國立交通大學

電控工程研究所

碩士論文

智慧型多特徵行人辨識系統設計

Intelligent multi-feature pedestrian recognition system design

研究生：梁瑋哲

指導教授：陳永平　教授

中華民國九十九年六月

智慧型多特徵行人辨識系統設計

Intelligent multi-feature pedestrian recognition system design

研 究 生：梁瑋哲　　　　Student：Wei-Tse Liang

指導教授：陳永平　　　　Advisor：Professor Yon-Ping Chen

國立交通大學
電控工程研究所
碩士論文

A Dissertation
Submitted to Institute of Electrical Control Engineering
College of Electrical Engineering and Computer Science
National Chiao Tung University
In Partial Fulfillment of the Requirements
For the degree of Master
In
Electrical Control Engineering
June 2010
Hsinchu, Taiwan, Republic of China

中華民國九十九年六月

智慧型多特徵

行人辨識系統設計

學生 ： 梁瑋哲　　　　　　　　指導教授: 陳永平 教授

國立交通大學電控工程研究所

摘 要

本篇論文針對行人辨識提出一個多特徵智慧型行人辨識系統，以達到提高行人辨識率的目的。此系統可主動在串列影像中找出移動中的物體，接著擷取物體的多種特徵，包刮梯度直方圖、Haar-like 特徵、全域平均值、梯度影像，最後判斷此物體是否為行人。在本篇論文中採用雙層的類神經網路，包刮初級層與次級層，其中初級層類神經網路先針對單一特徵進行訓練，接著再將初級層匯集至次級層進行多特徵的統合訓練，此雙層架構的設計除了可提高行人辨識率外，亦嘗試減少訓練資料的使用，由實驗結果可知此雙層架構確實可提高行人辨識率，並且在較少的訓練資料下得到相近的準確率。

Intelligent multi-feature pedestrian recognition system design

Student ： Wei-Tse Liang Advisor：Prof. Yon-Ping Chen

Institute of Electrical Control Engineering

National Chiao–Tung University

ABSTRACT

The thesis proposes an intelligent pedestrian recognition system to find out pedestrians from a sequence of images based on multi-features, including Histogram of Gradient, Haar-like feature, Global average and Gradient image. A two-staged neural network is adopted for the recognition system, which executes the training of single feature in the primary stage and then the training of multi-features in the secondary stage. The use of the two-staged neural network is not only to increase the accuracy rate but also to reduce the training data. From the experiment results, the two-staged neural network indeed improves the recognition performance and most importantly, it is workable in the case that a smaller amount of training data is used.

Acknowledgement

本篇論文得以順利完成，首先必須感謝指導教授　陳永平老師的諄諄教悔，使作者除了在研究及英文寫作上有長足的進步外，在為學處事態度上也有相當的成長，僅向老師致上最高的謝意；同時也感謝口試委員　楊谷洋老師以及　林進燈老師提供寶貴的意見，使論文得以更加平順。

此外，感謝實驗室的桓展、世宏學長在求學與專業研究的過程中給予適時的建議及鼓勵，並提供許多寶貴的經驗，同時感謝同學徐傳源以及學弟文俊、哲翰、文榜及大學到研究所的好朋友們為我研究所兩年帶來許多的歡樂及回憶，謝謝你們一路的相伴及幫助。最後，感謝爸、媽及妹妹在精神上的支持與鼓勵，還有生活上的照顧和關心，讓作者能安心的投入研究並擁有美好的碩士生活。

謹以此篇論文獻給所有關心我、照顧我的人。

梁瑋哲　2010.6

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Preliminary and thesis organization

Pedestrian detection is very important in many applications such as driver assistance and image surveillance. For driver assistance, it is used to detect pedestrian walking across the street. It is necessary when driving at night or bad weather. Most of the system is using infrared camera to detect the pedestrian. For image surveillance, it is necessary to detect pedestrian in different postures such as walking, running and standing. To solve this problem, different approaches have been released and tested. The first task is a segmentation procedure. The second one is a recognition procedure.

For the segment procedure, there are several steps should be done, first step is to find out the foreground, second the position of the object should be extract. For the first step, there are three common methods to obtain the foreground image: frame difference [10], optical flow [9] and background subtraction [8]. To obtain the exact position of the object, some paper released a fast way to extract the position of object named histogram projection [5]. After the position of the object is obtained, the recognition procedure is applied.

For the recognition procedure, first, the feature extraction is applied to generate features [4]. After the feature is extracted, the feature is sending into some intelligent machines [4] [3] such as Neural Network to learn. In the thesis, the author use Neural Network to train the features. After learning, the machines can recognize whether the feature is pedestrian or not.

The remainder of this thesis is organized as follows. Chapter 2 describes the related works of the system. Chapter 3 describes the pedestrian detecting system. Chapter 4 shows the experiment results. Chapter 5 is the conclusions of the thesis and the future works.

1.2 System Overview

For hardware architecture, the system shown in Fig. 1.1 is established by setting two cameras on a horizontal line and their lines of vision are parallel and fixed. In addition, the distance between two cameras is set as constant equal to 10 cm and these two cameras, QuickCam^{TM} Communicate Deluxe, have specification listed below. The experimental environment for testing is our laboratory and the deepest depth of the background is 180 cm.

- 1.3-megapixel sensor with RightLight™2 Technology
- Built-in microphone with RightSound™ Technology
- Video capture: Up to 1280 x 1024 pixels (HD quality) (HD Video 960 x 720 pixels)
- Frame rate: Up to 30 frames per second
- Still image capture: 5 megapixels (with software enhancement)
- USB 2.0 certified
- Optics: Manual focus

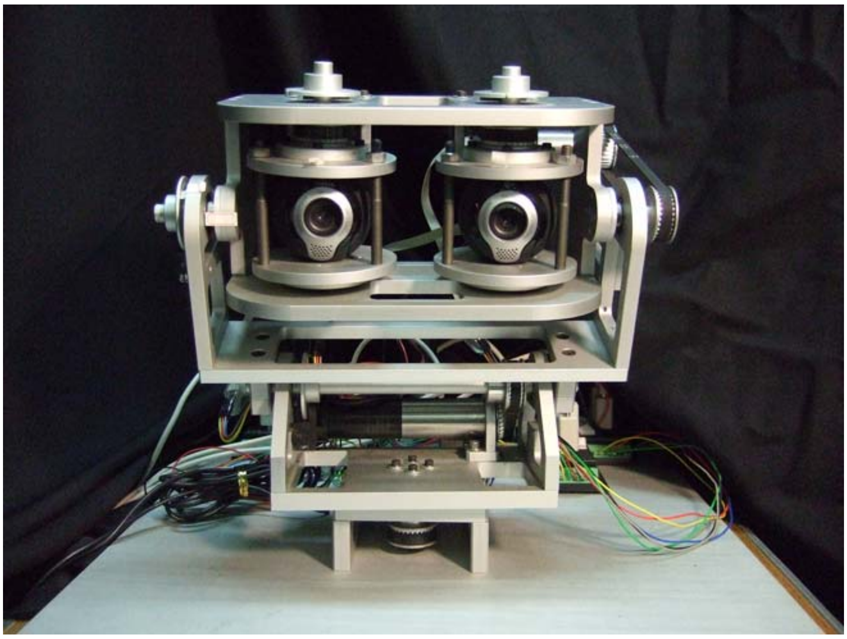


Fig. 1.1 The humanoid vision system.

For software architecture, the image shown in Fig. 1.2 is the flow chart of the proposed system. The moving object will be separate by the background subtraction. The background subrtraction often uses many images to generate background. After background is generated, foreground will be obtained by background subtraction. Because the shadow in the screen will influence the performance of detection shadow removal is applied to remove shadow [6] [15]. After shadow is removed morphology operation is appiled to removed noise. Morphology operation is a way to elliminate small area and enlarge big area. After the noise is removed, to locate the postion of the object image projection is applied. And the extract area will use feature extraction to obtain the feature. And then the data are feeding into neuralnetwork to classify if it is pedestrian.

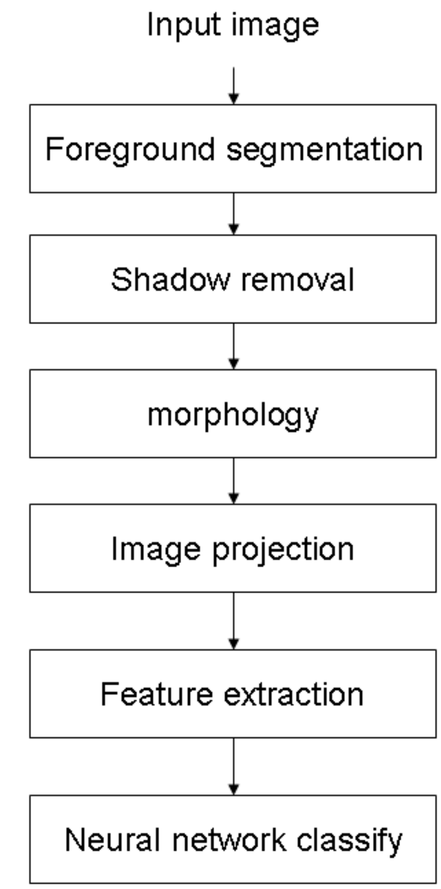


Fig. 1.2 The software architecture

Chapter 2

Related Work

2.1 Foreground segmentation

For the pedestrian detect system, the first task is to segment the region of the interesting(ROI) in the scene, there are three common way: frame difference, background subtraction, and optical flow.

Frame difference [10] method is to do pixel-based subtraction in successive frames, then a specific threshold is used to separate foreground pixels from background pixels. This method can quickly adapt to change of illumination and camera motion and lower computation. But it can only detect the ROI. This method does not yield good result when the interesting foreground regions are not sufficiently textured

Optical flow [9] reflects the image changes due to motion during a time interval, and the optical flow field is the velocity field that represents the three-dimensional motion of foreground points across a two-dimensional image. Compared with other two methods, optical flow can be more accurate to detect interesting foreground region. But optical flow computations are very intensive and difficult to realize in real time.

Background subtraction [8] is the most common method for segmentation of interesting regions in videos. This method has to build the initial background model firstly. The purpose of training background model is to subtract background image from current image for obtaining interesting foreground regions. Background subtraction method can detect the most complete of feature points of interesting foreground regions and real-time implementation. But this method can not used in presence of camera motion, and the background model must be updated in good time due to the illumination change and movement of background objects.

2.2 Shadow removal

Generally, the shadows are classified into two categories: cast shadows and self-shadows. Cast shadows refer to areas in the background model projected by objects in the direction of light ray, producing objects silhouettes. Therefore, these shadows would not exist in the extracted foreground regions, and they are ignored to regard as the part of background model. Self-shadows are parts of a segmented foreground region since there are identical movement between self-shadow and foreground region. But these shadows are unfavorable for the objects tracking, classification and incidents detection, so they are considered to remove from segmented foreground regions.

Li-Qun Xu, Jose Luis Landabaso and Montse Paradas [7] analyze foreground pixels and detect those that have similar color but lower brightness to the corresponding background regions in RGB color space. If it is true, the foreground pixel is regarded as self-shadow pixel.

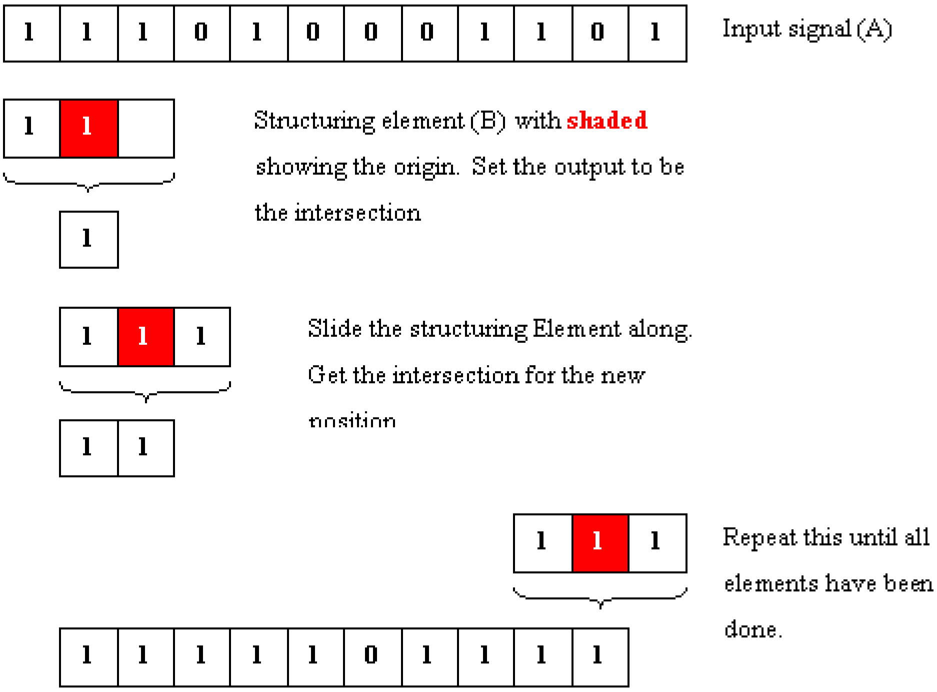
2.3 Morphology operation

Morphology has two simple function dilation and erosion.

Dilation is defined as:

$$A \oplus B = \left\{ x : (\hat{B})_x \cap A \neq \phi \right\} = \bigcup_{x \in B} A_x \qquad (2.1)$$

Where A and B are sets in Z. This equation simply means that *B* is moved over *A* and the intersection of *B* reflected and translated with *A* is found. Usually *A* will be the signal or image being operated on and *B* will be the structuring element. Figure 2.1 shows how dilation works.



Input signal (A)

Structuring element (B) with **shaded** showing the origin. Set the output to be the intersection

Slide the structuring Element along. Get the intersection for the new position

Repeat this until all elements have been done.

Fig. 2.1 Example of dilation

The opposite of dilation is known as erosion. This is defined as:

$$A\Theta B=\left\{x:(B)_x\subseteq A\right\}=\bigcap_{x\in B}A_x \qquad (2.2)$$

The equation simply says, erosion of A by B is the set of points x such that B translated by x is contained in A. Figure 2.2 shows how erosion works. This works in exactly the same way as dilation. However equation (2.2) essentially says that for the output to be a one, all of the inputs must be the same as the structuring element. Thus, erosion will remove runs of ones that are shorter than the structuring element. This thesis will applied two kind of this operation to process the image.

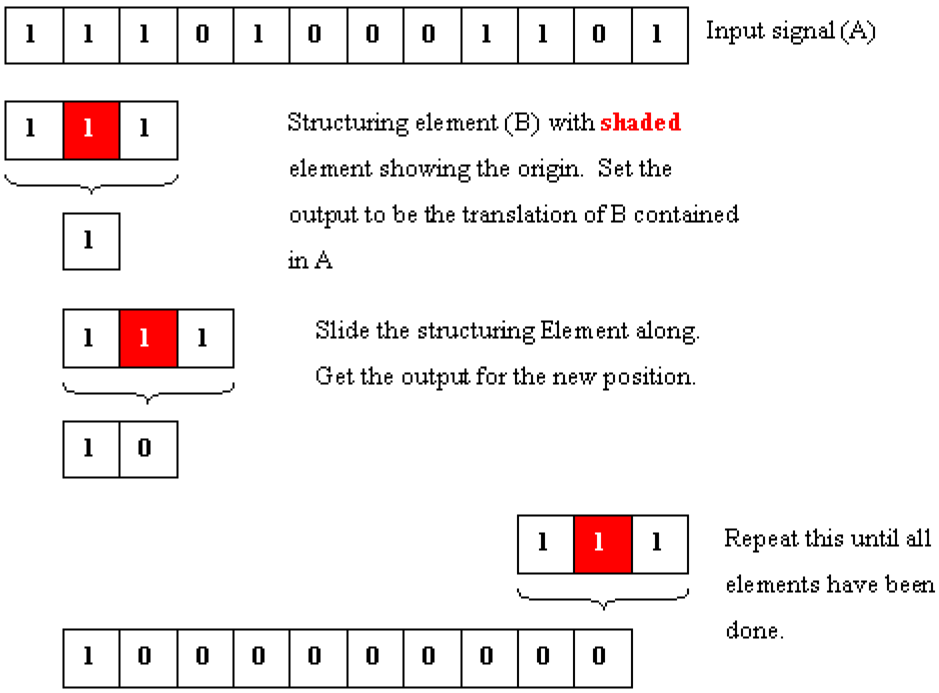

1 1 1 0 1 0 0 0 1 1 0 1
Input signal (A)
1 1 1
Structuring element (B) with shaded element showing the origin. Set the output to be the translation of B contained in A
1
1 1 1
Slide the structuring Element along. Get the output for the new position.
1 0
1 1 1
Repeat this until all elements have been done.
1 0 0 0 0 0 0 0 0 0

Fig. 2.2 Example of erosion

2.4 Neural network

2.4.1 Introduction to ANNs

The human nervous system consists of a large amount of neurons, including somas, axons, dendrites and synapses. Each neuron is capable of receiving, processing, and w_2 passing electrochemical signals from one to another. To mimic the characteristics of the human nervous system, recently investigators have developed an intelligent algorithm, called artificial neural networks (ANNs), to construct intelligent machines capable of parallel computation. This thesis will apply ANNs to the depth detection in an eyeball system through learning.

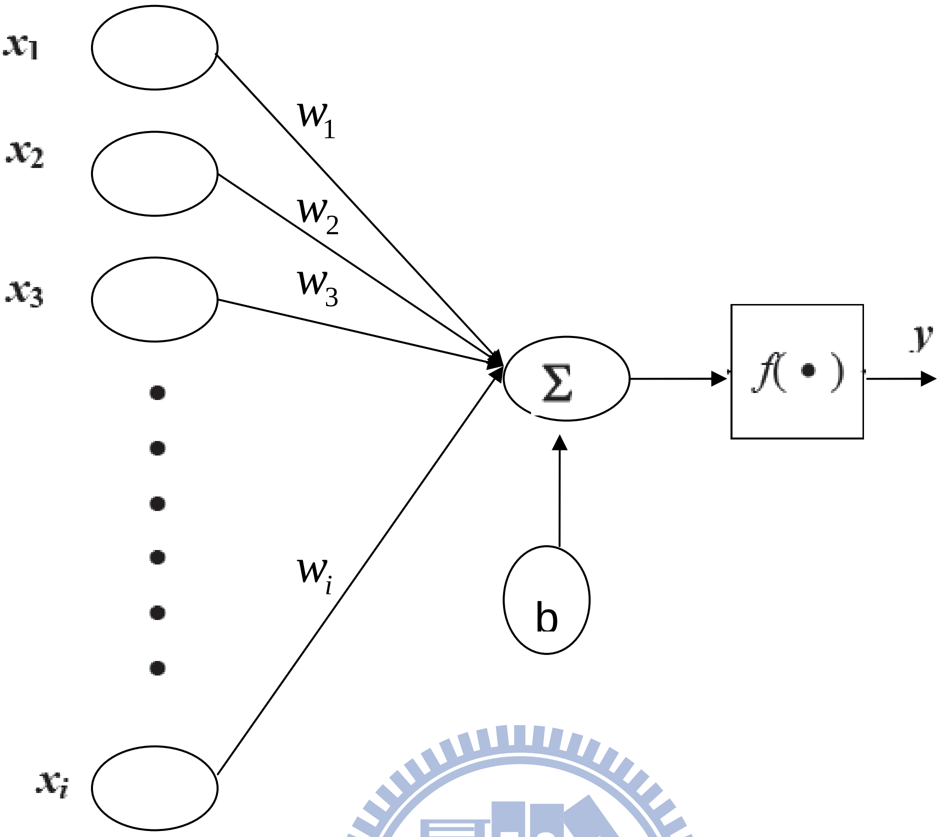


Fig. 2.3 Basic element of ANNs

ANNs can be divided into three layers which contain input layer, hidden layer, and output layer. The input layer receives signal form the outside world, which just includes input values without neuron. The neuron's number of output layer is depending on the output number. Form the output layer, the response of the net can be read. The neurons between input layer and output layer are belonging to hidden layer which does not exist necessarily. Here, each input is multiplied by a corresponding weight, analogous to synaptic strengths. The weighted inputs are summed to determine the activation level of the neuron. The connection strengths or the weights represent the knowledge in the system. Information processing takes place through the interaction among these units. The Basic element of ANNs, single layer net, is shown in Fig. 2.3 Basic element of ANNs which obeys the input-output relations

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{2.3}$$

where w_i is the weight at the input x_i and b is a bias term. The activation function $f(\bullet)$ has many types cover linear and nonlinear. Note that the commonly used activation function is

$$f(x)=\frac{1}{1+e^{-x}} \tag{2.4}$$

which is a sigmoid function. Base on the basic element, the commonest multilayer feed-forward net shown in Fig. 2.4 Multilayer feed-forward network, which contains input layer, output layer, and two hidden layers. Multilayer nets can solve more complicated problem than single layer nets, i.e. a multilayer nets is possible to solve some case that a single layer net cannot be trained to perform correctly at all. However, the training process of multilayer nets may be more difficult. The number of hidden layer and its neuron in the multilayer net are decided by complicated degree of the problem wait to solve.

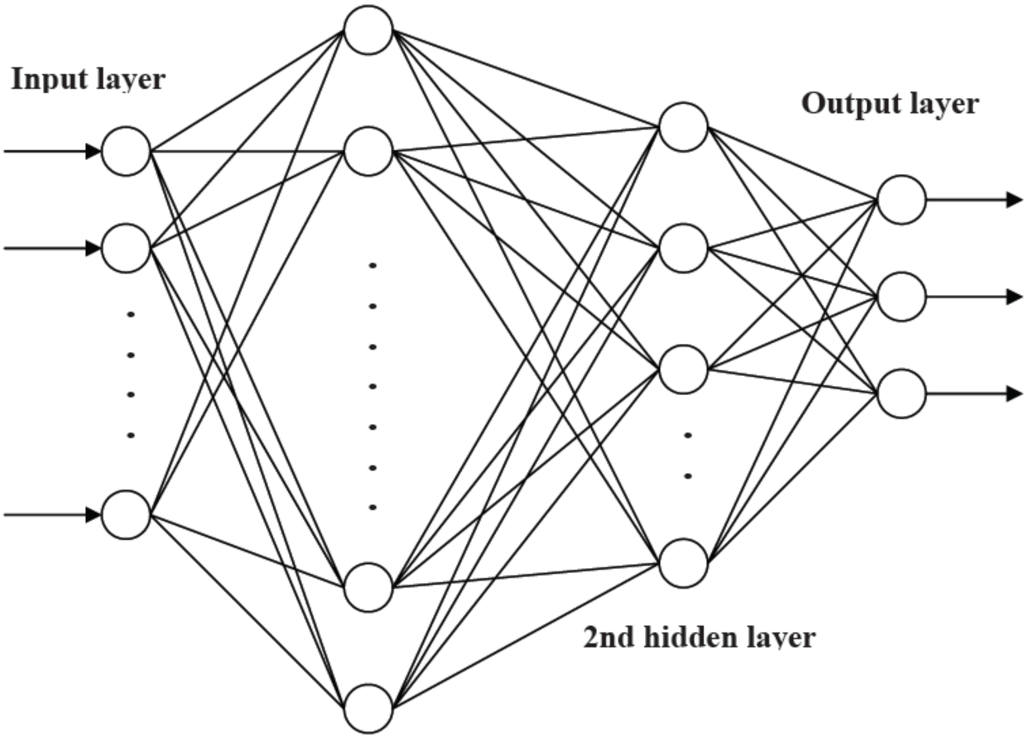


Fig. 2.4 Multilayer feed-forward network

In addition to the architecture, the method of setting the values of the weights is an important matter of different neural net. For convenience, the training for a neural network mainly classified into supervised learning and unsupervised learning. Training of supervised learning is mapping a given set of inputs to a specified set of target outputs. The weights are then adjusted according to various learning algorithms. Another type, unsupervised learning, can self-organize neural nets group similar input vectors together without the used of training data to specify what a typical member of each group looks like or to which group each vector belongs. For unsupervised learning, a sequence of input vector is provided, but no target vectors are specified. The net modifies the weights so that the most similar input vectors are assigned to the same output unit. In addition, there are nets whose weights are fixed without iterative training process, called structure learning, which change the network structure to achieve reasonable responses. In this thesis, the neural network learns the behavior by many input-output pairs, hence that is belongs to supervised learning.

2.4.2 Back-Propagation Network

In supervise learning, the back propagation learning algorithm, is widely used in most application. The back propagation, BP, algorithm was proposed in 1986 by Rumelhart, Hinton and Williams, which is based on the gradient steepest descent method for updating the weights to minimize the total square error of the output. The training by BP mainly is applied to multilayer feed-forward network which involves three stages: the feed-forward of the input training pattern, the calculation and back-propagation of the associated error, and the adjustment of the weights. Fig. 2.5 Back-propagation network shows a back-propagation network contains input layer with w_{gh} N_{inp} neurons, one hidden layer with N_{hid} neurons, and output layer with N_{out} neurons. In Fig. Back-propagation network, $x=[x_1 \quad x_2 \quad \cdots \quad x_{N_{inp}}]^T$, $h=[h_1 \quad h_2 \quad \cdots \quad h_{N_{hid}}]^T$, and $y = [y_1 \quad y_2 \quad \cdots \quad y_{N_{out}}]^T$ respectively represent the input, hidden, and out note of the network. In addition v_{ij} is the weight form the *i*-th neuron in the input layer to *j*-th neuron in the hidden layer and w_{gh} is the weight form the *g*-th neuron in the hidden layer to *h*-th neuron in the output layer.

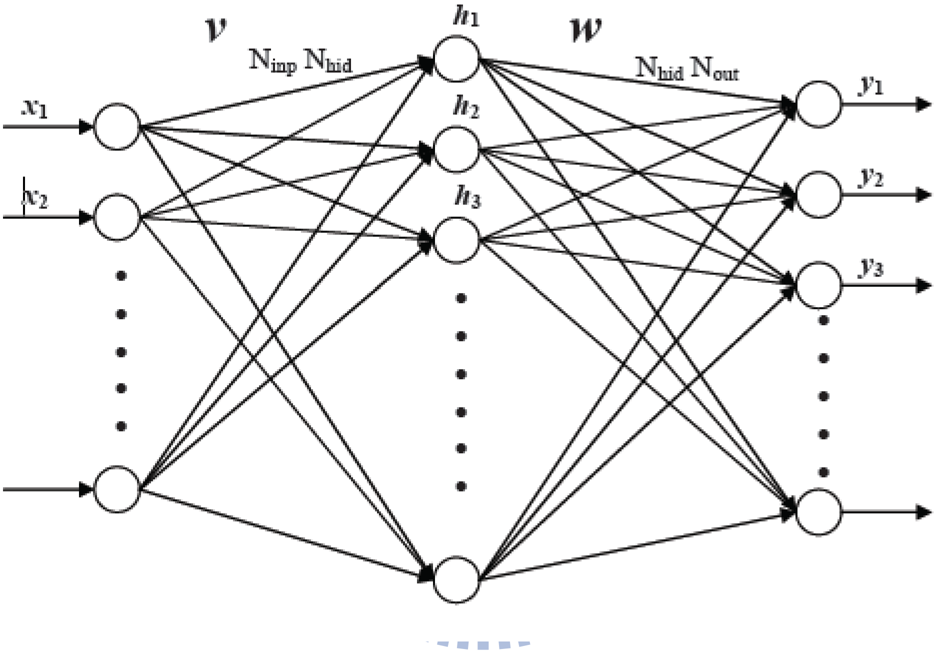


Fig. 2.5 Back-propagation network

The learning algorithm of BP is elaborated on below:

Step 1: Input the training data of input $x = [x_1 \quad x_2 \quad \cdots \quad x_{N_{inp}}]^T$ and desired output $t = [t_1 \quad t_2 \quad \cdots \quad t_{N_{inp}}]^T$. Set the maximum tolerable error $E_{\max}$ and learning rate η which between 0.1 and 1.0 to reduce the computing time or increase the precision.

Step 2: Set the initial weight and bias value of the network at random.

Step 3: Calculate the output of the m-th neuron in hidden layer

$$h_m = f_h\left(\sum_{k=1}^{N_{inp}} v_{km} x_k\right), m = 1, 2..., N_{hid} \tag{2.5}$$

where $f_h(\bullet)$ is the activation function of the neuron and the output of the i-th neuron in output layer

$$y_n = f_y\left(\sum_{q=1}^{N_{hid}} w_{mn} h_m\right), n = 1, 2..., N_{out} \tag{2.6}$$

where $f_y(\bullet)$ is the activation function of the neuron.

Step 4: Calculate the error function between network output and desired output.

$$E(w) = \frac{1}{2}\sum_{n=1}^{N_{out}} (d_n - y_n)^2 = \frac{1}{2}\sum_{n=1}^{N_{out}} \left[d_n - f_y\left(\sum_{q=1}^{N_{hid}} w_{mn} h_m\right)\right]^2 \tag{2.7}$$

Step 5: According to gradient descent method, determining the correction of weights.

$$\Delta w_{mn} = -\eta \frac{\partial E}{\partial w_{mn}} = -\eta \frac{\partial E}{\partial y_n}\frac{\partial y_n}{\partial w_{mn}} = \eta (d_n - y_n)\left[f_y'\left(\sum_{q=1}^{N_{hid}} w_{mn} h_m\right)\right] h_m = h\delta_{mn} h_m \tag{2.8}$$

and

$$\begin{aligned}\Delta v_{km} &= -\eta \frac{\partial E}{\partial v_{km}} = -\eta \sum_{n=1}^{N_{out}} \frac{\partial E}{\partial y_n}\frac{\partial y_n}{\partial h_m}\frac{\partial h_m}{\partial v_{km}} \\ &= \eta \sum_{n=1}^{N_{out}} \left[(d_n - y_n) f_y'\left(\sum_{q=1}^{N_{hid}} w_{mn} h_m\right) w_{mn}\right] f_h'\left(\sum_{k=1}^{N_{inp}} v_{km} x_k\right) x_k = \eta \delta_{kmn} x_k\end{aligned} \tag{2.9}$$

where

$\delta_{mn} = (d_n - y_n)\left[f_y'\left(\sum_{q=1}^{N_{hid}} w_{mn} h_m\right)\right]$ and $\delta_{kmn} = \sum_{n=1}^{N_{out}} \left[(d_n - y_n) f_y'\left(\sum_{q=1}^{N_{hid}} w_{mn} h_m\right) w_{mn}\right] f_h'\left(\sum_{k=1}^{N_{inp}} v_{km} x_k\right)$.

step 6: Propagate the correction backward to update the weights.

$$\begin{cases} w(n+1) = w(n) + \Delta w \\ v(n+1) = v(n) + \Delta v \end{cases} \tag{2.10}$$

Step 7: Check whether the whole training data set have learned already. Networks learn whole training data set once called a learning circle. If the network not goes through a learning circle, return to Step 1; otherwise, go to Step 8.

Step 8: Check whether the network converge. If $E < E_{\max}$, terminate the training process; otherwise, begin another learning circle by going to Step 1.

BP learning algorithm can be used to model various complicated nonlinear functions. Recently years The BP learning algorithm is successfully applied to many domain applications, such as: pattern recognition, adaptive control, clustering problem, etc. In the thesis, the BP algorithm was used to learn the input-output relationship for clustering problem.

Chapter 3

Pre-processing and detect algorithm of pedestrian detection

Before classifying objects to be pedestrian or non-pedestrian, there are two pre-processes required to make the classification system work well. First, determine the locations of moving objects, and then find out the object's location in the scene.

In general, there are three techniques adopted to determine the locations of moving objects, which are background subtraction [8], optical flow [9], and frame difference [10]. This thesis will employ the background subtraction since it consumes less computation time and promotes the efficiency of classification.

3.1 Foreground segmentation

Foreground segmentation is one of the most important techniques in the system, because it increases the processing efficiency and its performance decides the quality of classification. Conventional foreground segmentation algorithms are roughly classified into two categories. The first category [11] uses spatial homogeneity as a criterion, but consumes tremendous computation time. The second category [12] is based on the frame change or background mosaics, and can process fast to distinguish the object regions from a static background.

Collins *et al*. [13] combines the temporal difference and background to detect moving pixels. First, moving regions are detected from two frames in sequence with the temporal difference, and then the compact moving pixels are further extracted by subtracting the background in these regions. But they also model each pixel of background with a normal distribution.

The foreground segmentation algorithm adopted in this paper has been presented by Kim *et al*. [14] as shown in Figure 3.1, which belongs to the second category. It is known that the background information is very sensitive to the variation of illumination. To deal with such problem, in the initialization step two background masks, $I_{\min}$ and $I_{\max}$, are obtained from the first *N* frames and defined as

$$I_{\min}(x,y) = \min_{k=1,2,\ldots,N}\{I_k(x,y)\} \tag{3.1}$$

$$I_{\max}(x,y) = \max_{k=1,2,\ldots,N}\{I_k(x,y)\} \tag{3.2}$$

where $I_k(x, y)$ is the intensity of the *k*th frame at (*x*, *y*), *k*=1,2,…,*N*. Once finish the initialization, the frame difference mask is calculated as

$$I_{fd,k}(x,y) = \left|I_k(x,y) - I_{k-1}(x,y)\right|,\ \text{k>N} \tag{3.3}$$

Let $I_{fg,k}(x,y)$ be the foreground mask corresponding to the *kth* frame $I_k(x,y)$. Based on (3.1), (3.2) and (3.3), if $I_k(x,y)$ satisfies the following conditions

$$I_k(x,y) < I_{\min}(x,y) - Th_{bg} \tag{3.4}$$

$$I_k(x,y) > I_{\max}(x,y) + Th_{bg} \tag{3.5}$$

$$I_{fd,k}(x,y) > Th_{fd} \tag{3.6}$$

then $I_{fg,k}(x,y)=1$, representing the pixel at (*x*,*y*) belongs to foreground, otherwise $I_{fg,k}(x,y)=0$, representing the pixel at (*x*,*y*) belongs to background. Next the shadow removal method will be applied to remove the shadow in the mask $I_{fg,k}(x,y)$.

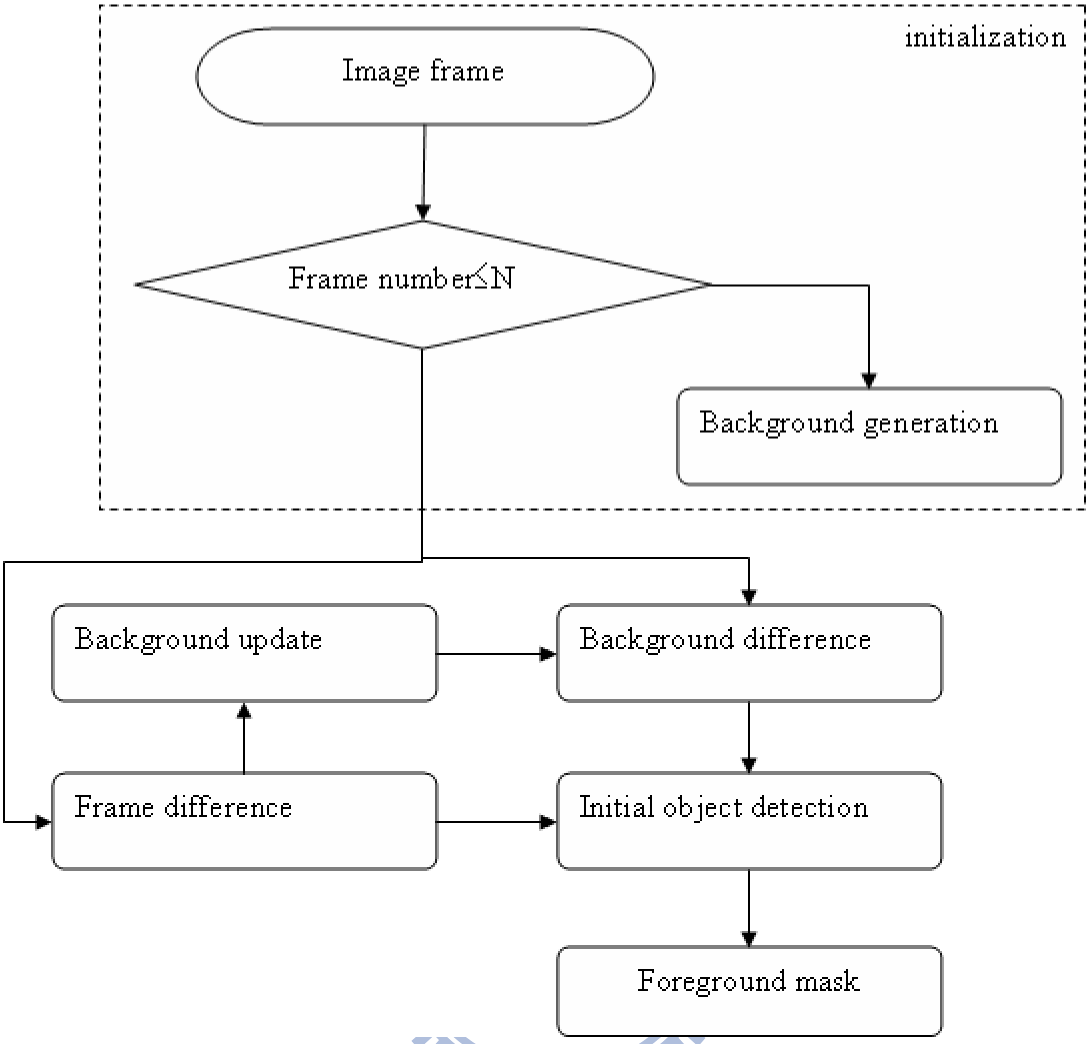


Fig. 3.1 Foreground segmentation algorithm

3.2 Shadow removal

In general, an object in a scene is unavoidable to have shadow caused by the light source. Since the shadow often leads to errors in classification and objects detection, it is required for a pedestrian detect system to remove the shadow of the object to be detected by the so-called shadow removal method. This thesis will refer to the method proposed by Cucchiara et al. [15], which uses the HSV color space, closer to human projection of color, and discriminates shadows from foreground more accurately.

It is known that a smaller difference exists in hue between shadow and background and the shadow often has lower saturation. Based on the *kth* frame $I_k(x,y)$ and the background image $B(x,y)$, the proposed algorithm [15] for shadow remove requires three conditions given as

$$|I_k^S(x,y) - B^S(x,y)| \le \tau_S \tag{3.7}$$

$$|I_k^H(x,y) - B^H(x,y)| \le \tau_H \tag{3.8}$$

$$\alpha \le \frac{I_k^V(x,y)}{B^V(x,y)} \le \beta \tag{3.9}$$

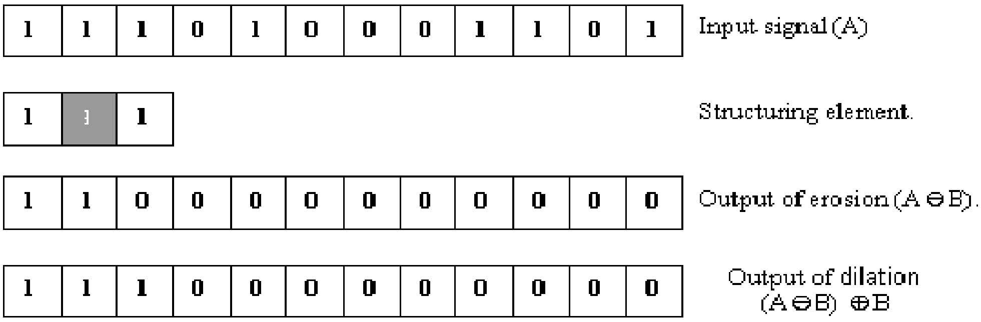
where τ_S, τ_H, α and β are the threshold values. The use of threshold α is to avoid the dark pixel being misclassified as shadow due to the similar hue and saturation information in the dark pixels. The use of threshold β is to eliminate noise resulted from slightly change in background model as shadow. The shadow removal operation will be applied only when foreground mask $I_{fg,k}(x,y)$ at (x,y) is equal to 1 and the value $I_{fg,k}(x,y)$ will be change into 0 while $I_k(x,y)$ satisfy three conditions. Next the morphology operation will be applied to reduce the noise in the foreground mask.

3.3 Morphology operation

After applying shadow removal operation, shadows are removed from the foreground mask, but some noise still exists therein. One of the conventional ways to eliminate noise regions is using the morphological operations to filter out isolated regions, smaller than 3x3. In the thesis, the isolated regions are eliminated by the following morphology opening operation defined as:

$$A \circ B = (A \Theta B) \oplus B \tag{3.10}$$

which combines the erosion operation and the dilation operation, as shown in Figure 3.2. As can be seen, the zeros are opened up. Any ones that are shorter than the structuring element are removed, but the rest of the signal is left unchanged.



1	1	1	0	1	0	0	0	1	1	0	1	Input signal (A)
1	1	1										Structuring element.
1	1	0	0	0	0	0	0	0	0	0	0	Output of erosion (A ⊖ B).
1	1	1	0	0	0	0	0	0	0	0	0	Output of dilation (A⊖B) ⊕B

Fig. 3.2 Example of opening operation

In order to smooth the boundaries of the foreground mask *f*(*j*,*k*) and eliminate holes therein to achieve a new foreground mask *F*(*j*,*k*), this thesis adopts a profile extraction technique which is processed by the following steps [16]:

Step-1:

For the *j*-th row, *j*=1,2,…,240, the index of the pixel (*j*,*k*) is chosen horizontally from *k*=1 to *k*=301 as below:

$$g_1(j,k) = \sum_{i=1}^{20} \left(1 - 0.05(i-1)\right) f(j, k+i-1) \tag{3.11}$$

where *f*(*j*,*k*) is the foreground mask. With the use of threshold *M*, which is determined by trial and error, the resulted image is obtained as

$$F_1(j,k)=\begin{cases}1 & \text{for } k\text{=1,2,...,301, and } g_1(j,k)>M, \\ 0 & \text{for } k\text{=1,2,...,301, and } g_1(j,k)\le M, \\ f(j,k) & \text{for } k\text{=302,...,320}\end{cases} \tag{3.12}$$

Step-2:

For the *k*-th column, *k*=1,2,…,320, the index of the pixel (*j*,*k*) is chosen vertically from *j*=1 to *j*=221 as below:

$$g_2\left(j,k\right)=\sum_{i=1}^{20}\left(1-0.05(i-1)\right)F_1(j-i+1,k) \tag{3.13}$$

With the same threshold *M*, the resulted image is found as

$$F_2(j,k)=\begin{cases}1 & \text{for } j\text{=1,2,...,221, and } g_2(j,k)>M, \\ 0 & \text{for } j\text{=1,2,...,221, and } g_2(j,k)\le M, \\ F_1(j,k) & \text{for } j\text{=222,...,240}\end{cases} \tag{3.14}$$

Step-3:

Different to Step-2, the index of the pixel (*j*,*k*) for the *k*-th column, *k*=1,2,…,320, is chosen vertically but reversely from *j*=240 to *j*=20 as below:

$$g_3\left(j,k\right)=\sum_{i=1}^{20}\left(1-0.05(i-1)\right)F_2(j-i+1,k) \tag{3.15}$$

Then, based on the same threshold *M*, the resulted image is attained as

$$F_3(j,k)=\begin{cases}1 & \text{for } j\text{=240,239,...20, and } g_3(j,k)>M, \\ 0 & \text{for } j\text{=240,239,...20, and } g_3(j,k)\le M, \\ F_2(j,k) & \text{for } j\text{=19,...,1}\end{cases} \tag{3.16}$$

Step-4:

With the reverse process to Step-1, the index of the pixel (*j*,*k*) for the *j*-th row, *j*=1,2,…,240, is chosen horizontally from *k*=320 to *k*=20 as below:

$$g_4\left(j,k\right)=\sum_{i=1}^{20}\left(1-0.05(i-1)\right)F_3(j,k-i+1) \tag{3.17}$$

Finally, the new foreground mask is generated as

$$F(j,k)=\begin{cases} 1 & \text{for } j\text{=320,319,...20, and } g_4(j,k)>M, \\ 0 & \text{for } j\text{=320,319,...20, and } g_4(j,k)\le M, \\ F_3(j,k) & \text{for } j\text{=19,...,1} \end{cases} \tag{3.18}$$

which will be used to detect the object positions by the histogram projection.

3.4 Histogram projection

The histogram projection is a fast way to detect the region of interest and the number of objects in the foreground mask. The histogram projection is created by counting pixels of nonzero for both the horizontal and vertical directions in the image. The process is shown as below:

Step-1:

In step-1, the number of objects and the boundary of each object along horizontal will be determined by horizontal histogram which is calculated as

$$H_i=\sum_{j=1}^{240}F(i,j)\,, \qquad i\text{=1,2,…,320} \tag{3.19}$$

where H_i denotes *i*-th column horizontal histogram and $F(i,j)$ denotes the final foreground mask. Then, digitize H_i as

$$D_i^H=\begin{cases} 1 & H_i>0 \\ 0 & H_i=0 \end{cases} \qquad i\text{=1,2,…,320} \tag{3.20}$$

Further define the horizontal boundary index as

$$B_i^H = D_{i+1}^H - D_i^H \qquad i=1,2,\ldots,320 \tag{3.21}$$

Clearly, $B_i^H = 1$ and $B_i^H = -1$ respectively represent the left and right boundaries of an object at *i*-th column. The amount of columns with $B_i^H = 1$ is the number of objects.

Step-2:

According to the horizontal boundary index, the columns related to each object are obtained and processed separately. Suppose that the *k*-th object is between *p*-th column and *q*-th column, i.e., $B_p^H = 1$ and $B_q^H = -1$, then the vertical histogram is calculated as

$$V(k)_j = \sum_{i=p}^{q} F(i,j), \qquad 1 \le j \le 240 \tag{3.22}$$

Then, digitize $V(k)_j$ as

$$D_j^{V(k)} = \begin{cases} 1 & V(k)_j > 0 \\ 0 & V(k)_j = 0 \end{cases} \qquad j=1,2,\ldots,240 \tag{3.23}$$

Further define the vertical boundary index as

$$B_j^{V(k)} = D_{j+1}^{V(k)} - D_j^{V(k)} \qquad j=1,2,\ldots,240 \tag{3.24}$$

Clearly, $B_j^{V(k)} = 1$ and $B_j^{V(k)} = -1$ respectively represent the upper and lower boundaries of an object at *j*-th row. The example is shown as Figure 3.3. After the histogram projection, the object's boundary can be found out. Next, the feature extraction will be applied to extract the feature from the region of interesting.

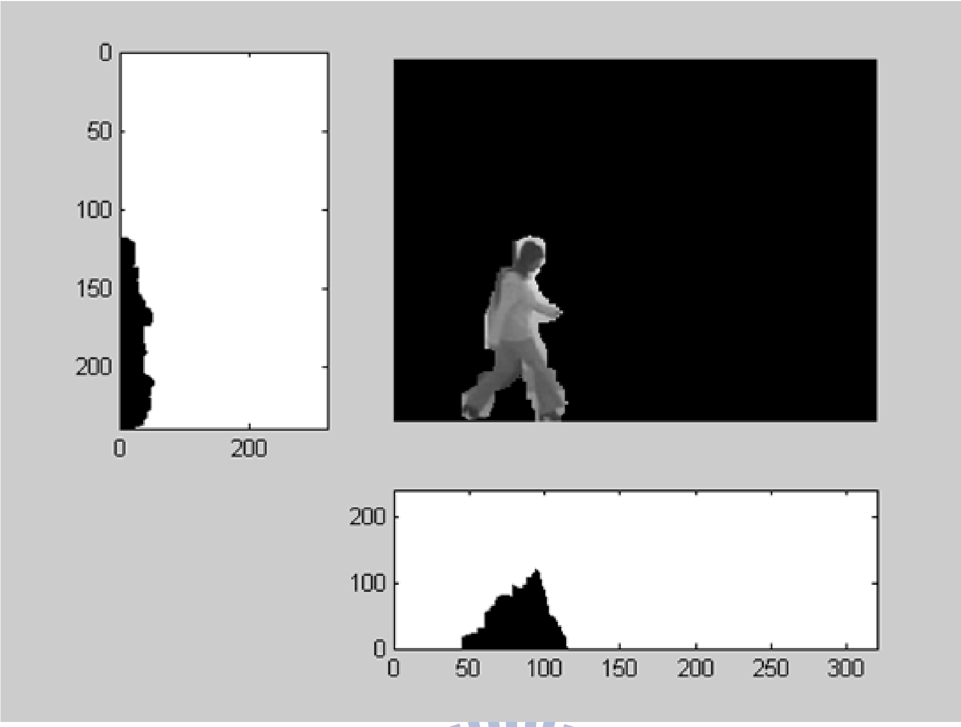


Fig. 3.3 Example of histogram projection

3.5 Feature extraction

The inputs of neural network are the features generated by feature extraction, including histogram-of-oriented-gradients (HOG), global averaging, Haar-like features and image gradient.

The feature HOG is first used for pedestrian detection by Shashua *et al*.[17]. They extract orientation histogram features from 13 fixed overlapping parts according to the different subregions and clustered training subsets. The method presented in the thesis is proposed by Dalal and Triggs[18] and Zhu *et al*.[19], which calculate the HOG by the following steps:

Step-1 :

Compute both the magnitude and orientation of the gradient, which are defined as below:

$$m(x,y)=\sqrt{F_x(x,y)^2+F_y(x,y)^2} \tag{3.25}$$

$$\theta(x,y)=\tan^{-1}\frac{F_y(x,y)}{F_x(x,y)} \tag{3.26}$$

where F_x and F_y are the respective gradients in the horizontal and vertical directions obtained by convolving the image with Sobel filters.

Step-2:

The gradient orientation is evenly divided into 9 bins over 0° to 180°. The sign of the orientation is ignored; thus, the orientations between 180° to 360° are deemed the same as those between 0° and 180°. Then, the gradient orientation histograms $E(i,\,j)_k$ in each orientation bin *k* of block $B(i,\,j)$ are obtained by summing all the gradient magnitudes whose orientations belong to bin *k* in $B(i,\,j)$

$$E(i,\,j)_k=\sum_{\substack{(x,y)\in B(i,j)\\ \theta(x,y)\in bin_k}} m(x,y) \tag{3.27}$$

And then apply normalization L2-norm to obtain normalized histograms within a cell of 2x2 blocks, expressed as

$$NE(i,\,j)=\left[NE(i,\,j)_1\ NE(i,\,j)_2\ \cdots\ NE(i,\,j)_9\right] \tag{3.28}$$

where

$$NE(i,j)_k = \frac{E(i,j)_k}{\sqrt{\left\| \sum_{l=1}^{9} \left(E(i,j)_l + E(i+1,j)_l + E(i,j+1)_l + E(i+1,j+1)_l \right) \right\|^2 + e^2}} \qquad (3.29)$$

and e is a small constant.

Step-3:

After normalization, all the normalized histograms $NE(i,j)$ are further grouped into a single vector expressed as

$$HOG = [NE(1)\ NE(2)\ \cdots\ NE(16)] \qquad (3.30)$$

where

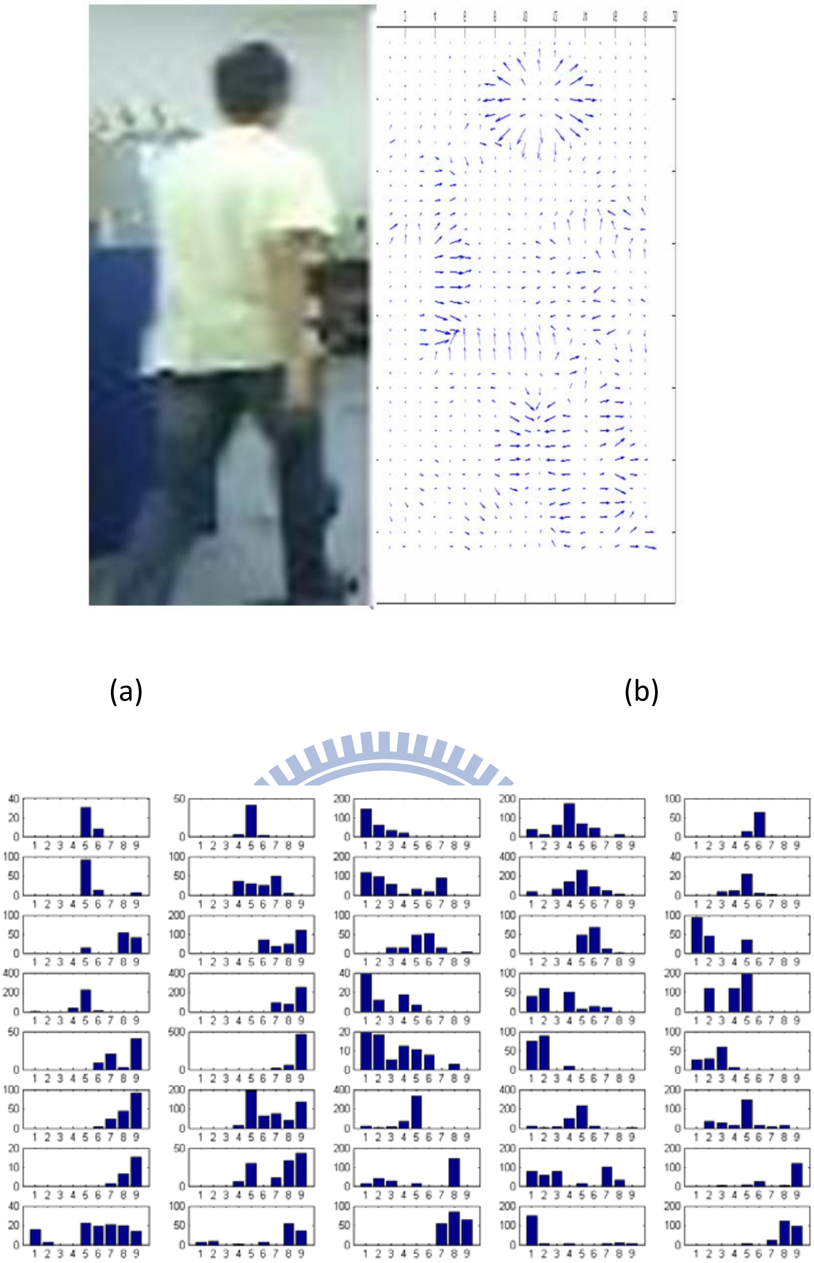
$$NE(i) = \left[NE(i,1)\ NE(i,2)\ \cdots\ NE(i,8) \right] \qquad (3.31)$$

An example of HOG is shown in Fig.3.4.

For the second feature, global averaging [21], a pedestrian image of size 36x18 pixels is segmented and separated into 162 blocks. Define the pattern average value of block i as:

$$PatAv_i = \frac{1}{2 \cdot 2} \sum_{l=1}^{2} \sum_{k=1}^{2} p_i(k,l), i = 1, 2, \ldots, 162 \qquad (3.32)$$

where $p_i(k,l)$ is the pixel value at coordinates k and l in block i. After all the pattern average values $PatAv_i$ are obtained, they will be used as the input to the neural network for both training and testing.



(a) (b)

(c)

Fig. 3.4 (a) Original image. (b) Gradient orientation and magnitude of each pixel are illustrated by the arrows' direction and length. (c) HOG computation in each cell

For the third feature, Haar-like feature, it can effectively capture the different appearance details of objects and have a fast algorithm with the help of integral images [20]. To generate the haar-like features of an image, it is required to separate the image into blocks, and then with the help of haar-like mask on each block a vector of haar-like features is obtained. For example, there are five kinds of masks shown in Fig. 3.5, each mask with feasible size and aspect ratio. In this thesis, the second type of mask will be used to generate the haar-like feature H^{Haar}, whose *i*th compoment is corresponding to the *i*th block B_i and obtained as

$$H^{Haar}(i) = S_{white}(B_i) - S_{black}(B_i) \tag{3.33}$$

where $S_{white}(B_i)$ and $S_{black}(B_i)$ are respectively the intensity summations subject to the white and black regions of the mask. In order to generate more dimension of the haar-like feature, the author modifies the equation as

$$H^{Haar}(i) = S_{white}(p(k,l)) - S_{black}(p(k,l)) \tag{3.34}$$

where *p*(*k*,*l*) is the pixel value at coordinates k and l.

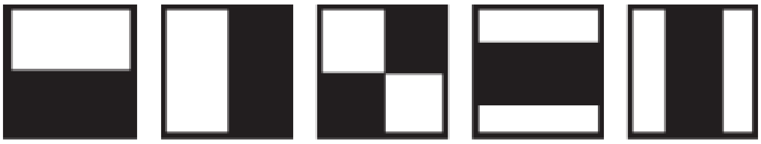


Fig. 3.5 Haar-like mask

For the fourth feature, Gradient feature, it is a vector formed by the gradient image with pixel (F_x , F_y). An example of gradient image is shown in Fig. 3.6.

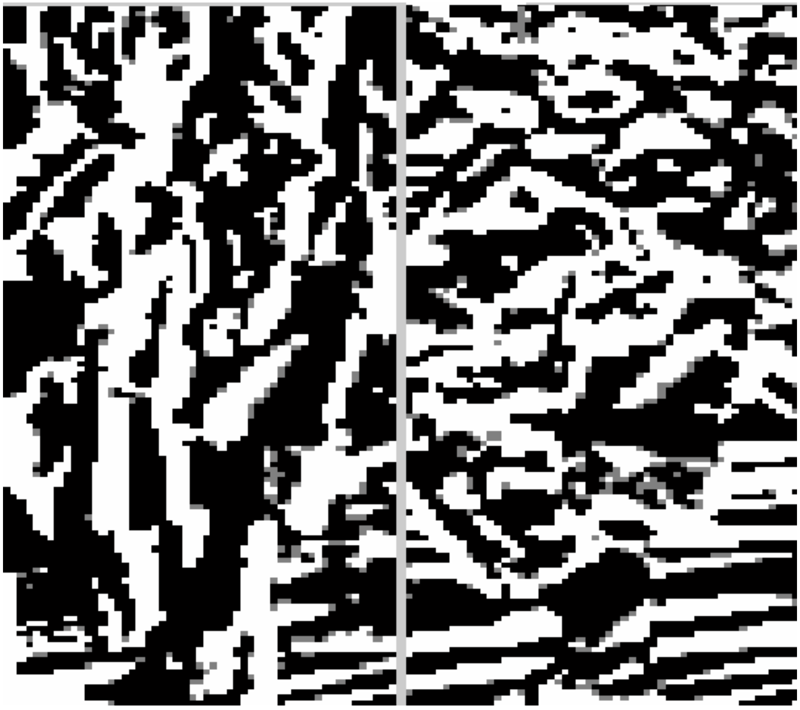


Fig. 3.6 The gradient image for F_x and F_y

After the four features are generated, they are used as the training patterns of a neural network to determine whether the image is pedestrian or not.

3.6 Detect Algorithm

In general, traditional way uses only one target object's feature for the pedestrian recognition. Unfortunately, pedestrian-related images are usually too complicated to recognize only by one feature. To improve such defect, intuitively the use of more features should lead to a higher recognition performance. But using multi-feature as the training pattern for a neural network will increase the difficulty of its learning. The thesis proposes a novel structure of neural network, which includes four primary neural networks and one secondary neural network as shown in Fig.3.10, to deal with the multi-feature problem.

There are two training stages required for the proposed structure for pedestrian recognition, one for the primary neural networks and one for the secondary neural network. For the first stage, there are four primary neural networks, named as HOG-neural network(HOGNN), Gradient-neural network(GRDNN), Haar-neural network(HARNN) and Global averaging neural network(GAVNN), each related to one feature and trained in back-propagation.

For HOGNN, there include one input layer with 1152 nodes, two hidden layers with 100 and 30 nodes, and one output layer with 1 node. There are 1152 components of the HOG feature vector in (3.x) and all the components, represented by HOG(p), p=1,2,…,1152, are sent into the 1152 nodes of the input layer, correspondingly. The pth input node is connected to the qth node of the first hidden layer with weighting $W1_{hog}(p,q)$. Hence, there exists a weighting array $W1_{hog}(p,q)$ of dimension 100x1152, p=1,2,…,1152 and q=1,2,…,100. Besides, the qth node of the first hidden layer is also added with an extra bias $b1_{hog}(q)$, q=1,2,…,100. Similarly, The qth node of the first hidden layer is connected to the rth node of the second hidden layer with weighting $W2_{hog}(q,r)$, which results in a weighting array $W2_{hog}(q,r)$ of dimension 30x100, q=1,2,…,100 and r=1,2,…,30. An extra bias $b2_{hog}(r)$, r=1,2,…,30, is added to the rth node of the second hidden layer. Finally, the rth node of the second hidden layer is connected to the output node with weighting $W3_{hog}(r)$, r=1,2,…,30, and a bias $b3_{hog}$ is added to the output node.

Let the activation function of the first hidden layer be the hyperbolic tangent sigmoid transfer function and then the output of its qth node $O1_{hog}(q)$ is expressed as:

$$O1_{hog}(q) = tansig(n_1) = \frac{2}{1+exp(-2n_1)} - 1, \quad q = 1,2,\ldots 100\,. \qquad (3.35)$$

where

$$n_1 = \sum_{p=1}^{1152} W1_{hog}(p,q)HOG(p) + b1_{hog}(q) \tag{3.36}$$

Similarly, for the second hidden layer its *r*th node $O2_{hog}(r)$ is expressed as:

$$O2_{hog}(r) = tansig(n_2) = \frac{2}{1+exp(-2n_2)} - 1, \quad r = 1,2,\ldots,30. \tag{3.37}$$

where

$$n_2 = \sum_{q=1}^{100} W2_{hog}(q,r)O1_{hog}(q) + b2_{hog}(r) \tag{3.38}$$

Let the activation function of the output layer be the log-sigmoid transfer function and then the output of its *l*th node $O3_{hog}(l)$ is expressed as:

$$O3_{hog}(l) = tansig(n_3) = \frac{1}{1+exp(n_3)}, \quad l = 1. \tag{3.39}$$

where

$$n_2 = \sum_{r=1}^{30} W3_{hog}(r)O2_{hog}(r) + b3_{hog} \tag{3.40}$$

the structure is shown as Fig.3.7

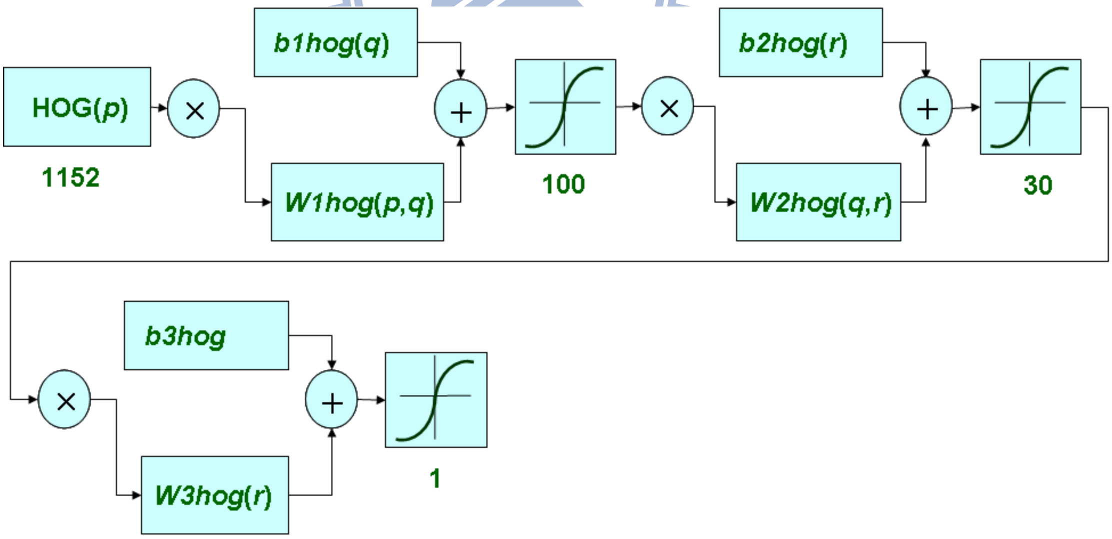


Fig. 3.7 HOGNN

For GRDNN, there include one input layer with 494 nodes, three hidden layers with 120, 30 and 30 nodes, and one output layer with 1 node. There are 494 components of the GRD feature vector and all the components, represented by GRD(*p*), *p*=1,2,…,494, are sent into the 494 nodes of the input layer, correspondingly. The *p*th input node is connected to the *q*th node of the first hidden layer with weighting $W1_{GRD}(p,q)$. Hence, there exists a weighting array $W1_{GRD}(p,q)$ of dimension 120x494, p=1,2,…,494 and q=1,2,…,120. Besides, the *q*th node of the first hidden layer is also added with an extra bias $b1_{GRD}(q)$, q=1,2,…,120. Similarly, The *q*th node of the first hidden layer is connected to the *r*th node of the second hidden layer with weighting $W2_{GRD}(q,r)$, which results in a weighting array $W2_{GRD}(q,r)$ of dimension 30x120, q=1,2,…,120 and r=1,2,…,30. An extra bias $b2_{GRD}(r)$, r=1,2,…,30, is added to the *r*th node of the second hidden layer. The *r*th node of the second hidden layer is connected to the output node with weighting $W3_{GRD}(r,l)$, r=1,2,…,30, and an extra bias $b3_{GRD}(l)$, r=1,2,…,30, is added to the *l*th node of the third hidden layer. Finally, the *l*th node of the third hidden layer is connected to the output node with weighting $W4_{GRD}(l)$, l=1,2,…,30, and a bias $b4_{GRD}$ is added to the output node.

Let the activation function of the first hidden layer be the hyperbolic tangent sigmoid transfer function and then the output of its *q*th node $O1_{GRD}(q)$ is expressed as:

$$O1_{GRD}(q) = tansig(n_1) = \frac{2}{1+exp(-2n_1)} - 1, \quad q = 1,2,\ldots 120. \tag{3.41}$$

where

$$n_1 = \sum_{p=1}^{494} W1_{GRD}(p,q) HOG(p) + b1_{GRD}(q)$$

Similarly, for the second hidden layer its *r*th node $O2_{GRD}(r)$ is expressed as:

$$O2_{GRD}(r) = tansig(n_2) = \frac{2}{1+exp(-2n_2)} - 1, \quad r = 1,2,\ldots,30. \tag{3.42}$$

where

$$n_2 = \sum_{q=1}^{120} W2_{GRD}(q,r) O1_{GRD}(q) + b2_{GRD}(q)$$

For the third hidden layer its *l*th node $O3_{GRD}(l)$ is expressed as:

$$O3_{GRD}\left(l\right) = tansig\left(n_2\right) = \frac{2}{1+exp\left(-2n_2\right)} - 1, \qquad l = 1,2,\ldots,30\,. \tag{3.43}$$

where

$$n_2 = \sum_{r=1}^{30} W3_{GRD}(r,l)O2_{GRD}(r) + b3_{GRD}(r) \tag{3.44}$$

Let the activation function of the output layer be the log-sigmoid transfer function and then the output of its *l*th node $O4_{GRD}\left(m\right)$ is expressed as:

$$O4_{GRD}\left(m\right) = tansig\left(n_3\right) = \frac{1}{1+exp\left(n_3\right)}, \qquad m = 1\,. \tag{3.45}$$

where

$$n_2 = \sum_{l=1}^{30} W4_{GRD}(l)O3_{GRD}(l) + b4_{GRD} \tag{3.46}$$

the structure is shown as Fig.3.8.

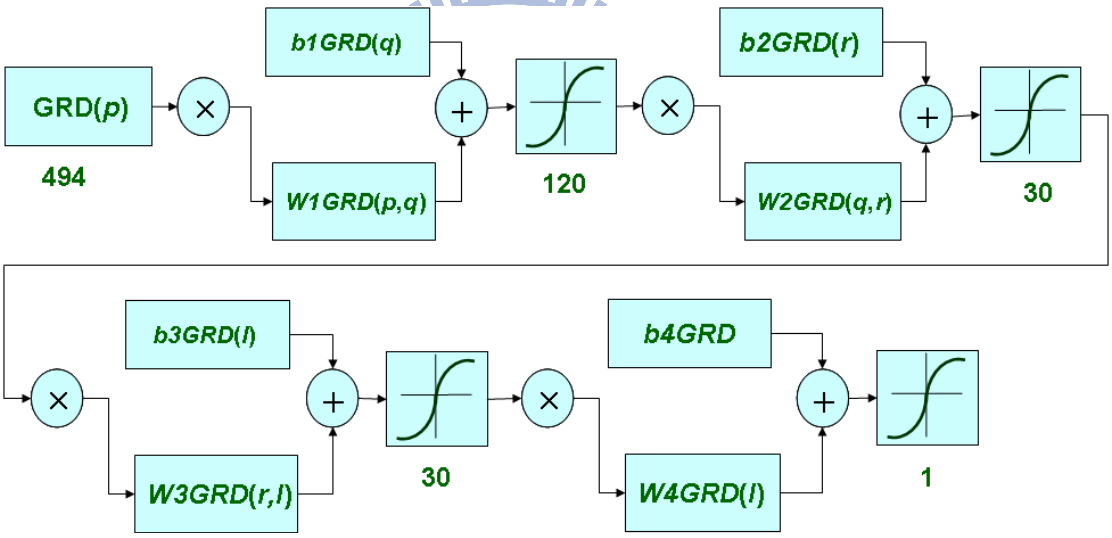


Fig. 3.8 GRDNN

For HARNN, there include one input layer with 495 nodes, two hidden layers with 200 and 30 nodes, and one output layer with 1 node. There are 495 components of the HAR feature vector in (3.x) and all the components, represented by HAR(*p*), *p*=1,2,…,495, are sent into the 495 nodes of the input layer, correspondingly. The *p*th input node is connected to the *q*th node of the first hidden layer with weighting $W1_{HAR}(p,q)$. Hence, there exists a weighting array $W1_{HAR}(p,q)$ of dimension 100x495, *p*=1,2,…,495 and *q*=1,2,…,200. Besides, the *q*th node of the first hidden layer is also added with an extra bias *b1*$_{HAR}$(*q*), *q*=1,2,…,200. Similarly, The *q*th node of the first hidden layer is connected to the *r*th node of the second hidden layer with weighting *W2*$_{HAR}$(*q*,*r*), which results in a weighting array *W2*$_{HAR}$(*q*,*r*) of dimension 30x200, *q*=1,2,…,100 and *r*=1,2,…,30. An extra bias *b2*$_{HAR}$(*r*), *r*=1,2,…,30, is added to the *r*th node of the second hidden layer. Finally, the *r*th node of the second hidden layer is connected to the output node with weighting *W3*$_{HAR}$(*r*), *r*=1,2,…,30, and a bias *b3*$_{HAR}$ is added to the output node.

Let the activation function of the first hidden layer be the hyperbolic tangent sigmoid transfer function and then the output of its *q*th node $O1_{HAR}\left(q\right)$ is expressed as:

$$O1_{HAR}\left(q\right) = tansig(n_1) = \frac{2}{1+exp\left(-2n_1\right)} - 1, \quad q = 1, 2, \ldots 200. \tag{3.47}$$

where

$$n_1 = \sum_{p=1}^{495} W1_{HAR}(p,q)HAR(p) + b1_{HAR}(q) \tag{3.48}$$

Similarly, for the second hidden layer its *r*th node $O2_{HAR}\left(r\right)$ is expressed as:

$$O2_{HAR}\left(r\right) = tansig(n_2) = \frac{2}{1+exp\left(-2n_2\right)} - 1, \quad r = 1, 2, \ldots, 30. \tag{3.49}$$

where

$$n_2 = \sum_{q=1}^{200} W2_{HAR}(q,r)O1_{HAR}(q) + b2_{HAR}(r) \tag{3.50}$$

Let the activation function of the output layer be the log-sigmoid transfer function and then the output of its *l*th node $O3_{HAR}\left(l\right)$ is expressed as:

$$O3_{HAR}\left(l\right) = tansig(n_3) = \frac{1}{1+exp\left(n_3\right)}, \quad l = 1. \tag{3.51}$$

where

$$n_2 = \sum_{r=1}^{30} W3_{HAR}(r)O2_{HAR}(r) + b3_{HAR} \tag{3.52}$$

the structure is shown as Fig.3.9.

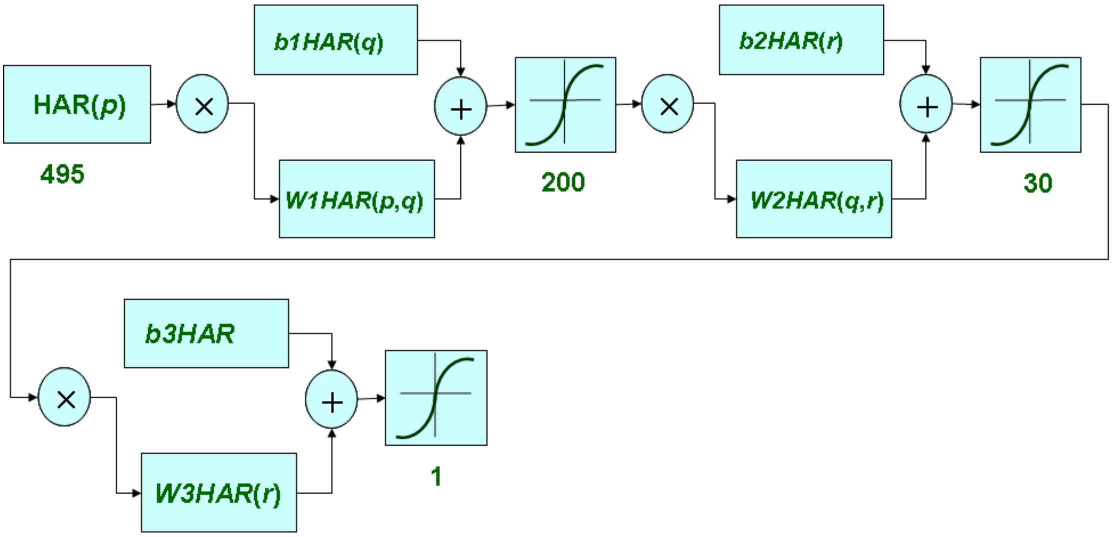


Fig. 3.9 HARNN

For GAVNN, there include one input layer with 162 nodes, three hidden layers with 200, 30 and 30 nodes, and one output layer with 1 node. There are 162 components of the GAV feature vector and all the components, represented by GAV(p), p=1,2,…,162, are sent into the 162 nodes of the input layer, correspondingly. The pth input node is connected to the qth node of the first hidden layer with weighting $W1_{GAV}(p,q)$. Hence, there exists a weighting array $W1_{GAV}(p,q)$ of dimension 200x162, p=1,2,…,162 and q=1,2,…,200. Besides, the qth node of the first hidden layer is also added with an extra bias $b1_{GAV}(q)$, q=1,2,…, 200. Similarly, The qth node of the first hidden layer is connected to the rth node of the second hidden layer with weighting $W2_{GAV}(q,r)$, which results in a weighting array $W2_{GAV}(q,r)$ of dimension 30x200, q=1,2,…,200 and r=1,2,…,30. An extra bias $b2_{GAV}(r)$, r=1,2,…,30, is added to the rth node of the second hidden layer. The rth node of the second hidden layer is connected to the output node with weighting $W3_{GAV}(r,l)$, r=1,2,…,30, and an extra bias $b3_{GAV}(l)$, r=1,2,…,30, is added to the lth node of the third hidden layer. Finally, the lth node of the third hidden layer is connected to the output node with weighting $W4_{GAV}(l)$, l=1,2,…,30, and a bias $b4_{GAV}$ is added to the output node.

Let the activation function of the first hidden layer be the hyperbolic tangent sigmoid transfer function and then the output of its *q*th node $O1_{GAV}\left(q\right)$ is expressed as:

$$O1_{GAV}\left(q\right) = tansig(n_1) = \frac{2}{1+exp\left(-2n_1\right)} - 1, \quad q = 1, 2, ...200\,. \tag{3.53}$$

where

$$n_1 = \sum_{p=1}^{162} W1_{GAV}(p,q)GAV(p) + b1_{GAV}(q)$$

Similarly, for the second hidden layer its *r*th node $O2_{GAV}\left(r\right)$ is expressed as:

$$O2_{GAV}\left(r\right) = tansig(n_2) = \frac{2}{1+exp\left(-2n_2\right)} - 1, \quad r = 1, 2, ..., 30\,. \tag{3.54}$$

where

$$n_2 = \sum_{q=1}^{200} W2_{GAV}(q,r)O1_{GAV}(q) + b2_{GAV}(q) \tag{3.55}$$

For the third hidden layer its *l*th node $O3_{GRD}\left(l\right)$ is expressed as:

$$O3_{GAV}\left(l\right) = tansig(n_3) = \frac{2}{1+exp\left(-2n_3\right)} - 1, \quad l = 1, 2, ..., 30\,. \tag{3.56}$$

where

$$n_3 = \sum_{r=1}^{30} W3_{GAV}(r,l)O2_{GAV}(r) + b3_{GAV}(r) \tag{3.57}$$

Let the activation function of the output layer be the log-sigmoid transfer function and then the output of its *l*th node $O4_{GRD}\left(m\right)$ is expressed as:

$$O4_{GAV}\left(m\right) = tansig(n_4) = \frac{1}{1+exp\left(n_4\right)}, \quad m = 1\,. \tag{3.58}$$

where

$$n_4 = \sum_{l=1}^{30} W4_{GAV}(l)O3_{GAV}(l) + b4_{GAV} \tag{3.59}$$

The structure is shown as Fig. 3.10.

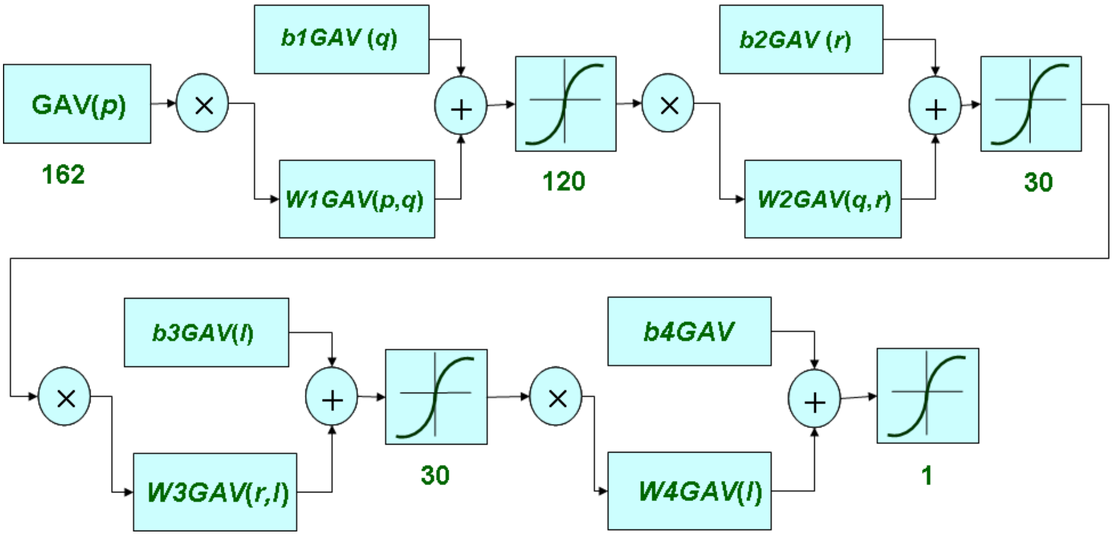


Fig. 3.10 GAVNN

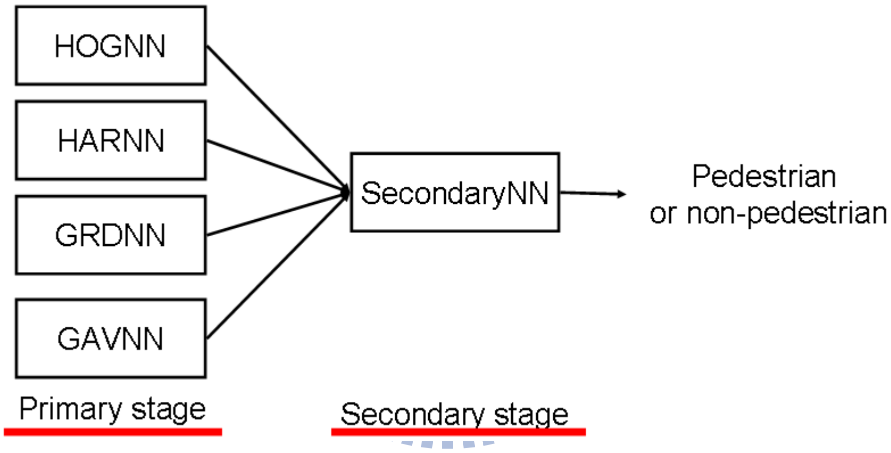


Fig. 3.11 Proposed structure

For the secondary stage, the output of each primary neural network is feeding into the secondary neural network to obtain final output, whose structure is shown in Fig.3.11. The DaimlerChrysler dataset [22] is used for the training of the proposed structure. The DaimlerChrysler dataset contains human images and non-human images cropped into 18 x 36 pixel, which has been widely used as pedestrian detection benchmark dataset. Some samples of the dataset are shown in Fig. 3.12. The related information of the dataset is shown in Table 3.1, which includes three sets of training data and two sets of test data. This thesis only adopts two sets of training data and one set of test data for demonstration.

Dataset Name	DaimlerChrysler Pedestrian Classification Benchmark Dataset
Web site	http://www.science.uva.nl/research/isla/downloads/pedestrians/
Training data	4800 x 3 human images 5000 x 3 non-human images
Test data	4800 x 2 human images 5000 x 2 non-human images
Image size	18 x 36 pixels

Table 3.1 Pedestrian detection benchmark dataset

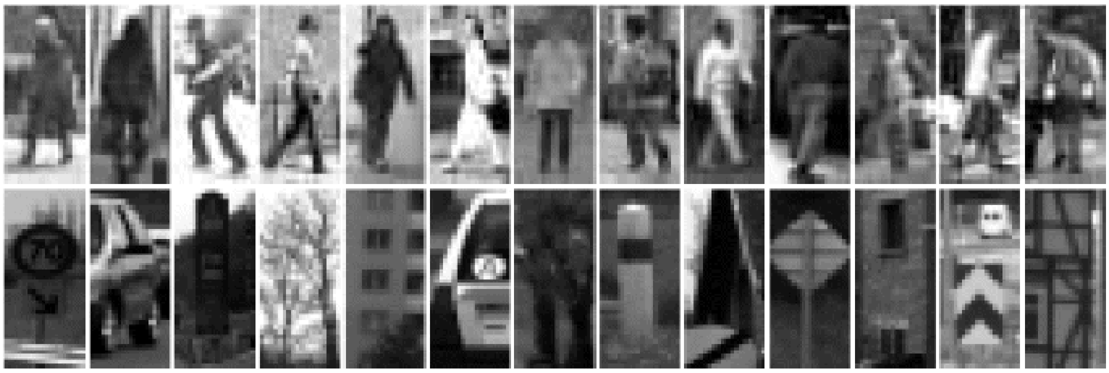


Fig. 3.12 The DaimlerChrysler dataset. Upper rows are images of humans and lower rows are images of non-humans.

With the training dataset, the features HOG, GRD, HAR and GAV are generated for the training of the primary stages to obtain the HOGNN, GRDNN, HARNN and AVNN, respectively. The training method is based on back-propagation, which has been briefly described in Chapter 2.

After the primary stages are created, to generate secondary stage, the HOG feature, GRD feature, HAR feature and GAV feature are feeding into HOGNN, GRDNN, HARNN and GAVNN. For secondary stage, there include one input layer with 4 nodes, one hidden layers with 40 nodes, and one output layer with 1 node. There are 4 components from the primary stage, represented as Pri(*p*), *p*=1,2,3,4, are sent into the 4 nodes of the input layer, correspondingly. The *p*th input node is connected to the *q*th node of the first hidden layer with weighting $W1_{SEC}(p,q)$. Hence, there exists a weighting array $W1_{SEC}(p,q)$ of dimension 40x4, p=1,2,3,4 and q=1,2,…,40. Besides, the *q*th node of the hidden layer is also added with an extra bias $b1_{SEC}(q)$, q=1,2,…,40. Finally, the *q*th node of the hidden layer is connected to the output node with weighting $W2_{SEC}(q)$, q=1,2,…,40, and a bias $b2_{SEC}$ is added to the output node.

Let the activation function of the hidden layer be the hyperbolic tangent sigmoid transfer function and then the output of its *q*th node $O1_{SEC}(q)$ is expressed as:

$$O1_{SEC}(q) = tansig(n_1) = \frac{2}{1+exp(-2n_1)} - 1, \quad q = 1,2,...40\,. \tag{3.60}$$

where

$$n_1 = \sum_{p=1}^{4} W1_{SEC}(p,q)SEC(p) + b1_{SEC}(q) \tag{3.61}$$

Let the activation function of the output layer be the log-sigmoid transfer function and then the output of its *r*th node $O2_{SEC}(r)$ is expressed as:

$$O2_{SEC}(r) = tansig(n_3) = \frac{1}{1+exp(n_3)}, \quad \mathrm{r} = 1. \tag{3.62}$$

where

$$n_3 = \sum_{r=1}^{40} W2_{SEC}(r)O1_{SEC}(r) + b2_{SEC} \tag{3.63}$$

The structure is shown as Fig. 3.13.

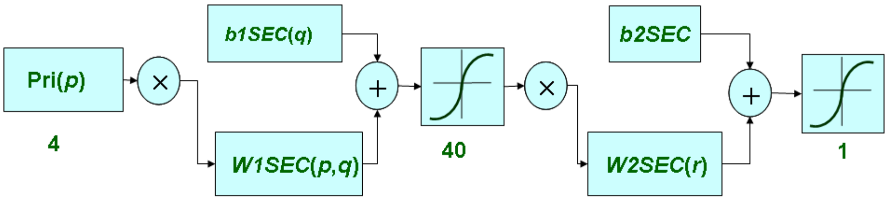


Fig. 3.13 secondary stage

For secondary stages, the input vector is 4x1 which is generated by primary stage. The training method is based on back-propagation, which has been briefly described in Chapter 2.

Chapter 4

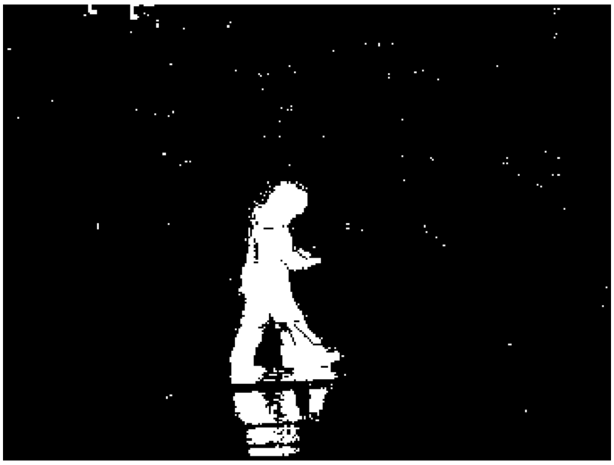
Experiment results

In this chapter, the results of the proposed algorithm will be obtained by MATLAB R2007a and Opencv C++ and then compared them to the results of other algorithms. For 4.1 the pedestrian detect system result will be shown. For 4.2 the performance definition and the performance of the system will be discussed.

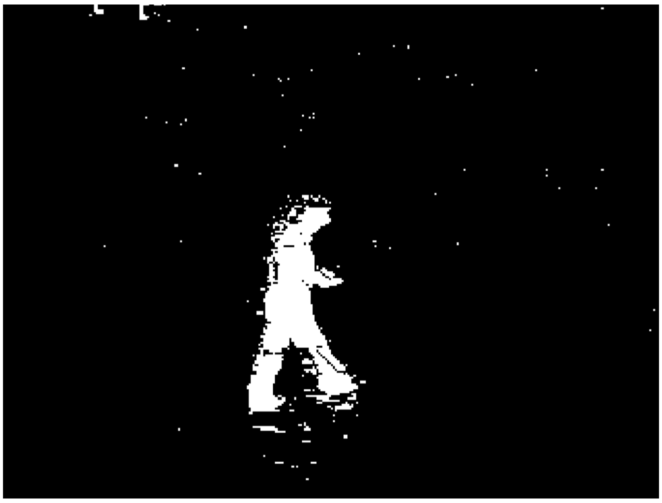
4.1 Moving object detection

In the result of moving object detection, the test data is from [23] which contain some pedestrian walking. The result will show how pedestrian is found in the image. First, the foreground segmentation is to find out the moving objects different from background, after the foreground segmentation the objects belong to background will be eliminated shown as Fig. 4.1(a). In the real situation there will be a lot of shadow in the scene, because the shadow will reduce the accuracy of classifying, shadow removal is applied to remove the shadow shown as Fig. 4.1(b). After shadow removal, there are still many noise in the scene which is caused by temperate illumination change or shadow removal, to reduce the noise, morphology operation is applied to eliminate small area and to clear the connect region shown as Fig. 4.1(c). Therefore, the possible area will be obtained from the image. To find out how many objects and the boundary of each object in the scene, image projection is applied, and then the position of objects will be extract shown as Fig. 4.1(d). When there are two objects in the scene, the result of image projection for one of the object will be shown as Fig. 4.1(e). After the objects are extracted, the feature extraction is applied to extract feature which are going to feed into neural network. And then use neural network to recognize whether the object is pedestrian or not. If the object belong to pedestrian the system will mark it with green rectangular which shown as Fig. 4.1(f).

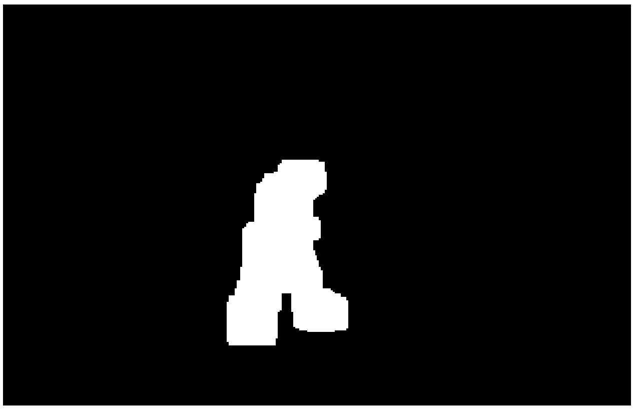
For the moving object detection, the parameter used in the system is shown as Fig. 4.1(g).



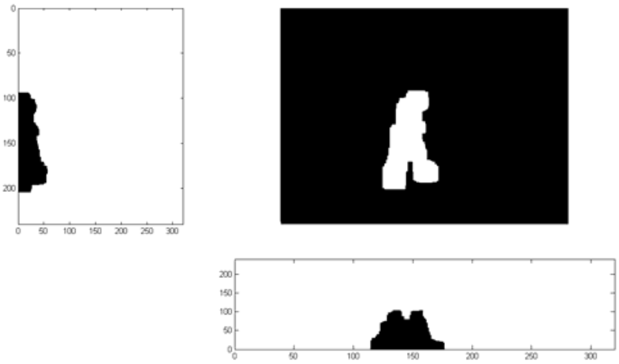
(a)foreground segmentation



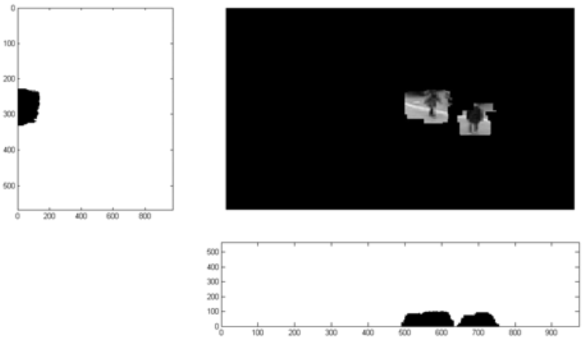
(b)shadow removal



(c)morphology operation



(d)image projection for one people



(e)image projection for two people

(f)classify result

parameter	value
Th_{fd}	2
Th_{tol}	2
β	3
τ_{H}	0.2
τ_{S}	0.2
α	1

(g) the parameter in the system

Fig. 4.1

4.2 Pedestrian recognition

To compare the performance of the detection, the thesis first introduces the Receiver Operating Characteristic [24] curve and defines the accuracy rate, and then discusses four cases concerning the pedestrian recognition. In Case 1, consider five neural networks, where four of them are trained by the features of HOG, Haar, Global average and Gradient, called the primary neural networks(PNNs in brief), and the other one is trained by all these four features, called the combined neural network(CNN in brief). Because the recognition performance of the PNNs can not be improved by the CNN, it is required to employ the two-staged neural network proposed in Chapter 3. In Case 2, with the use of two-staged neural network, the performance is indeed improved, which is judged from the ROC curve and the accuracy rate. In addition to the recognition performance, the proposed two-staged neural network can also reduce the amount of the training data, which is discussed in Case 3. As for the executing time in on-line process, even the two-staged neural network requires twice or triple amount of time when compared to the PNNs, its accuracy rate is increased about 2% to 10%, shown in Case 4.

In general, the major objective of a detection system is to recognize pedestrians form an image or a sequence of images. In pedestrian recognition, there are four possible events given in Table 4.1, including True Positive(TP), True Negative(TN), False Positive(FP), and False Negative(FN). Judging from the actual condition and test result, these four events are listed as below:

1. True Positive, TP, means a real pedestrian is detected by the neural network as pedestrian.
2. True Negative, TN, means a non-pedestrian is detected by the neural network as non-pedestrian.
3. False Positive, FP, means a non-pedestrian is detected by the neural network as pedestrian.
4. False Negative, FN, means a real pedestrian is detected by the neural network as non-pedestrian.

With these four events, the detect rate *DR* and false positive rate *FPR* can be respectively defined as below:

$$\mathrm{DR}=\frac{\mathrm{TP}}{\mathrm{TP+FN}}\times 100\%$$

$$\mathrm{FPR}=\frac{\mathrm{FP}}{\mathrm{TN+FP}}\times 100\%$$

A detect rate of 100% means all pedestrians are detected correctly, while a false positive rate of 0% means any non-pedestrian is not detected as a pedestrian. To compare the performance of the system, the accuracy rate *AR* is defined as below:

$$AR=\frac{TP+TN}{TP+TN+FP+FN}\times 100\%$$

and a higher accuracy rate implies a better recognition performance.

		Actual Condition	
		1	0
Test Result	1	TP	FP
	0	FN	TN

Table 4.1 TP, FP, FN, TN table

<u>Case 1</u>: PNN and CNN

In this case, the combined neural network CNN is compared to the four primary neural networks PNNs. The input vector of the CNN is defined as

$$CNN = \left[HOG \quad HAR \quad GAV \quad GRD \right]$$

where $HOG \in \Re^{1152\times 1}$, $HAR \in \Re^{495\times 1}$, $GAV \in \Re^{162\times 1}$ and $GRD \in \Re^{494\times 1}$ are the feature vectors of HOG, Haar, Global average and Gradient, and $CNN \in \Re^{2314\times 1}$ contains all these four features. the combined structure is shown in Fig. 4.2 which include one input layer with 2314 nodes, two hidden layers with 100 and 30 nodes, and one output layer with 1 node. There are 2314 components of the CNN feature vector, represented by CNN(*p*), *p*=1,2,…,2314, are sent into the 2314 nodes of the input layer, correspondingly. The *p*th input node is connected to the *q*th node of the first hidden layer with weighting $W1_{CNN}(p,q)$. Hence, there exists a weighting array $W1_{CNN}(p,q)$ of dimension 100x2314, *p*=1,2,…,2314 and *q*=1,2,…,100. Besides, the *q*th node of the first hidden layer is also added with an extra bias $b1_{CNN}(q)$, *q*=1,2,…,100. Similarly, The *q*th node of the first hidden layer is connected to the *r*th node of the second hidden layer with weighting $W2_{CNN}(q,r)$, which results in a weighting array $W2_{CNN}(q,r)$ of dimension 30x100, *q*=1,2,…,100 and *r*=1,2,…,30. An extra bias $b2_{CNN}(r)$, *r*=1,2,…,30, is added to the *r*th node of the second hidden layer. Finally, the *r*th node of the second hidden layer is connected to the output node with weighting $W3_{CNN}(r)$, *r*=1,2,…,30, and a bias $b3_{CNN}$ is added to the output node.

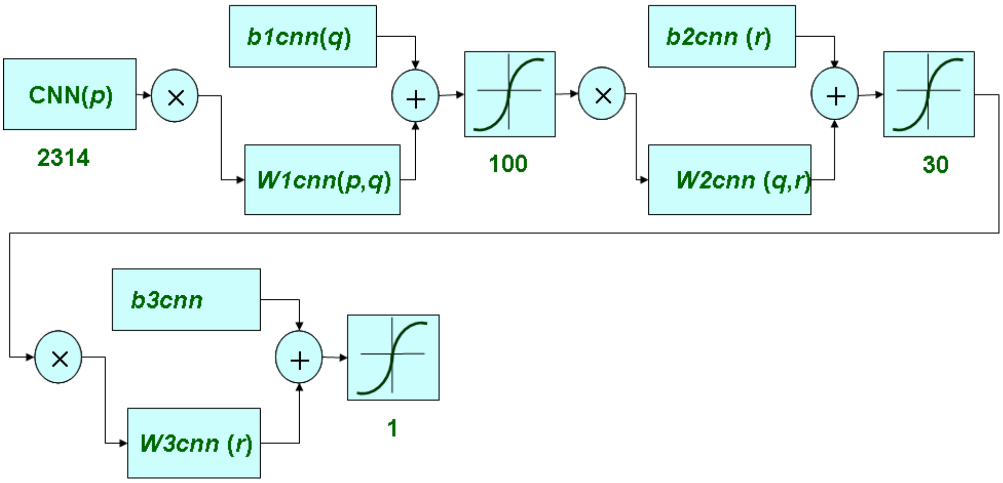


Fig. 4.2 CNN

For training the CNN, because the dimension of the CNN is very big the memory caused is very large. Due to the memory concerning, the thesis first trained the primary stage use 1000 training data which contain 500 pedestrian data and 500 non-pedestrian data. And the test data has 9000 data which contain 4500 pedestrian and 4500 non-pedestrian. For primary stage, there are four primary stage should be trained. For HOGNN, there include one input layer with 1152 nodes, two hidden layers with 100 and 30 nodes, and one output layer with 1 node. For GRDNN, there include one input layer with 494 nodes, three hidden layers with 120, 30 and 30 nodes, and one output layer with 1 node. For HARNN, there include one input layer with 495 nodes, two hidden layers with 200 and 30 nodes, and one output layer with 1 node. For GAVNN, there include one input layer with 162 nodes, three hidden layers with 200, 30 and 30 nodes, and one output layer with 1 node. The transfer function of the hidden layer is hyperbolic tangent sigmoid. The transfer function of the output layer is log-sigmoid transfer function. The ROC curve which has lower false positive rate and higher detect rate the performance is better. The Fig 4.3 shows the ROC curve of the primary stage and the accuracy of the primary stage is also label at figure. Clearly, the HOGNN has better performance than other primary stage. The thesis compared the HOGNN with the CNN to observe if the performance is better.

The Fig 4.4 shows the ROC curve and accuracy of the HOGNN and CNN. Clearly, the performance became worse. To use the multi-feature the thesis use another way to combine the feature which will be discuss in Case 2.

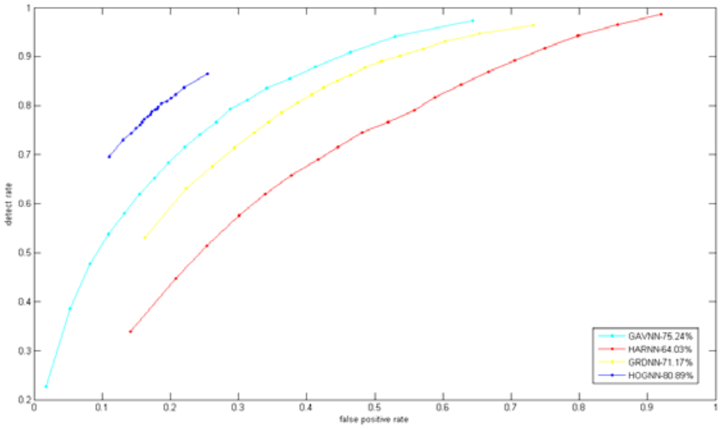


Fig. 4.3 The roc curve of primary stage

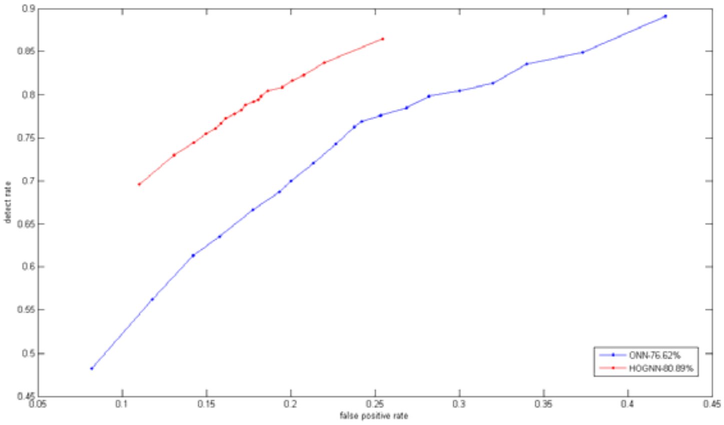


Fig. 4.4 The roc curve of HONN and CNN

<u>Case 2</u>: Two-staged neural network and PNNs

To improve the performance the thesis considers another way to combine the feature. The proposed method combines the feature using 4 neural networks as the primary stages and 1 neural network as the secondary stage. For primary stage, the neural network structure mentioned at previous case is used. For secondary stage, there include one input layer with 4 nodes, one hidden layers with 40 nodes, and one output layer with 1 node. The transfer function of the hidden layer is hyperbolic tangent sigmoid. The transfer function of the output layer is log-sigmoid transfer function. Because the dimension of the secondary neural network has only 4 inputs, there is lesser memory concerning. The training data of the Case 2 use two kinds of sub-case. For the first sub-case 15000 training data which contain 7500 pedestrian data and 7500 non-pedestrian data. And the test data has 9000 data which contain 4500 pedestrian and 4500 non-pedestrian. The Fig. 4.5 shows the ROC curve of the primary stage and Table 4.2 shows the accuracy of the primary stage. Clearly, the HOGNN has better performance than other feature-NN. The Fig 4.6 shows the ROC curve of the proposed method and HOGNN. And the accuracy of the proposed method is 88.24%. For the second sub-case 1000 training data which contain 500 pedestrian and 500 non-pedestrian. And the test data has 9000 data which contain 4500 pedestrian and 4500 non-pedestrian. The Fig 4.3 shows the ROC curve and accuracy of the primary stage. Clearly, the HOGNN has better performance than other feature-NN. The Fig 4.7 shows the ROC curve and accuracy of the proposed method and HONN. Clearly, the proposed method has better performance than other method. It improves the accuracy. To show the training data the proposed method can reduce. The thesis uses many kinds of training result to observe.

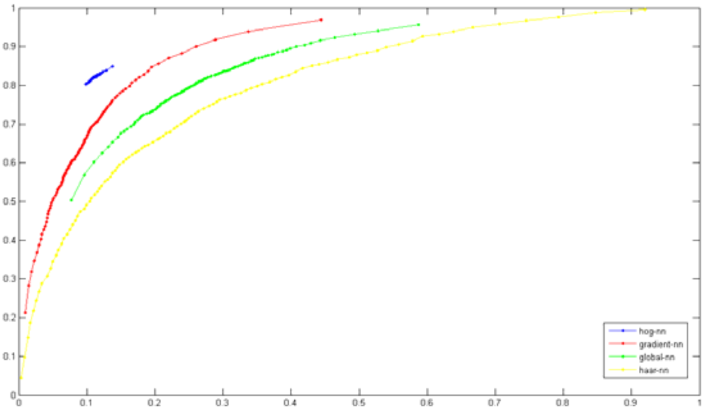


Fig. 4.5 Primary stage for 15000 training data

Feature-NN	Accuracy
Hog-neural network	85.57%
Gradient-neural network	77.28%
Global average-neural network	82.23%
Haar-neural network	73.49%

Table 4.2 The accuracy of feature-NN

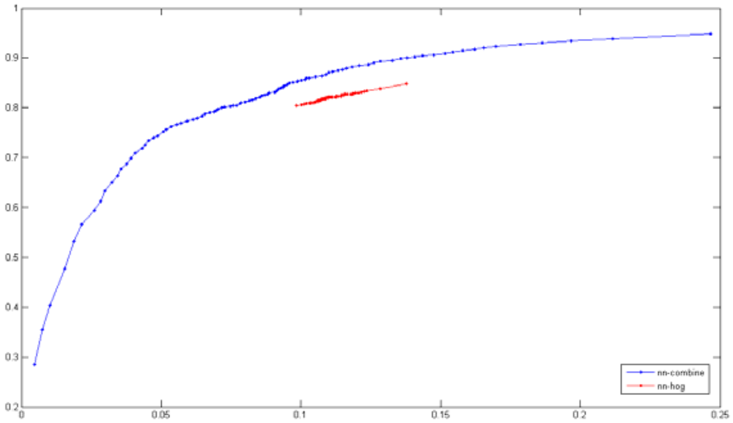


Fig. 4.6 The roc curve of two-staged neural network and HOGNN with 15000 training data

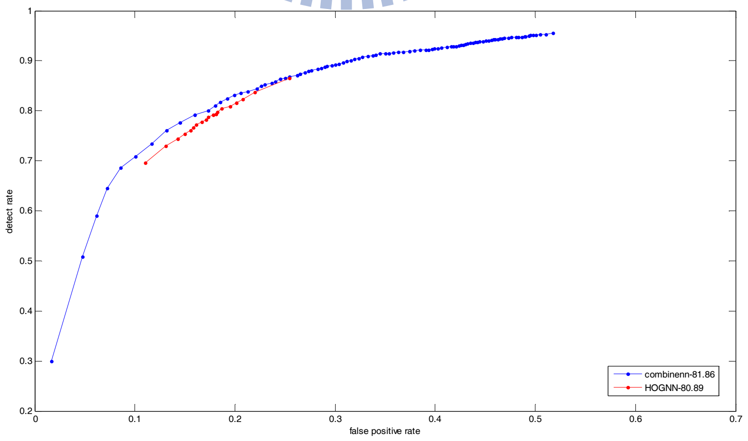


Fig. 4.7 The roc curve of two-staged neural network and HOGNN with 1000 training data

<u>Case 3</u>: Reduced the Training data

The thesis uses many kinds of training data to compare the ROC curve and the accuracy of the HOGNN. There are three kinds of training data: 7500, 5000 and 1000. The training data is collect from the dataset which contain half pedestrian and half non-pedestrian. The test data has 9000 data which contain 4500 pedestrian and 4500 non-pedestrian. The structures of three HOGNN are all the same as mentioned before. For Fig 4.8 shows the ROC curve and the accuracy of the HOGNN with different training data. It shows that the HOGNN's accuracy and ROC curve will increase slower with training data increase. Compare the 5000 with 7500 the training data increase 2500 but the accuracy only increase 1%. For Fig 4.9 the proposed method uses only 7500 training data to achieve almost the same accuracy which use 15000 training data. It reduce 7500 training pattern to achieve the same accuracy and ROC curve.

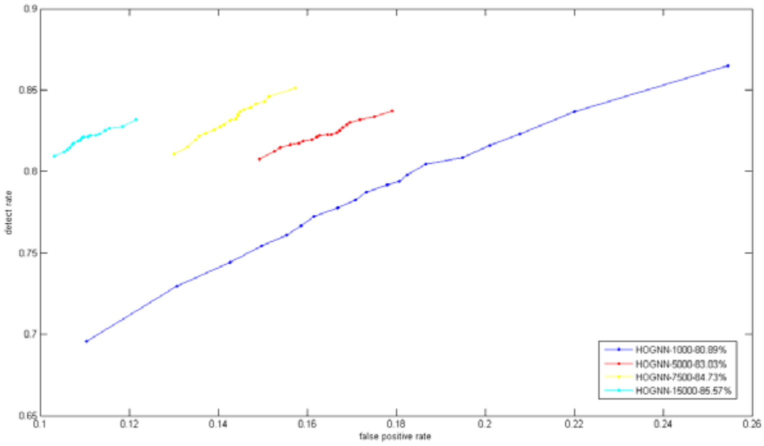


Fig. 4.8 the HONN roc curve with different amount of training data

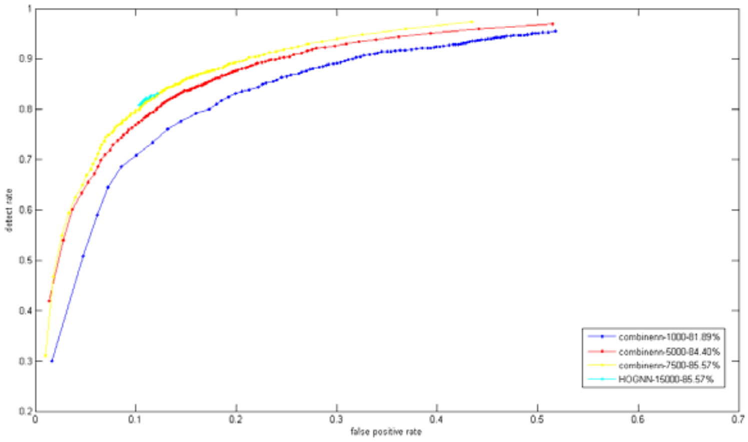


Fig. 4.9 the roc curve of the two-staged neural network with different amount of training data and HOGNN with 15000 training data

<u>Case 4</u>: Executing time

In the time consume issue, because the proposed method needs to extract multi-features, it will also cost more time to generate the features. The process time is show as Table 4.3. Clearly, the proposed method cost more executing time than other method. The accuracy is better than other method.

Feature_NN	Accuracy	Extraction time	NN process time	Total process time
Gradient_NN	77.28%	0.01sec	0.01sec	0.02sec
HOG_NN	85.57%	0.01sec	0.03sec	0.04sec
Propose_NN	88.24%	0.027sec	0.05sec	0.077sec

Table 4.3 The executing time and accuracy of two-staged neural network and some of the primary stages

Chapter 5

Conclusions and Future Works

In the thesis a new method is presented in using multi-feature. The proposed method uses a two stage neural network which includes primary stage and secondary stage. For the primary stage, it is generated by many features. There are four primary stage used in the thesis. Each primary stage neural network use different features to train. For the secondary neural network, it is to combine all the output from the primary stage. From the experiment result, this kind of two stage neural network can improve accuracy rate and most importantly, it is workable in the case that a smaller amount of training data is used. Also, the executing time is close to other method with higher accuracy rate.

To improve the proposed system, it is important to find out other feature which has better accuracy rate. There is another feature named Gabor which can improve the proposed method. The ROC curve and accuracy rate is shown as Fig. 6.1. In the application, the proposed application can only used in static environment. To use in dynamic environment, stereo algorithm must be implement. In Fig. 6.2 shows an intelligent way to remove the ground using neural network. There is some problem should be solved before using into the system.

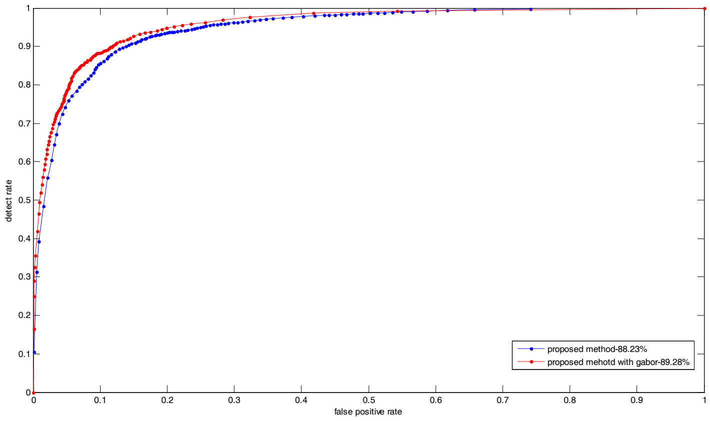
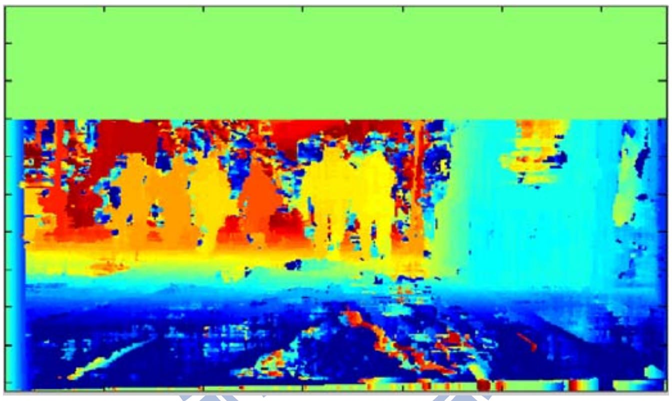
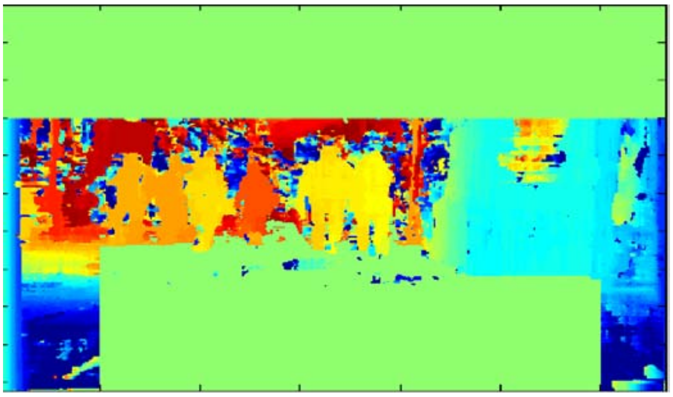


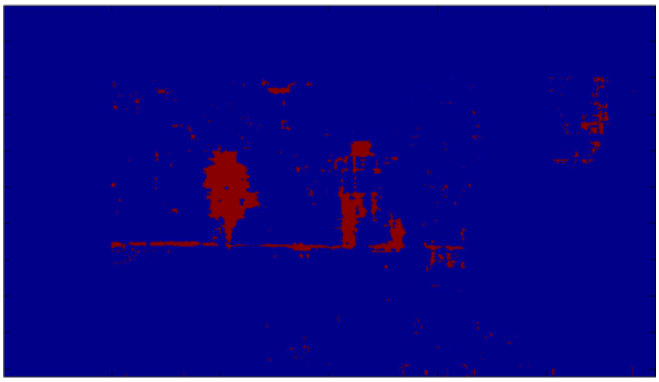
Fig. 5.1 The roc curve of the two-staged neural network and the two-staged neural network with gabor feature added



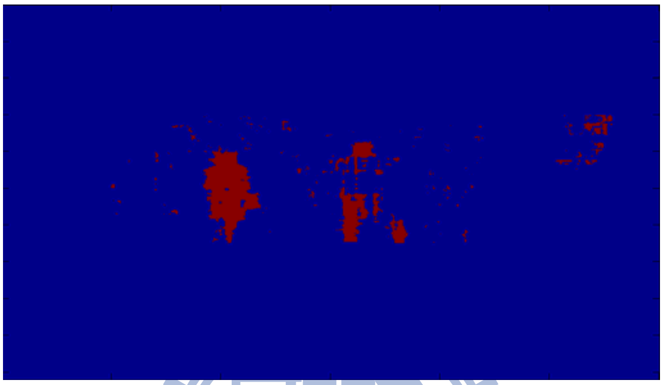
(a)



(b)



(c)



(d)

Fig. 5.2 (a) the stereo image before ground removed (b) the stereo image before ground removed (c) the pedestrian extract before ground removed (d) the pedestrian extract after ground removed

Reference

[3] L. Zhao and C. Thorpe. “Stereo- and neural network-based pedestrian detection.” *IEEE Trans. on ITS*, 1(3), 2000.

[4] Y.-T. Chen and C.-S. Chen. “Fast human detection using a novel boosted cascading structure with meta stages.” *IEEE TIP*, 17(8):1452–1464, 2008.

[5] J. Wang et al. “An Adjacent Multiple Pedestrians Detection BASED on ART2 Neural Network.” *ISNN 2006, LNCS 3972*, pp. 244-252, 2006

[6] D. Duque, H. Santos, and P. Cortez. “Moving Object Detection Unaffected by Cast Shadows, Highlights and Ghosts.” *IEEE International Conference image processing*, 2005

[7] Li-Qun Xu, Jose Luis Landabaso, Montse Pardas, “Shadow Removal with Blob-based Morphological Reconstruction for Error Correction” *IEEE ICASSP*. vol. 4, No. 5, 2005.

[8] Heikkilä, J. and O. Silvén, “A real-time system for monitoring of cyclists and pedestrians,” *IEEE Workshop on Visual Surveillance*, Fort Collins, CO, Jun.26, 1999, pp.74-81.

[9] Cédras, C. and M. Shah, “Motion-based recognition: a survey,” *Image and Vision Computing*, vol.13, No.2, pp.129-155, March 1995.

[10] A. Neri, S. Colonnese, G. Russo, and P. Talone, “Automatic moving object and background separation,” *Signal Process*. 66, 219-232(1998).

[11] S. Chien, Y. Huang, and L. Chen, “Predictive watershed: a fast watershed algorithm for video segmentation,” *IEEE Trans. Circuits Syst. Video Technol.* 13(5), 453-461 (2003)

[12] S. Chien, S. Ma, and L. Chen, “Efficient moving object segmentation algorithm using background registration technique,” *IEEE Trans. Circuits Syst. Video Trchnol*. 12(7), 577-586 (2002).

[13] Collins, R. T., A. J. Lipton, and T. Kanade, “A System for Video Surveillance

and Monitoring", *Technical Report*, CMU-RI-TR-00-12, Robotics Institute, Carnegie Mellon University, May 2000.

[14] Kim et al., "Real-time disparity estimation using foreground segmentation for stereo sequences." *Optical Engineering* 45(3), 037402 (2006)

[15] R.Cucchiara, C. Grana, M. Piccardi, A. Prati, and S.Sirotti, "Improving shadow suppression in moving object detection with HSV color information." *IEEE Int'l Conference on Intelligent Transportation Systems*, Aug. 2001,pp.334-339.

[16] P. Kumar, K. Sengupta, and S. Ranganath, "Real time detection and recognition of human profiles using inexpensive desktop cameras." *in Proc. ICPR'00*, pp. 1096-1099, *IEEE Computer Soc.*, (2000).

[17] A. Shashua, Y. Gdalyahu, and G. Hayun, "Pedestrian detection for driving assistance systems: Single-frame classification and system level performance." *in Proc. IEEE Intell. Vehicles Symp.* Jun. 2004, pp. 1-6.

[18] N. Dalal and B. Triggs. "Histograms of oriented gradients for human detection." *IEEE Comput. Soc. Conf. Comput. Vision and Pattern Recogn.*, 2005.

[19] Q. Zhu, C. Yeh, T. Cheng, and S. Avidan, "Fast human detection using a cascade of histograms of oriented gradients," *IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2006, pp. 1491-1498

[20] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," *IEEE Proc. Int. Conf. Computer Vision and Pattern Recognition*, 2001, vol. 1, pp. 511-518.

[21] A. Khashman., "Intelligent Face Recognition: Local Versus Global Pattern Averaging." *Berlin Springer*, 2006.

[22] S. Munder, D.M. Gavrila: An experimental study on pedestrian classification. *IEEE Trans. Pattern Anal*. Mach. Intell. 28(11), 1863-1868 (2006)

[23] http://groups.inf.ed.ac.uk/vision/CAVIAR/CAVIARDATA1/

[24] D.J. Hand, R.J. Till. "A simple generalization of the area under the ROC curve to multiple class classification problems." *Machine Learning*, 45, 171-186. (2001).