Pedestrian Detection and Tracking

Chatchai Suppitaksakul

A thesis submitted for fulfilment of the requirements
of Northumbria University for the degree of
Doctor of Philosophy

Research undertaken in the School of Computing,
Engineering & Information Sciences

June 2006

Abstract

This report presents work on the detection and tracking of people in digital images. The employed detection technique is based on image processing and classification techniques. The work uses an object detection process to detect object candidate locations and a classification method using a Self-Organising Map neural network to identify the pedestrian head positions in an image. The proposed tracking technique with the support of a novel prediction method is based on the association of Cellular Automata (CA) and a Backpropagation Neural Network (BPNN). The tracking employs the CA to capture the pedestrian's movement behaviour, which in turn is learned by the BPNN in order to the estimated location of the pedestrians movement without the need to use empirical data. The report outlines this method and describes how it detects and identifies the pedestrian head locations within an image. Details of how the proposed prediction technique is applied to support the tracking process are then provided. Assessments of each component of the system and on the system as a whole have been carried out. The results obtained have shown that the novel prediction technique described is able to provide an accurate forecast of the movement of a pedestrian through a video image sequence.

Acknowledgements

The work presented in this report has been carried out in the School of Computing, Engineering and Information Sciences, Northumbria University. Firstly, I would like to express my sincere gratitude to my director of studies, Dr. Graham Sexton and my second supervisor, Dr. Peter Minns for assisting me in every way possible. Throughout my study, I have had the invaluable expertise, guidance and support from them. I would like to thank Rajamangala University of Technology Thanyaburi (RMUTT) and Thai Government for their support and funding for this study.

Also I would like to thank my friends, especially Dr. Mark Paul Leach, Dr. Karel Sterckx and Dr. Souvapak Suktakulvas for kindly giving me suggestions towards academic writing in English. Besides this, I would like to thank Dr. Wanatana Panupintu for help in Mathematics solution and Dr. Wai Pang Ng for kindly providing me a viva rehearsal.

Without the encouragements and supporting from my lovely son (Papinwit), my family and friends, especially Mr. Wisit Leodrammachakra, Mrs. Vincci Lee Ng, Mr. Chong Ng, Dr. Prarichat Sterckx, and Makulsawatudom family (Aung Aon and O), this work would not have been a success. I really like to thank for them.

DECLARATION

I hereby declare that this thesis is entirely my own work and has not been submitted in support of an application of another degree or qualification of this or any other university, institute of learning or industrial organisation

Signature:	
Name:	Chatchai Suppitaksakul
Date:	29/11/06

2-D	Two dimensions
3-D	Three dimensions
1D-CA	One dimensional Cellular Automata
1-DWT	One dimensional Discrete Wavelet Transform
2D-CA	Two dimensional Cellular Automata
2-DWT	Two dimensional Discrete Wavelet Transform
ANN	Artificial Neural Network
BPNN	Backpropagation Neural Network
CA	Cellular Automata
CCM	Classification and Clustering Module
DWT	Discrete Wavelet Transform
FEM	Feature Extraction Module
MD	Mean Deviation
MDF	Model-specified Directional Filter
NN	Neural Network
PCA	Principal Component Analysis
PDC	Pedestrian Detection based Classification
PM	Prediction module
SOM	Self-Organising Map
SVM	Support Vector Machine
WT	Wavelet Transform

$I_{back}(i,j)$	Background Image
$I_{binary}(i,j)$	Binary Image
$I_{ped}(i,j)$	Pedestrian Image
$I_{sub}(i,j)$	Background-Subtracted Image
I_{mean}	Mean value of the image
I_{std}	Standard deviation value of the image
T_{lower}	Lower Threshold
T_{upper}	Upper Threshold
$Ph(i)$	Horizontal Projection
$Pv(j)$	Vertical Projection
$P_{smoothed}$	Smoothed Projection
$\mathbf{H}$	Low-pass filter component
$\mathbf{G}$	High-pass filter component
$\downarrow 2$	Downsampling
A^j	Approximation or Low-Pass Residue Coefficient
D^j	Wavelet Detail Coefficient
$\phi(t)$	Scaling Function
$\psi(t)$	Wavelet Function
E_n^j	Energy value of Wavelet Coefficient
MD_n^j	Mean Deviation value of Wavelet Coefficient
Md_i^W	Feature-Weighting
S_d	Standard Deviation Value
$\bar{M}$	Mean Value
α	Variance
β	Inversely proportional to the Decreasing rate of Histogram Peak
Γ	Gamma Function
λ	Eigenvalue

FV_{eng}	Energy FeatureVector
FV_{fw}	Feature-Weighting Vector
FV_{hist}	Histogram Feature Vector
FV_{eig}	Eigenvector Feature Vector
$h_{ci}(t)$	Neighbourhood Function
$\alpha(t)$	Scalar Valued Learning Rate
$\sigma(t)$	Kernel Width
dH_f	Distance between Input and Head Codebook Vector
dNH_f	Distance between Input and Non-Head Codebook Vector
f_{in}	Input Feature Vector
H_{fcode}	Head Codebook Feature Vector
NH_{fcode}	Non-Head Codebook Feature Vector
θ	The direction of the movement
Δx	Distance between two points in horizontal (column)
Δy	Distance between two points in vertical (row)

List of Figures

Figure 2.1 A side view image 8

Figure 2.2 A top-down view image 9

Figure 2.3 Principal component analysis on a data set of pedestrians represented by B-splines; shown is the shape variation along the principal component [19] 13

Figure 2.4 The tracking results from Tissainayagam and Suter [22] 14

Figure 2.5 The exampe outcome from Kang *et al.* [23] 15

Figure 2.6 The convex hull approximation from Jae-Won Kim *et al.* [34] 21

Figure 3.1 Object detection block diagram 24

Figure 3.2 The background subtraction process 26

Figure 3.3 The background-subtracted image and its histogram 26

Figure 3.4(a) Testing of two background subtracted image 27

Figure 3.4(b) Histogram of the background subtracted image 27

Figure 3.5 Histogarm of absolute background subtracted image 29

Figure 3.6 The subtracted image and its binary image 30

Figure 3.7 A binary image and its enhanced image 31

Figure 3.8 The co-ordinate generation process 32

Figure 3.9 Examples of image of people's walking-posture 33

Figure 3.10 The horizontal and vertical projections 35

Figure 3.11 The horizontal projection and its smoothed 37

Figure 3.12 The responses of pixel difference detector 38

Figure 3.13 The comparison of the second order response of horizontal projection using two types of smoothing technique 39

Figure 3.14 The example of the object detection outcomes with two methods 40

Figure 3.15 The process of co-ordinate classifying 41

Figure 3.16 The example of the object detection outcomes with two methods 42

Figure 3.17 The results of marked images with various numbers of gradient samples 42

Figure 3.18 The graph represents the result of adjusting the number of gradient samples 43

Figure 3.19 Image data that obtains from one window 45

Figure 3.20 Two test images for threshold adjusting 46

Figure 3.21 The results of threshold image, threshold= σ 46

Figure 3.22 The results of threshold image, threshold level at 2.5σ some residual background is still around the objects 47

Figure 3.23 The results of threshold image, threshold level at 3σ, 3.5σ and 5σ 48

Figure 3.24 Structuring element size 3-by-3 pixels and 7-by-7 pixels 49

Figure 3.25 The enchanced images using 3-by-3 structuring element 49

Figure 3.26	The enchanced images with 11-by-11 structuring element	50
Figure 3.27	The plotting of the data from Tables 3.6.3.2 and 3.6.3.3	54
Figure 3.28	Outcome of the smoothing process using 11 and 31 gradient samples of the figure 3.20 No.1 (left column)	55
Figure 3.29	Outcome of the smoothing process using 11 and 31 gradient samples of the figure 3.20 No.2 (right column)	55
Figure 3.30	The images for testing from two different sources of capturing	56
Figure 3.31	Demonstrates the number of co-ordinates correspond to the objects	60
Figure 3.32	Shows the co-ordinates that fail to plot on the object area	60
Figure 3.33	The detector fails to detect two objects in the image	61
Figure 3.34	The outcomes of the detector (Frame01 -Frame08)	63
Figure 3.34	The outcomes of the detector (Frame09 -Frame16 continue)	64
Figure 4.1	Diagram of Feature Extraction Module	68
Figure 4.2	The basic process of the one-scale 1-DWT	69
Figure 4.3	The 3-scale 1-DWT schemes	71
Figure 4.4	The process of the one-scale 2-DWT	71
Figure 4.5	The process of the two-scale 2-DWT	73
Figure 4.6	The three-scale 2-DWT decomposition: (a) original image; (b) first, (c) second and (d) third step	73
Figure 4.7	The two-scale 2-DWT of Lena's image	75
Figure 4.8	One energy feature obtained from the 2-scale 2-DWT image	77
Figure 4.9	One histogram feature obtained from 2-scale 2-DWT image	79
Figure 4.10	One Eigenvector feature obtained from 1-scale 2-DWT image	81
Figure 4.11	Head pattern images	83
Figure 4.12	Non-head pattern images	83
Figure 4.13	Feature Map using $FV_{eng}[1]$ VS $FV_{eng}[4]$ of Head and Non-head patterns	84
Figure 4.14	Feature Map using $FV_{fw}[4]$ VS $FV_{fw}[6]$ of Head and Non-head patterns	85
Figure 4.15	Feature Map using $FV_{hist}[1]$ VS $FV_{hist}[2]$ of Head and Non-head patterns	86
Figure 4.16	Feature Map using $FV_{eig}[1]$ VS $FV_{eig}[6]$ of Head and Non-head patterns	87
Figure 5.1	An overview of CCM	89
Figure 5.2	Structure of a 2-D Self-Organising Map	90
Figure 5.3	A 2-D SOM showing a square neighbourhood function which starts as $h_{ci}(t1)$ and reduces in size to $h_{ci}(t3)$ over time.	92
Figure 5.4	Illustrates 345 test-images of head pattern, which were captured from different frames of the sequential images	97
Figure 5.5	Illustrates 300 test-images of Non-head pattern, which were obtained from the image data	98
Figure 6.1	Shows the entire system of pedestrian detection based classification technique	101

Figure 6.2	The PDC results using 4 different features as (a) Energy feature, (b) Feature-weighting, (c) Histogram feature and (d) Eigenvector feature	102
Figure 6.3	The PDC results of an image Frame 02: (a) Energy feature, (b) Feature-weighting, (c) Histogram feature and (d) Eigenvector feature	103
Figure 6.4	The PDC results of the image that contains more pedestrians: (a) Energy feature, (b) Feature-weighting, (c) Histogram feature and (d) Eigenvector feature	104
Figure 6.5	The PDC results of the image Frame 02: (a) Energy feature, (b) Feature-weighting, (c) Histogram feature and (d) Eigenvector feature	105
Figure 7.1	A cell and its neighbours of 1D-CA	109
Figure 7.2	CA rule ***126***	110
Figure 7.3	The result of applying CA rules to two-state 1D-CA	111
Figure 7.4	The simulation of 20 steps of two-state one-dimensional CA	112
Figure 7.5	Simulation results by applying rule ***184*** (left) and ***30*** (right)	112
Figure 7.6	Two-dimensional cellular automata	113
Figure 7.7	Two-layer BPNN architecture with 3 inputs, 1 output and 3 hidden neurons	115
Figure 7.8	Log-sigmoid and linear transfer functions [85]	119
Figure 7.9	Unsuccessful training result of BPNN with three hidden neurons	120
Figure 7.10	Successful training result of BPNN with seven hidden neurons	120
Figure 7.11	BPNN simulation results	122
Figure 7.12	The recognising of Moore neighbourhood to BPNN inputs	124
Figure 8.1	A block diagram of the prediction module	128
Figure 8.2	The latticed images witn two different grid dimensions	129
Figure 8.3	The process of organizing the latticed image to vector form	130
Figure 8.4	An example of the test image for defining the grid dimension against the number of frame running per cell	133
Figure 8.5	The example of the latticed images using the static grid establishing that the grid is unable to cover the whole object	134
Figure 8.6	The example of the latticed images using the dynamic grid establishing that thegrid is able to cover the whole object at any locations within the image	135
Figure 8.7	The estabished grid of the interested object does not cover another object within the image	136
Figure 8.8	CA planes of each object by using dynamicgrid establishing	136
Figure 8.9	The test image with manully set for the location, direction: moving east and initial of moving speed: 10 frames per cell	138
Figure 8.10	The Von Neumann neighbourhood	138
Figure 8.11	The tracking results of the prediction method	139
Figure 8.12	The tracking results of the improved prediction method	141
Figure 8.13	The tracking results of the prediction method	142
Figure 8.14	The locations plotting of prediction tracking	143

Figure 8.15	The object detector outcome after improving	145
Figure 8.16	The test of the object detector on the image that contains more than one object	145
Figure 8.17	The example image for defining of thedirection movement	147
Figure 8.18	The initial zones for the prediction applying	149
Figure 8.19	The outcomes of automatic tracking	149
Figure 8.20	The outcomes of automatic tracking on the image with two objects	150

List of Tables

Table 3.1	An example result of inceasing the number of gradient samples of the test image Figure 3.16	43
Table 3.2	The calculation results after erosion of the left column of Figure 3.23	51
Table 3.3	The calculation results after erosion of the right column of Figure 3.23	51
Table 3.4	Setting parameters	52
Table 3.5	The results of increasing the number of gradient samples of the test image that contains an object in Figure 3.20 No.1 (left column)	53
Table 3.6	The results of increasing the number of gradient samples of the test image that contains 2 objects in Figure 3.20 No.2 (right column)	53
Table 3.7	Setting parameters for testing on threshold changing	56
Table 3.8	The results of adjusting the threshold level of the test image no.1 in Figure 3.30 (contains one person)	57
Table 3.9	The results of adjusting the threshold level of the test image no.1 in Figure 3.30 (contains 2 persons)	57
Table 3.10	Setting parameters for testing on gradient samples changing	58
Table 3.11	The results of 25 gradient samples of image No.1 in Figure 3.30	59
Table 3.12	The results of 33 gradient samples of image No.2 in Figure 3.30	59
Table 3.13	Setting parameters	62
Table 4.1	List of scaling and wavelet coefficients of Daubechies 4	74
Table 5.1	Ilustrates the used SOM dimension for each type of feature	96
Table 5.2	Shows the results of the classification performed on the test-images	96
Table 7.1	Training results	119
Table 7.2	Results of several CA patterns training by BPNN	121
Table 7.3	BPNN training results of 256 CA patterns with fixed target patterns	124
Table 7.4	BPNN training results of 256 CA patterns with random target patterns	124
Table 8.1	Shows the result of grid dimension adjusting	134

Contents

Abstract i

Acknowledgement ii

Declaration iii

Glossary of Abbreviations iv

Glossary of Symbols v

List of Figures vii

List of Tables xi

Chapter 1: Introduction

1.1 Why digital image processing? 1
1.2 Research objectives 3
1.3 Outline of the thesis 4
1.4 Original contributions 6
1.5 Published Work 6

Chapter 2: Review

2.1 Introduction 7
2.2 Human motion analysis 8
2.3 2-D approaches using the side view image 10
2.4 2-D approaches using the top-down view image 18
2.5 Summary 22

Chapter 3: Object detection & Data collection

3.1 Introduction 23
3.2 Blackground subtraction 25
3.3 Binary transformation 28
3.4 Co-ordinate generation 32
3.4.1 Amplitude projection 32
3.4.2 Smoothing method 35
3.4.3 Pixels difference method 37
3.5 Data collection 44

3.6 Experiments 45

3.6.1 Threshold adjustment 45

3.6.2 Changing shape and size of the structuring element in the erosion operation 49

3.6.3 Changing the number of gradient samples in the smoothing process 52

3.6.4 Perfroming the presented method on sequential images 56

3.6.5 The object detector simulation 61

3.7 Summary and discussion 65

Chapter 4: Feature extraction module(FEM)

4.1 Introduction 66

4.2 Pre-processing method 68

4.2.1 Wavelet transform (WT) 68

4.2.1.1 The one-dimensional discreate wavelet transform(1-DWT) 69

4.2.1.2 The two-dimensional discreate wavelet transform(2-DWT) 71

4.2.1.3 Scaling and wavelet functions 74

4.3 Feature generation 75

4.3.1 Texture features 75

4.3.1.1 Energy and Mean deviation signatures 76

4.3.1.2 The feature weighting 77

4.3.1.3 The histogram signatures 78

4.3.2 Eigenvector feature 80

4.4 Results 82

4.5 Summary and discussion 87

Chapter 5: Classification and Clustering Module (CCM)

5.1 Introduction 88

5.2 The Classification and Clustering Module (CCM) 89

5.2.1 The Self-Organising Map (SOM) 90

5.2.2 Optimal size for a Self-Organising Map 92

5.2.3 The Codebook generation 93

5.3 Experimental verification 95

5.4 Summary and discussion 99

Chapter 6: Experiments

6.1 Introduction 100

6.2 The pedestrian detection based classification technique (PDC) 100

6.3 Experiments 101

6.4 Discussion 106

Chapter 7: The association of Cellular Automata and Neural Network

7.1 Introduction 107

7.2 Cellularautomata (CA) 108

7.2.1 One-dimensional Cellular Automaton 109

7.2.2 Two-dimensional Cellular Automata 113

7.3 Neural Network (NN) 114

7.4 Association of the CA and BPNN 117

7.4.1 1D-CA experiments 117

7.4.2 2D-CA experiments 123

7.5 Summary and discussion 126

Chapter 8: Prediction Module(PM)

8.1 Introduction 127

8.2 Prediction module (PM) 128

8.2.1 Grid-generating block 128

8.2.2 CA plane to vector block 129

8.2.3 BPNN Classification 130

8.2.4 Predicted location provider block 131

8.3 Data collection for BPNN 131

8.4 The prediction module (PM) experiments 132

8.4.1 Defining of grid dimension 132

8.4.2 The tracking process using the prediction method 137

8.4.2.1 The test on the prediction tracking using manual locating 137

8.4.2.2 The test on the prediction tracking using automatically locating 144

8.5 Evaluation of the method 150

Chapter 9: Conclusion and future work

9.1 Conclusion 152

9.2 Future work 154

References 156

Appendixes:

Appendix A: Least-square Parabolic Smoothing and Differentiation 162

Appendix B: Publications 164

Chapter One

Introduction

1.1 Why digital image processing?

Over the last decade, a new application domain of computer vision has emerged dealing with the analysis of images involving humans. This domain covers, among others, face recognition, hand gesture recognition and whole-body tracking. The huge interest in this domain has been motivated by the desire for improved man-machine interaction, for which, there are many promising applications [1-6].

One of these applications is the automated counting of people, which has been used for various purposes such as safety monitoring and crowd control, security access control, counting customers for market surveys, pedestrian flow surveys and building usage surveys [7,8]. Depending on the purpose, the accuracy required from the people counting system can vary. For example, for crowd control purposes [9], an estimate of the number of people in an image is found to be adequate information. On the other hand for purposes such as counting customers for market surveys, pedestrian flow surveys and building usage surveys, a higher level of the accuracy is desired.

Note that, apart from video, a number of other sensor technologies such as light beams, microwaves, ultra-sound and infrared, have been investigated in order to count people. However, the video-imaging sensor has proven to be very suitable because of its reasonable cost, large monitoring area, high speed and suitability for a variety of conditions [10]. The increasing popularity of digital image processing techniques as a means to solve the people detection problem is probably also due to the fact that video cameras, and image processing techniques present a convenient, non-intrusive and cost effective solution [7,8].

A major problem in the technology exists in how to identify and distinguish between people in video images, as images of pedestrians are non-rigid objects with a high degree of variation in size, shape, colour, and texture. The problem is challenging and has prompted researchers to investigate new methods for solutions in order to increase the efficiency of counting people by means of video images.

1.2 Research objectives

The main aim of this study is to research a prospective selection and combination of techniques for people detection and tracking in video images in order to increase the accuracy of automated pedestrian tracking. This main aim is expressed in terms of six objectives in order to provide a structured approach to the research:

1. To develop a theoretical basis for pedestrian detection and tracking then determine its limitations, by means of researching previous work and the current status of other ongoing works in similar areas.

2. To develop a theoretical basis for the detection and location of objects within an image and to estimate their direction and speed.

3. To develop computer programmes in order to demonstrate the validity of the object detection along with direction and speed estimation.

4. To develop a theoretical basis for predicting pedestrian movement in video images using a combination of Artificial Neural Network and Cellular Automata.

5. To develop computer programmes in order to demonstrate the validity of this prediction technique for the tracking of pedestrians through image sequences.

6. To critically analyse the results and provide suggestions for improvement.

1.3 Outline of the thesis

Before the thesis addresses the particulars of the conducted research, Chapter 2 provides a general overview of people tracking in video images. This chapter outlines the merits and drawbacks of this form of tracking, advances that were made in recent years and the related techniques that have been presented.

Chapter 3 details the techniques used for object detection in a pedestrian image and the technique for data collection. The chapter then outlines experiments that were conducted on each component of the object detector, and comments on the outcomes of these experiments.

The use of a discrete wavelet transform (DWT) for pre-processing in the feature extraction module is the subject of Chapter 4. This chapter outlines the Feature Extraction Module (FEM) and describes how to generate texture and Eigenvector features from the transformed image using the DWT. This chapter then shows graphs of the FEM results that were obtained using training sets, which consisted of a set of person's head images, and a set of other parts of a person's body or background images, referred to as non-head images, and discusses the outcomes.

The use of an artificial neural network for a person's head pattern classification is the subject of Chapter 5. This chapter outlines the Classification and Clustering Module (CCM) and details the unsupervised learning neural network with Self-Organising Map (SOM), the latter's algorithm and the optimisation of the SOM size. The chapter then explains how to generate a person's head and non-head codebooks. It also outlines the experiments that were conducted on the CCM and discusses their results.

Chapter 6 examines the overall performance of the proposed pedestrian detection based classification technique when it processes sequential pedestrian images.

Chapter 7 describes a novel prediction technique based on the association of Cellular Automata (CA) and the supervised learning neural network with the Backpropagation learning algorithm. The chapter then gives details of how the CA and NN can be associated. Experiments were carried out on the feasibility of using NN simulation in order to support the author's conjecture.

In Chapter 8, the prediction module and its block diagram is introduced along with details of how to capture a pedestrian's movement behaviour using a CA. The purposes of using the Neural Network in the prediction module are given. It also outlines the experiments, which were conducted on the prediction module and discusses the outcomes of the experiments.

Finally, a conclusion of the presented study and the direction for future work is provided in Chapter 9.

1.4 Original Contribution

This research work looks at image processing used for the application of pedestrian tracking in video images. The motivation behind this work is to improve the efficiency of automated pedestrian tracking using the image information in a more accurate and cost effective manner. First, an object detector based on amplitude projections has been employed to obtain candidate locations for recognition and tracking. Experiments on the object detector have been performed and presented in Chapter 3. Second, application of the discrete wavelet transform (DWT) has been made together with a Self-organizing Map (SOM) neural network to identify a person's head within an image. This has been demonstrated in Chapters 4, 5 and 6. Finally, simulations have been used to show that the association of Cellular Automata (CA) and Backpropagation Neural Network (BPNN) can be successfully employed as a prediction method for aiding pedestrian tracking systems in Chapters 7 and 8.

1.5 Published Work

The following are publications made in support of the work detailed in Chapters 7 and 8 of this thesis.

- C. Suppitaksakul, G. Sexton and P.D. Minns, "A prediction technique using the association of Cellular Automata and a Neural Network", on *Electrical Engineering Conference 28th, EECON-28, Phuket, Thailand, 20-21 October 2005*, pp. 969 -972.
- C. Suppitaksakul, G. Sexton and P.D. Minns, "A pedestrian tracking using the association of Cellular Automata and a Neural Network" *on Communication Systems, Networks and Digital Signal Processing the 5th International Symposium, Patras, Greece, 19-21 July 2006.*

Chapter Two

Review

2.1 Introduction

The use of computer vision for the analysis of images involving humans or "*people tracking*" has grown exponentially over the last decade. It has attracted such great interest from computer vision researchers because of its potential for application in areas such as virtual reality (e.g., virtual studio, games, character animation, teleconferencing), surveillance systems (e.g., access control, supermarkets, department stores, vending machines, ATMs, traffic), advanced user interfaces (e.g., social interfaces, sign-language translation), motion analysis (e.g., clinical studies of orthopaedic patients, personalisation training in golf, tennis, etc.) and model-based coding (e.g., very low bit-rate video compression) [1].

In each of these areas the requirements of the vision analysis are varied and so different approaches have been developed to meet the needs of each system. An overview of the current methods and ongoing research in this area is provided in the following sections: Sections 2.2 – 2.5 provide a description and summary of the human motion analysis methodologies that will be utilised later in this thesis. A discussion of all of the techniques is not possible due to the vast number available.

2.2 Human motion analysis

Human motion analysis or "*people tracking*" concerns the detection, tracking and recognition of people, and commonly the understanding of human behaviours, from image sequences involving humans [2]. Various surveys on the relevant topics in this area have been presented since 1994 as can be seen in [2-6]. Each survey used different criteria to classify the previous works depending on its purpose.

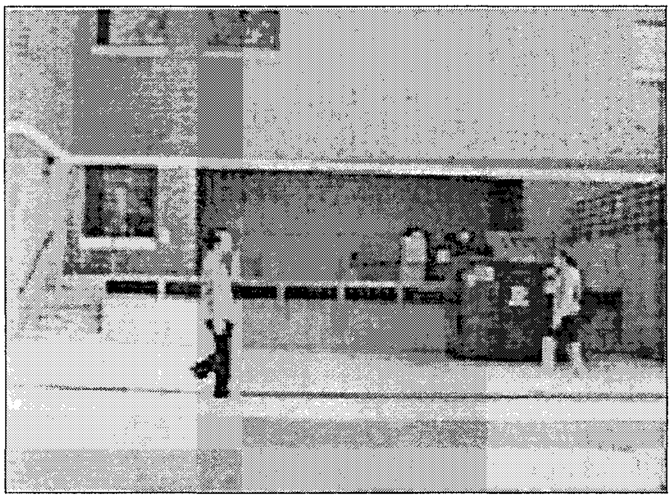


Figure 2.1: A side view image [11]

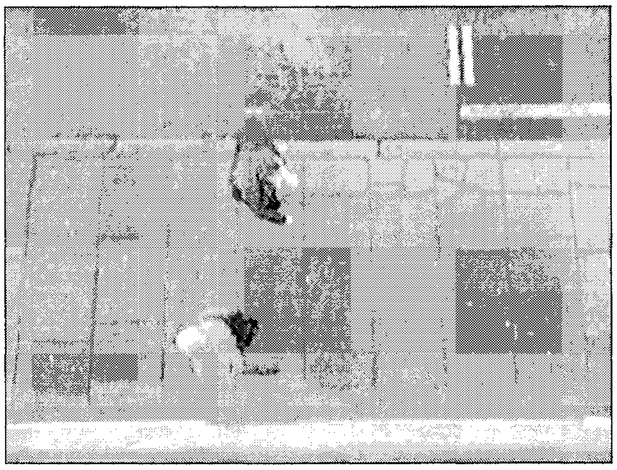


Figure 2.2: A top-down view image

One of the applications of human motion detection considered in this thesis is surveillance. In this application it is only the movement of the subject that is of interest and which tracking is required for [4]. The tracking is performed on two-dimensional (2-D) images. Therefore the literatures of interest in this review are those that are based around methods for tracking human movement that directly focus on the motion of the whole body and the dimensionality of tracking space (2-D vs.3-D). The literatures can be grouped using the angle of view criteria; these are side and top-down views (as shown in Figures 2.1 and 2.2). An overview of the ideas, concepts and techniques used in the selected works are listed and discussed in the next sections.

2.3 2-D approaches using the side view image

In this section, works on "*people-tracking*" using side view images obtained from a single static camera as illustrated in Figure 2.1, are discussed. In a side view image a person's frontal, rear and lateral information can be obtained and therefore can be used for detecting, tracking and recognition of persons from the image. Tracking is equivalent to establishing correspondence of the image structure between consecutive frames based on features related to position, velocity, shape, texture and colour. The process of tracking concerns matching between images employing pixels, points, lines and blobs based on their motion, shape and other visual information [4]. Various techniques in have been proposed to approach to "*people-tracking*" as follows:

Shio and Sklansky [12] proposed a method for segmenting and tracking of people in motion from a sequence of images. The motion field in each frame is obtained by matching intensity features, which is based on a quasi–cross correlation technique of successive frames and it is smoothed both temporally and spatially, to remove the effects of non-rigid motion and measurement errors. Later, it is split into regions having the same quantized direction of motion. Neighbouring sub-regions are then iteratively merged based on the direction of motion and the expected size of the bounding box around a person. The disadvantages of this technique are the processing costs of gray-scale feature matching and iterative merging of motion sub-regions. No temporal integration is performed on the segmentation outcomes.

Segen and Pingali [13] used a model-based algorithm for a motion clustering approach to object tracking. Features in each video frame are identified and then matched across frames to produce feature "paths". Then the system groups short-lived and partially overlapping feature paths into longer living trajectories representing the motion of individual persons. Compared to [12], their technique uses less computation in the matching process and due to their system relies on multiple trajectories so it is more robust if all matches are correctly resolved. However, the robustness of the matching process is doubtful being dependent on the condition of natural clothing and non-rigid human shapes.

Another technique relies on a learning-based approach to detect persons. Oren *et al.* [14] applied the object-detection technique using Haar wavelet coefficients as low-level intensity features for the detection of frontal and rear views of pedestrians in static images. They segment pedestrians in a number of images and generate a common template based on Haar wavelets. These transformed templates are used as features for training a support vector machine (SVM) classifier (a universal feed forward network that was introduced by Vapnik [15-17]). In the run-time or detection stage, windows of various sizes are scanned over the image, the selected features are extracted, and then applied to the SVM classifier to verify whether the desired object is present or not. An advantage of using the wavelet template is that it is robust and is not sensitive to spurious details. However, the results presented do not report on the problem of overlap between objects in the image. A large number of features is required to construct a neural classifier and is too large to ensure fast computation; also, detection has been restricted to rear and front views of pedestrians.

Poggio and Papageorgio [18] modified the object detection approach from [14] for application into a real-time system. They improved the pattern classification technique by removing the need to perform the feature selection process. Instead, the entire set of Haar wavelet features are used to train a SVM classifier that can effectively handle high-dimensional feature spaces populated by a small data set. By extending the static representation into the time domain they claimed that the system is able to provide a level of information approaching that of a dynamic system. The system is able to learn the important characteristics of the physical structure and dynamics of people as they appear in the video sequences. In this manner, the need for explicit shape or motion models is completely avoided. A drawback of this system is in the searching process, since they applied object templates to match with the objects in the image the process becomes time consuming.

Statistical shape models have also been applied to detect and track the contours of persons. An early work was proposed by Baumberge and Hogg [19]. They used the combination of an Active Shape Model (or Point Distribution Model) and a Kalman filter for real time pedestrian tracking. In this system, the initial contour position and scale were calculated by a segmentation scheme using background subtraction. The active shape models, which represented pedestrians in several shapes based on B-spline as showed in Figure 2.3, are then applied to contour the pedestrian in the subtracted image. In the stage of tracking, the Kalman filter is used in order to control spatial scale for a feature search over successive frames. This method provides good results for contouring a moving pedestrian as the pedestrian poses and views are sufficiently well represented in the training set. Unfortunately, the Kalman filter is restricted to situations where the probability distribution of the state-parameters is unimodal. In the presence of occlusion or a cluttered background resembling the objects being tracked, and complex dynamics, the distribution is likely to be multi-modal [5].

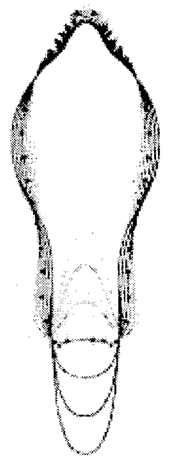


Figure 2.3: Principal component analysis on a data set of pedestrians represented by B-splines; shown is the shape variation along the principal component [19]

The work of Isard and Blake [20] described a complex motion model with the stochastic differential equation then combined this approach with deformable templates to track people. They proposed the CONDENSATION algorithm [21] as a tracking method. The CONDENSATION algorithm uses "factored sampling" in which the probability distribution is represented by a randomly generated set. CONDENSATION uses learned dynamical models, together with visual observation, to propagate the random set over time. The result is the highly robust tracking of agile motion. However, the

CONDENSATION tracker has difficulty with real-time implementation for multiple objects tracking since it requires complicated shape models and a large number of samples for precise tracking performance.

Recent work by Tissainayagam and Suter [22] applied a cubic B-spline to contour the silhouette of moving objects and tracked them with a lower dimensional shape space that proved to be fast and efficient. In order to make the tracker robust and reliable, when the object of interest is moving with multiple motion they coupled the tracker, based on a Kalman filter, with an automatic motion-model switching algorithm based on the interacting multiple model (IMM) algorithm. As the example result of this method shows in Figure 2.4, the tracker is able to recover from the occlusion.

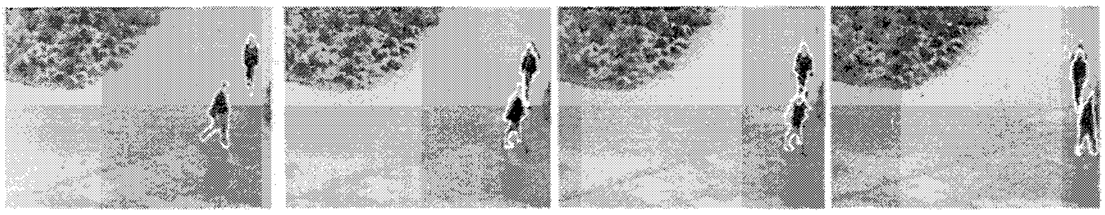


Figure 2.4: The tracking results from Tissainayagam and Suter [22]

Another recent work of Kang *et al.* [23] employs B-spline for contouring the object and proposes improvement of the CONDENSATION tracker for real-time multiple people tracking. They introduced Competitive CONDENSATION, based on the Self-Organizing Map (SOM) as a discrete shape model for human shape representation. This allows the reduction of the shape space to one discrete valued parameter. They then used the competition rule, which requires a small number of samples to track multiple people. The outcome of this technique, shown in Figure 2.5, is quite promising and effective as it performed in real-time.

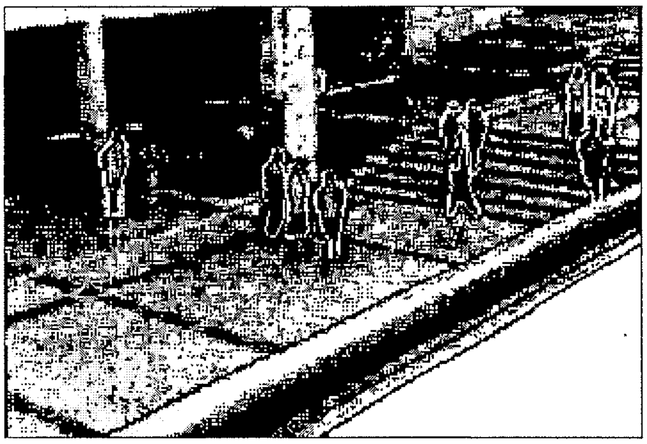


Figure 2.5: The example outcome from Kang *et al.* [23]

Silhouettes are also employed as cues for tracking and detection. For instance, Haritaoglu *et al.* [24] introduced the W^4 system for real-time people tracking in monochromatic imagery obtained from a stationary camera. W^4 uses stereo information in combination with shape analysis to locate and track people and their parts (head, hands, feet, torso). The system detects objects through a background subtraction process and then simply classifies those objects as people, vehicle, or other on the basis of static size and shape properties and through the analysis of shape periodicities. The shape information is implemented using a cardboard model that represents the relative positions and sizes of the body parts. Along with the second order predictive motion models for the body and its parts, the cardboard model can be used to predict the positions of the individual body parts from frame to frame. Template matching is employed to track body parts instead of using the shape model when a person is occluded. However, W^4 has a limited capacity to handle occlusion and to identify people after occlusion.

Later, Haritaoglu *et al.* [25] developed the Hydra system to deal with tracking multiple people in monochromic imaginary, which was an essential an extension of the W^4 system. Hydra attempts to detect the heads of people in groups and track them through occlusions. It employs silhouette-based shape models and temporal texture appearance models. Although effective in many situations, these 2-D appearance models will not cope well with large rotations in depth during occlusions. Hydra is therefore quite effective at tracking people through occlusions as they walk past one another but it does not cope well when people leave a group in a different direction from that in which they entered it.

Apart from the use of silhouettes, colour is another cue that used for tracking people. Wren *et al.* [26] presented the use of the small blob feature to track a person in an indoor environment. In their real-time person finder system "Pfinder", models and trunks the human body are defined using a set of "blobs". Each blob is described in statistical terms by a spatial (x, y) and colour (Y, U, V) Gaussian distribution over the pixels it consists of. The blobs typically, correspond to the person's hands, head, feet, shirt, and pants. A statistical model is also constructed for the background region where a Gaussian distribution in terms of colour values describes each pixel. At initialisation, the background model is used to identify a foreground region with pixel values other than those given the background model. A model-building process follows where blobs are placed over the foreground region. This process is guided by a 2-D contour shape analysis that attempts to identify various body parts using heuristics. Tracking involves a loop of predicting the appearance of the person in the new image, determining for each pixel the likelihood that it is part of one the models, and updating the statistical models. The strength of these approaches is the comprehensive feature selection it offers, which could result in more a robust performance when the cost of computation is not a major concern.

Heisele *et al.* [27] used groups of pixels as basic units for tracking. A clustering technique that combined colour (RGB) and spatial (x,y) dimensions is used for pixel grouping, because the adding of spatial information makes clustering more stable than using only colour information. The obtained pixel groups are adapted iteratively from one image to the next image using a k-means clustering algorithm. Due to the fixed number of pixel groups and the encoded one-to-one correspondence over time, tracking these units is straightforward. Although, the outcomes of the proposed technique have shown promising initial efficiencies on tracking people, there is no guarantee that units will remain locked onto the same physical entity during tracking.

McKenna *et al.* [11,28] introduced an adaptive background subtraction method that combines colour (RGB) and gradient information to cope with shadows and unreliable colour cues, to track groups of people. The system tracks people through mutual occlusions as they form groups and separate from one another. They employed colour information to disambiguate occlusion and to provide qualitative estimates of depth ordering and position during occlusion.

Wenmiao Lu, and Yap-Peng Tan [29] used a colour (HSV) histogram based recognition technique for tracking moving people. Their system aims to resolve the identity of each tracked person after an occlusion. The system also uses a background subtraction method for detecting moving people in the scene and localizes them into confined regions through an analysis of image moments. Then they employ HSV colour histograms to each of the people to aid in the tracking process. They claim that their proposed recognition method is robust. Although the robustness is provided by the colour-based technique, concern remains regarding the computationally cost.

2.4 2-D approaches using the top-down view image

This section discusses works that detect and track human movement from the top view image. Top view images are obtained from a camera positioned looking down on a scene, as in Figure 2.2. Little work has so far been carried out in this area despite the clear advantage is offers in terms of avoiding the overlap of people present and multiple occlusions. It also offers the advantage of needing to track movement perpendicular to the image since depth does not change. This makes this image approach highly suitable for applications involving people counting.

Rossi and Bozzoli [30] developed a system for tracking and counting moving people in greyscale images. Their system applied a motion detection module using temporal change detection and histograms to detect whether any person has entered the scene. In the tracking stage, a tracking module based on a prediction method using a simple trajectory module and template matching based on a correlation method with a three-step search procedure is applied to follow people until they reach the counting line. The template matching process in this system was a major drawback because it is considered to be extremely time-consuming and therefore unsuitable for real-time applications. Also, the detection mechanism in this system uses motion and therefore can only be applied to detect people in consequent images which have a static background.

Zhang and Sexton [10,31,32] applied a Model-specified Directional Filter (MDF) and a simple circular model matching process to track head locations in the greyscale images. The MDF is a form of a specially shaped Gaussian filter that combines the general Gaussian filter with a particular shape model. It is introduced for use as a detector that is able to provide initial or candidate locations for diminishing the searching time in an image. A circular matching technique is then applied to contour the pedestrian's head. An advantage of this method is it can be applied to detect and track pedestrians in an image with both static and dynamic backgrounds. However, this technique is found to have difficulties in detecting unusual head shapes corresponding to pedestrians wearing hats or with unusual hairstyles.

An algorithm called "*Closed-world tracking*" used for real-time tracking was proposed by Intille and Bobick [33]. The tracking method is based on contextual information, used to simultaneously track multiple, complex, non-rigid objects. In the detection stage, a YUV-based background subtraction technique that uses colour information was applied, to remove the background. The thresholding and three dilated operations were performed to eliminate noise and gather the broken parts of the objects respectively.

Then a fast bounding box-merging algorithm was employed to cluster the small group of blobs and the resulting blob image for one frame was provided. Four properties of each object, which are average colour, position, velocity, and size, are computed and employed in the tracking process. Unfortunately, the system is found to be very sensitive to threshold setting in the matching process.

Schofield *et al.* [7,8] applied a neural network to identify different conditions of the background scene in order to count the number of people in the image. They used sub-samples of the binary images of each of the background scenes, which were obtained by thresholding the background image, as training sets for classification. The RAM-based neural network classifier is applied to the training set for discriminating between parts of the background scene and non-background objects. In the form described in this work it is unable to provide information about the actions or intended actions of people in the scene and it is unsuitable for application to images with moving backgrounds such as images of people on an escalator.

Jae-Won Kim *et al.* [34] proposed an approximated convex hull algorithm to track passing people into the building for a counting purpose. This work also employs background subtraction and thresholding methods in the detection process where the interested regions are provided. A boundary box is then estimated for each of the regions in order to enclose each object. A sophisticated geometry algorithm is applied involving the determination of the smallest convex set or *"**convex hull**"* containing a discrete set of points to extract the object features such as area, centre and boundary inside the bounding box. As argued previously, these features are less changeable therefore they can be used for tracking. In the tracking process, the simple convex hull of each person within the bounding box in the tracking area is approximated. Then the bounding box is rotated in 15 degree steps as illustrated in Figure 2.6 to provide more tracking information and in order to make the tracking more accurate. This work provided a promising 96% accuracy for counting people correctly. However, this technique is unsuitable for images with dynamic backgrounds.

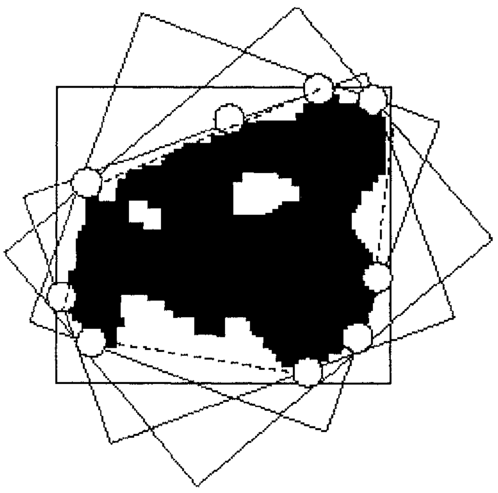


Figure 2.6: The convex hull approximation from Jae-Won Kim *et al.* [34]

2.5 Summary

A review of the literature related to human motion analysis or ***"people-tracking"*** has been provided. The scope of this review was limited to those works regarding analysis of human movement that focus on the motion of the whole body and human detection and tracking methods in 2-D. Two main approaches were discussed: 2-D approaches using a side view image and 2-D approaches using a top-down view. Although a number of promising works in this area already exist, many issues are still open such as image segmentation, use of models, tracking versus initialisation, multiple persons, occlusion and computational cost etc. Consequently, some of these issues will be considered in this thesis.

In the following seven chapters, a novel motion detection and person tracking system will be developed and analysed. First, the object detection technique used to identify candidate locations will be presented. The data accumulated from the object detection technique will be used to build a database, which in turn will be employed as a source of data for training of neural networks and system testing in subsequent chapters.

Chapter Three

Object Detection & Data collection

3.1 Introduction

In this chapter, discussions regarding the areas of object detection and data collection are presented. The purpose of object detection is to detect and locate objects within an image in order to provide candidate locations for recognition and tracking. These given locations are used as initialisation for the recognition process and the tracking of objects. Simple methods based on background subtraction and amplitude projectors are used on "*intensity*" or "*grayscale*" images for object detection.

The object detector consists of three essential parts shown as a block diagram in Figure 3.1. The first block is the ***Background subtraction*** block; the inputs to the block are the captured image and a previously stored static background image (which is a previously captured input image without objects present). The output from the block is the subtraction of the static background image from the input image. The result is an image that represents objects presence by means of pixels of various intensities, i.e. shades of grayscale. Such shades of grayscale are redundant information as far as a simple detection is concerned and the subtracted image is therefore passed through a ***Binary transformation*** block where it is converted to a binary image that has only two intensity levels, namely black and white. Then it is passed through a ***Co-ordinate provider*** block where the co-ordinates, which consist of row and column positions of the objects in the image, are defined and given. In order to view the object locations, the provided co-ordinates are then fed into an ***Image marking*** block where the input image is plotted using the positions of the co-ordinates obtained. The techniques used in these four blocks of Figure 3.1 are explained in more detail in the following sections: Section 3.2 background subtraction, Section 3.3 binary transformation and 3.4 co-ordinate provider.

The outcome of the object detection block is affected by both the data collection and recognition processes. Consequently, the manner in which data is collected is explained and demonstrated in Section 3.5. Section 3.6 details experiments and evaluations that have been performed on this detection method and Section 3.7 summarises and discusses the outcomes of the detection method.

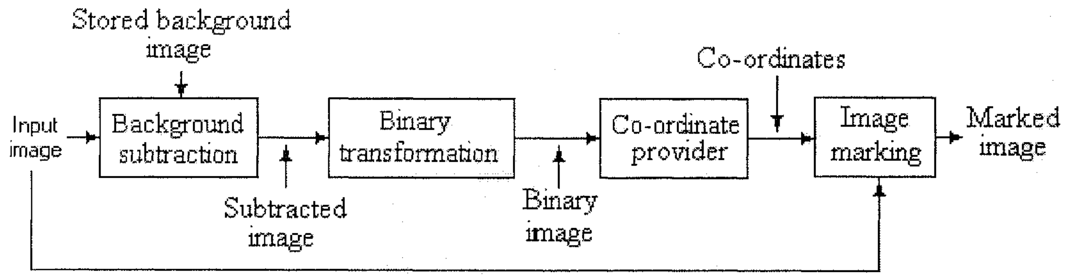


Figure 3.1: Object detection block diagram

3.2 Background subtraction

Background subtraction is a simple technique used to obtain figure-ground segmentation. The use of background subtraction is very popular and has been applied to many works as in [24,27-29]. In this work, background subtraction is employed to extract objects from the background image. The technique involves subtracting a stored background image $I_{back}(i,j)$, which contains no objects, from the input image $I_{ped}(i,j)$. The result of this subtraction provides the contents difference in each pixel position in terms of intensity. Since the background image was used for subtraction, this resulting image is called a ***background subtracted image*** $I_{sub}(i,j)$ and it can be defined as [35]:

$$I_{sub}(i,j) = I_{ped}(i,j) - I_{back}(i,j) \qquad (3.1)$$

where i denotes the pixel row and j the pixel column

The subtraction process is demonstrated in Figure 3.2. The magnified background-subtracted image is shown in Figure 3.3. Note that due to the presence of negative intensities the background-subtracted images, shown in Figures 3.2 and 3.3, have been adjusted for clarity.

The background-subtracted image (I_{sub}) contains both negative and positive intensities as can be seen in the histogram illustrated in Figure 3.3. This is simply because the object intensity could be greater or less than the background intensity in any given image. For example where the intensity of the object is brighter than that of the background the difference in contrast is positive and vice-versa.

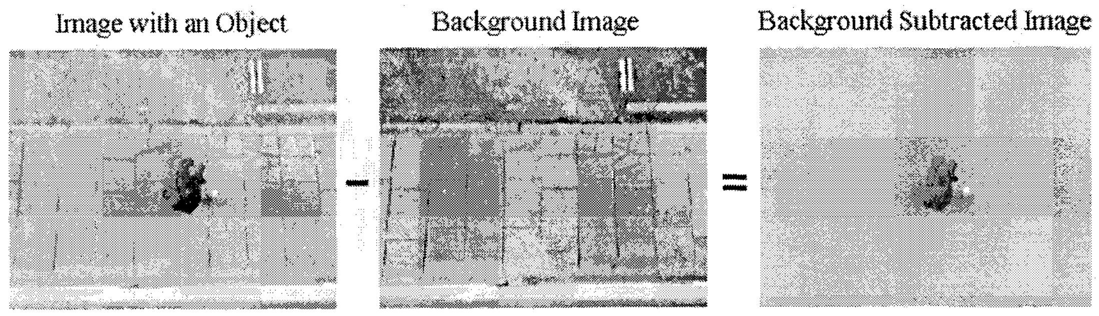


Figure 3.2: The background subtraction process

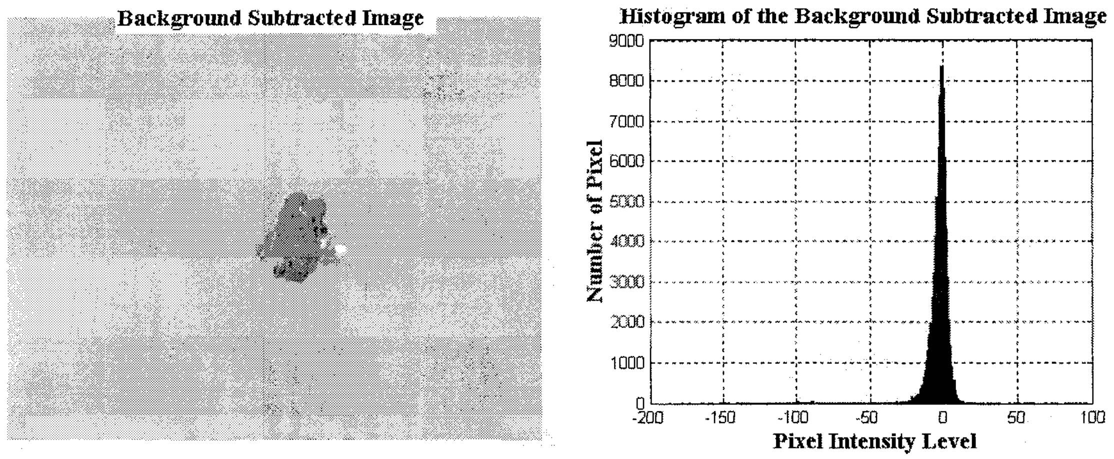


Figure 3.3: The background-subtracted image and its histogram

From the histogram it can be assumed that those pixels with an intensity level around zero corresponds to the background because the area represented by the background in the subtracted image is larger than that which corresponds to the object. Although, the backgrounds of both the input image and the stored image are of the same scene, the subtraction of the two will not lead to zero intensities since the two images are captured at different times with slightly different light conditions. This also apparent if two background only images are subtracted, this is shown in Figures. 3.4(a) and (b).

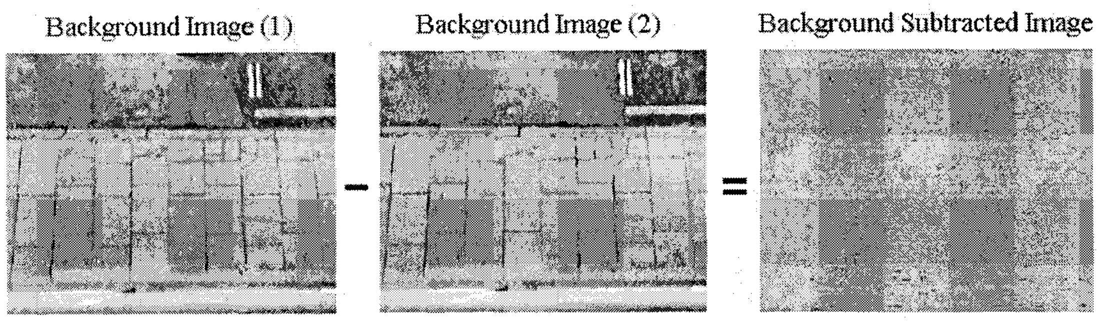


Figure 3.4 (a): Testing of two-background subtraction

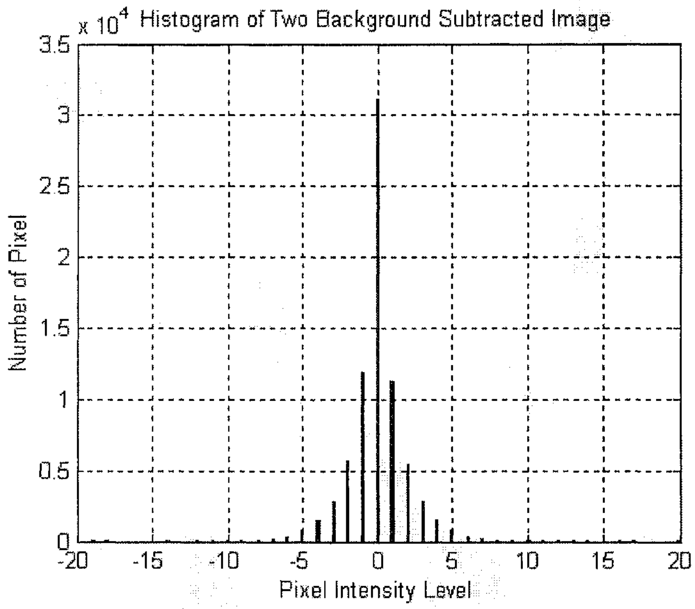


Figure 3.4 (b): Histogram of the background subtracted image

From the histogram, it can be concluded that most of the background intensities lie around the zero area after subtraction. This information is very useful in determining a threshold to eliminate the foreground from the background. How to use the information from the histogram to define the threshold is discussed in the next section.

3.3 Binary transformation

The background-subtracted image shows foreground and background as pixels of various intensities, which is still too detailed for segmentation purposes, which need two intensity levels. In order to distinguish between the foreground and background pixels, two levels of intensity suffice. To reduce the information within I_{sub} it is therefore appropriate to transform the background-subtracted image to a so-called ***binary image*** in which each pixel assumes only one of two discrete values, which correspond to on (binary 1), and off (binary 0) [36]. This binary transformation can be accomplished through thresholding, which is a simple and fast method [37]. By defining a range of brightness values in I_{sub}, pixels within this range are regarded as belonging to foreground whereas all other pixels are considered as being part of the background. Using the histogram of I_{sub} as discussed in the previous section, the threshold can be defined.

From the histograms in Figures 3.3 and 3.4 (b), it can be assumed that the intensity values around zero with large numbers of pixels correspond to the background pixels. In order to distinguish the foreground from the background, two thresholds (negative and positive), are required for segmentation. Only one threshold will suffice if only the absolute intensity values of the subtracted image are used. The histogram of the absolute intensity values is shown in Figure 3.5. The standard deviation of the absolute intensity histogram is useful in defining the threshold value.

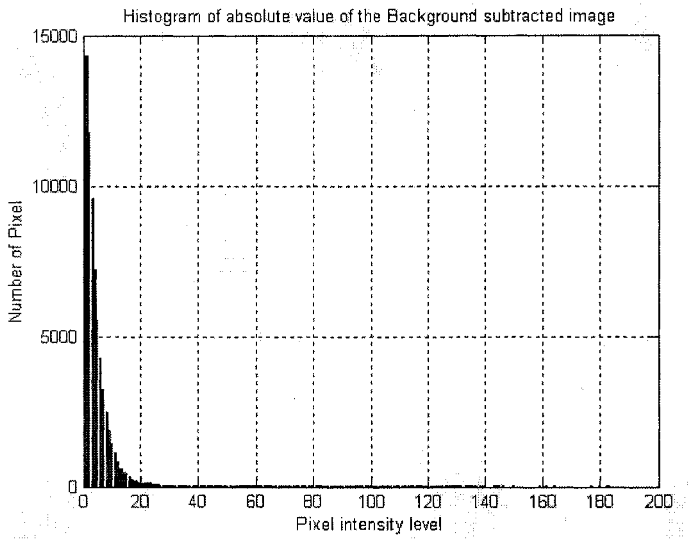


Figure 3.5: Histogram of absolute background subtracted image

The standard deviation (σ), I_{std}, based on the elements of the absolute subtracted image is defined as [38]:

$$I_{std} = \sqrt{\frac{1}{(m \times n) - 1} \sum_{i=1}^{n} \sum_{j=1}^{m} (|I_{sub}(i,j)| - I_{mean})^2} \qquad (3.2)$$

$$I_{mean} = \frac{1}{m \times n} \sum_{i=1}^{n} \sum_{j=1}^{m} |I_{sub}(i,j)| \qquad (3.3)$$

where $|I_{sub}(i,j)|$ is the absolute subtracted image, I_{mean} the average value of the subtracted image, m the number of pixel columns and n the number of pixel rows.

Therefore the threshold value is defined as:

$$Threshold = T \times I_{std} \qquad (3.4)$$

Where T is the multiplier number in order to adjust the threshold value

Having defined the threshold; the binary image $\boldsymbol{I_{binary}}$ is obtained by applying the threshold to the absolute subtracted image as follows:

$$I_{binary}(i,j) = \begin{cases} 1 & for \ |I_{sub}(i,j)| \leq Threshold \\ 0 & for \ |I_{sub}(i,j)| > Threshold \end{cases} \tag{3.5}$$

After thresholding, the foreground intensities can be easily separated from the background intensities as shown in Figure 3.6. The outcome of thresholding is a binary image containing white-background and black-objects. As can be seen in the images, this simple technique is highly effective in the rough extraction of objects from the background. Experiments carry out on the threshold defining is discussed in Section 3.6.

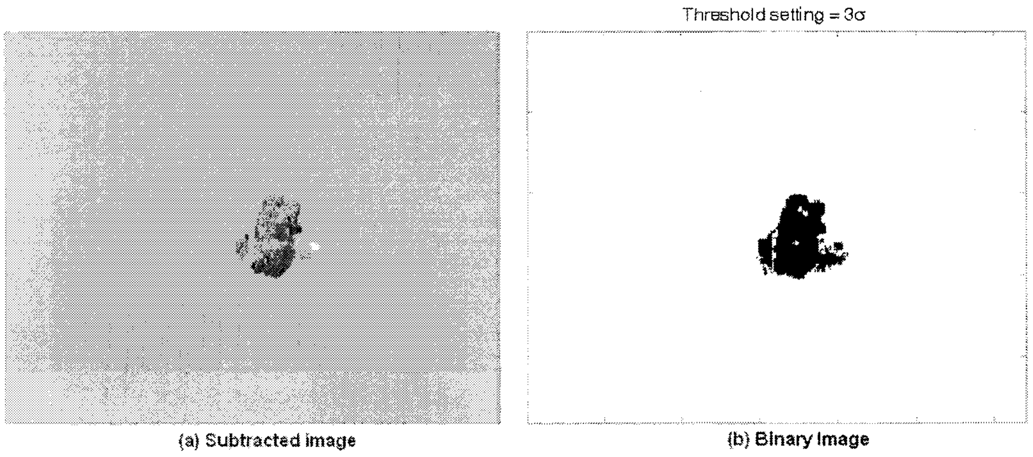


Figure 3.6: The subtracted image and its binary image

Although, the thresholding process is able to provide a rough location of the object it is still unclear and the boundaries of the object are disjointed. To enhance the binary image, one of the so-called ***morphological operations*** [37], which are methods for processing binary images based on shapes, was employed. The method used in this work relies on so-called ***erosion operations***, these operations use rules that state that the condition of any given pixel in the output image is determined by applying a so-called ***operation rule of erosion*** to its neighbouring pixels. The operation rules of erosion are defined by assuming that if every pixel in the input pixel's neighbourhood is on (or equal to one) then the output pixel is on. In a similar manner, if the neighbouring pixels are off (or equal to zero) then the corresponding pixel in the output image is off. It can be of arbitrary shape and size, and is represented by a structuring element, which is a matrix consisting of only 0's and 1's. The centre pixel in this structuring element represents the pixel of interest, while all other elements in the matrix represent the neighbourhood. Figure 3.7 (b) illustrates the image obtained after the application of the erosion operation to Figure 3.6. The erosion operation applied used a structuring element consisting of a 3-by-3 block of pixels containing 1's. Figure 3.7 (b) shows that the disjointed boundary of the object is clearer after enhancement. The neighbourhood choices for erosion are discussed further in Section 3.6.

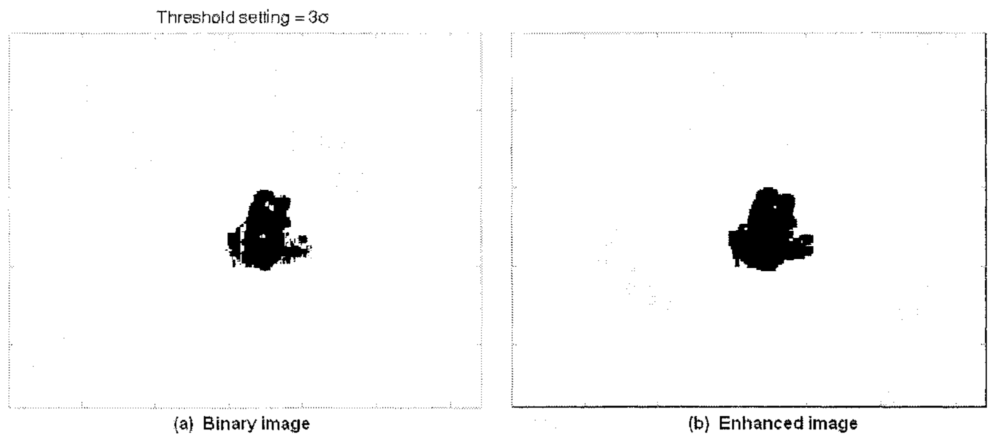


Figure 3.7: A binary image and its enhanced image

3.4 Co-ordinate generation

The purpose of co-ordinate generation is to obtain the co-ordinates that are used to locate the pedestrians in the original input image using the information contained in the binary image. The co-ordinate generation process consists of three processes as follows: firstly the amplitude projection process secondly the smoothing process and thirdly the pixels difference process. The three blocks of the co-ordinate generator are shown in Figure 3.8.

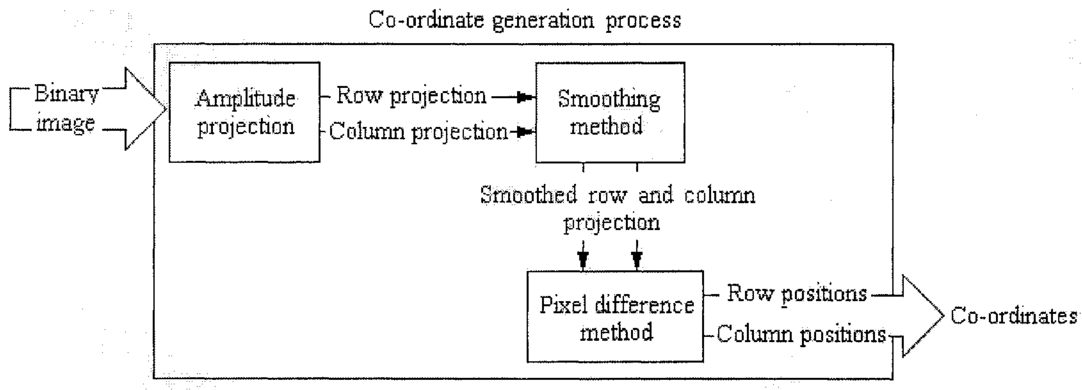


Figure 3.8: The co-ordinate generation process

3.4.1 Amplitude projection

The desired information within the binary image is object size, which consists of a height and a width. To extract this information, the binary image needs to be projected onto two graphs where the height and the width of the object within the binary image are analysed. Information about the approximate size of the person's image is obtained by measuring from test images. Examples of a person's image size are shown in Figure 3.9, the height of a person (Figure 3.9, H) in the image varies as it moves and the size varies between 50-90 pixels depending on a person's walking-posture, the width (Figure 3.9, W) in the image is not affected by a person's walking-posture and the width range lies between 60-80 pixels.

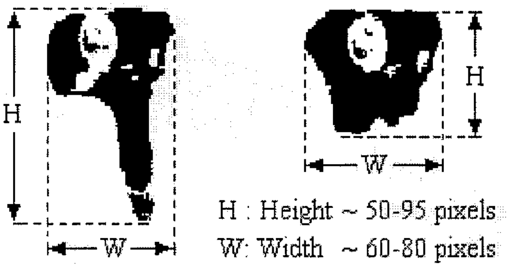


Figure 3.9: Examples of image of people's walking-posture

To process the projection, each row and column of the binary image is averaged as follows [39]:

$$Ph(i) = \frac{1}{M}\sum_{j=1}^{M} I_{binary}(i,j) \qquad (3.6)$$

$$Pv(j) = \frac{1}{N}\sum_{i=1}^{N} I_{binary}(i,j) \qquad (3.7)$$

where ***M*** is the total number of rows, ***N*** the total number of columns

The obtained ***Ph***, a so-called ***horizontal or row projection***, contains information about people's height in the image by means of its amplitude. The maximum amplitude, which equals 1, represents a column position that contains no object pixels. Column positions with a ***Ph*** amplitude lower than the maximum has pixels that belong to people as well as pixels that belong to noise which may be caused by the binary transform process. Averaging the amplitude of the pixels of the binary image row by row and column by column can be regarded as projecting these amplitudes onto one point. The averaging technique is therefore referred to as ***amplitude projection***.

A similar method has been applied to ***Pv***, a so-called ***vertical or column projection*** that contains information about a person's width in the image by means of its amplitude. The projections are normalised and their maximum amplitudes set to 1 (no pedestrian's pixel presented in that row or column) to simplify the evaluation. Therefore the areas in the projections with amplitudes of less than one, especially at deep valley areas, represent the pedestrian's areas and the approximate height and width of people shown in Figure 3.9 are very useful in aiding to identify a pedestrian area in the projections.

An example result of ***Ph*** and ***Pv*** of a binary image are plotted and illustrated in Figure 3.10. From this figure, it is obvious that the deep valley areas in ***Ph*** and ***Pv*** occur due to the replacement of the person's height and width with their relevant projections. These areas of the projections with high gradients can be used to define the positions of the pedestrians. Once the high gradients in the projections have been detected the object positions can be obtained. Consequently, gradient detection is desired in order to obtain the object position, however, there are small gradients within the projections, which cause redundant co-ordinates. Therefore the obtained projections need to be smoothed or filtered before passing through the pixel difference block. A method used for smoothing is discussed in the next section.

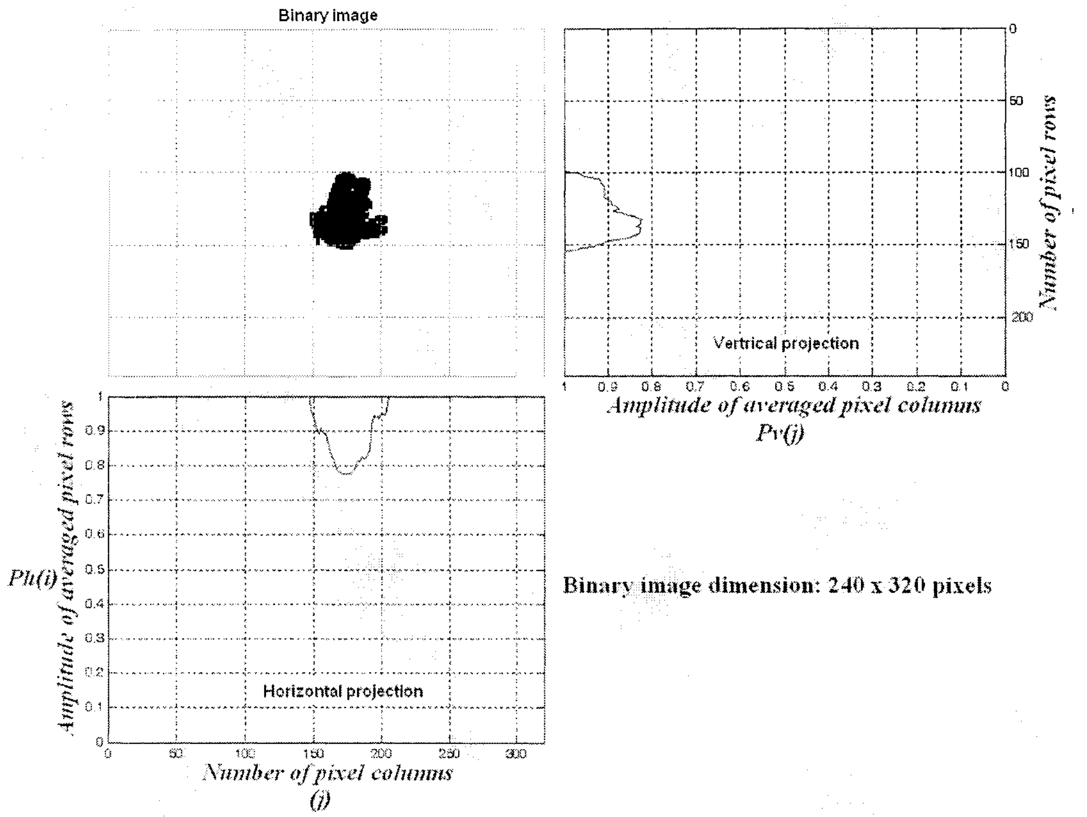


Figure 3.10: The horizontal and vertical projections

3.4.2 Smoothing method

Due to the nature of the obtained projections, some filtering is required because of the presence of small gradients or noise from the projection process. Several techniques are available for this operation. A simple moving average technique such as median filter [40] cannot eliminate as much of the noise in the projections as desired. Consequently, a more complex but more satisfactory type of smoothing is needed. Steinberg *et al.* [41] have proposed a smoothing technique which uses a parabolic function, which can reduce small point-to-point data variations, consequently, this technique is chosen for this application. Since this smoothing technique is based on a parabolic function it is referred to as the ***least square parabolic*** smoothing technique [41] for more details see appendix A.

The smoothed projection, $\boldsymbol{P_{smoothed}}$, is defined as [41]:

$$P_{smoothed}(l+i) = \frac{3}{4n(n^2-4)} \times \sum_{i=1}^{i=n} (3n^2 - 7 - 20k(i)^2) Px(l+i) \tag{3.8}$$

$$k = \left(-\frac{(n-1)}{2} \; to \; \frac{(n-1)}{2} \right) \; , \; l = b \times n \; \; and \; \; b = 0,1,2,...,m/n$$

where ***Px*** are the samples of projection which need to be smoothed, ***k*** the coefficients of the parabolic function, ***n*** the number of the projection or gradient samples in odd numbers that require smoothing and ***m*** the total number of the projection samples.

The number of the gradient samples, ***n***, can be adjusted in order to increase the length of the smoothing process, as will be explained in the experiment in Section 3.6. After smoothing by the least-square parabolic technique, the small gradients in the projections, ***Pv*** and ***Ph***, are reduced. An example with ***Ph*** has been performed showing smoothed ***Ph*** using 9 projection sample plots in Figure 3.11 (c). From these results in Figure 3.11, it can be seen that the least-square parabolic technique can eliminate the small amounts of noise within the projection. Although the slopes of high gradients are steeper than the slopes of the original projection, these are very useful for generating the co-ordinates that are used to locate the pedestrians. The merit of the steep slope can be seen when the smoothed projection passes to the next stage of the system described in the next section.

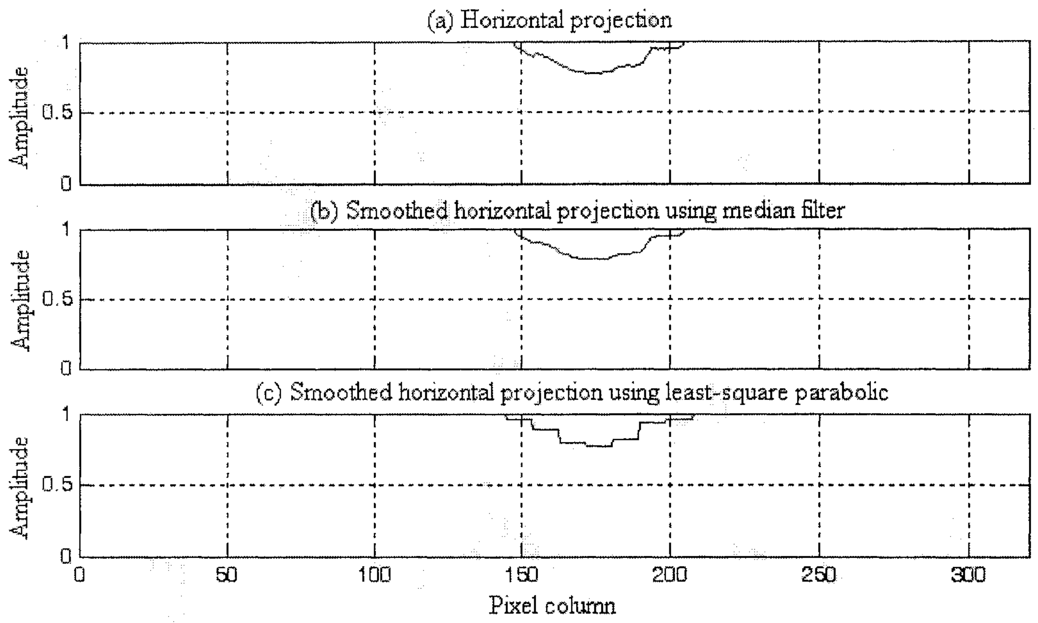


Figure 3.11: The horizontal projection and its smoothed

3.4.3 Pixel difference method

The smoothed projections are fed to a pixel difference block in order to generate the co-ordinates, which are obtained by detecting high gradients in the projections. The simplest method of gradient detection is to calculate the difference between pixels along the rows of ***Pv*** and the columns of ***Ph***. Before calculating the projections, the response of the gradient detection must be explained in order to clearly understand the method. The first calculation of the pixel difference is referred to as the first order pixel difference. The response of the first order pixel difference is positive when the slope of the gradient increases (See 2a in Figure 3.12). In contrast, the response is negative, as the slope of the gradient decreases (See 2b in Figure 3.12). If the response of the second order pixel difference is calculated then it will contain a zero crossing the direction of which is determined by the sign of the gradient. If the gradient or first order response is positive the zero crossing is in a negative direction [35] (See 3a in Figure 3.12) and vice versa (See 3b in Figure 3.12).

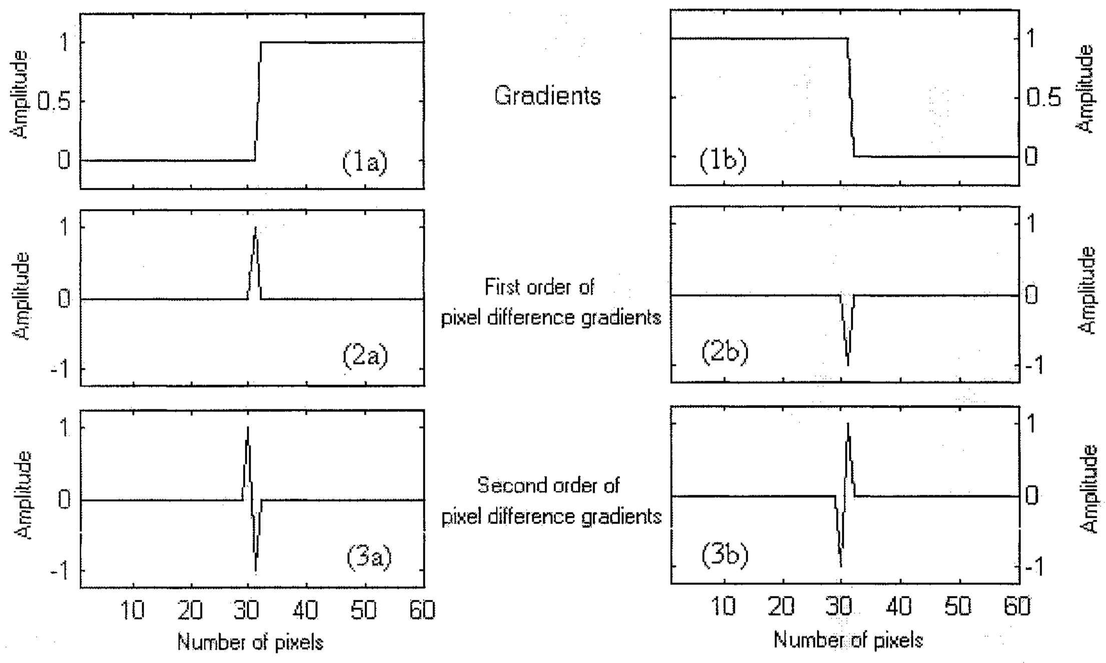


Figure 3.12: The responses of pixel difference detector

To obtain the co-ordinates, the first and second order pixel differences are applied to the projections. Due to both the increasing and decreasing gradients in the projections it is possible to indicate the object locations, the second order response is desired because its zero crossing provides the positive response for either increasing or decreasing gradients. Then only the positive positions of the second order response are found and used as components of co-ordinates that consist of row and column positions.

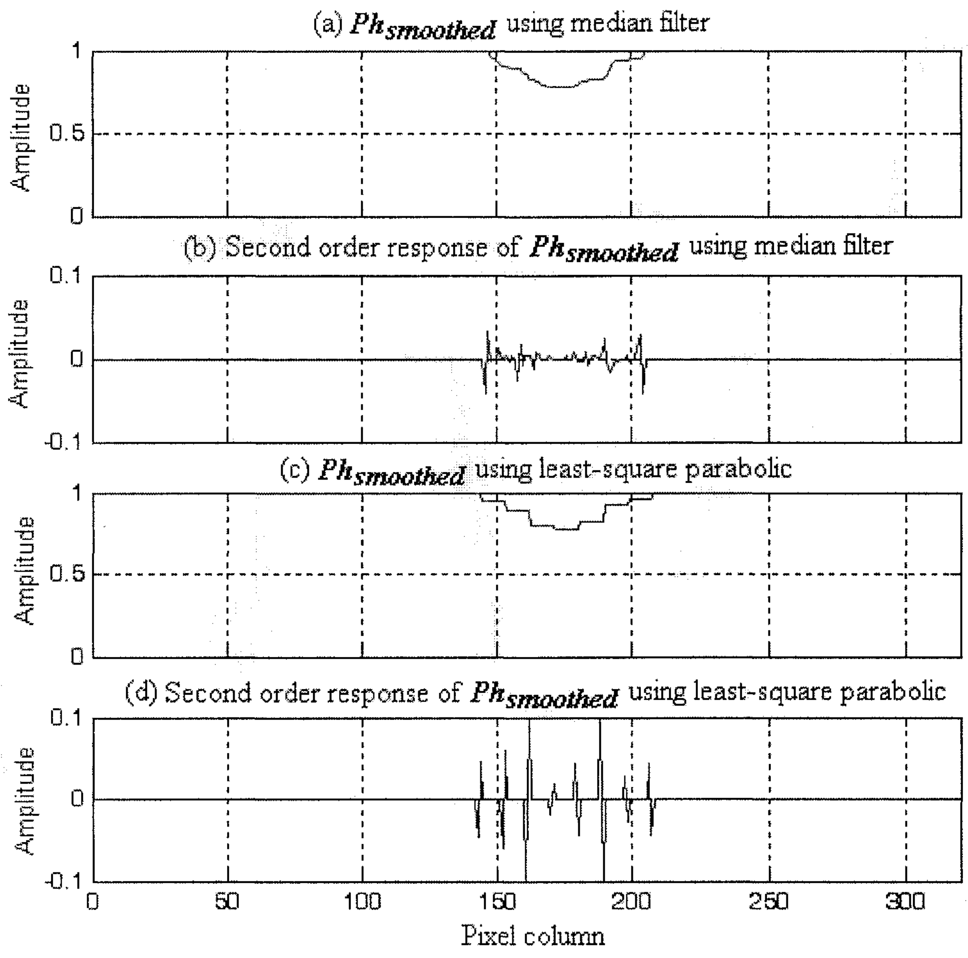


Figure 3.13: The comparison of the second order response of horizontal projection using two types of smoothing technique

The technique mentioned above is performed on ***Pv*** and ***Ph*** in order to obtain co-ordinates. Each positive position of the second order response provided from ***Pv*** and ***Ph***, is used as a row and a column component in each co-ordinate respectively. Consequently, the number of co-ordinates depends upon the number of high gradients that are provided from the projections. Figure 3.13 illustrates the second order response of $Ph_{smoothed}$ using two types of smoothing technique, median filter and the least square parabolic, in order to compare the results of these smoothing techniques.

From Figure 3.13, it is clear that the smoothing technique is a significant stage in the proposed object detection process because of the redundant co-ordinates that will be generated if the noise in the projections can not be eliminated, leading to costs in computing time. In addition, these results also show that a simple smoothing technique is not adequate in this application.

After the co-ordinates are obtained, they are passed through the image-marking block where, the input image is fed in and cross-signs are applied to it, located by the co-ordinates obtained providing a marked image. A comparison of the marked images obtained using both the median filter and least square parabolic with 9-projection sampling in the smoothing process, is illustrated in Figure 3.14.

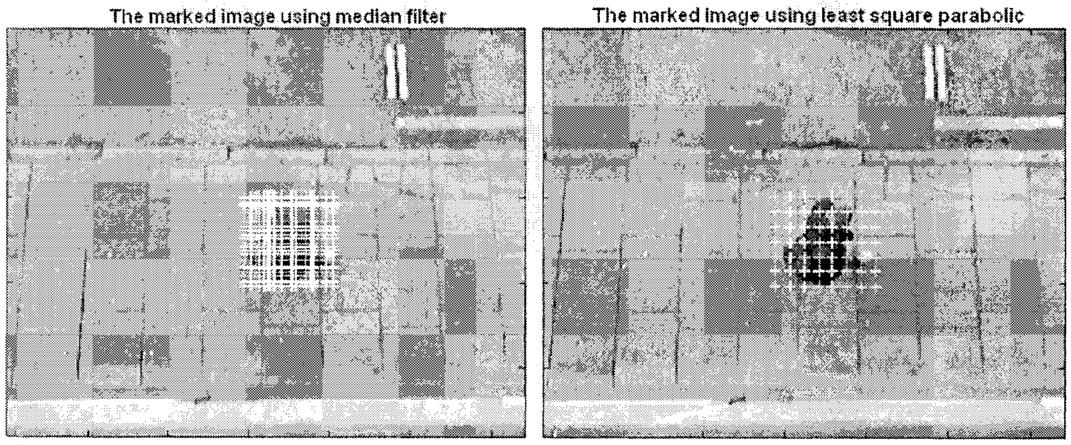

The marked image using median filter
The marked image using least square parabolic

Figure 3.14: The example of the object detection outcomes with two methods

As shown in the marked images, many cross-signs are marked on the image in and outside of the object area. In order to remove the co-ordinate positions outside of the object area, the co-ordinates have to be classified before they are plotted on the image. To classify the generated co-ordinates, it is necessary to ensure that each co-ordinate is in the object area. A mean value of a square metric that is captured the pixel of the co-ordinate and its neighbourhood from the binary image, is used to identify the co-ordinate locations in the area of interest. If the square metric mean is under 50% of the representative object value, it is classified as true and plotted on the image, this is because the mean value of the metric represents the connectivity of the pixel, otherwise the co-ordinate is ignored. An example of this classification process is shown in Figure 3.15 and the results of its application to the images in Figure 3.14 are illustrated in Figure 3.16.

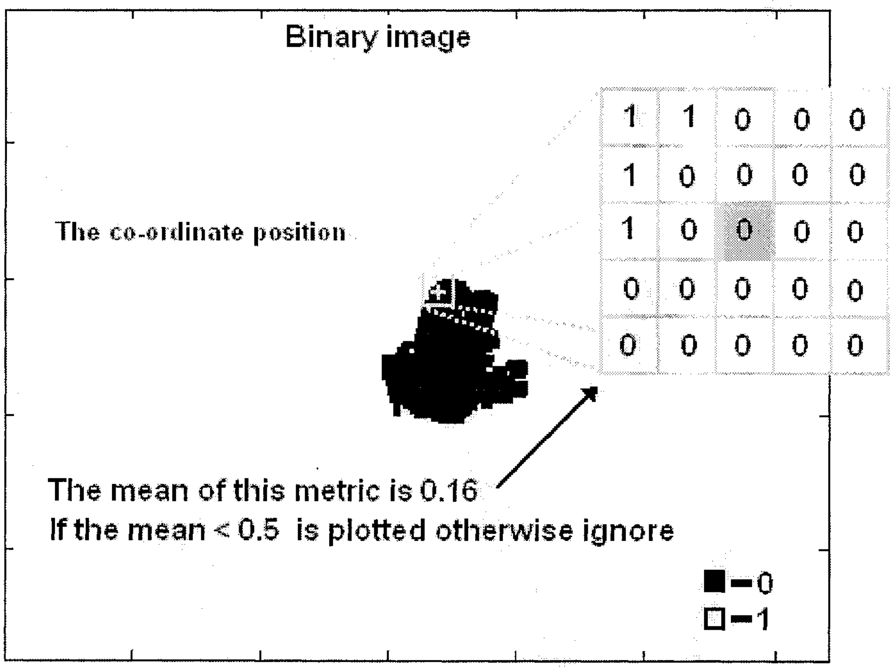


Figure 3.15: The process of co-ordinate classifying

Most of the co-ordinate positions are now located on the object area, however, there is still a large number of co-ordinates per object, where in fact, each object requires only a single cross-sign to identify it. This is because the two projections used to obtain the co-ordinates are smoothed.

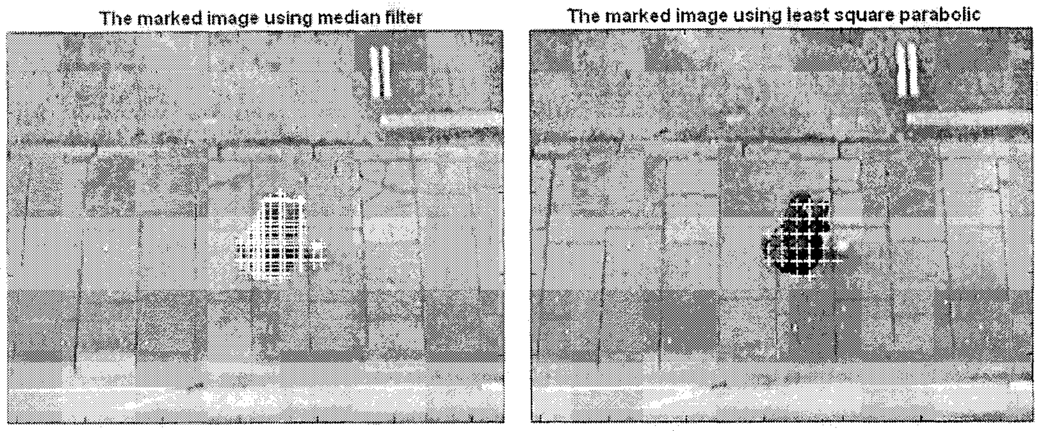

The marked image using median filter
The marked image using least square parabolic

Figure 3.16: The example of the object detection outcomes with two methods

The task of reducing the level of redundancy is focused on the sampling of the chosen smoothing method, i.e. the least square parabolic. By increasing the number of gradient samples of the least square parabolic, the projections of the binary image are smoother, which in turn results in a decrease in the number of co-ordinates obtained as listed in Table 3.1. This information is using in selecting of the gradient sample in the smoothing the projections of the binary image. The criterion for selecting the gradient samples will be discussed in details in Section 3.6. Some marked images obtained for different levels of sampling are shown in Figure 3.17. Also Figure 3.18 shows the number of gradient samples against the number of co-ordinates obtained.

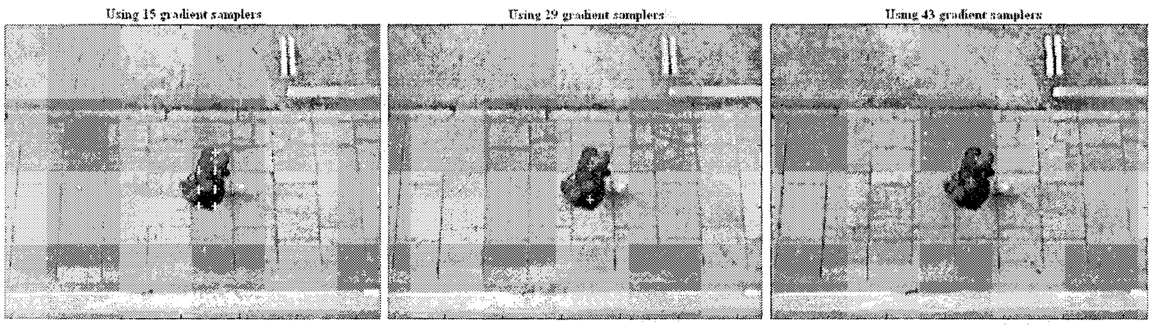

Using 15 gradient samplers
Using 29 gradient samplers
Using 43 gradient samplers

Figure 3.17: The results of marked images with various numbers of gradient samples

Table 3.1: An example result of increasing the number of gradient samples of the test image Figure 3.16

Number of gradient samples	Number of obtained co-ordinates	Number of gradient samples	Number of obtained co-ordinates
9	26	31	2
11	17	33	2
13	11	35	2
15	9	37	2
17	8	39	0
19	6	41	1
21	6	43	1
23	3	45	1
25	3	47	1
27	3	49	0
29	2	51	0

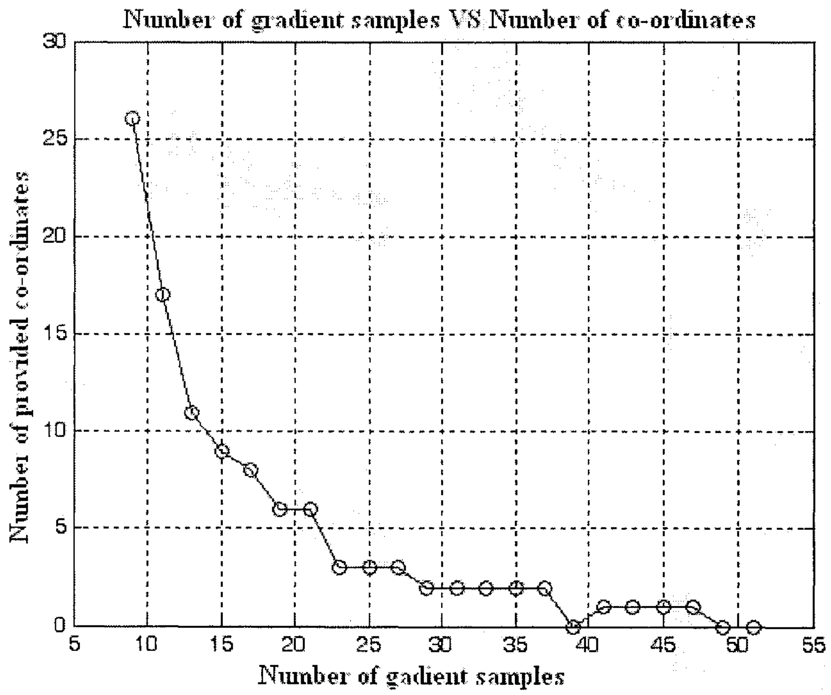


Figure 3.18: The graph represents the result of adjusting the number of gradient samples

Further experiments carried out on the adjustment of the gradient samples are discussed in Section 3.6.

3.5 Data collection

After the objects in the image have been located using the co-ordinates provided by the object detection process, the sub-images that contain people's heads need to be collected for use as data sets for classification which aims to classify people's heads. Unfortunately, the obtained co-ordinates can only be used to roughly locate the pedestrian, in most cases the location of pedestrian's head cannot be directly identified. However, the application of a window around the co-ordinates can solve this problem. The window selects a co-ordinate as a centre and has to be made big enough to incorporate the head, even in the event that the selected co-ordinate is on part of the body. From the information provided about the size of the pedestrian in Section 3.4, it is found that a suitable window size for this task is 64-by-64 pixels. Having selected the window area a sub-window large enough to contain a whole head is used to perform a sub-scan of the selected window region. A suitable size for the sub-window of 32-by-32 pixels was obtained by measuring test images. The sub-window is scanned through the region in a raster scan pattern using a shift of 4 pixels in each axis; each window will therefore provide 64 sub-windows. These sub-windows that may contain either head or non-head images are collected and used as training set in a recognition process to be discussed in the following chapters. An example of the data collection process is shown in Figure 3.19.

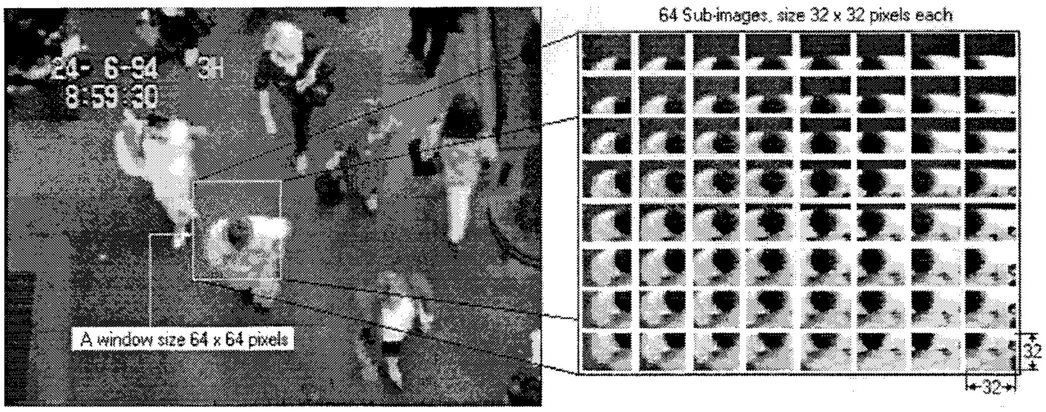

24- 6-94
3H
8:59:30
A window size 64 x 64 pixels
64 Sub-images, size 32 x 32 pixels each
32
32

Figure 3.19: Image data that obtains from one window

3.6 Experiments

In this section the experiments performed to test the detection method are explained and the results obtained are presented. The experiments involve threshold adjustment, changing the size and shape of the neighbourhood in the erosion operation, changing the number of gradient samples in the smoothing process and simulation of the object detector as a whole.

3.6.1 Threshold adjustment

This experiment considers the effects of adjusting the threshold level with regard to the appearance of the threshold-image. The thresholding process is the first block of the detector having as its input the subtracted image it is therefore important that appropriate threshold settings are used since the quality of the output will affect the remainder of the detection process. The initial threshold level is selected using the standard deviation (SD) of the absolute subtracted image; this is then increased using the multiplier (*T*) as in Eq. 3.4.

The two test images shown in Figure 3.20 are background subtracted and then applied to the thresholding block. The threshold setting is varied by the multiplier values that are increased in steps of 0.5 from 1 to 5.

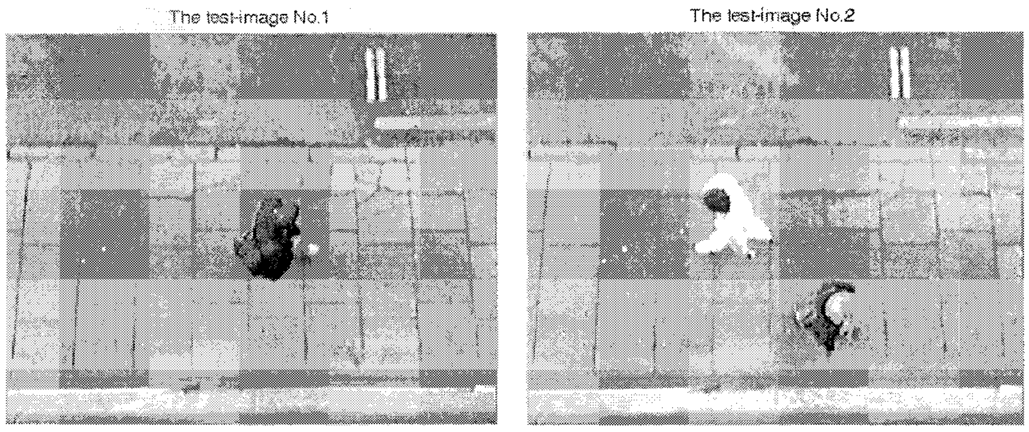


Figure 3.20: Two test-images for threshold adjusting

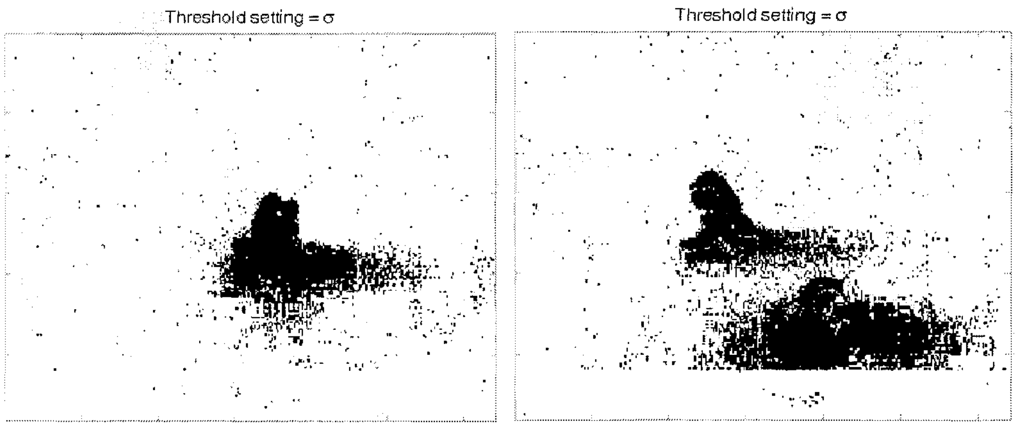


Figure 3.21: The results of threshold-image, threshold = σ.

In Figure 3.21 the threshold level is set at σ, the background and object shadow are still apparent in the images which indicates that some of the background intensities are larger than the threshold at σ. In Figure 3.22 the threshold level has been increased to 2.5σ, in this case almost all of the background pixels have been removed with some shadows remaining. Some of the object pixels have also been removed.

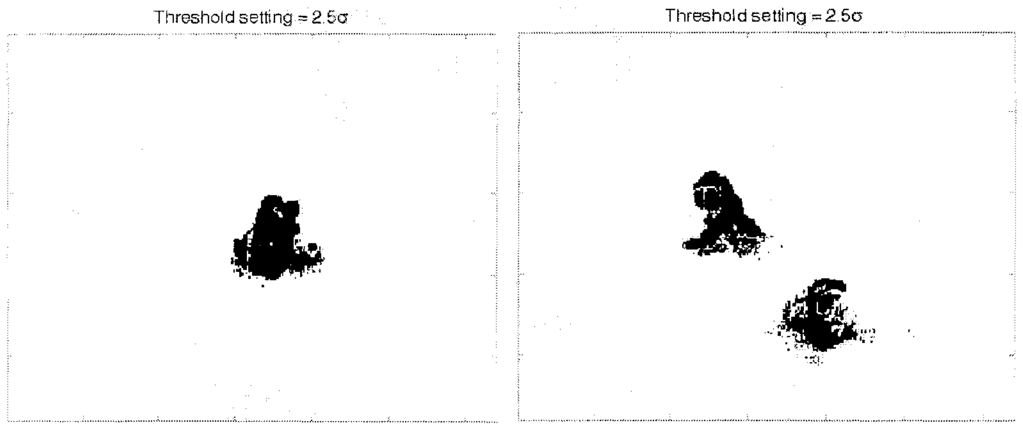


Figure 3.22: The results of threshold-image, threshold level at 2.5σ some residual background is still around the objects

By further increasing the threshold level, not only are most of the background pixels removed but also more object pixels. This is not necessarily a problem since the erosion process is able to replace some of the object pixels that have been removed. Figure 3.23 illustrates the results of thresholding the images at 3σ, 3.5σ and 5σ. The larger the threshold level becomes the more of the object information that is lost. Thus the criterion used to choose the threshold level is the point at which most of the background is removed for the loss of less of the object.

From the results presented, the choice of the thresholding level to produce the binary image transformation based on the given criterion is 3.5σ. Although some of the object information is lost, it can be enhanced by the erosion process discussed in the next section.

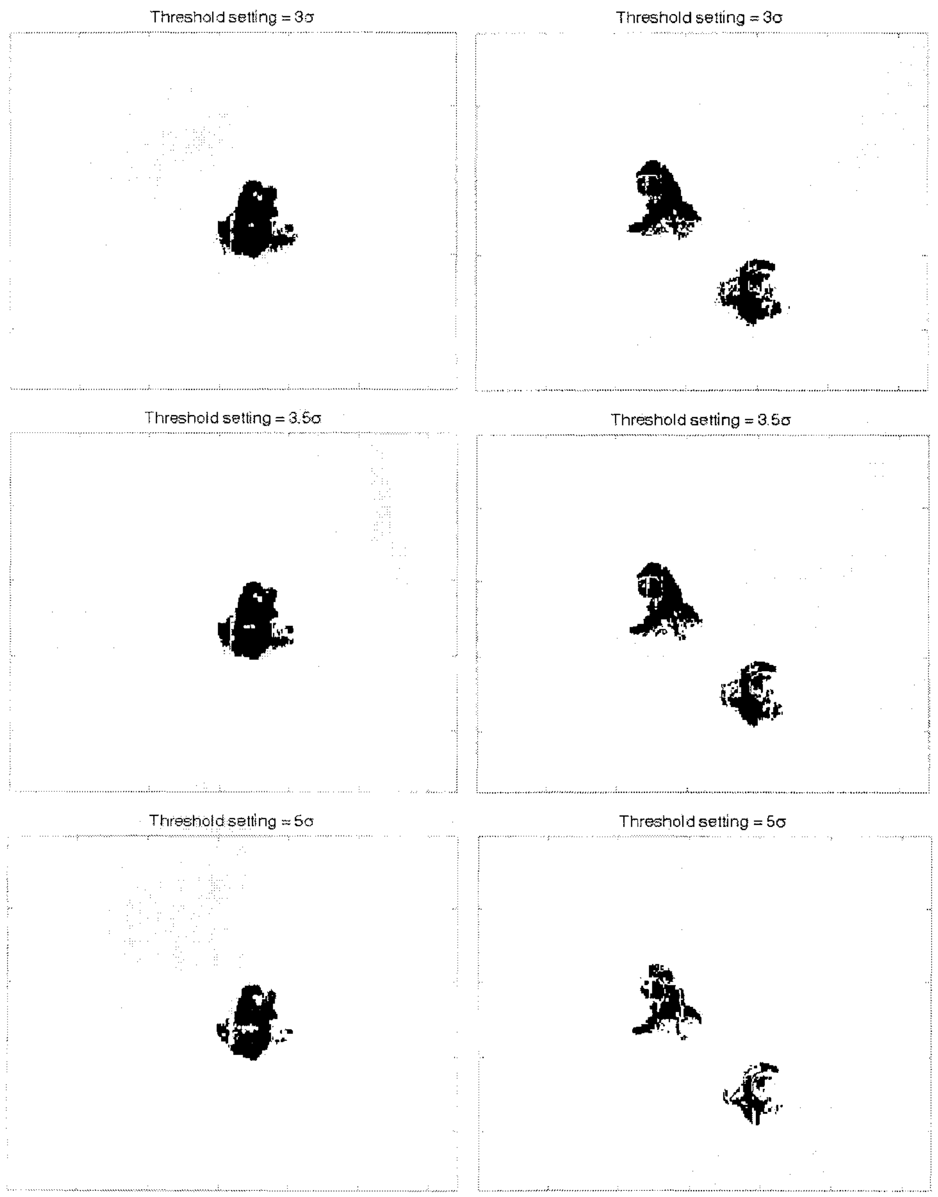


Figure 3.23: The results of threshold-image, threshold level at 3σ, 3.5σ and 5σ.

3.6.2 Changing shape and size of the structuring element in the erosion operation

In this experiment the shape and size of the structuring element is discussed in order to analyse the affect of the erosion operation on the binary image. The threshold level was set at 3.5σ according to the discussion in the previous section. The structuring element has a square shape of size of 3-by-3 pixels all containing 1's, as shown in Figure 3.24. The square is increased in size using odd numbers (i.e. 5-by-5, 7-by-7). The results of the tests are illustrated in Figures 3.25 and 3.26.

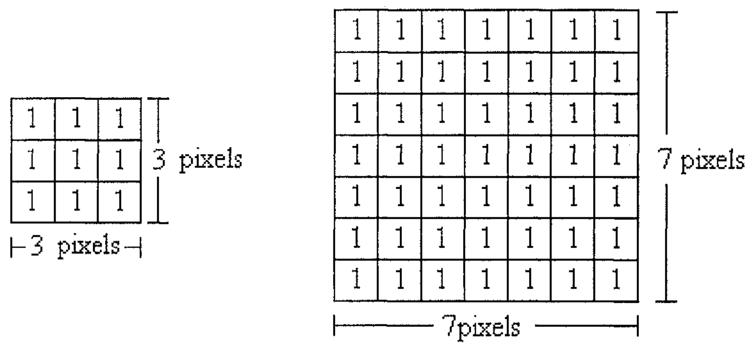


Figure 3.24: Structuring element size 3-by-3 pixels and 7-by-7 pixels

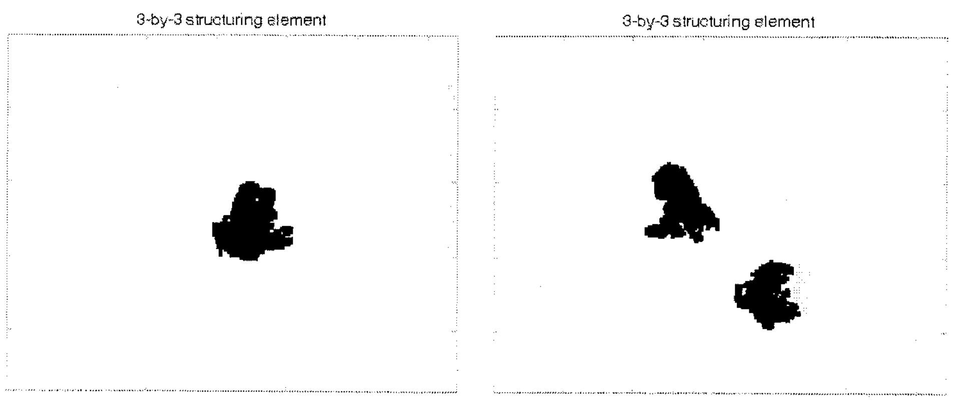


Figure 3.25: The enhanced image using 3-by-3 structuring element

From the images in Figure 3.25, it can be seen that the boundary disjoints of the image have been improved but the object still contains unfilled areas. Further increasing the size of the structuring element results in larger object dimensions though the unfilled pixels in the objects are now filled. Figure 3.26 shows the result of using a structuring element of 11-by-11 pixels.

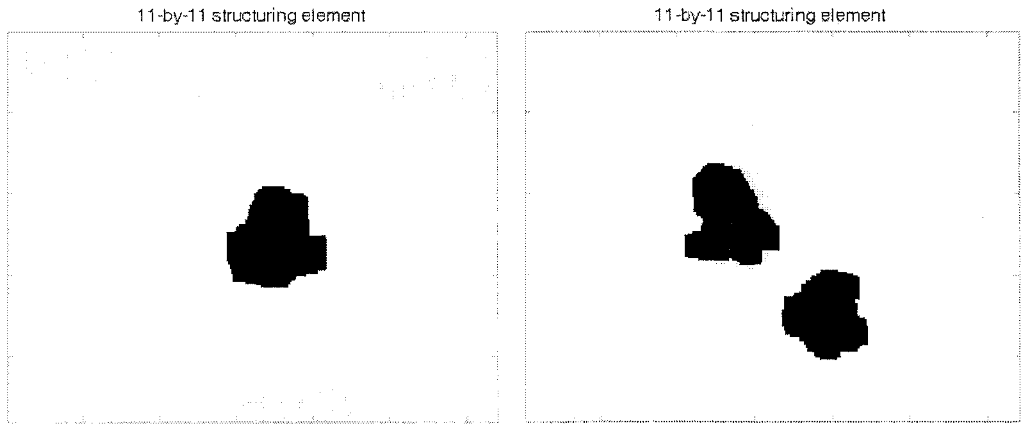


Figure 3.26: The enhanced images with 11-by-11 structuring element

From Figures 3.25 and 3.26 it has been noted that the larger the structuring element the larger the object dimensions become. The increase in the number of object pixels as the structuring element size increases is listed in Tables 3.2 and 3.3.

The three columns in the following tables show: the structuring element dimension, the increase in the number of object pixels after the erosion and the percentage increase in comparison to the image prior to its enhancement.

Table 3.2: The calculation results after erosion of the left column of Figure 3.23.

Structuring element dimension	Number of object pixels (pixels)	Pixel increasing compare to the threshold image (%) per object
Threshold image	1545	0
3-by-3	1942	25.7
5-by-5	2207	42.85
7-by-7	2456	58.96
9-by-9	2702	74.89
11-by-11	2952	91.1

Table 3.3: The calculation results after erosion of the right column of Figure 3.23.

Structuring element dimension	Number of object pixels (pixels)	Pixel increasing compare to the threshold image (%) per object
Threshold image	2033	0
3-by-3	2967	22.97
5-by-5	3488	35.78
7-by-7	3964	47.5
9-by-9	4431	58.98
11-by-11	4913	70.83

The results show that the number of the object pixels increases with the size of the structuring element. The object image area gets bigger and all the objects in the image are solid, also the object's shape is maintained. There is a trade off between increasing the object size and achieving a solid object shape. When the enhanced image is too large, when points are plotted onto the object some will be located outside of the actual object image, also the cost of unnecessary computation is introduced to the detector.

In our application, the 3-by-3 structuring element is found to provide satisfactory results. This decision is based on the fact that it is able to form a jointed object boundary, fill most of the empty pixels inside the object and has a negligible affect on the size of the enhanced image. Although, some of empty pixels still remain, it is very small number in comparison to the total number of object pixels.

Having selected two of the parameter settings for the blocks of the object detector. The last setting parameter, the number of gradient samples of the smoothing process, is now discussed in the next section.

3.6.3 Changing the number of gradient samples in the smoothing process

As previously demonstrated in Section 3.4.2, the smoothing of the horizontal and vertical projections can reduce the number of candidate co-ordinates found. In order to obtain a smaller number of co-ordinates, more representative of the number of people in the image, the number of gradient samples in the smoothing process needs to be adjusted. The two gray-scale pedestrian images shown in Figure 3.20 are employed for testing. The initial conditions of the detection need to be defined before performing the test. In this experiment the initial parameters of the threshold and structuring element are set as shown in Table 3.4.

Table 3.4: Setting parameters

Parameters	Setting values
Gradient samples	Adjusting
Threshold	3.5σ
Structuring element of erosion operation	3-by-3 pixels

The initial number of gradient samples used is 9 (the smoothing at this sampling level is negligible), this is then increased to 51 samples. The number of co-ordinates, marked on the object in the test image is observed, while the step of gradient sample increases. The relation between the gradient samples and the number of co-ordinate is used to evaluate the smoothing process; the results are shown in Tables 3.5 and 3.6.

Table 3.5: The results of increasing the number of gradient samples of the test image that contains an object in Figure 3.20 No.1 (left column)

Number of gradient samples	Number of obtained co-ordinates	Number of gradient samples	Number of obtained co-ordinates
9	26	31	2
11	17	33	2
13	11	35	2
15	9	37	2
17	8	39	0
19	6	41	1
21	6	43	1
23	3	45	1
25	3	47	1
27	3	49	0
29	2	51	0

Table 3.6: The results of inceasing the number of gradient samples of the test image that contains 2 objects in Figure 3.20 No.2 (right column)

Number of gradient samples	Number of obtained co-ordinates	Number of gradient samples	Number of obtained co-ordinates
9	39	31	5
11	28	33	4
13	18	35	3
15	15	37	2
17	11	39	2
19	8	41	2
21	7	43	1
23	5	45	3
25	5	47	1
27	5	49	1
29	3	51	1

In the tables the first and third columns show the number of gradient samples and their resulting co-ordinate responses are shown in columns two and four. The results in Tables 3.5 and 3.6 show that as the number of gradient samples increases then the number of co-ordinates decreases. A graph of these results is shown in Figure 3.27. Some examples of the results of the smoothing process are illustrated in Figures 3.28 and 3.29.

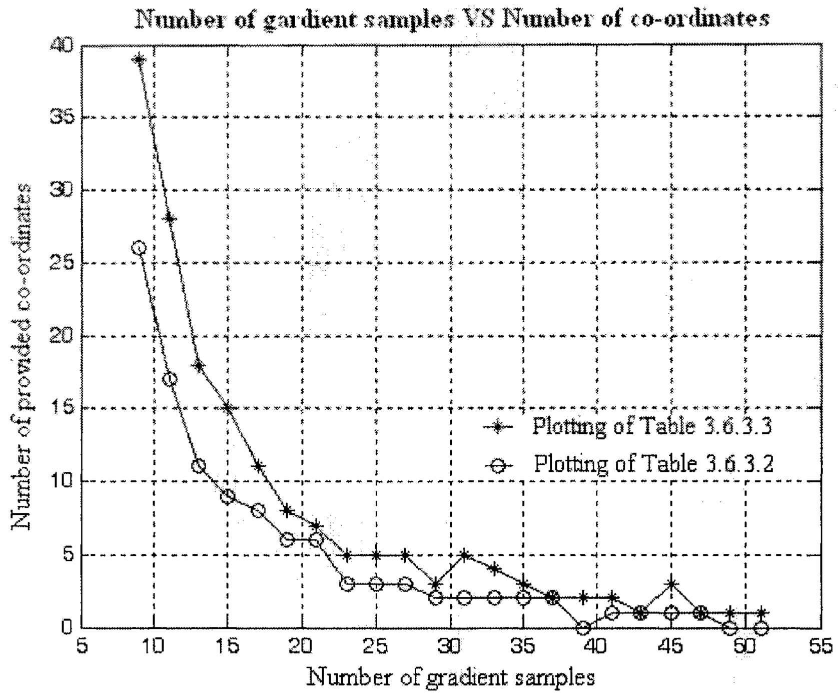


Figure 3.27: The plotting of the data from Tables 3.5 and 3.6

Increasing the number of samples in the smoothing process the small gradients in the projections are minimised, which results in a decrease in co-ordinates. From this result a suitable number of smoothing samples is obtained and is used as the smoothing parameter setting. A small number of co-ordinates are desired as a reduction in the number of redundant co-ordinates is obtained, minimising the cost in computing time and the number of images in the data set. By examining the number of gradient samples that provides a small number of co-ordinates at satisfactory error levels in people counting, a range from 25 to 39 gradient samples is deemed suitable. Some co-ordinate errors are still obtained, however these errors are useful for the training stage as they provide a balanced data set [42,43]. Consequently, these numbers of gradient samples are utilized in the following investigations.

False locations may be caused due to shadows or limbs of the pedestrians' bodies (e.g. legs), which cannot be eliminated by the background subtraction or thresholding processes. Some of these errors can be minimised in the smoothing stage, some errors still remain though a greatly reduced number.

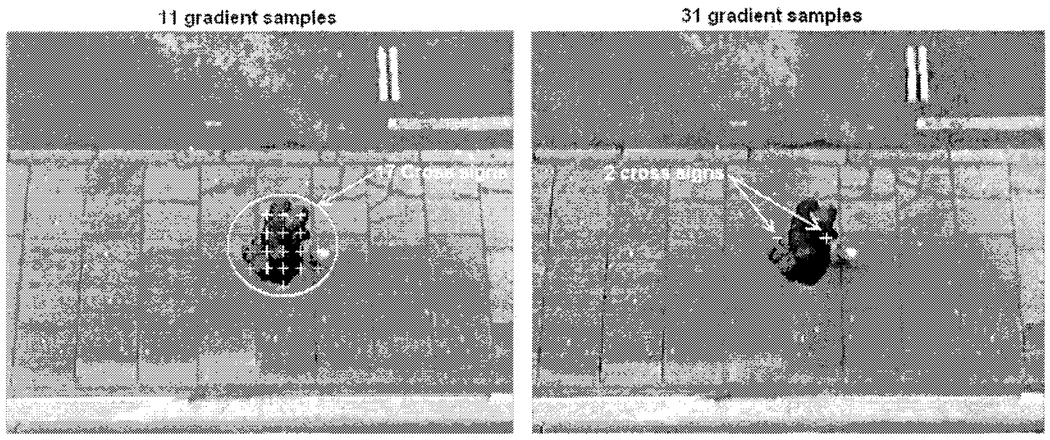


Figure 3.28: Outcomes of the smoothing process using 11 and 31 gradient samples of the Figure 3.20 No.1 (left column)

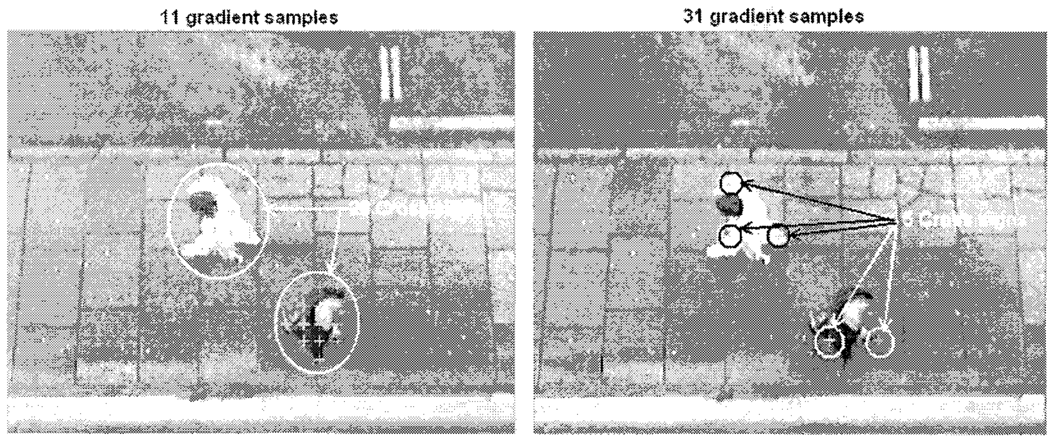


Figure 3.29: Outcomes of the smoothing process using 11 and 31 gradient samples of the Figure 3.20 No.2 (right column)

3.6.4 Performing the presented method on sequential images

In this experiment, sequential images are used to evaluate the proposed method. Two sequences of 13 images are captured with the first image of each sequence shown in Figure 3.30. The threshold and the number of gradient samples are discussed. First, an experiment is carried out on the threshold level and discussed in terms of sensitivity to the detection outcome. The parameter settings discussed in the previous sections are applied in the test. These values are listed in Table 3.7.

Table 3.7: Setting parameters for testing on threshold changing

Parameters	Setting values
Gradient samples	31
Threshold	Adjusting
Structuring element of erosion operation	3-by-3 pixels

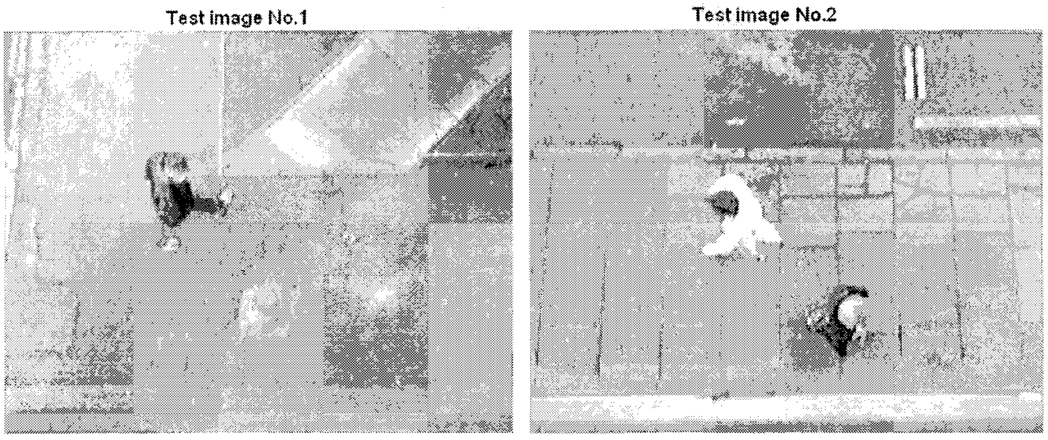


Figure 3.30: The images for testing from two different sources of capturing

According to the previous discussion in Section 3.6.1, the initial threshold setting is 3.5σ, this is varied by the $\pm\sigma$ to 2.5σ and 4.5σ.

Table 3.8: The results of adjusting the threshold level of the test image No.1 in the Figure 3.30 contains one person

Image frame No.	Threshold levels				
	2.5σ	3σ	3.5σ	4σ	4.5σ
1	2	2	2	2	2
2	1	1	1	1	1
3	2	2	1	1	1
4	2	2	2	2	2
5	3	2	2	2	2
6	3	3	2	2	2
7	1	1	1	1	1
8	1	1	1	1	1
9	4	2	1	1	1
10	2	2	2	2	2
11	2	2	2	2	2
12	3	3	2	2	2
13	2	2	1	1	1

Table 3.9: The results of adjusting the threshold level of the test image No.2 in the Figure 3.30 (contains 2 persons)

Image frame No.	Threshold levels				
	2.5σ	3σ	3.5σ	4σ	4.5σ
1	3	3	3	2	2
2	4	3	3	3	3
3	4	4	2	2	2
4	4	4	4	4	4
5	5	5	5	4	4
6	5	4	4	4	4
7	4	4	4	4	4
8	3	3	2	2	2
9	7	5	5	5	5
10	6	6	6	6	6
11	3	3	3	2	2
12	4	3	3	3	2
13	4	3	3	3	3

The first column shows the number of the image in the sequence. The remaining columns show the number of co-ordinates plotted on the images with regards to the threshold level. The number of co-ordinates is used to evaluate the sensitivity of the detector as the threshold is adjusted.

The results in Tables 3.8 and 3.9 show that the change in threshold level through ±100% of standard deviation value has little effect on the number of co-ordinates obtained on the output image. Consequently, it can be concluded that the detector is not sensitive to the changing of threshold levels.

The second experiment on the sequence of images is based on the choice of the number of gradient samples. As aforementioned, the range of gradient samples in which suitable results are expected to lie is 25 to 39 samples, this range was obtained using the two initial images. To provide the actual number for use in the detector an investigation of the effect of varying the number of smoothing samples on the sequence of images is necessary. The parameter settings are listed in Table 3.10.

Table 3.10: Setting parameters for testing on gradient samples changing

Parameters	Setting values
Gradient samples	Adjusting from 25 to 39
Threshold	3.5σ
Structuring element of erosion operation	3-by-3 pixels

The initial the number of gradient samples is set as 25 and then increased in odd steps to 39 samples, this means that the experiment produces 16 tables of data, 8 for each of the images. It is unnecessary to display all of this data here, it is sufficient to observe a selection of the results that provide an insight that allows choice of the best number of gradient samples to be made. Table 3.11 expresses the experimental results attained from the sequence following test image No.1 in Figure 3.30 and Table 3.12 shows the results attained from the sequence following test image No.2 in Figure 3.30; these images contain one and two persons respectively.

Table 3.11: The results of 25 gradient samples of image No.1 in Figure 3.30

Image frame no.	Number of obtained co-ordinates	Number of co-ordinates correspond to object location	Number of co-ordinates fail to locate object	Number of object that error to locate	Number of objects that correspond by co-ordinates
1	2	2	0	0	1
2	4	4	0	0	1
3	3	2	1	0	1
4	2	2	0	0	1
5	3	2	1	0	1
6	4	3	1	0	1
7	2	2	0	0	1
8	3	2	1	0	1
9	3	2	1	0	1
10	3	2	1	0	1

Table 3.12: The results of 33gradient samples of image No.2 in Figure 3.30

Image frame no.	Number of obtained co-ordinates	Number of co-ordinates correspond to object location	Number of co-ordinates fail to locate object	Number of object that error to locate	Number of objects that corresponded by co-ordinates
1	3	3	0	0	2
2	1	1	0	1	1
3	3	3	0	0	2
4	4	4	0	0	2
5	4	4	0	0	2
6	3	3	0	1	1
7	4	4	0	0	2
8	2	2	0	1	1
9	3	3	0	0	2
10	4	4	0	0	2

The first column of the table lists image number in the sequence the second column shows the number of co-ordinates obtained from the detector. The number of these co-ordinates that correspond to actual object locations and the number that fail to locate an object (as shown in Figures 3.31 and 3.32) are given in the third and the forth columns respectively. The fifth column represents the number of objects the detector fails to locate (as demonstrated in Figure 3.33) and the last column shows the number of objects corresponding to by the co-ordinates.

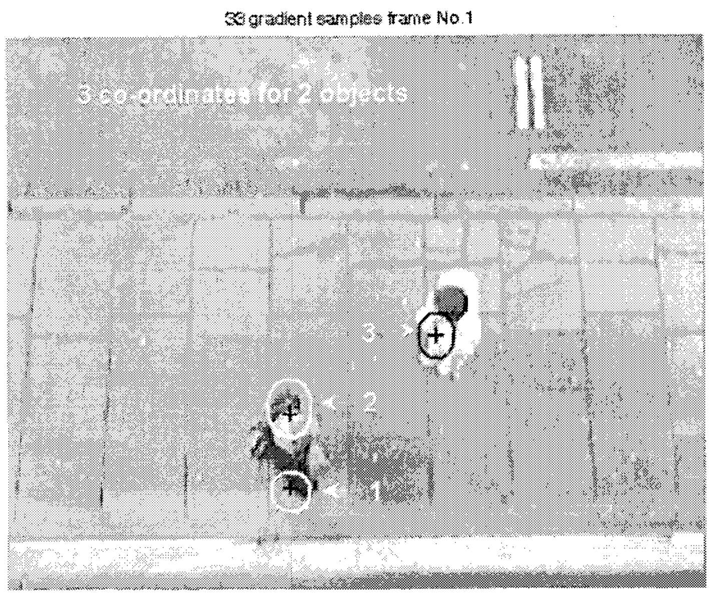


Figure 3.31: Demonstrates the number of co-ordinates correspond to the objects

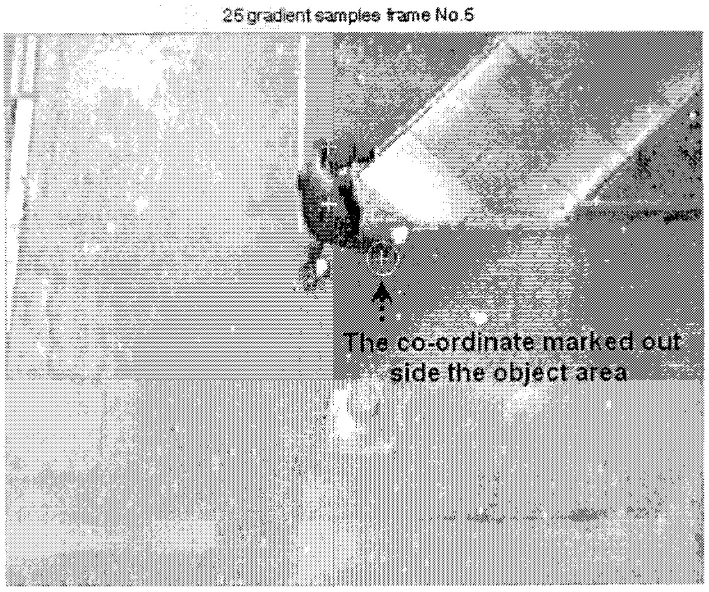

25 gradient samples frame No.5
The co-ordinate marked out side the object area

Figure 3.32: Shows the co-ordinates that fail to plot on the object area

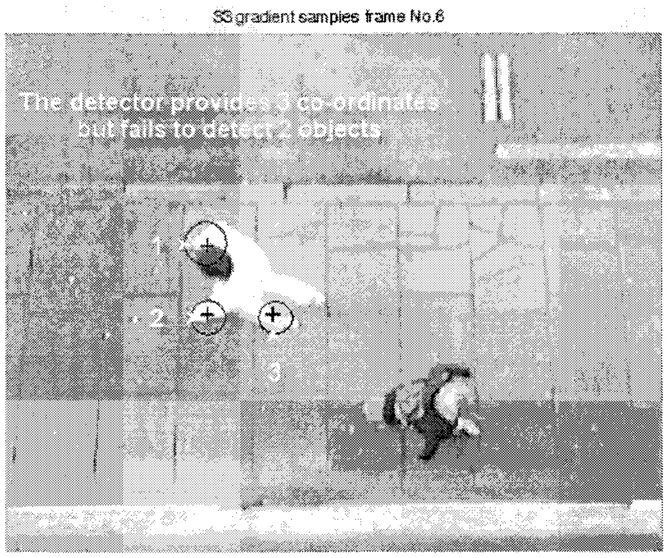

33 gradient samples frame No.6
The detector provides 3 co-ordinates but fails to detect 2 objects
1
2
3

Figure 3.33: The detector fails to detect two objects in the image

From the results of the investigation, the number of gradient samples to be used is chosen as 31, as this provides the most satisfactory outcome. The correct numbers of objects are detected in every frame, for both testing images, for only a small number of errors. Consequently, this number of gradient samples is employed as the setting for the smoothing parameter in the detector. All of the parameters for proposed the object detection method have now been defined. The final object detector is now tested using a different set of sequential images to allow evaluation of the method in the next section.

3.6.5 The object detector simulation

To evaluate the object detection method presented, an investigation utilizing one hundred sequential images will be employed. The threshold, erosion operator and smoothing coefficient are as discussed in the previous sections; summarized in Table 3.13.

Table 3.13: Setting parameters of the object detector

Parameters	Setting values
Gradient samples	31
Threshold levels	3.5σ
Structuring element of erosion operation	3-by-3 pixels

For each of the one hundred images the number of people identified by the detector is compared with the actual number of people obtained manually by counting. This allows the accuracy of the detector to be found as a percentage. The detector is able to provide an accuracy of around 80.2% over the one hundred images. There are several possible causes of the failure of the detector. Firstly, the outcome from background subtraction and binary transformation process may produce pedestrian intensity levels close to those of the background intensity; this would result in their possible elimination by the thresholding block. Although some of its intensity may remain and could be improved by the erosion stage, it may not be sufficient to provide a high enough gradient in the horizontal and vertical projections. Secondly, if the pedestrian's location is very close to the edge of image frame this can also result in failure, since if the distance between the obtained co-ordinate and the edge of the image is very small, then marking is ignored by the marking technique used. Finally, the number of gradient samples used in the smoothing technique may need refining as it was obtained using only two test images. Some of the results from the detection process are illustrated in Figure 3.34. The co-ordinates obtained which represent people's locations are used to create a training set for a neural network.

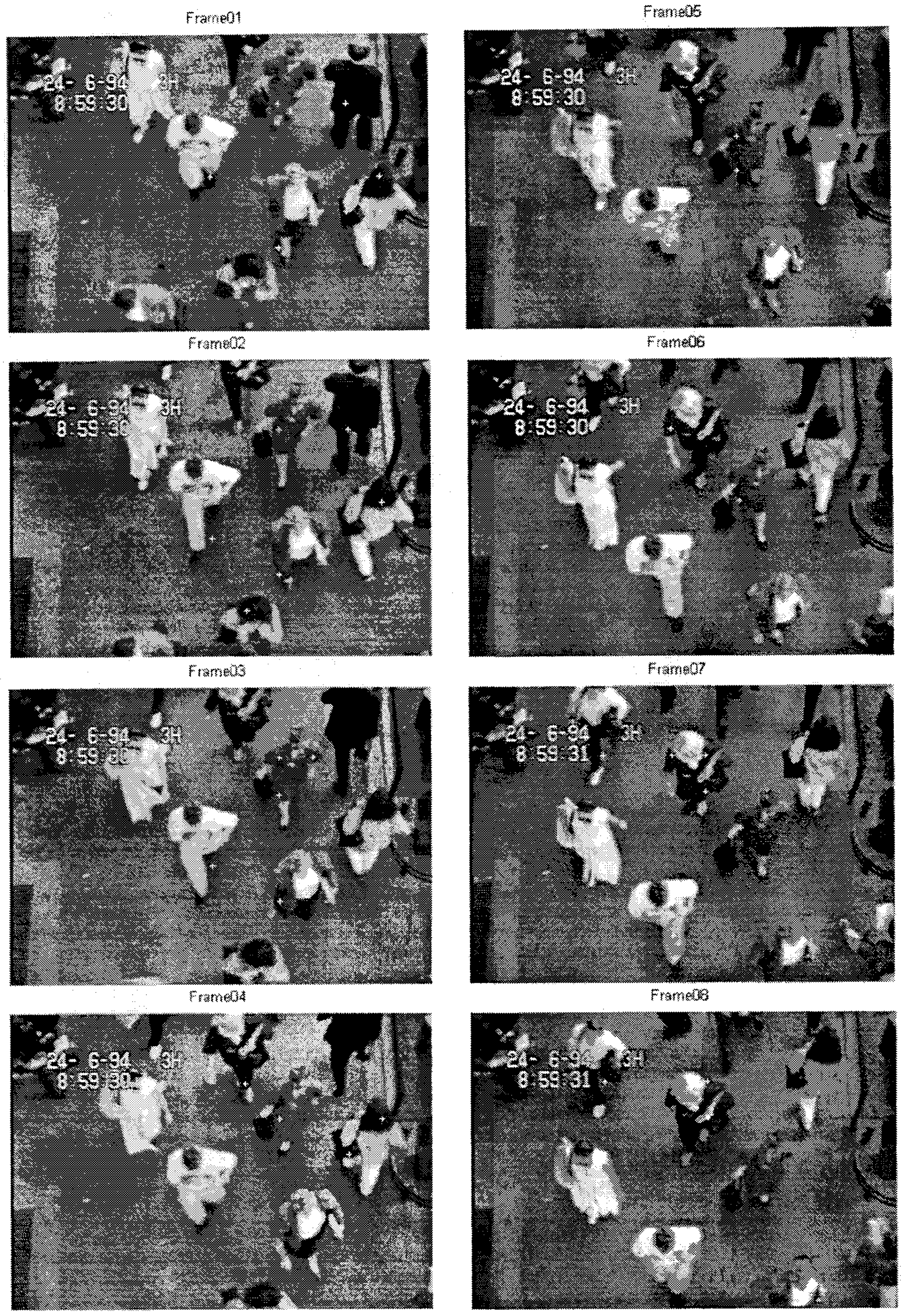


Figure 3.34: The outcomes of the detector (Frame01-Frame08)

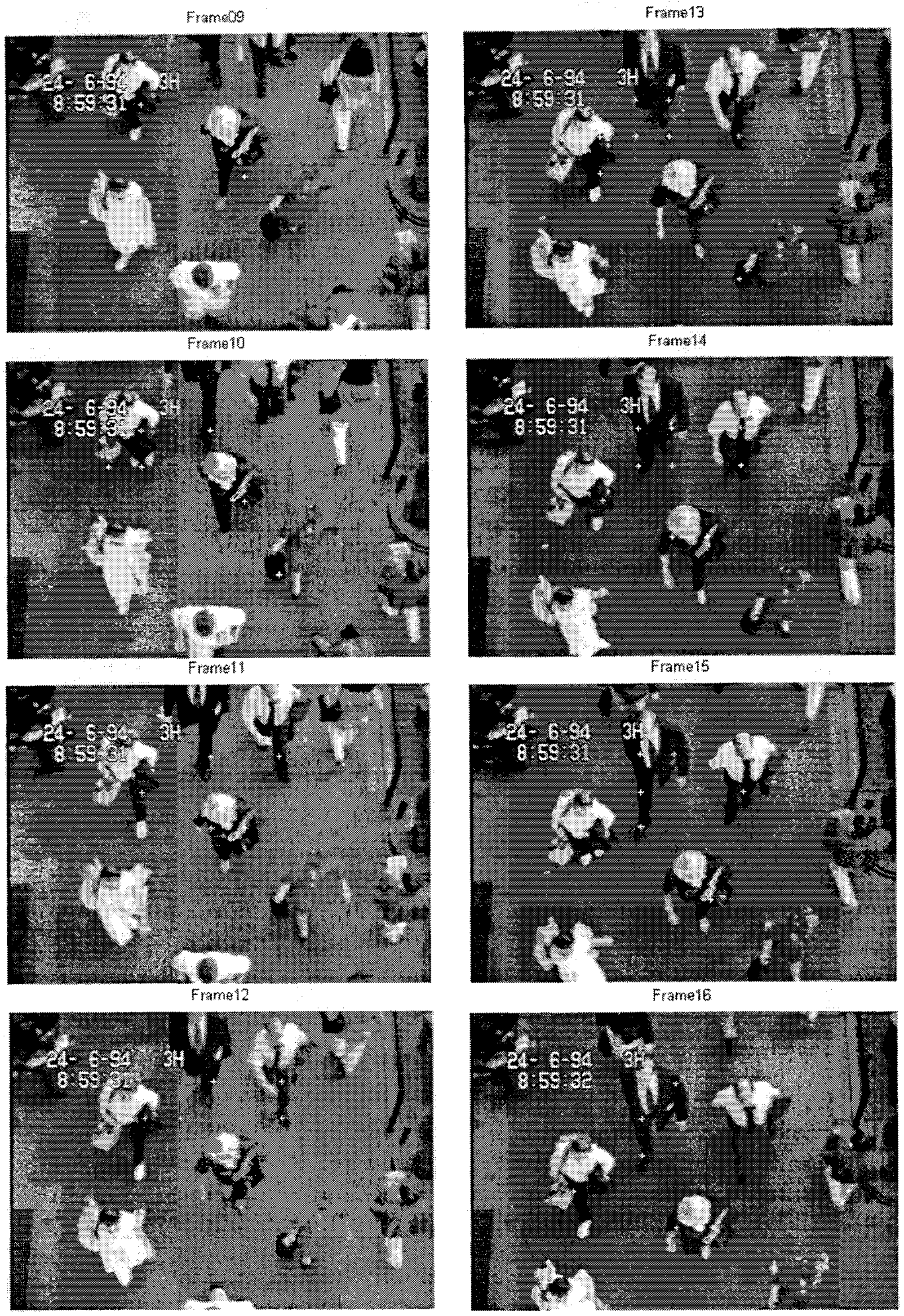


Figure 3.34: The outcomes of the detector (Frame09-Frame16)

3.7 Summary and discussion

An object detection method has been described and discussed. The three parameters required by the detector: the threshold level, structuring element size and shape and the number of gradient samples, have been investigated and suitable values determined. By experimentation it has been shown that the accuracy of the detector is not sensitive to the choice of threshold level within a given range, the dominant factor in determining the detector accuracy is the choice of smoothing parameter.

Although the technique does not provide 100% accuracy, it is sufficient for using as a data collection method. Due to its small time consumption, approximately 1 second per image for the whole detection process, an abundance of data that can be obtained in a short period of time. As aforementioned, provided that the data contains error data as well as the people's head data, the training set obtained will be balanced and offer a suitable test of the recognition method. The data obtained needs to be selected and then passed through some pre-processing; this is presented in the next chapter, prior to information extraction for training of the artificial neural network.

Chapter Four

Feature extraction module (FEM)

4.1 Introduction

Feature extraction is a highly important part of the image classification process applied in this study, since before the image can be fully classified the features within the image must be correctly identified. It typically involves two stages: training and classification. The training stage involves the extraction of features for every image in a training set. In the classification stage, features are extracted from an unknown image and compared to the features of each image in the training set. The method of image analysis chosen for feature extraction is clearly critical to the success of the image analysis [44].

In this work the data image is grey-scale and as such contains some texture information that could be used to identify and distinguish a person's head within the image. Consequently, methods that can extract the texture information from the image are required. Over the past decade, a number of approaches for texture feature extraction have been proposed such as Fourier domain energy, co-occurrence matrices, texture energy measure, Markov random field models, second-order grey level statistics, two-dimensional autoregressive model and fractal dimension, to name but a few [45]. In most of these works, the texture of the feature is analysed for a single resolution, which provides results with varying degrees of precision and need to be improved. Recently, the texture extraction method based on multi-channel or multi-resolution analysis, such as Gabor transform and wavelets, has received a lot of attention [46] and some promising applications have been proposed [47]. Among these techniques, texture feature extraction based on the wavelet transform is found to provide some advantages over the Gabor technique [48].

Other than texture, Principal Component Analysis (PCA) or the Karhunen-Loeve transform, which have been used in the application of face recognition [49,50], could be applied to identify a person's head image. Since each person's head is different e.g. hairstyle of head shape. A technique that uses a combination of the wavelet transform and PCA has been proposed to improve the limitations of PCA in face recognition [51]. The feature extraction techniques within this chapter are mainly focused on the texture and PCA extraction methods.

The diagram of the FEM is shown in Figure 4.1. The data image is fed into the ***Pre-processing*** block where the data image is transformed into another form in order to reduce the dimensions of the image, but retain its significant information; this transformation is made using the ***wavelet transform*** **(WT)**. The result is a transformed image represented by ***wavelet coefficients***. These coefficients are then passed through a ***feature generation*** block where the coefficients are calculated and represented in term of features vectors, suitable for recognition. The two blocks of Figure 4.1 are explained in more detail in the following sections: Section 4.2 pre-processing method based on the wavelet transform technique and Section 4.3 presents feature generation methods. Section 4.4 shows the results of the FEM and finally a summary and discussion is provided in Section 4.5.

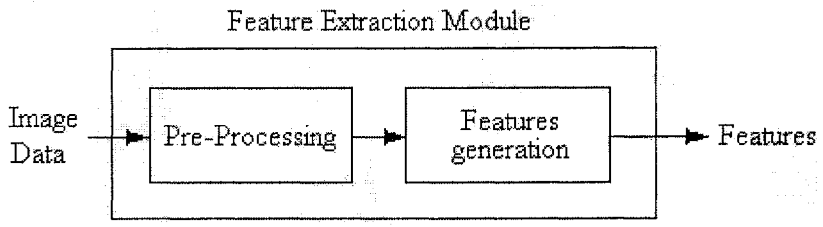


Figure 4.1: Diagram of Feature Extraction Module

4.2 Pre-processing method

Pre-processing is an important step in image recognition. It is used to reduce the size of the raw data before it is encoded into feature vectors suitable for classification. To achieve this goal, the wavelet transform (WT) is chosen because its properties make it an excellent tool for multi-resolution analysis, which is required for feature extraction [46]. In this section, the fundamental concepts of the wavelet transform are presented and described.

4.2.1 Wavelet transform (WT)

The wavelet transform is a technique for time-frequency analysis and has become widely used in signal processing areas. The wavelet transform was developed from three techniques: 1) filter bank theory, 2) multi-resolution or time-scale analysis, particularly using pyramid representations and 3) sub-band coding [52]. Consequently, the giving of a full wavelet theory overview would require more than a single book; some books and review papers available can be found in [53-57]. In this section the basic concepts, which are necessary for this application will be represented.

Since the data image is composed of two-dimensional digital signals the two-dimensional discrete wavelet transform (2-DWT) must be employed. The one-dimensional discrete wavelet transform (1-DWT) is explained first in section 4.2.1.1 since the 2-DWT is based on the 1-DWT. Then the more complex 2-DWT is discussed in section 4.2.1.2. The scaling and wavelet functions are discussed in section 4.2.1.3.

4.2.1.1 The one-dimensional discrete wavelet transform (1-DWT)

The basic process of the one-scale 1-DWT [55] is illustrated in Figure 4.2.

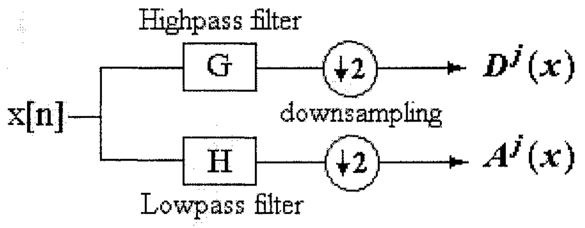


Figure 4.2: The basic process of the one-scale 1-DWT

The wavelet transform process consists of a complementary low-pass filter H and high-pass filter G. The transform can be performed on any one-dimensional signal, x[n], which is fed to H and G, where x[n] is convoluted with two sets of filter coefficients that are produced by two separate functions; the scaling function $\phi(t)$ for H, and wavelet function $\psi(t)$ for G. The details of these functions are discussed in section 4.2.3. The filtered signals are then downsampled to generate the outputs, which are a ***low-pass residue*** or ***approximation coefficients*** $A^j(x)$, and the ***high-pass subband*** or ***detail coefficients*** $D^j(x)$, each with half the length of x[n] as:

$$A^j(x) = \left[H * x^{j-1}\right]_{\downarrow 2} \qquad (4.1)$$

$$D^j(x) = \left[G * x^{j-1}\right]_{\downarrow 2} \qquad (4.2)$$

where the superscript (*j*) indicates scale, $\downarrow 2$ the downsampling along the filtered signal and $*$ denotes the convolution operator.

The total signal output from the transform is the same length as the input signal. The process of downsampling may introduce aliasing into the filtered signals. However, when the appropriate decomposition and reconstruction filters are used, the aliasing effect of downsampling is completely cancelled out and it is possible to perfectly reconstruct the original signal [56]. Since the WT in this work is applied for its ability to reduce the size of the data image, it is necessary to discuss the decomposition process only.

Iteration of the filtering and downsampling is performed on $A^j(x)$ as shown in Figure 4.3. The first level transformation provides the second-level approximation and detail coefficients. If the same process is continually applied to each successive approximation coefficient, the scale is increased at each level of analysis. As a result of iterating, the multi-resolution 1-DWT, of which x[n] is decomposed into several scales, is obtained. An example of a 3-scale 1-DWT scheme is shown in Figure 4.3.

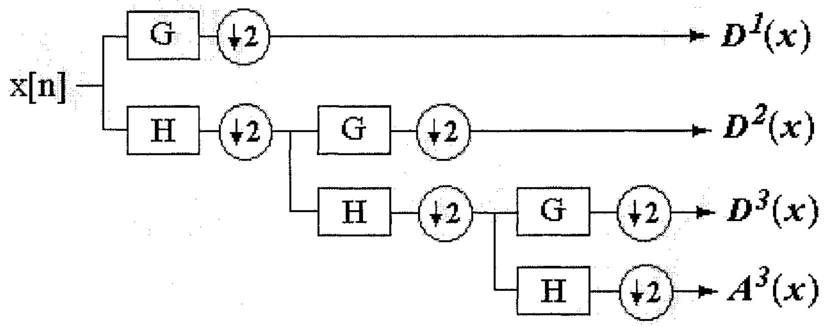


Figure 4.3: The 3-scale 1-DWT schemes

4.2.1.2 The two-dimensional discrete wavelet transform (2-DWT)

The two-dimensional discrete wavelet transform is more complex than the one-dimensional case; its structure is shown in Figure 4.4.

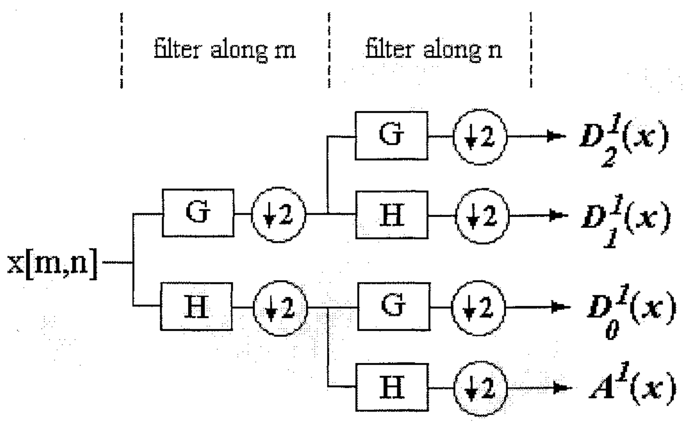


Figure 4.4: The process of the one-scale 2-DWT

As a digital image is a two-dimensional signal, the image transformation is performed using the 2-DWT, which is developed from the 1-DWT described in Section 4.2.1. The one-level or one-scale decomposition of the 2-DWT is accomplished by applying the 1-DWT to each row of the input image, x[m,n]. Following this the same process of transformation is applied along each column of the transformed rows. The results yield four types of coefficients for the one-level decomposition, which are named as follows: ***diagonal detail*** or ***corner coefficient*** $(D_2^1(x))$, ***horizontal detail coefficient*** $(D_1^1(x))$, ***vertical detail coefficient*** $(D_0^1(x))$ and ***low-pass residue*** or ***approximation coefficient*** $(A^1(x))$.

The computing of 2-DWT is shown as [52]:

$$A^j(x) = \left[Hy * \left[Hx * x^{j-1} \right]_{\downarrow 1,2} \right]_{\downarrow 2,1} \tag{4.3}$$

$$D_n^j(x) = \left[Gy * \left[Hx * x^{j-1} \right]_{\downarrow 1,2} \right]_{\downarrow 2,1} \tag{4.4}$$

$$D_n^j(x) = \left[Hy * \left[Gx * x^{j-1} \right]_{\downarrow 1,2} \right]_{\downarrow 2,1} \tag{4.5}$$

$$D_n^j(x) = \left[Gy * \left[Gx * x^{j-1} \right]_{\downarrow 1,2} \right]_{\downarrow 2,1} \tag{4.6}$$

where the subscript ***n*** (***n***=0,1,2) denote vertical, horizontal and diagonal detail coefficients, respectively. $\downarrow 2,1$ ($\downarrow 1,2$) the sub-sampling along the rows (columns). ***H*** and ***G*** are the low-pass and high-pass filters, respectively.

The approximation coefficient from the previous scale is used for transformation in the next scale and the same method is applied for the next scale until the desired scale is reached. The two-scale 2-DWT scheme is illustrated in Figures 4.5 and 4.6 shows three scales of decomposition of an image, x[m,n].

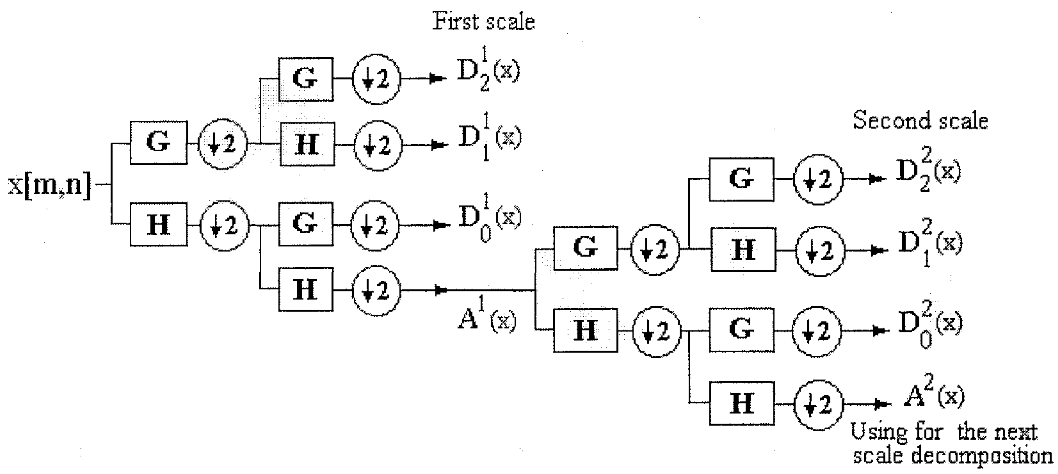


Figure 4.5: The process of the two-scale 2-DWT

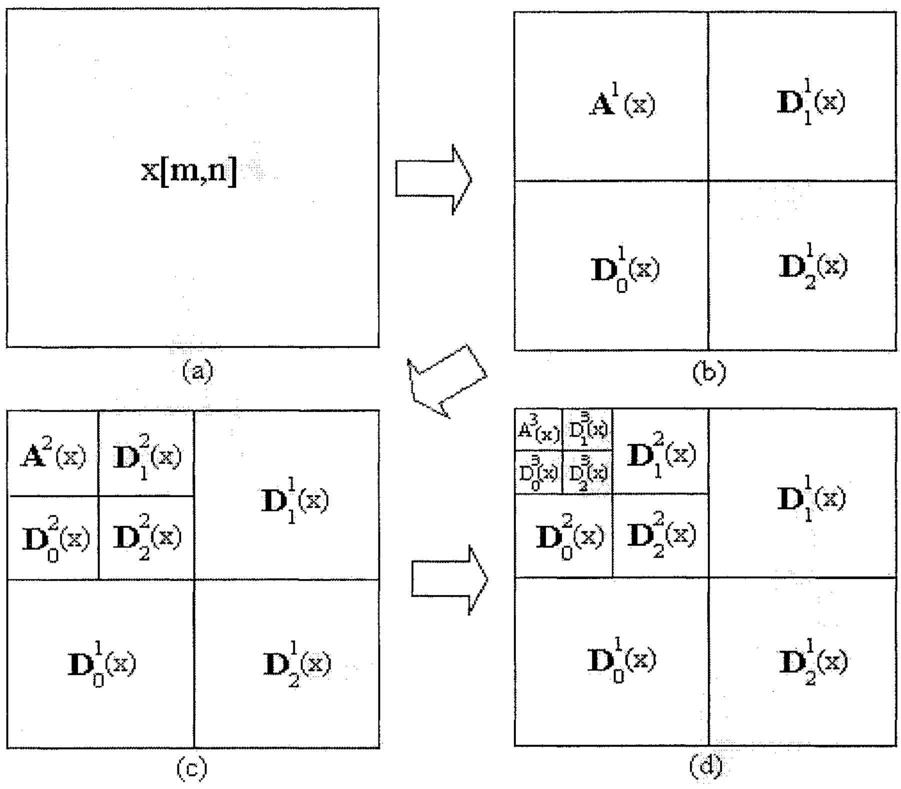


Figure 4.6: Steps of the three-scale 2-DWT decomposition: (a) original image; (b) first, (c) second and (d) third step.

4.2.1.3 Scaling and wavelet functions

The two sets of filter coefficient used in the low-pass and high-pass filters are obtained from the so-called ***scaling, $\phi(t)$*** and ***wavelet, $\psi(t)$*** functions respectively. There are a number of wavelet functions or wavelet families that exist. One of them is Daubechies wavelets, which was discovered by Ingrid Daubechies in 1987 [58]. It has become a well-known family of orthogonal wavelets, which offers some compromise between compact support and smoothness. This is needed for wavelet properties and makes discrete wavelet analysis practicable [56]. Daubehies wavelets have several types which produce different numbers of coefficients such as Daubechies 4 coefficients (Daub4), Daubechies 6 coefficients (Daub6), ... ,Daubechies 40 (Daub40) etc., however, it has been suggested that Daubechies 4 coefficients is suitable for applications of feature determination [53]. Moreover, the promising face recognition and the high texture classification performance of Daubechies wavelets are shown in [44,48]. Consequently, Daubechies 4 coefficients is chosen for this application. The Daub4 scaling and wavelet coefficients are shown in Table 4.1.

Table 4.1: List of scaling and wavelet coefficients of Daubechies 4 [53]

N	$\phi(t)$	$\psi(t)$
0	0.48296291314453	-0.12940952255126
1	0.83651630373781	-0.22414386804201
2	0.22414386804201	0.83651630373781
3	-0.12940952255126	-0.48296291314453

In order to demonstrate the transformation of the 2-DWT, Lena's gray-scale image, size 256x256 pixels, is used as the input image for the transformation. The transformed Lena's image is shown in Figure 4.7.

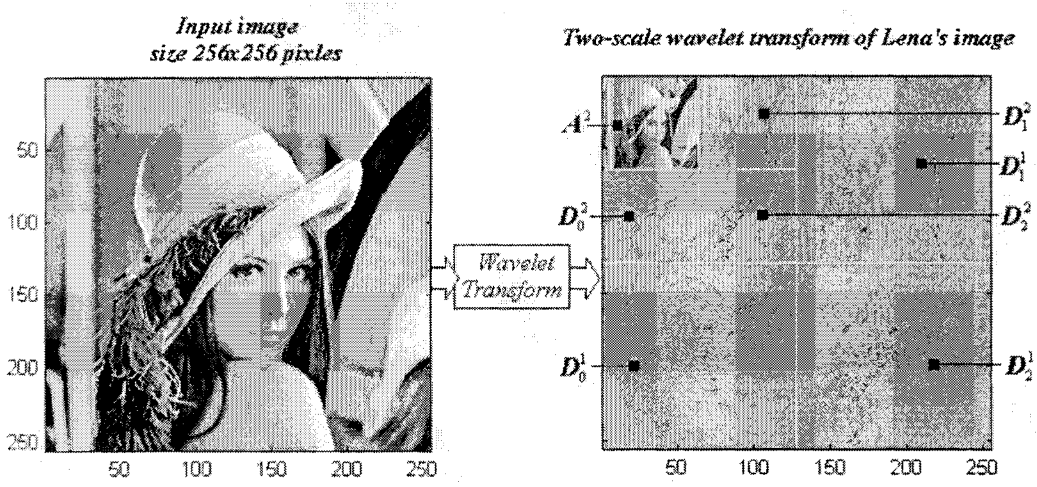


Figure 4.7: The two-scale 2-DWT of Lena's image

4.3 Feature generation

The three techniques for texture and a technique for Eigenvectors feature generation are used in this study. In this section the functions used for calculating these features from the transformed image are introduced and discussed.

4.3.1 Texture features

The wavelet detail coefficients or wavelet detail is found to be rich in statistical information, which can be used to identify the image in term of texture [59]. Using the wavelet detail the texture features are obtained by computing the following values: energy, mean deviation or l_1-*norm*, feature weighting of mean deviation and histogram qualitative.

4.3.1.1 Energy and Mean deviation signatures

The energy feature is a set of average energy values, which are obtained from each wavelet detail so it is referred to as a ***wavelet energy signature.*** The wavelet energy signatures reflect the distribution of energy along the frequency axis over scale and orientation and have proven to be very powerful for texture characterisation. Since most relevant texture information has been removed by low-pass filter iterating, the energy of the approximation coefficients is generally not considered a texture feature [59].

The energy of each wavelet detail coefficient is defined as [59]:

$$E_n^j = \frac{1}{N}\sum_{x,y}(D_n^j(b_x, b_y))^2 \qquad (4.7)$$

where E_n^j is the energy of each wavelet detail, D_n^j the wavelet detail at each scale, N the total number of the coefficients in D_n^j , b_x and b_y the row and column of D_n^j.

Similarity, the mean deviation (MD) or l_1 *-norm* feature of the wavelet details is referred to as a so-called ***wavelet mean deviation signature (MD signatures)*** and can be used as an alternative texture feature. The MD of each wavelet detail is defined as [59]:

$$MD_n^j = \frac{1}{N}\sum_{x,y}\left|D_n^j(b_x, b_y)\right| \qquad (4.8)$$

where MD_n^j is the mean deviation of each wavelet detail.

The numbers of vector components within one feature depend on the number of the wavelet details obtained from the decomposition. An example of a feature consisting of six vector components is obtained from the two-scale decomposition (See Figure 4.7).

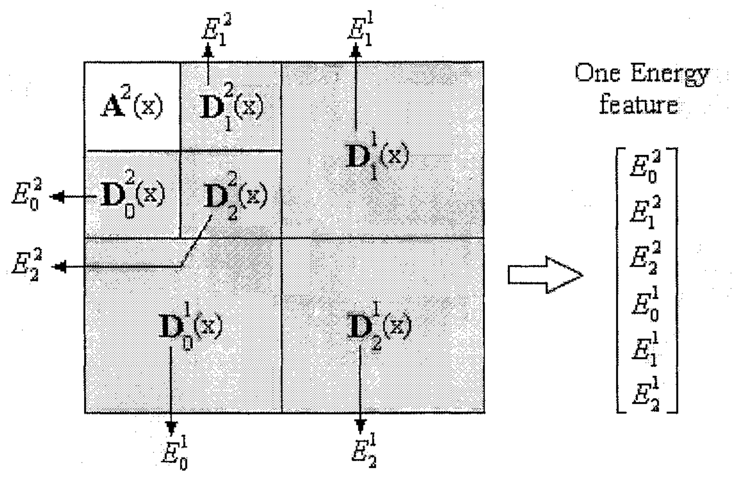


Figure 4.8: One energy feature obtained from the 2-scale 2-DWT image

4.3.1.2 The feature weighting

In 1999, Gaohong *et al.* [46] introduced feature weighting to improve texture classification. The result of using feature weighting has been shown to offer excellent improvement in the performance of texture classification. Consequently, feature weighting is chosen as a texture feature in this application.

Feature weighting is obtained by the weighting of the MD features (4.2) with their own degree of dispersion. Each MD feature (*Md*) contains a number of components as: $Md = [Md_1, Md_2, ..., Md_n]^T$ where ***n*** is the number of vector components in *Md* and the standard deviation (S_d) of *Md* is defined as [46]:

$$S_d = \sqrt{\frac{1}{n-1}\sum (Md_i - \bar{M})^2} \qquad i = 1,2,3,...,n \qquad (4.9)$$

where $\bar{M}$ is the mean value of Md defined as:

$$\bar{M} = \frac{1}{n}\sum_{i=1}^{n} Md_i \qquad (4.10)$$

Then the weighting vector components of Md are defined as:

$$Md_i^W = \frac{Md_i}{S_d} \qquad i = 1,2,3,...,n \qquad (4.11)$$

Therefore the feature weighting of Md is expressed as:

$$Md^W = [Md_1^W, Md_2^W, ..., Md_n^W]^T \qquad (4.12)$$

4.3.1.3 Histogram signatures

The Histogram qualitative of wavelet details can be used to provide texture features. Wouwer *et al.* [59] has introduced the texture feature using the parameters of histogram modeling, obtained from the wavelet detail. By using a family of exponential functions, the histogram of wavelet detail is modeled and two parameters of the model are provided and used as texture features. The first parameter is inversely proportional to the rate at which the peak decreases (β) and the second parameter is the width of the histogram peak (α) or (variance).

The model parameters α and β can be defined as [59]:

$$\alpha = m_1 \frac{\Gamma(1/\beta)}{\Gamma(2/\beta)} \quad where \quad \Gamma(x) = \int_0^\infty e^{-t} t^{x-1} dt \qquad (4.13)$$

$$\beta = F^{-1}\left(\frac{m_1^2}{m_2}\right) \quad where \quad F(x) = \frac{\Gamma^2(2/x)}{\Gamma(3/x)\Gamma(1/x)} \qquad (4.14)$$

where m_1 is the mean deviation value (4.8), m_2 the energy value(4.7).

Since the features are obtained from the histogram of the wavelet detail they are called ***wavelet histogram signatures.*** The number of vector components provided in one feature of the histogram feature is twice of the number of the components in the energy feature because two parameters are obtained from each wavelet detail (See Figure 4.8).

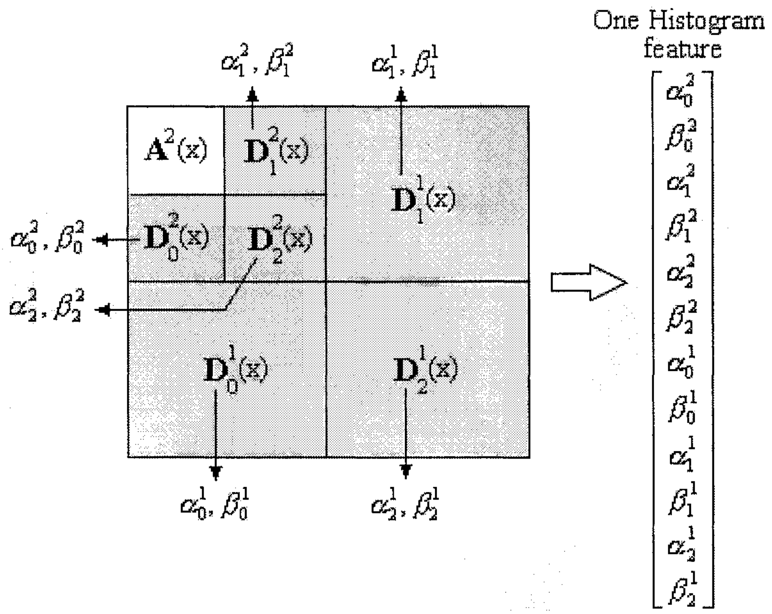


Figure 4.9: One histogram feature obtained from 2-scale 2-DWT image

4.3.2 Eigenvectors feature

A PCA is a linear dimensional reduction procedure, which has been used for several applications such as face recognition, data compression and data dimensional reduction [49,50]. The PCA for face recognition, commonly known as ***eigenface***, has been presented in many articles [49,50,60,61,62]. The properties of PCA are such that its features can be used to identify a person's face so it is conjectured that it could also be applied to identify head patterns within an image. Yuen *et al.* [51] has pointed out some limitations of PCA for face recognition when it is used with a large database and has proposed the combination of multi-resolution using wavelet transform and PCA to resolve this problem. In this application, the combination of WT and PCA is used for extracting features from the image of a person's head. The approximation coefficient obtained from the wavelet transform is used, and then the PCA is applied to obtain features.

Let $Ax = \{Ax_k, k = 1,...,M\} \in R^{dxd}$ be an example of vectors, which are formed by the approximation coefficients row of the transformed image with resolution d x d. Then the PCA is applied to Ax as follows: Firstly, the mean of the Ax vectors are computed as [35]:

$$E(Ax) = \frac{1}{M}\sum_{k=1}^{M} Ax_k \qquad (4.14)$$

Then subtracting the mean vector ($E(Ax)$) from each vector of Ax as:

$$\bar{A}x = Ax_n - E(Ax) \qquad (4.15)$$

The modified version of Ax is $\bar{A}x = \{\bar{A}x_n, n = 1, ...N\}$

Then the covariance matrix of Ax is defined as:

$$Cx = \frac{1}{M}\sum_{k=1}^{M} \bar{A}x_k \bar{A}x_k^{\ T} \qquad (4.16)$$

As ***Cx*** is real and symmetric, finding a set of *n* orthonormal eigenvectors is always possible [35]. This set of eigenvectors or a so-called ***principal component*** corresponds to the characteristic vectors of Ax. So it is used for the features of Ax [4].

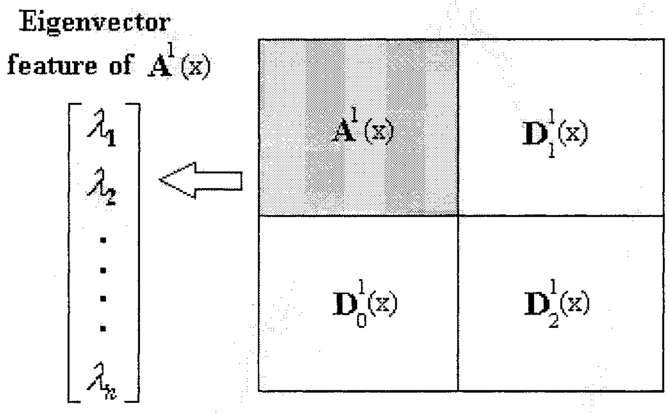


Figure 4.10: One Eigenvectors feature obtained from 1-scale 2-DWT image

4.4 Results

The collected training set is fed to the FEM in order to produce features. The used training set contains 192 head and 192 non-head images with size 32x32 pixels per image as shown in Figures 4.10 and 4.11. The head images were obtained from 32 persons and the non-head images were randomly selected from the whole set of images collected. These were obtained from the object detector. For each image, four types of feature will be provided.

Pre-processing using both 1-scale and 2-scale 2-DWT is applied to the training images to generate one Eigenvector feature (FV_{eig}) and three texture features, respectively. One FV_{eig} contains 16 vector components because the obtained approximation coefficient, $A^1(x)$, size of the 1-scale 2-DWT for an image size of 32x32 pixels is 16x16 pixels so each row of $A^1(x)$ provides one Eigenvalue.

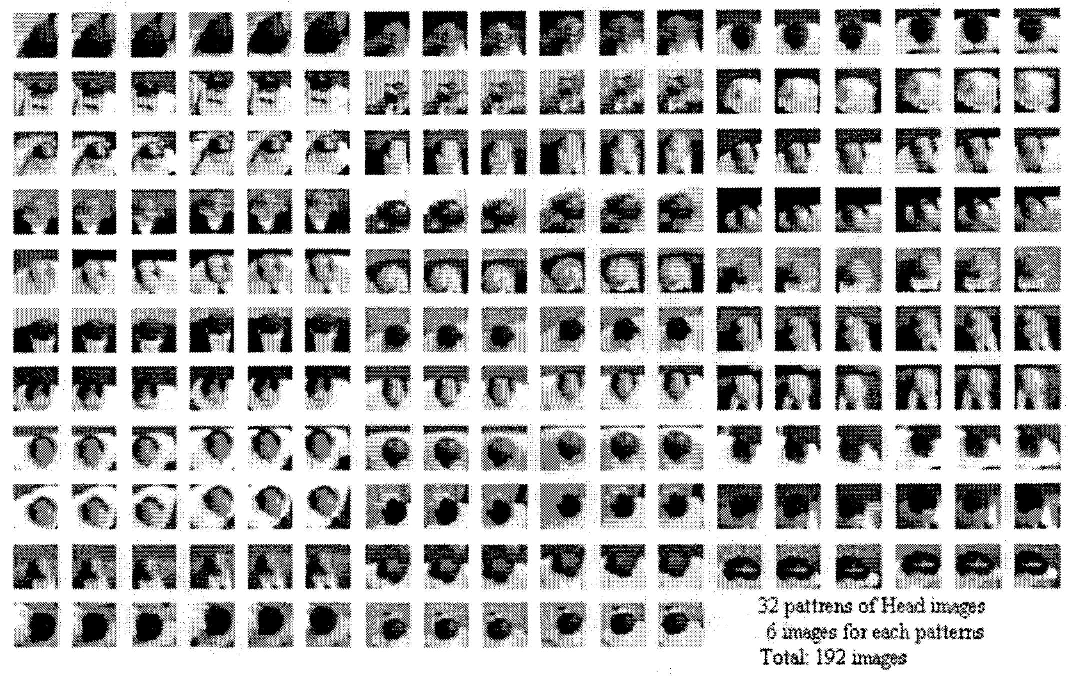


Figure 4.11: Head pattern images

Figure 4.12: Non-head pattern images

For the three texture features, energy (FV_{eng}), feature weighting (FV_{fw}) and histogram (FV_{hist}), 6 vector components are contained in one feature for FV_{eng} and FV_{fw} and 12 vector components for FV_{hist} because 6 wavelet details were provided from the 2-scale 2-DWT image.

These features are plotted to display the distribution of the training data that is to be used in selecting features for classification. Since there are 6, 12 and 16 components in each type of feature, it is difficult to show the resulting feature vectors in a two-dimensional plane. For illustration purposes the feature map between the two chosen components of each feature type that provide the best view are plotted and shown in the Figures 4.13-4.16.

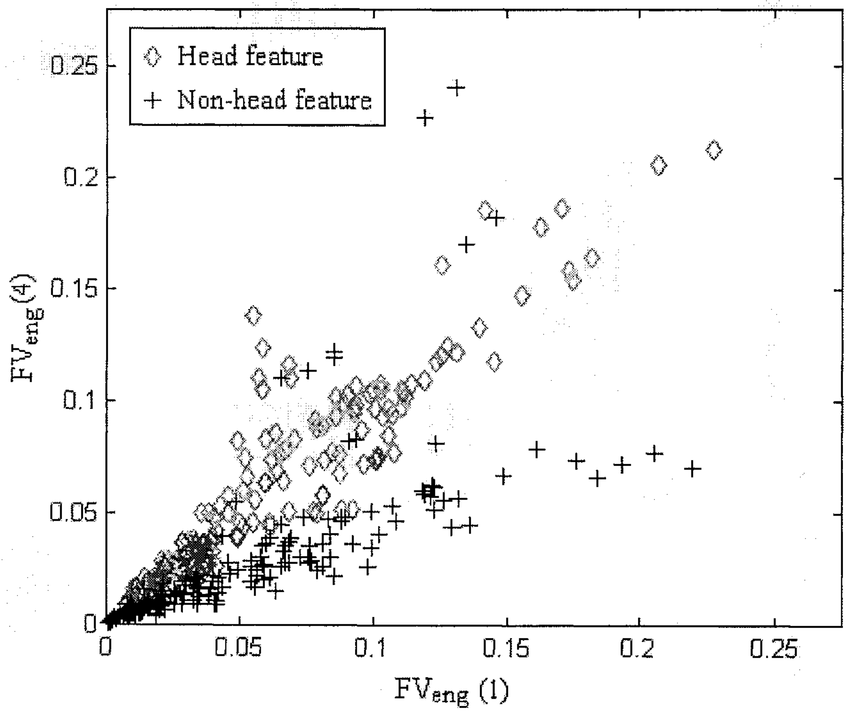


Figure 4.13: Feature Map using $FV_{eng}[1]$ Vs $FV_{eng}[4]$ for Head and Non-head patterns

Firstly, an example plot of the energy feature vectors (components 1 and 4) belonging to two classes, head and non-head feature, is shown in Figure 4.13. It can be seen that the distribution of the data is concentrated in the area with a value of less than 0.05 and formed as an elongated cluster. Also the boundary of two clusters is very close and overlaps. Consequently, it is obvious that clustering errors are unavoidable.

Secondly, the plotting of feature-weighting feature using components 4 and 6 is illustrated in Figure 4.14. The distribution of the data tends to group into two classes with large within-class variance and small between-class distances. Also, there is some overlapping of the two classes in some areas so the clustering error cannot be avoided.

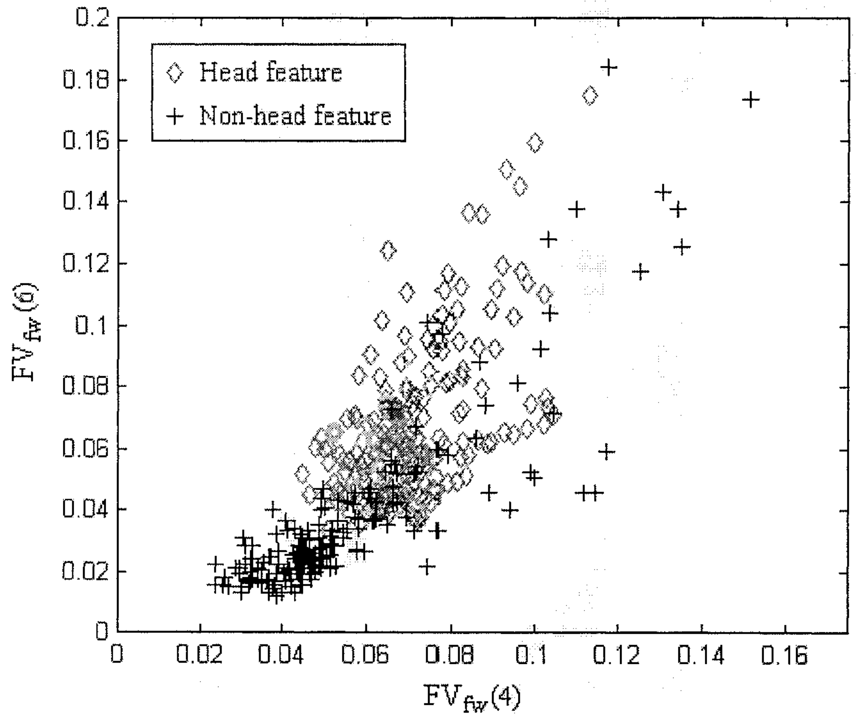


Figure 4.14: Feature Map using $FV_{fw}[4]$ Vs $FV_{fw}[6]$ for Head and Non-head patterns

Thirdly, the plotting of the histogram feature, components 1 and 2, is shown in Figure 4.15. In this case the two groups are dispersed and cannot be distinguished from each other.

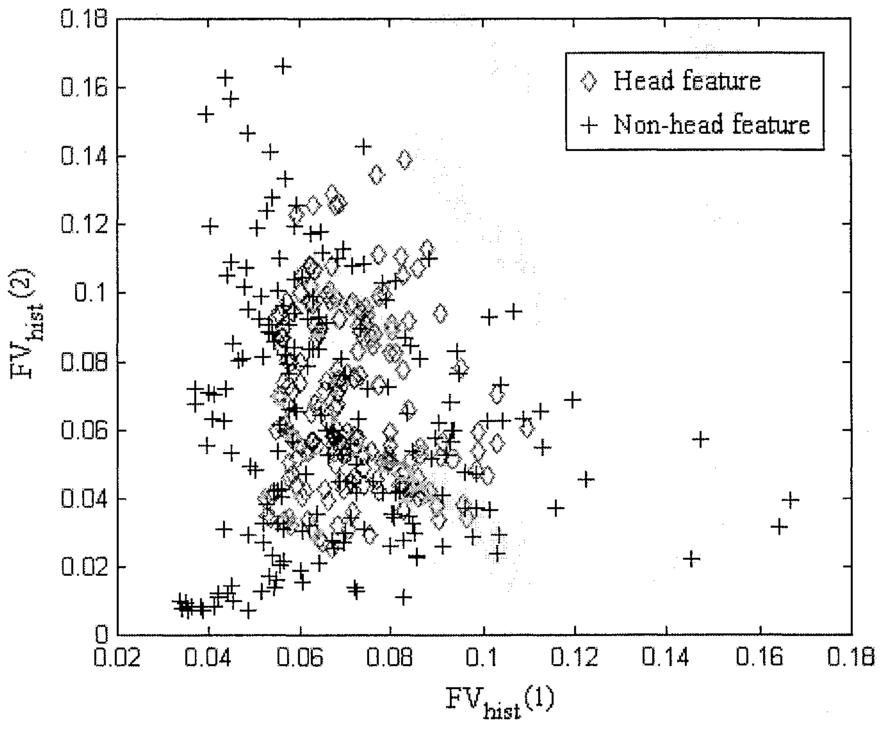


Figure 4.15: Feature Map using $FV_{hist}[1]$ Vs $FV_{hist}[2]$ for Head and Non-head patterns

Finally, the plot of the eigenvector feature using components 1 and 6 is shown in Figure 4.16. In this type of feature, the two groups are compacted and overlapped. Thus it is also not easily separable as in the previous case.

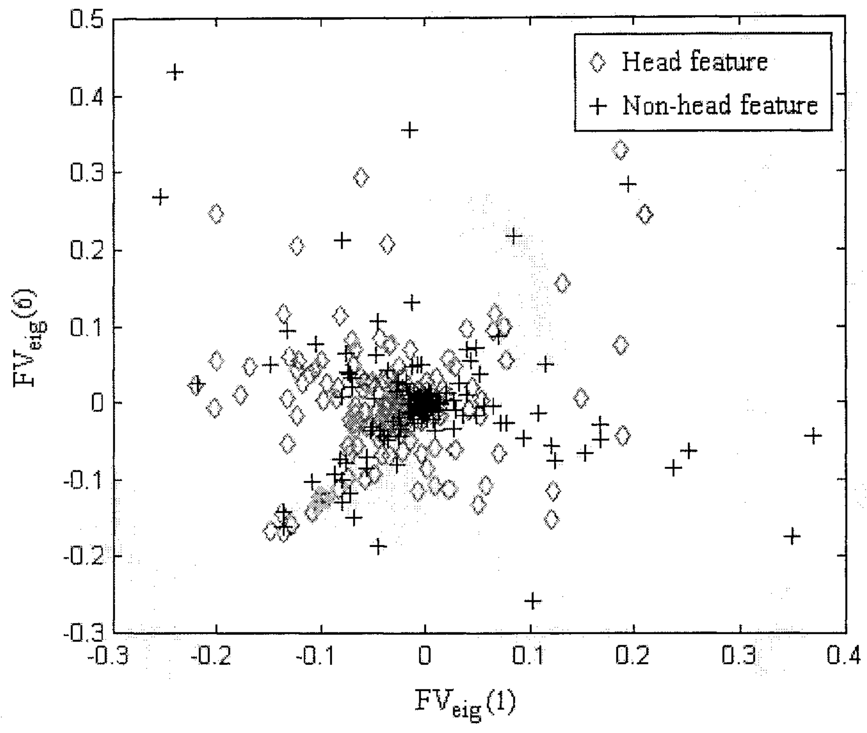


Figure 4.16: Feature Map using $FV_{eig}[1]$ Vs $FV_{eig}[6]$ for Head and Non-head patterns

4.5 Summary and discussion

A number of different techniques have been investigated to identify suitable feature generation methods based on the DWT for the representation of a person's head and non-head images. The results in Section 4.4 found that head and non-head features could be uniquely represented using the energy and feature-weighting features, although there is some intersection areas of two features. This can be seen in Figures 4.12 and 4.13 respectively. For the results of histogram and Eigenvectors features, it can be seen from Figures 4.14 and 4.15 that the head and non-head features are vague and not easily separable. However, there is a method that could be used to deal with these vague features and it will be proposed in the next chapter.

Chapter Five

Classification and Clustering Module (CCM)

5.1 Introduction

The Classification and Clustering Module (CCM) used to classify a person's head features is detailed in this chapter. The technique to be used for CCM is based on an Artificial Neural Network (ANN) called a ***Self-Organising Map (SOM)***. The SOM has been seen to produce good results for feature classification [63,64,65]. The SOM uses unsupervised learning in which there is no pre-specified target output node and no teacher to help it in evaluating an error function that could modify the location of the output node [66]. This makes it suitable for use as a classifier for a person's head features and non-head features that are obtained from the FEM as presented in Chapter 4. The SOM and its application to a person's head pattern classification are now described. The chapter is organised into the following sections: Section 5.2 The Classification and Clustering Module (CCM) using the SOM is described, Section 5.3 how experiments of CCM were carried out on this clustering method and Section 5.4 summarises and discusses the results of CCM.

5.2 The Classification and Clustering Module (CCM)

An overview of the Classification and Clustering Module (CCM) is shown in Figure 5.1. The CCM consists of two SOM blocks, a codebook of Head patterns, a codebook of Non-Head patterns and the Feature measuring block. The two SOM's are individually trained by the head and non-head feature vectors. Once both of the SOM's have been trained they are used to generate the two codebooks. The outputs of the codebooks are passed to the feature measuring stage, where the unknown input feature vector is measured and compared with the obtained vectors from the two codebooks in order to provide the output class of the input feature. The SOM and its algorithm are detailed in Section 5.2.1 and the optimal size of the SOM is discussed in Section 5.2.2. Section 5.2.3 gives an explanation of codebook generation and the feature-measuring block is presented.

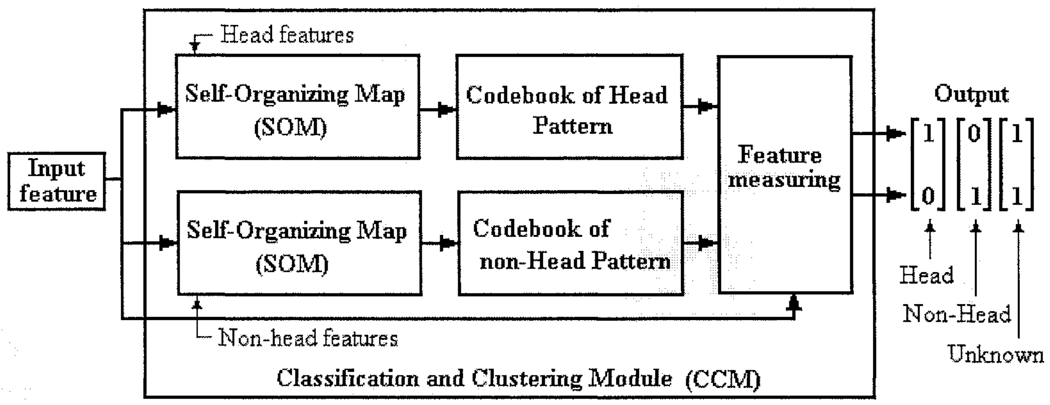

Head features
Self-Organizing Map (SOM)
Codebook of Head Pattern
Input feature
Feature measuring
Self-Organizing Map (SOM)
Codebook of non-Head Pattern
Non-head features
Classification and Clustering Module (CCM)
Output
$\begin{bmatrix}1\\0\end{bmatrix}$ $\begin{bmatrix}0\\1\end{bmatrix}$ $\begin{bmatrix}1\\1\end{bmatrix}$
Head
Non-Head
Unknown

Figure 5.1: An overview of CCM

5.2.1 The Self Organising Map (SOM)

The SOM, introduced by Kohonen [65,67], uses an unsupervised learning process, which learns the distribution of a set of patterns without any class information. A pattern is projected from an input space to a position in the map. The map information is coded as the location of an activated node. The SOM is unlike most classification or clustering techniques in that it provides a topological ordering of the classes. The similarity in input patterns is preserved in the output of the process. The topological preservation of the SOM process makes it especially useful in the classification of data that includes a large number of classes [68]. The structure of a 2-D SOM is shown in Figure 5.2.

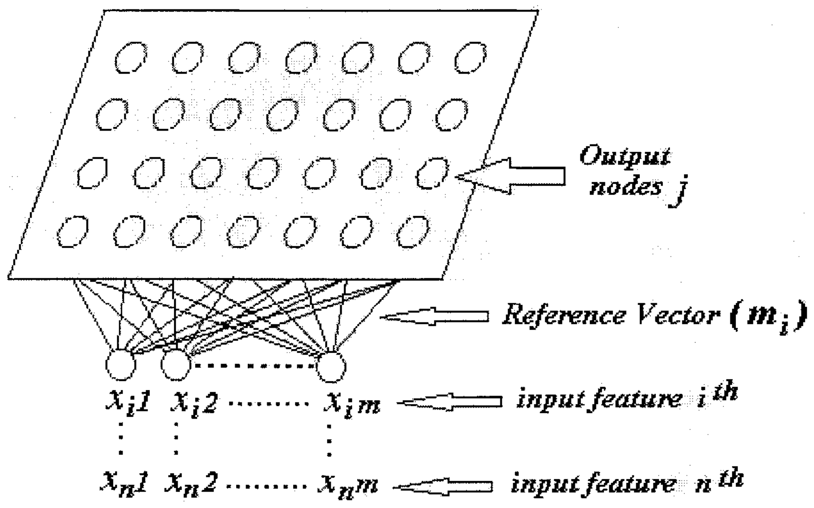


Figure 5.2: Structure of a 2-D Self-Organising Map

Lawrence *et al.* [68] provides a brief description of the SOM algorithm presented in this section, for more detail see [67]. The SOM defines a mapping from an input space $\Re^n$ onto a topologically ordered set of nodes, usually in a lower dimensional space.

An example of a 2-D SOM is shown in Figure 5.3. A reference vector in the input space, $m_i \equiv [\mu_{i1}, \mu_{i2}, ..., \mu_{in}]^T \in \Re^n$, is assigned to each node in the SOM. During training, each input vector, ***x***, is compared to all of the ***m****ᵢ*, obtaining the location of the closest match ***m****c* (given by $|x - m_c| = \min_i \{|x - m_i|\}$ where $|a|$ denotes the norm of vector a). The input point is mapped to this location in the SOM. Nodes in the SOM are updated according to

$$m_i(t+1) = m_i(t) + h_{ci}(t)[x(t) - m_i(t)] \qquad (5.1)$$

where ***t*** is the time during learning and ***h****ci(t)* is the ***neighbourhood function***, a smoothing kernel, which is maximum at ***m****c*. Usually, $h_{ci}(t) = h(|r_c - r_i|, t)$ where r_c and r_i represent the location of the nodes in the SOM output space. r_c is the node with the closest weight vector to the input sample and r_i ranges over all nodes. ***h****ci(t)* approaches 0 as $|r_c - r_i|$ increases and also as ***t*** approaches ∞. A widely applied neighbourhood function is

$$h_{ci} = \alpha(t) \exp\left(-\frac{|r_c - r_i|^2}{2\sigma^2(t)} \right) \qquad (5.2)$$

where $\alpha(t)$ is a scalar value representing the learning rate and $\sigma(t)$ defines the width of the kernel.

They both generally decrease monotonically with time [67]. The use of the neighbourhood function means that nodes, which are topographically close in the SOM structure, are moved toward the input pattern along with the winning nodes. This creates a smoothing effect that leads to a global ordering of the map. It should be noted that $\sigma(t)$ should not be reduced too far, as the map will lose its topographical order if the neighbouring nodes are not updated along with the closest node. The SOM can be considered a non-linear projection of the probability density, $p(x)$ [67].

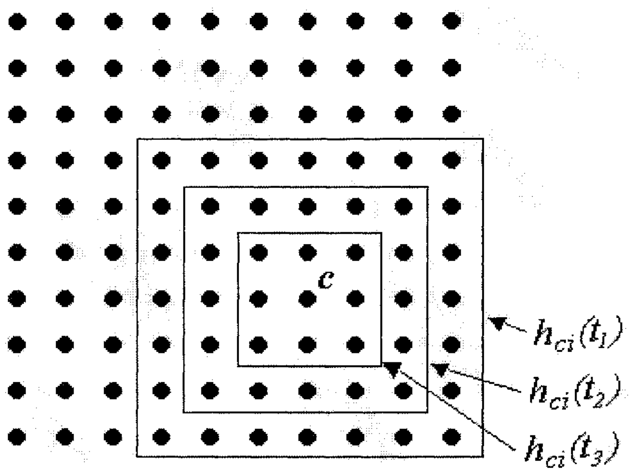


Figure 5.3: A 2-D SOM showing a square neighbourhood function which starts as $h_{ci}(t_1)$ and reduces in size to $h_{ci}(t_3)$ over time.

5.2.2 Optimal size for a Self Organizing Map

It is difficult to determine the number of neurones or the output size of the SOM accurately at the design stage, because it is data dependent [69]. If the output size of the SOM is too large, then the clusters will be formed on the decision surface, which can lead to ambiguous classifications. In addition, if the SOM output size is too small, then the SOM may be unable to classify the input patterns. Consequently, the SOM output size needs to be varied during the experimental phase in order to define the most appropriate output size for the application [69].

There is a method for determining the output size of the SOM that has been investigated and presented by *Jian and Penman* [66]. As the result of their investigation, the relationships between the size of the input patterns and number of output neurones can be defined as [66]:

$$N = \sqrt{m \times 2 - 1} \qquad (5.3)$$

where N defines the number of elements that form an N by N SOM and $\boldsymbol{m}$ the number of identifiably different input patterns, or input vectors to be resolved.

It is found that if N is less than $\sqrt{m \times 2 - 1}$, then the SOM is not be able to learn the input patterns $\boldsymbol{m}$ and map them onto a unique position on the two dimensional output surface. In the other word, the classes representing each of the input patterns will overlap [66].

5.2.3 The Codebook Generation

The purpose of generating the codebook is to create codebook vectors, which represent winning neurons. The codebook vectors are defined from the information obtained by the winning neurons of the SOM. Since the SOM has already been trained, the used training feature is fed through the SOM again as input features in order to define the winning neuron for each input feature. Then the training feature vector that is pointed to by the winning neuron is used as the codebook vector of that neuron. If the neuron represents more than one training feature, these feature vectors are averaged and the average is employed as a codebook vector for that neuron. Any neurons that have never fired or won any of the training features are set to zero. Consequently, the number of feature vectors within the codebook is equal to the number of SOM outputs.

The use of the codebook in this application is to classify an unknown input feature into three classes (head, non-head and unknown class) by using the winning neurons of the two SOM's as the indexes of the two codebooks. The distance between the input vector and the two codebooks, head and non-head, is measured and compared in the feature-measuring block. Then the minimum distance is used to indicate the output class of the input feature as:

$$dH_f = \left\| f_{in} - H_{fcode} \right\| \qquad (5.4)$$

$$dNH_f = \left\| f_{in} - NH_{fcode} \right\| \qquad (5.5)$$

$$if \begin{cases} dH_f < dNH_f,\ f_{in} = [1 \quad 0]^T \ \textit{or Head class} \\ dH_f > dNH_f,\ f_{in} = [0 \quad 1]^T \ \textit{or Non-Head class} \\ dH_f = dNH_f,\ f_{in} = [1 \quad 1]^T \ \textit{or Unknown class} \end{cases} \qquad (5.6)$$

where f_{in} is the input feature, H_{fcode}, NH_{fcode} the obtained head and non-head codebook features, respectively; dH_f and dNH_f are the distances between the input vectors with head and non-head codebook features, respectively. The superscript ($[\ ]^T$) denotes the transpose of the matrix.

Therefore if dH_f is less than dNH_f then the input feature, f_{in}, belongs to the head class. In the opposite case, f_{in} belongs to the non-head class and if dH_f equals dNH_f, f_{in} is set to the unknown class.

5.3 Experimental verification

The four types of features obtained from the FEM are used to train the SOM, in order to cluster the head and the non-head features. Due to some vague outcomes from the FEM, it is impossible to cluster and distinguish head features from non-head features. Moreover, it can be difficult to define the number of SOM outputs if these features are trained with the same training set, since the non-head features are obtained by taking a random selection from the whole data set. Consequently, the head features and non-head features need to be trained separately from the method proposed in the CCM.

In defining the number or size of the SOM output, the head feature is chosen and Eq.5.3 is used to calculate the initial SOM size because the number of head patterns per person is known, 6 patterns per person with a total of 192 patterns from 32 persons. The SOM size starts with 8x8 neurons with a fixed learning rate of 0.9 and 5000 epochs; it is then increased in steps of 2x2 neurons in each direction. The experiments were performed by varying the SOM size between 8x8 and 30x30 neurons. The SOM outputs that won or fired with the same input features, while the number of SOM dimensions is increased, can be used as the criterion to decide if the SOM dimensions are sufficiently large. In addition if the SOM dimensions continue to increase, it will result in some redundancy, producing no different classification results even though the wining output is different for each input feature. According to the experiments, it was found that a suitable SOM for each type of head feature is shown in Table 5.1. The obtained SOM sizes of head pattern are then employed for setting the SOM size for non-head patterns. The proposed technique is applied in Section 5.2.3 the head and non-head codebooks are produced and the proposed FEM and CCM are evaluated.

Table 5.1: Illustrates the used SOM dimension for each type of feature.

Feature types	SOM size
Energy feature	10x10 neurons
Feature-weighting	12x12 neurons
Histogram feature	10x10 neurons
Eigenvector feature	16x16 neurons

In order to evaluate the proposed FEM and CCM, 345 test-images of head patterns, which were captured from several frames of sequential images as displayed in Figure 5.4, are employed. For the test-images of non-head patterns, 300 images obtained from the collected image data as shown in Figure 5.5 are used and the results of the test are shown in Table 5.2.

Table 5.2: Show the results of the classification performed on the test-images.

Feature types	Test-image	Correctly classify (%)	Incorrectly classify (%)	Unable to classify (unknown) (%)
Energy feature	Head	37.10	62.9	-
	Non-Head	88.67	24.67	-
Feature-weighting feature	Head	40.87	59.13	-
	Non-Head	79.33	20.67	-
Histogram feature	Head	58.55	37.39	4.06
	Non-Head	59.33	36	4.67
Eigenvector feature	Head	36.23	63.77	-
	Non-Head	83.00	17.00	-

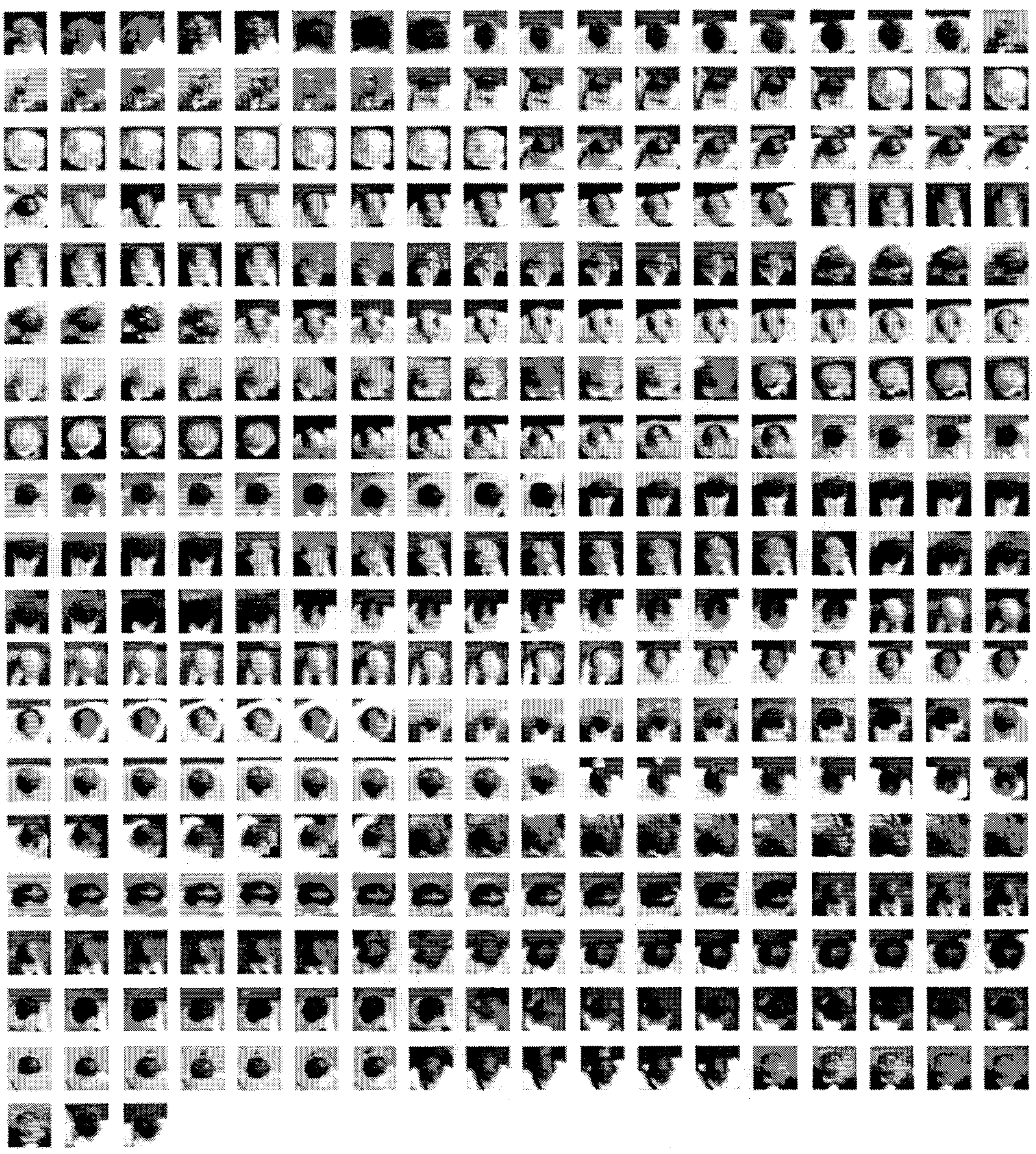


Figure 5.4: Illustrates 345 test-images of head pattern, which were captured from different frames of the sequential images.

Figure 5.5: Illustrates 300 test-images of non-head pattern, which were obtained from the image data.

5.4 Summary and discussion

By observing the results from the classification, it can be seen that the histogram feature provides the best result for classifying the head images out of the four feature types and the poorest non-head images classifications are also given from the histogram feature as well. Consequently, it can be concluded that the texture and the Eigenvector features are not suitable to be employed as features for this application although portions of correct results are provided. Though the outcomes of the proposed technique for classification are unsatisfactory, it is premature to conclude that the idea of applying classification to detect pedestrians does not work. The entire system of a pedestrian detection based classification technique is demonstrated in the next chapter.

Chapter Six

Experiments

6.1 Introduction

In this chapter the classification technique to be used in the pedestrian motion tracking system is proposed and investigated by experimentation, to offer an evaluation of the performance of the proposed combination of sub-systems. In Section 6.2 a block diagram of the classification system is presented and explained. Then experiments that test the ability of the classification system to extract pedestrian head features from sequential images are performed and analysed in section 6.3. Finally, section 6.4 offers further discussion of the results obtained and a summary.

6.2 The pedestrian detection based classification technique (PDC)

The PDC as shown in Figure 6.1 consists of four blocks as follows: 1) the object detector block, 2) the sub-image capture block, 3) the classifier block and 4) the marking block. The object detector block uses the technique presented in Chapter 3 to provide initial locations of candidate objects for use in capturing sub-images.

Then a similar technique to that used in the data collection in Section 3.5 is applied in the sub-image capture block, where the obtained candidate locations are used to capture the sub-images with a size of 32x32 pixels each. These are then passed to the classifier block, which employs FEM and CCM, as presented in Chapter 4 and 5. In the classifier block, the sub-images are categorised into classes as head, non-head and unknown classes. The location of the sub-image classified as head images are then passed to the marking block in which they are used to place crosses on the input image corresponding their location in the image, as shown in Figure 6.1.

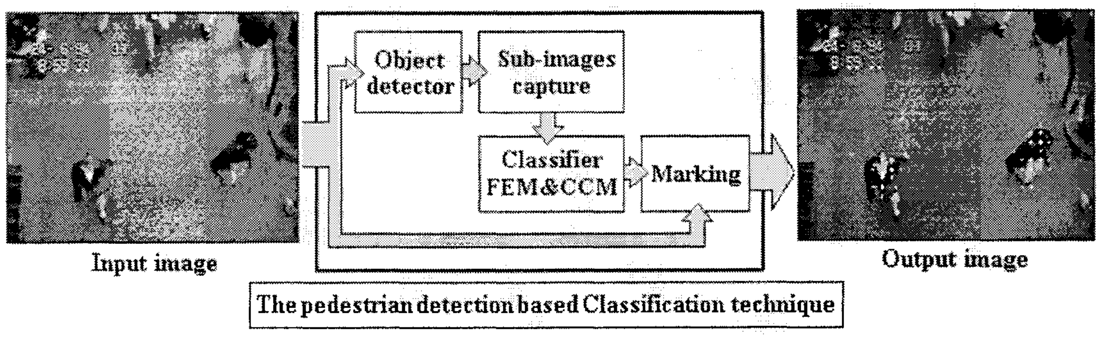


Figure 6.1: Shows the entire system of pedestrian detection based classification technique.

6.3 Experiments

The same sequential pedestrian images and the detection process used in Chapter 3 are employed in the testing of the classifier system. Each of the four types of feature extraction and the clustering method as discussed in Chapters 4 and 5 is applied, to allow a comparison of their performance to be seen. For each image 4 output images will be obtained. Some examples of the experimental results are shown in Figures 6.2 - 6.5.

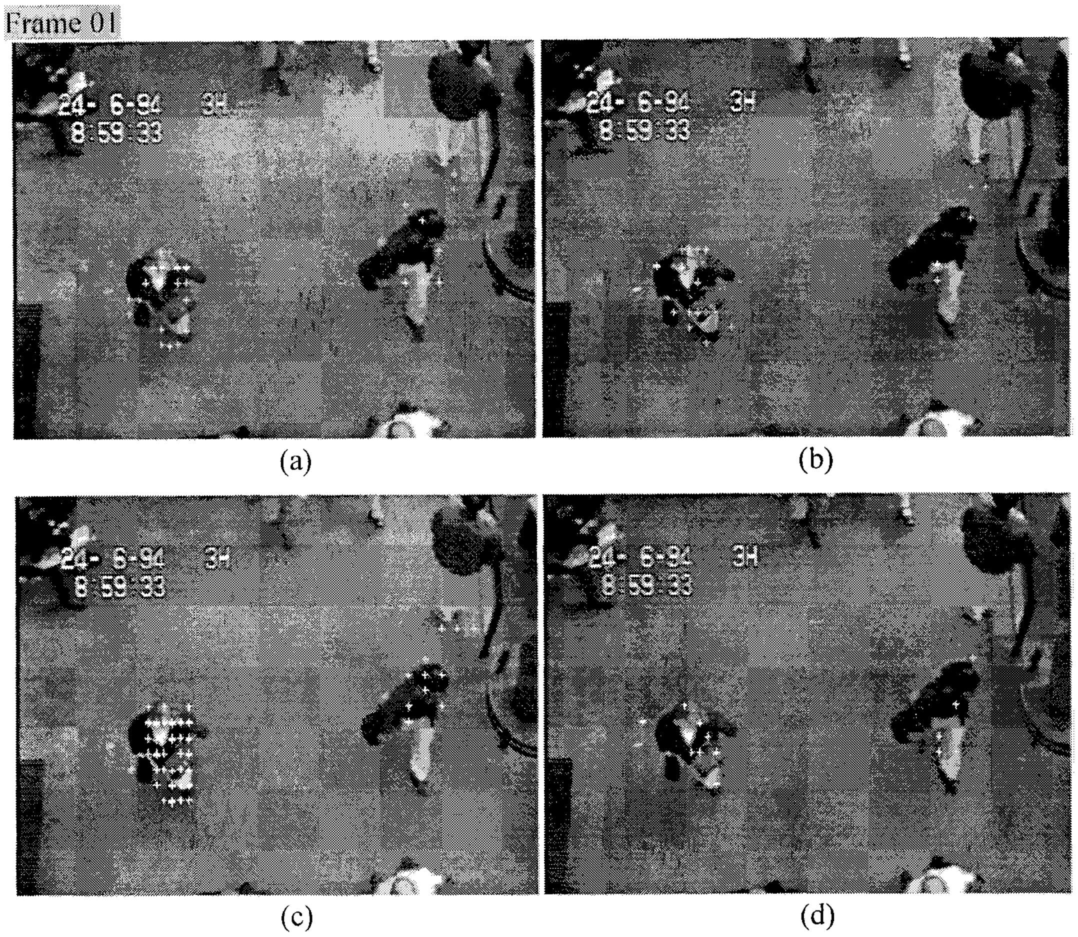


Figure 6.2: The PDC results using 4 different features as (a) Energy feature, (b) Feature weighting, (c) Histogram feature and (d) Eigenvector feature

The PDC using 4 features is applied to an example image (Frame 01) containing multiple pedestrians as illustrated in Figure 6.2. Hence the same detection process is used so it can be assumed that the results represent the efficiency of the pre-processing and the classification methods. Each cross-sign is marked on the area that represents a head feature. The applied method works much as expected as they should have many cross-signs marked on the objects due to the error from the clustering process as discussed in Chapter 5. Each type of feature provides a different number of cross-signs, the histogram feature (Figure 6.2 (c)) provides the most cross-signs and the eigenvector feature (Figure 6.2 (d)) the least cross-signs. Although there are many cross-signs of errors occurred, it can be seen that the PDC is able to locate the head of pedestrians to some degree. Another example from later frames (Frame 02) is shown in Figure 6.3.

Frame 02

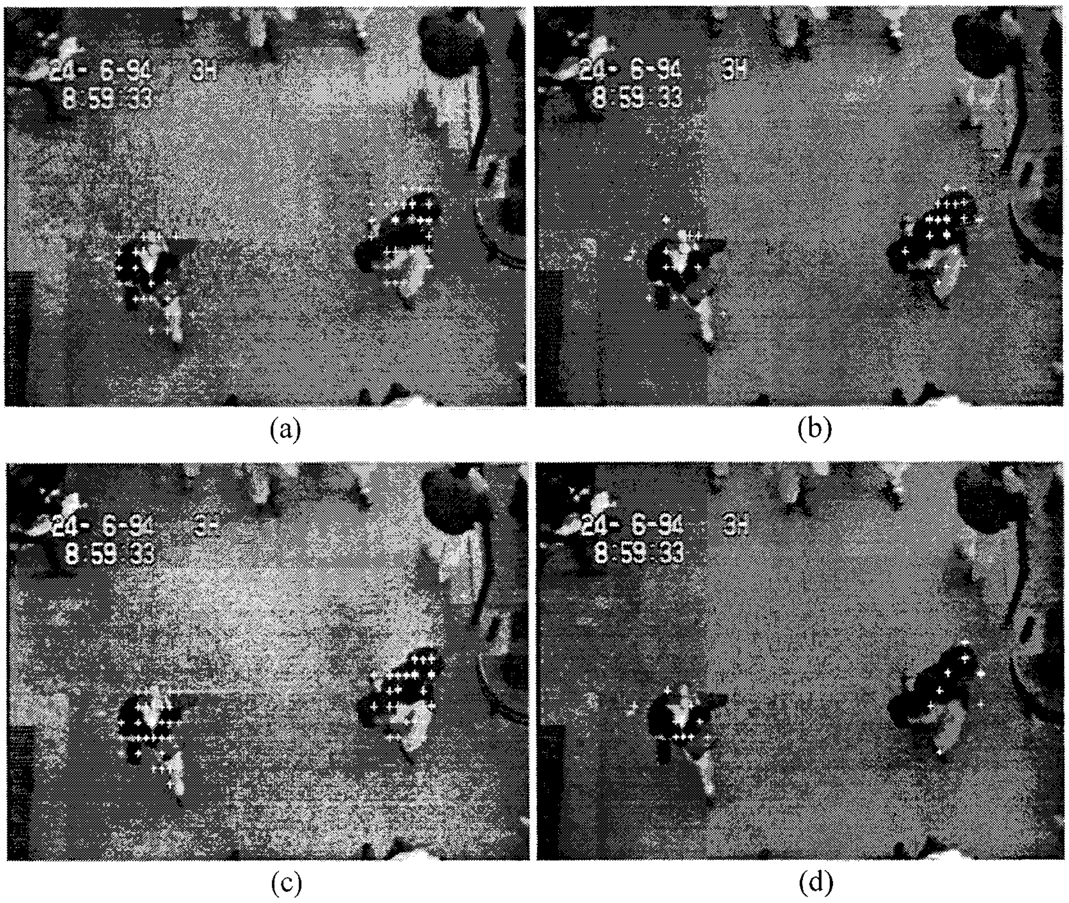


Figure 6.3: The PDC results of an image Frame 02: (a) Energy feature, (b) Feature weighting, (c) Histogram feature and (d) Eigenvector feature

The performance of PDC is applied to the image that contains more pedestrians as illustrated in Figures 6.4. In this case, it is obvious that the initial location provider obtained the candidate locations and can detect 4 persons within the image. As a result, the PDC fails to detect the pedestrian's head that can be seen in Figure 6.4 (b) and (c).

Also errors increased as the number of pedestrians increases because the distance between pedestrians is reduced so which may result in gaps being determined as head features as seen in Figure 6.4 (a) and (c) in the gap between two ladies. Another example (Frame 02) is shown in Figure 6.5.

Frame 01

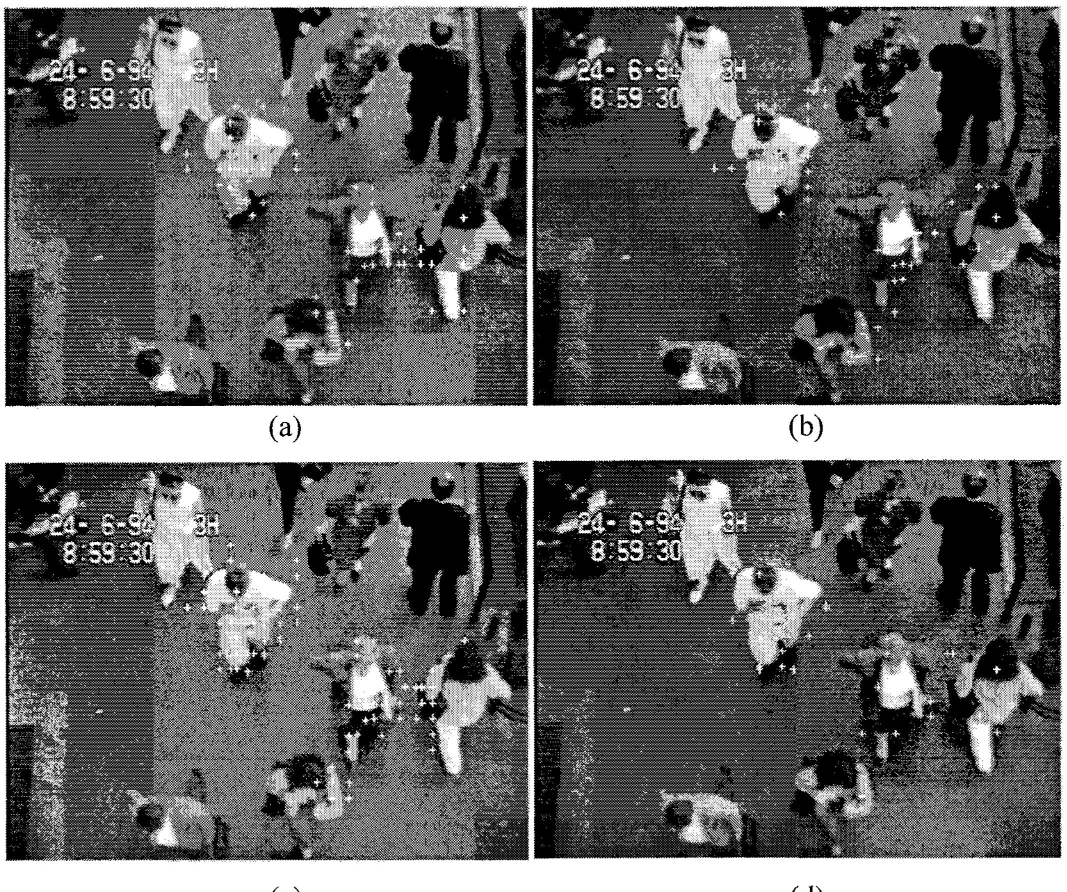


Figure 6.4: The PDC results of the image that contains more pedestrians: (a) Energy feature, (b) Feature weighting, (c) Histogram feature and (d) Eigenvector feature

Frame 02

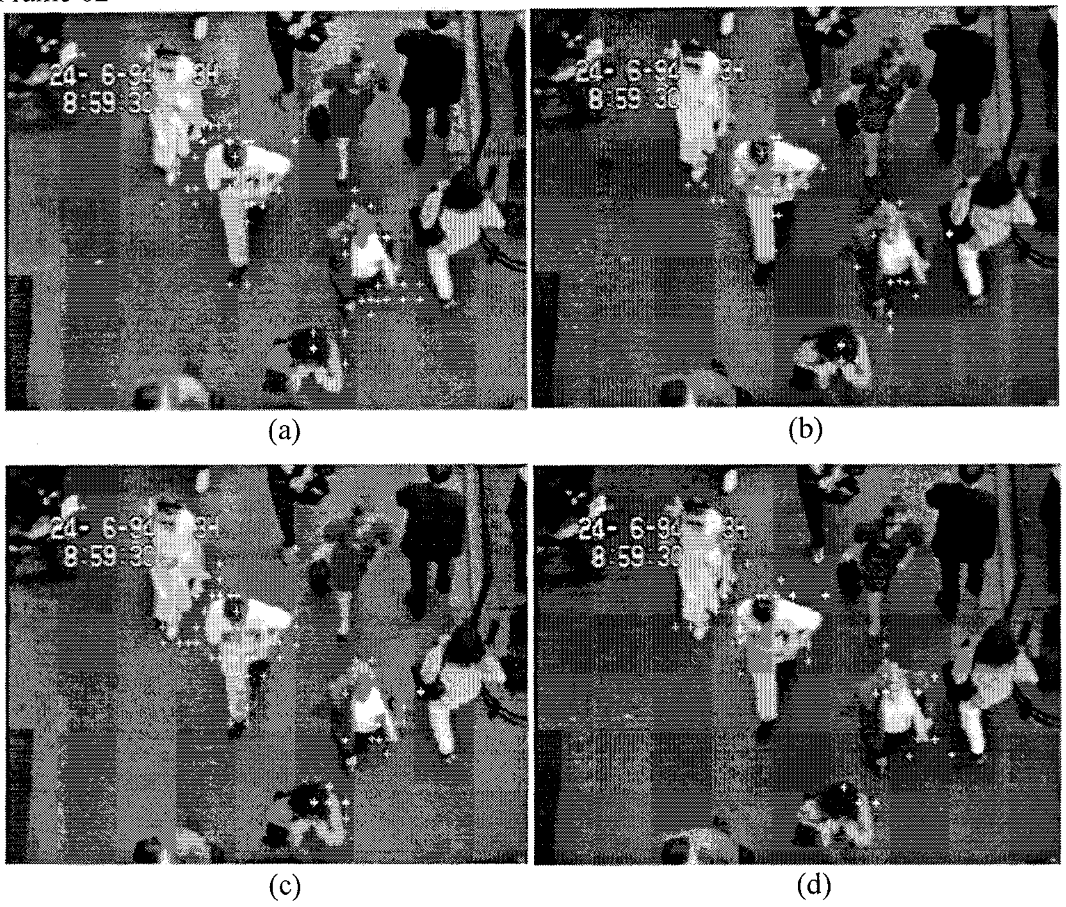


Figure 6.5: The PDC results of the image Frame 02: (a) Energy feature, (b) Feature weighting, (c) Histogram feature and (d) Eigenvector feature

According to the experimental results, it can be seen that the texture is unsuitable to use as feature extraction for pedestrian tracking particularly when applied to grayscale images. Although the PDC works to some degree it is not accurate enough for use in real applications.

6.4 Discussion

From the results of running the PDC developed with the sequential images, it can be concluded that the PDC is able to detect and locate a pedestrian's head, although many errors are also obtained. There are two main error sources, firstly, the techniques used to provide the initial candidate locations are imprecise, and as a result means that the performance of the rest of the system blocks especially the sub-image capture block is severely impeded. In the sub-image capture block the scan size becomes larger than necessary leading to the formation of redundant sub-images. The FEM&CCM block performs to the same degree of accuracy as presented in Chapter 5. Techniques are proposed and investigated in Chapter 7 aimed at improving the system performance.

Chapter Seven

The association of Cellular Automata and Neural Network

7.1 Introduction

In this chapter an introduction to both Cellular Automata (CA) and Neural Networks (NN) is provided. This is followed by a discussion on how the two can be merged together to form a predictor. The prediction module proposed in the next chapter is based on this association and is used to estimate a pedestrian's location in the following frames, and hence reduce the search space for objects within the image frame.

The chapter is organized as follows: In Section 7.2, the theory and applications of Cellular Automata (CA), as used for capturing human movement behaviour is explained in detail. In Section 7.3, the theory and applications of the NN, as used to classify CA patterns, is discussed. In Section 7.4, the conjecture of association of CA and NN for the prediction module is proposed. Simulations are performed to provide support to the fusing of CA and NN. A discussion and summary of the work presented in this chapter is given in Section 7.5.

7.2 Cellular Automata (CA)

Cellular Automata (CA) are mathematical idealizations of physical systems in which space and time are discrete, and physical quantities take on a finite set of discrete values. A cellular automaton consists of a regular uniform lattice or array, usually infinite in extent with a discrete variable at each site or cell. The state of a cellular automaton is completely specified by the values of the variables at each site. A cellular automaton evolves in discrete time steps, with the value of the variable at one site being affected by the values of variables at sites in its "neighbourhood" on the previous time step. The neighbourhood of a site is typically taken to be the site itself and all immediately adjacent sites. The variables at each site are updated simultaneously, based on the values of the variables in their neighbourhood at the preceding time step, and according to a definite set of "local rules"[70].

In the early 1950's, CA were first introduced by Von Neumann. At this time the Universal Turing Machine [71] was the only computing machine available to run this CA and so was limited in capacity. Since then, many other studies on CA as abstract models of computation have been made and covered a large scope of applications including biology, physics, operational research, sociology and computer science [72]. In recent years, CA microsimulation has emerged as an effective technique for modelling complex behaviours such as the modelling of artificial life. CA microsimulation has also been successfully applied to the modelling of vehicular flows and traffic networks as presented by Nagel and Schadschneider in [73,74] respectively.

They have been proven to provide a good approximation of complex traffic flow patterns, over a range of densities. While the field of vehicular flow modelling using the CA is well established, the tasks on modelling pedestrian flow are becoming increasingly interesting to researchers. Blue and Adler [75-79] applied the CA microsimulation to model pedestrian flows that are uni, bi-, cross-, and four-directional. Fukui and Ishibashi [80,81] have also proposed the CA model to mimic pedestrian flow in order to analyse pedestrian behaviour. As the CA can be used to mimic the pedestrian movement behaviour, it is conjectured that they could be adopted as a tool for estimating the pedestrian locations in future frames. Therefore, the principle of the CA is studied and sought for adopting as a prediction tool. Two fundamental CA models are discussed one and two-dimensional.

7.2.1 One-dimensional Cellular Automaton

The simplest model is a one-dimensional cellular automaton (1D-CA) used for simple physical and biological models. It consists of a line of elements in which each site carries values that are updated in parallel. The neighbourhood of a site is typically taken to be the two adjacent sites. This characteristic is known as a spatially local rule. The relationship between sites is depicted in Figure 7.1.

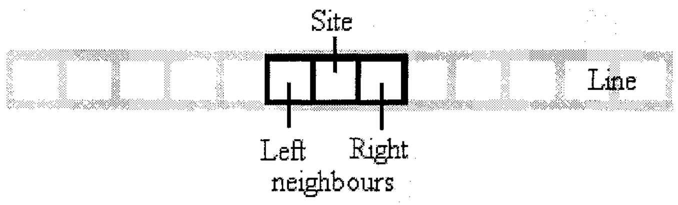


Figure 7.1: A cell and its neighbours of 1D-CA

The evaluation of the sites is controlled by a definite set of rules, depending on the values for a fixed number of preceding steps. Usually, the preceding step is 1. This is the temporally local rule. Such very simple local evaluation rules can produce very complex even chaotic behaviour for the overall system. An example of a two-state (0,1) 1D-CA simulation is shown in Figure 7.2. The rules used for the CA in the simulation are as expressed below:

$$\textit{New state} = \begin{cases} 0 \ \textit{if} \ x = [0 \ \ 0 \ \ 0] \ \textit{or} \ [1 \ \ 1 \ \ 1] \\ 1 \ \textit{if} \ x = [1 \ \ 1 \ \ 0], [1 \ \ 0 \ \ 0], \\ \qquad [1 \ \ 0 \ \ 1], [0 \ \ 1 \ \ 0], \\ \qquad [0 \ \ 1 \ \ 1], [0 \ \ 0 \ \ 1] \end{cases} \qquad (7.2.1)$$

Where x is a set of three cells

These rules can also be described pictorially or as the binary calculation shown in Figure 7.2.

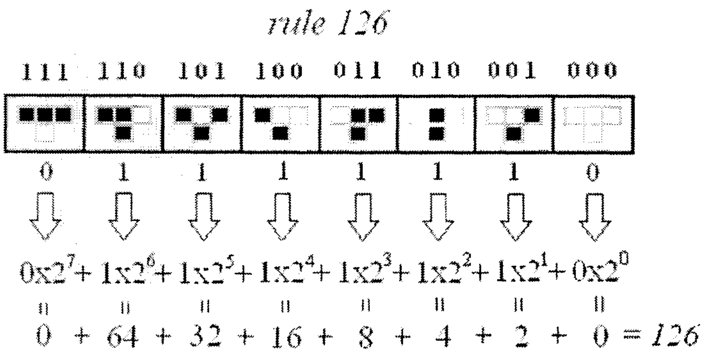


Figure 7.2: CA rule ***126***

As time increases, the cells change from state to state in parallel using the CA rules in (7.2.1). The pattern produced by applying this CA over 20 steps to a given input, is illustrated in Figures 7.3 and 7.4. The results show that a pattern is repeatedly reproduced as the number of steps is increased. By applying a different set of rules to the CA with the same initial state, a different result is obtained two examples are shown in Figure 7.5. More CA simulation patterns using different rule sets can be found in [70].

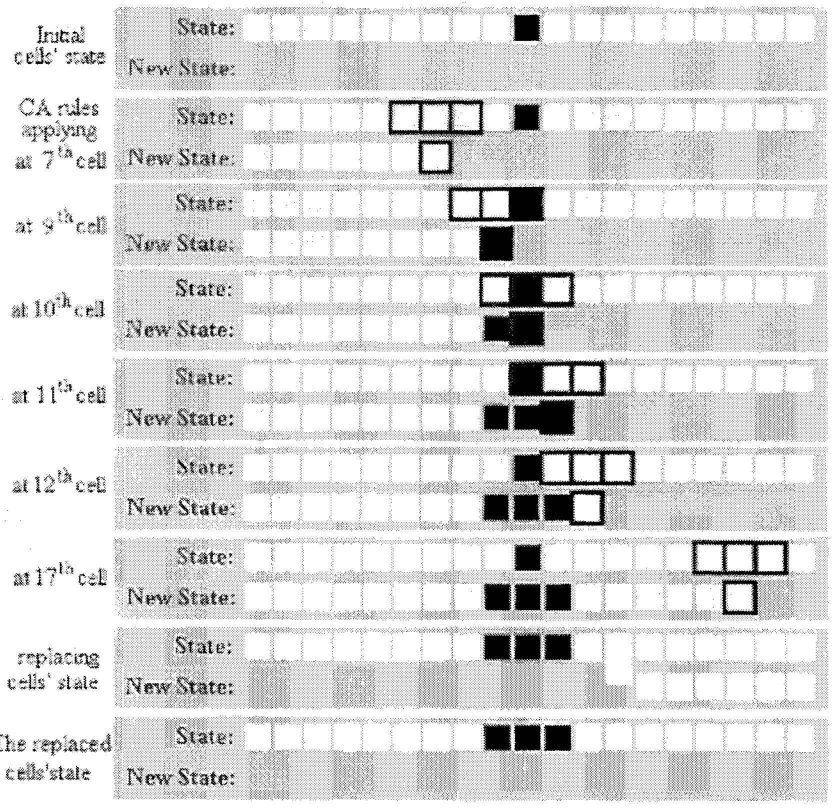


Figure 7.3: The result of applying CA rules to two-state 1D-CA

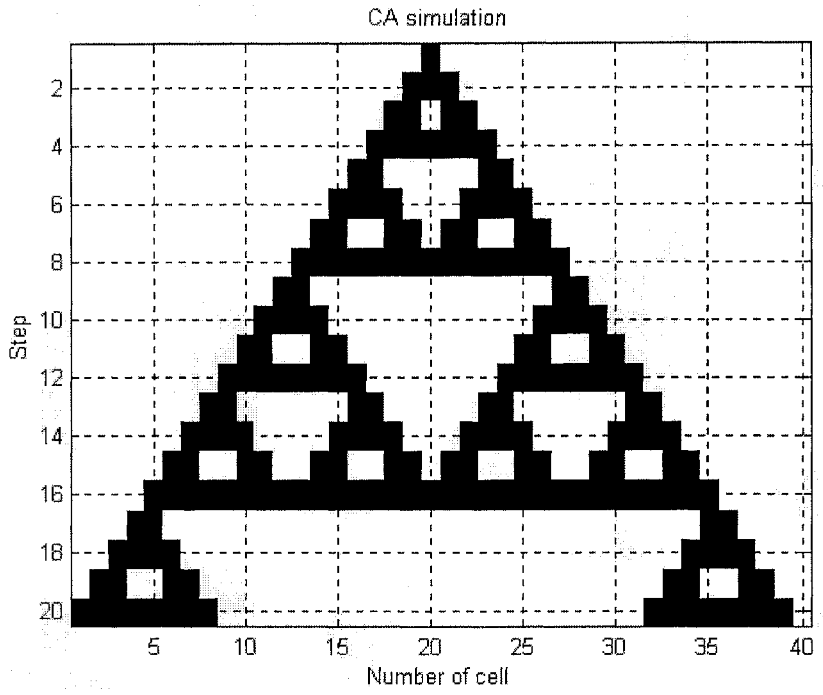


Figure 7.4: The simulation of 20 steps of two-state one-dimensional CA

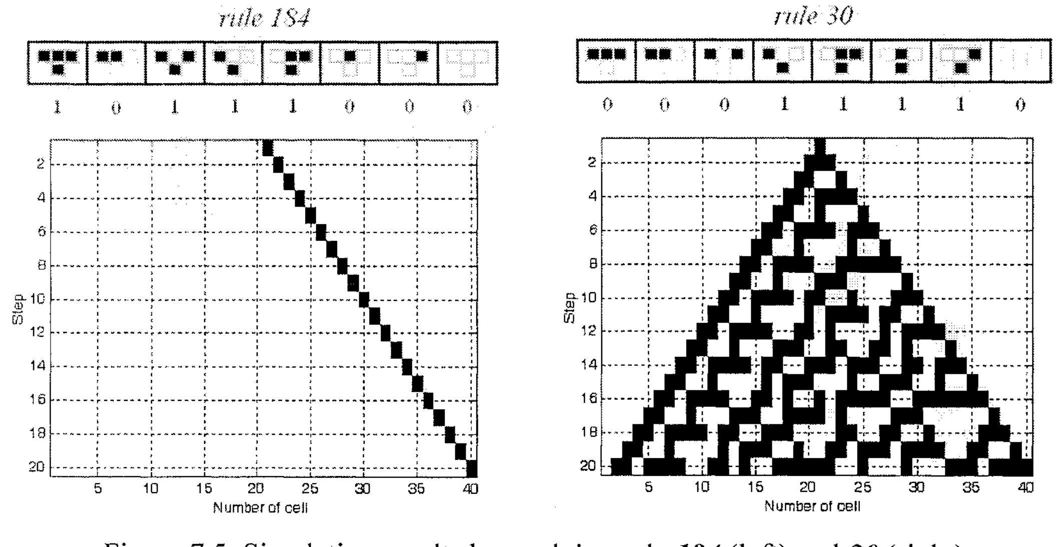


Figure 7.5: Simulation results by applying rule ***184*** (left) and ***30*** (right)

7.2.2 Two-dimensional Cellular Automata

The basic principle of the two-dimensional automata is identical to that of the one-dimensional cellular automata (discrete space, time and state). The main differences reside in the neighbourhood. There are several possible lattices and neighbourhood structures for two-dimensional cellular automata [82]. The first one, defined by Von Neumann [83], is the square cellular automaton, as illustrated in Figure 7.6 *a*. A second important two-dimensional cellular automaton is known as the Moore neighbourhood cellular automaton as depicted in Figure7.6 *b*.

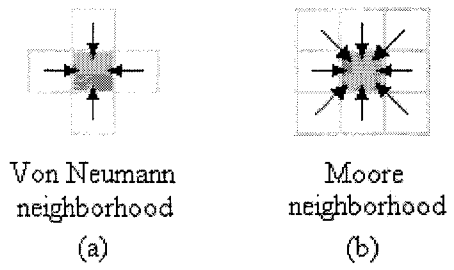


Figure 7.6: Two-dimensional cellular automata

Each cell is only connected to the four or eight neighbours around it respectively. Any functions on the chosen neighbourhood can then be evaluated. One of the simplest computations allowed by such an arrangement is a sum of the cell values and has been widely described in the cellular automaton theory [70,82-85].

As the CA model uses empirical data for defining the CA rule in order to run the simulation, there will be a number of patterns of CA neighbourhood that can be used for the model. However, if the model has an automated method for defining the CA rule set without using empirical data, it would be useful for CA rule selecting. For this reason, the model requires a technique that can learn and classify the CA patterns.

With its potential to learn from examples, the application of a NN to classify CA patterns is described in the next section.

7.3 Neural Network (NN)

A Neural Network (NN) is a computational structure inspired by the study of biological neural network processing [85]. Although there are many types of neural network, each NN algorithm is adopted for different purposes. In the presented work the focus will be on a backpropagation neural network (BPNN) because it is suitable for the required application of CA pattern classification. Details of the BPNN is now discussed.

The BPNN [86] is a feed-forward Multilayer Perceptron (MLP) consisting of (1) an input layer with nodes representing the input variables to the problem, (2) an output layer with nodes representing the dependent variables (i.e. what is being modelled), and (3) one or more hidden layers containing nodes to help capture the non-linearity in the data. Using supervised learning with the error-correction learning rule (ECL), these networks can learn the mapping from one data space to another using examples. The term backpropagation refers to the way the error computed at the output side is propagated backward from the output layer, to the hidden layer, and finally to the input layer. In a BPNN, the data is fed forward into the network without feedback. The neurons in a BPNN can be fully or partially interconnected. The architecture of a BPNN is illustrated in Figure 7.7. These networks are highly versatile and can be used for data modelling, classification, forecasting, control, data and image compression, and pattern recognition.

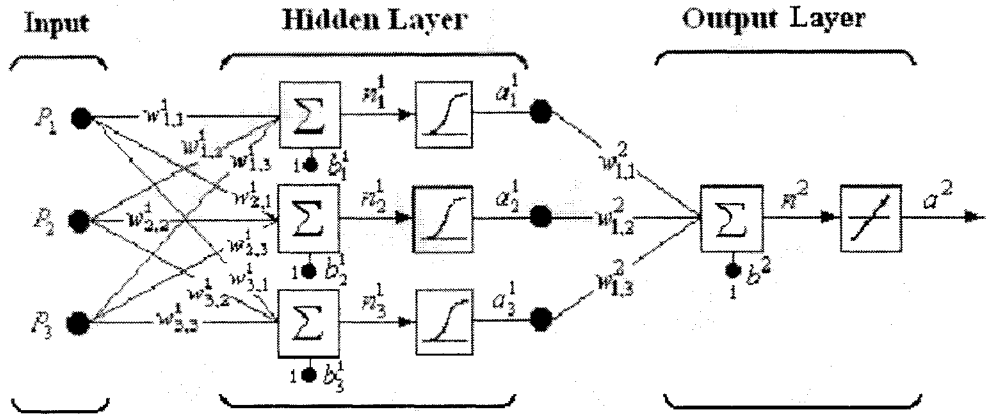


Figure 7.7: Two-layer BPNN architecture with 3 inputs, 1 output and 3 hidden neurons

The feedforward error-backpropagation (BP) learning algorithm is the most famous procedure for training NNs. BP is based on searching an error surface (error as a function of NN weights) using gradient descent for point(s) with minimum error. Each iteration or epoch in a BPNN constitutes two sweeps: forward activation to produce a solution, and a backward propagation of the computed error to modify the weights. In an initialised BPNN, the forward sweep involves presenting the network with one training set. This starts at the input layer where each input node transmits the value received forward to each hidden node in the hidden layer. The collective effect on each of the hidden nodes is summed up by performing the dot product of all values of input nodes and their corresponding interconnection weights, as described in Eq.(7.2.2).

$$y = \begin{cases} 1, & \text{if } \sum_{i=1}^{n} w_i x_i \geq b, \\ 0, & \text{if } \sum_{i=1}^{n} w_i x_i < b, \end{cases} \tag{7.2.2}$$

Where $\boldsymbol{y}$ is the output node, $\boldsymbol{x}$ is the input node and $\boldsymbol{w}$ is the interconnection weight.

Once the net effect at that node is calculated using a transfer function to yield an output between 0 and +1 or −1 and +1. Various transfer functions are described in [87]. The amount of activation obtained represents the new signal that is to be transferred forward to the subsequent layers that are hidden and output layers. The same procedure of calculating the net effect is repeated for each hidden node and for all hidden layers. The net effect calculated at the output nodes are consequently transformed into activations using a transfer function. The activations calculated at the output nodes represent the BPNN solution of the training set, which may deviate considerably from the target solution due to the arbitrary selected interconnection weights. In the backward sweep, the different or error between the BPNN and target outputs is employed to adjust the interconnection weights, starting from the output layer, through all hidden layers, to the input layer. The forward and backward sweeps are performed repeatedly until the BPNN solution agrees with the target value within a pre-specified tolerance. The error-backpropagation learning algorithm provides the needed weight adjustments in backward sweep.

7.4 Association of the CA and BPNN

The basic principles behind the CA and the BPNN have been discussed in the previous sections. This section discusses the manner in which the two are associated and used as a prediction technique. Experiments of the association are conducted and demonstrated.

According to the principle of the CA, the evaluation of each site (in 1D-CA and 2D-CA) is controlled by a set of CA rules that can be regarded as patterns in terms of vectors. These patterns could be learnt by a BPNN. Therefore, it is conjectured that after appropriately training the BPNN, it will provide the same result as the CA. If the conjecture is true, the capturing of pedestrians behaviour from the real image is feasible and the need to determine the rules for the CA from empirical data can be avoided.

In order to support the conjecture, a simulation of the BPNN is adopted to perform the same task as the 1D-CA example in Section 7.2.1 the results are provided in the following section.

7.4.1 1D-CA experiments

In this section an experiment is undertaken to show the ability of a BPNN to learn the patterns of a 1D-CA and then operate in the same way.

The supervised learning process involves the training of the BPNN, target outputs are provided to the BPNN for all possible input patterns. The BPNN then uses the deviation (error) of the BPNN solution from corresponding target value to determine the required amount by which each node weight should be adjusted. The inputs and outputs are determined by consideration of the CA patterns.

In the first experiment, the CA patterns are determined as inputs and the new stage value is employed as the output target for BPNN. From the 1D-CA example in Section 7.2.1, the pattern of the 3 inputs and one target output can be represented in terms of vectors. The total number of patterns is determined by the number of input cells and the number of values each cell can take, in this example, each pattern has 3 cells and each cell can have one of two possible values (0,1), therefore the possible number of patterns is $2^3 = 8$ patterns with 8 output targets as expressed in (7.2.3).

$$\begin{aligned} \textit{input patterns} &= \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \\ \textit{output target} &= \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{bmatrix} \end{aligned} \qquad (7.2.3)$$

In this experiment, a two-layer BPNN, with log-sigmoid transfer function in the hidden layer and a linear transfer function in the output layer (Figure7.8) is employed. As an initial guess, three neurons in the hidden layer are used, however, the number of neurons in this layer is increased if the training performance does not agree with the target value within the prespecified tolerance. The network has an output neuron since there is one target for each pattern.

After a training session consisting of 10,000 iterations, the validation error did not reach the target error (Figure7.9); therefore the number of neurons in the hidden layer is increased. The experiment is performed 8 times, each time increasing the number of neurons by 1, to show the training capability of the BPNN. The results are shown in Table 7.1.

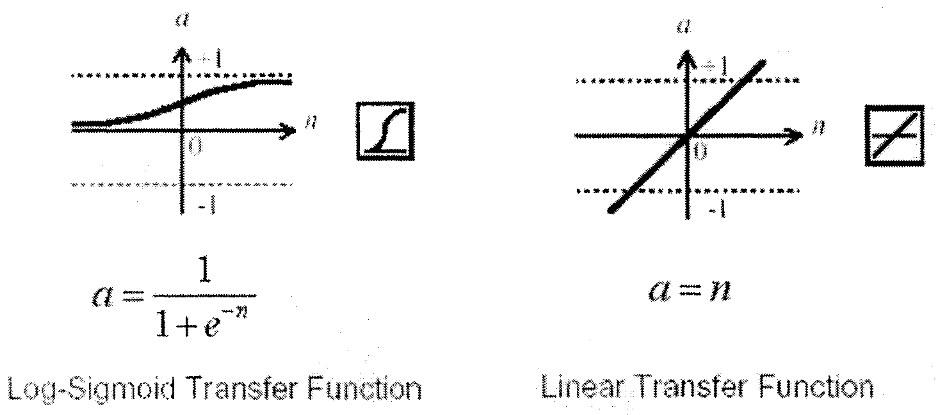


Figure7.8: Log-sigmoid and linear transfer functions [87]

Table 7.1: Training results

Number of hidden neurons	Training results of 10 times		Percent of successful training (%)
	Successful	Unsuccessful	
3	1	9	10
4	8	2	80
5	8	2	80
6	9	1	90
7	10	0	100
8	10	0	100
9	10	0	100
10	10	0	100

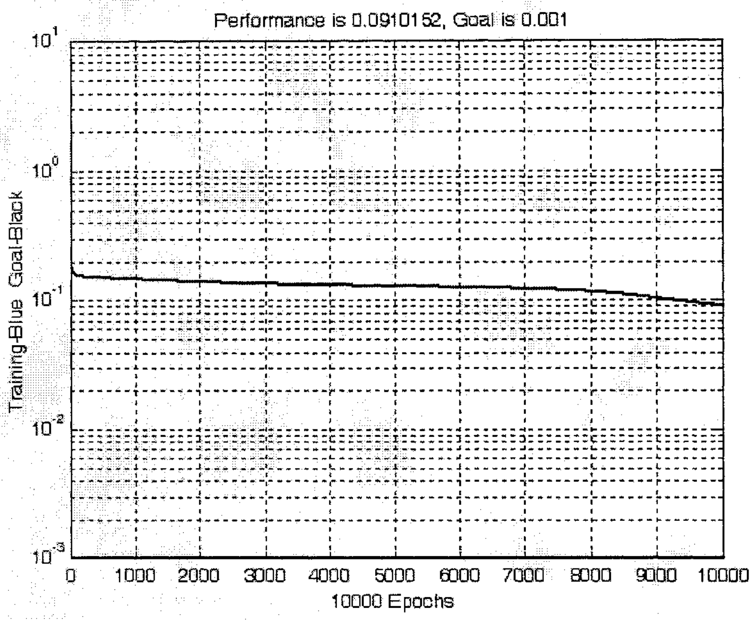


Figure7.9: Unsuccessful training result of BPNN with three hidden neurons

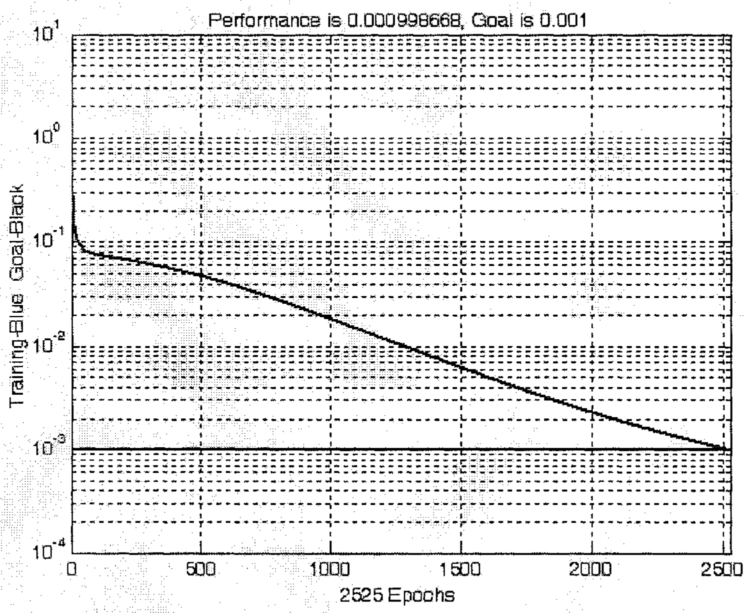


Figure7.10: Successful training result of BPNN with seven hidden neurons

The results show that providing a large enough number of neurons is provided, in this case a minimum of 7, then the BPNN can be trained to perform the function of the CA.

It can be seen that this minimum number of neurons is more than twice the number of inputs. It is surmised that, an initial guess for number of neurons in the hidden layer should be more than two times of the number of inputs.

The trained BPNN is applied to the same problem as that given to the CA in Figure 7.4 and the same result is obtained. To further substantiate this evidence 6 more CA rule sets are used to train the BPNN containing 7 neurons and then the output compared to that of the CA for a given input. The 6 rules used are ***184, 30, 55, 90, 163*** and ***170*** (selected at random) and the training results are shown in Table 7.2. The 20 step BPNN simulation patterns of these rules are illustrated in Figures 7.5 and 7.11.

Table 7.2: Results of several CA patterns training by BPNN

CA rule	Training results of 10 times		Percent of successful training (%)
	Successful	Unsuccessful	
30	10	0	100
55	10	0	100
90	10	0	100
163	10	0	100
170	10	0	100
184	10	0	100

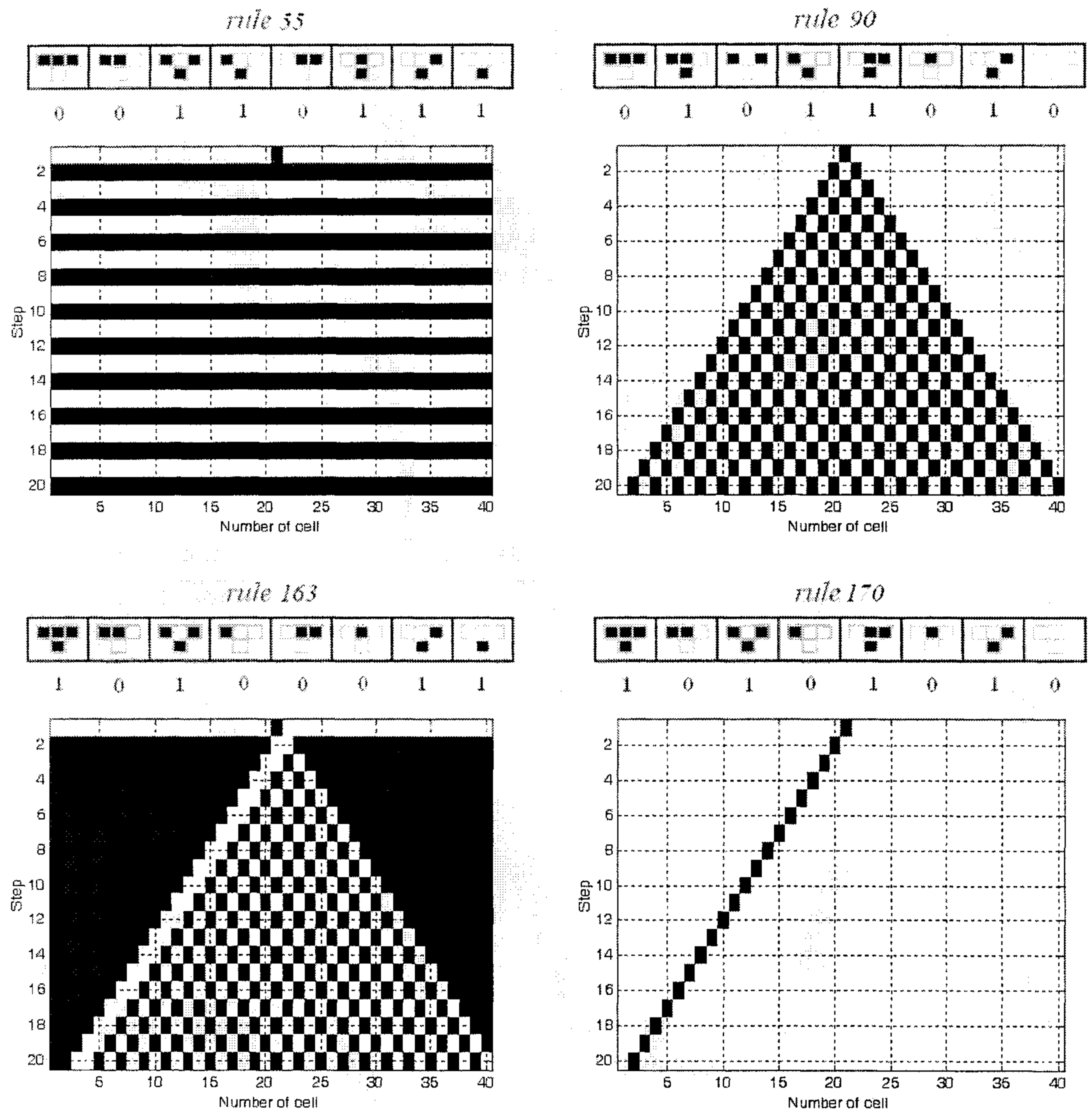


Figure 7.11: BPNN simulation results

From the experimental results, it is clear that any 1D-CA pattern can be learnt by the BPNN providing strong support to our conjecture. However, in this application two-dimensional images are being used and therefore it is necessary to extend the experiments to show that a BPNN can also perform the same as a 2D-CA.

7.4.2 2D-CA experiments

In this section, experiments are performed to show the ability of the BPNN to learn the behaviour of the 2D-CA. From Figure 7.6 it is clear that the number of 2D-CA neighbours in the 2D-CA is greater than in the 1D-CA, therefore this test is focussed on the training of the BPNN for an increased number of inputs. The Moore neighbourhood, in which each cell is connected to its eight immediate neighbours, is used, so BPNN now has 8 inputs and 1 output target. From the 1D investigation in the previous section it is known that the number of neurons required should be more than double the number of inputs. As an initial estimate of the number of neurons required in this case, 18 initial hidden neurons are chosen as illustrated in Figure 7.12.

If each neighbour has two possible states (0,1), then number of possible patterns at the input to the 2D-CA when the Moore neighbourhood is applied is $2^8 = 256$ patterns. All 256 CA patterns are set as input patterns for the BPNN, the target for each of the patterns is initially generated randomly and then applied in each test. Starting with 18 neurons in the hidden layer, the test is performed 10 times and the average success rate is recorded. The number of neurons is then doubled and the test performed again until the average success rate doesn't increase. The outcome of this test is listed in Table 7.3.

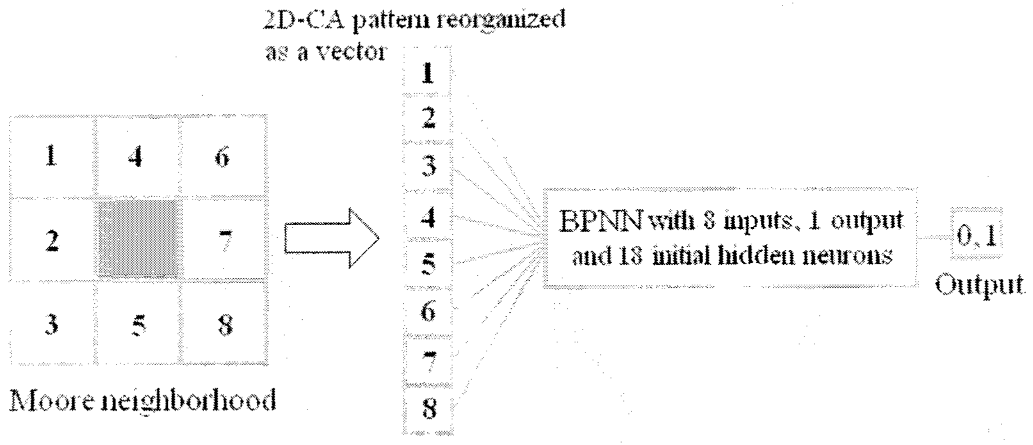

2D-CA pattern reorganized as a vector
1
2
3
4
5
6
7
8
1 4 6
2 7
3 5 8
Moore neighborhood
BPNN with 8 inputs, 1 output and 18 initial hidden neurons
0,1
Output

Figure7.12: The reorganising of Moore neighbourhood to BPNN inputs

Table 7.3: BPNN training results of 256 CA patterns with fixed target patterns

Number of hidden neurons	Training results of 10 times		Percent of average accurate patterns (%)
	Number of average error patterns	Percent of average error patterns (%)	
18	65.7	25.67	74.33
36	35.15	13.73	86.37
72	3.35	1.3	98.7
144	3.35	1.3	98.7

The test is then performed again but between each of the 10 cycles a new set of randomly generated targets is applied to the BPNN, this shows the ability of the BPNN to perform under different sets of CA rule. The results of this experiment are shown in Table 7.4.

Table 7.4: BPNN training results of 256 CA patterns with random target patterns

Number of hidden neurons	Training results of 10 times		Percent of average accurate patterns (%)
	Number of average error patterns	Percent of average error patterns (%)	
18	68.3	26.68	73.32
36	33.2	12.97	87.03
72	3.5	1.36	98.64
144	39.1	15.27	84.73

The average number of error patterns shows the average number of patterns that did not agree with their targets. For example in Table 7.3, with 18 hidden neurons an average of 65.7 patterns out of the total of 256 patterns did not agree with the desired target over the 10 training cycles.

The results show that as the number of hidden neurons increases then the average number of patterns that result in an error decreases. A satisfactory level of 98% accuracy from the training was in both tests. However, although the BPNN is able to provide 98% accuracy, the memory required to produce this result was overwhelming and would be costly to apply in real time. As in the 1D-CA simulation, it was seen that the BPNN can be trained to learn the 2D-CA patterns and therefore with the same input the 2D-CA and BPNN will produce the same pattern to an accuracy of 98% on average.

7.5 Summary and Discussion

The relevant basic theory behind both the CA and BPNN and the manner in which they can be associated has been presented. From the test results of BPNN training on 1D-CA patterns, it was found that the relation between the number of BPNN inputs and the number of hidden neurons could be used to set up an initial number of hidden neurons. When a small number of input patterns are employed, the initial number of hidden neurons should be set to at least twice of the number of BPNN inputs. However, the exact number of hidden neurons required to achieve a suitable error level from the training cannot be determined. Further experimentation is needed to allow the required number of hidden neurons to be calculated. As the number of inputs and patterns increase a longer time is needed for training.

The results of the experiments strongly support the prior conjecture that a BPNN can be trained to synthesise the CA patterns up to an accuracy of 98%, which is deemed to be more than sufficient in the application of interest. Consequently, it can be concluded that the application of fusing a CA and BPNN as a prediction method is feasible. An investigation of this prediction system is undertaken in the next chapter.

Chapter Eight

Prediction Module (PM)

8.1 Introduction

In Chapter 7 an investigation into the feasibility of fusing a CA and NN to form a motion prediction system was carried out and found to be in support of the initial conjecture. In this chapter the proposed algorithm employed in the prediction module is tested on actual consecutive images to observe its accuracy.

The chapter is organized as follows: In Section 8.2, the proposed prediction module (PM) is introduced and detailed. Section 8.3 describes how data is collected for BPNN training. Experiments are performed in Section 8.4, to observe the ability of the prediction module to track a pedestrian's movement through consecutive images. Finally, an evaluation of the method and conclusions are provided in Section 8.5.

8.2 Prediction Module (PM)

The proposed prediction module is made up of four parts as follows: 1) Grid-generating block 2) CA plane to vector block 3) BPNN classification and 4) predicted locations provider. The block diagram of the prediction module is depicted in Figure 8.1, the functionality of each block is described in detail below.

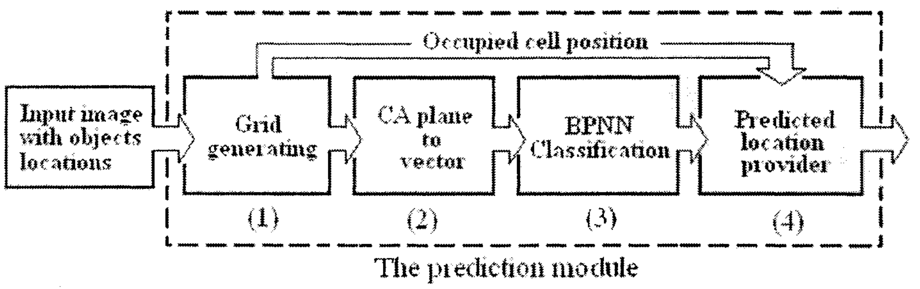


Figure 8.1: A block diagram of the prediction module

8.2.1 Grid-generating block

The purpose of this block is to superimpose a grid on the actual image to allow the CA algorithm to be applied. The image with object locations, obtained from the object detector, is fed through the block and the grid is generated. The dimension of the grid is an essential parameter as it is directly related to the movement of an object between frames. If a large grid is used then many frames will be required to move an object from one grid square to another. In contrast, the smaller the grid square requires fewer frames to move the object between cells. It is desirable that the dimensions of the grid squares match that of the object (person) dimension in the image. A suitable grid square size should be able to contain most of the object area as shown in Figure 8.2. The object location, obtained from the object detector, is used as an initial position to establish the grid or lattice on the input image, which is then passed through the CA plane to the vector block in order to establish the CA plane. Experimental work performed to allow definition of the grid dimensions are demonstrated and discussed in Section 8.4.

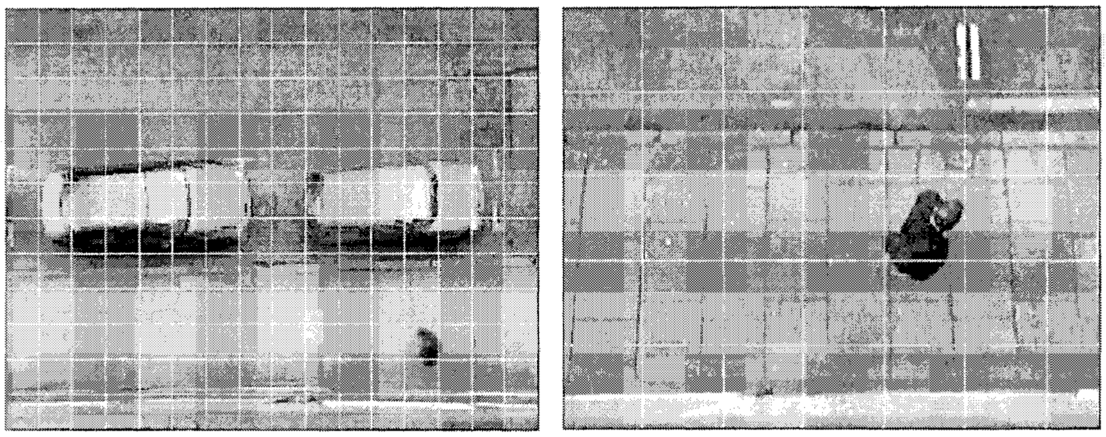


Figure 8.2: The latticed images with two different grid dimensions

8.2.2 CA plane to vector block

After superimposing the grid on the image, the latticed image is used to establish the CA plane; the information of the latticed image is then mapped on to the CA plane. One represents a cell occupied by an object and zero represents an empty cell. Consequently, the dimensions of the CA plane are equal to the number of lattice cells generated on the image. Since the pedestrian dynamic is generically two-dimensional [88] the 2D-CA principle is applied to the CA plane after the mapping process. Each CA site and its neighbourhood is reorganized into vector form then used as an input vector to the BPNN in order to provide the corresponding output with regards to the input vector. An example of this process is illustrated in Figure 8.3.

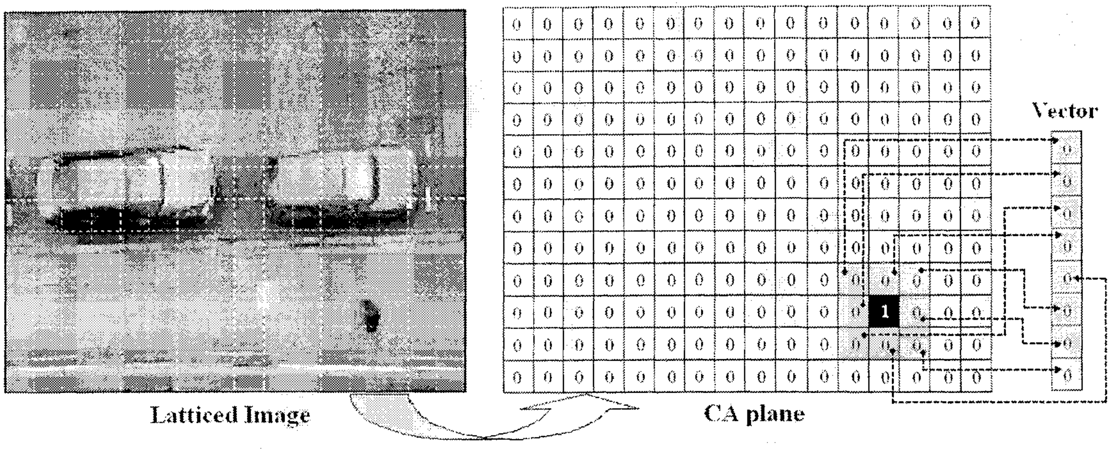


Figure 8.3: The process of organizing the latticed image to vector form

8.2.3 BPNN Classification

According to the CA algorithm, the CA rule set has to be applied to every cell on the CA plane in order to have the new stage value. As aforementioned in Chapter 7, the CA rule set is obtained from empirical data that is unconventional for our application because the pedestrian movement behaviour concerns more than one CA rule. Consequently, the BPNN is proposed for use as a classification tool that offers the following advantages: (i) the behaviour of pedestrian movement can be captured and learned from actual consecutive images which means the model is independent from the use of empirical data and (ii) The application of the CA rule to every cell on the CA plane can be avoided because only the occupied cell is of interest and therefore the cell with value 1 is located to attain the CA pattern of only that cell, and (iii) it makes the model more flexible allowing it to be applied to other applications for which the CA algorithm is suitable.

The vector from the CA plane is classified by the BPNN to provide an output to be used to make decisions regarding cell movement and hence object movement in the next block. The data used to train the BPNN will be discussed in more detail in Section 8.3.

8.2.4 Predicted Location Provider

The function of this block is to obtain the estimated location of the object cell after recognition processing. The output obtained from the BPNN classification and the object position on the CA plane are evaluated in this block then the future location on the CA plane of the following few frames is given. It is used to confirm the location of the object obtained from the object detector.

8.3 Data collection for BPNN

In this section, the data collection for BPNN training is explained. The data is collected using actual consecutive images applied to the same process. The required data consists of input patterns and output targets. The input patterns can be obtained from the CA plane and passed to the vector block at any time (t) and the output target is taken as the pattern once the object has passed to the next cell. This may take more than one frame ($t+n$) and is dependant on the speed at which the object is moving. The number of frames taken can be found by observing the point at which the current cell becomes empty. The input pattern and its target can be collected using this method. The collected data is used to train the BPNN.

8.4 The prediction module (PM) experiments

In this section, the experiments performed on the proposed prediction method are demonstrated and discussed. These experiments are designed to obtain a suitable grid dimension for the grid-generation block and testing of the tracking process using the prediction method.

8.4.1 Defining of grid dimension

Due to the link between the grid dimension and the object cell movement it is essential that the relationship between grid size and the number of required frames to obtain accurate outcomes is defined. Two types of grid generation, static and dynamic also need to be considered. Firstly, the grid dimension against the number of frames running per cell movement is investigated using a sequence of images. An example of the test image is shown in Figure 8.4.

Figure 8.4 shows (a) the initial location of the object and then the location after (b) 10 frames, (c) 11 frames, (d) 12 frames, (e) 13 frames, and (f) and 14 frames. This example shows that for the given grid size of 48x48 pixels then it takes 13 frames for the object to completely move from one cell to the next. Results obtained using different grid sizes with the same image sequence are listed in Table 8.1.

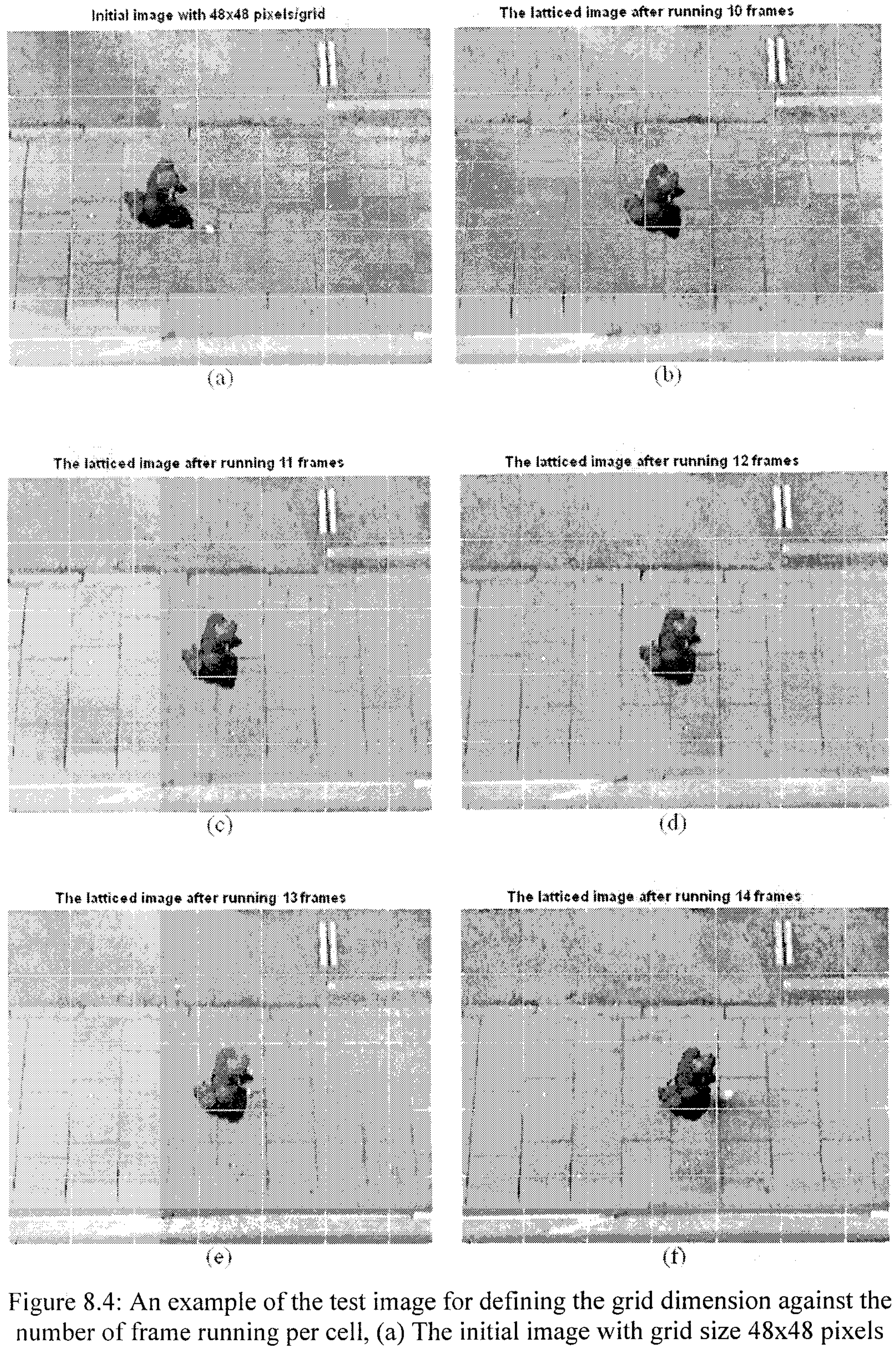

Initial image with 48x48 pixels/grid
The latticed image after running 10 frames
(a)
(b)
The latticed image after running 11 frames
The latticed image after running 12 frames
(c)
(d)
The latticed image after running 13frames
The latticed image after running 14 frames
(e)
(f)

Figure 8.4: An example of the test image for defining the grid dimension against the number of frame running per cell, (a) The initial image with grid size 48x48 pixels

Table 8.1: Shows the result of grid dimension adjusting

Grid dimension (pixels)	Number of running frames per cell
40x40	10
44x44	11
48x48	13
52x52	14
56x56	15

From Table 8.1, it can be seen that when the grid dimension increases the number of running frames for a cell to move increase. Consequently, it can be concluded that the number of frames required for cell movement is directly proportional to the grid dimension.

Two different methods used to establish the grid are now considered. In static grid formation the grid dimension the initial location of the grid is fixed regardless whether the object initially appears in one, two or more of the squares. Consequently, it is very difficult to fit the initial object within a single grid square (see Figure 8.5), which will result in prediction error. In contrast, the dynamic grid is established using object locations to determine the initial location of the grid to ensure it lies within a single grid square. Examples of the dynamic grid are shown in Figure 8.6.

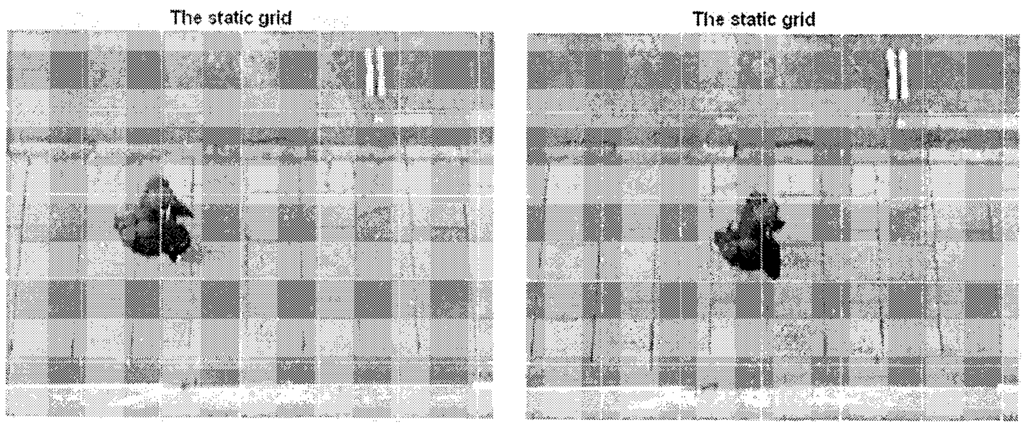


Figure 8.5: The example of the latticed images using the static grid establishing that the grid is unable to cover the whole object.

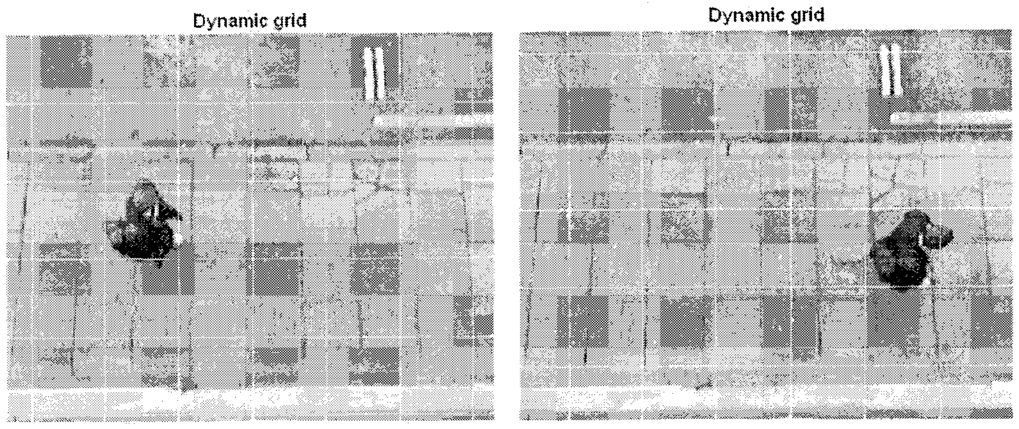


Figure 8.6: The examples of the latticed image using the dynamic grid establishing that the grid is able to cover object at any locations within the image.

Application of the dynamic gird is used in this system due to its advantages over the static grid, however, in the case where more than one object exists in the image then it may not be possible for the grid to align itself such that all objects lie within a single cell. The occlusion of the objects is also not possible since the lattice is established using information from the detector which has already separated each object into individual cells. Figure 8.7 shows an example of a latticed image with two objects, one is chosen as the object of interest and as a result the other is found on the line adjoining two cells. Since the CA requires an individual site that is representative of each object it is necessary to apply further grids for each individual object. Consequently, the number of CA planes generated individually as layers is equal to the number of objects in the image, each object therefore has its own CA plane; the prediction can be performed on each CA plane layer as shown in Figure 8.8. The information on the CA plane, which is the cell site and its neighbours, is updated and exchanged between the CA planes.

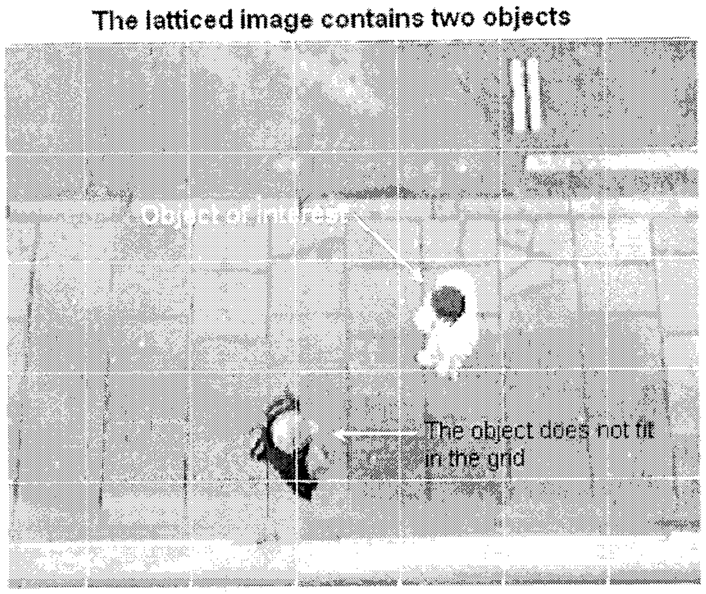

The latticed image contains two objects
Object of interest
The object does not fit in the grid

Figure 8.7: The established grid of the interested object does not cover another object within the image

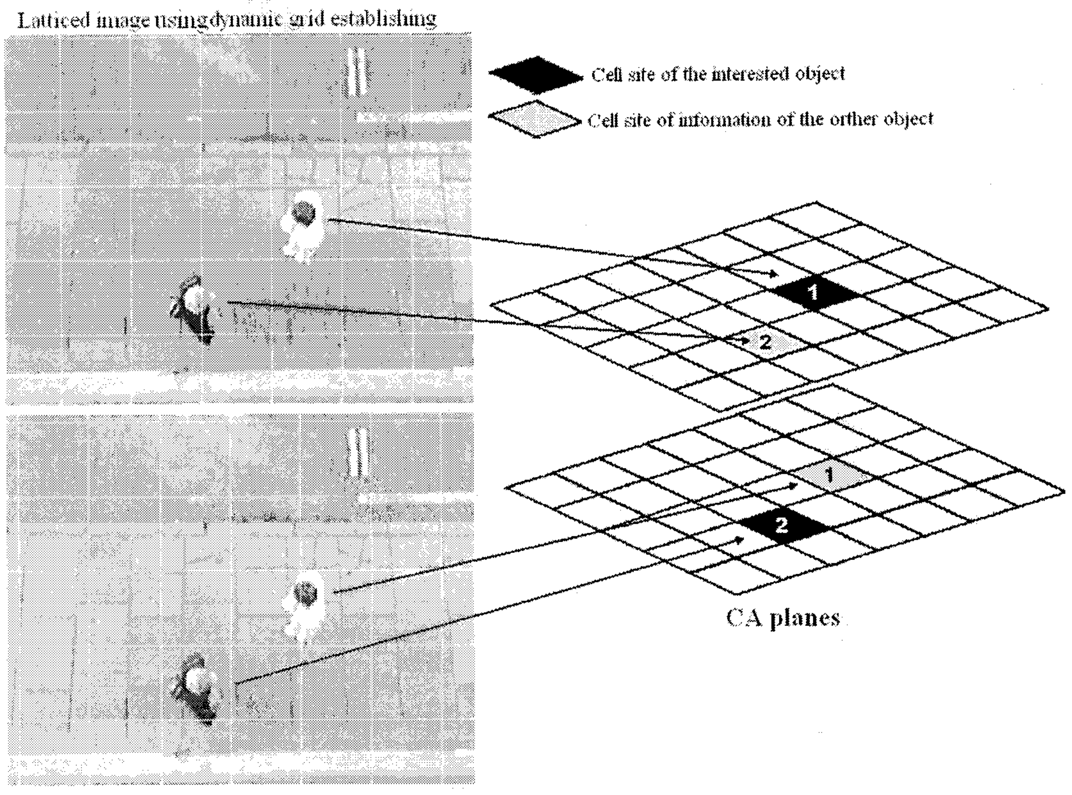

Latticed image usingdynamic grid establishing
Cell site of the interested object
Cell site of information of the orther object
1
2
1
2
CA planes

Figure 8.8: CA planes of each object by using dynamic grid establishing

Having discussed the factors concerning the generation of the grid on the image experiments of the prediction method are now performed on the image sequence in the next section.

8.4.2 The tracking process using the prediction method

The prediction method is applied to the image sequence in order to evaluate its ability to track the motion of the objects through the sequence. The prediction module requires three essential parameters regarding the object of interest these are (i) object location, (ii) direction of movement and (iii) the speed of movement. It is possible to obtain these parameters from the object detector; the manner in which this is achieved is discussed detail in the automatic tracking test. Firstly, the method needs to be tested using a simple case to see how it functions when applied to actual images.

8.4.2.1 The test on the prediction tracking using manual locating

The test is performed on a sequence of images containing bi-directional pedestrian movement. The initial location direction and speed of the pedestrian in the image is set manually by pre observation of the sequence. The prediction tracking process is then monitored step-by-step in order to evaluate its performance. The first image of the test sequence is shown in Figure 8.9, on which the location and direction are manually defined and the speed is set at 10 frames per cell. The Von Neumann neighbourhood shown in Figure 8.10 is employed. Use of this neighbourhood means that the value of the cell of interest at the next time-step is dependent on the values of the cells above (north), below (south), right (east) and left (west). Due to the behaviour of human movement the forward direction is defined as the direction in which the person is facing, the Von Neumann neighbourhood sufficiently covers the possibilities for bi-directional movement.

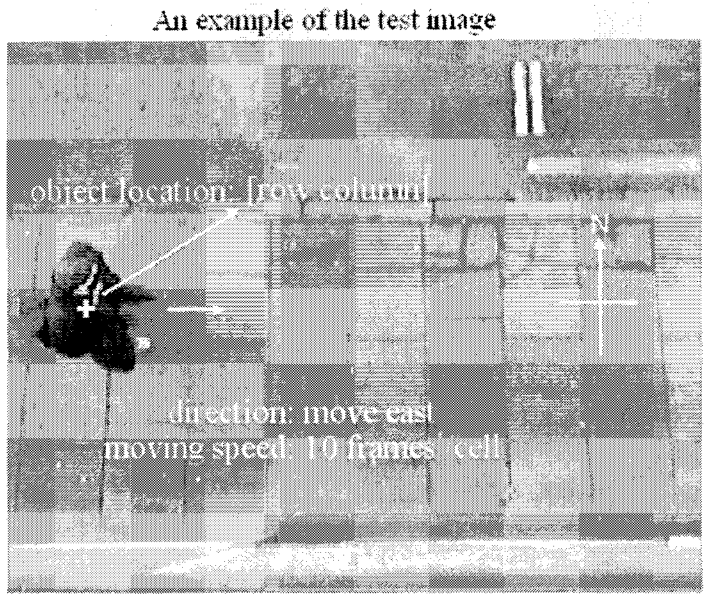

An example of the test image
object location: [row column]
N
direction: move east
moving speed: 10 frames/cell

Figure 8.9: The test image with manually set for the location, direction: moving east and initial of moving speed:10 frames per cell.

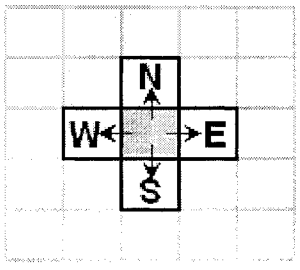


Figure 8.10: The Von Neumann neighbourhood

The results of using the image sequence starting from Figure 8.9 to test the motion prediction system are shown in Figure 8.11. The white square on each of the images represents the cell obtained from the predictor. The dynamic grid generated using the initial location manually taken from Figure 8.9.

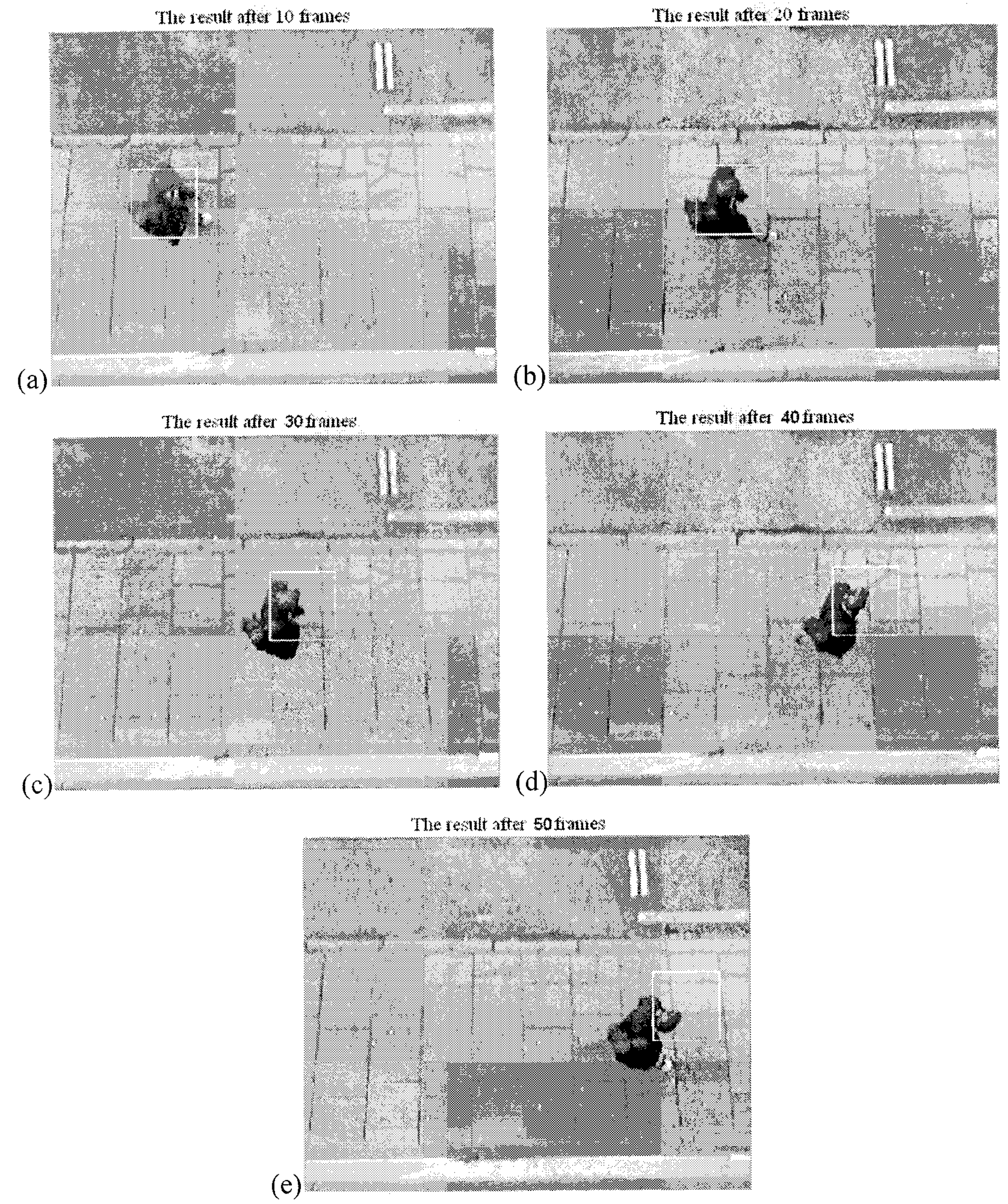
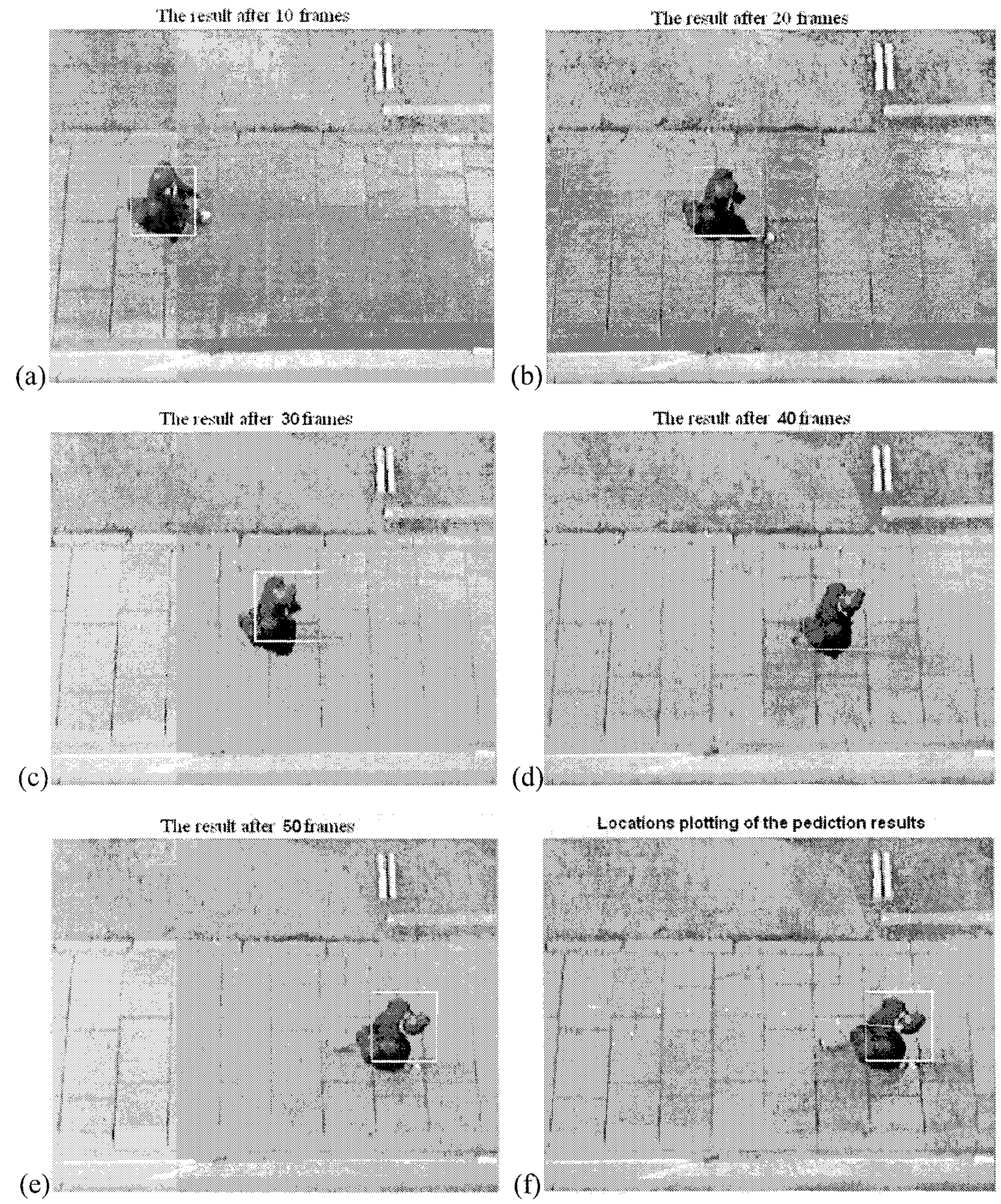

The result after 10 frames
(a)
The result after 20 frames
(b)
The result after 30 frames
(c)
The result after 40 frames
(d)
The result after 50 frames
(e)

Figure 8.11: The tracking results of the prediction method

Figure 8.11 shows the ability of the motion tracking system to prediction the pedestrian's movement. It can be seen that good predictions of the pedestrian's movement are obtained over the first 20 frames of the sequence. Beyond this time the location predicted begins to shift in front of the object, until after 50 frames the location predicted contains only the person head. This suggests that the object began to move more slowly after the 20th frame, this is because of the camera lens used to capture the images. Another reason that the whole object is not contained within the predicted cell is that the pedestrian's movement is flexible, not limited to lanes as it would be in, for example, vehicular flow. However, these shortcomings could be compensated for or at least improved upon by updating of the dynamic grid each time the predicted cell location changes. For instance, the confirmed location provided by the previous simulation is used to generate the grid and predict the next movement.

The results obtained by making this suggested improvement are shown in Figure 8.12. It is seen that the tracking of the pedestrians movement have been improved with the cell located more accurately upon the object than in the previous test. Consequently, it can be concluded that the solution conjectured to tackle the shortcomings seen in the previous test is able to solve these drawbacks.

The proposed prediction method has been seen to provide satisfactory results on an image sequence that shows one moving pedestrian. It should therefore also provide a satisfactory level of performance on images containing more than one pedestrian since the prediction is made on each objects individual CA plane. An experiment is required to provide confirmation of this statement.


The result after 10 frames
The result after 20 frames
(a)
(b)
The result after 30 frames
The result after 40 frames
(c)
(d)
The result after 50 frames
Locations plotting of the pediction results
(e)
(f)
Figure 8.12: The tracking results of the improved prediction method

The same system is now applied to an image sequence contains two pedestrians both of which are moving; the initial image in the sequence is that of Figure 8.7. There are two persons moving in the different directions, the speed of their movement is determined as is 13 frames per cell as shown in Table 8.1. The results of the simulation, in which the initial locations directions and speeds of the pedestrians are manually supplied, are shown in Figure 8.13.

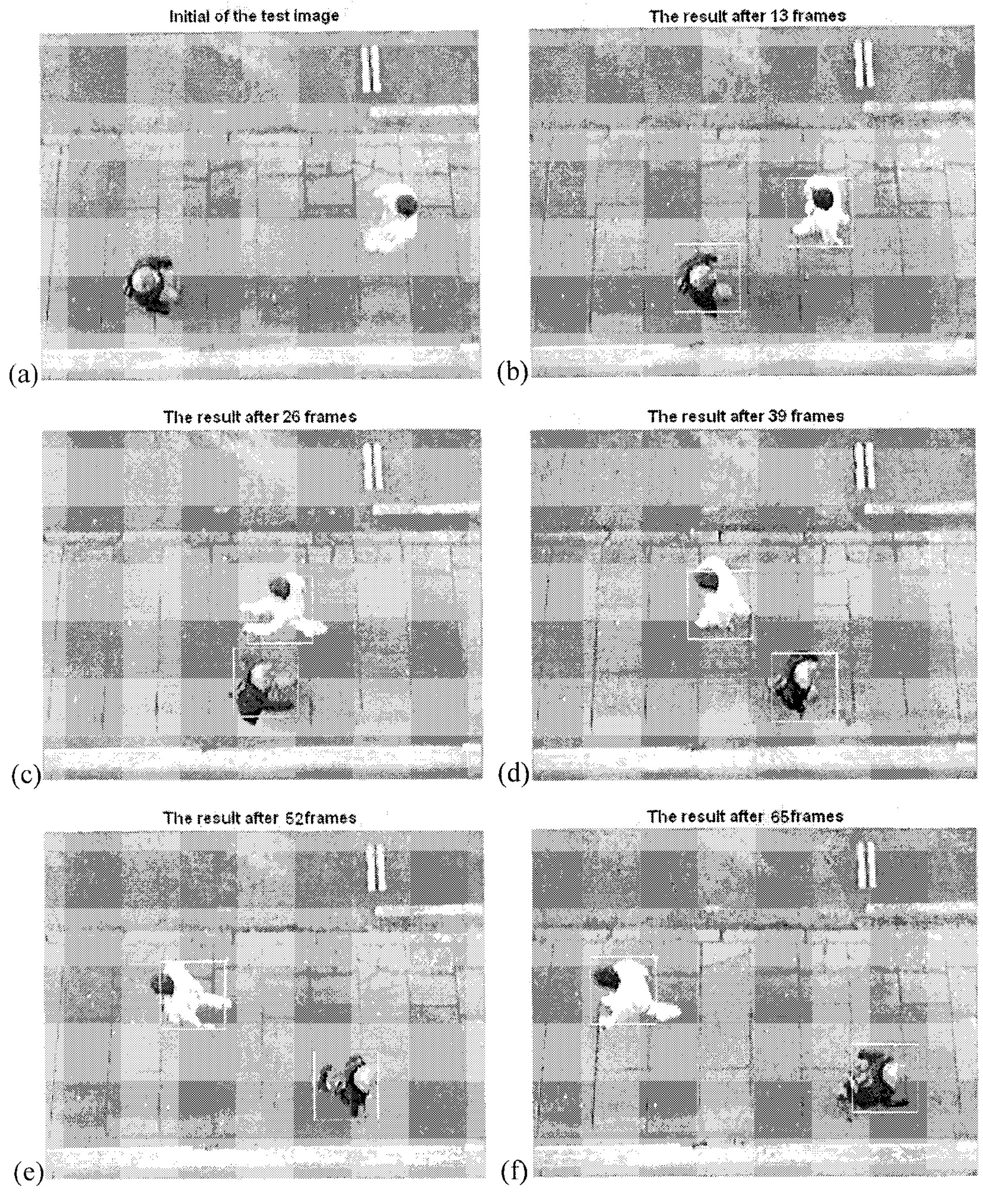

Initial of the test image
The result after 13 frames
(a)
(b)
The result after 26 frames
The result after 39 frames
(c)
(d)
The result after 52frames
The result after 65frames
(e)
(f)

Figure 8.13: The tracking results of the prediction method

Figure 8.13 (a) is the initial image and the initial locations of two pedestrian are manually provided. The outcomes of the motion predictor are seen in Figure 8.13 (b) (c) (d) (e) and (f) after running 13, 26, 39,52 and 65 frames images, respectively.

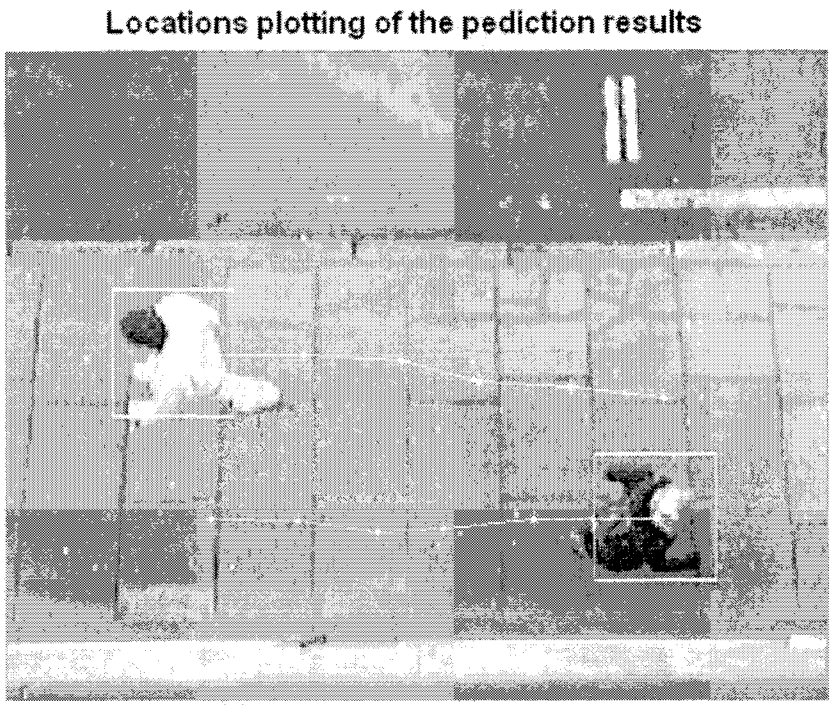


Figure 8.14: The locations plotting of prediction tracking

The results show that the prediction method is able to provide accurate predictions of the pedestrian positions over the next several frames. From the experiments it is noted that the speed of pedestrian's movement is an essential parameter and has a significant effect on the prediction accuracy. With respect to the experiments so far, it can be concluded that the proposed technique is able to perform the desired tracking process. In the next section the system is automated so that the initial locations direction and speed are no longer required as initial conditions and its performance is tested.

8.4.2.2 The test on the prediction tracking using automatically locating

The previous tests have demonstrated that the prediction technique offers the desired functionality. In order to perform the tracking automatically, the initial information regarding location direction and speed is obtained from the object detector as discussed in Chapter 3. Prior to the use of the object detector from Chapter 3, modifications are required to provide parameters suitable for the prediction module.

The object location needed as an input to the prediction module should locate the centre of the object, however from the performance indicated in Chapter 3 it cannot currently provide such accurate information. Changing the number of gradient samples in the smoothing process can account for the required increase in accuracy. In order to be able to accurately determine the centre coordinate of the object a reasonable number of coordinate locations from the object detector is required from which the centre can be determined. Therefore by decreasing the number of gradient samples, the number of the coordinates on the object area is increased, as shown in Figure 8.15 (a). From testing, 11 gradient samples are found to be suitable for the application since this produces a reasonable number of coordinates.

Where only one object is present in the image it is relatively easy to locate the central coordinate. Subtracting the minimum coordinate position from the maximum in both the horizontal (column) and vertical (row) dimensions and then adding half of this value to the minimum coordinate allows the centre to be obtained. The result obtained using the method described to locate the centre of the object is shown in Figure 8.15 (b).

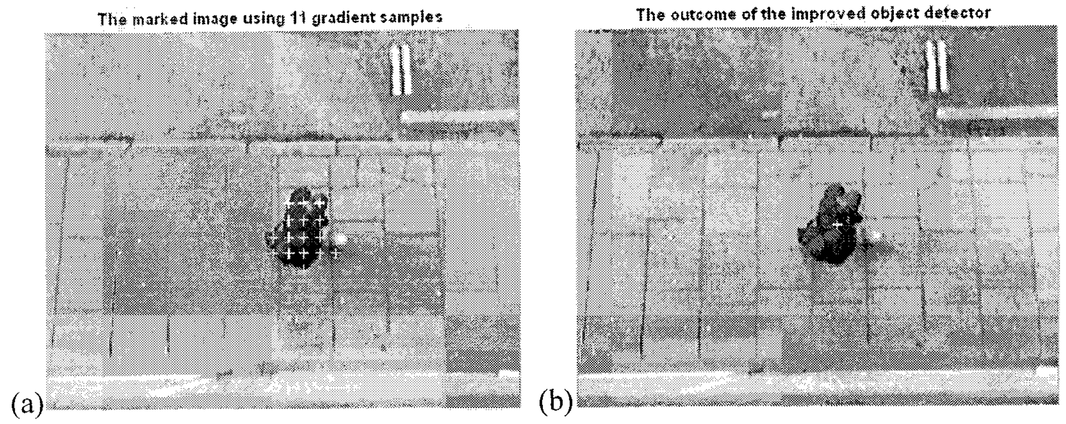

The marked image using 11 gradient samples
The outcome of the improved object detector
(a)
(b)

Figure 8.15: The object detector outcome after improving

Where there is more than one object in the image, the vertical and horizontal projections are used to determine the locations of the different objects. In the example of Figure 8.16 (a) it is clear by observation especially of the vertical projection that two distinct regions will be identified representing the two objects, from this information the centre of each object can then be found as seen in Figure 8.16 (b). This process was also discussed in Chapter 3.

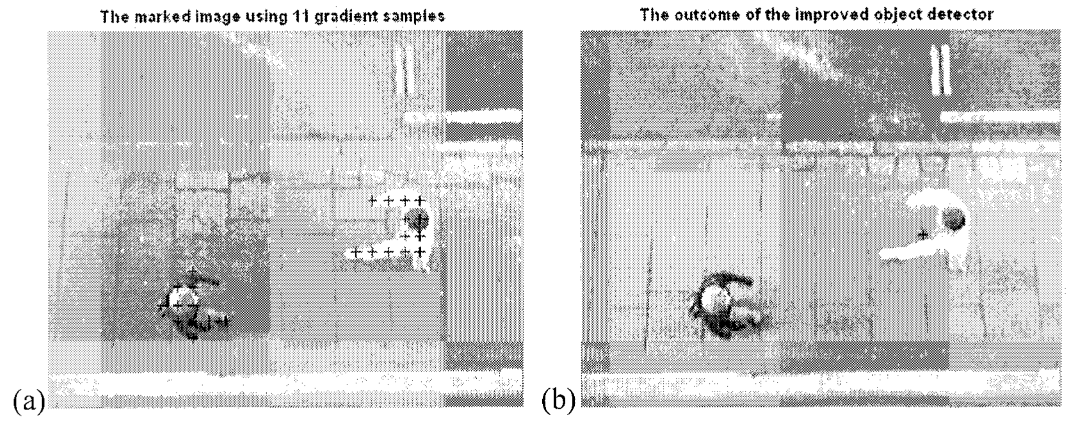

The marked image using 11 gradient samples
The outcome of the improved object detector
(a)
(b)

Figure 8.16: The test of the object detector on the image that contains more than one object

The initial direction of movement of each object is also required for the prediction module. For bi-directional movement this can be obtained by calculating the initial locations at time *t* and then the locations at time *t*+2. The distance between the two points is defined in term of Δx and Δy. Trigonometry is then used to calculate the direction in terms of an angle in degrees expressed as:

$$\theta = \tan^{-1}\left(\frac{\Delta y}{\Delta x}\right) \qquad (8.1)$$

$$\Delta x = x(t+2) - x(t) \textit{ and } \Delta y = y(t+2) - y(t)$$

where: Δx is the distance between two points in horizontal (column), Δy the distance between two points in vertical (row) and θ the direction of the movement.

An example of obtaining the direction of the movement of an object is shown in Figure 8.17. From the initial image the coordinate of the object is row: 115 ($y(t)$) and column: 54 ($x(t)$). For the following image the coordinate provided is 120 ($y(t+2)$) and column: 65 ($x(t+2)$).

Therefore:

$$\Delta x = 65 - 54 = 11 \qquad \theta = \tan^{-1}\left(\frac{5}{11}\right) = 24.44^\circ$$
$$\Delta y = 120 - 115 = 5$$

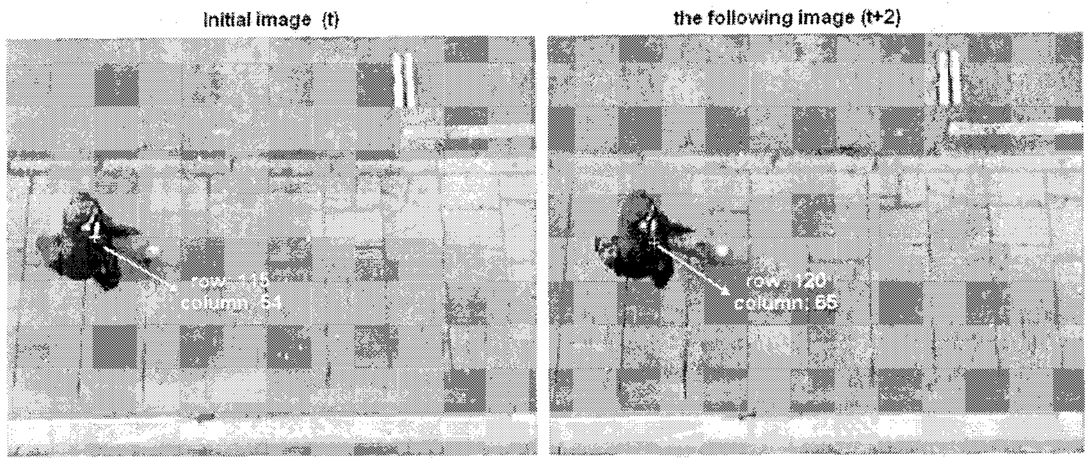


Figure 8.17: The example image for defining of the direction movement

From the results it is seen that θ = 24.44° and the positive sign attained from the subtraction of $\Delta x(t)$ - $\Delta x(t+2)$ means that the object is moving to the right (East). In this case the value of θ provides confirmation of this direction, for right hand movement θ should lie within ±45°. In contrast if Δx provides a negative sign the direction of movement should be to the left (West) and θ should lie within 180°±45°. Consequently, it has been seen that this calculation can be used to define the direction of movement.

Finally, the predictor requires the speed of movement of the object this can be attained using the information from which the direction of movement was calculated. The speed of movement is defined as the number of frames per cell; having already found the distance the object has moved as 11 pixels over a period of three frames and with a cell size of 48x48 pixels (i.e. the object has to move through 48 pixels to move to the adjacent cell) then the speed can be defined as a ratio. The speed of movement in this example is as follows: the object has moved 11 pixels in 3 frames; therefore it will move 48 pixels in 13 frames (i.e. the speed is defined as 13 frames per cell).

To show that the suggested modifications to the operation of the object detector are able to provide suitable information to the prediction module so that it can still accurately predict the movement of the object without requiring manual initialisation it is necessary to perform a test.

Initial zones over which the predictor works need to be defined to ensure that the whole object appears on the image prior to operation. Without this it may not be possible for the object detector to determine the movement and speed of the object. In the test the initial zones are set as 48 pixels from both the left and right sides of the image as shown in Figure 8.18. The middle line in the figure is used for counting when objects cross the line.

The outcomes of the test is shown in Figure 8.19, the location, direction and speed, were calculated as described using the first two images following the object passing the initial left hand line. The number of locations plotted from the automatic test is less than that obtained in the manual test because the initial zones shorten the effective image length. A second test is performed on the image containing two objects, as shown in Figure 8.20. The object locations and direction of movement are obtained correctly by the object detector but the speed of the objects is predicted as faster than the actual speed which leads to prediction error. Instead of using a variable speed calculated for each object in the sequence, a fixed speed based on Table 8.1 is employed for testing.

Although, some of the predicted locations attained were ahead of the actual object, the grid square in which the object is predicted and located is the same; the results are shown in Figure 8.20.

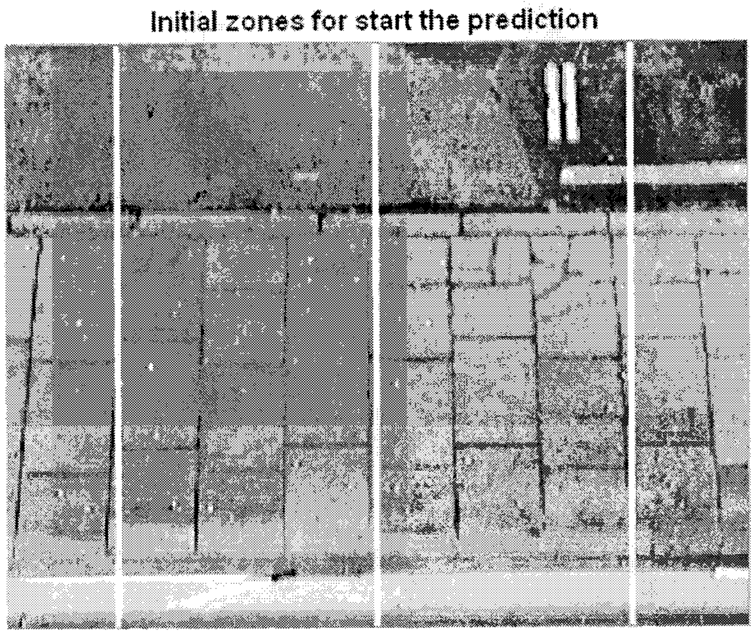


Figure 8.18: The initial zones for the prediction applying

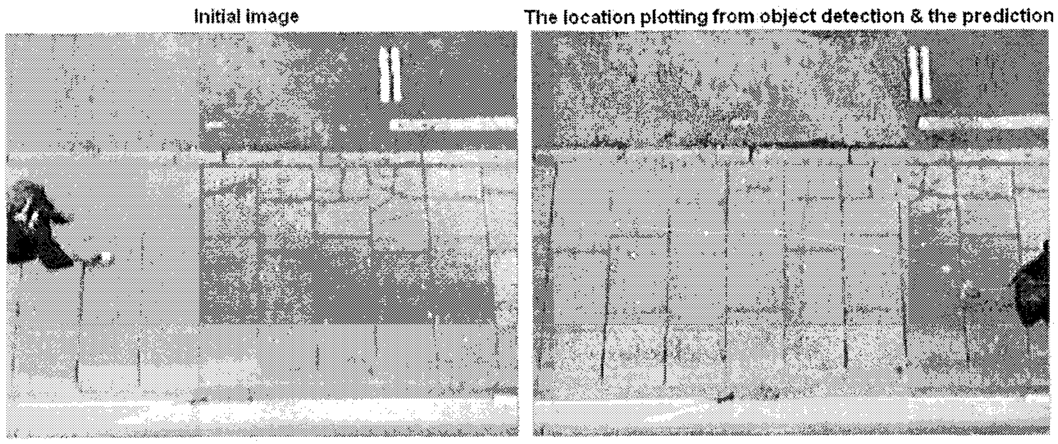

Initial image
The location plotting from object detection & the prediction

Figure 8.19: The outcomes of automatic tracking

Further testing has been carried out on images containing a higher density of objects and it was found that the object detector has difficultly in locating each object correctly. This leads to problems in the automatic calculation of the direction of movement and speed of the objects making prediction impossible.

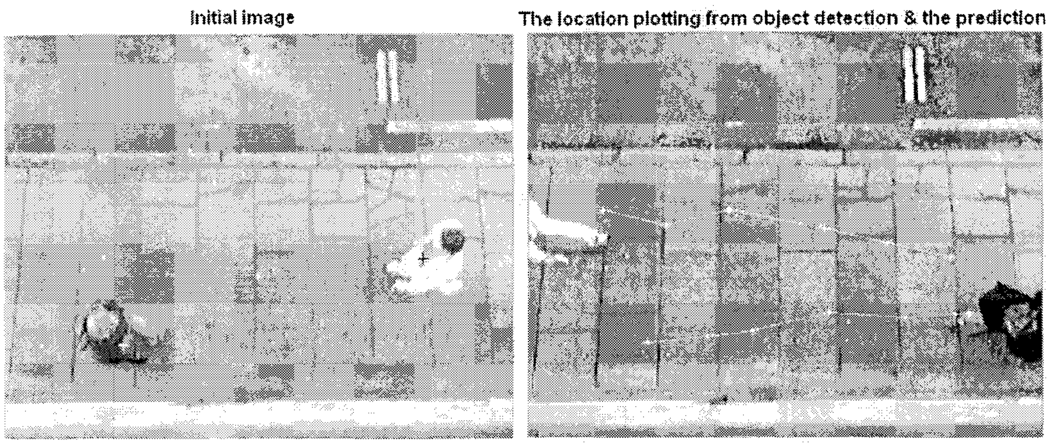

Initial image
The location plotting from object detection & the prediction

Figure 8.20: The outcomes of automatic tracking on the image with two objects

8.5 Evaluation of the method

The experiments used to test the ability of the prediction system have supporting have presented and only the performance of the prediction algorithm is now evaluated. From the investigations it is clear that the prediction method is able to provide accurate tracking of the objects movement, especially when the initial object locations, directions and speeds are manually provided as inputs. However, the limitations of the object detector during automatic prediction decrease the predictor's level of performance especially as the object density of the images is increased, though it is still able to perform satisfactorily when there is a low object density. Where the object density is too high the object detector is unable to provide the required information to the predictor and so as a whole the pedestrian counter cannot be evaluated.

This is the first time a CA and NN has been utilized for the prediction of pedestrian movement and it has been shown that provided the desired information can be supplied to the predictor then it is able to perform the required operation. The limitations in the system as a whole lie with the object detection process, which has been utilised in this work to test the prediction technique. The field of initial object detection that can provide a suitable input to the predictor leaves an interesting platform for further research work.

Chapter Nine

Conclusions and Future Work

This chapter provides the conclusion drawn from the experience gained during the research work and summaries the performance and advantages of the proposed pedestrian detection and tracking techniques. A discussion of the direction in which this work can be taken forward and where the findings may be applied in other areas is also presented.

9.1 Conclusions

The motivation behind this work was derived from the desire to improve the accuracy and computational cost of pedestrian tracking systems using video images. The studies were focused on the applications of pattern recognition and classification techniques with the aim of improving accuracy as presented in Chapters 5, 7 and 8. Not only are pattern recognition techniques involved in the system, but also image processing plays an essential role in terms of pre-processing input information (i.e. video images) prior to being able to apply the use of any recognition technique. The preprocessing was mainly dealt with in Chapters 3 and 4 as a pre-requisite to being able to develop and test the prediction technique.

The object detector in Chapter 3 applies basic image processing techniques to locate the objects of interest within the image. The results obtained from this relatively simplistic object detector were deemed suitable for use in the development of the motion predictor while providing fast processing performance. This allowed a large number of training sets to be obtained in a short period of time for use in the training of the SOM used as part of the recognition technique employed later.

The feature extraction technique was discussed in Chapter 4, here it was seen that texture as an image feature is unsuitable for use as information in the recognition of a person's head. Investigation also showed that an individual person's head is very difficult to distinguish from others. It was found that the feature extraction technique has the greatest effect in terms of the classification and recognition tasks.

The novel prediction method presented in Chapters 7 and 8 has proven to provide good results for prediction when the initial required inputs to the system are supplied manually, (i.e. the correct inputs are pre-determined). However, the inability of the object detector to adequately predict the direction and speed of objects in the images meant that use of the predictor in an automatic mode led to limited accuracy in object tracking. Nevertheless, from the experiments it can be assumed that providing the object detector was able to provide the required information that the system would perform as desired. Since the prediction algorithm is able to learn it is more flexible than those used in previous systems and could therefore be applied in other areas such as vehicle detection or gesture movement detection.

9.2 Future work

The positive results obtained from the testing of the novel prediction method presented have highlighted the importance and value of this work. However, the initial system constructed to perform the measurements has physical limitations, which have been appropriately discussed. One area of further work lies in dealing with these limitations.

The main area of future work resulting from this thesis that needs to be undertaken is the further development of the object detection system. Although the object detection process applied in this thesis was produced in order to allow the development and investigation of the prediction technique conjectured, in order to make a truly automatic pedestrian counting system an investigation into more object detection techniques is required.

Also with the increasing power of the computer, it is possible to perform far greater levels of processing in real-time. In terms of this system this will allow for more features to be used to identify objects and allow them to be tracked. This includes information such as colour to be examined.

A more rigorous investigation of the various operators applied in different sections of the work for example the erosion function in the object detector or the neighbourhood function in the motion predictor needs to be undertaken to find the most suitable shape for each with respect to this application.

Finally, further experimentation is required on the system when it is applied to the monitoring of pedestrians in non-controlled environments (e.g. outdoors) over long durations. This will test the systems durability as the initial conditions change, i.e. as the light decreases or when it is raining.

Reference:

[1] D. M. Gavrila, "The Visual Analysis of Human Movement: A Survey", in *Computer Vision and Image Understanding, January 1999, Vol. 73, No. 1,* pp. 82-98.

[2] W.H. Liang Wang, and Tieniu Tan, "Recent developments in human motion analysis", *Pattern Recognition, 2003, Vol. 36*, pp. 585-601.

[3] J. K. Aggarwal, Q. Cai, W. Liao, and B. Sabata, "Articulated and elastic non-rigid motion: a review", *Proceedings of the 1994 IEEE Workshop on Motion of Non-Rigid and Articulated Objects, 1994*, pp. 2 -14.

[4] J. K. Aggarwal, and Q. Cai, "Human Motion Analysis: A Review", *Computer Vision and Image Understanding, 1999, vol. 73*, pp. 428-440.

[5] Thomas B. Moeslund, and Erik Granum, "A Survey of Computer Vision-Based Human Motion Capture", *Computer Vision and Image Understanding, 2001, vol. 81*, pp. 231-268.

[6] Jessica Jun, Lin Wang, and Sameer Singh, "Video analysis of human dynamics– a survey", *Real-Time Imaging, vol. 9, 2003*, pp. 321-346.

[7] A. J. Schofield, T. J. Stonham and P.A. Mehta,"Automated people counting to aid lift control", in *Automation in Construction, 1997,vol.6,* pp. 437-445.

[8] A. J. Schofield, P. A. Mehta and T. J. Stonham, "A System for Counting People in Video Images Using Neural Networks to Identify the Background Scene", in *Pattern recognition, 1996,vol. 29, No. 8,* pp. 1421-1428.

[9] A.N. Marana, S.A. Velastin, L.F. Costa, and R.A. Lotufo, "Estimation of crowd density using image processing", in *IEE Coll. Image processing for security applications, Savoy Place, London, March 1997*, pp. 11/1-11/7.

[10] X. Zhang and G. Sexton, "Automated human head location for pedestrian counting", in *IEE IPA, July 1997,* pp.535-540.

[11] J. S. McKenna, S. Jabri, Z. Duric, A. Rosenfeld and H. Wechsler, "Tracking Groups of People", *Computer Vision and Image Understanding, 2000,vol. 80, No. 1,* pp. 42-56.

[12] A. Shino and J. Sklansky, "Segmentation of people in motion", in *IEEE Workshop on Visual Motion, 1991*, pp. 325-332.

[13] J. Segen, and P. Sarma, "A camera-based system for tracking people in real time", *Proceedings of the 13th International Conference on Pattern Recognition, 1996.*

[14] M. Oren, C. Papageorgiou, P. Sinha, E. Osuna, and T. Poggio, "Pedestrian detection using wavelet templates", in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition, San Juan, 1997*, pp. 193-199.

[15] B.Boser, I. Guyon, and V.N.Vapkin, "A training algorithm for optimal margin classifiers", *presented at Fifth Annual Workshop on Computational Learning Theory, San Mateo, CA:Morgan Kaufmann, 1992.*

[16] Cortes C. and V.N.Vapnik , "Support Vector Networks", *Machine Learning,1995, vol.20,* pp.273-297.

[17] V. N. Vapkin, Statistical Learning Theory. *New York: Wiley, 1998.*

[18] C. Papageorgiou, and T. Poggio, "A pattern classification approach to dynamical object detection", *in Proc. of the Seventh IEEE International Conference on Computer Vision, 1999.*

[19] A. Baumberg and D. Hogg, " An efficient method for contour tracking using active shape models, *in Proc. of IEEE Workshop on Motion of Non-Rigid and Articulated Objects, Austin, 1994*, pp. 194-199.

[20] M. Isard and A. Blake, "Contour Tracking by stochastic propagation of conditional density", *in Proc. of European Conference on Computer Vision, Freiburg, Germany, 1996.*

[21] M. Isard and A. Blake, "CONDENSATION—Conditional Density Propagation for Visual Tracking", *in International Journal of Computer Vision, 1998, vol.29*, pp. 5-28.

[22] P. Tissainayagam and D. Suter, "Contour tracking with automatic motion model switching", *in Pattern Recognition, 2003,vol. 36*, pp. 2411-2427.

[23] D. K. Heegu Kang and Sung Yang Bang, "Real-time multiple people tracking using competitive condensation", *in Proc. of 16th International Conference on Pattern Recognition, 2002.*

[24] I. Haritaoglu, D. Harwood and L.S. Davis, "W4: Who? When? Where? What? A real time system for detecting and tracking people", *in Proc. of Third IEEE International Conference on Automatic Face and Gesture Recognition. 1998.*

[25] I. Haritaoglu, D. Harwood and L.S. Davis, "Hydra: multiple people detection and tracking using silhouettes", *in Proc. of International Conference on Image Analysis and Processin, 1999.*

[26] C. Wren, A. Azarbayejani, T. Darrell and A. Pentland, " Pfinder: Real-time tracking of the human body", *in IEEE Transcations on Pattern Analysis and Machine Intelligence, 1997, vol. 19, No. 7,* pp. 780-785.

[27] B. Heisele, U. KreBel and W.Ritter, "Tracking Non-Rigid, Moving Objects Based on Color Cluster Flow", *in IEEE, vol. No. 1997,* pp. 257-260.

[28] J. S. McKenna, S. Jabri, Z. Duric, A. Rosenfeld and H. Wechsler, " Tracking interacting people", *in Proc. of Fourth IEEE International Conference on Automatic Face and Gesture Recognition, 2000.*

[29] Wenmiao Lu, and Yap-Peng Tan, "A color histogram based people tracking system", *in The 2001 IEEE International Symposium on Circuits and Systems,ISCAS 2001.*

[30] M. Rossi and A. Bozzoli, "Tracking and counting moving people", *in Proc. of ICIP-94, IEEE International Conference on Image Processing, 1994.*

[31] X. Zhang and G. Sexton, "Automatic pedestrian counting using image processing techniques", *Electronic letters, 1995, vol. 31*, pp. 863-865.

[32] X. Zhang and G. Sexton, "A New Method for Pedestrian counting", *presented at Image Processing And Its Applications, 1995.*

[33] Stephen S. Intille, and Aaron F. Bobick, "Closed-World tracking", *in Proc. Int. Conf. Comp. Vision, 1995.*

[34] Jae-Won Kim, K.-S.C., Byeong-Doo Choi, and Sung-Jea Ko, "Real-time Vision-based People Counting System for the Security Door" , *in The 2002 International Technical Conference On Circuits/Systems, Computers and Communications (ITC-CSCC 200), 2002, Phuket, Thailand.*

[35] Rafael C. Gonzalez, Richard E. Woods, Digital image processing, *Addison-Wesley, Wokingham, 1992.*

[36] MATLAB: Image Processing toolbox User's Guide Version 2,*The MathWorks.Inc., 1998.*

[37] John C. Russ, The image processing handbook, *3rd ed. CRC Press, Boca Raton, Heidelberg, Springer-Verlag, 1999.*

[38] Pratt, William K., Digital image processing, *2nd ed., New York, N.Y., Wiley, 1991.*

[39] M. Sonka, V. Hlavac and R. Boyle, Image processing analysis and machine vision, *2nd ed., PWS Publishing, London, 1999.*

[40] MATLAB: Signal Processing toolbox User's Guide Version 4.2,*The MathWorks.Inc.,1999.*

[41] C. A. Steinberg, S. Abraham and C. A. Caceres, "Pattern Recognition in the clinical electrocardiogram", *IRE Trans. Bio-med. Electron, vol. BME-9, Jan 1962*, pp. 23-30.

[42] Tarassenko L., A Guide To Neural Computing Applications, *Arnold, London, 1998.*

[43] Bishop CM., Neural Network For Pattern Recognition, *Oxford University Press, Oxford, 1995.*

[44] R. Porter and N. Canagarajah, " Robust rotation-invariant texture classification: Wavelet, Gobor filter and GMRF based schemes", *in IEE Proc.-Vis. Image Signal Proces., June 1997,vol. 144, No. 3*, pp. 180-188.

[45] J. Zhang and S. Oe, " Texture Image Segmentation Method Based on Wavelet Transform and Neural Network ", *in IEEE Conference, 1998*, pp. 4595-4600.

[46] W. Gaohong, Zhang Y. and L. Xinggang, "Wavelet Transform-based Texture Classification with Feature Weighting", *in IEEE Conference, 1999*,pp. 435-439.

[47] M. Unser, "Texture Classification and Segmentation Using Wavelet Frames", *in IEEE Trans. Image Processing, November 1995,vol. 4, No.11*, pp. 1549-60.

[48] T Chang and C. –C. J Kuo, "A Wavelet Transform Approach to Texture", *in IEEE Conference, 1992*,pp. IV661-4.

[49] M. Kirby and L Sirovich, "Application of Karunen-Loeve Procedure for Characterization of Human face", *IEEE Trans. Pattern Anal. And Mach. Intell.,1990, vol. 12, No. 7*, pp. 103-108.

[50] J. Daugman," Face and Gesture recognition: Overview", *IEEE Trans. Pattern Anal. And Mach. Intell., 1997, vol. 19, No. 7,* pp. 675-76.

[51] P. C. Yuen, D. Q. Dai and G. C. Feng, " Wavelet-based PCA for Human Face Recognition", *in IEEE Conference, 1998,* pp.223-228.

[52] G. Van de Wouwer, "Wavelets for Texture Analysis*", PhD thesis, University of Antwerp, 1998.*

[53] James S. Walker, A Primer on Wavelets and their Scientific Applications, *Chapman& Hall/ CRC, 1999.*

[54] G Strang and T. Nguyen, Wavelets and Filter Banks, *Wellesley-Cambridge Press, USA, 1996.*

[55] Sergio J. Garcia Galan and Nuria Gonzalez Prelcic , " *Uvi_Wave* Wavelet toolbox user' guide", *University of Vigo, 1996.*

[56] M. Misiti, Y. Misiti, G Oppenheim and J. M. Poggi, MATLAB: Wavelet toolbox User's Guide, *The MathWorks.Inc., 1996.*

[57] C.S. Burrus, R.A. Gopinath and Haitao Guo, "Introduction to Wavelets and Wavelet Transforms", *Prentice Hall, Inc. New Jersey, 1998.*

[58] E. Aboufadel and S. Schlicker, Discovering wavelets, *John wiley & Sons ,INC, 1999.*

[59] G. Van de Wouwer, P. S. Cheunders and D. Van Dyck, " Statistical Texture characterization from Discrete Wavelet Representations", *IEEE Trans. on Image Processing, April 1999, vol. 8, No. 4,* pp. 592-598.

[60] M. Turk and A. Pentland, "Eigenfaces for Recognition", in *Journal of Cognitive Neuroscience, 1991,vol. 3, No. 1,* pp. 71-86.

[61] D. L. Swets and J. Weng,"Using discriminat eigenfeatures for image retrieval", in *IEEE Trans. on Pattern Anal. and Mach. Intell., 1996, vol. 18, No. 8,* pp. 831-836.

[62] A. O'Toole, H Abdi, K Deffenbacher and D. Valentin, "Low-dimensional representation of faces in higher dimension of the face space", in *J. Opt. Soc. Am. A., 1993,vol. 10, No. 3,* pp. 405-411.

[63] Ugur Halici and Guclu Ongun, "Fingerprint Classification Through Self-Organizing Feature Maps Modified to Treat Uncertainties", *Proceeding .of IEEE, October 1996,vol. 84, No. 10,* pp. 1497-1512.

[64] D.R. Hush and B. G. Horne, "Progress in supervised neural networks", *IEEE Signal Processing Mag., 1993,* pp. 8-39.

[65] T. Kohonen, Self-Organization and Associative Memory, *3rd ed., Berlin: Springer Series in Inform. Sci.,1989.*

[66] H Jiang and J Penman, "Using Kohonen Features Maps to Monitor the Condition of Synchronous Generators", in *IEEE, 1994,* pp. 89-94.

[67] T. Kohonen, Self-Organization Maps, *2rd ed., Berlin: Springer, 1997.*

[68] S. Lawrence, C. L. Giles, Ah Chung Tsoi and A. D. Back, "Face Recognition: A Convolutional Neural-Network Approach", *in IEEE Trans. on Neural Networks, 1997,vol.8, No. 1,* pp. 98-113.

[69] DTI, Neural Computing Learning solution: Best Practice Guidelines for Developing Neural Computing Applications, *DTI's Neural Computing Technology Transfer Programme, London, 1994.*

[70] S. Wolfram, Theory and Applications of Cellular Automata: Selected Papers, 1983-86 (Advanced Series in Complex Systems), *Singapore, World Scientific Publishing, 1986.*

[71] E. Goles, S. Martinez, Neural and Automata Networks: Dynamical Behavior and Applications, *Kluwer Academic Publishers, Dordrecht, 1990.*

[72] N. Takashi," Jamming transition induced by a stagnant street in traffic-flow model", *Physica A: Statistical and Theoretical Physics, September 1993, vol.198, No. 1-2,* pp. 108-116.

[73] K. Nagel and S. Rasmussen, "Traffic at the edge of chaos", *In: Artificial life IV, Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems, 1994*, pp. 222-225.

[74] A. Schadschneider, "Traffic flow: a statistic physics point of view", *Physica A: Statistical Mechanics and its Applications, 2002, vol.313, No. 1-2,* pp. 153-187.

[75] Blue V. J., Embrechts M. J. and Adler J.L, "Cellular automata modelling of pedestrian movements", *IEEE International Conference on Computational Cybernetics and Simulation, 1997, vol.3,* pp. 2320-23.

[76] Blue V. J. and Adler J.L, "Cellular automata microsimulation for modelling bi-directional pedestrian walkways", *Transportation Research Part B: Methodological, 2001, vol.35, No. 3,* pp. 293-312.

[77] Blue V. J. and Adler J.L, "Cellular Automata Model Of Emergent Collective Bi-Directional Pedestrian Dynamics", *Artificial Life VII, The Seventh International Conference on the Simulation and Synthesis of Living Systems, Reed College, Portland Oregon, 1-6 August 2000.*

[78] Blue V. J. and Adler J.L, "Modeling Four-Directional Pedestrian Movements", *Presented at the 79th Annual meeting of the Transportation Research Board, January 2000, and accepted for publication by Transportation Research Record, Journal of the Transportation Research Board.*

[79] Blue, V.J. and Adler, J.L, "Using Cellular Automata Microsimulation to Model Pedestrian Movements", *Proceedings of the 14th International Symposium on Transportation and Traffic Theory, A. Ceder ed. Elsevier Science Ltd., July 1999*, pp. 235-254.

[80] Fukui-M and Ishibashi-Y, "Jammimg transition in cellular automaton models for pedestrians on passageway", *In Journal of the Physical Society of Japan, November 1999, vol. 68, No. 11,* pp. 3738-9.

[81] Fukui-M and Ishibashi-Y, "Self-Organized Phase Transitions in Cellular Automaton Models for Pedestrians", *In Journal of the Physical Society of Japan, August 1999, vol. 68, No. 8,* pp. 2861-63.

[82] S. Wolfram, "Statistical mechanics of cellular automata", *Review of Modern Physics, July 1963, vol.55, No.3,* pp. 601-644.

[83] J. Von Neumann, "Theory of self-reproducing automata", *A.W. Burk Editor, University of Illinois Press, 1966.*

[84] N.H. Packard and S. Wolfram, "Two dimensional cellular automata", *in Journal of Statistical Physics, 1985, vol.38, No. 5/6.*

[85] V. Rao, H. Rao and V.B. Rao, C++ Neural Networks and Fuzzy logic, *2rd ed., Berlin: Springer, 1997.*

[86] I. A. Basheer and M. Hajmeer, "Artificial neural networks: fundamentals, computing, design, and application", *Journal of Microbiological Methods, 2000, vol.43,* pp. 3-31.

[87] M.T. Hagan, H. B. Demuth and M. Beale, Neural Network Design, *PWS Publishing Company,Boston, USA,1996.*

[88] C. Burstedde, K. Klauck, A. Schadschneider and J. Zittartz, "Simulation of pedestrian dynamics using a two-dimensional cellular automaton", *Physica A: Statistical Mechanics and its Applications, 2001, vol. 295, No. 3-4,* pp. 507-525.

Appendixes

Appendix A: Least-square Parabolic Smoothing and Differentiation

To find the number of points over which a least-square parabola can be fitted to provide the best compromise between smoothing and signal degradation, it was necessary to develop a general expression for the value of the smoothed data in terms of the number of sample points.

The equation of such a parabola is

$$Y = a_0 + a_1 x + a_2 x^2, \qquad \text{(A-1)}$$

where $\boldsymbol{Y}$ is the value of the parabola at sample number $\boldsymbol{x}$ and $\boldsymbol{a_0}$, $\boldsymbol{a_1}$ and a_2 are constants.
If $X = 0$ is defined as the location of the parabola at any sample point $\boldsymbol{k}$ is

$$Y_k^1 = a_0 + ka_1 + k^2 a_2, \qquad \text{(A-2)}$$

where

$$-\left(\frac{(n-1)}{2}\right) \leq k \leq \left(\frac{(n-1)}{2}\right)$$

The difference between the value of the parabola ($\boldsymbol{e_k}$) and the data sample ($\boldsymbol{Y_k}$) is

$$e_k = Y_k^1 - Y_k$$
$$e_k = a_0 + ka_1 + k^2 a_2 - Y_k. \qquad \text{(A-3)}$$

The total mean square difference is then

$$\sum_{k=-(n-1)/2}^{k=(n-1)/2} e_k^{\;2} = \sum_{k=-(n-1)/2}^{k=(n-1)/2} (a_0 + ka_1 + k^2 a2 - y_k)^2, \qquad \text{(A-4)}$$

where $\boldsymbol{n}$ is the odd-numbered point

To minimize the mean square difference, (A-4) is differentiated with respect to ***a₀*** →

To minimize the mean square difference, (A-4) is differentiated with respect to $\boldsymbol{a_0}$, $\boldsymbol{a_1}$ and a_2, and the resultant three derivatives are set to zero. These equations are then solved simultaneously to obtain the values of $\boldsymbol{a_0}$, $\boldsymbol{a_1}$ and a_2, which will determine the parabola that provides a least mean-square fit with the n samples of the data.

This process yields the following three sets of values for $\boldsymbol{a_0}$, $\boldsymbol{a_1}$ and $\boldsymbol{a_2}$, for the least-square parabola:

$$a_0 = \frac{3}{4n(n^2-4)} \times \sum_{k=-(n-1)/2}^{k=(n-1)/2} (3n^2 - 7 - 20k^2) y_k$$

$$a_1 = \frac{12}{n(n^2-1)} \times \sum_{k=-(n-1)/2}^{k=(n-1)/2} k y_k$$

$$a_2 = \frac{180}{n(n^2-1)(n^2-4)} + \sum_{k=-(n-1)/2}^{k=(n-1)/2} k^2 - (n^2-1)/12 y_k^{\;2}$$

The value of the smoothed data is the value of the parabola at $X = 0$, or merely $\boldsymbol{a_0}$. Therefore the set of $\boldsymbol{a_0}$ is applied to perform as the smoothing method.

A prediction technique using the association of Cellular Automata and a Neural Network

C Suppitaksakul[1,2], G Sexton[2] and P D Minns[2]
[1]Department of Electrical Engineering, Faculty of Engineering
Rajamangala University of Technology Thanyaburi, Pathumthani, 12110, Thailand
Phone 0-2549-3420-9, Fax 0-2549-3422, E-mail: s.chai@unn.ac.uk
[2]Department of Electrical and Electronic Engineering, School of Engineering
Northumbria University, Newcastle, UK
Phone: 44-191227-3232, E-mail: g.sexton@unn.ac.uk, pminns@unn.ac.uk

Abstract

This paper describes a new movement prediction technique for use in conjunction with a pedestrian tracking system. The proposed prediction technique aims to learn the behavior of pedestrian movement directly from images, it is based on an association of Cellular Automata (CA) and a Neural Network (NN). A CA is applied to capture the pedestrian movement. The derived CA patterns are then used to train the NN. The trained NN is used for simulating pedestrian movement, and then used to estimate the pedestrian future position. The results indicate that the effectiveness of the technique depends on the following factors: the number of pedestrians and the movement patterns. Applications of the technique on other objects apart from pedestrians are introduced.

Keywords: Prediction, Pedestrian movement behavior, Cellular Automata, Neural Network.

1. Introduction

In the last decade, the use of computer vision for *"human motion analysis"* [1] has grown exponentially, because of its promising applications such as visual surveillance, perceptual user interface, motion analysis and model-based coding. This attracts many researchers in exploring new techniques for the application of human motion analysis, which involves the detection, tracking and recognition of consecutive human images. One of the main areas explored so far is the technique used for tracking human images. Prediction techniques have been used to support human tracking in various manners and purposes. Moeslund and Granum [2] employed prediction using a model of velocity and acceleration to estimate the 3D pose of a human arm. Rohr [3] applied a Kalman filter, which is used for estimating the current movement state of 3D positions and posters, in order to predict the human pose in consecutive frames. Rossi and Bozzoli [4] used a simple trajectory model to provide a reasonable estimate of the regions of interest for tracking people movement in consecutive frames.

This paper presents an alternative prediction method, which like Rossi and Bozzoli [4] is used to support the tracking of human movements. However, in the present study rather than using the simple trajectory model, the association of CA and NN is used instead. The proposed prediction algorithm can be split into two parts: the movement behavior captured by using a CA and the use of a NN for CA pattern learning and simulating. In Section 2, the principle of CA and NN are briefly introduced. An association of CA and NN and the application of a CA to capture pedestrian movement and the NN to learn the CA patterns are explained in Section 3. In Section 4, experiments which were carried out on actual consecutive images are demonstrated. Finally in Section 5, the conclusions regarding the proposed technique are reported.

2. The principles of the CA and the NN

In this section, the principles of CA and NN are briefly explained in order to provide an overview of these two tools.

2.1 Cellular Automata

Cellular Automata (CA) are mathematical idealizations of physical systems in which space and time are discrete, and physical quantities take on a finite set of discrete values. A cellular automaton consists of a regular uniform lattice or array, usually infinite in extent with a discrete variable at each site or cell. The state of a cellular automaton is completely specified by the values of the variables at each site. A cellular automaton evolves in discrete time steps, with the value of the variable at one site being affected by the values of variables at sites in its neighborhood on the previous time step. The neighborhood of a site is typically taken to be the site itself and all immediately adjacent sites. The variables at each site are updated simultaneously, based on the values of the variables in their neighborhoods at the preceding time step, and according to a definite set of *"local rules"*[5].

CA were first introduced by Von Neumann in the early 1950s. His CA has computing capabilities of the Universal Turbine Machine [6]. Since then, many other studies on CA as abstract models of computation have been made and cover a large scope of applications such as physics, image processing, and computer science [5]. In recent years, CA microsimulation has been successfully applied to model vehicular flows and traffic networks [7,8] then extended to model pedestrian flow [9-12].

2.1.1 One-dimensional Cellular Automaton

The simplest model is a one-dimensional cellular automaton (1D-CA). It consists of a line each site carries values, which are updated in parallel. The neighborhood of a site is typically taken to be the two adjacent sites. This characteristic is known as a spatially local rule. The relationship between sites is depicted in Fig. 1.

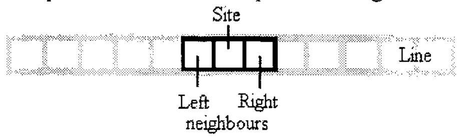


Fig. 1: A cell and its neighbors of 1D-CA

DS019

The evaluation of the sites is controlled by a definite set of rules, depending on the values for a fixed number of preceding steps. Usually, the preceding step is 1. This is the temporally local rule. Such very simple local evaluation rules can produce very complex even chaotic behavior for the overall system. An example of two-state (0,1) 1D-CA simulation is shown in Fig. 2. The used CA rules in the simulation are set as following: the new state of the cell will be white (0) if all three cells are white or black (1). Otherwise, the new state of the cell will be black (1). The new state and the CA rules can also be expressed as:

$$\textit{New state} = \begin{cases} 0 \;\; \textit{if} \;\; x = [0 \;\; 0 \;\; 0] \;\textit{or}\; [1 \;\; 1 \;\; 1] \\ 1 \;\; \textit{if} \;\; x = [1 \;\; 1 \;\; 0], [1 \;\; 0 \;\; 0], \\ \qquad\quad [1 \;\; 0 \;\; 1], [0 \;\; 1 \;\; 0], \\ \qquad\quad [0 \;\; 1 \;\; 1], [0 \;\; 0 \;\; 1] \end{cases} \qquad (1)$$

Where x is a set of three cells

Over time, the cells change from state to state by using CA rules in (1). The pattern obtained from a simulation for 20 steps is illustrated in Fig. 3. As the results from the simulation, the self-reproduce the patterns are more complex if the step of simulation is increased.

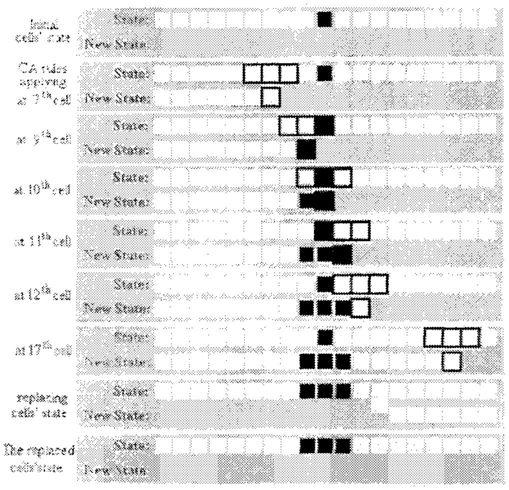


Fig. 2: One-step of applying CA rules to two-state 1D-CA

2.1.2 Two-dimensional Cellular Automata

The basic principle is identical to a one-dimensional cellular automaton. The main differences reside in the neighborhood. There are several possible lattices and neighborhood structures for two-dimensional cellular automata (2D-CA) [13]. The first one, defined by Von Neumann [14], is the square cellular automaton, as illustrated in Fig. 4 *a*. A second important two-dimensional cellular automaton is known as the Moore neighborhood cellular automaton depicted on Fig.4 *b*.

In the two cases shown, each cell is connected to either the four or the eight neighbors around it, respectively. Any function on the neighborhood can then be evaluated accordingly. One of the simplest computations allowed by such an arrangement is a sum of the cell values and has been widely described in cellular automaton theory [5,13]. This study is interested in the 2D-CA, which will be used to capture pedestrian movement. The adaptation of the 2D-CA to our application is described in Section 3.

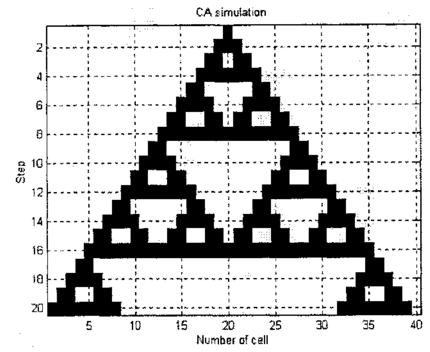


Fig. 3: The simulation of 20 steps of two-state 1D-CA

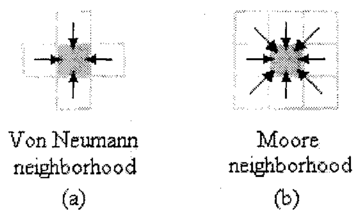


Fig.4: Two-dimensional cellular automata

2.2 Neural Network

A Neural Network (NN) is a computational structure inspired by the study of biological neural network processing. In this study, a Backpropagation Neural Network (BPNN) is used because it is suitable for our application.

The BPNN is a feed-forward multilayer perceptron (MLP) consisting of (1) an input layer with nodes representing input variables to the problem, (2) an output layer with nodes representing the dependent variables, and (3) one or more hidden layers containing nodes to help capture the non-linearity in the data. Using supervised learning with the error-correction learning rule (ECL), these networks can learn the mapping from one data space to another using examples. The term backpropagation refers to the way the error computed at the output side is propagated backward from the output layer, to the hidden layer, and finally to the input layer. In the BPNN, the data are fed forward into the network without feedback. The neurons in the BPNN can be fully or partially interconnected [15]. The architecture of the BPNN is illustrated in Fig. 5.

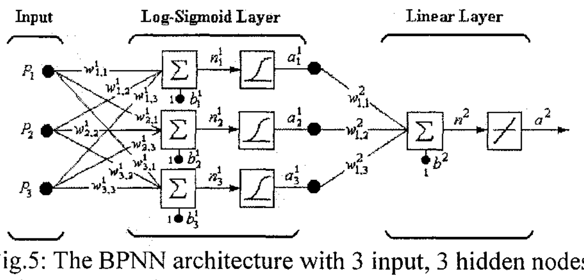


Fig.5: The BPNN architecture with 3 input, 3 hidden nodes and 1 output node

3. The association of CA and BPNN

The principles of the CA and the BPNN have been explained in the previous section. This section describes how the CA is associated with the BPNN to provide a prediction technique. According to the principle of a CA, the evaluation of a site (in 1D-CA and 2D-CA) is controlled by a set of CA rules that can be regarded as patterns in terms of vectors. These patterns could be used to train BPNN. Therefore, it is conjectured that after the BPNN is trained, it should behave the same as the CA models. If the conjecture is true, capturing of pedestrian behaviors from real images is feasible and the defining of CA rules from empirical data can be avoided.

A grid is required to be superimposed on the image in order to applying the CA, to capture the pedestrian dynamic that is generically two-dimensional in nature [12] therefore the 2D-CA is used. Each grid size depends on the people dimension that turn up in the image; however, the grid should be big enough to contain most of the people dimension as illustrated in Fig. 6. While the pedestrian appears in the image, the grid is established and a CA pattern, which contains the site and its neighborhood, is captured. Then it is employed as the BPNN input. The pedestrian position in the next few frames, which depends on the speed of the pedestrian, is used as the target for the BPNN to provide its training. After training the BPNN is used for pedestrian dynamic simulating, instead of using the CA rule set. The proposed prediction technique block diagram is illustrated in Fig. 9. Experiments that were carried out to prove the conjecture are demonstrated in Section 4.

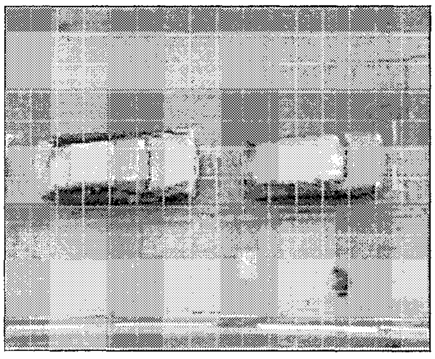


Fig.6: Latticed image

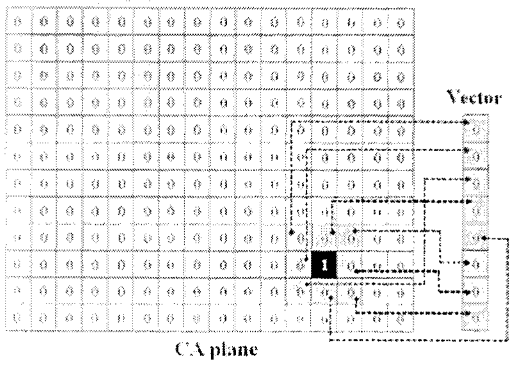


Fig.7: A site with Moore neighborhood

4. Experiments

In order to prove our conjecture, the experiment is carried out on the BPNN simulation and compared to a CA simulation. Firstly, we apply the CA patterns of (1) for training BPNN. The set of three cells, *x*, is fed through BPNN as inputs (3 inputs nodes) and the new state, 0 or 1 (1 output node), is the target of BPNN. For the number of hidden nodes, we begin with 3 nodes and increase 1 node each time of new training in case that training fails. The number of hidden nodes is increased in order to achieve the goal of training. The result of the BPNN simulation provides the same outcome as the CA simulation showed in Fig. 3. More experiments were carried out using different CA rules in the simulation then extended to 2D-CA. In the experiment of 2D-CA, the number of BPNN inputs and CA patterns are increased according to Moore neighborhood as shown in Fig. 7and Fig. 8 illustrated the pedestrian movement simulation on the CA plane. The BPNN and CA produced the same outputs, showing that the BPNN was able to follow the CA.

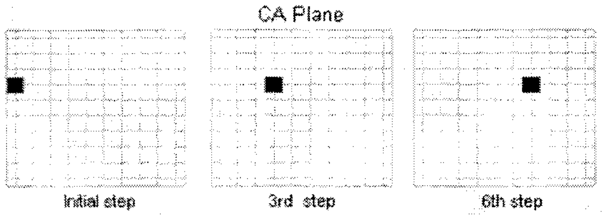


Fig. 8: The pedestrian movement simulation 6 steps of two-state 2D-CA

As the result of the CA simulation offer proof that the BPNN can be used instead of the CA simulation, the test is extended to a prediction experiment. Actual consecutive images were used for testing. The prediction requires the pedestrian location and direction in the simulation; the initial pedestrian location and direction were manually set. 2D-CA with Moore neighborhood, shown in Fig. 4(b), was applied for the test. The speed of pedestrian movement was assumed to be constant at 6 frames per cell. As a result, it is can be seen that the pedestrian position in the next several frames can be estimated and tracked. The outcomes of the prediction process are shown in Fig. 10.

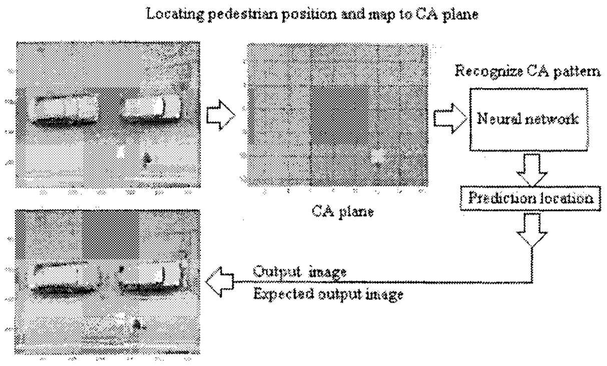


Fig.9: The block diagram of the prediction technique

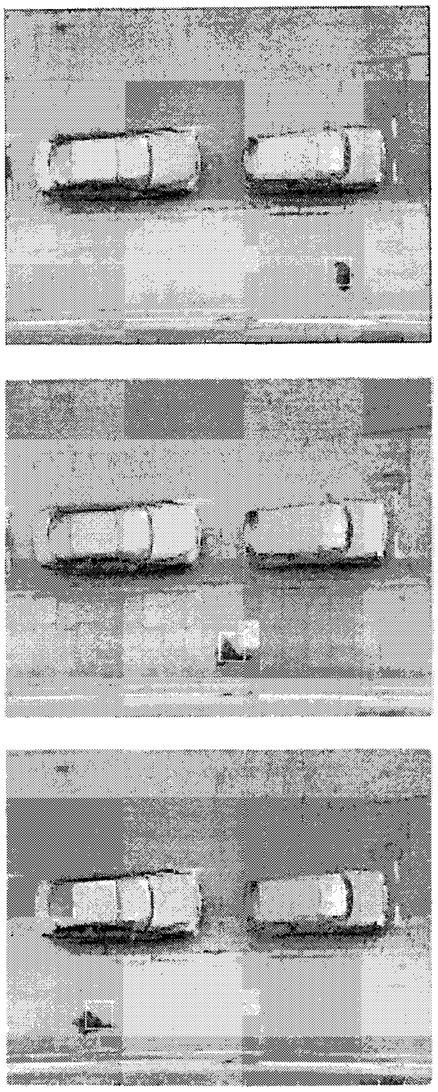


Fig.10: The images of the prediction outcomes

5. Conclusion and discussion

The concepts of a new prediction technique have been presented. The co-operation of CA and BPNN is applied to actual images to support tracking used for classifying the input CA pattern then employed for simulation. In fact the presented technique is required to use with a detection method in order to provided an automated system for pedestrian tracking.

However, the results indicate the effectiveness of the technique depends on the following factors: the number of pedestrians and the movement patterns that concern probability, speed and direction of pedestrian movement. These factors are topics for further investigation and studies in order to improve the effectiveness of our prediction method. Besides, the automated pedestrian tracking with the proposed method is also focused for the further study.

The prediction technique could be extended to support tracking systems for other applications that the CA can be applied such as the vehicles tracking, path planning for robot movement etc.

References

[1] D.M. Gavrila, "The Visual Analysis of Human Movement: A Survey", in Computer Vision and Image Understanding, Vol. 73, No. 1, January 1999, pp. 82-98.

[2] T. B. Moeslund and E. Granum, "Multiple cue used in model based motion capture", in The fourth International Conference on Automatic Face and Gesture Recognition, Grenoble, France, March 2000.

[3] K. Rohr, " Human movment Analysis Based on Explicit Motion Models, chap. 8, pp. 171-198, Kluwer Academic, Dordrecht/Boston, 1997.

[4] M. Rossi and A. Bozzoli, "Tracking and counting moving people," presented at Proceedings. ICIP-94, IEEE International Conference on Image Processing, 1994.

[5] S. Wolfram, Theory and Applications of Cellular Automata: Selected Papers, 1983-86 (Advanced Series in Complex Systems), *Singapore, World Scientific Publishing, 1986.*

[6] E. Goles, S. Martinez, Neural and Automata Networks: Dynamical Behavior and Applications, Kluwer Academic Publishers, Dordrecht, 1990.

[7] K. Nagel and S. Rasmussen, "Traffic at the edge of chaos", In: Artificial life IV, Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems, 1994, pp. 222-225.

[8] A. Schadschneider, "Traffic flow: a statistic physics point of view", Physica A: Statistical Mechanics and its Applications, 2002, Vol.313, No. 1-2, pp. 153-187.

[9] Blue V. J., Embrechts M. J. and Adler J.L, "Cellular automata modelling of pedestrian movements", *IEEE International Conference on Computational Cybernetics and Simulation, 1997, Vol.3,* pp. 2320-23.

[10] Fukui-M and Ishibashi-Y, "Jammimg transition in cellular automaton models for pedestrians on passageway", In Journal of the Physical Society of Japan, November 1999, Vol. 68, No. 11, pp. 3738-9.

[11] Fukui-M and Ishibashi-Y, "Self-Organized Phase Transitions in Cellular Automaton Models for Pedestrians", In Journal of the Physical Society of Japan, August 1999, Vol. 68, No. 8, pp. 2861-63.

[12] C. Burstedde, K. Klauck, A. Schadschneider and J. Zittartz, "Simulation of pedestrian dynamics using a two-dimensional cellular automaton", Physica A: Statistical Mechanics and its Applications, 2001, Vol. 295, No. 3-4, pp. 507-525.

[13] S. Wolfram, "Statistical mechanics of cellular automata", Review of Modern Physics, July 1963, Vol.55, No.3, pp. 601-644.

[14] J. Von Neumann, "Theory of self-reproducing automata", A.W. Burk Editor,University of Illinois Press, 1966

[15] I.A. Basheer and M. Hajmeer, "Artificial neural networks: fundamentals, computing, design, and application", Journal of Microbiological Methods, 2000, Vol.43, pp. 3-31.

A Pedestrian Tracking Using the Association of Cellular Automata and a Neural Network

C. Suppitaksakul, G. Sexton and P. D. Minns

Northumbria Communication Research Laboratory (NCRL)
School of Computing, Engineering & Information Sciences
Northumbria University, Newcastle upon Tyne, NE1 8ST, UK
E-mail: s.chai@unn.ac.uk

Abstract — **This paper describes a pedestrian tracking system that uses a new movement prediction technique based on an association of Cellular Automata (CA) and a Neural Network (NN). The proposed prediction technique aims to learn the behaviour of pedestrian movement directly from images. CA are applied to capture the pedestrian movement. The derived CA patterns are then used to train the NN. The trained NN is employed to simulate pedestrian movement, and then to estimate the pedestrian future position. The results indicate that the effectiveness of the technique depends on the following factors: the speed of pedestrians and the movement patterns. Other applications of the technique are also mentioned.**

Keywords: Prediction, Pedestrian, Cellular Automata, Neural Network.

I. Introduction

Over the last decade, the use of computer vision for "*human motion analysis*" [1] has increased exponentially because of its promising applications such as perceptual user interface, visual surveillance, motion analysis and model-based coding. This attracts many researchers in exploring new techniques for the application of human motion analysis, which involves the detection, tracking and recognition of consecutive human images. One of the main areas explored so far is the technique used for tracking human images. Prediction techniques have been used to support human tracking in various manners and for varying purposes. Rohr [2] applied a Kalman filter, which is used for estimating the current movement state of 3D positions and postures, in order to predict the human pose in consecutive frames. Moeslund and Granum [3] performed prediction using a model of velocity and acceleration to estimate the 3D pose of a human arm. Rossi and Bozzoli [4] used a simple trajectory model to provide a reasonable estimate of the regions of interest for tracking people movement in consecutive frames.

This paper presents an alternative prediction method, which like Rossi and Bozzoli [4], is used to support the tracking of human movement. However, in the present study rather than using the simple trajectory model, the association of CA and NN is applied. The proposed prediction algorithm can be split into two parts: the movement behaviour captured using CA and a NN to learn and simulate the CA pattern. In Sections II and III, the principle of CA and NN are briefly detailed. The association of CA and NN and the application of CA to capture pedestrian movement and the NN to learn the CA patterns are explained in Section IV. In Section V, experiments which were carried out on actual consecutive images are demonstrated. Finally in Section VI, the conclusions regarding the proposed technique are reported.

II. The Cellular Automata

Cellular Automata (CA) are mathematical idealizations of physical systems in which space and time are discrete, and physical quantities take on a finite set of discrete values. A cellular automaton consists of a regular uniform lattice or array, usually infinite in extent with a discrete variable at each site or cell. The state of a cellular automaton is completely specified by the values of the variables at each site. A cellular automaton evolves in discrete time steps, with the value of the variable at one site being affected by the values of variables at sites in its neighbourhood on the previous time step. The neighbourhood of a site is typically taken to be the site itself and all immediately adjacent sites. The variables at each site are updated simultaneously and according to a definite set of *"local rules"*[5]. CA were first introduced by Von Neumann in the early 1950s. His CA has the computing capabilities of the Universal Turing Machine [6]. Since then, many other studies on CA as abstract models of computation have been made and cover a large scope of applications such as physics, image processing, and computer science [5]. In recent years, CA microsimulation has been successfully applied to model vehicular flows and traffic networks [7,8] then extended to model pedestrian flow [9-11].

A. One-dimensional Cellular Automaton

The simplest model is a one-dimensional cellular automaton (1D-CA). It consists of a line of sites each carrying values that are updated in parallel. The neighbourhood of a site is typically taken to be the two adjacent sites. This characteristic is known as a spatially local rule. The relationship between sites is depicted in Fig. 1. The evaluation of the sites is controlled by a definite set of rules, depending on the values for a fixed number of preceding steps, usually taken as 1. This is the temporally local rule. Such very simple local evaluation rules can produce very complex even chaotic behaviour for the overall system. An example pattern obtained from a two-state (0,1) 1D-CA simulation is shown in Fig. 2.

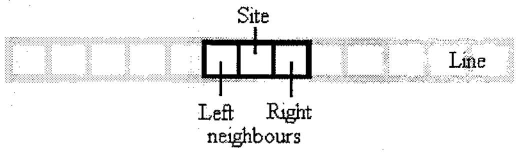


Fig. 1. A cell and its neighbours of 1D-CA

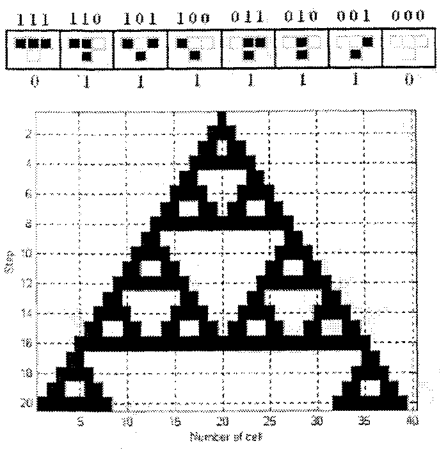


Fig. 2. The simulation of 20 steps of a two-state 1D-CA

The CA rules applied in the simulation are: the new state of the cell will be white (0) if all three cells are white or black (1). Otherwise, the new state of the cell will be black (1). The new state and the CA rules can also be expressed as (1):

$$\textit{New state} = \begin{cases} 0 \ \textit{if} \ x=[0\ \ 0\ \ 0] \ \textit{or} \ [1\ \ 1\ \ 1] \\ 1 \ \textit{if} \ x=[1\ \ 1\ \ 0],[1\ \ 0\ \ 0], \\ \qquad [1\ \ 0\ \ 1],[0\ \ 1\ \ 0], \\ \qquad [0\ \ 1\ \ 1],[0\ \ 0\ \ 1] \end{cases} \qquad (1)$$

Where x is a set of three cells

Over time, the cells change from state to state using the CA rules in (1). As in the results of the simulation, the self-reproducing pattern becomes more complex as the number of steps in the simulation is increased.

B. Two-dimensional Cellular Automata

The basic principle is identical to a one-dimensional cellular automaton. The main differences reside in the neighbourhood shape. There are several possible lattices and neighbourhood structures for two-dimensional cellular automata (2D-CA) [12]. The first one, defined by Von Neumann [13], is the square cellular automaton, as illustrated in Fig. 3*a*. A second important two-dimensional cellular automaton is known as the Moore neighbourhood cellular automaton, depicted in Fig. 3*b*. In the two cases shown, each cell is connected to either four or eight neighbours. Any function on the neighbourhood can then be evaluated accordingly. One of the simplest computations allowed by such an arrangement is a sum of the cell values and has been widely described in cellular automaton theory [5,12]. This study is interested in the 2D-CA, which will be used to capture pedestrian movement. The adaptation of the 2D-CA to our application is described in Section IV.

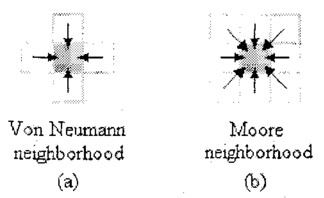


Fig. 3. Two-dimensional cellular automata

III. Neural Network

A Neural Network (NN) is a computational structure inspired by the study of biological neural network processing. Supervised training of the NN is required in this application, however since the number of input patterns is not significant then the training speed of the network is not a major concern. The use of a feed-forward multiplayer perceptron (MLP) with an error back-propagation learning algorithm is therefore deemed to be sufficient in this application. The MLP consisting of (1) an input layer with nodes representing input variables to the problem, (2) an output layer with nodes representing the dependent variables, and (3) one or more hidden layers containing nodes to help capture the non-linearity in the data. Using supervised learning with the error-correction learning rule (ECL), these networks can learn the mapping from one data space to another using examples. The term back-propagation refers to the way the error computed at the output side is propagated backward from the output layer, to the hidden layer, and finally to the input layer. In the MLP, the data are fed forward into the network without feedback. The neurons in the MLP can be fully or partially interconnected [14].

IV. The Association of CA and MLP

This section describes how the CA is associated with the MLP to provide a prediction technique. A block diagram of the system is illustrated in Fig. 4. According to the principle of a CA, the evaluation of a site (in 1D-CA and 2D-CA) is controlled by a set of CA rules that can be regarded as patterns in terms of vectors. These patterns could be used to train MLP. Therefore, it is conjectured that after the MLP is trained, it should behave the same as the CA models. If the conjecture is true, the capture of a pedestrian's behaviour from real images is feasible and the definition of CA rules from empirical data can be avoided.

A grid is required to be superimposed on the image in order to applying the CA, to capture the pedestrian dynamic that is generically two-dimensional in nature [11] and therefore the 2D-CA is employed. Each grid size depends on the dimension of the person in the image. The grid should be big enough to contain most of the person, as illustrated in Fig. 5. While the pedestrian appears in the image, the grid is established and a CA pattern, which contains the site and its neighbourhood, is captured. This is then employed as the MLP input. The pedestrian position in the next few frames, which depends on the speed of the pedestrian, is used as the target for the MLP to provide its training. After training, the MLP is used for pedestrian dynamic simulating, instead of using the CA rule set. This offers three advantages: (i) the behaviour of pedestrian movement can be captured and learned, which makes the model independent from the use of empirical data, (ii) The application of the CA rules to every cell on the CA plane is avoided and (iii) it makes the model more flexible allowing it to be applied in other applications for which the CA algorithm is suitable. Experiments that were carried out to investigate the conjecture and are demonstrated in Section V.

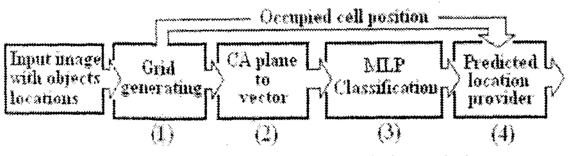


Fig. 4. The block diagram of the prediction technique

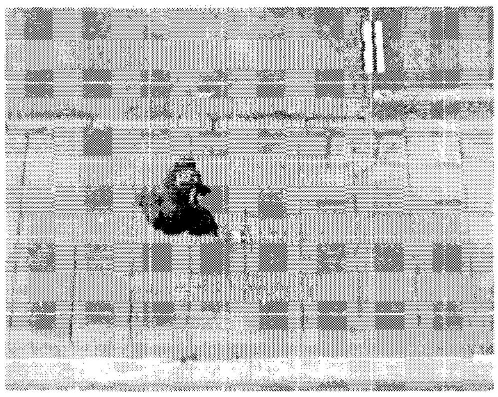


Fig. 5. An obtained latticed image from the grid generating

V. EXPERIMENTS

The experiment is carried out on the MLP simulation and compared to a CA simulation in order to prove our conjecture. A MLP with log-sigmoid transfer function in the hidden layer, linear transfer function in the output layer and a gradient decent algorithm is employed in the NN. Firstly, the MLP with is trained using the CA patterns of (1). Sets of three cells, x, are fed into the MLP, which has 3 inputs nodes. The MLP then evolves itself by correctly weighting the nodes in the hidden layer to achieve the desired output (1 output node) for the given input. The initial number of nodes in the hidden layer is three and is increased by 1 each time the desired output is not attained. As shown by the training results in Table I, it is found that the MLP operates as per the CA when the number of hidden nodes is twice that of the number of input nodes. Multiple experiments were performed using different CA rules and the results obtained supported the initial conjecture. The experiments were then extended to 2D-CA. Simulation of the 2D-CA required the number of MLP inputs and CA patterns to be increased according to the neighbourhood used. The simulation of a pedestrian's movement on the CA plane is illustrated in Fig. 7. The MLP and CA produced the same outputs, showing that the MLP was able to follow the CA.

TABLE I
TRAINING RESULTS

Number of hidden neurons	Training results of 10 times		Percent of successful training (%)
	Successful	Unsuccessful	
3	1	9	10
4	8	2	80
5	8	2	80
6	9	1	90
7	10	0	100
8	10	0	100
9	10	0	100
10	10	0	100

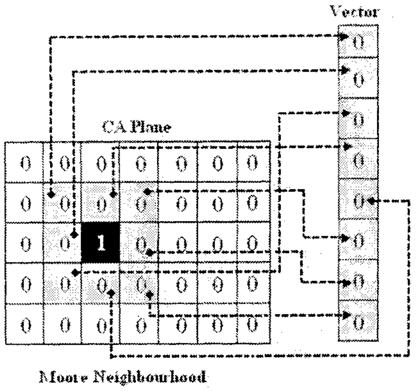


Fig. 6. The reorganising of Moore neighbourhood for MLP inputs

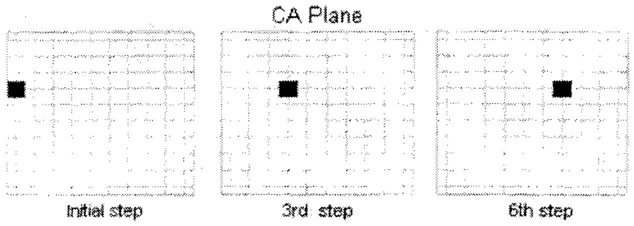


Fig. 7. The pedestrian movement simulation 6 steps of two-state 2D-CA

This shows that the MLP can be used instead of the CA simulation. Finally the experiment is extended by using actual image data. The prediction technique requires the pedestrian's initial location and direction, which in this case have been manually set. 2D-CA with Von Neumann neighbourhood, shown in Fig. 3(a), was applied. The speed of pedestrian movement was assumed to be constant at 10 frames per cell obtained by inspection. The test image with the initialisation and the simulation results are illustrated in Figs. 8 and 9, respectively. As a result, it is can be seen that the pedestrian position in the next several frames can be estimated and tracked to a satisfactory degree. The error observed occurs as a result of the grid applied to divide into cells the image. The grid is established around the pedestrian's initial location and is constant until they leave the image. This could be improved by regenerating the grid as the pedestrian moves from cell to cell. Another source of error arises from the speed at which the pedestrian moves, because the camera lens used to capture the images is convex.

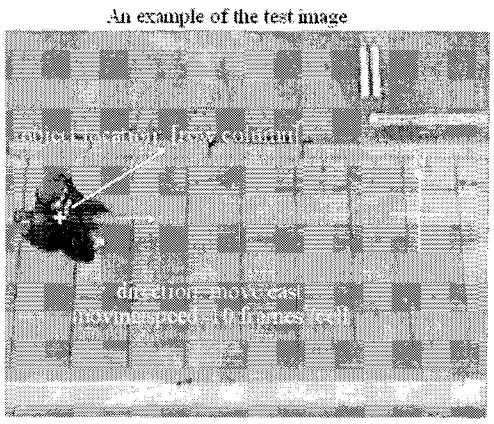

An example of the test image
object location (row, column)
direction: move east
moving speed: 10 frames/cell

Fig. 8. The test image with manually set for location and direction

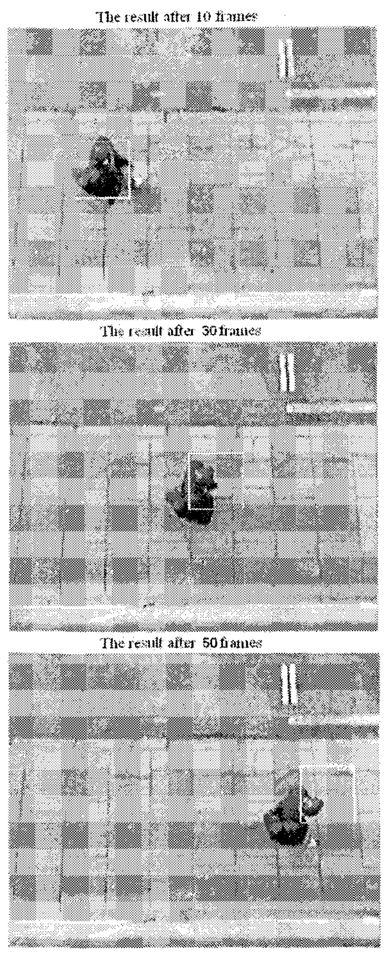

The result after 10 frames
The result after 30 frames
The result after 50 frames

Fig. 9. The tracking results of the prediction method

The results obtained by performing the grid regeneration at each cell are shown in Fig. 10. It is seen that the tracking of the pedestrian's movement is improved with the cell located more accurately upon the object than in the previous test. The technique is able to predict up to 50 frames ahead when the average walking speed is set at 10 frames per cell. However, the speed per cell will vary as the size of the grid changes.

VI. Conclusion and Discussion

The concepts of a new prediction technique have been presented. The co-operation of CA and MLP has been applied to actual images to support tracking used for classifying the input CA pattern then employed for simulation. The presented technique requires use of an object detection method that can supply the initial location and direction of the pedestrian in order to provide an automated system for pedestrian tracking.

It has been shown that the position of the pedestrian can be accurately predicted up to 50 frames ahead. Errors attained in excess of 50 frames may be in part due to the convex shape of the camera lens. Since the experiment was performed for only a single pedestrian in the sequence of images, the motion dynamic was not chaotic as has been shown in the demonstration of the CA. However, if the number of pedestrian in the image sequence increases the cells in the CA plane become occupied, which would result in chaotic behavior.

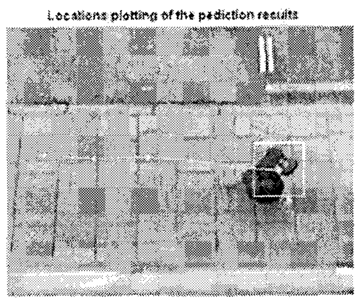


Fig. 10. The tracking results of the improved prediction method

The results indicate that the effectiveness of the technique depends upon the following factors: the movement patterns that concern probability, speed and direction of pedestrian movement. These factors are topics for further investigation. The number of pedestrians is another factor requiring further discussion, as the investigation presented here looks at the system performance for when only a single pedestrian is present in the image. Further more the development of a suitable detection method is required in order that a fully automated pedestrian tracking system based on the prediction technique can be achieved.

The prediction technique could be extended to support tracking systems for other applications for which the CA can be applied such as vehicle tracking or path planning for robot movement.

References

[1] D.M. Gavrila, "The visual analysis of human movement: A survey", *Computer Vision and Image Understanding*, vol. 73, no. 1, pp. 82-98, January 1999.

[2] K. Rohr, *Human movement Analysis Based on Explicit Motion Models*, Dordrecht/Boston: Kluwer Academic, 1997.

[3] T. B. Moeslund and E. Granum, "Multiple cue used in model based motion capture",*The fourth International Conference on Automatic Face and Gesture Recognition*, Grenoble, France, March 2000.

[4] M. Rossi and A. Bozzoli, "Tracking and counting moving people," *Proceedings ICIP-94, IEEE International Conference on Image Processing*, 1994.

[5] S. Wolfram, *Theory and Applications of Cellular Automata: Selected Papers, 1983-86 (Advanced Series in Complex Systems)*, Singapore: World Scientific Publishing, 1986.

[6] E. Goles, S. Martinez, *Neural and Automata Networks: Dynamical Behaviour and Applications*, Dordrecht: Kluwer Academic, 1990.

[7] K. Nagel and S. Rasmussen, "Traffic at the edge of chaos", *In: Artificial life IV, Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, pp. 222-225, 1994.

[8] A. Schadschneider, "Traffic flow: a statistic physics point of view", *Physica A: Statistical Mechanics and its Applications*, vol. 313, no. 1-2, pp. 153-187, 2002.

[9] Blue V. J., Embrechts M. J. and Adler J.L, "Cellular automata modelling of pedestrian movements", *IEEE International Conference on Computational Cybernetics and Simulation*, vol. 3, pp. 2320-23, 1997.

[10] Fukui M. and Ishibashi Y., "Self-Organized Phase Transitions in Cellular Automaton Models for Pedestrians", *Journal of the Physical Society of Japan*, vol. 68, no. 8, pp. 2861-63, August 1999.

[11] C. Burstedde, K. Klauck, A. Schadschneider and J. Zittartz, "Simulation of pedestrian dynamics using a two-dimensional cellular automaton", *Physica A: Statistical Mechanics and its Applications*, vol. 295, no. 3-4, pp. 507-525, 2001.

[12] S. Wolfram, "Statistical mechanics of cellular automata", *Review of Modern Physics*, vol. 55, no. 3, pp. 601-644, July 1963.

[13] J. Von Neumann, *Theory of self-reproducing automata*, A.W. Burk Editor, University of Illinois Press, 1966.

[14] I. A. Basheer and M. Hajmeer, "Artificial neural networks: fundamentals, computing, design, and application", *Journal of Microbiological Methods*, vol. 43, pp. 3-31, 2000.