervisor Prof. Irene Gu who advised and encouraged me all the way during this thesis work. I specially thank her for her infinite patience. The discussions I had with her were invaluable.
I want to say a big thanks to Electric Power Engineering Department for kindly providing us the infrared thermal camera. Also, I would like to say special thanks to Mohamed Hashim and Masih Emami who made the Chalmers Visual / Infrared Face Database together with me during this thesis work.
Besides, I am grateful to all the Chalmers teaching staff and students who generously contributed their face images to our database.
My final words go to my family. I want to thank my family, for their love and support all these years.

Yixiao Yun, Gothenburg 6/10/2011

Contents

Chapter 1

Introduction

Computers have been widely used in our daily lives, since they can handle data and computation more efficiently and more accurately than humans. Therefore, it is natural to further exploit their capabilities for more intelligent tasks, for example, analysis of visual scenes (images or videos) or speeches (audios), which is followed by logical inference and reasoning. For we humans, such tasks are performed hundreds of times every day so easily from subconscious, sometimes even without any awareness. Take the example of human visual system in recognizing object poses. Our daily lives are filled with all kinds of objects in their different poses, where the poses of human faces and cars are two of the major concerns. For human faces, there are 5 classes of poses, which are frontal, left, right, upward and downward, while for cars there are 4 classes of poses, which are frontal, rear, left and right. Each class has a huge within-class variation. For example, the exact face shape, skin color and hair style of a person vary a lot, not to mention whether the person has beard, mustache, glasses or hat, etc. Also, the exact type, color and size of a car are so much diversified. However, these factors are irrelevant to the decision that in which pose the object appears. Similarly, human beings are able to detect the poses of an object under widely varied conditions, irrespective of the partial occlusions, illumination or background clutter. So far, computers are still far behind humans in performing such analysis and classification tasks.

As a result, one topic of researchers who work in computer vision and machine learning is to grant computers the ability to see, that is, to analyze and interpret images or videos. One of the primary tasks is the detection of different classes of object poses in images and videos. Such a capability would have a wide range of applications, for example, surveillance in public places and traffic safety on the road.

This chapter introduces the problem of pose classification, in particular the poses of human faces and cars, discusses the challenges involved, and briefly presents the outline of the thesis. Section 1.1 gives a general description of the goal in pose classification. Section 1.2 discusses the difficulties in classification of object poses from visual and infrared images. The structure of this thesis is provided in Section 1.3.

1.1 The Goal

This thesis work addresses the problem about classification of object poses in images, which can be considered as a special case in object recognition. In particular, it targets the issue of designing object pose classifiers from a computer vision point of view, where the classifiers analyze given images containing objects and identify the pose class of the object. For a more precise definition of the goal, the object pose classifier is viewed as a combination of two major building blocks: a feature extraction algorithm that compiles input images into feature vectors, and a classifier that makes use of the feature vectors to decide which pose the object has. The compilation of image into feature vectors is fundamental to building robust object pose classifiers due to the ambiguity of objects in images. There are many aspects contributing to the ambiguity, such as within-class variations, background clutter, variations in illuminations, partial occlusions. More details of the challenges involved are discussed in Section 1.2. If properly selected, the image feature descriptors can mitigate these negative effects and simplify the classification task, thus allowing object poses to be discriminated with less training data and less complex learning methods.

1.2 The Challenges

The most notable difficulties in designing a robust classifier of object poses in visible light images are the many variations, which include:

- Firstly, almost every class of natural objects have large variations within the class. For example, for each human face, face shape, skin color and hair style, etc. change significantly and differences in beard, mustache and wearing glasses and hat result in further changes. As for cars, the model, the color and the size also change considerably between each other. A robust classifier must try to achieve independence of these variations.

- Secondly, background clutter is one common issue. The images may be taken from various background settings like outdoor scenes in cities and indoor environments. The classifier must be able to distinguish object class from those complex backgrounds.

- Thirdly, illumination condition varies a lot, ranging from direct sunlight and shadows during the day to artificial or dim lighting inside buildings or at night. Although some solutions for illumination invariance have been adopted, they are still extremely ineffective when compared to human visual systems in being adaptive to such changes. Therefore, a robust classifier of object poses should also provide extensive invariance to the changes of illumination and lighting conditions.

- Finally, partial occlusions may further degrade the classification performance since only part of the object in the image is available for processing. However, in this thesis work, it is assumed that all the objects in the images are fully displayed and no partial occlusion happens.

For the infrared thermal images, the effect of background clutter and illumination variance in visual images can be significantly suppressed, since they present the object edges based on temperature difference. However, this kind of images may still suffer from partial occlusions. Besides, they also have an inevitable shortage, that is, if the background temperature is close to the object temperature, the edge information will be completely damaged. In this thesis, such unfavored situations are avoided.

1.3 The Outline of the Thesis

This chapter introduced the classification problem of object poses, and described the overall goals of the thesis. The remaining chapters are arranged as follows:

- **Chapter 2** reviews the related work and background theories in object classification, with particular focus on the techniques that have been chosen for classification of object poses in this thesis work. It first describes some previous work on feature descriptions of objects in images, and then summarizes the key classification models. Lastly, image fusion strategies are also looked into.

- **Chapter 3** mainly describes the work that has been done in this thesis, for classification of object poses from visual and infrared images. Before that, it also presents the motivations for the selection of feature descriptors for each type of images and learning algorithms for object pose classification. It describes the overall classifier framework as well as some implementation details. This includes: (1) implementation of feature extraction methods using histogram of oriented gradients (HOG) and Gabor filters, (2) implementation of feature selection using principal component analysis (PCA) and feature level fusion applying Fisher linear discriminant (FLD), (3) implementing multi-class AdaBoost classifier.

- **Chapter 4** gives an overview of the datasets used, the parameter settings and the key experimental results. Emphasis will be put on testing and evaluation of pose classification on face and car databases.

- **Chapter 5** provides a discussion of the advantages and limitations of the work. It then summarizes this thesis and gives some ideas in future work.

Chapter 2

Background Theories

Analysis and classification of object poses in images and videos have received much attention in the research of computer vision and pattern recognition recently. As shown in Chapter 1, it is a challenging yet promising task, which has many potential applications. This chapter makes some literature review on the related works and some background theories in object classification, from feature extraction, pattern classification models to image fusion, and emphasis will be placed on the techniques that have been chosen for classifying object poses in this thesis work. First, some previous work on feature representation methods of objects in images will be described. After that, some popular classification algorithms and fusion techniques will be investigated.

2.1 Image Features

For object detection or classification, the image features should carry enough information of the object in the image and should not contain any irrelevant and redundant knowledge from the extraction. They should be easy to compute in order to make the approach feasible for a large collection of images and rapid extraction. They should relate well with the human perceptual characteristics since users will finally determine the suitability of the retrieved images. Besides, the image features should also provide invariance to changes in illumination, background, etc. To achieve these goals, rather than directly applying raw image intensities or gradients, one often uses some form of more advanced local image feature descriptors. Such features can be based on points, blobs, intensities, gradients, color or their combinations. In a word, the final feature descriptor need to represent the image sufficiently well for the detection and classification tasks.
There are various kinds of approaches for image feature representation, which in general can be divided into four categories. One consists of sparse representations based on points, image fragments or regions, while the other three contains dense representations using image intensities, gradients or wavelets, respectively.

2.1.1 Key Points / Region of Interest Based Approach

This type of methods extract local image features sparsely at or around some distributed points, also called key points or region of interest. The classification results are thus based on the feature vectors calculated from these key points or regions. The idea behind these approaches is that the detected key points are located in relatively more stable and more reliable image regions, which are especially informative about local image content. The overall performance in object classification then depends on the reliability, accuracy and repeatability of key point or region detection for each object class and the amount of relevant information carried by the detected points or the image regions surrounding these points. Some common techniques in interest point detection include Förstner [1], Laplacian [2] or Difference of Gaussians (DoG) [3], scale invariant Harris-Laplace [4] and also some combinations of them. Then, feature vectors or descriptors can be obtained by computing over the local image regions surrounding the key points found. The major advantage of key point / region of interest based method is the compactness of the representation, since the dimension of feature descriptors based on key points or regions are usually much smaller than the total number of image pixels, thus accelerating the following classification process. However, such methods may have limitations for general object class because most of them are designed only to handle particular objects. One of the most popular approaches for this kind of feature extraction is Scale-Invariant Feature Transformation (SIFT) [3][5].

SIFT (Scale-Invariant Feature Transform)

The SIFT feature detection technique can be used for object recognition. According to [3], the features extracted from images are invariant to image rotation and scaling, which can perform reliable matching between different views of an object or a scene. Also, the features have been shown to be robust against affine distortion, additional noise and illumination variance. The method is efficient in identifying and computing features.
The stages for SIFT feature extraction are scale-space extrema detection, key point localization, orientation assignment and generation of key point descriptors. In the following these stages are investigated in detail.

- **Scale-space extrema detection**

  For SIFT features, the interest points correspond to local extrema of Difference-of-Gaussians (DoG) filters at different scales.

  Given a Gaussian blurred image

$$L(x,y,\sigma) = G(x,y,\sigma) * I(x,y) \tag{2.1}$$

  where

$$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(x^2+y^2)}{\sigma^2}\right) \tag{2.2}$$

is a variable scale Gaussian, and the result of convolving an image with a DoGs filter

$$G(x,y,k\sigma) - G(x,y,\sigma) \tag{2.3}$$

is given by

$$D(x,y,\sigma) = L(x,y,k\sigma) - L(x,y,\sigma). \tag{2.4}$$

This is just the difference of the Gaussian blurred images at scale σ and $k\sigma$.

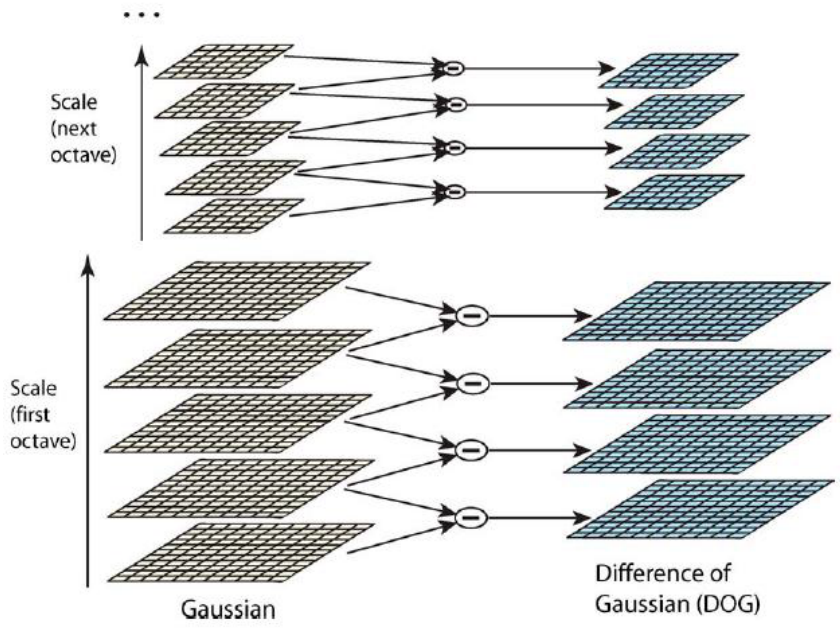


Figure 2.1: The blurred images at different scales and the resulting DoG images (from [3]).

As the preparation step for key point localization, the image is convolved with Gaussian filters at different scales and DoG images are generated from the difference of adjacent blurred images, as shown in Figure 2.1.

The convolved images are grouped by octave which is set to doubling the value of σ, and the value of k is selected so that a fixed number of blurred images per octave

are obtained. This also ensures that the same number of DoG images per octave are obtained.

It should be noted that the DoG filter provides an approximation to the scale-normalized Laplacian of Gaussian (LoG) $\sigma^2 \nabla^2 G$, so the DoG filter is actually a tunable bandpass filter.

- **Key point localization**

  Interest points, also called key points in the SIFT framework, are identified as local maxima or minima of the DoG images across scales. Each pixel in the DoG images is compared to its 8 neighbors at the same scale, and also to the 9 corresponding neighbors at neighboring scales. If the pixel turns out to be a local maximum or minimum, it is selected as a candidate key point, as illustrated in Figure 2.2.

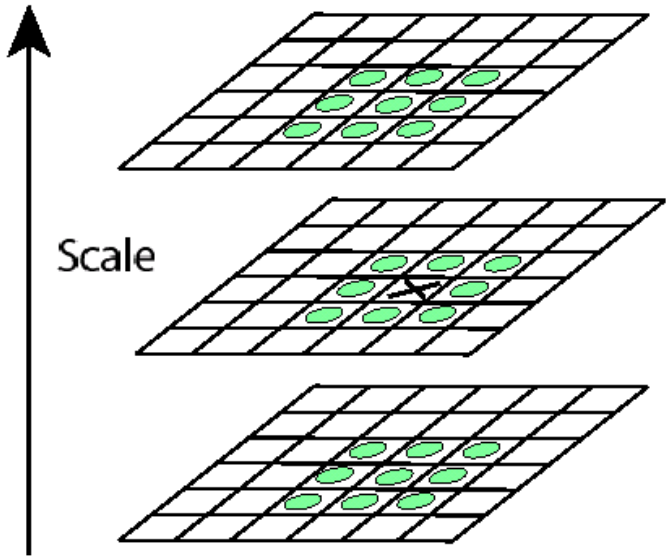


Figure 2.2: Local extrema detection, the pixel marked $\times$ is compared against its 26 neighbors in a 3×3×3 neighborhood that spans adjacent DoG images (from [3]).

For each candidate key point, the interpolation of nearby data can be applied to more accurately determine its position, since the key points with low contrast will be removed and responses along edges will be eliminated. After that, the key point is assigned an orientation.

- **Orientation assignment**

  In order to determine the orientation of the key point, a histogram of gradient orientation is computed in the neighborhood of the key point using the Gaussian image

at the closest scale to that of the key point. The contribution of each neighboring pixel is weighted by the gradient magnitude and a Gaussian window with a σ that is 1.5 times the scale of the key point.

Peaks in the histogram indicate the dominant orientations. A separate key point is created for the direction corresponding to the maximum value in the histogram, and for any other direction that has a value at least 80% of the histogram maximum.

At this stage, all the properties of the key point are measured with respect to its orientation, invariance to rotation is thus achieved.

- **Generation of key point descriptor**

  Once the orientation of a key point is selected, the feature descriptor can be computed as a set of orientation histograms on 4×4 pixel neighborhoods. The histograms are relative to the key point orientations, which derive from the Gaussian image closest in scale to that of the key point.

  Same as above, the contribution of each pixel is weighted by the gradient magnitude, and by a Gaussian window with σ 1.5 times the scale of the key point.

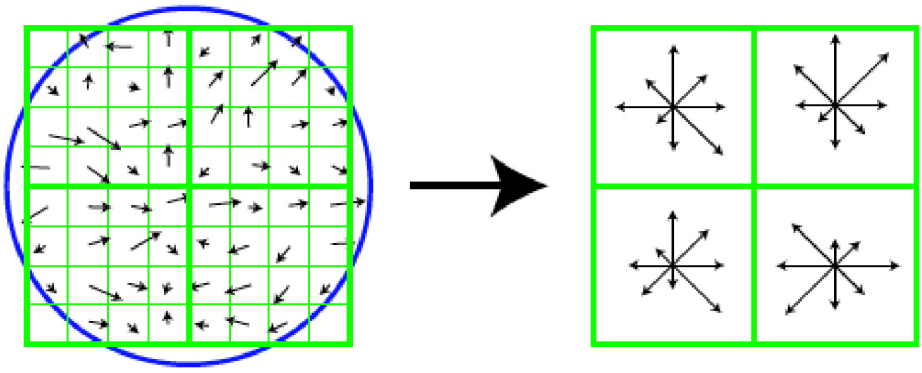


Figure 2.3: SIFT feature description. Left: image gradients. Right: key point descriptor (from [3]).

Histograms contain 8 bins each, and each descriptor contains an array of 4 histograms around the key point, as shown in Figure 2.3. This leads to a SIFT feature vector with $4 \times 4 \times 8 = 128$ elements. This vector is normalized to enhance invariance to changes in illumination.

2.1.2 Image Intensity Based Approach

One of the initial works of feature representation using simple image intensities is the eigenfaces approach in [6] and [7], where the pixel values of face images are collected into a high-dimension feature vector and Principle Component Analysis (PCA) is applied to

reduce the dimensions while keeping the most relevant features. Another work using image intensities is the face detection system in [8] where lighting conditions of the images are corrected by performing histogram equalization before classification. Here the histogram of intensities is introduced, with two operators closely connected to it, which are contrast stretching and histogram equalization.

Histogram of Intensities

The histogram of intensity is the measure of the number of pixels in an image at each different intensity value found in that image. Mathematically, the histogram for a grayscale image is a discrete function $h(r_k) = n_k$, where r_k is the k-th gray level and n_k is the number of pixels in the image that have gray level r_k.
Histograms can also to applied to color images. For color images, either individual histograms of red, green and blue (RGB) channels can be created, or a 3D histogram can be produced, with three axes representing the RGB channels and the value at each point representing the pixel count.
The histogram processing is very simple. The image intensities are considered as random variables with a probability density function (PDF), and the PDF can be estimated from the empirical data provided by the image itself. Therefore, every pixel in the image is scanned and a count of the number of pixels found at each intensity value is kept. In other words, the frequency distribution of image intensities is recorded. This is then used to construct a histogram of intensities for the image, as shown in Figure 2.4.

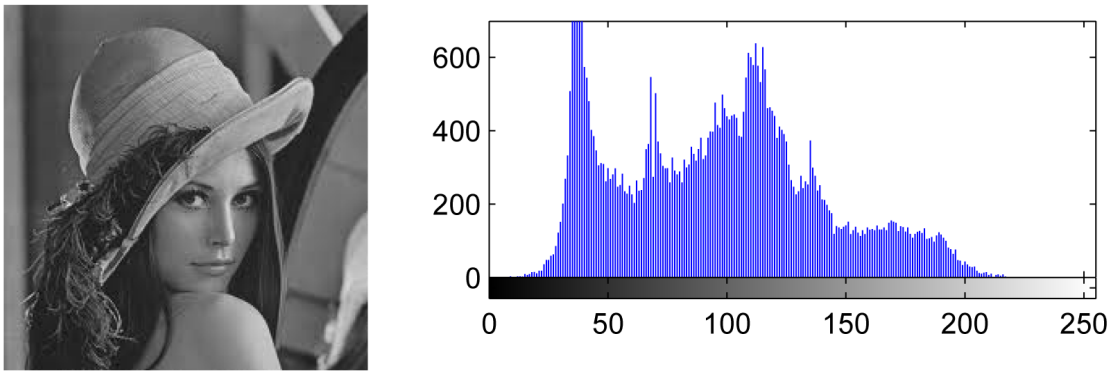


Figure 2.4: Left: sample image. Right: corresponding intensity histogram.

The histogram should be normalized by dividing each bin by the total number of pixels in the image, in order to give an estimate for the PDF. Then, each element of the array shows the probability of that intensity occurring at a randomly selected pixel. So the histogram of intensities contains only global information of the entire image, which can

be used as a very compact feature descriptor for object classification. However, the major drawback of this approach is that each image has only one histogram, and all spatial information of the image is discarded. One possible solution could be to integrate the histogram of intensities with some other feature types to jointly produce some robust feature representations of the image. Another possible solution could first divide the image into dense grid of uniformly distributed pixel regions, compute the histogram of intensities over each cell, and then concatenate all the histograms into one large feature vector.
The histogram of intensities can be improved by many image enhancement operators. Two operators which are commonly used are contrast stretching and histogram equalization. They are both based on the assumption that an image has to use the full intensity range to display the maximum contrast.

- **Contrast stretching**

  Contrast stretching aims to improve the contrast in an image by stretching the range of intensity values to the maximumly allowed extent. The process is done by applying a linear scaling function to the image pixel values.

  First of all, it is necessary to specify the upper and lower bound of image intensities over which the image is to be normalized. In most cases, these limits will be the maximum and minimum values that the image type allows. Let a and b denote the upper and the lower limits as , respectively.

  The simplest kind of normalization searches for the highest and lowest pixel values currently present in the image, which are denoted by c and d. Then each pixel is scaled using the following function

$$\hat{I}(x,y) = (I(x,y) - d)\left(\frac{a-b}{c-d}\right) + b. \tag{2.5}$$

  Thus, the contrast in the image is improved without distorting relative intensities too significantly, which is illustrated in Figure 2.5.

- **Histogram equalization**

  Unlike contrast stretching, histogram equalization uses a non-linear and monotonic mapping. The idea of histogram equalization is that the pixels should be distributed evenly over the whole intensity range, i.e. the aim is to transform the image so that the output images has a flat histogram (Figure 2.6).

  Consider a histogram of intensities $h(r_k) = n_k$, where r_k is the k-th intensity level and n_k is the number of pixels in the image that have intensity level r_k. The probability of a intensity r_k occurring at a randomly selected pixel is

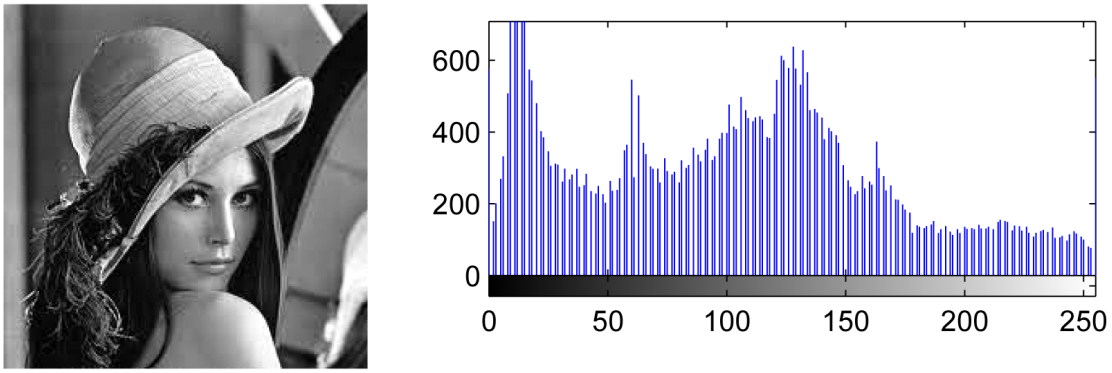


Figure 2.5: Left: contrast-stretched image. Right: resultant intensity histogram.

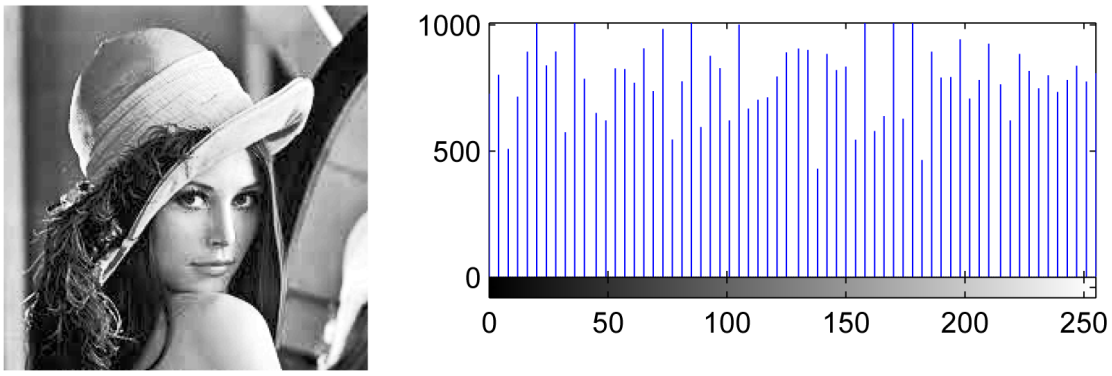


Figure 2.6: Left: histogram-equalized image. Right: resultant intensity histogram.

$$p(r_k) = \frac{n_k}{n}, \quad 0 \leq k < L \tag{2.6}$$

where n is the total number of pixels in the image and L is the totally number of intensity levels. In fact, $p(r_k)$ is equal to $h(r_k)$ if $h(r_k)$ has been normalized to [0,1]. Now define the cumulative distribution function (CDF) that corresponds to $p(r_k)$ as

$$P(r_k) = \sum_{j=0}^{k} p(r_j) \tag{2.7}$$

which is also the accumulative normalized histogram of the image.

A transformation is created so that the new CDF will be linearized across the value range:

$$s_k = T(r_k) = P(r_k) \tag{2.8}$$

As a result, the transformation T has mapped the intensity levels into the range [0,1].

2.1.3 Gradient Based Approach

Edge in the images has been one of the most widely used features in object detection and classification. The basic idea behind this kind of approaches is that edges give dominant gradient magnitudes over the entire image, so the edge features can be represented in various ways using image gradients. A popular approach is the pedestrian detection system using histogram of oriented gradients (HOG) in [9] and [10], where gradients are used to compute feature descriptors based on histogram of dominant orientations within dense and overlapping image regions.

Histogram of Oriented Gradients (HOG)

Histogram of oriented gradients can be used as feature descriptors for the purpose of object detection, where the occurrences of gradient orientation in localized parts of an image play important roles. This technique is similar to scale-invariant feature transform (SIFT) but differs in that it operates on a dense grid of uniformly spaced cells and uses local contrast normalization on overlapping blocks for improved accuracy. HOG feature descriptors are first described in [9], where great success has been achieved on pedestrian detection in both images and videos, as well as on a variety of common animals and vehicles in static pictures.
The basic idea behind HOG is that the appearances and shapes of local objects within an image can be well described by the distribution of intensity gradients as the votes for dominant edge directions. Such feature descriptor can be obtained by first dividing the image into small contiguous regions of equal size, called cells, then collecting a histogram of gradient directions for the pixels within each cell, and at last combining all these histograms. In order to improve the detection accuracy against varied illumination and shadowing, local contrast normalization can be applied by computing a measure of the intensity across a larger region of the image, called a block, and using the resultant value to normalize all cells within the block.
Thus, the HOG feature descriptor holds some crucial advantages over other techniques. For one thing, this method results in significant invariance to geometric and photometric transformations since it operates on localized cells so that those changes would only appear in larger spatial regions. For another, according to [9], HOG features are more tolerant

to the individual body movement of pedestrians as long as they keep a roughly upright position so this descriptor is particularly suitable for human detection.
In fact, there are four variants of HOG block scheme: Rectangular HOG (R-HOG), Circular HOG (C-HOG), as shown in Figure 2.7, Bar HOG and Center-Surround HOG [10], among which R-HOG is the most popular. Regardless of the block type, some crucial procedures of HOG feature extraction are summarized below (Figure 2.8). More details can be found in [10].

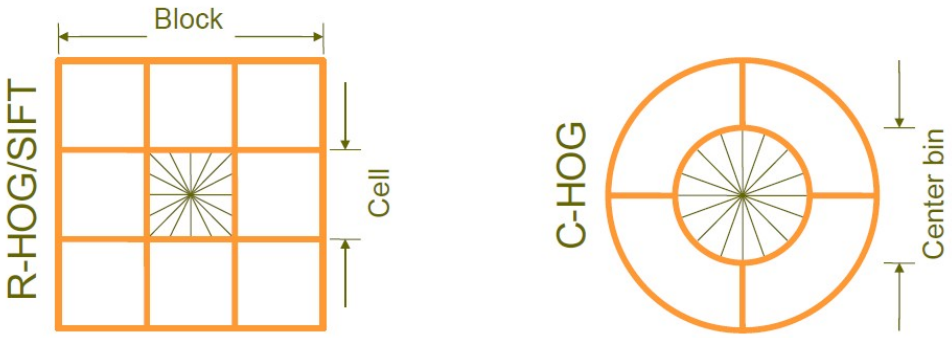


Figure 2.7: Left: R-HOG block. Right: C-HOG block (from [10]).

- **Gradient computation**

  The first step in HOG feature extraction is the computation of image gradients. This is done by applying the 1D centered, point discrete derivative mask in both the horizontal and vertical directions, which in specific are filter kernels of the following form:

  $$[-1,0,1] \quad \text{and} \quad [-1,0,1]^T$$

  In fact, there are many more complex masks, such as Sobel, Prewitt, Canny or diagonal masks, but these masks generally result in poorer performance [9]. Then, the magnitude and orientation at each pixel $I(x,y)$ is calculated by

  $$\begin{cases} G_{mag}(x,y) = \sqrt{G_x^2(x,y) + G_y^2(x,y)} \\ \theta(x,y) = \arctan\left(G_y(x,y)/G_x(x,y)\right) + \pi/2 \end{cases} \tag{2.9}$$

  where $G_x(x,y)$ and $G_y(x,y)$ are the gradient values at each pixel in horizontal and vertical direction, respectively. For color images, the channel with the largest magnitude gives the pixel's dominant magnitude and orientation. It should be noted that $\pi/2$ is necessary since the arctan operator results in a range between $-\pi/2$ and

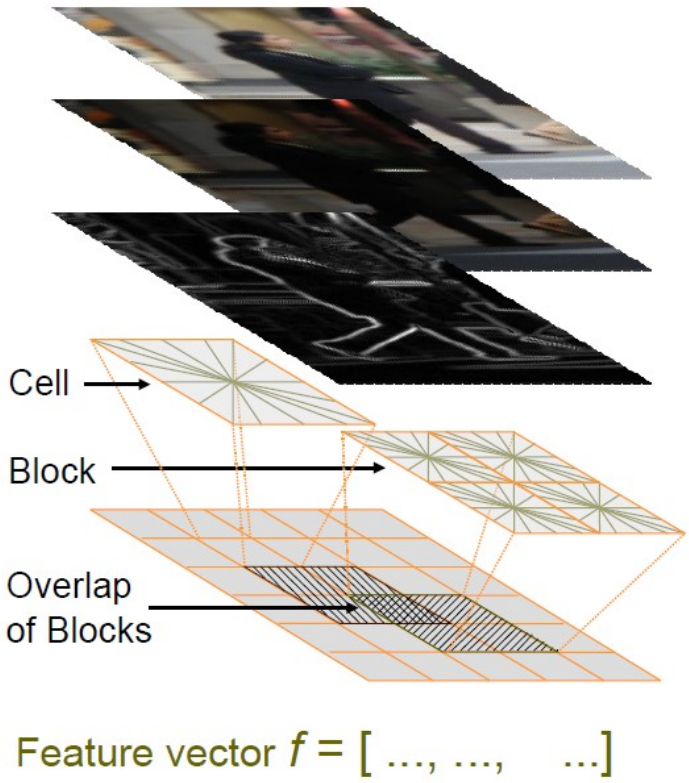


Figure 2.8: The processing chain of HOG feature descriptor (from [10]).

$\pi/2$, but for unsigned orientation scheme which gives better performance, it ranges from 0 to π.

- **Orientation binning**

  In this procedure, the histograms for each cell are created. The cells are pixel regions that are either rectangular or radial in shape, and the histogram bins are evenly expanded from 0° to 180° (or from 0° to 360° in the case of signed orientation), so every histogram bin has a spread of 20°. Every pixel in the cell casts a weighted voting into one of the 9 histogram bins that its orientation belongs to. As for the weight of votes, it can either be the gradient magnitude itself, or some function of the magnitude, for example, the square root or square of the gradient magnitude, or some clipped version of the magnitude. Generally, the gradient magnitude is directly used.

- **Descriptor blocks**

  To obtain the robustness against various illumination and contrast, the gradient strengths must be locally normalized. This leads to grouping the cells into larger

pixel regions called blocks. These blocks overlap with neighboring blocks, so that each cell can contribute its orientation distribution more than once. As mentioned before, totally 4 block geometries exist, with one most commonly used: Rectangular HOG blocks. R-HOG blocks are usually square grids, and the optimal parameters are found to be 2×2 cell blocks of 8×8 pixel cells (or 3×3 cell blocks of 6×6 pixel cells). Besides, a Gaussian spatial window can be applied to each block before histogram voting so that the weight of each pixel around the edge of the block can be significantly suppressed.

- **Block normalization**

  There are four different way proposed to normalize the blocks. Let $\boldsymbol{v}$ denote the non-normalized feature vector that collects all cell histograms from a given block, $\|\boldsymbol{v}\|_k$ denote its k-norm for $k = 1,2$ and eps denote some small constant. Then the normalization schemes have the following forms:

$$\text{L2-norm:} \quad \hat{\boldsymbol{v}} = \frac{\boldsymbol{v}}{\sqrt{\|\boldsymbol{v}\|_2^2 + eps^2}} \tag{2.10}$$

$$\text{L1-norm:} \quad \hat{\boldsymbol{v}} = \frac{\boldsymbol{v}}{(\|\boldsymbol{v}\|_1 + eps)} \tag{2.11}$$

$$\text{L1-sqrt:} \quad \hat{\boldsymbol{v}} = \sqrt{\frac{\boldsymbol{v}}{(\|\boldsymbol{v}\|_1 + eps)}} \tag{2.12}$$

  L2-Hys is computed by re-normalizing the clipped L2-norm. All the normalization schemes provide much better performance than non-normalized case.

  The final HOG feature descriptor is then the vector containing the elements of the normalized cell histograms from all of the block regions.

2.1.4 Wavelet Based Approach

Wavelets that are created to have different frequencies can be combined and convolved with many different kinds of data, including audio signals and images, to extract information from the data. Mathematically, the wavelet will resonate if the signal being convolved contains information of similar frequency, which is the core concept for many practical application of wavelets, e.g., feature extraction based on Haar wavelets [11] and Gabor wavelets [16].

Haar-like Features

Haar-like features can be used in object recognition, especially for human face detection. This kind of features get the name due to their intuitive similarity with Haar wavelets.

The initial type of this feature set is discussed in [12], which is based on Haar wavelets. Later, this idea of using Haar wavelets was adapted in [11] and was further developed into the so called Haar-like features. In general, a Haar-like feature sums up the intensity values of the pixels in each rectangular region all inside a subsection of an image and computes the difference between the sums. The difference can be then used to identify subsections of the image. For example, among all human faces the region of eyes is darker than the region of cheeks. As a result, a Haar-like feature for human faces can be obtained by placing two adjacent rectangles over the eye and the cheek region.

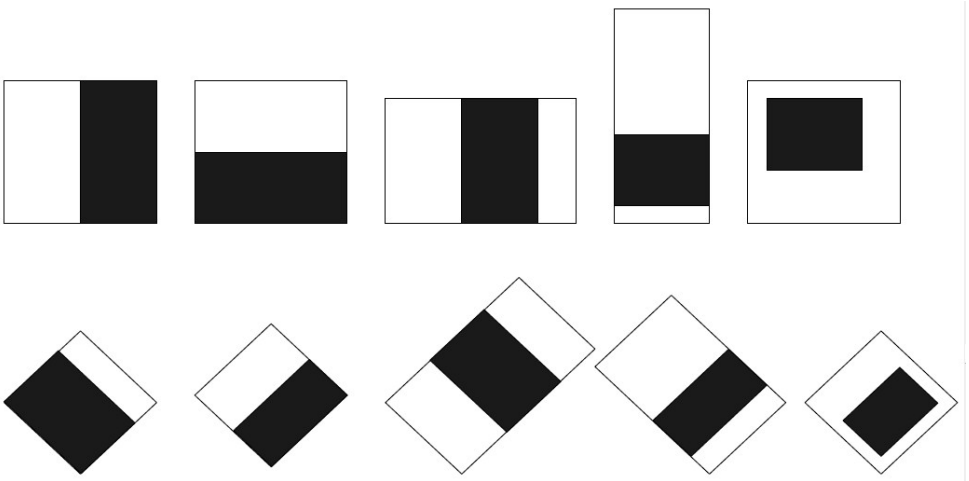


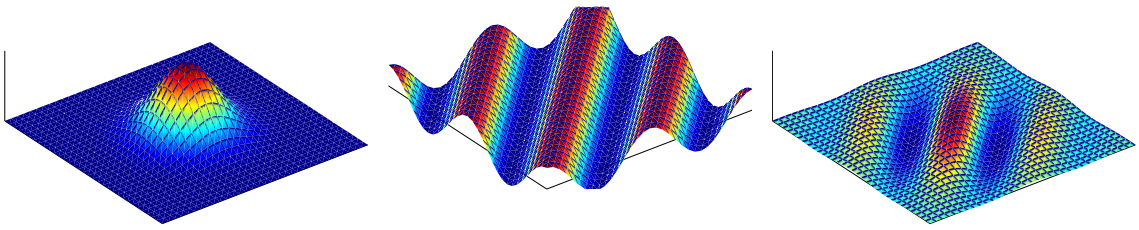
Figure 2.9: Different types of Haar-like features.

According to [11], a bounding box of proper size slides over the input image, and for each subsection of the image, Haar-like feature is computed. Each time, the difference is compared to a learned threshold for object / non-object decision. However, this kind of classifier works only slightly better than random guessing, so a ensemble scheme is used to group a large number of Haar-like features in a classifier cascade to jointly produce a strong classifier which can describe an object with sufficient accuracy.

The major advantage of Haar-like features is the fast calculation speed, which makes it rather competitive in real-time object recognition. The fast computation of Haar-like features is achieved by using summed area tables [13], also called integral images. For improved performance of object recognition in more orientations, tilted and rotated Haar-like features are also proposed, in [14] and [15]. Figure 2.9 shows how it can be done for common and tilted Haar-like features.

Gabor Features

Gabor filter, named after Dennis Gabor, can be used to detect object edges in an image. In the spatial domain, a 2D Gabor filter is usually considered as a Gaussian kernel function modulated by a sinusoidal plane wave, as illustrated in Figure 2.10. Gabor filters with properly selected frequencies and orientations can be assembled to form a filter bank which is similar to the model of human visual system, thus being found to be particularly appropriate for pattern classification use. There are more than one representation of 2D Gabor filters, and here one common definition is introduced [16].



(a) a 2D Gaussian envelope.

(b) Real part of a sinusoidal plane wave.

(c) Real part of the corresponding 2D Gabor filter.

Figure 2.10: A 2D Gabor filter is obtained by modulating a sinusoidal plane wave with a Gaussian envelope.

As is mentioned above, a 2D Gabor filter $g(x,y)$ is formed from two components, a complex sinusoidal carrier $s(x,y)$ and a Gaussian envelope $w(x,y)$.

$$g(x,y) = s(x,y) \cdot w(x,y) \tag{2.13}$$

The envelope has a Gaussian profile (Figure 2.10a) and is described by the following equation:

$$w(x,y) = \exp(-\frac{1}{2}\left(\frac{\hat{x}^2}{\sigma_x^2} + \frac{\hat{y}^2}{\sigma_y^2}\right)) \tag{2.14}$$

where

$$\begin{cases} \hat{x} = x\cos\theta + y\sin\theta \\ \hat{y} = -x\sin\theta + y\cos\theta \end{cases} \tag{2.15}$$

θ represents the rotation angle, σ_x and σ_y are the standard deviations of the envelope along x- and y- dimensions. If $\sigma_x = \sigma_y = \sigma$, then the 2D Gaussian function can be rewritten as:

$$w(x,y) = \exp(-\frac{\hat{x}^2 + \hat{y}^2}{2\sigma^2}) \tag{2.16}$$

The complex carrier takes the form:

$$s(x,y) = \exp(j2\pi f\hat{x}) \tag{2.17}$$

The real part of the function (Figure 2.10b) is given by:

$$\Re\{s(x,y)\} = \cos(2\pi f\hat{x}) \tag{2.18}$$

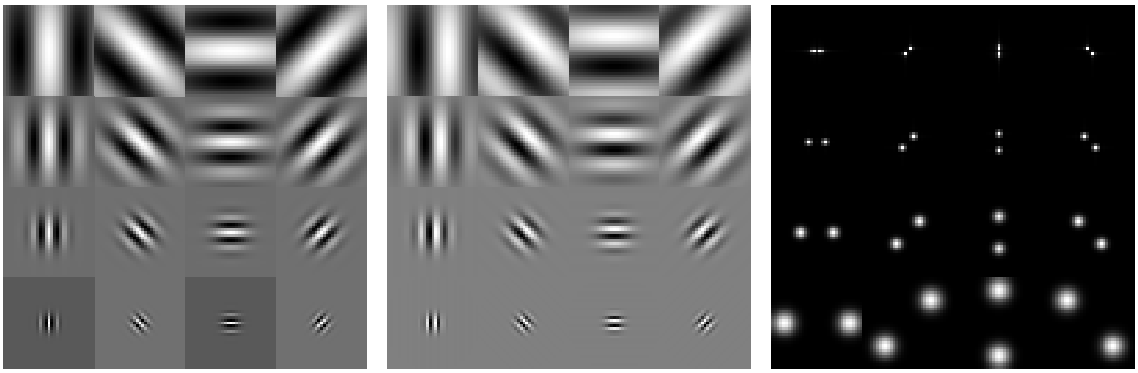
and the imaginary part of the function is given by:

$$\Im\{s(x,y)\} = \sin(2\pi f\hat{x}) \tag{2.19}$$

where f defines the spatial frequency.
The complex Gabor filter kernel is therefore as follows:

$$g(x,y) = \exp(-\frac{\hat{x}^2 + \hat{y}^2}{2\sigma^2})\exp(j2\pi f\hat{x}) \tag{2.20}$$

and the real component of the Gabor filter is shown in Figure 2.10c.
If the frequency f and the orientation θ are properly chosen, a bank of Gabor filters which covers the entire frequency domain can be obtained, which can be observed in Figure 2.11 and 2.12.

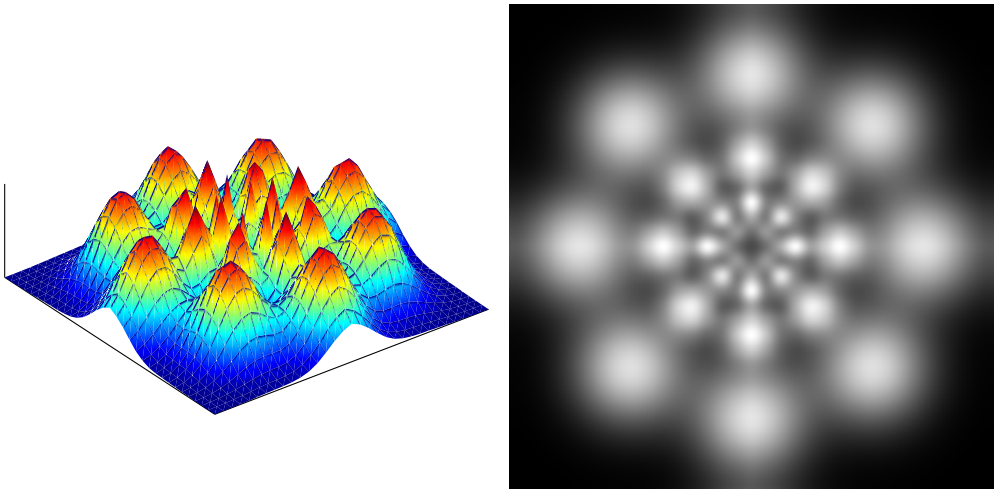


(a) Real part of Gabor filters in spatial domain.

(b) Imaginary part of Gabor filters in spatial domain.

(c) Gabor filters in the frequency domain.

Figure 2.11: A bank of Gabor filters. For each column of the sub-figures from left to right: $\theta = 0(\pi), \pi/4(5\pi/4), \pi/2(6\pi/4), 3\pi/4(7\pi/4)$, for each row of from top to bottom, $\sigma_x = \sigma_y = 16, 8, 4, 2$, where $f = 1/\sigma_x$.



(a) 3D view. **(b)** Corresponding 2D view.

Figure 2.12: A bank of Gabor filters in the frequency domain. For each layer, $\theta = 0(\pi), \pi/4(5\pi/4), \pi/2(6\pi/4), 3\pi/4(7\pi/4)$, for each circle from outer layer to inner layer, $\sigma_x = \sigma_y = 2, 4, 8, 16$, where $f = 1/\sigma_x$.

Commonly, the Gabor representation of an image is computed by convolving the image with the Gabor filters in the bank [17]. Let $I(x,y)$ be the intensity at the coordinate (x,y) in a grayscale image, its convolution with a Gabor filter $g(x,y)$ is defined as:

$$h(x,y) = I(x,y) * g(x,y) \tag{2.21}$$

where the filter responses are complex valued, so either the real part $\Re\{h(x,y)\}$ or the magnitude $\sqrt{\Re^2\{h(x,y)\} + \Im^2\{h(x,y)\}}$ of the filter response is taken, which are then respectively reshaped into 1D vectors and normalized for enhanced robustness against illumination variance. The Gabor feature vector is thus obtained by concatenating these 1D vectors.

2.2 Classification Methods

All classification algorithms are based on the assumption that the data in question holds one or more features, each of which belongs to one of several distinct and exclusive classes. Classification algorithms typically includes two successive procedures: training and testing. In the initial training phase, a unique description of each class is made by learning with typical features extracted from the training samples and separating them in the feature space. In the subsequent testing phase, these feature space separations are used to classify newly input feature vectors extracted from the testing dataset. Therefore, the

classification problem can also be viewed as determining to which sub-space class each feature vector belongs.
One of the most popular techniques in machine learning is AdaBoost. It is widely used for object recognition and classification, due to its outstanding performance and the ease to use. Moreover, its capabilities to automatically select the most relevant feature descriptors from large feature sets are also often exploited.

2.2.1 AdaBoost

AdaBoost, short for Adaptive Boosting, is a technique which can be used to improve the performance of many learning algorithms. Generally, AdaBoost sequentially applies a given learning algorithm with respect to a set of training samples and adds each prediction to an ensemble. When being added to the emsemble, the prediction is typically weighted according to its accuracy. After this, the dataset is also reweighted: samples that are misclassified gain weights and samples that are correctly classified lose weights. Thereby each successive classifier is forced to focus on those samples that are misclassified by previous ones in the sequence. AdaBoost is chosen in this thesis work since its basic idea is quite simple but still very successful, with performances comparable to more complex methods such as Support Vector Machines [18].

Original AdaBoost

In fact, AdaBoost is originally intended only for boosting binary classifiers, so it can not be directly applied to multi-class cases. For multi-class problems, there are many extensions and modifications of AdaBoost, but all derive from the same kind of model, that is, the forward stagewise additive modeling.
Forward stagewise additive modeling approaches the optimization problem by sequentially adding new basis functions to the expansion without adjusting the parameters and coefficients of those that have been already added. AdaBoost is equivalent to this model and it uses the exponential loss function below for binary case:

$$L(y,f(\boldsymbol{x})) = \exp(-yf(\boldsymbol{x})) \tag{2.22}$$

For binary AdaBoost, training samples are input as feature vectors $\{\boldsymbol{x_i}\}$ with their desired outputs $\{y_i\} \in \{-1,1\}$, where $i = 1,2,...,N$, the basis functions in the forward stagewise additive model are the weak learners $T^{(m)}(\boldsymbol{x}) \in \{-1,1\}$. Using the exponential loss function, the problem becomes solving:

$$(\alpha^{(m)},T^{(m)}) = \arg\min_{\alpha,T} \sum_{i=1}^{N} \exp[-y_i(f^{(m-1)}(\boldsymbol{x}_i) + \alpha T(\boldsymbol{x}_i))] \tag{2.23}$$

for the weak learner $T^{(m)}$ and its corresponding weight coefficient $\alpha^{(m)}$ to be added at each step. This can expressed as:

$$(\alpha^{(m)},T^{(m)}) = \arg\min_{\alpha,T} \sum_{i=1}^{N} w_i^{(m)} \exp(-\alpha y_i T(\boldsymbol{x}_i)) \tag{2.24}$$

where

$$w_i^{(m)} = \exp(-y_i f^{(m-1)}(\boldsymbol{x}_i)). \tag{2.25}$$

Since $w_i^{(m)}$ is independent on α and $T(\boldsymbol{x}_i)$, it can be regarded as a weight factor that is applied to each training samples. This weight depends on $f^{(m-1)}(\boldsymbol{x}_i)$ so the weight changes during each iteration m.
It can be easily observed that

$$\begin{cases} \text{If } y_i = T(\boldsymbol{x}_i), & \text{then } y_i \cdot T(\boldsymbol{x}_i) = 1; \\ \text{If } y_i \neq T(\boldsymbol{x}_i), & \text{then } y_i \cdot T(\boldsymbol{x}_i) = -1. \end{cases} \tag{2.26}$$

Therefore, the criterion in (2.24) can be expressed as

$$e^{-\alpha} \sum_{y_i = T(\boldsymbol{x}_i)} w_i^{(m)} + e^{\alpha} \sum_{y_i \neq T(\boldsymbol{x}_i)} w_i^{(m)}, \tag{2.27}$$

which in turn can be rewritten as

$$(e^{\alpha} - e^{-\alpha}) \sum_{i=1}^{N} w_i^{(m)} \mathbb{I}(y_i \neq T(\boldsymbol{x}_i)) + e^{-\alpha} \sum_{i=1}^{N} w_i^{(m)} \tag{2.28}$$

Apply gradient descent method to (2.28) and solve for α, by taking partial derivative respect to α and set the resulting equation is to 0, one obtain α as

$$\alpha^{(m)} = \frac{1}{2} \log \frac{1 - err^{(m)}}{err^{(m)}} \tag{2.29}$$

where $err^{(m)}$ is the minimized weighted error rate

$$err^{(m)} = \frac{\sum_{i=1}^{N} w_i^{(m)} \mathbb{I}(y_i \neq T^{(m)}(\boldsymbol{x}_i))}{\sum_{i=1}^{N} w_i^{(m)}}. \tag{2.30}$$

As a result, the approximation can be updated as

$$f^{(m)}(\boldsymbol{x}) = f^{(m-1)}(\boldsymbol{x}) + \alpha^{(m)} T^{(m)}(\boldsymbol{x}). \tag{2.31}$$

So the weight for the next iteration can be accordingly updated as

$$w_i^{(m+1)} = \exp(-y_i f^{(m)}(\boldsymbol{x}_i)) = w_i^{(m)} \cdot e^{-\alpha^{(m)} y_i T^{(m)}(\boldsymbol{x}_i)}. \tag{2.32}$$

Considering the fact that

$$-y_i T^{(m)}(\boldsymbol{x}_i) = 2 \cdot \mathbb{I}(y_i \neq T(\boldsymbol{x}_i)) - 1, \tag{2.33}$$

the updating scheme of sample weights becomes

$$w_i^{(m+1)} = w_i^{(m)} \cdot e^{\beta^{(m)} \mathbb{I}(y_i \neq T(\boldsymbol{x}_i))} \cdot e^{-\alpha(m)} \tag{2.34}$$

where $\beta^{(m)} = 2\alpha(m)$. The multiplication factor $e^{-\alpha(m)}$ is applied to all weights so it can be ignored.

At this stage, the original AdaBoost algorithm can be summarized in Table 2.1.

When AdaBoost is asked to classify an previously unknown sample, each classifier in the ensemble contributes its own weight $\beta^{(m)}$ to either one of the two classes it predicts, and in the end, the class with the higher value is chosen as the final prediction. For better illustration, a two-class problem on a 2D feature space is taken as the example in Figure 2.13.

During each boosting round, the weights of wrongly classified samples are increased. In this way, the weak learner for the next boosting round will be forced to pay attention on those misclassified. When combining the predictions after each boosting round, the training error rate is thus decreased to some extent, as shown in Figures 2.13.

At last, one can see in Figure 2.13 that the training error rate has been significantly reduced by AdaBoost, where each weak learner are combined together in a smart way, that is, assigning weights to each prediction made by the weak learners according to their accuracy.

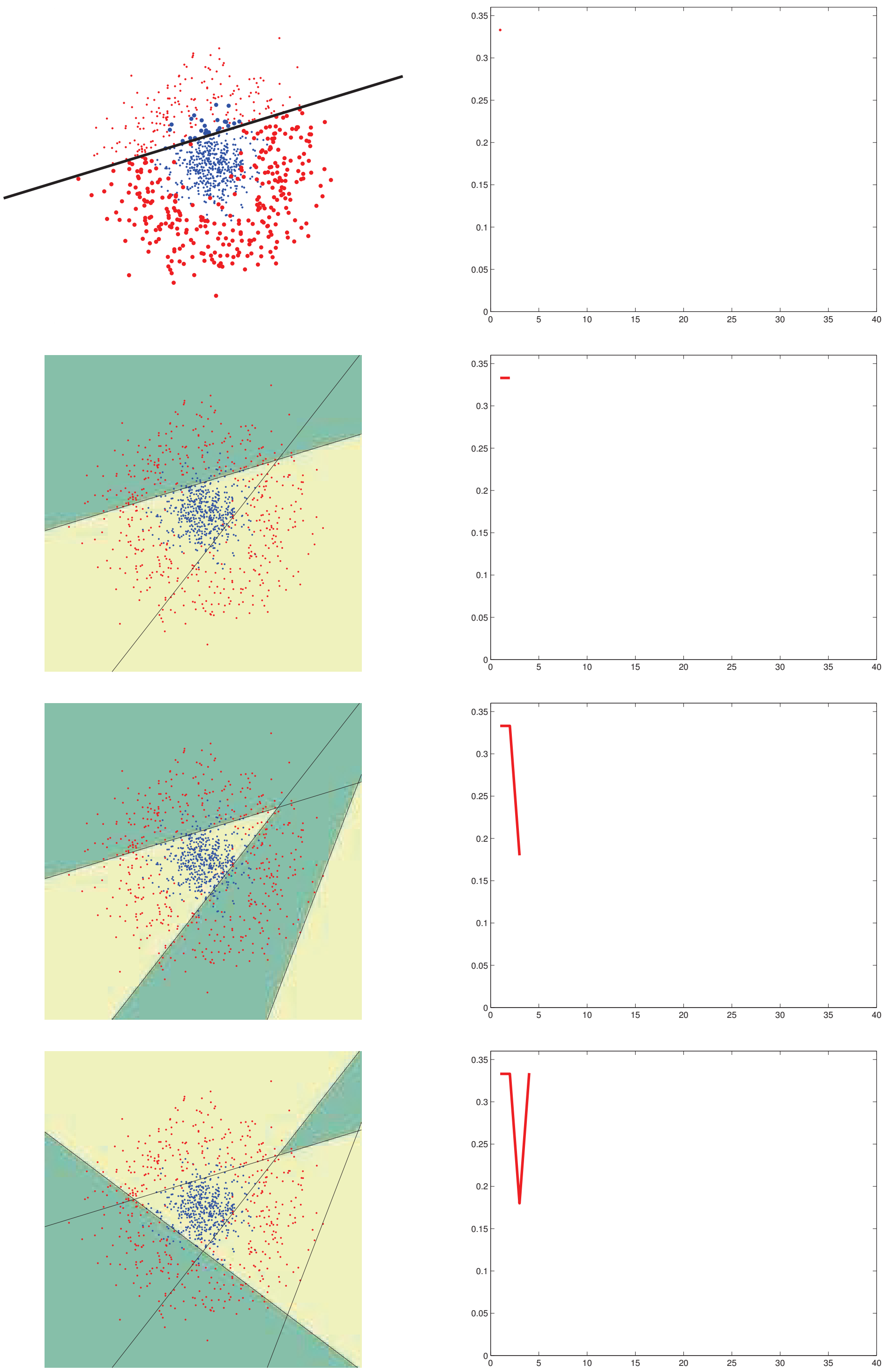
It should be noted that the weight for each classifier in the ensemble should be a positive value, that is,

$$\beta^{(m)} = \log \frac{1 - err^{(m)}}{err^{(m)}} > 0. \tag{2.35}$$

The solution to this inequality is

$$0 < err^{(m)} < 0.5, \tag{2.36}$$

so each weak learner must have an accuracy greater than 50%, otherwise the weight distribution for the training dataset would not be updated or to be updated towards the wrong direction thus causing AdaBoost out of work. This is also the reason why original AdaBoost algorithm can easily fail to work when facing multi-class classification problems that are more complicated than binary cases.


0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0 5 10 15 20 25 30 35 40

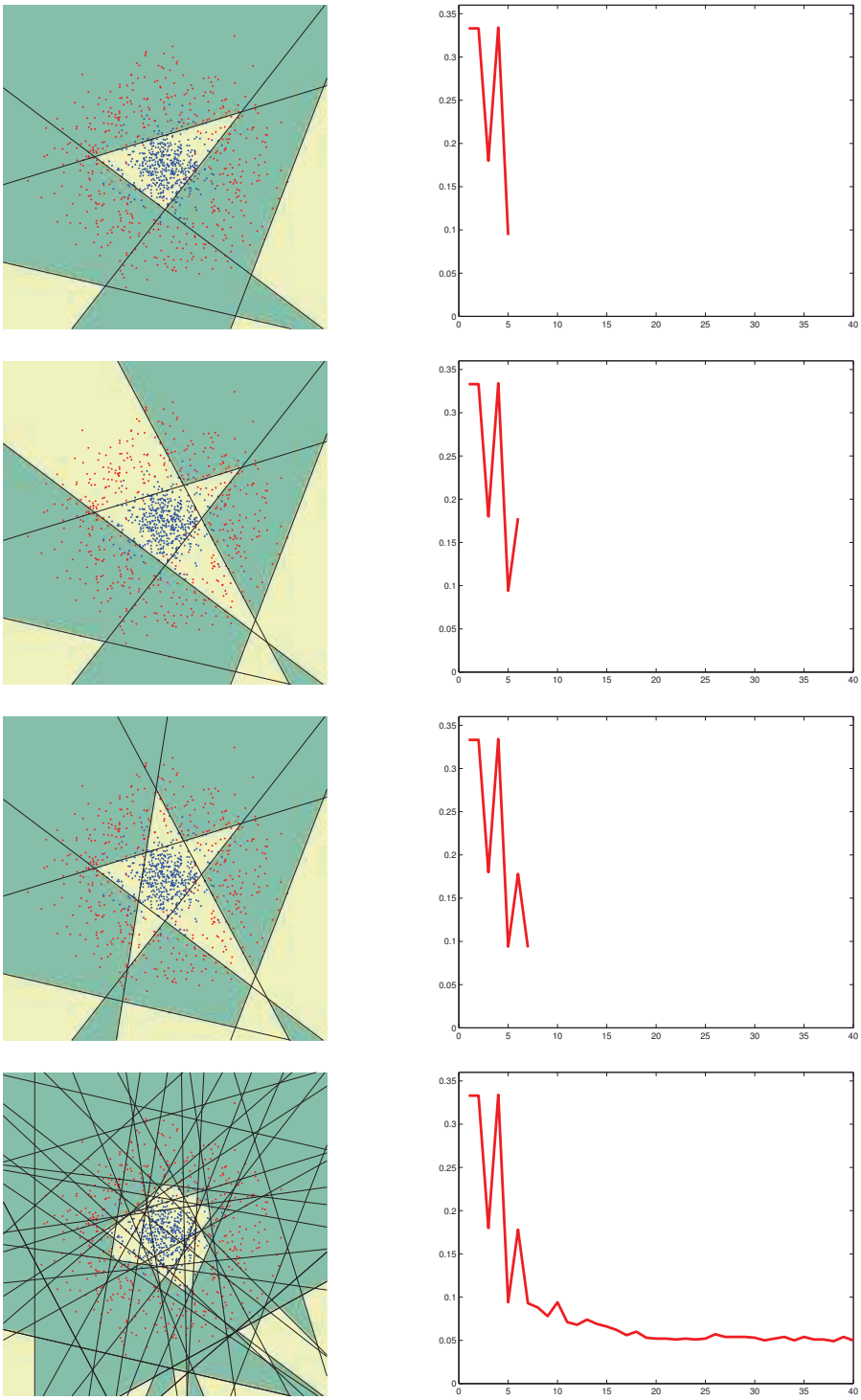


Figure 2.13: Visualization of AdaBoost (from [19]). For each row from top to bottom: boosting round 1, 2, 3, 4, 5, 6, 7, 40. Right column: corresponding training error rate after each round.

1. Initialize the weight for each training sample, $w_i = 1/N, i = 1,2,...,N$.
2. For $m = 1$ to M:
   (a) Fit a classifier $T^{(m)}(\boldsymbol{x})$ to the training dataset using weights w_i.
   (b) Compute the weighted training error rate for the classifier:
   $$err^{(m)} = \sum_{i=1}^{N} w_i \mathbb{I}\left(c_i \neq T^{(m)}(\boldsymbol{x}_i)\right) / \sum_{i=1}^{N} w_i.$$
   (c) If $err^{(m)} \leq 0$ or $err^{(m)} \geq 0.5$, then abort loop.
   (d) Compute the weight for the classifier in the ensemble:
   $$\beta^{(m)} = \log \frac{1 - err^{(m)}}{err^{(m)}}.$$
   (e) Update the weight for each training sample:
   $$w_i \leftarrow w_i \cdot \exp\left(\beta^{(m)} \cdot \mathbb{I}\left(c_i \neq T^{(m)}(\boldsymbol{x}_i)\right)\right),$$
   for $i = 1,2,...,N$.
   (f) Re-normalize the sample weight distribution:
   $$w_i \leftarrow w_i / \sum_{i=1}^{N} w_i.$$
3. Output the classification predictions:
$$C(\boldsymbol{x}) = \arg\max_k \sum_{m=1}^{M} \beta^{(m)} \cdot \mathbb{I}\left(T^{(m)}(\boldsymbol{x}) = k\right)$$

Table 2.1: Algorithm summary of original AdaBoost.

Extensions of Original AdaBoost for Multi-Class Problems

As a result, many extensions of original AdaBoost to the multi-class classification problem have been designed, however, the weak classifiers are still required to have an accuracy

higher than 50%. One possible and popular approach is to transform the multi-class problem into several binary subproblems, which can be done by using one-against-all or one-against-one strategy [20].

- **One-against-all strategy for each class**:

  The one-against-all strategy constructs one model for each class, where each model is trained to separate the samples of its corresponding class from the samples of all remaining classes. When a new sample of unknown class is taken in, it will be assigned to the class whose model has the maximum output value among all. In other words, each predefined class has a probabilistic binary classifier to distinguish its kind from others, and each classifier will make a class prediction for an unknown sample with some probability for that class. The class prediction of the classifier that returns the highest probability will thus be chosen for this unknown sample.

- **One-against-one strategy for all pairs of classes**:

  On the other hand, the one-against-one strategy constructs one model for each pair of classes, so for a multi-category problem with K ($K > 2$) classes, $K(K-1)/2$ models in total are trained to divide the samples of one class from the samples of the other class in all pairs. When a new sample of unknown class is taken in, it will be sorted to the class with maximum voting, where each model votes for one class. This pairwise learning method may sound computational consuming, but in fact it is not, and if the classes are evenly distributed, it will be at least as fast as any other multi-class solution. The reason is that each pairwise subproblem only takes training samples of two classes into consideration, other than the whole training dataset. For example, if N samples are divided evenly among K classes, there will be $2N/K$ samples per subproblem. Suppose the runtime of a binary classification algorithm is proportional to the number of training samples it learns, then the total runtime will be proportional to $K(K-1)/2 \cdot 2N/K$, which is $(K-1)N$. That means, this method only scales linearly with the number of classes.

In a word, if the weak learners boosted by AdaBoost are inherently incapable of producing multi-class predictions, the above alternatives can be particularly useful.

Multi-Class AdaBoost

However, for this thesis work, an approach that handles multi-class cases directly without reducing them to multiple two-class problems will be used, known as multi-class AdaBoost. As mentioned before, AdaBoost is originally designed only for boosting two-class cases. If using one-against-all or one-against-one strategies, it can be extended to solve multi-class problems. However, this still requires the weak learners to have a classification rate higher than 50%, which is quite difficult for multi-class cases, where the number of classes

$K \geq 3$ and the probability for random guessing is $1/(K-1)$. As a result, the real multi-class AdaBoost algorithm, also called SAMME, is proposed in [21], which successfully avoids reducing the multi-class problems to two-class subproblems and only requires the classification rate of weak learners better than $1/(K-1)$.
In fact, SAMME algorithm is very similar to the original AdaBoost. It is also based on forward stagewise additive modeling using an exponential loss function. However, this time the exponential loss function has been modified into a multi-class version.
For multi-class (the number of classes $K \geq 3$) classification problem, SAMME encodes the class prediction (denoted by c_i) as $\boldsymbol{y}_i = (y_1,y_2,...,y_K)^T, i = 1,2,...,N$, with

$$y_k = \begin{cases} 1, & \text{if } c_i = k \\ -\frac{1}{K-1}, & \text{otherwise} \end{cases} \tag{2.37}$$

where $k = 1,2,...,K$.
Then if $f = (f_1,f_2,...,f_K)^T$ and $\sum_{k=1}^{K} f_k = 0$, the multi-class loss optimized by SAMME is

$$L(y,f) = \exp\left(-\frac{1}{K}\boldsymbol{y}^T f\right) \tag{2.38}$$

This time, the basis functions in the forward stagewise additive model become multi-class weak learners $T^{(m)} \in \mathcal{Y}$, where

$$\mathcal{Y} = \left\{ \begin{array}{c} (1, -\frac{1}{K-1},..., -\frac{1}{K-1})^T \\ (-\frac{1}{K-1},1,..., -\frac{1}{K-1})^T \\ \vdots \\ (-\frac{1}{K-1},..., -\frac{1}{K-1},1)^T \end{array} \right\} \tag{2.39}$$

Again, the problem becomes solving:

$$(\alpha^{(m)},T^{(m)}) = \arg\min_{\alpha,T} \sum_{i=1}^{N} \exp[-\frac{1}{K}\boldsymbol{y}_i^T(f^{(m-1)}(\boldsymbol{x}_i) + \alpha T(\boldsymbol{x}_i))] \tag{2.40}$$

for the weak learner $T^{(m)}$ and its corresponding weight coefficient $\alpha^{(m)}$ to be added at each step. This can expressed as:

$$(\alpha^{(m)},T^{(m)}) = \arg\min_{\alpha,T} \sum_{i=1}^{N} w_i^{(m)} \exp(-\frac{1}{K}\alpha\boldsymbol{y}_i^T T(\boldsymbol{x}_i)) \tag{2.41}$$

where

$$w_i^{(m)} = \exp(-\frac{1}{K}\boldsymbol{y}_i^T f^{(m-1)}(\boldsymbol{x}_i)). \tag{2.42}$$

Again, since $w_i^{(m)}$ is independent on α and $T(\boldsymbol{x}_i)$, it can be regarded as a weight factor that is applied to each training samples. This weight depends on $f^{(m-1)}(\boldsymbol{x}_i)$ so the weight changes during each iteration m.
In a similar way to the binary case, it can be obtained that

$$\begin{cases} \text{If } \boldsymbol{y}_i = T(\boldsymbol{x}_i), & \text{then } \boldsymbol{y}_i^T T(\boldsymbol{x}_i) = \frac{K}{K-1}; \\ \text{If } \boldsymbol{y}_i \neq T(\boldsymbol{x}_i), & \text{then } \boldsymbol{y}_i^T T(\boldsymbol{x}_i) = -\frac{K}{(K-1)^2}. \end{cases} \tag{2.43}$$

Therefore, the criterion in (2.41) can be expressed as

$$\exp\left(-\frac{\alpha}{K-1}\right) \cdot \sum_{\boldsymbol{y}_i = T(\boldsymbol{x}_i)} w_i^{(m)} + \exp\left(\frac{\alpha}{(K-1)^2}\right) \cdot \sum_{\boldsymbol{y}_i \neq T(\boldsymbol{x}_i)} w_i^{(m)}, \tag{2.44}$$

which in turn can be rewritten as

$$\left(e^{\frac{\alpha}{(K-1)^2}} - e^{-\frac{\alpha}{K-1}}\right) \cdot \sum_{i=1}^{N} w_i^{(m)} \mathbb{I}(\boldsymbol{y}_i \neq T(\boldsymbol{x}_i)) + e^{-\frac{\alpha}{K-1}} \cdot \sum_{i=1}^{N} w_i^{(m)} \tag{2.45}$$

Apply gradient descent method to (2.45) and solve for α, by taking partial derivative respect to α and set the resulting equation is to 0, one obtain α as

$$\alpha^{(m)} = \frac{(K-1)^2}{K}\left(\log\frac{1-err^{(m)}}{err^{(m)}} + \log(K-1)\right) \tag{2.46}$$

where $err^{(m)}$ is the minimized weighted error rate

$$err^{(m)} = \frac{\sum_{i=1}^{N} w_i^{(m)} \mathbb{I}(\boldsymbol{y}_i \neq T^{(m)}(\boldsymbol{x}_i))}{\sum_{i=1}^{N} w_i^{(m)}}. \tag{2.47}$$

As a result, the approximation of multi-class problem can be updated as

$$f^{(m)}(\boldsymbol{x}) = f^{(m-1)}(\boldsymbol{x}) + \alpha^{(m)} T^{(m)}(\boldsymbol{x}). \tag{2.48}$$

So the weight for the next iteration can be accordingly updated as

$$w_i^{(m+1)} = \exp(-\frac{1}{K}\boldsymbol{y}_i^T f^{(m)}(\boldsymbol{x}_i)) = w_i^{(m)} \cdot \exp(-\frac{1}{K}\alpha^{(m)} \boldsymbol{y}_i^T T^{(m)}(\boldsymbol{x}_i)). \tag{2.49}$$

This is equivalent to

$$\begin{cases} w_i^{(m)} \cdot \exp(\frac{K-1}{K}\beta^{(m)}), & \text{if } c_i = T^{(m)}(\boldsymbol{x}_i)); \\ w_i^{(m)} \cdot \exp(\frac{1}{K}\beta^{(m)}), & \text{if } c_i \neq T^{(m)}(\boldsymbol{x}_i)). \end{cases} \tag{2.50}$$

where

$$\beta^{(m)} = \log \frac{1 - err^{(m)}}{err^{(m)}} + \log(K - 1). \tag{2.51}$$

At this stage, the multi-class AdaBoost algorithm (SAMME) can be summarized in Table 2.2 [21].
One can easily notice the extra term $\log(K-1)$ in the update scheme for classifier. It has been shown that this term derives from the forward stagewise additive modeling which uses the multi-class exponential loss function. In additional, if $K = 2$, the algorithm returns to binary AdaBoost. Moreover, the updating of sample weights seems different from (2.50), but actually they are equal, since the one in algorithm is the normalized version.

2.2.2 Weak Learners for AdaBoost

A weak learner is defined to be a classifier which has a classification rate only slightly better chance, in other words, it can categorize samples better than random guessing.

Stumps

Stumps, also known as decision stump, is a classifier that makes use of one-level decision tree. It is one-level because it has one root which is directly connected to the terminal nodes. A decision stump produces a prediction based on a single rule with respect to the input feature value [22]. Moreover, each decision stump pays attention to only a single element of the input feature vector.
The type of stumps varies according to the different types of input features. For example, given nominal features, a stump consisting of terminal nodes for each possible feature value may be built, or in other cases, a stump can be created with two terminal nodes, one of which corresponding to one specified class, and the other leading to all the other classes. For binary features these two examples are actually the same. A missing value is treated as a legitimate value, which means it is considered as another class.
For continuous features, a threshold for the feature values can be selected. That is, the stump contains two terminal nodes, one for the values below the threshold and the other for the ones above the threshold. However, sometimes multiple thresholds may also be used and the stump therefore contains three or more nodes.
Decision stumps are often used as the weak learners in ensemble models such as bagging and boosting. For example, [11] employs AdaBoost with decision stumps as weak learners.

Perceptron

Perceptron, being considered the simplest type of artificial neural network, is a linear classifier which has very good performance as long as the samples being classified are

1. Initialize the weight for each training sample, $w_i = 1/N, i = 1,2,...,N$.
2. For $m = 1$ to M:
   (a) Fit a classifier $T^{(m)}(\boldsymbol{x})$ to the training dataset using weights w_i.
   (b) Compute the weighted training error rate for the classifier:
   $$err^{(m)} = \sum_{i=1}^{N} w_i \mathbb{I}\left(c_i \neq T^{(m)}(\boldsymbol{x}_i)\right) / \sum_{i=1}^{N} w_i.$$
   (c) If $err^{(m)} \leq 0$ or $err^{(m)} \geq fracK - 1K$, then abort loop.
   (d) Compute the weight for the classifier in the ensemble:
   $$\beta^{(m)} = \log \frac{1 - err^{(m)}}{err^{(m)}} + \log(K - 1).$$
   (e) Update the weight for each training sample:
   $$w_i \leftarrow w_i \cdot \exp\left(\beta^{(m)} \cdot \mathbb{I}\left(c_i \neq T^{(m)}(\boldsymbol{x}_i)\right)\right),$$
   for $i = 1,2,...,N$.
   (f) Re-normalize the sample weight distribution:
   $$w_i \leftarrow w_i / \sum_{i=1}^{N} w_i.$$
3. Output the classification predictions:
$$C(\boldsymbol{x}) = \arg\max_k \sum_{m=1}^{M} \beta^{(m)} \cdot \mathbb{I}\left(T^{(m)}(\boldsymbol{x}) = k\right)$$

Table 2.2: Algorithm summary of Multi-Class AdaBoost (SAMME).

linearly separable. However, if the samples are not linearly separable, Perceptron classifier becomes weak.

A simplest Perceptron can be seen as a training model for a single neuron, which is defined by a logic unit with a weight vector and a threshold (bias). These internal parameters map input data $\boldsymbol{x}$ to a single binary state output $f(\boldsymbol{x})$:

$$f(\boldsymbol{x}) = \begin{cases} +1 & \text{if } \boldsymbol{w}^T \cdot \boldsymbol{x} + b > 0 \\ -1 & \text{otherwise} \end{cases} \tag{2.52}$$

where $\boldsymbol{w}$ is the real-valued weight vector, $\boldsymbol{w}^T \cdot \boldsymbol{x}$ computes the weighted sum, and b is the bias.
With the above function, the problem of Perceptron algorithm reduces to finding the most suitable weights and bias so that the projection of sample $\boldsymbol{x}$ onto $\boldsymbol{w}$ has the same sign as its desired output.
Now consider a two-class ($K = 2$) classification problem, training samples $\{\boldsymbol{x}_i\}$ ($i =$ 1,2,...,N) with feature dimension D need to be distinguished by a decision hyperplane that divides the feature space into two sub-space. A linear discriminant function is constructed to fulfill the task:

$$f(\boldsymbol{x}) = \boldsymbol{w}^T \cdot \boldsymbol{x} + b = b + \sum_{d=1}^{D} w_d \cdot x_d \tag{2.53}$$

where $\boldsymbol{w}$ is the $D-$dimensional weight vector and b is the threshold or bias.
Let $\boldsymbol{x} = [1,x_1,...x_D]^T$ and $\boldsymbol{w} = [b,w_1,...w_D]^T$, Then the function can be rewritten as:

$$f(\boldsymbol{x}) = \boldsymbol{w}^T \cdot \boldsymbol{x}. \tag{2.54}$$

For this binary classification, there are two classes, $c_1 = +1$ and $c_2 = -1$. Given an input sample $\boldsymbol{x}_i$, if $\boldsymbol{w}^T \cdot \boldsymbol{x}_i > 0$, the linear classifier maps $\boldsymbol{x}_i$ to c_1, otherwise, $\boldsymbol{x}_i$ is assigned to c_2.
On the other hand:

$$\begin{aligned} &\text{if } \boldsymbol{x}_i \text{ belongs to } c_1, \qquad \boldsymbol{w}^T \cdot \boldsymbol{x}_i > 0. \\ &\text{if } \boldsymbol{x}_i \text{ belongs to } c_2, \qquad \boldsymbol{w}^T \cdot \boldsymbol{x}_i < 0 \text{ or } \boldsymbol{w}^T \cdot (-\boldsymbol{x}_i) > 0. \end{aligned} \tag{2.55}$$

Replacing $-\boldsymbol{x}_i$ by $\boldsymbol{x}_i$ if $\boldsymbol{x}_i$ belongs to c_2, then conclusion can be made:

$$\boldsymbol{w}^T \cdot \boldsymbol{x}_i > 0 \quad \text{if } \boldsymbol{x}_i\text{s are correctly classified.} \tag{2.56}$$

The Perceptron criterion function is defined as:

$$J(\boldsymbol{w}) = \sum_{\boldsymbol{x} \in \Psi(\boldsymbol{w})} -\boldsymbol{w}^T \cdot \boldsymbol{x}. \tag{2.57}$$

where $\Psi(\boldsymbol{w})$ represents the set of samples misclassified by $\boldsymbol{w}$. If all the samples are correctly classified, the criterion $J(\boldsymbol{w})$ would reach its minimum value 0.

Then the gradient descent method is applied to find out the updating scheme of $\boldsymbol{w}$, so partial derivative is taken with respect to $\boldsymbol{w}$:

$$\bigtriangledown J(\boldsymbol{w}) = \frac{\partial J(\boldsymbol{w})}{\partial \boldsymbol{w}} = -\boldsymbol{x}. \tag{2.58}$$

As a result, the argumented weight vector $\boldsymbol{w}$ is updated as follows:

$$\boldsymbol{w}_{m+1} = \boldsymbol{w}_m - \rho_m \cdot \bigtriangledown J(\boldsymbol{w}) = \boldsymbol{w}_m + \rho_m \cdot \boldsymbol{x}. \tag{2.59}$$

where ρ_m is a positive scaling factor (learning rate) usually set to 1 and m represents the iteration index.
Then the binary Perceptron learning algorithm is summarized in Table 2.3.

---

1. Initialize the weight vector $\boldsymbol{w}$ and the bias b of the neuron randomly.
2. For $m = 1$ to M:
   (a) Compute $i = m \mod N$.
   (b) Present a new sample $\boldsymbol{x}_i$ with its desired output c.
   (c) Update the weight vector and the bias as:

$$\boldsymbol{w}_{m+1} = \begin{cases} \boldsymbol{w}_m + \boldsymbol{x}_i & \text{if } \boldsymbol{w}_m^T \cdot \boldsymbol{x}_i + b \leq 0 \text{ and } c = +1, \\ \boldsymbol{w}_m - \boldsymbol{x}_i & \text{if } \boldsymbol{w}_m^T \cdot \boldsymbol{x}_i + b > 0 \text{ and } c = -1, \\ \boldsymbol{w}_m & \text{otherwise.} \end{cases}$$

$$b_{m+1} = \begin{cases} b_m + 1 & \text{if } \boldsymbol{w}_m^T \cdot \boldsymbol{x}_i + b \leq 0 \text{ and } c = +1, \\ b_m - 1 & \text{if } \boldsymbol{w}_m^T \cdot \boldsymbol{x}_i + b > 0 \text{ and } c = -1, \\ b_m & \text{otherwise.} \end{cases}$$

   (d) If $\sqrt{\|\boldsymbol{w}_{m+1} - \boldsymbol{w}_m\|^2 + (b_{m+1} - b_m)^2} < \theta$ (predefined criterion), then abort the loop and end the learning process.
3. Return the weight vector $\boldsymbol{w}$ and the bias b.

Table 2.3: Algorithm summary of binary Perceptron.

The Perceptron learning algorithm is proved to be convergent for linearly separable samples [23]. Since the classification hyperplane of the Perceptron is linear in nature, the standard Perceptron algorithm is not so capable to handle nonlinear classification problems. However, Perceptron still holds many important properties that can be exploited. First of all, Perceptron is easy to implement, the samples are presented one by one and the weights are updated only according to the current sample. Second, Perceptron is fast for its simplicity. The fast learning of Perceptron makes real-time and dynamic development of a system possible. Last but not least, Perceptron can be modified to fit for multi-class classification, which will be shown as follows.
Based on the Perceptron algorithm for binary cases, now consider a multi-class ($K \geq 3$) classification problem. Given input $D-$dimensional feature vectors with desired output $\{(\boldsymbol{x_i}, y_i), i = 1,2,...,N\}$ as the training samples, totally K linear discriminant functions need to be constructed to classify them into each corresponding categories. The combination of these linear discriminant functions is also called linear machine, which is defined as:

$$f_k(\boldsymbol{x}) = \boldsymbol{w}_k^T \cdot \boldsymbol{x} + b_k = b_k + \sum_{d=1}^{D} w_d^{(k)} \cdot x_d \quad k = 1,2,...,K. \tag{2.60}$$

where $\{\boldsymbol{w}_k\}$ are the $D-$dimensional weight vectors for each class.
Let $\boldsymbol{x} = [1,x_1,...x_D]^T$ and $\boldsymbol{w}_k = [b_k,w_1^{(k)},...w_D^{(k)}]^T$, Then the K linear discriminant functions can be rewritten as:

$$f_k(\boldsymbol{x}) = \boldsymbol{w}_k^T \cdot \boldsymbol{x}. \tag{2.61}$$

For multi-class classification, there are K classes $\{c_k\}$, where $k = 1,2,...K$. An input sample $\boldsymbol{x}$ is assigned to class c_k, if $f_k(\boldsymbol{x}) > f_j(\boldsymbol{x})$ for all j $(j \neq k)$.
So the problem here is to find K solution weight vectors $\{\boldsymbol{w}_k\}$, where $k = 1,2,...K$, to minimize the training error rate, which for linearly separable case can reach 0. In other words, the problem becomes finding $\boldsymbol{w}_k$ such that $(\boldsymbol{w}_k - \boldsymbol{w}_j)^T \cdot \boldsymbol{x} > 0$, for all j $(j \neq k)$, and for all $\boldsymbol{x}$ that belongs to class c_k.
Similarly to the binary case, the criterion function is defined as:

$$J(\boldsymbol{w}_1,\boldsymbol{w}_2,...,\boldsymbol{w}_k,...) = \sum_{k=1}^{K} \sum_{\boldsymbol{x} \in \Psi_k(\boldsymbol{w}_k)} \left(-(\boldsymbol{w}_k - \boldsymbol{w}_j)^T \cdot \boldsymbol{x}\right) \tag{2.62}$$

where $\Psi_k(\boldsymbol{w}_k)$ is the set of samples which belong to class c_k but are misclassified to class c_j, for any j $(j \neq k)$.
Rewrite the criterion:

$$J(\boldsymbol{w}_1,\boldsymbol{w}_2,...,\boldsymbol{w}_k,...) = \sum_{k=1}^{K} J_k \tag{2.63}$$

where

$$J_k = \sum_{\boldsymbol{x} \in \Psi_k(\boldsymbol{w}_k)} \left(-(\boldsymbol{w}_k - \boldsymbol{w}_j)^T \cdot \boldsymbol{x}\right). \tag{2.64}$$

In fact, each time only one misclassified sample is considered, so only $\boldsymbol{w}_k$ and $\boldsymbol{w}_k$ will be involved ($j \neq k$).
Therefore, when applying gradient descent method, only the partial derivatives of J_k to $\boldsymbol{w}_k$ and $\boldsymbol{w}_k$ are respectively taken:

$$\frac{\partial J_k}{\partial \boldsymbol{w}_k} = -\boldsymbol{x} \quad \text{and} \quad \frac{\partial J_k}{\partial \boldsymbol{w}_j} = \boldsymbol{x}, \quad \boldsymbol{x} \in \Psi_k(\boldsymbol{w}_k) \tag{2.65}$$

As a result, the argumented weight vectors $\{\boldsymbol{w}_k\}$ are updated as follows:

$$\begin{cases} \boldsymbol{w}_k^{m+1} = \boldsymbol{w}_k^m + \rho_m \cdot \boldsymbol{x} \\ \boldsymbol{w}_j^{m+1} = \boldsymbol{w}_j^m - \rho_m \cdot \boldsymbol{x} \end{cases} \tag{2.66}$$

where ρ_m is a positive scaling factor (learning rate) usually set to 1 and m represents the iteration index.
As a result, the multi-class Perceptron algorithm that is implemented in this thesis work can be summarized in Table 2.4.
The convergence property of multi-class Perceptron algorithm can be proved in the same way as the binary case for linearly separable problems [23].

2.3 Fusion Techniques

In pattern classification systems, image fusion aims to combine difference sources of information for intelligent tasks. Commonly, image fusion strategies are often categorized by pixel level fusion, feature level fusion, and decision level fusion.

- **Pixel level fusion**

  Pixel level fusion, also called low level fusion or measurement level fusion, where the fusion is done on the raw data. In this scheme, the raw images obtained from different sensors are fused together to produce a new image, which is supposed to have more complete information. Therefore, pixel level fusion can help human observers or computers more easily identify potential targets.

  As the name suggests, operations in this case are mostly done at pixel level, in either spatial or frequency domain. Popular pixel level fusion techniques include pyramid-decomposition-based fusion and wavelet-transform-based fusion.

1. Initialize the weight vectors $\{\boldsymbol{w}_k\}$ and the bias $\{b_k\}$ of the neurons randomly, where $k = 1,2,...,K$.

2. For $m = 1$ to M:

   (a) Compute $i = m \mod N$.

   (b) Present a new sample $\boldsymbol{x}_i$ with its desired output y_i.

   (c) If $\boldsymbol{x}_i$ is misclassified from its original class c_k to the wrong class c_j, then update the involved weight vectors and the biases as:

$$\begin{cases} \boldsymbol{w}_k^{(m+1)} = \boldsymbol{w}_k^{(m)} + \boldsymbol{x}_i \\ \boldsymbol{w}_j^{(m+1)} = \boldsymbol{w}_j^{(m)} - \boldsymbol{x}_i \end{cases}$$

$$\begin{cases} b_k^{(m+1)} = b_k^{(m)} + 1 \\ b_j^{(m+1)} = b_j^{(m)} - 1 \end{cases}$$

   (d) If $\sqrt{\|\boldsymbol{w}_k^{(m+1)} - \boldsymbol{w}_k^{(m)}\|^2 + (b_k^{(m+1)} - b_k^{(m)})^2} < \theta$ (predefined criterion), for all $k, k = 1,2,...,K$, then abort the loop and end the learning process.

3. Output the weight vectors $\{\boldsymbol{w}_k\}$ and the bias $\{b_k\}$, where $k = 1,2,...,K$.

Table 2.4: Algorithm summary of multi-class Perceptron.

- **Feature level fusion**

  Feature level fusion is also called medium level fusion or high level fusion. The basic idea is to compute features from images of each separate sensor, and then combine these features in a joint feature space.

- **Decision level fusion**

  Decision level fusion, also known as high level fusion and sometimes top level fusion. Approaches of this kind of fusion include voting methods, statistical modeling, methods based on fuzzy logic theory and theory of incomplete knowledge, etc.

2.3.1 Feature Level Fusion

Because of the simplicity, feature level fusion is very commonly used and is also chosen for this thesis work. The feature fusion scheme typically achieves boosted system performance or robustness, which attracts much attention in many research fields especially for computer vision and pattern classification. Although the importance of feature level fusion is obvious, there are very few techniques that can manipulate this idea in generalized ways. In most cases, the existing techniques are still designed to solve each specific problems in a specialized manner. As a result, multiple feature fusion remains an open issue.

The advantage of feature level fusion is obvious. Different feature vectors extracted from the same object can reflect characteristics of the object from different aspects, that is why improved reliability and enhanced capability is expected from feature level fusion, since redundant information and complementary information are provided at the same time.

There are mainly two schemes in feature level fusion, which are serial feature fusion and parallel feature fusion. Serial feature fusion combines two or more sets of feature vectors into one union-vector and then extract the final features from the high-dimensional real feature space. On the other hand, parallel feature fusion groups two sets of feature vectors to jointly create a complex feature space and then extract the final features from the high-dimensional complex feature space. For simplicity, serial feature fusion is adopted for the thesis work.

For either serial scheme or parallel scheme, the basic kind of feature level fusion directly concatenates or integrates several types of feature together, which is also the conventional approach. The advantage of this method is obvious, it is simple and can be easily implemented. However, the systems that applies such fusion scheme may not result in better or more robust performance than using single features, sometimes even worse. This is probably because the information held by different features is not equally represented or measured, in other words, the values of different features may be significantly unbalanced. Therefore, concatenating different features with equal weights can be quite suboptimal, since in many cases some features will dominate the entire feature ensemble.

The simplest way to weight the features is to normalize the ranges or scales of each feature so that they are well balanced before integration. For example, the normalized features could have zero mean and unit variance [24], or they could also have equal sum of eigenvalues on eigenvector-based features [25]. Though this is one possible solution to unbalance feature fusion problem, it should still be noted that in most cases each kind of feature will be extracted from the same dataset, then there will be a great chance that these features are highly correlated to each other. Thus, the simple normalization or weighting scheme would not be of much help to make the fused feature effective for classification purpose. However, in this thesis work, each set of feature is independently extracted from a different data source, i.e. HOG features from the visual band camera and Gabor features from the IR thermal sensor, so the correlation problem is well avoided

and this normalization scheme is still worth trying.
Last but not least, another possible way for the feature weighting is to perform joint dimensionality reduction or subspace learning by preserving the correlation between different features.

Chapter 3

Classification of Object Poses

3.1 The Big Picture

Generally, the task of this thesis work is to design a classification system that is able to analyze and classify different types, activities and events of tracked object, e.g. different poses of human faces and moving cars. Such multi-class object classification models can be used to fit in various applications such as surveillance in public places and traffic safety on the road. For these purposes, this chapter investigates into multi-class object classification systems using a visual band camera or / and an infrared camera as the sensors to recognize different poses of objects. First, each type of images are used individually, and then the possibilities of feature level fusion from visual and infrared images are discussed.

Working under the assumption that the objects have been perfectly tracked with tight bounding boxes, the systems receive the images of the tracked object, analyze the data and classify them into each pose classes. The main stages in this whole process is described as well as some implementation details.

3.2 Classification Using Visual Images

3.2.1 Block Diagram

The cascades of the classification system using only visual images are shown in Figure 3.1. From left to right: The histogram of oriented gradients (HOG) features of visible light images are obtained, then the feature dimension is reduced by Principal Component Analysis (PCA), and after that the features are put into AdaBoost classifier for object pose classification.

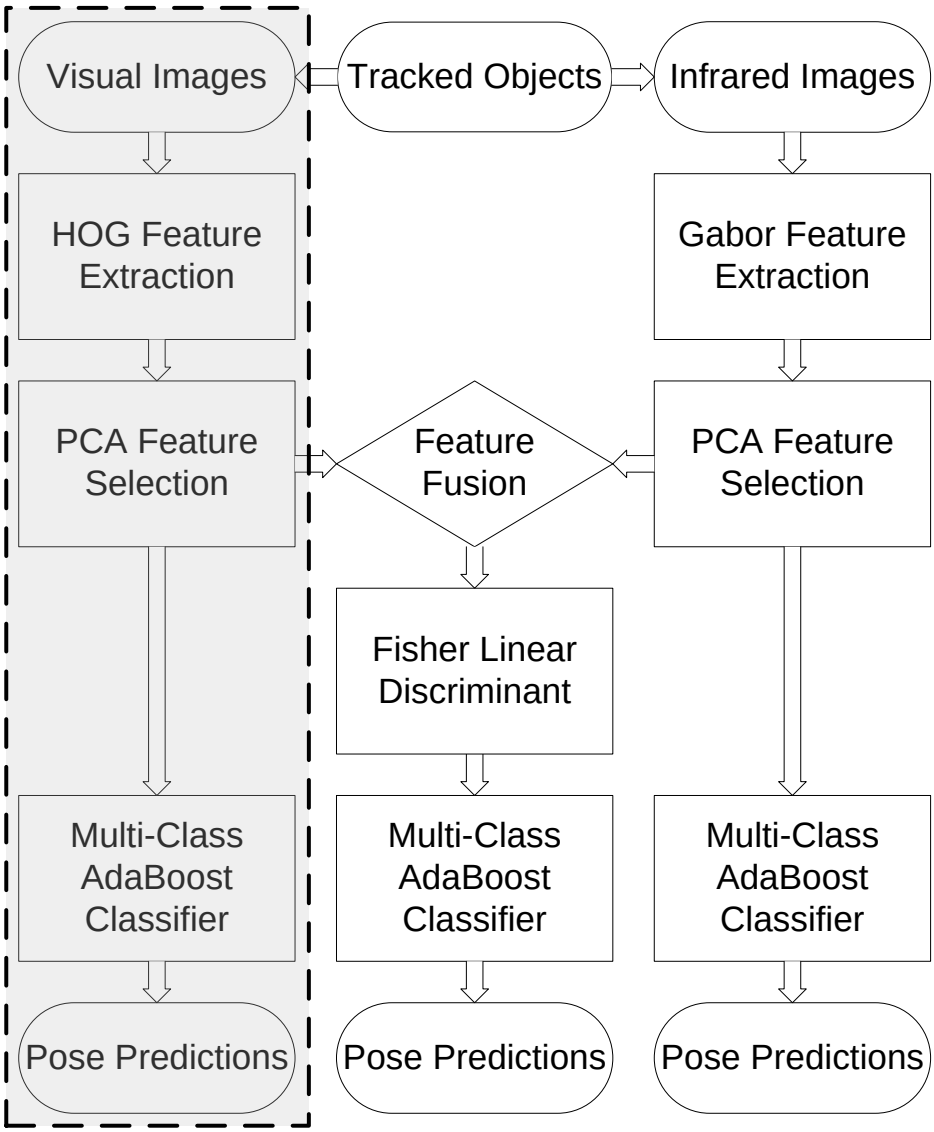


Figure 3.1: Block diagram for classification using visual images.

3.2.2 HOG Features for Visual Images

Given enough visible light, Visual Images in most cases provide sharp and strong edges of objects. As we know, edges give dominant gradient values in the image, so gradient based features HOG are especially suitable as the feature descriptor of visual images. Although HOG is initially intended for pedestrian detection in images and videos, later it is extended to many other applications and has achieved great success as well. Therefore, this popular feature descriptor has been chosen to represent the object poses in visual images for this

thesis work. The HOG algorithm implemented in this thesis work computes the R-HOG feature descriptor of the input image, which is based on [10].
First of all, the input image is loaded into the system, without any gamma correction. Then the input image is resized to 32×32 and converted from RGB color space to grayscale, as shown in Figure 3.2. Also, it is converted into double precision and normalized to the dynamic range of [0,1] for computation convenience.

(a) Original image. **(b)** Image after pre-processing.

Figure 3.2: Image pre-processing for HOG feature extraction.

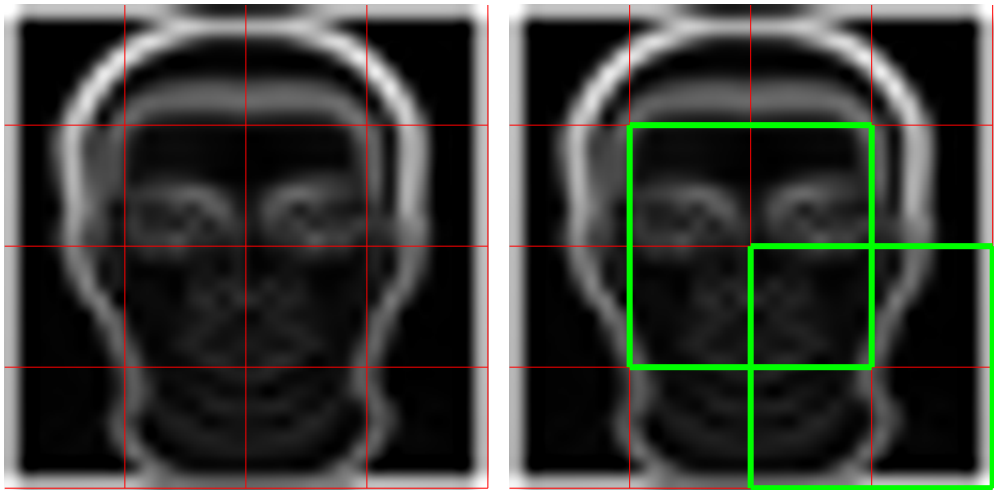
At the second stage, first order image gradients are computed by convolving the image with mask [+1,0,−1] along x and y-axes with no smoothing, where G_x and G_y are obtained as the gradient matrices in each direction, as shown in Figure 3.3a and 3.3b. Then the gradient magnitudes (Figure 3.3c) are calculated by taking the norm of G_x and G_y, i.e. $G_{mag} = \sqrt{G_x^2 + G_y^2}$, which capture contour, silhouette and some texture information of the object and also provide resistance to illumination variations. The resultant gradient magnitude image is displayed in Figure 3.2, where the object edges are detected because edges give dominant gradient values. After that, the orientation at each pixel is computed by taking the inverse tangent function, i.e. $\arctan(G_y/G_x) + \pi/2$. It should be noted that $\pi/2$ is necessary here since $\arctan(G_y/G_x)$ ranges between $-\pi/2$ and $\pi/2$ and only the case of unsigned orientation is considered which ranges from 0 to π.
Then the image window is divided into a dense uniformly sampled grid of points. For each point, the square pixel image region centered on it which is called block is split into cells, as Figure 3.4 illustrates, and the block steps in both x and y directions are 8 pixels, which means the overlapping rate is set as 50% so that each cell can contribute more than once to block histograms.
For each cell, a 1D histogram of gradient or edge orientations over all the pixels in the

(a) Gradient image along x-axis.

(b) Gradient image along y-axis.

(c) Gradient magnitude by taking the norm of gradient images along x and y-axes.

Figure 3.3: Edge detected by computing gradient magnitudes.



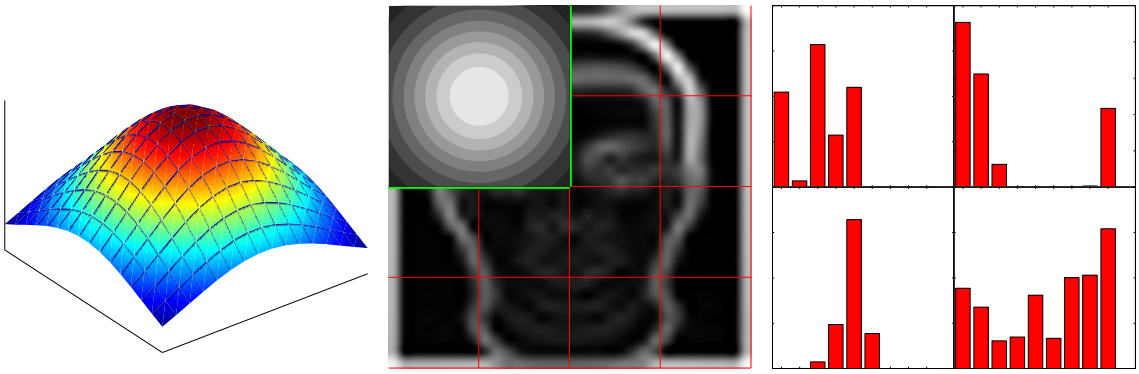
(a) Image divided into cells of size 8×8.

(b) Cells grouped into blocks with 50% block overlapping rate.

Figure 3.4: Image pre-processing for HOG feature extraction.

cell is accumulated. This combined cell-level 1D histogram divides the gradient angle range into a fixed number of predetermined bins. For this thesis work, the range is set from 0° to 180° with a step size of 20°, so this makes 9 bins per histogram. The gradient magnitudes of the pixels in the cell are then used to vote into the orientation histogram, in the meanwhile, a Gaussian spatial window with $\sigma = 8$ pixels (half the block side length), as shown in Figure 3.5a and 3.5b, is applied to each block, so that the voting weight of each pixel around the edge of the block can be significantly suppressed. In this way, for

each block, a vector of length 36 will be obtained (the number of cells in a block times the number of bins per histogram), as Figure 3.5c shows.



(a) Gaussian spatial window.

(b) Applying Gaussian spatial window to each block.

(c) Histograms formed by weighted voting.

Figure 3.5: Orientation binning by weighted voting within each block.

This process is repeated for each block, and the resulting vector is normalized before being collected into a large HOG feature vector in sequence. As the block window slides to the right bottom end of the image, the feature vector containing all the block HOG is then completed. In this thesis work, images are all resized to 32×32 so the totally number of blocks with 50% overlapping rate is 9, so the final HOG feature vector in this implementation has a dimension of 324 (9×36).

3.2.3 PCA for HOG Feature Selection

If the dimensions of feature vectors are too high, it may cause the effect of the curse of dimensionality. That is, the running time of classification is considerably long, and overfitting problem will occur where the trained classifier will have extremely poor performance when handling new samples. Therefore, to avoid the effect of the curse of dimensionality, covariance matrix is computed from the feature vectors, and PCA algorithm is used to select a subset of principal components to obtain lower dimensional features, which keeps the most relevant feature information and discards irrelevant information and noise.
The basic idea of PCA is to seek the most accurate feature representation in a lower dimensional space, by projecting the data to the directions of largest variance. As we know, the directions of largest variance is given by eigenvectors corresponding to the largest eigenvalues of covariance. That is why PCA makes use of covariance matrix of the data. By selecting a subset of the eigenvectors as basis vectors for projection, the dimensions of the data can be thus significantly reduced.
In this implementation, cumulative energy content is chosen as the criterion in selecting the subset of eigenvectors. After obtaining the eigenvectors and the corresponding

eigenvalues from the covariance matrix of the HOG features, the eigenvalues are sorted in ascending order. By adding the eigenvalues one by one to the cumulative sum and dividing it by the total sum of eigenvalues, one can get a subset of eigenvectors according to the eigenvalues that have already been added which eventually make the ratio above some predefined threshold. For HOG features extracted visual images in this thesis work, 95% cumulative energy content is kept. As a result, the HOG feature after PCA feature selection has been reduced from 324 to 218. Then, the feature vectors are put into AdaBoost classifier for classification results.

3.3 Classification Using Infrared Images

3.3.1 Block Diagram

The cascades of the classification system using only infrared images are shown in Figure 3.6. From left to right: The Gabor features of infrared images are obtained, then the feature dimension is reduced by Principal Component Analysis (PCA), and after that the features are put into AdaBoost classifier for object pose classification.

3.3.2 Gabor Features for Infrared Images

For extracting the feature of infrared thermal images, Gabor filters are used. The HOG feature descriptor used for visual images has been replaced because the edges in infrared thermal images are kind of blurred, so image energies are mainly concentrated at lower frequency bands other than high frequencies. In this case, gradient based edge detector will not be so suitable, so instead wavelet based edge detector Gabor features are exploited. In this way, each point is represented by local Gabor filter responses. As mentioned before, a 2D Gabor filter can be obtained by modulating a 2D sinusoidal plane wave at particular frequencies and orientations with a Gaussian envelope. Actually there are many expressions of Gabor wavelets and in this thesis work the notation follows [16]. Thus, the 2D Gabor filter kernel is defined as:

$$g(x,y,\theta_k,f) = \exp(-\frac{1}{2}\left(\frac{\hat{x}^2}{\sigma_x^2} + \frac{\hat{y}^2}{\sigma_y^2}\right)) \exp(j2\pi f\hat{x}) \tag{3.1}$$

where

$$\begin{cases} \hat{x} = x\cos\theta + y\sin\theta \\ \hat{y} = -x\sin\theta + y\cos\theta \end{cases} \tag{3.2}$$

σ_x and σ_y are the standard deviations of the Gaussian envelope along the x and y-axes, respectively. θ_k and f are the orientation and the frequency, respectively. The rotation of the plane by an angle θ_k results in a Gabor filter at orientation θ_k, which is defined by

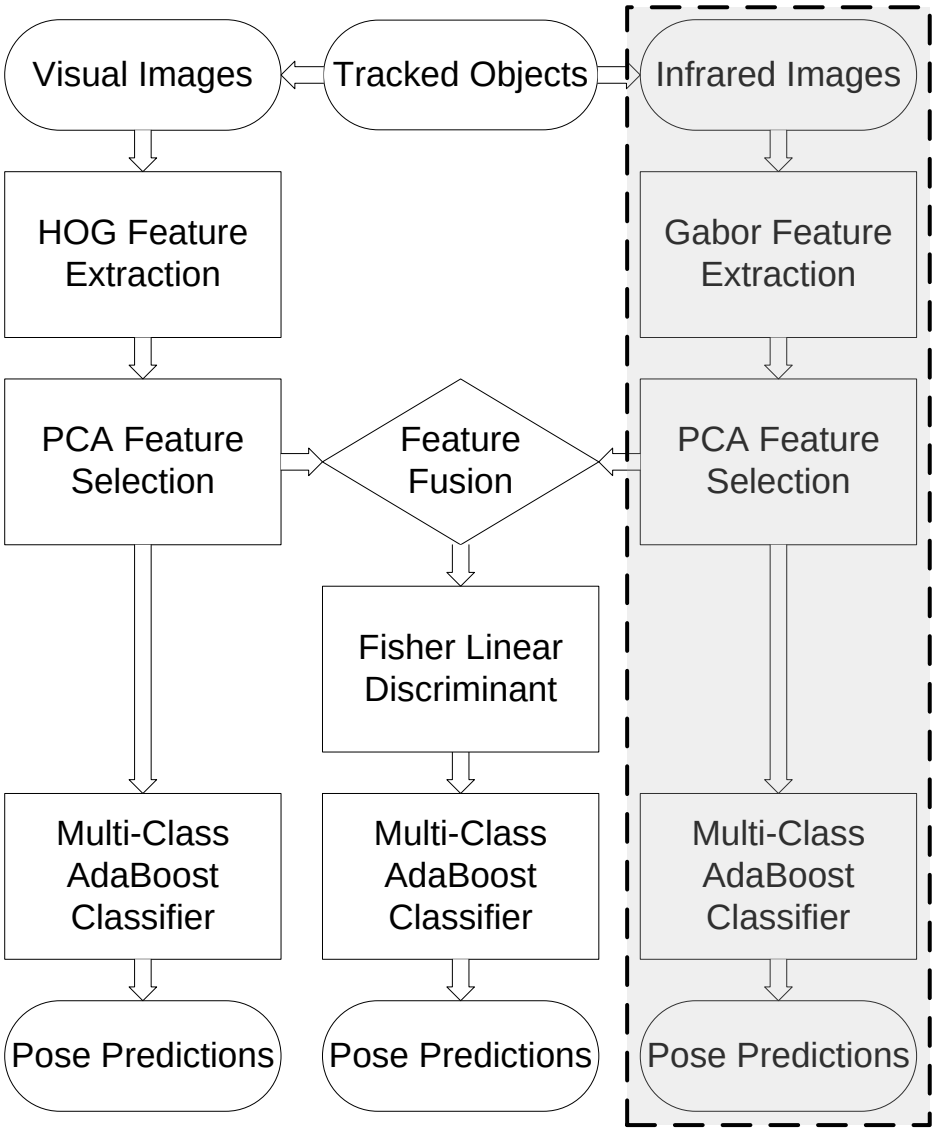
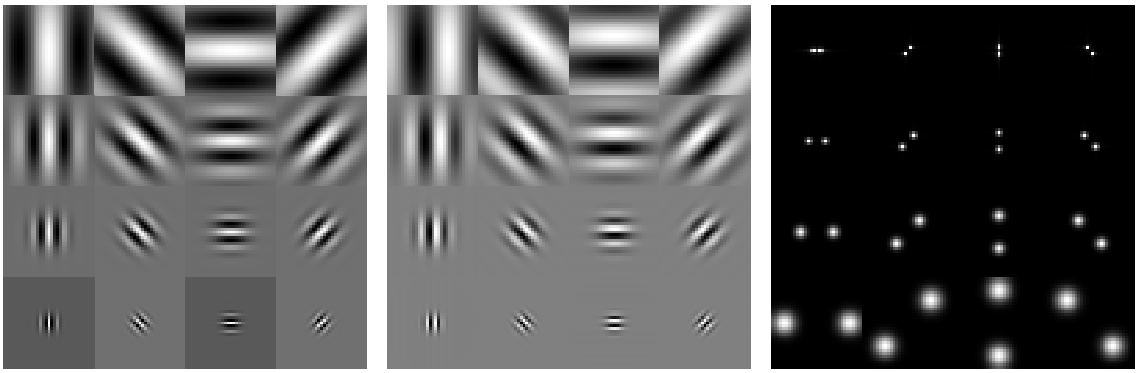


Figure 3.6: Block diagram for classification using infrared images.

$$\theta_k = \frac{\pi}{n}(k-1) \quad k = 1,2,...,n \tag{3.3}$$

where n denotes the number of orientations.
The Gabor feature at a point (x,y) of the IR thermal image can be viewed as the response of all different Gabor filters located at that point. A filter response is obtained by convolving the filter kernel of specific θ_k and f with the image. If the frequency f and the orientations θ_k are properly chosen, a bank of Gabor filters which covers the entire

frequency domain can be obtained, as shown in Figure 3.7 and 3.8.

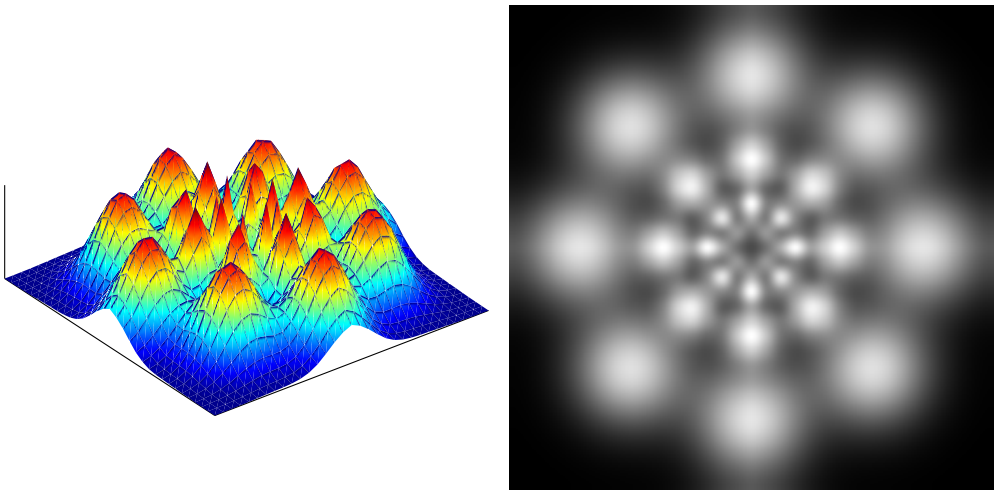


(a) Real part of Gabor filters in spatial domain.

(b) Imaginary part of Gabor filters in spatial domain.

(c) Gabor filters in the frequency domain.

Figure 3.7: A bank of Gabor filters. For each column of the sub-figures from left to right: $\theta = 0(\pi), \pi/4(5\pi/4), \pi/2(6\pi/4), 3\pi/4(7\pi/4)$, for each row of from top to bottom, $\sigma_x = \sigma_y = 16, 8, 4, 2$, where $f = 1/\sigma_x$.



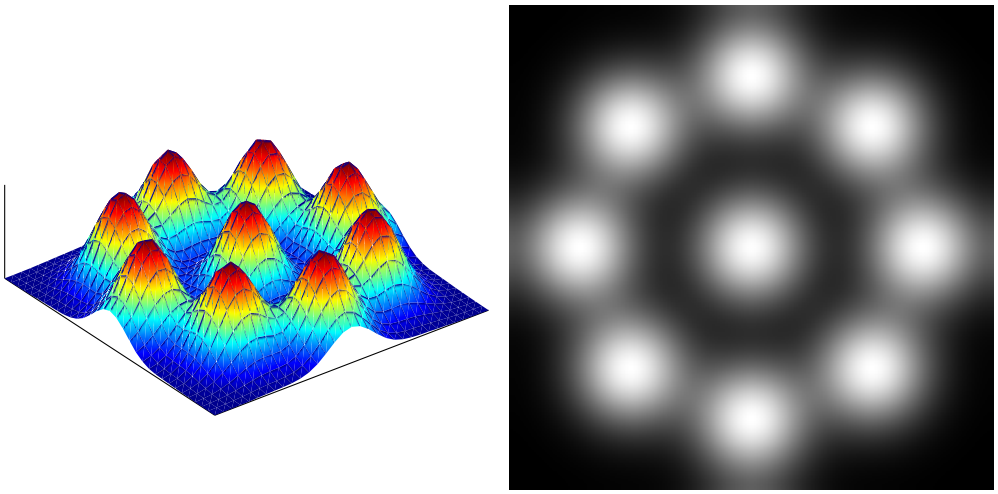
(a) 3D view.

(b) Corresponding 2D view.

Figure 3.8: A bank of Gabor filters in the frequency domain. For each layer, $\theta = 0(\pi), \pi/4(5\pi/4), \pi/2(6\pi/4), 3\pi/4(7\pi/4)$, for each circle from outer layer to inner layer, $\sigma_x = \sigma_y = 2, 4, 8, 16$, where $f = 1/\sigma_x$.

However, in this thesis work, Gabor kernels with 8 orientations and only 1 scales/frequency band is used. Also, unlike conventional Gabor filters, DC component is kept where $f = 0$. The basic idea here is that for infrared thermal images, energy is mainly concentrated at

lower frequency bands and DC part. So there are 9 Gabor filters in the bank, as shown in Figure 3.9.



(a) 3D view. **(b)** Corresponding 2D view.

Figure 3.9: Gabor kernels with 8 orientations in 1 frequency and a DC component. For the outer layer, $\sigma_x = \sigma_y = 2$ and $f = 1/(\sqrt{2} \cdot \sigma_x)$.

The reason for using 1 frequency band other than more frequency layers is simple. The number of Gabor filters in the bank has been considerably reduced and still similar performance remains.
The Gabor representation of an image is computed by convolving the image with the Gabor filters. Let $f(x,y)$ be the intensity at the coordinate (x,y) in a grayscale image, its convolution with a Gabor filter $g(x,y)$ is defined as

$$h(x,y) = f(x,y) * \Re\{g(x,y)\}. \tag{3.4}$$

The output of each filter are down-sampled on a 8×8 uniform grid before being reshaped into 1D vectors and these vectors are each normalized by L2-norm

$$\hat{v} = \frac{v}{\sqrt{||v||_2^2 + eps^2}}, \tag{3.5}$$

for enhanced robustness against lighting conditions, where *eps* is a small constant. After that, these vector are concatenated into the final Gabor feature vector.
For an IR image of size 32×32, the dimension of resultant feature vector will be 576 (8×8×9), which is even higher than that of HOG features for visual images.

3.3.3 PCA for Gabor Feature Selection

Similarly, for computation efficiency and avoiding the curse of dimensionality, PCA algorithm is applied to reduce the Gabor feature dimensions before they are put into AdaBoost classifier. In this case, 85% cumulative energy content is kept, which results in a lower feature dimension of 208.

3.4 Feature Fusion for Improved Classification

3.4.1 Block Diagram

The cascades of the classification system applying feature level fusion are shown in Figure 3.10. From top to bottom: The HOG features of visible image and the Gabor features of its corresponding infra-red thermal image are obtained respectively, and their feature dimensions are reduced by PCA individually. Then these two features are fused by Fisher Linear Discriminant, after that the feature ensembles are detected by AdaBoost classifier. The only difference in this mode when compared to the previous two classification modes is the feature fusion part. Therefore, the feature extractions and selections by using HOG features for visual images and using Gabor features for infrared images and PCA algorithm are omitted in this case.

3.4.2 Fisher Linear Discriminant for Feature Fusion

Let $X_1 \in \mathbb{R}^{N_1}$ and $X_2 \in \mathbb{R}^{N_2}$ represent the HOG and Gabor feature of an image of object pose, where N_1 and N_2 are the dimensionality of the HOG and Gabor feature spaces, respectively. The covariance matrices of X_1 and X_2 are $C_1 \in \mathbb{R}^{N_1 \times N_1}$ and $C_2 \in \mathbb{R}^{N_2 \times N_2}$, and their eigenvector matrices, $V_1 \in \mathbb{R}^{N_1 \times N_1}$ and $V_2 \in \mathbb{R}^{N_2 \times N_2}$, respectively. To improve the generalization performance of classification, one should choose only a subset of principal components to derive the lower dimensional HOG and Gabor features, so that $Y_1 \in \mathbb{R}^{m_1}$ and $Y_2 \in \mathbb{R}^{m_2}$, where m_1 and m_2 are the dimensions of the reduced HOG feature space and Gabor feature space, respectively. In fact, using more principal components may decrease the performance for object recognition. The reason for this is that the trailing eigenvalues correspond to high frequency components and mainly contain noise. Therefore, when these trailing but small valued eigenvalues are used to define the reduced PCA space, the subsequent Fisher linear discriminant procedure has to fit for noise as well and as a consequence overfitting will occur.
Then the issue becomes how to determine the dimensions for the reduced HOG feature space (m_1) and the reduced Gabor feature space (m_2), respectively. On one hand, as much relevant information of the original features as possible need to be kept after transformation from the high dimensional space to the low dimensional space. On the other hand, the small trailing eigenvalues of the within-class covariance in the reduced feature space

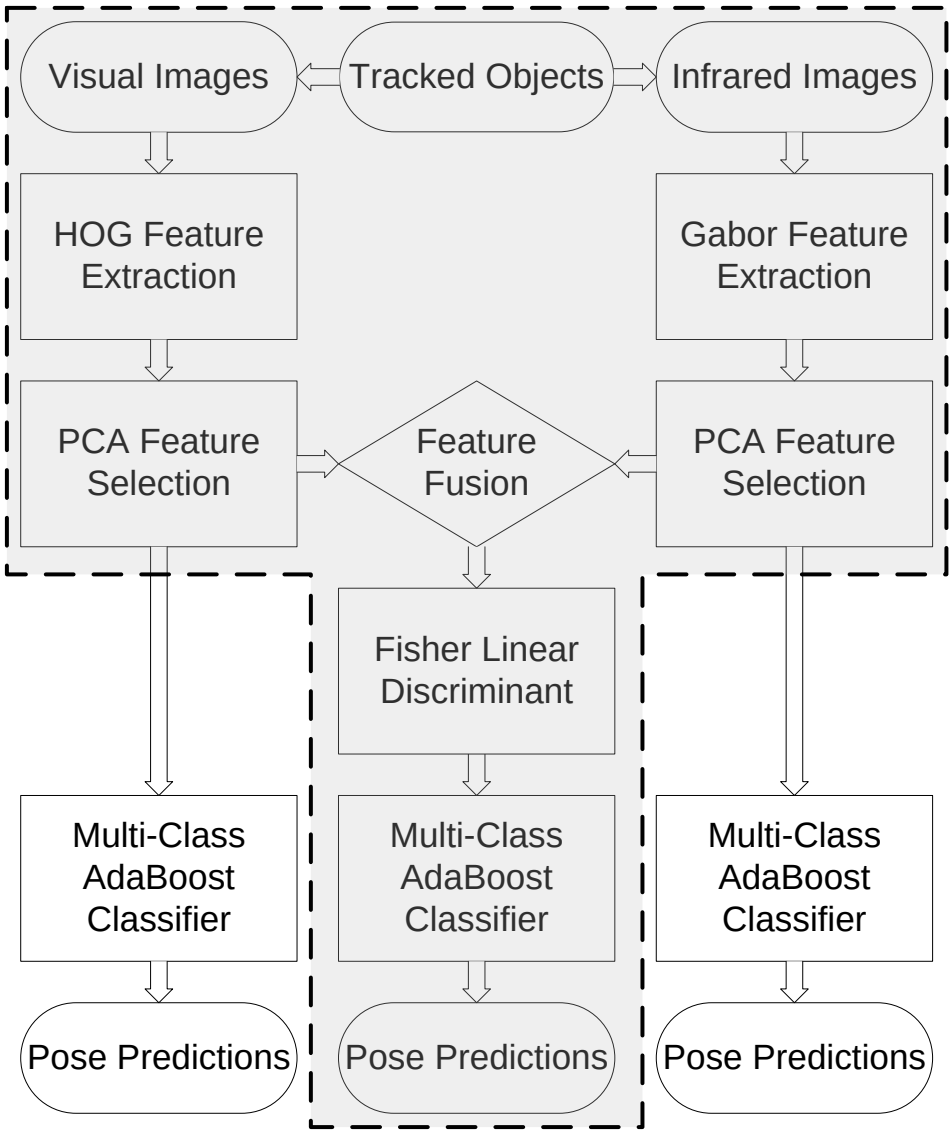


Figure 3.10: Block diagram for classification applying feature level fusion.

should be eliminated. In this way, more robust projection of Fisher linear discriminant can be obtained.

In this thesis work, 95% cumulative energy is kept for HOG features and 85% cumulative energy is kept for Gabor features during PCA feature selection, so that the resulting feature dimensions $m_1 > m_2$. The aim here is to give HOG features some more dimensional weights in the subsequence union-feature space.

The low dimensional features are then combined together in a union-vector which now contains both visual band and infrared band information. Before integration, they both

are simply normalized to have unit norms, under the assumption that they are equally important discriminating information. So no weight factors are assigned to the feature components.

$$Z = \left[\frac{Y_1^T}{\|Y_1\|_1} \quad \frac{Y_2^T}{\|Y_2\|_1} \right]^T \tag{3.6}$$

where $Z \in \mathbb{R}^{m_1+m_2}$, and m_1 and m_2 are the dimensions of the reduced HOG and Gabor feature spaces, respectively.
Fisher linear discriminant (FLD) is a commonly used discriminant criterion which measures the between-class scatter normalized by the within-class scatter. Let w_1,w_2,...,w_K and n_1,n_2,...,n_K denote the classes and the number of samples within each class. Let μ_1,μ_2,...,μ_K be the means of each class and μ be the total mean of all classes. The within-class and between-class covariance matrices C_w and C_b can be defined as follows:

$$C_w = \sum_{k=1}^{K} P(w_k)\mathbb{E}[(Z-\mu_k)(Z-\mu_k)^T|w_k] \tag{3.7}$$

$$C_b = \sum_{k=1}^{K} P(w_k)(\mu_k-\mu)(\mu_k-\mu)^T \tag{3.8}$$

where $P(w_k) = n_k / \sum_{k=1}^{K} n_k$ is a prior probability for k-th class.
FLD can find a projection matrix V that maximizes the ratio $|V^T C_b V|/|V_T C_w V|$. This ratio is maximized when V contains the eigenvectors of the matrix $C_w^{-1}C_b$, that is,

$$C_w^{-1}C_b V = VD \tag{3.9}$$

where V and D are the eigenvector and eigenvalues matrices of $C_w^{-1}C_b$. Thus, FLD has built the most discriminating feature space from Z for subsequent classification, and the final fused features F can be derived as

$$F = V^T Z. \tag{3.10}$$

At this stage, the fused features can be put into AdaBoost classifier for classification, from which better performance is expected than using single feature type from only visual images or infrared images.

Chapter 4

Experimental Results

Classifying human faces and moving cars of different poses from video or images is an important real-world application for many purposes. As a result, in the experiments, the implemented classification system is demonstrated mainly by classifying face poses and car poses. This chapter firstly presents the datasets built or collected during this thesis work, then it provides experimental results on these datasets, and lastly some evaluations are made as well.

4.1 Datasets

There are mainly 4 datasets used in the experiments. For simplicity, these datasets are called Dataset-1, Dataset-2, Dataset-3 and Dataset-4. Before looking into the details of these datasets, synthetic data for initial tests are introduced.

4.1.1 Synthetic Data

Synthetic data is generated to verify the implementation of pose classification system. They are geometric graphics in 3 classes, rectangle, ellipse and triangle. Each class contains 200 images, which vary in size, rotation angle, transformation and position in the 32×32 image, etc. The synthetic data can be seen in Figure 4.1.

4.1.2 Dataset-1

This dataset is a mixture of many ready-made databases of human face provided by some research institutes. They are MIT-CBCL Face Recognition Database [26], CVL Face Database [27], Stanford Medical Student Face Database [28], GTAV Face Database [29], FEI Face Database [30]. These face images were captured mostly in indoor environment with uniform background that clearly shows the object edges.

Figure 4.1: Synthetic data containing geometric graphics in 3 different classes. For each column from left to right: rectangle, ellipse and triangle.

The combined visual band face dataset, Dataset-1, contains 771 frontal face images, 593 left views and 469 right views of different persons, so there are 3 classes of face poses. Face images are automatically aligned, cropped out from the selected images and resized to 32×32. Figure 4.2 gives some sample images from this dataset.

4.1.3 Dataset-2

This dataset gathers a large number of car images downloaded from Internet. It contains 515 front, 451 left, 595 right, and 993 back views of various kinds of cars. These car images were mostly captured in outdoor environment, where the backgrounds and lighting conditions vary a lot. Car images are manually aligned, cropped out from the selected images. After that, they are automatically resized to be 48×32. Some preview of Dataset-2 is provided in Figure 4.3.

4.1.4 Dataset-3

This dataset is a part from OTCBVS [31], which contains simultaneously acquired thermal and visible face images under variable illuminations, expressions and poses. The original image size is 320×240 for both thermal and visible images. However, during the experiments, only the infrared thermal face images of 3 poses are used. Therefore, 350 front, 443 left and 383 right face images in infrared band only are selected from the original

Figure 4.2: Dataset-1 containing visual images of face poses in 3 different classes (front, left and right).

database. These images are automatically resize to 32×24, without cropping. Figure 4.4 shows some sample images of Dataset-3.

Figure 4.3: Dataset-2 consisting of visual images of car poses in 4 different classes (front, left, right and rear).

4.1.5 Dataset-4

This dataset is built up during this thesis work, by using the Fluke Ti45 infra-red thermal imager. The Fluke camera consists of two sensors, one captures visible light images and the other produces IR thermal images of the same scene, which is especially useful for the topic, that is to fuse the features extracted from both the visual band image and the IR thermal image of objects to perform robust multi-class classification. The capture work lasted for about two weeks, for which over 500 persons at Chalmers are involved, including many teaching staffs and students at all levels.

This visual / infrared face dataset, which is called Dataset-4 here, contains face images of 5 poses at both visual and IR bands, where 506 front, 500 left, 500 right, 456 upward and 460 downward views are collected. These face pose images were captured in many different environments, e.g. group and computer rooms, cafes and corridors in E-Building, A-Building, M-Building, Library, Student Union, etc. and the streets on campus, so the backgrounds and lighting conditions vary significantly. All the images are manually aligned, cropped with tight bounding boxes and then automatically resized to 32×32.

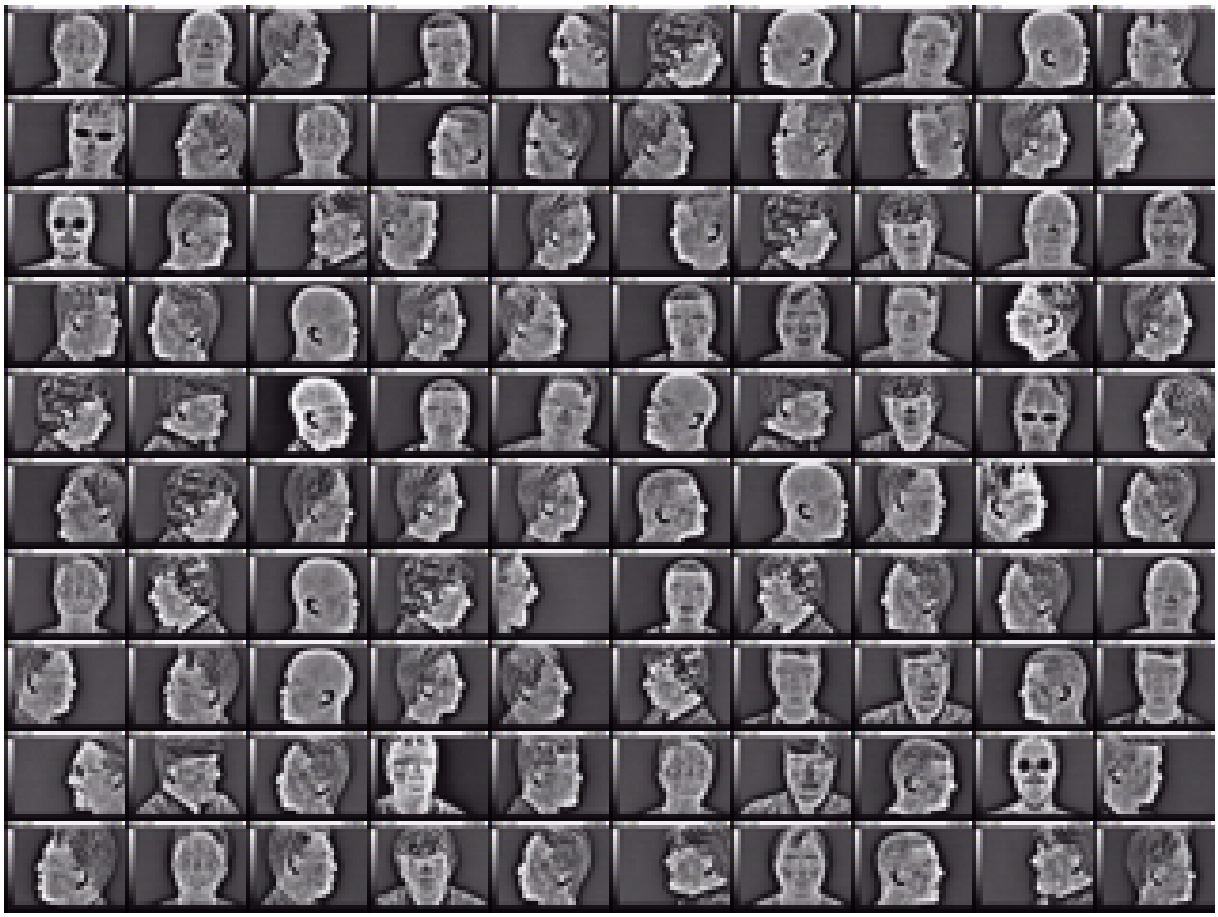


Figure 4.4: Dataset-3 containing infrared thermal images of face poses in 3 different classes (front, left and right).

Each sample of the visual band images and their corresponding IR thermal images are given in Figure 4.5.

4.2 Results and Discussions

All the experiments are divided into two phases, the training procedure and the testing part. For each dataset, 20% of the samples are kept only for testing purpose. The remaining 80% of the samples are used to perform 10-fold cross validation, where the remaining samples are randomly partitioned into 10 subsets. Of the 10 subsets, a single subset is retained as the validation data for validating the classification model, and the remaining 9 subsets are used as the training data. The cross validation process is repeated 10 times, with each of the 10 subsets used exactly once as the validation data. The 10

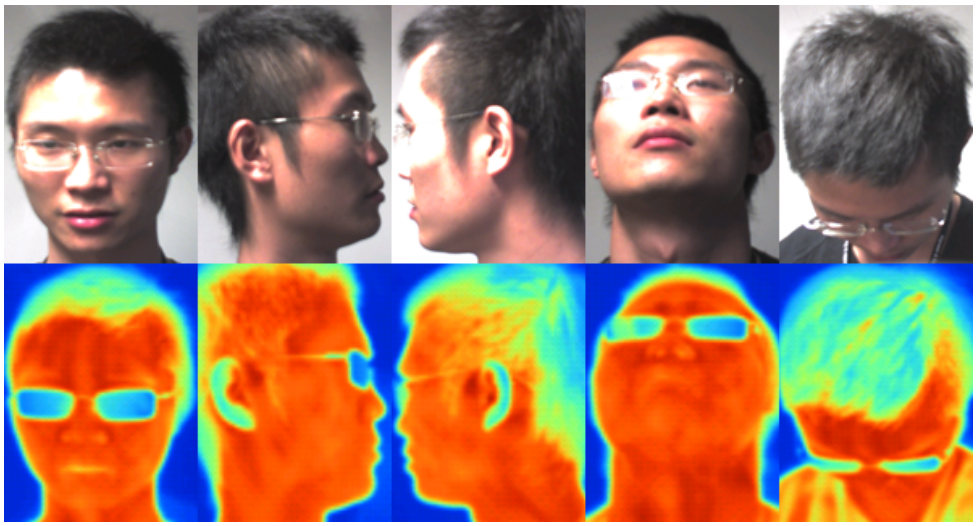


Figure 4.5: Dataset-4 consisting of visual and infrared images of face poses in 5 different classes (front, left, right, upward and downward).

results from the folds then can be averaged to produce a single estimation.
The classifications are generally aimed at achieving the best possible accuracy in prediction of the object class. The idea is that the best prediction has the lowest misclassification rate, which is measured in terms of proportion of misclassified cases.
Let R_{miss} denote the misclassification rate, N_0 denote the number of misclassified samples in one dataset, and N denote the total number of samples being classified. Then, the misclassification rate can be defined as

$$R_{miss} = \frac{N_0}{N}. \tag{4.1}$$

Accordingly, the classification rate R_c can be obtained by

$$R_c = 1 - R_{miss}. \tag{4.2}$$

Type-I and type-II errors are also worth investigating in the experiments, since they provide some more details of the classification performance. Type-I error, also called false positive, occurs if a sample does not belongs to class 1, but a classification prediction wrongly states that the sample belongs to class 1. Type-II error, also known as false negative, happens when a sample indeed belongs to class 1, but a classification result wrongly shows that the sample belongs to a class other than class 1.
So, let N_{01} be the number of samples wrongly classified to the given class, and N_{02} be the number of samples that belongs to the given class but are misclassified to another class. Thus, the type-I error rate and type-II error rate for a given class are defined as follows

$$\alpha = \frac{N_{01}}{N_k} \quad \text{and} \quad \beta = \frac{N_{02}}{N_k}, \tag{4.3}$$

where N_k is the totally number of sample within the given class. In fact, $1-\alpha$ and $1-\beta$ give the specificity and sensitivity of the experiment for that class, respectively.

4.2.1 Test on Synthetic Data

First of all, synthetic data is used to mainly verify the implementation of HOG feature extraction and multi-class classifiers used in this thesis work, namely, AdaBoost and its weak learner Perceptron. There are totally 3 classes of synthetic images created, which are rectangle, ellipse and triangle. These geometric objects vary in size, rotation angle, transformation and position in the image window, etc.

The datasets used for experiments are described in Table 4.1. Every class of these geometric objects has 200 samples, so there are totally 600 samples, in which 20% (120 samples) are kept for testing purpose and the other 80% (480 samples) are used as the training set and the validation set.

Dataset	#Samples	Class#
Rectangular	200	1
Ellipse	200	2
Triangle	200	3

Table 4.1: Synthetic data of each class used for verification experiments.

The experimental results for classifying synthetic data are given in Table 4.2 and 4.3.

Dataset	#Samples	Classification rate (%)
Training set	432	100
Validation set	48	100
Testing set	120	100

Table 4.2: Classification rates for synthetic images using HOG features.

Class#	Type-I training error (%)	Type-II training error (%)	Type-I valid. error (%)	Type-II valid. error (%)	Type-I testing error (%)	Type-II testing error (%)
1	0.00	0.00	0.00	0.00	0.00	0.00
2	0.00	0.00	0.00	0.00	0.00	0.00
3	0.00	0.00	0.00	0.00	0.00	0.00

Table 4.3: Type I and II errors for each class in experiment on synthetic data.

As a result, it is verified that the HOG feature extraction and the AdaBoost classifier work in good condition.

4.2.2 Test on Dataset-1

In this experiment, visual band face images of 3 poses (front, left and right) are considered. The purpose of this experiment is to first try simple cases of classification of object poses, which is a 3-class problem here. The datasets used are described in Table 4.4, in which 20% (366 samples) are kept for testing purpose and the other 80% (1467 samples) are used as the training set and the validation set.

Dataset	#Samples	Class#
Visual face front	771	1
Visual face left	593	2
Visual face right	469	3

Table 4.4: Visual face images of 3 poses in Dataset-1.

The experimental results for classifying 3-pose visual face images are given in Table 4.5 and 4.6.

Dataset	#Samples	Classification rate (%)
Training set	1320	100
Validation set	147	98.33
Testing set	366	95.73

Table 4.5: Classification rates on Dataset-1 using HOG features.

Class#	Type-I training error (%)	Type-II training error (%)	Type-I valid. error (%)	Type-II valid. error (%)	Type-I testing error (%)	Type-II testing error (%)
1	0.00	0.00	1.33	0.33	3.07	0.67
2	0.00	0.00	0.33	0.67	0.67	3.33
3	0.00	0.00	0.00	0.67	0.53	0.27

Table 4.6: Type I and II errors in experiment on Dataset-1.

It can observed that, the classification model using HOG features for visual images performs well in this 3-class problem, since the classification rate of testing set has reached as high as 95.73%. Based on the result, the classification can be thus extended to problems with more pose classes.

4.2.3 Test on Dataset-2

Classifying car poses is also a topic that attracts much interests. In this experiment, visual band car images of 4 poses are considered, which are front, left, right and rear. So far, no infrared thermal car images have been collected to make the sense of fusion as comparison, therefore the result from this part may probably be used in future works.

The datasets used are described in Table 4.7, in which 20% (510 samples) are kept for testing purpose and the other 80% (2053 samples) are used as the training set and the validation set.

Dataset	#Samples	Class#
Visual car front	515	1
Visual car left	451	2
Visual car right	595	3
Visual car rear	993	4

Table 4.7: Visual car images of 4 poses in Dataset-2.

The experimental results for classifying 4-pose visual car images are given in Table 4.8 and 4.9.

Dataset	#Samples	Classification rate (%)
Training set	1849	99.86
Validation set	204	94.66
Testing set	510	94.18

Table 4.8: Classification rates on Dataset-2 using HOG features.

Class#	Type-I training error (%)	Type-II training error (%)	Type-I valid. error (%)	Type-II valid. error (%)	Type-I testing error (%)	Type-II testing error (%)
1	0.04	0.09	2.40	2.25	2.94	1.43
2	0.03	0.02	0.34	0.34	0.55	1.10
3	0.00	0.00	0.44	0.69	1.14	0.55
4	0.06	0.03	2.16	2.06	1.20	2.75

Table 4.9: Type I and II errors in experiment on Dataset-2.

As illustrated in Table 4.8 and 4.9, the experiment on visual car images has obtained comparable results with that of 3-pose visual face images. This means the classification model using visual images is not limited to classifying face poses only, thus the applications can be extended in the future.

4.2.4 Test on Dataset-3

The major purpose of this experiment is to verify the Gabor feature extraction. Also, parameters for face representation from infrared images using Gabor wavelets are determined. In this case, face images in infrared thermal band are used, which also have 3 pose classes, i.e. front, left and right. The datasets used are described in Table 4.10, in which 20% (235 samples) are kept for testing purpose and the other 80% (941 samples) are used as the training set and the validation set.

Dataset	#Samples	Class#
Infrared face front	350	1
Infrared face left	443	2
Infrared face right	383	3

Table 4.10: Thermal face images of 3 poses in Dataset-3.

The experiments are done with different parameter combinations, e.g. 4 frequency bands without DC component, 1 frequency band with or without DC component, etc. Optimal parameter setting is found that using 1 center frequency at $\frac{1}{2\sqrt{2}}$ with DC component gives the overall best performance. So, the corresponding experimental results for classifying 3-pose thermal face images are given in Table 4.11 and 4.12.

Dataset	#Samples	Classification rate (%)
Training set	847	86.66
Validation set	94	84.04
Testing set	235	83.40

Table 4.11: Classification rates on Dataset-3 using Gabor features.

Class#	Type-I training error (%)	Type-II training error (%)	Type-I valid. error (%)	Type-II valid. error (%)	Type-I testing error (%)	Type-II testing error (%)
1	2.60	5.79	1.06	8.51	2.13	7.23
2	5.67	3.78	3.19	7.45	4.68	8.09
3	5.08	3.78	11.70	0.00	9.79	1.28

Table 4.12: Type I and II errors in experiment on Dataset-3.

The results in this experiment seem not so good, but the reason is obvious, the images are not cropped so many irrelevant background information has been encoded as well. With a tight bounding box of the object, the performance will be improved, which can be shown in the next experiment.

4.2.5 Test on Dataset-4

As a final stage of the experiments, the dataset built up during this thesis work is used. Since it contains the same object poses in both visual and infrared band, comparisons can be made between classifications using visual images only, using infrared images only and applying feature level fusion from both image types.

Classification Using Visual Images Only

This part of work is similar to the previous experiment on visual face images, the only difference is that this time the face poses have been extended to 5 classes, with two more categories in upward and downward views. The datasets used are described in Table 4.13, in which 20% (484 samples) are kept for testing purpose and the other 80% (1938 samples) are used as the training set and the validation set.

Dataset	#Samples	Class#
Visual face front	506	1
Visual face left	500	2
Visual face right	500	3
Visual face upward	456	4
Visual face downward	460	5

Table 4.13: Visual face images of 5 poses in Dataset-4.

The experimental results for classifying 5-pose visual face images are given in Table 4.14 and 4.15.

Dataset	#Samples	Classification rate (%)
Training set	1744	99.47
Validation set	194	92.23
Testing set	484	94.01

Table 4.14: Classification rates on Dataset-4 (visual) using HOG features.

Class#	Type-I training error (%)	Type-II training error (%)	Type-I valid. error (%)	Type-II valid. error (%)	Type-I testing error (%)	Type-II testing error (%)
1	0.13	0.34	2.64	2.18	3.33	1.07
2	0.01	0.04	0.62	1.14	0.45	0.68
3	0.01	0.05	0.62	1.04	0.33	0.62
4	0.13	0.09	1.19	1.71	0.50	1.83
5	0.27	0.04	2.69	1.71	1.38	1.80

Table 4.15: Type I and II errors in experiment on Dataset-4 (visual) using HOG features.

Classification Using Infrared Images Only

This part of work is similar to the previous experiment on infrared thermal face images, the major differences are that this time the face poses have been extended to 5 classes, with two more categories in upward and downward views, and the images have been cropped with tight bounding boxes. The datasets used are described in Table 4.16, in which 20% (484 samples) are kept for testing purpose and the other 80% (1938 samples) are used as the training set and the validation set.

Dataset	#Samples	Class#
Infrared face front	506	1
Infrared face left	500	2
Infrared face right	500	3
Infrared face upward	456	4
Infrared face downward	460	5

Table 4.16: Infrared face images of 5 poses in Dataset-4.

The experimental results for classifying 5-pose infrared face images are given in Table 4.17 and 4.18.

Dataset	#Samples	Classification rate (%)
Training set	1744	96.05
Validation set	194	89.64
Testing set	484	87.19

Table 4.17: Classification rates on Dataset-4 (infrared) using Gabor features.

Class#	Type-I training error (%)	Type-II training error (%)	Type-I valid. error (%)	Type-II valid. error (%)	Type-I testing error (%)	Type-II testing error (%)
1	2.06	1.83	3.42	3.11	3.31	4.96
2	0.06	0.11	1.14	1.55	1.24	1.65
3	0.00	0.06	0.52	1.04	0.62	0.83
4	1.38	0.86	3.63	2.59	3.72	3.10
5	0.63	1.26	1.55	2.07	3.93	2.27

Table 4.18: Type I and II errors in experiment on Dataset-4 (infrared) using Gabor features.

This time, one can easily observe that with tight bounding boxes, the classification performance using infrared images has been improved, even with more pose classes.

Classification Applying Feature Fusion

As the final stage of the thesis work, HOG features extracted from visual face images and Gabor features derived from infrared face images are fused by the scheme discussed in the previous chapter. In brief, their feature elements are firstly selected by PCA to keep the principal components and reduce the feature dimensions, respectively. In this thesis work, the resultant dimensions of HOG feature vectors are higher than that of Gabor features, which means HOG features have gain more weight in the fusion scheme. After that, they are each normalized to have unit norm and combined together into a union-vector. This union-vector is processed by Fisher linear discriminant to produce the final fused features for classification.

The datasets used are the same as described in Table 4.13 and 4.16. The experimental results for classifying 5-pose face images using feature level fusion are given in Table 4.19 and 4.20.

Dataset	#Samples	Classification rate (%)
Training set	1744	100
Validation set	194	95.34
Testing set	484	97.11

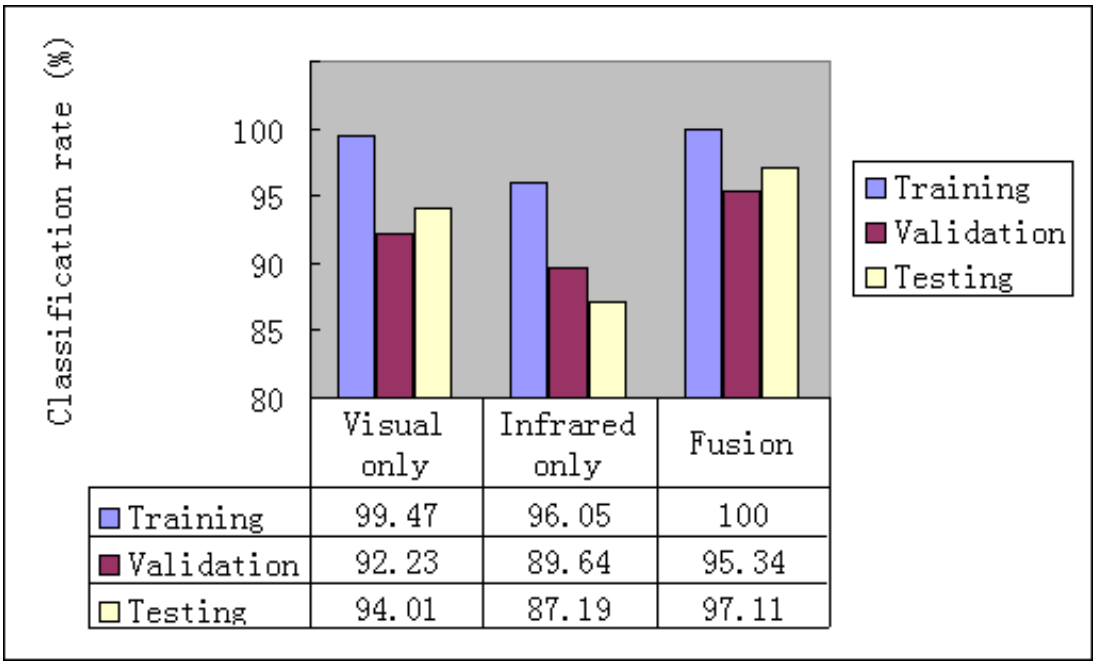
Table 4.19: Classification rates on Dataset-4 using feature fusion.

Class#	Type-I training error (%)	Type-II training error (%)	Type-I valid. error (%)	Type-II valid. error (%)	Type-I testing error (%)	Type-II testing error (%)
1	0.00	0.00	2.07	1.55	1.86	0.62
2	0.00	0.00	0.00	0.52	0.00	0.41
3	0.00	0.00	0.52	0.52	0.21	0.21
4	0.00	0.00	1.55	1.55	0.00	0.83
5	0.00	0.00	0.52	0.52	0.83	0.83

Table 4.20: Type I and II errors in experiment on Dataset-4 using feature fusion.

4.3 Evaluations

From the experimental results, one can see that in test on Dataset-4, classification applying feature level fusion has achieved best performance, when compared to using single features for any type of images in Dataset-4. Classification using HOG features for visual images in Dataset-4 follows up and the classification using Gabor features for infrared images in Dataset-4 falls behind. This can be easily observed in Figure 4.6.
To note that the classification rate for testing set of Dataset-4 has achieved 97.11% when using feature fusion, which is more than 3% higher than that of using HOG features for visual images. The reason behind this is quite simple, by fusion at feature level, reliability and robustness has been improved since HOG features from visual band and Gabor features from infrared band have provided redundant and complementary information for each other.
Classification on infrared images using Gabor feature has reached lowest accuracy, which is probably because the parameters for building the bank of Gabor filters are still not



	Visual only	Infrared only	Fusion
Training	99.47	96.05	100
Validation	92.23	89.64	95.34
Testing	94.01	87.19	97.11

Figure 4.6: Classification rates on Dataset-4 in bar chart for comparison.

optimally selected. In fact, the parameters assigned in this implementation are obtained by choosing the best one from a limited number of experiments. As a result, this empirical approach may not effectively derive the optimal parameter settings, so some analytical way could be used instead for improved performance in the infrared band. This may also lead to further boosted results on classification in feature fusion mode.
Also, it should be noticed that, when compared the experimental results on Dataset-3 and the single Gabor features on Dataset-4, one can see that with or without tight bounding box for the objects in the images has made quite a difference. That is, the classification rates on Dataset-4 (infrared only) is over 5% higher in average than that of Dataset-3. To conclude, tight bounding boxes has eliminated irrelevant background information and unwanted noise to a great extent.

Chapter 5

Conclusions

This thesis has made use of two types of images: one is from the visual band and another is from the thermal infrared band. Due to different properties and characteristics in these two types of images, different types of features are extracted. For visual band images, HOG is used as the main feature descriptor. For thermal IR images, Gabor features are used.

For classification of object poses, classifiers are tested separately by using visual band images only, and thermal IR images only. Performance is then evaluated for these two types of classifiers. Attempts are also made on classifiers through fusing these two types of features in visual and IR images.

Experimental results have shown that the feature extraction and classification models in this thesis work provide robust performance. In particular, when using feature level fusion of HOG features from visual images and Gabor features from infrared images, the classification of human face in 5 poses in this thesis work achieves best result, which is followed by classification using HOG features alone of visible light images, and Gabor features alone of infrared thermal images.

Suggestions of Future Work

For future work, how to enhance system performance against partial occlusion should be considered. In this thesis work, for simplicity, it is assumed that all the objects in the images are fully displayed and no partial occlusion happens. However, when facing practical issues, partial occlusion can not be avoided. If only part of the object in the image is available for processing, HOG features and Gabor features used in this thesis may not be so efficient in feature representation since they are both based on the global information of the entire image. As a result, some key point or region of interest (ROI) based features can be exploited, e.g. SIFT, to better describe the objects that are being analyzed and classified, especially when partial occlusion occurs.

Then, based on the robustness and effectiveness of classifying object poses in images, the system can be extended to analyze and classify objects in videos or image sequences.
Besides, it has been shown in this thesis work that tight bounding box around the object can sufficiently eliminate the irrelevant background information and unwanted noise for improved classification. However, during this thesis, most work on cropping the images has been done manually, which is extremely time consuming and inefficient, especially when handling large datasets. Therefore, an efficient and automatic way to crop the objects with tight bounding boxes from images should be found.
Moreover, other level of fusion can also be tried, e.g. pixel level fusion and decision level fusion. For decision level, the classification system built in this thesis can produce multiple predictions in parallel, so it is natural to exploit decision level fusion for enhanced performance. Possible approaches may make use of confidence level of each classifier, or simply produce the final decision on majority vote. The possibility of multi-level fusion can also be discussed.
Lastly, how to develop a faster and an on-line training algorithm can be investigated, since off-line training is used in this thesis work and the running time is extremely long.

References

[1] Rodehorst, V. and Koschan, A.: *Comparison and Evaluation of Feature Point Detectors*, In Grün*d*ig and Altan (Eds.), Proceedings of 5^{th} Turkish-German Joint Geodetic Days, Berlin, 8 p, 2006

[2] Lindeberg, T.: *Feature Detection with Automatic Scale Selection*, International Journal of Computer Vision, 30(2):79-116, 1998

[3] Lowe, D. G.: *Distinctive Image Features from Scale-Invariant Keypoints*, International Journal of Computer Vision, 60(2):91-110, 2004

[4] Mikolajczyk, K. and Schmid, C.: *Scale and Affine Invariant Interest Point Detectors*, International Journal of Computer Vision, 60(1):63-86, 2004

[5] Lowe, D. G.: *Object Recognition from Local Scale-Invariant Features*, In Proceedings of the 7^{th} International Conference on Computer Vision, Kerkyra, Greece, pages 1150-1157, 1999

[6] Sirovitch, L. and Kirby, M.: *Low-Dimensional Procedure for the Characterization of Human Faces*, Journal of the Optical Society of America, 2:586-591, 1987

[7] Turk, M. and Pentland, A.: *Face Recognition using Eigenfaces*, In Proceedings of the Conference on Computer Vision and Pattern Recognition, Maui, Hawaii, USA, pages 586-591, 1991

[8] Rowley, H. A., Baluja, S. and Kanade, T.: *Neural Networks Based Face Detection*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 20(1):22-38, 1998

[9] Dadal, N. and Triggs, B.: *Histogram of Oriented Gradients for Human Detection*, In Proceedings of IEEE Conference Computer Vision and Pattern Recognition, San Diego, USA, pages 886-893, June 2005

[10] Dadal, N. and Triggs, B.: *Finding People in Images and Videos*, PhD thesis, French National Institute for Research in Computer Science and Control (INRIA), July 2006

[11] Viola and Jones: *Rapid Object Detection Using Boosted Cascade of Simple Features*, Computer Vision and Pattern Recognition, 2001

[12] Papageorgiou, Oren and Poggio: *A General Framework for Object Detection*, International Conference on Computer Vision, 1998

[13] Crow, F.: *Summed-Area Tables for Texture Mapping*, In Proceedings of SIGGRAPH, 18(3):207-212, 1984

[14] Lienhart, R. and Maydt, J.: *An Extended Set of Haar-like Features for Rapid Object Detection*, ICIP02, pp. I: 900-903, 2002

[15] Messom, C. H. and Barczak, A. L. C.: *Fast and Efficient Rotated Haar-like Features Using Rotated Integral Images*, Australian Conference on Robotics and Automation (ACRA2006), pp. 1-6, 2006

[16] Lee, T. S.: *Image Representation Using 2D Gabor Wavelets*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 18, No. 10, October 1996

[17] Bhuiyan, A. and Liu, C. H.: *On Face Recognition using Gabor Filters*, World Academy of Science, Engineering and Technology 28, 2007

[18] Freund, Y. and Schapire, R. E.: *A Short Introduction to Boosting*, Journal of Japanese Society for Artificial Intelligence, 14(5): 771-780, September 1999

[19] Matas, J. and Šochman, J.: *AdaBoost*, Center for Machine Perception, Czech Technical University, Prague

[20] Rodriguez, M.: *Multi-class Boosting*, Notes on AdaBoost algorithms, Department of Computer Science, University of California, Santa Cruz, December 2009

[21] Zhu, J., Rosset, S., Zou, H., and Hastie, T.: *Multi-Class AdaBoost*, Statistics and its Interface, 2, pages 349-360, 2009

[22] Iba, W. and Langley, P.: *Induction of One-Level Decision Trees*, Proceedings of the 9^{th} International Conference on Machine Learning, 1992

[23] Duda, R. O., Hart, P. E. and Stork, D. G.: *Pattern Classification, 2nd Edition*, Wileys, John & Sons, Incorporated, New York, pages 275-279, December 1999

[24] Zhou, X. and Bhanu, B.: *Feature Fusion of Face and Gait for Human Recognition at a Distance in Video*, In IEEE Conference on ICPR, pages 529-532, Washington, D.C., USA, 2006

[25] Wang, X. and Tang, X.: *Using Random Subspace to Combine Multiple Features for Face Recognition*, In IEEE Conference on FGR, pages 284-289, 2004

[26] Weyrauch, B., Huang, J. and Blanz, V.: *Component-based Face Recognition with 3D Morphable Models*, First IEEE Workshop on Face Processing in Video, Washington, D.C., 2004

[27] Solina, F., Peer, P., Batagelj, B., Juvan, S. and Kovac, J.: *Color-based Face Detection in the '15 Seconds of Fame' Art Installation*, In: Mirage 2003, Conference on Computer Vision / Computer Graphics Collaboration for Model-based Imaging, Rendering, image Analysis and Graphical special Effects, March 10-11 2003, INRIA Rocquencourt, France, Wilfried Philips, Rocquencourt, INRIA, 2003, pp. 38-47

[28] Diaco, A., DiCarlo, J. and Santos, J.: *Stanford Medical Students Database (2000)*, `http://scien.stanford.edu/class/ee368/projects2001/dropbox/project16/med_students.tar.gz`

[29] Tarrés, F. and Rama, A.: *GTAV Face Database*, Available at `http://gps-tsc.upc.es/GTAV/ResearchAreas/UPCFaceDatabase/GTAVFaceDatabase.htm`

[30] Thomaz, C. E. and Giraldi, G. A.: *A New Ranking Method for Principal Components Analysis and Its Application to Face Image Analysis*, Image and Vision Computing, vol. 28, no. 6, pp. 902-913, June 2010

[31] IEEE OTCBVS WS Series Bench; DOE University Research Program in Robotics under grant DOE-DE-FG02-86NE37968; DOD/TACOM/NAC/ARC Program under grant R01-1344-18; FAA/NSSA grant R01-1344-48/49; Office of Naval Research under grant #N000143010022