Particle Swarm Optimisation for Edge Detection in Noisy Images

by

Mahdi Setayesh

A thesis
submitted to the Victoria University of Wellington
in fulfilment of the
requirements for the degree of
Doctor of Philosophy
in Computer Science.

Victoria University of Wellington
2013

Abstract

Detection of continuous and connected edges is very important in many applications, such as detecting oil slicks in remote sensing and detecting cancers in medical images. The detection of such edges is a hard problem particularly in noisy images and most edge detection algorithms suffer from producing broken and thick edges in such images. The main goal of this thesis is to reduce broken edges by proposing an optimisation model and a solution method in order to detect edges in noisy images.

This thesis suggests a new approach in the framework of particle swarm optimisation (PSO) to overcome noise and reduce broken edges through exploring a large area and extracting the global structure of the edges. A fitness function is developed based on the possibility score of a curve being fitted on an edge and the curvature cost of the curve with two constraints. Unlike traditional algorithms, the new method can detect edges with greater continuity in noisy images. Furthermore, a new truncation method within PSO is proposed to truncate the real values of particle positions to integers in order to increase the diversity of the particles.

This thesis also proposes a local thresholding technique for the PSO-based edge detection algorithm to overcome the problem of detection of edges in noisy images with illuminated areas. The local thresholding technique is proposed based on the main idea of the Sauvola-Pietkinen method which is a way of binarisation of illuminated images. It is observed that the new local thresholding can improve the performance of the PSO-based edge detectors in the illuminated noisy images.

Since the performance of using static topologies in various applications and in various versions of PSO is different , the performance of six different static topologies (fully connected, ring, star, tree-based, von Neumann

and toroidal topologies) within three well-known versions of PSO (Canonical PSO, Bare Bones PSO and Fully Informed PSO) are also investigated in the PSO-based edge detector. It is found that different topologies have different effects on the accuracy of the PSO-based edge detector.

This thesis also proposes a novel dynamic topology called spatial random meaningful topology (SRMT) which is an adoptation version of a gradually increasing directed neighbourhood (GIDN). The new dynamic topology uses spatial meaningful information to compute the neighbourhood probability of each particle to be a neighbour of other particles. It uses this probability to randomly select the neighbours of each particle at each iteration of PSO. The results show that the performance of the proposed method is higher than that of other topologies in noisy images in terms of the localisation accuracy of edge detection.

Produced Publications

1. M. Setayesh, M. Zhang, and M. Johnston, "A novel particle swarm optimisation approach to detecting continuous, thin and smooth edges in noisy images," *Information Sciences*, 2012, accepted with minor changes.

2. M. Setayesh, M. Zhang, and M. Johnston, "A spatial random-meaningful neighbourhood topology in PSO for edge detection in noisy images," in *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation*, 2012, pp. 1403–1404.

3. M. Setayesh, M. Zhang, and M. Johnston, "Effects of static and dynamic topologies in particle swarm optimisation for edge detection in noisy images," in *Proceedings of the 2012 IEEE Congress on Evolutionary Computation*. IEEE Press, 2012, pp. 8–15.

4. M. Setayesh, M. Zhang, and M. Johnston, "A novel local thresholding technique in PSO for detecting continuous edges in noisy images," in *Proceedings of the 26th International Conference on Image and Vision Computing, New Zealand*. IEEE Press, 2011, pp. 333–339.

5. M. Setayesh, M. Zhang, and M. Johnston, "Investigating particle swarm optimisation topologies for edge detection in noisy images," in *Proceedings of 24th Australasian Joint Conference on Artificial Intelligence*, ser. Lecture Notes in Computer Science, vol. 7106. Springer, 2011, pp. 609–618.

6. M. Setayesh, M. Zhang, and M. Johnston, "Detection of continuous, smooth and thin edges in noisy images using constrained particle swarm optimisation," in *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, 2011, pp. 45–52.

7. M. Setayesh, M. Zhang, and M. Johnston, "Edge detection using constrained discrete particle swarm optimisation in noisy images," in *Proceedings of the 2011 IEEE Congress on Evolutionary Computation*. IEEE Press, 2011, pp. 246–253.

8. M. Setayesh, M. Zhang, and M. Johnston, "Improving edge detection using particle swarm optimisation," in *Proceedings of the 25th International Conference on Image and Vision Computing, New Zealand*. IEEE Press, 2010, pp. 1–8.

9. M. Setayesh, M. Johnston, and M. Zhang, "Edge and corner extraction using particle swarm optimisation," in *AI 2010: Advances in Artificial Intelligence*, ser. Lecture Notes in Computer Science, J. Li, Ed. Springer Berlin / Heidelberg, 2011, vol. 6464, pp. 323–333.

10. M. Setayesh, M. Zhang, and M. Johnston, "A new homogeneity-based approach to edge detection using PSO," in *Proceedings of the 24th International Conference on Image and Vision Computing, New Zealand*. IEEE Press, 2009, pp. 231–236.

Acknowledgments

It is time to express my highest thanks to my surpervisor, Prof. Megnjie Zhang for his non-stop and relentless support in helping me succeed in completing my PhD studies in computer science at Victoria University of Wellington. I also extend my highest appreciation to my second supervisor, Dr. Mark Johnston. I greatly appreciate their patience and hard work in the past three years while I was finishing my thesis.

I must acknowledge the financial assistance of Victoria University of Wellington (Victoria PhD Scholarships) during these three years.

I was so privileged to be next to my wife who has been helping me morally and intellectually during these three years. I would also like to offer my special thanks to my parents and my family-in-law who supported me in every step of this process to reach to this excellence.

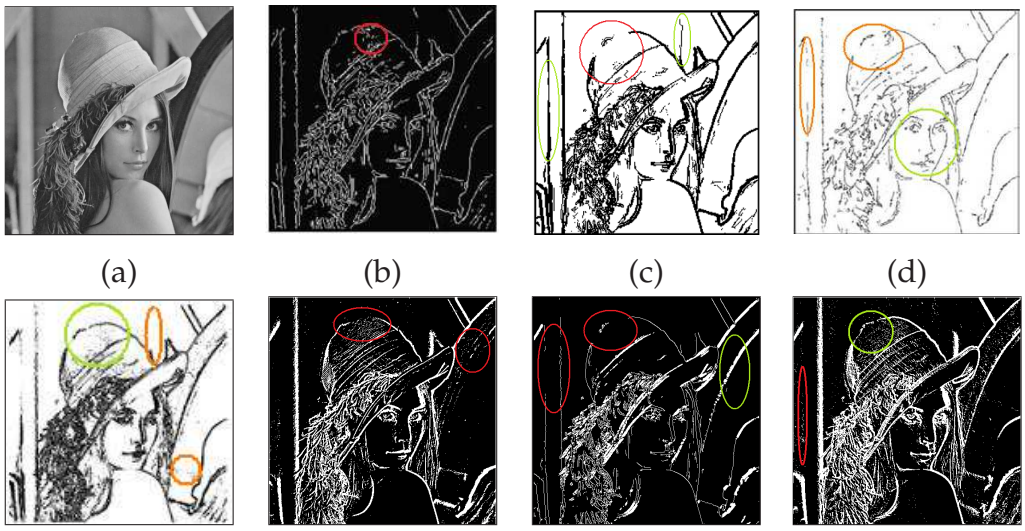
Contents

Chapter 1

Introduction

In many computer vision systems, orientation and intensity information about edges in images are used as inputs for further processing to detect objects. Precise information about edges is vital to the success of such systems [1]. Information about edges is widely used in image segmentation, image registration, image classification and pattern recognition. Hence, detection of exact edges is a very important part of image processing algorithms [2].

From an application-level view, an edge detection algorithm is one which could be able to provide continuous contours of the object boundaries [1]. However, the computations required to establish these continuous contours would be very time consuming and complex. From a pixel-level view, the edges are the areas of an image where the pixel intensities undergo a sharp change. These areas shape the contours which represent the boundary of objects. Although many edge detection algorithms have been proposed in the literature over the past three decades to improve precision of recognized edges, they still suffer from producing broken edges [3]. Noise phenomena is the most important obstacle to the detection of continuous edges [4]. It causes some variation of pixel intensities and accordingly reduces the performance of an edge detection algorithm in noisy images. Another important barrier which complicates the operation of



(a) (b) (c) (d)

(e) (g) (h) (i)

Figure 1.1: Some results of traditional and soft computing-based edge detectors applied to Lena image (a) original Lena image, (b) fuzzy-based edge detector [6], (c) ant colony-based edge detector [7], (d) neural network-based edge detector [8], (e) genetic-based edge detector [9], (f) Sobel edge detector, (g) Canny edge detector [10] and robust rank-order [11].

edge detection is illumination phenomena which causes the magnitude of the edges in the illuminated areas to become weak [5]. Since most edge detection algorithms utilise a thresholding technique to classify a pixel as an edge or non-edge based on its magnitude, a pixel with a weak magnitude may be recognised as non-edge and accordingly the edges become broken.

Traditional edge detection algorithms are very fast but they cannot perform well on noisy images and usually produce broken edges or noise spots. Advanced edge detection algorithms, which usually utilise soft computing techniques such as neural networks and support vector machines for edge detection, are highly problem-dependent and domain spe-

cific [12]. Examples of edge maps resulting from some edge detection algorithms are illustrated in Figure 1.1. These algorithms are applied to the Lena image (a) which is clean without any noise. There are several illuminated areas in this image. As can be seen from this figure, although their performance is acceptable in the areas highlighted by green ovals, they cannot perform well and generate broken edges and speckles in the areas highlighted by red ovals. If noise is added to the Lena image, the number of broken edges and speckles is increased.

This thesis proposes a novel optimisation methodology that uses a Particle Swarm Optimisation (PSO) algorithm for edge detection in noisy images. The new methodology addresses the problem of broken edges in noisy images through developing fitness functions, particle encoding schemes and information exchanging mechanisms among particles.

1.1 Motivation

In spite of human knowledge about the edge concept, there is no comprehensive definition for it. The edge can have different meanings in various contexts. Accordingly, different edge detection algorithms can recognise edges in different forms of representation and each of them can be considered as a genuine edge detection algorithm based on the definition of their interest. For example, in an edge detection algorithm, an edge can be (1) a single pixel with a local discontinuity in intensity [1], (2) a contour which links such edge pixels and shapes the boundary of an object [13] or (3) a boundary which divides an area of an image into two regions [2]. According to these definitions, the complexity of an edge detection algorithm varies, and the way by which the edges are recognised is different.

Most edge detection algorithms process a single pixel on an image at a time and calculate a value which shows the edge magnitude of the pixel, and the edge orientation. Then a thresholding technique is utilised to recognise whether or not the pixel is an edge. After applying the thresh-

olding technique, the result will be a binary image which indicates the location of all existing edges on the original image. The resulting edge maps are usually examined either by an objective index measure through comparing with ground truth images [14] or by human eyes through comparing with original images [2]. Since the edges detected by these algorithms are not usually linked and accordingly there is no relation among the edge pixels, most applications utilise a linking technique, e.g., the Hough transform. However, the linking process in such techniques cannot be perfect except for the edges on simple shapes such as circles or lines. If the main goal of an edge linking technique is to join edge pixels and remove all isolated edge pixels which are usually created by noise or digitising images, existing edge linking techniques are sufficient to address this problem. However, if the goal is to detect an edge to be as continuous as possible, traditional edge linking techniques are not a good choice because very few of them can effectively utilise any characteristics of the edge pixels other than their position, orientation and magnitude [15].

Most recent research papers in the area of edge detection have been devoted to developing algorithms that could overcome the noise phenomena. Unfortunately, most of these algorithms have a side effect of producing broken edges [3]. Therefore, an edge detector is required to detect the edges with greater continuity in noisy images and reduce the shortcomings of traditional edge detectors such as the Gaussian-based and statistical-based edge detection algorithms.

1.2 Why Particle Swarm Optimisation?

Most edge detection algorithms use the information from a small area to detect a pixel as an edge. This means that a limited area is considered in these algorithms to mark a pixel of an image as an edge. Area size has a strong effect on accuracy: the larger the area, the less the sensitivity to noise, but at the same time, the localisation accuracy is lower. If we

want to increase the localisation accuracy of the algorithm, we need to consider all edge patterns. However, this increases the computation time exponentially [16]. Therefore a heuristic algorithm is required to explore a large area to overcome the noise and consider the global structure of the edges to reduce broken edges in a reasonable time. Particle Swarm Optimisation (PSO) as a heuristic algorithm has good potential for edge detection, but has so far never been applied.

There are some general and particular advantages in using PSO for edge detection in comparison with other heuristic algorithms. The most important general advantages of PSO, which make it attractive for researchers, are:

1. High speed of convergence: there are two main reasons which make PSO faster than other heuristic algorithms. The first is the use of the topology that defines how particles are connected to each other as an information sharing mechanism. Once a better position is found by a particle, the information of the better position is quickly transfered to the other particles which are connected through the topology. This allows all particles to rapidly converge to a local optimum. The second is the use of the velocity concept to calculate the new position of each particle. For these reasons, the PSO algorithm has a higher rate of convergence compared to other heuristic algorithms.

2. The ease of the implementation: PSO is easy to implement due to simplicity of its process.

3. Fewer operators: PSO has one simple operator, the velocity calculation. Other heuristic algorithms have often more than one operator.

4. A limited memory for each particle: there are two types of memory, cognitive and social, which influence the movement of the particles. The cognitive memory saves the best previous position visited by each particle and the social memory keeps the position of the best

point in search space visited by all swarm particles. These two memory types allow the particles to retain knowledge acquired so far in spite of being updated periodically.

Particular advantages of the use of PSO for edge detection in noisy images include:

1. The speed of an edge detection algorithm is important in many applications. Therefore edges as low-level features must be detected in a reasonable time. An edge detection algorithm also needs to explore a large area of an image to overcome noise and consider all edge patterns to increase edge localisation accuracy. The high speed of PSO in convergence makes it a good candidate for edge detection.

2. Since PSO does not use the gradient information of the functions being optimised, it has a high capability to optimise noisy functions [17]. Parsopoulos and Vrahatis [18] experimentally showed that PSO in the presence of noise is very stable and efficient. The results indicated that the presence of noise can help PSO to avoid local optimum and detect the global optimum of an objective function. They also showed that even in the cases where the noise level is high, PSO may move closer to the position of the global optimum. PSO has been successfully applied to many problems in noisy environments, such as noise cancellation in images [19], vision tracking [20] and image segmentation [21]. It is very likely that PSO can deal with noise in edge detection.

1.3 Goals

The overall goal of this thesis is to investigate the capability of PSO for edge detection. We will develop a particle swarm optimisation-based (PSO)

approach to improving edge detection to deal with noise and reduce broken edges. This thesis aims to investigate a novel approach to the detection of edges **with greater continuity in noisy images** using PSO.

This thesis concentrates on using the PSO algorithm to handle the edge detection better than conventional vision approaches in noisy environments through exploring a large area and extracting the global structure of the edge **without** any pre-processing (such as using a smoothing technique as a noise removal filter) or post-processing algorithms (such as a linking technique).

To achieve the overall goal stated above, this thesis focuses on five major aspects within the PSO system:

1. Developing a new PSO-based approach to detecting edges with greater continuity in noisy images and comparing it with conventional edge detection methods.

2. Developing a constrained PSO-based algorithm for edge detection.

3. Improving efficiency of the newly developed algorithm through developing a novel local thresholding technique.

4. Investigating the effects of different static topologies in PSO for edge detection in noisy images.

5. Developing a novel dynamic topology in PSO in order to improve the effectiveness of the new PSO-based edge detector.

To achieve these goals, this thesis focuses on finding answers to the following research question:

> **How can PSO be used to explore a large area in order to overcome noise and extract the global structure of an edge in order to detect edges with greater continuity?**

To address the goal of developing a novel PSO approach, we intend to focus on developing a new fitness function and a new encoding scheme for edge detection in noisy images. When the goal is to use a constrained PSO, we need to use constraint handling methods in PSO in order to improve the efficiency and effectiveness of the PSO-based edge detector. When the goal is to develop a novel local thresholding technique, we intend to focus on local features to estimate a threshold value for the PSO-based edge detector. In the two last goals, we aim to investigate the effect of using different static and dynamic topologies in PSO as a mechanism of information sharing among particles and develop a novel dynamic topology to increase the effectiveness of the algorithm.

1.4 Major Contributions

The major contributions of this thesis are:

1. The thesis proposes a novel optimisation approach to detecting edges with a greater continuity in noisy images. Unlike many existing edge detection algorithms that usually process a single pixel at a single run without considering the global structure of the edges, the proposed system recognises a continuous edge as a sequence of connected pixels in order to decrease the number of broken edges and explore a larger area in comparison to the traditional edge detection algorithms in order to reduce the effect of noise. A collection of fitness functions along with different encoding schemes are proposed for PSO and applied to different synthetic and real noisy images. Our results show that the proposed algorithm significantly improves the accuracy of edge detection in noisy images in comparison to the traditional methods. The results have been published in [22][23][24].

2. The thesis proposes two constrained PSO-based algorithms in order to increase the accuracy of edge detection in noisy images. Two con-

straint handling methods are investigated, and their performance is compared with each other in terms of efficiency and effectiveness. The results show that the PSO-based edge detector with a penalising method is more efficient than when it is equipped with a preservation method [25][26].

3. The thesis proposes a novel local thresholding technique which is used inside the PSO-based edge detector. The novel technique is based on an existing image binarisation method with a high performance in illuminated noisy images. The results show that the performance of the PSO-based edge detector equipped with the local thresholding technique is higher than when it uses a global thresholding technique [27].

4. The thesis investigates the effect of using different static topologies in three well-known versions of PSO when they are applied to edge detection in noisy images. The results show that different topologies have different effects on the accuracy of three versions of PSO which use different information sharing mechanisms among particles [28][29].

5. The thesis finally proposes a novel dynamic topology as an information sharing mechanism in order to increase the accuracy of the PSO-based edge detector. The novel topology is compared with several existing static and dynamic topologies. The results show that the novel topology can share information among particles in a more effective way in comparison to other existing topologies and accordingly increase the accuracy of the PSO-based edge detector [30].

1.5 Thesis Organisation

1.5.1 Structure

The main contribution chapters of this thesis are presented in Chapters 4–7. Each chapter in this thesis correspondingly addresses a major goal (aspect) described in Section 1.3 (except that Chapter 4 addresses the first two aspects). Each chapter follows a similar high-level structure. Each chapter starts with an introduction and background followed by the proposed algorithms. Then the proposed algorithm is examined against the traditional algorithms followed by a discussion on the empirical results.

1.5.2 Outline

The outline of this thesis is as follows:

- *Chapter 2: Literature Review*
  This chapter reviews important existing edge detection algorithms followed by a discussion of their strengths and limitations. The fundamental concepts of edge detection are covered followed by the main concepts of optimisation and swarm intelligence. It also covers the main concepts of particle swarm optimisation and its applications in image processing and computer vision.

- *Chapter 3: Image Sets*
  This chapter presents the image sets which are used in all experiments throughout this thesis. In this chapter, impulse and Gaussian noise generators are described in detail.

- *Chapter 4: Novel Edge Detection Algorithms Robust to Noise using PSO*
  In this chapter, a novel edge detection algorithm is introduced and designed to be robust to noise and reduce the broken edges. The newly developed algorithm is compared to the state-of-the art edge detectors by subjective and objective methods. This chapter shows

how PSO explores a large area to consider the global structure of the edges and detect them with greater continuity through a suitable particle encoding and an effective fitness function. The sensitivity of the new PSO-based algorithm to noise (Gaussian noise and impulse noise in different noise levels) is tested and compared with that of traditional edge detectors. In this chapter, two well-known applicable constraint handling methods are used within the PSO-based edge detector and their performance is examined in terms of efficiency and effectiveness.

- *Chapter 5: Novel Thresholding Technique for the PSO-based Edge Detector*
  This chapter proposes a novel local thresholding technique which uses local features extracted from an area of an image to estimate a required threshold value for the PSO-based edge detector. The accuracy of the PSO-based edge detector equipped with the novel local topology is compared with a well-known global thresholding technique as the state-of-the-art.

- *Chapter 6: Investigating Effects of Static Topologies on Accuracy*
  In this chapter, the effects of using different static topologies within three well-known versions of PSO are investigated. This chapter shows how the information sharing mechanism among particles can affect the accuracy of the PSO-based edge detector when it uses the mechanisms proposed in these three versions.

- *Chapter 7: A Novel Spatial-Random Dynamic Topology*
  This chapter proposes a novel topology which uses spatial information in order to more effectively share information among particles. The novel topology is compared with well-known dynamic and static topologies in terms of the accuracy of the PSO-based edge detector.

- *Chapter 8: Conclusions and Future Work*

The conclusions from the experiments are reviewed and overall conclusions are drawn in this chapter. Future research directions are also discussed in this chapter.

Chapter 2

Literature Review

This chapter reviews some general concepts of image processing, image analysis and computer vision followed by the fundamental concepts of feature and edge detection related to this thesis. This chapter also gives a brief introduction to edge detection algorithms which have been developed in different frameworks, the importance of edge detection in image analysis and computer vision, and then an overview of evolutionary computation techniques as well as swarm intelligence and particle swarm optimisation. This chapter will also give a brief introduction to some edge linking techniques and some common objective approaches to assessing the edge detection algorithms. This chapter also discusses the limitations of existing methods for edge detection in noisy images and supports the motivation of the thesis.

2.1 Image Processing, Image Analysis and Computer Vision

Vision is the most significant of the human senses. It is obvious that images play an important role in human perception. However, human vision can only cover visual bands of the electromagnetic spectrum, whereas ma-

chine vision covers the whole of the electromagnetic range from gamma to radio waves such as X-ray images in medicine and gamma-ray images taken by satellites. Therefore, computer imaging encompasses wide and various domains of applications [2].

The field of digital image processing usually refers to processing of a digital image through a digital computer. A digital image is composed of some finite image elements each of which has a particular value and location. These elements are named image elements, pels (sub-pixels) or picture elements (pixels) [2]. Pixel is the common term that is widely used to indicate elements of the image.

There is no general agreement among researchers where the image processing boundary starts and the boundary of other related research areas such as computer vision and image analysis stops. Some authors define image processing as a discipline in which both input and output of a process are an image. This definition is a limiting and somewhat artificial boundary [1]. For example, even the average intensity of an image, which yields a single number, would therefore not be considered an image processing operation. Another definition divides all related processes in these areas into three different levels: low, mid and high-levels. Low-level processes include low-level tasks such as image preprocessing to decrease the noise, image sharpening and enhancing. In this level, processes are characterised by the fact that both input and output are images. Mid-level processes involve operations such as segmentation (partitioning an image into objects and region) and the description of those objects to reduce them to a suitable form for classification and object recognition tasks. These sorts of processes are characterised by the fact that inputs of the processes are usually images, but their outputs are attributes which are extracted from the images. These attributes can be information about edges, contours or individual and simple objects. High-level processes include operations in which the object of interest is completely recognised based on those attributes [31]. Under this definition, image processing involves

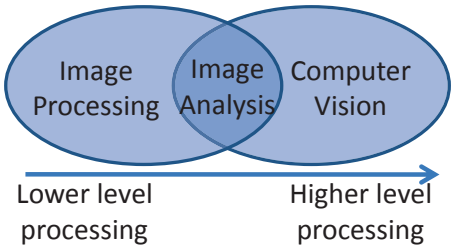


Figure 2.1: Computer imaging.

all low-level operations and some mid-level tasks, whereas computer vision includes some mid-level tasks along with all high-level operations. In fact, image analysis has a considerable overlap with image processing and computer vision as shown in Figure 2.1.

Image processing includes some operations for image enhancement and low-level feature detection. Outputs of all image enhancement algorithms are images which can be seen by human eyes. Low-level features are features which are detected by image processing algorithms. These features can be basic features such as edges, lines, curvatures, corners and other kinds of interest points which can be used as inputs in mid or high-level processing algorithms.

The aim of image analysis is to extract meaningful features from image data in order to reduce computational processing cost in higher level processes. Image analysis can be considered as a data reduction process and its operations usually focus on reducing image data [2]. Its operations are divided into two categories: global operations and local operations. The local operations are those that operate locally and extract data based on a smaller area. Global operations are those that operate globally and extract data based on all pixels of an image [32]. The image analysis process can be broken into three steps as shown in Figure 2.2.

Computer vision is the highest level of the computer imaging process [13]. It is used widely in a variety of real world applications, which include

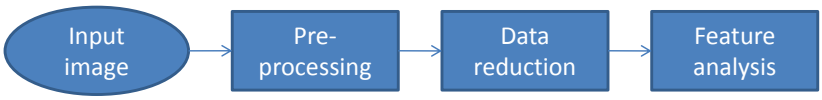


Figure 2.2: Three steps of the image analysis process.

the following examples (edge detection algorithms have been utilised in these chosen samples).

- Optical character recognition (OCR): widely used in digital library in order to digitise documents and books [33], in detection of vehicle number plates [34], and in recognition of handwritten characters [35].

- Machine inspection: fast machine-based inspection used for quality assurance without human participation, e.g., defect detection in steel casting using X-ray vision [36].

- Automated checkout lane: in many shops (e.g. supermarkets), there are automated checkout lanes in which an object recognition system is applied to identify items bought by a customer [13].

- Photogrammetry: one of the widely used computer vision applications is 3D model building, e.g., Bing maps (http://www.bing.com) [13].

- Medical imaging: registering pre-operative and intra-operative imagery, or performing long-term studies of human brain morphology with ageing [37]

- Safety in automotive industries: detection of unexpected objects such as a pedestrian or any other obstacle while driving, where active vision techniques like radar do not work well [38][39].

- Surveillance: monitoring pedestrians, vehicles on highways, or people in shops [40].
- Fingerprint recognition and biometric-based security systems: used for automatic access control systems, as well as forensic identification [41].

2.2 Feature Detection

The initial input of a computer vision system is an image. This image often includes too much data to be processed, so meaningful features need to be extracted from the image to reduce the size of the data to be processed. Extracted features from an image might be low-level or high-level. In computer vision and image processing, low-level feature detection refers to methods which are utilised to compute an abstraction of image data and making a decision at each point whether or not there is a specified image feature in that point. Low-level features are basic features that are automatically extracted from an image without having any information about existing shapes in the image [42]. High-level features refer to those features which contain information about shapes and components of objects occurring in an image. These components and shapes could be eyes, nose, and ears in a face detection system [43] or wheels, headlights and taillights in a vehicle detection system [44]. These features are typically used for high-level tasks such as object classification.

2.3 Edge Detection

Edge detection as low-level feature detection is one of the critical elements in image processing, because there is a high proportion of image information on edges. The main function of edge detection is to find the boundaries of image regions, based on properties such as intensity and texture

[11]. While applying an edge detector to an image considerably reduces the amount of the data to be processed, it still preserves the main shape of the objects present in an image. The shape of edges depends on many parameters, such as geometrical and optical properties of an image, illumination conditions and noise level in the image [45]. The edge detection process typically results in a edge map which is usually a binary image. This image describes the classification of each pixel of the image, as well as some other edge attributes such as magnitude and orientation [46].

Edge detection algorithms often utilise three different operations in order to generate an edge map. These operations generally include (1) smoothing and noise reduction, (2) image differentiation (calculating edge magnitudes and their orientations) and (3) labelling steps. At the first step, a smoothing technique is used to reduce noise and regularise the image for numerical differentiation. At the second step, discrete image differentiation is often applied to the image to calculate desired derivatives. Finally, a labelling operation is used to classify the pixels on the image as edges or non-edges. This section presents the background knowledge necessary for the reader to understand the main concepts of these three operations in edge detection.

2.3.1 Image Differentiation

The calculation of the first and second order derivatives of an image are the most popular image differentiation techniques in edge detection [47]. Let $G(x, y) : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$ be a function to represent the image intensity of a pixel on an analogue image at location (x, y). To calculate the first derivative of G along direction $\vec{r}$, the partial derivatives of G with respect to x and y are used as:

$$\begin{aligned} \frac{\partial G}{\partial r} =& \frac{\partial G}{\partial x}\frac{\partial x}{\partial r} + \frac{\partial G}{\partial y}\frac{\partial y}{\partial r} = G_x cos(\varphi) + G_y sin(\varphi) \\ G_x =& \frac{\partial G}{\partial x} \quad , \quad G_y = \frac{\partial G}{\partial y} \end{aligned} \tag{2.1}$$

where r is a parameter along the direction represented by $\vec{r}$ and φ is the angle between $\vec{r}$ and axis x. So $x = rcos(\varphi)$ and $y = rsin(\varphi)$. The gradient of G (∇G), by definition, is a vector with the same direction as the maximum directional derivative for which

$$\frac{\partial}{\partial \varphi}\left(\frac{\partial G}{\partial r}\right) = 0.$$

So its direction and magnitude are computed as follows [48]:

$$\varphi_{\nabla} = arctan(\frac{G_y}{G_x}) \tag{2.2}$$

$$|\nabla G| = \sqrt{{G_x}^2 + {G_y}^2} \tag{2.3}$$

According to the definition of the gradient, edge pixels are where their modulus of the gradient is maximum and the gradient direction is orthogonal to the direction of the contour on these pixels. Since the edge detection algorithms based on the definition of gradient give the maximum response when they are aligned with the orthogonal direction of the edges, these algorithms are directional.

Edge detection algorithms based on second order derivatives utilise one of two frequently used operators, i.e, the second derivative along the direction of the gradient and the Laplacian operator. The second derivative of G along direction $\vec{r}$ is calculated as:

$$\begin{aligned} \frac{\partial^2 G}{\partial r^2} &= \frac{{G_x}^2 G_{xx} + 2G_x G_y G_{xy} + {G_y}^2 G_{yy}}{{G_x}^2 + {G_y}^2} \\ G_{xx} = \frac{\partial^2 G}{\partial x^2},\ \ G_{yy} &= \frac{\partial^2 G}{\partial y^2},\ \ G_{xy} = \frac{\partial^2 G}{\partial x \partial y} \end{aligned} \tag{2.4}$$

The Laplacian of G ($\nabla^2 G$), which is defined as Equation (2.5), is an approximation of the second order derivative along the direction of the gradient [47].

$$\nabla^2 G = G_{xx} + G_{yy} \tag{2.5}$$

There are at least three main advantages of using the Laplacian in comparison to the second derivative along the gradient direction. First, the computation of the Laplacian is simple because it is sum of its two second order derivatives. Second, in opposite to $\partial/\partial r^2$ which is non-linear, the Laplacian operator is linear. Finally, it is a non-directional operator. Accordingly, the Laplacian operator does not need to determine the most appropriate direction which is required by most operators [2].

2.3.2 2D Discrete Differentiation

All 2D digital images are represented by 2D arrays of pixels which are quantified samples. When the differentiation concepts are applied to the digital images, we encounter at least two problems. The first problem is how to calculate image differentiation in the digital images. In these images, we need to calculate a discrete approximation of the differential operators. The second problem is the amplification of high frequency noise when a differentiation operator is applied to the digital images [2]. Several solutions have been proposed to address these two problems. This section presents some of these solutions.

A digital image, obtained from sampling and quantization of an analogue image G, is defined as function I which is a mapping $X \times Y \xrightarrow{I} P$ where $X = \{0, 1, \ldots, N_c - 1\}$, $Y = \{0, 1, \ldots, N_r - 1\}$ and $P = \{0, 1, \ldots, N_p\}$. Here, N_c is the number of columns in the image, N_r is the number of its rows and N_p is the maximum intensity level of the pixels of the image. The calculation of the first differences along the main axes, x and y is one of easiest ways to estimate the first order derivatives G_x and G_y [42]:

$$\begin{aligned} G_x(c,r) &\cong I_x(c,r) = I(c,r) - I(c+1,r) \\ G_y(c,r) &\cong I_y(c,r) = I(c,r) - I(c,r+1) \end{aligned} \tag{2.6}$$

where $I_x(c,r)$ and $I_y(c,r)$ are the approximations of $G_x(c,r)$ and $G_y(c,r)$

respectively. These operators are often denoted as masks, such as:

$$\begin{array}{rclcrcl} I_x & = & M_x \begin{bmatrix} I(c,r) \\ I(c+1,r) \end{bmatrix}, & & I_y & = & \begin{bmatrix} I(c,r) & I(c,r+1) \end{bmatrix} M_y \\ M_x & = & \begin{bmatrix} \mathbf{1} & -1 \end{bmatrix}, & & M_y & = & \begin{bmatrix} \mathbf{1} \\ -1 \end{bmatrix} \end{array} \tag{2.7}$$

As can bee seen in Equation (2.7), this operator is not symmetric. In this equations, the bold value shows the origin of the masks. To avoid this problem, an odd number of mask elements is usually used to define the mask as:

$$M_x = \begin{bmatrix} 1 & \mathbf{0} & -1 \end{bmatrix}, \quad M_y = \begin{bmatrix} 1 \\ \mathbf{0} \\ -1 \end{bmatrix}$$

Roberts, Prewitt, Sobel and Frei-Chen are four other commonly used first order derivative approximations along two main perpendicular axes [49][50][1] which are defined as follows:
Roberts:

$$M_1 = \begin{bmatrix} \mathbf{0} & +1 \\ -1 & 0 \end{bmatrix}, \quad M_2 = \begin{bmatrix} \mathbf{+1} & 0 \\ 0 & -1 \end{bmatrix} \tag{2.8}$$

Prewitt:

$$M_x = \begin{bmatrix} -1 & 0 & +1 \\ -1 & \mathbf{0} & +1 \\ -1 & 0 & +1 \end{bmatrix}, \quad M_y = \begin{bmatrix} +1 & +1 & +1 \\ 0 & \mathbf{0} & 0 \\ -1 & -1 & -1 \end{bmatrix} \tag{2.9}$$

Sobel:

$$M_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & \mathbf{0} & +2 \\ -1 & 0 & +1 \end{bmatrix}, \quad M_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & \mathbf{0} & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{2.10}$$

Frei-Chen:

$$M_x = \begin{bmatrix} -1 & 0 & +1 \\ -\sqrt{2} & \mathbf{0} & +\sqrt{2} \\ -1 & 0 & +1 \end{bmatrix}, \quad M_y = \begin{bmatrix} +1 & +\sqrt{2} & +1 \\ 0 & \mathbf{0} & 0 \\ -1 & -\sqrt{2} & -1 \end{bmatrix} \tag{2.11}$$

Except the mask in Equation (2.7), all other masks above are odd and approximate the first derivative along two perpendicular axes [42]. The Roberts operator estimates the first derivative along the axes rotated 45 degrees with respect to the usual orientation of the column and row. To apply these operators, an internal product between the mask of interest (M_α) and image I is used as follows:

$$M_\alpha(c, r) = \sum_i \sum_j I(c + i, r + j).\, H_{\alpha_{ji}} \tag{2.12}$$

Here, α is the direction of interest which is either 1 or 2 for the Robert operator, and either x or y for all other operators; $H_{\alpha ji}$ is element of row j and column i of mask m_α. Note that the element, being represented in bold in mask M_α, corresponds to element ($i = 0$, $j = 0$).

Although two directional derivatives are enough to calculate the gradient, there are several operators which use more than two directional derivatives for noise suppression reasons. In these operators, the gradient is approximated by the directional derivative with the highest magnitude. The Kirsch operator [51] is one of the most well-known operators which have more than two masks as follows:

$$\begin{aligned} M_E \rightarrow &= \begin{bmatrix} -3 & -3 & +5 \\ -3 & \mathbf{0} & +5 \\ -3 & -3 & +5 \end{bmatrix}, & M_{NE} \nearrow &= \begin{bmatrix} -3 & +5 & +5 \\ -3 & \mathbf{0} & +5 \\ -3 & -3 & -3 \end{bmatrix} \\ M_N \uparrow &= \begin{bmatrix} +5 & +5 & +5 \\ -3 & \mathbf{0} & -3 \\ -3 & -3 & -3 \end{bmatrix}, & M_{NW} \nwarrow &= \begin{bmatrix} +5 & +5 & -3 \\ +5 & \mathbf{0} & -3 \\ -3 & -3 & -3 \end{bmatrix} \end{aligned} \tag{2.13}$$

Here, M_E, M_{NE}, M_N and M_{NW} denote the masks in direction east, north east, north and north west respectively (see the arrows in Equation (2.13). Masks M_{NE} and M_{NW} are generated by the rotation of 45 degrees of masks M_E and M_N correspondingly.

Second differences can be used to approximate the second order derivatives in a similar way as first differences for the estimation of the first order derivatives [42]. The second differences along the main axes are defined as:

$$\begin{aligned} G_{xx} \approxeq I_{xx}(c,r) &= I_x(c-1,r) - I_x(c,r) \\ &= I(c-1,r) - 2I(c,r) + I(c+1,r) \\ G_{yy} \approxeq I_{yy}(c,r) &= I_y(c,r-1) - I_y(c,r) \\ &= I(c,r-1) - 2I(c,r) + I(c,r+1) \end{aligned} \tag{2.14}$$

These approximations can be represented by the following masks:

$$M_{xx} = \begin{bmatrix} 0 & 0 & 0 \\ +1 & \mathbf{-2} & +1 \\ 0 & 0 & 0 \end{bmatrix}, \qquad M_{yy} = \begin{bmatrix} 0 & +1 & 0 \\ 0 & \mathbf{-2} & 0 \\ 0 & +1 & 0 \end{bmatrix} \tag{2.15}$$

According to the definition of the Laplacian in Equation (2.5), its discrete approximation can be calculated as:

$$M_{xx+yy} = M_{xx} + M_{yy} = \begin{bmatrix} 0 & +1 & 0 \\ +1 & \mathbf{-4} & +1 \\ 0 & +1 & 0 \end{bmatrix} \tag{2.16}$$

2.3.3 Convolution

Convolution is generally operated on two signals, and measures the overlap of one signal (G) with another delayed or shifted signal (H) [2]. The continuous version of the 2D convolution operator $\star$ is defined as:

$$[G \star H](s,t) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} G(c,r)H(s-c,t-r)dcdr. \tag{2.17}$$

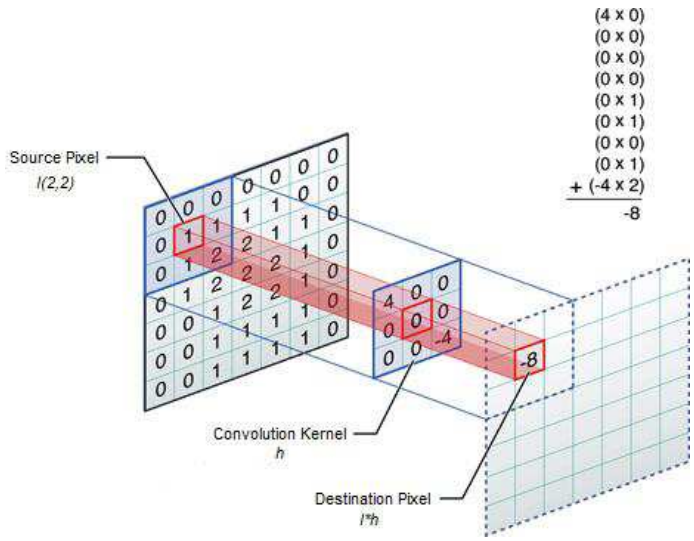


Figure 2.3: Discrete 2D Convolution (adapted from [2]).

In the field of image processing, the convolution operator is a multi-purpose filter which can produce different effects by various convolution kernels. Convolving an image with different kernels or masks produces different results. For example, a uniform kernel behaves as a box filter, averaging, smoothing or noise removal filter. The convolution operator with a difference kernel acts as an edge detector (compare Equations (2.12) and (2.18)). A discrete version of the convolution operator in 2D for functions I and h is given by

$$[I \star h](s,t) = \sum_{c=0}^{W-1} \sum_{r=0}^{H-1} I(c,r)h(s-c,t-r) \tag{2.18}$$

Here, I is the function representing the original image and h is the convolution kernel; W and H is the width and height of the image. The convolution calculates a new value for each pixel by adding the weighted values of all its surrounding pixels together as can be seen in Figure 2.3.

Since most edge detection operators are sensitive to noise, they typically need to use an image from which noise has been removed or reduced. As depicted in Figure 2.4, the first derivative of a function corrupted by

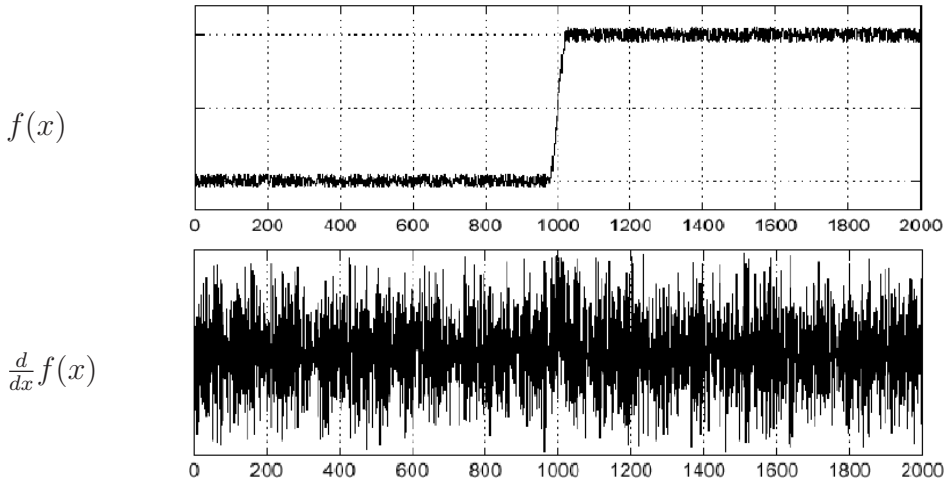


Figure 2.4: First derivative of a function corrupted by noise adapted from [2].

noise cannot localise a step edge within the function.

To reduce or remove noise from function f as shown in Figure 2.5, an appropriate convolution kernel such as a Gaussian function (see function h in this figure) need to be applied.

2.3.4 Smoothing using Convolution

The convolution operator is usually used to smooth an image in order to remove or reduce noise. A simple-looking way is to simply average the intensity of neighbouring pixels through a convolution kernel with the size of $(2n+1) \times (2n+1)$ where each element of the kernel has a value of $\frac{1}{(2n+1)^2}$. This kernel is called a box filter. For example, a five by five version of such a mask would be:

$$h = \begin{bmatrix} 1/25 & 1/25 & 1/25 & 1/25 & 1/25 \\ 1/25 & 1/25 & 1/25 & 1/25 & 1/25 \\ 1/25 & 1/25 & 1/25 & 1/25 & 1/25 \\ 1/25 & 1/25 & 1/25 & 1/25 & 1/25 \\ 1/25 & 1/25 & 1/25 & 1/25 & 1/25 \end{bmatrix}$$

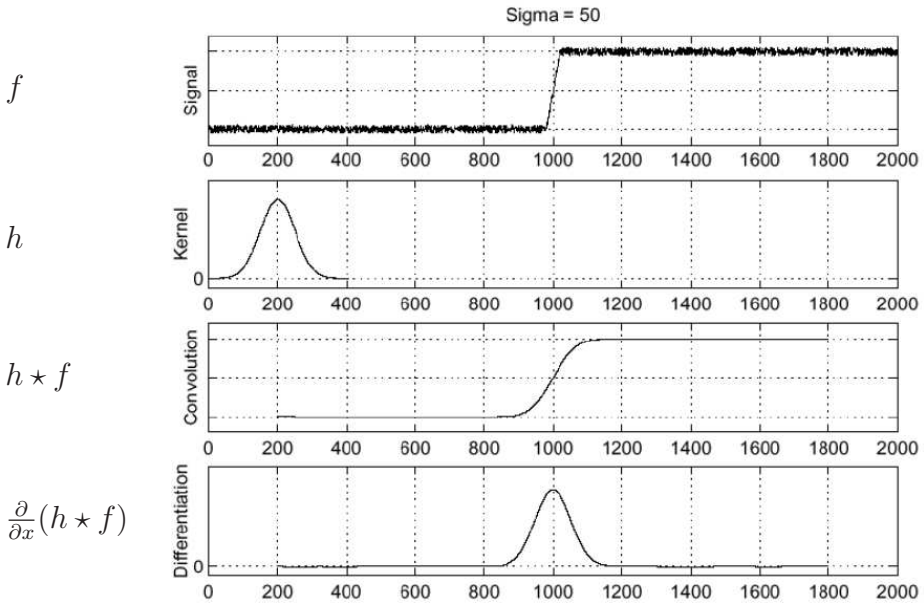


Figure 2.5: Convolution of a noisy function with a Gaussian function in order to reduce noise adapted from [2].

The convolution of this kernel with an image computes the average value of all neighbouring pixels by simply summing $1/25$ of each of 25 values. When this kernel mask is applied to a noisy image, the Gibbs phenomena may be occurred and ringing artifacts are resulted [52]. This phenomena is caused by the box filter when there is a sharp discontinuity in an area of an image. A weighted mask, such as a triangle or Gaussian filter is usually used to avoid this problem. Gaussian filters use the canonical bell-shaped or normal distribution to weight each element of the convolution mask. Unlike the box and triangle filters, the Gaussian filters perform reasonably well in practice. The two dimensional version of this filter is defined as:

$$\begin{aligned} G_\sigma(x,y) &= \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \\ G_\sigma(r) &= \frac{1}{2\pi\sigma^2} e^{-\frac{r^2}{2\sigma^2}} \end{aligned} \tag{2.19}$$

where r is the distance of point (x, y) from the origin.

Since $\int\int G_\sigma(x,y)dxdy = 1$, $\int\int G_\sigma(x,y) \star f(x,y)dxdy = \int\int f(x,y)dxdy$. In other words, convolving a Gaussian filter with a function preserves the area under the function. While the analog version of the Gaussian filter is defined across the entire domain $[-\infty, +\infty]$, in its discrete version we can truncate it after some point because the value of the Gaussian function becomes very small beyond the point. Since the value of $G_\sigma(4\sigma)$ is very close to zero, we generally use the Gaussian filter in the range of -4σ and $+4\sigma$. If we truncate the Gaussian filter too quickly (for example at $\pm\sigma$), the Gibbs phenomena occurs like in the boxing filter. That is because the values of the filter near its edge are relatively large. Accordingly, the filter produces a large amount of high frequency noise when the filter is moved from one position to the next. The discrete version of the filter should be truncated such that the sum of the weights in this filter is equal to 1 like its analog version. The simplest way to calculate the discrete versions of the Gaussian filter with different σ, is to compute the unnormalised Gaussian function as:

$$\mathcal{G}_\sigma(i,j) = e^{-\frac{i^2+j^2}{2\sigma^2}} \tag{2.20}$$

where (i,j) is an integer grid point ranging from -4σ to $+4\sigma$. The centre of the filter is the origin $(0,0)$. After calculating the unnomalised value of each element of the filter, the value of the element is divided by the total sum of all elements of the filter in order to normalise its value. So, $G_\sigma(i,j) = \frac{\mathcal{G}_\sigma(i,j)}{\sum\sum\mathcal{G}_\sigma}$ [47]. For example, when $\sigma = 0.5$, the filter should be a 5×5 mask where its centre is at $(0,0)$, and the range of i and j is from -4×0.5 to $+4 \times 0.5$. The sum of its elements equals to 1 and the mask is symmetric around the origin as the following:

$$\begin{bmatrix} 6.96 \times 10^{-8} & 2.80 \times 10^{-5} & 2.07 \times 10^{-4} & 2.80 \times 10^{-5} & 6.96 \times 10^{-8} \\ 2.80 \times 10^{-5} & 0.0113 & 0.0837 & 0.0113 & 2.80 \times 10^{-5} \\ 2.07 \times 10^{-4} & 0.0837 & 0.618 & 0.0837 & 2.07 \times 10^{-4} \\ 2.80 \times 10^{-5} & 0.0113 & 0.0837 & 0.0113 & 2.80 \times 10^{-5} \\ 6.96 \times 10^{-8} & 2.80 \times 10^{-5} & 2.07 \times 10^{-4} & 2.80 \times 10^{-5} & 6.96 \times 10^{-8} \end{bmatrix}$$

If i and j range from -4σ to $+4\sigma$ (in the example above from -2 to $+2$), the general discrete convolution of the Gaussian filter with digital image I is given by

$$I_s(x,y) = \sum_{i=-4\sigma}^{4\sigma} \sum_{j=-4\sigma}^{4\sigma} \mathcal{G}_\sigma(i,j) I(x-i, y-j) \tag{2.21}$$

where I_s is the smoothed version of image I. In the Gaussian filter, there is a trade-off between the size of the filter (σ) and its ability to locate an edge. If its size is large, edges will be smoothed and accordingly edge detection algorithms cannot perform well in these areas.

2.3.5 Edge Labelling

Edge labeling is the process of the localisation of edges. This process depends on the edge operator convolved on an image. For example, in the gradient-based edge detectors, the localisation of edges is performed by thresholding the gradient magnitude of the edges. The edges resulting from a simple thresholding technique are usually thick and consequently a skeletonisation technique should be utilised. One of the commonly used techniques is non-maximum suppression (NMS) which was proposed by Canny [53]. This technique finds the local maxima along the direction of the gradient vector. For the localisation of zero-crossing in the second-order derivative-based edge detectors, the output for a given pixel is compared with the output for its neighbours at its left and below. If the outputs for these three pixels do not have the same sign, a zero-crossing exists at this pixel. Chen and Medioni showed that the localisation ability of an operator can be improved by considering more than two principal directions (horizontal and vertical) [54].

Non-Maximum Suppression (NMS)

NMS is used as a post processing technique to essentially locate the highest points in the edge magnitude data by the use of edge direction infor-

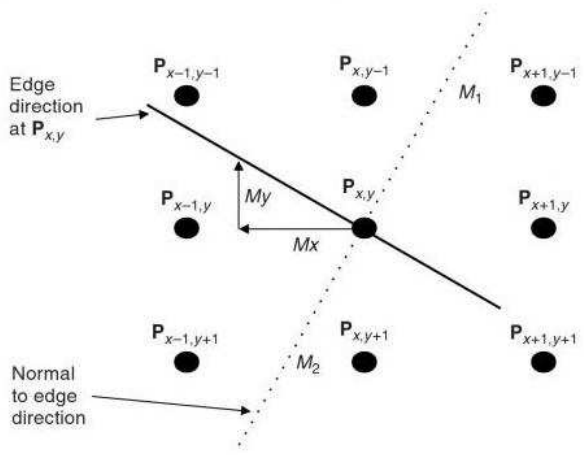


Figure 2.6: Interpolation in NMS [42].

mation. This method checks whether or not points are at the peak of a ridge. In a 3×3 area, a point is a local maxima if the gradient magnitude on both sides of it are less than its gradient. Generally, we need to consider all points along a line which is normal to the edge at the point. Figure 2.6 shows the neigbours of point $P_{x,y}$, its edge direction and the normal to the edge direction at $P_{x,y}$. If the gradient magnitude of point $P_{x,y}$ is larger than the gradient at point M_1 and M_2, point $P_{x,y}$ is marked as a local maxima by the NMS technique. Since we have a discrete neighbourhood in digital images, M_1 and M_2 can be those neighbours that the angle between the neighbourhod vector and the normal is the smallest among other angles between other neighbours and the normal. In some NMS techqniues, M_1 and M_2 are interpolated by first-order interpolation using the gradient at point $P_{x,y}$ in direction x (M_x) and y (M_y) as follows [42]:

$$M_1 = \frac{M_y}{M_x} M(x+1, y-1) + \frac{M_x - M_y}{M_x} M(x, y-1) \tag{2.22}$$

$$M_2 = \frac{M_y}{M_x} M(x-1, y+1) + \frac{M_x - M_y}{M_x} M(x, y+1) \tag{2.23}$$

Thresholding Techniques

The output of most edge detection operators is a floating point value which shows the magnitude of edges. A high value is where there is a strong edge and a low value is where there is a weak edge or no edge. Thresholding techniques are often used to remove unwanted weak edges. The output of a threshold technique is a binary edge map that shows which pixels are edges and which ones are not. In some thresholding techniques, a single threshold value is used to determine the plausibility values of true edges. The threshold value is the minimum acceptable value for the plausibility value of true edges. Due to the variation of this value from an image to another image, the edges resulting from such a thresholding technique are usually broken. Therefore, Canny proposed a hysteresis thresholding technique to improve the continuity of edges [10].

In the hysteresis thresholding technique, two threshold values are used: high and low. Since important edge pixels are along continuous curves, we can easily follow a weak segment of a given line which is along a strong segment of it and suppress a few noisy spots that do not belong the line but have a large gradient [55]. This technique starts from the points whose magnitudes exceeds the high threshold and then traces edges from these points using the edge orientation information estimated by an edge detection operator. While tracing the edges, the low threshold value is applied to trace weak segments as long as another point, whose magnitude exceeds the high threshold, is found. All strong and weak edges are processed to provide a binary image map whose pixels are marked as either an edge or or a non-edge.

2.4 Edge Detection Algorithms

In the past three decades, many edge detection algorithms have been proposed with specific applications. This section provides a summary of edge

detection algorithms which have been proposed in different frameworks. This section contains a brief overview of edge detectors based on first and second derivatives, Gaussian filters, statistics, soft computing techniques and different transforms followed by a discussion of their advantages and disadvantages. These algorithms are closely related to this thesis, and some of them are used for comparison purposes.

2.4.1 First Derivative-based Edge Detectors

This category of edge detection algorithms uses first derivatives to detect the edges of an image. These operators are based on the gradient operator ∇ [48]. The gradient magnitude indicates the strength of the edge and the orientation gives the direction of the greatest change or the direction of the edge as early shown in Equation (2.2). Robert, Sobel, Prewitt, Kirsch, Robinson, Frei-Chen, Deatsch and Fram, Nevatia and Babu, Ikonomopoulos, Davies, Kitchen and Malin, Hancook and Kittler, Woodhall and Linquist, and Young-Won and Udpa are all examples of the first derivative-based edge detectors [49][50][45][1] (see section 2.3.2).

The main advantages of these algorithms are their simplicity, ability to estimate edge orientation, and speed. The most important of their disadvantages are their sensitivity to noise and inaccuracy to localise the edges [56]. Since these algorithms do not utilise any preprocessing and postprocessing techniques, such as NMS, the recognised edges are often thick [56].

2.4.2 Zero Crossing Edge Detectors

This category of edge detection algorithms uses the second derivatives to detect edges. The edges are where the values of the second derivative of the image are zero. For two dimensional functions, the second derivative can be approximated by the Laplacian as in Equation (2.5). The Laplacian equation has an interesting property that it is rotationally invariant, i.e.,

the chosen direction does not matter as the sum of the second derivatives is the same in any two orthogonal directions. An early second derivative-based edge detection algorithm is the Marr-Hildreth edge detector [57]. Its major advantage over the first-derivative-based edge detectors is its good localisation ability. The main disadvantage of this algorithm is its sensitivity to noise and also its inability to calculate edge orientation which is required by most edge thinning and linking techniques [56]. Therefore, the edges produced by the algorithm are often broken or thick.

2.4.3 Gaussian-based Edge Detectors

The edge detectors in this category use the Gaussian filter as a noise removal operator. The Gaussian filter was originally proposed by Marr and Hildreth in 1980 [57], however it was used as a smoothing operator by Shen and Castan to reduce the noise for the first time in 1993 [58]. The Laplacian of Gaussian (LoG) operator was the first edge detector that utilised the Gaussian filter. If an image is first blurred by use of this filter and then the Laplacian operator is applied, the resulted algorithm will be the LoG operator. This algorithm is less sensitive to noise than the first and second derivative-based edge detectors but it cannot detect edge orientation because of using second derivatives [59]. In LoG, the Gaussian and the Laplacian operators are usually implemented by a single operator which is the Laplacian of the Gaussian function as follows:

$$\nabla^2 G_\sigma(x,y) = -\frac{1}{\pi\sigma^4}\left(1-\frac{x^2+y^2}{2\sigma^2}\right)e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.24}$$

Canny [10] proposed an edge detector which is widely considered as a standard edge detection algorithm and still outperforms many recently developed algorithms [60]. The Canny edge detector [10] determines the edges of an image based on an optimisation process to find a maxima of the gradient magnitude of an image which has been smoothed by the Gaussian filter. Equation (2.25) shows an example of a 5×5 Gaussian filter with standard deviation $\sigma = 1.4$:

$$G_{1.4}(x,y) = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} \tag{2.25}$$

In the Canny algorithm, after applying the Gaussian filter, an edge detector (for example Roberts, Prewitt or Sobel) is used to calculate the first derivative of the image in the horizontal and the vertical directions. The edge orientation estimated by a first derivative-based edge detector is rounded to one of four directions, i.e, vertical, horizontal and two diagonals. Then NMS and hysteresis thresholding techniques are applied for a good localisation. The Canny algorithm has been revised many times since it was proposed. Typical steps of the Canny edge detector are as follows [10]:

1. Smoothing an image to reduce the noise using the Gaussian filter;

2. Calculating the gradient magnitude and direction for each pixel in the image;

3. Using non-maxima suppression (NMS) algorithm to suppress non-maxima edges through which there is no pixel among its two neighbours in the gradient direction with larger gradient magnitude. If there is not such pixel, the pixel is marked as an edge, otherwise as the background; and

4. Applying hysteresis thresholding.

A revised version of the Canny edge detector introduced the concept of minor and major edges and it changed step three of the original algorithm [61]. In this method, after determination of the gradient magnitude and gradient direction at each pixel, the following steps are pursued:

1. If the gradient magnitude of a pixel is larger than those of its two neighbours in the gradient direction, the pixel will be marked as a major edge. If the gradient magnitude of the pixel is larger than those of all neighbours in any direction, it will be marked as a minor edge. Otherwise, it will be marked as a background pixel.

2. Partition the minor edges at the branch or connection points.

3. Remove all edges in a partition if there is no major edge in that partition; then rename the minor edges that are adjacent to a major edge as major edges.

4. Removing the weak major edges by hysteresis thresholding.

This algorithm is very popular because of its good detection, good localisation, and single response to an edge. However this algorithm suffers from the detection of the edges at the junction points because of using a Gaussian filter [62][61]. In Figure 2.7, an example with this problem is shown.

Unfortunately, the Canny edge detector cannot detect the high frequency edges either, such as the edges on a one pixel width line, and so double edges appear in these areas as shown in Figure 2.8 [63].

Jeong and Kim [64] proposed a method to automatically determine the optimum size or scale of the Gaussian filter for each pixel. In the proposed method, the scale of the Gaussian filter is determined by minimising a predefined energy function. This method sets the scale of the Gaussian filter at a large value in uniform intensity regions and at a small value in ridges where the intensity sharply changes.

The Gaussian-based edge detectors have been used widely to date because of their desirable features in noisy environments, however some researchers have demonstrated that the edge detectors that use this filter do not give satisfactory results. They suffer from edge displacement, removed edges, and also false edges [62].

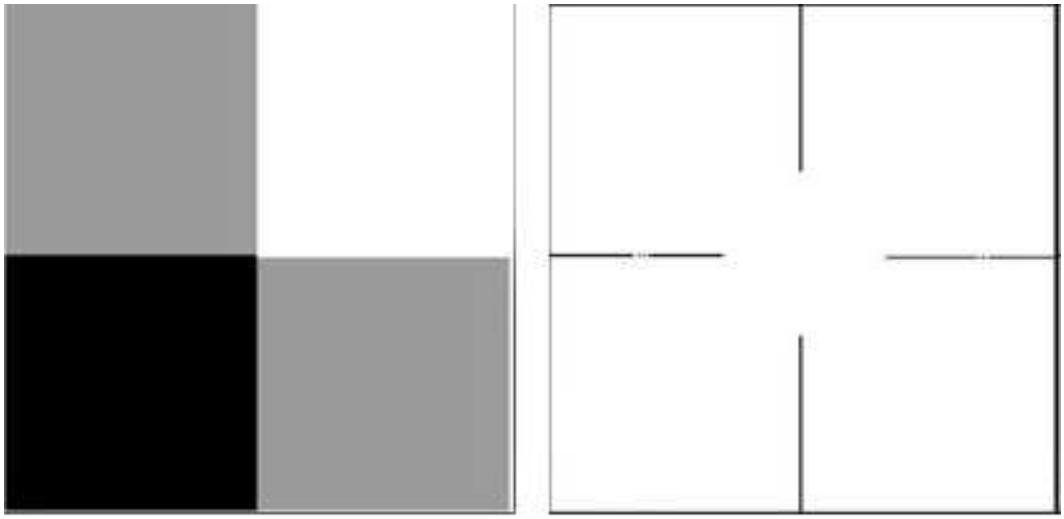


Figure 2.7: Applying the Canny edge detector to an image with four well-defined homogeneous regions after smoothing with a Gaussian filter (with standard deviation 11)

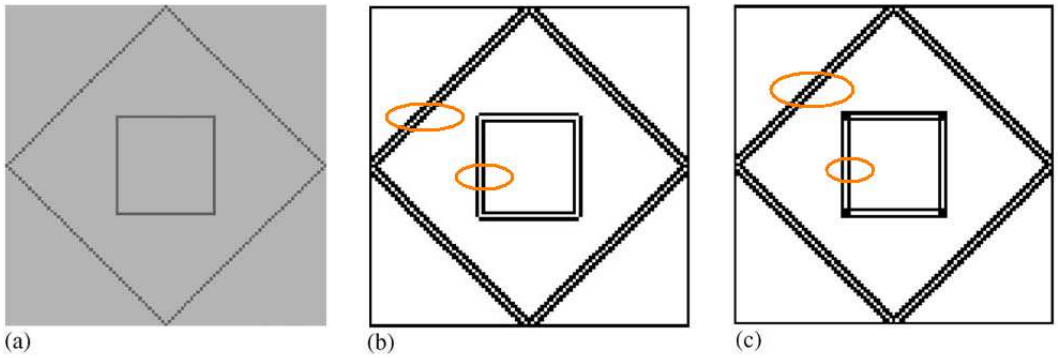


Figure 2.8: Double and speckle edges. (a) a simple image with high frequency edge information, (b) detected edges by the Canny edge detector and (c) detected edges by Sobel.

2.4.4 Multi-scale based Edge Detectors

Multi-scale methods generate several scales of an image using different scales of the Gaussian filter to improve the detection of edges in noisy images. These methods are based on multi-scale theory [65]. The main challenges in these methods are how to select a proper range for the scale, how to combine the resulting outputs corresponding to different scales, and how to adapt to the level of noise in an image. There are many works in this area, but we only briefly review those closely related to this thesis in this subsection.

Some approaches based on the Canny edge detector have been proposed in the literature that use multi-scale theory [65] to detect the edges [66] [67]. Schunck [68] proposed an edge detection algorithm which utilised the Gaussian filters at multiple scales of resolution. The first step of this algorithm is based on the Canny algorithm. In this step, the Canny algorithm is first applied with a large scale of the Gaussian filter. The edge map resulting from this step contains large ridges corresponding with the strong and major edges. Then, the scale of the Gaussian filter decreases and the Canny algorithm is applied again. The edge map resulting from this step will contain both large and small ridges which correspond to the major and weak edges. In this step, a few unwanted edges may appear in the edge map. The gradient magnitude maps resulting from different scales from the chosen range are multiplied together to produce a composite edge magnitude image.

The ridges at the smallest scale corresponding to major edges will be strengthened by the ridges at larger scales. The strength of ridges, which do not appear at the smallest scale, will be reduced because of their absence at larger scales. Accordingly, in the composite edge magnitude image, the strength of the ridges corresponding to major edges are much higher than that of the ridges corresponding to weak edges. Then, the NMS technique is applied using the gradient orientation obtained from the largest scale. The scales of the filters in this algorithm are different by

a factor of two. However, this algorithm is insensitive to noise, but it is sensitive to the number of filters. Furthermore, this algorithm may lose the important details which appear at smaller scales [62].

Bergholm [69] used the scale space theory and an edge focusing technique to combine edge information resulting from a coarse-to-fine scale. This algorithm uses a rule-based method for estimating scale parameters. Both the Canny and Marr-Hildreth edge-detectors can be used in edge focusing. In this algorithm, a Gaussian filter with a large scale is first used to smooth and then an edge detector is applied followed by an adaptive thresholding technique. Since edges are displaced by at most two pixels per a unit change in the scale, the edges can be precisely localiased by tracking them over decreasing scales. Accordingly, the output resulting from an edge detector with a specified scale is used to forecast the location of the edges resulting from the edge detector with the next smaller scale. A Gaussian filter causes an image to be blured and, accordingly, the localisation ability of an edge detector becomes poor. The main goal of this algorithm utilising the edge focusing technique is to reverse the effect of bluring. This algorithm first starts with the edges recognised at the coarse scale and then gradually focuses them back to their exact locations in the fine scale. The edge focusing technique encounters several problems. The most important problem of this algorithm is that how the starting and ending scales are determined for the Gaussian filter. Although Bergholm suggested a range between 3 and 6 for the starting scale, he did not discuss the end scale. Additionally, this algorithm produces broken and discontinuous edges [62].

Lacroix [70] proposed another scale space-based edge detector which tracks edges from a fine scale to a coarse scale. This method is based on the Canny algorithm and considers three different scales: σ_0 (the smallest and the detection scale), σ_1 (the intermediate scale), and σ_2 (the largest and

the bluing scale). The intermediate resolution, σ_1 is computed as follows:

$$\sigma_1 = \sigma_0 + \frac{\sigma_2 - \sigma_0}{3} \tag{2.26}$$

This method suffers from poor localisation as it determines the location of the edges in the coarsest resolution. This method also does not explain how the smallest and largest scale should be determined.

Goshtasby [71] proposed an algorithm to modify the scale-space representation of an image. In this representation, an image is created by recording the signs of pixels after applying the LoG operator. In this method, the scale-step size is determined by an adaptive way as follows. The results of convolution of an image at two scales are layed on each other. If more than two regions with the same sign lay on top of each other, some edges between these two scales are missing. Accordingly, an intermediate scale between these two scales must be considered. Otherwise, there is no need to consider an intermediate scale. In this way, Goshtasby's method solves the problem of choosing the step size but needs a large amount of memory in order to store the three-dimensional (3D) edge images and still produces broken edges [62].

2.4.5 Statistical-based Edge Detectors

Several edge detection methods have been proposed in the framework of statistics. Bovik et al. [72] proposed a statistical-based edge detector based on several non-parametric tests for edge detection in noisy images without any objective assessments. To understand how an statistical-based edge detector works, a brief overview of the method proposed by Bovik et al. is summarised here. In this method, two $n \times n$ square neighbourhoods are considered for each pixel where n is odd as shown in Figure 2.9. Most edge detection algorithms first compute a quantity which measures the edge magnitude and then compare this quantity to a threshold value. In statistical methods, another parameter, δ is used for sensitivity control which is called the minimum allowable edge height. For the mask in Figure 2.9, let

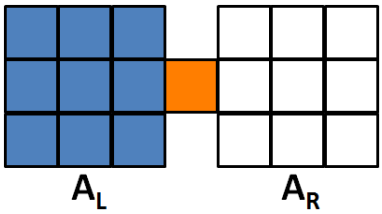


Figure 2.9: The window mask for detecting vertically oriented edges in the method of Bovik et al. [72]

A_L be the set of the intensity of pixels in the left neighbourhood and A_R be the set of the intensity of pixels in the right neighbourhood:

$$A_L = \{X_1, X_2, \ldots, X_{n^2}\} \tag{2.27}$$

$$A_R = \{X_{n^2+1}, X_{n^2+2}, \ldots, X_{2n^2}\} \tag{2.28}$$

Let α_i and β_i be defined as

$$\alpha_i = \begin{cases} X_i + \delta & X_i \in A_L \\ X_i & X_i \in A_R \end{cases} \tag{2.29}$$

$$\beta_i = \begin{cases} X_i - \delta & X_i \in A_L \\ X_i & X_i \in A_R \end{cases} \tag{2.30}$$

This method arranges different hypothesis tests based on different assumptions about the two sets, $\{\alpha_i\}$ and $\{\beta_i\}$ in order to decide whether there is any gray level difference (or edge). This edge detector rejects the edges with a height less than δ. Bovik et al. defined a test statistic as a function of δ-modified observations ($\{\alpha_i\}$ and $\{\beta_i\}$) which its value is compared with a threshold value, T. Based on this comparsion, the method classifies the pixel as an edge or not. In this method, the Wilcoxon and linear rank tests, as two non-parametric statstical tests, are used.

Huang and Tseng [73] proposed a statistical-based edge detector established upon the likelihood ratio test. This edge detector is computationally expensive and sensitive to noise. Aron and Kurz [74] utilised

the analysis of variance (ANOVA) tests for their proposed edge detector. Hou and Koh [75] proposed an edge detector including two steps: detection and localisation. In the detection step, a robust one-way analysis of variance is used to detect edges and then a robust contrast test are used to localise them. Lim and Jang [76] quantitatively compared the performance of three edge detection algorithms based on the Kolmogorov-Smirnov, Wilcoxon, t-tests. They showed that the localisation accuracy of the algorithm is higher in synthetic images corrupted by impulse noise when the Kolmogorov-Smirnov test is applied. However, the algorithm performs better in the images corrupted by a small amount of Gaussian noise when the Wilcoxon test and t-test are used.

Another algorithm was proposed by Lim [11] based on the rank-robust order (RRO) test. This method considers eight possible $r \times r$ windows which are spatially partitioned into two regions. The RRO test is used by this method to decide whether there are any significant differences in gray level value between two adjacent pixel neighbourhoods around a given pixel. Lim showed that the localisation accuracy of his proposed algorithm is higher than other statistical-based methods and more robust to two different types of noise, i.e., Gaussian and impulse noise.

Although statistical-based edge detectors are often insensitive to noise, the recognised edges are often thick. These methods are often data-driven unlike traditional methods which are model-driven. Therefore, they are not able to find the orientation and magnitude of edges. Accordingly, edge thinning techniques cannot be applied to the resulting edge maps.

2.4.6 Transform-based Edge Detectors

Two major types of this category of edge detectors are based on the Fourier and wavelet transforms. To describe the wavelet transform, we first review the differences between these two transforms. Let $f(t)$ be a signal in the time domain and t be a moment in time. When the Fourier trans-

form is applied to the signal, a function $F(\omega)$ is obtained which takes an input as a frequency and returns a complex number. This number shows the strength of the frequency in the original signal f. The strength of the cosine and sine parts of the frequency are shown by the real and imaginary parts of this complex number respectively. To obtain the Fourier transform of an input signal, it should repeatedly be correlated to the cosine and sine waves. In the Fourier transform, its coefficients will be high when the signal is high valued. If the signal is close to zero, the Fourier transforms coefficients will be low. Since the domains of the sine and cosine functions are between $-\infty$ and $+\infty$, the Fourier analysis encounters a big problem. The effect of a frequency is analysed as if it was spread over the entire original signal while this is not the case for most signals. Fourier analysis can determine which frequencies exist in a signal, but not where they are [77]. The wavelet transform addresses this problem. The output of the wavelet transform is a set of functions $W_s f(t)$. These functions describe the strength of a wavelet which is scaled by factor s at time t. Since a wavelet covers only a short period, its effects are restricted to a small interval of time surrounding t. The output of the wavelet transform will contain the strengths of the frequencies of a signal at time t unlike the Fourier transform that will give information about the strength of a frequency in the whole of a signal. A wavelet is defined as a function, $\Psi(t)$ whose average is zero between $-\infty$ and $+\infty$:

$$\int_{-\infty}^{+\infty} \Psi(t)dt = 0 \tag{2.31}$$

A wavelet function can be expanded by a scale factor s and translated by parameter u giving: $\Psi_{s,u}(t) = \frac{1}{\sqrt{s}}\Psi(\frac{t-u}{s})$. Wavelet functions, unlike the sine and cosine functions, quickly move toward zero as time t approach $-\infty$ or $+\infty$. For a 2D signal, a wavelet function can be defined as:

$$\Psi_{s,u_x,u_y}(x,y) = \frac{1}{s^2}\Psi\left(\frac{x-u_x}{s}, \frac{y-u_y}{s}\right) \tag{2.32}$$

Mallat and Zhong [78] showed that the multi-scale Canny edge detector is equivalent to the detection of local maxima of the wavelet transform of an image. In fact, the first derivative of the Gaussian filter, ∇G is a wavelet function which is known as the first derivative Gaussian wavelet ($\Psi(x, y) = \nabla G$). So,

$$W_s I(x, y) = I \star \Psi_s(x, y) = s(\nabla(I \star G_s(x, y)) \tag{2.33}$$

where $I(x, y) \in L^2(R^2)$.

Heric and Zazula [79] proposed an edge detection algorithm which uses the Haar wavelet transform. Shih and Tseng [80] presented an algorithm which is a combination of a gradient-based edge detector to detect edges and a wavelet multi-scale operator to track them. Despite the success of wavelet transform edge detectors, they have a limited ability in dealing with directional information [81] and produce broken edges [82].

2.4.7 Soft Computing Approaches

In this section, some of the most important soft computing techniques such as fuzzy sets, artificial neural networks, genetic algorithm and ant colony optimisation, which have been used to solve edge detection problem, are briefly reviewed.

A fuzzy-based edge detector was introduced by Kim et al. [83] to automatically adjust the threshold value which is used to remove weak edges. This method is faster than the Canny edge detector, but false edges may be detected. The quality of detected edges was comparable to the conventional gradient-based edge detectors. The competitive fuzzy edge detector (CFED) [6] is another fuzzy-based edge detector which classify the edges into six different classes and then determines which class an edge belongs to. This method usually generates speckles in a particular texture area. It can be used to enhance the detected edges.

An artificial neural network-based edge detector was proposed by Pinho

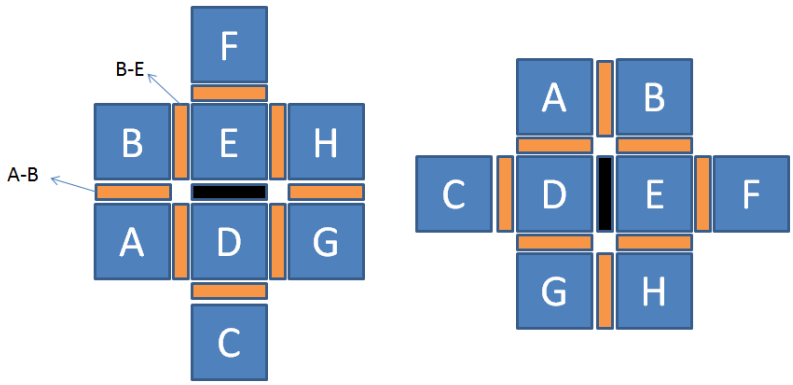


Figure 2.10: Some local features are extracted to be used as input values of an ANN (adapted from [84]).

and Almeida [84]. In this method, some local features are first extracted and then these features are used as the input values of the neural networks. As shown in Figure 2.10, an inter-pixel "crack" (as shown by black and orange rectangles in Figure 2.10) is used to represent an edge. This kind of representation is unambiguous in representing the edges. In this method, eight pixels generate nine different inter-pixel cracks which form the input of the ANN which classifies the central black crack. This crack can be either horizontal or vertical as can be seen in Figure 2.10. A training set must be constructed to train the ANN. This method can extract the edges more smoothly than other methods, but some edges may be missed. Another method based on cellular neural networks was proposed by citeauthorEfficientedgedetectionindigitalimagesusingacellularneuralnetworkoptimizedbydifferentialevolutionalgorithm [8]. This method considers a wider area in comparison to other methods, but it needs to be trained.

Furthermore, Bhandarkar et al. [9] applied a genetic algorithm (GA) to detect edges. This method represents the chromosomes in the population as a binary array. Zero or one represents a non-edge pixel or an edge pixel respectively in the chromosome's gene. Figure 1.1(e) shows the resulting

edge map when the GA-based edge detector is applied to the Lena image. As can be seen from this figure, there are thick and broken edges in the edge map.

2.4.8 Coloured Edge Detectors

Most of the edge detectors reviewed in the previous sections cannot be applied to coloured images, because these images represent a pixel through a vector in their colour space. The colour space can be modelled by many models, such as RGB (red, green, blue), CMYK (cyan, magenta, yellow, and key or black), YIQ (Y luma information, in-phase, and quadrature), and HSL (hue, saturation, and lightness). Each of these models has its own properties in colour science [1]. Some of the coloured edge detectors use an extension of a gray level edge detector for colour images [85]. In these detectors, an edge may not be detected when it is located in the neighbourhood of the pixels which have the same value in any of colour components. Vector order statistics colour edge detectors, such as VR (vector range), vector dispersion (VD), minimum vector range (MVR), were proposed to detect edges in colour images [86]. The vector order statistics colour edge detectors are famous methods for processing coloured images. They utilise order statistics which play a significant role in signal analysis.

2.4.9 Edge Linking Techniques

One of the most serious drawbacks of traditional edge detection algorithms is broken edges. Therefore, many edge linking methods as post-processing algorithms have been proposed to improve this disadvantage of the algorithms. The edges in a broken area may contain important information. A few techniques have been proposed to compensate for the broken edges. These techniques are listed as follows:

1. Mask-based edge linking techniques: they draw some reasonable lines between the endpoints of the broken edges. Hajjar and Chen

[87] proposed a method which uses a mask to obtain these lines. This method is very fast and simple, but the extraction of incomplete edge structure is its main disadvantage. Accordingly, these techniques are very inaccurate [3].

2. The Hough transform: one of the most commonly used techniques is the Hough transform [88][1]. The Hough transform is an image transform technique to detect geometric features, such as straight lines, circles, ellipses, and generally some fixed shapes [89].

3. Sequential edge linking (SEL): an application of a sequential searching technique is to link the broken edges [90]. Multi-resolution sequential edge linking (M-SEL) technique is also commonly used as an edge linking technique [91]. This technique uses a multi-resolution image pyramid to better guide the SEL technique at higher resolution through global edge information obtained in lower resolutions. These two techniques consider at most two endpoints of the broken edges to be compensated and also the local information of the original image is not well analysed to link the broken edges. Therefore, the accuracy of these algorithms is low [3].

4. ACO-based linking techniques: Some other proposed techniques use ant colony optimisation to solve this problem [3]. Although, this method is very slow and takes a long time (around 60 seconds) only for the edge linking step in the edge detection algorithms, their accuracy is often low because of producing false edges.

2.4.10 Objective Evaluation of Edge Detectors

Until 1999, there was no objective method for the evaluation of edge detectors and they were often compared to each other through a subjective assessment method. Shin et al. used an object recognition system for an objective comparison of edge detectors [92]. The Hausdorff distance [93]

was utilised to recognise an object in this system. The Hausdorff distance is defined as follows:

$$H(A, B) = max(h(A, B)), h(B, A)) \tag{2.34}$$

$$h(A, B) = max_{a \in A} min_{b \in B} ||a - b|| \tag{2.35}$$

where $A = \{a_1, a_2, ..., a_p\}$ and $B = \{b_1, b_2, ..., b_q\}$ are two finite sets of points, and a_i and b_j correspond to the locations of two edges in two images A and B. Here, $||a - b||$ is the distance between two points a and b. The function $h(A, B)$ is called the directed Hausdorff distance from A to B.

Since 1999, many measures have been proposed to objectively evaluate the performance of edge detectors. Ground truth images are required for most of these measures. The receiver operating characteristic curve (ROC curve) is one of the commonly used techniques for an objective evaluation. This technique was used by Bowyer et al. [94]. The ROC curve in an edge detection problem is a plot of the fraction of true positive edges (TPR = true positive rate) versus the fraction of false positives edges (FPR = false positive rate). In this method, area under the curve (AUC) is the *traditional* metric for comparing ROC curves resulting from different edge detectors.

Martin et al. presented another method which is called precision vs recall curves (PR) [95]. The *precision* (vertical axis) means the proportion of the edges resulting from an edge detector which are true positives rather than false positives and the *recall rate* (horizontal axis) means the proportion of true positives that are recognised rather than missed. F-measure is a measurement to be extracted from these curves as Equation (2.36).

$$F_\beta = \frac{precision \times recall}{(1 - \alpha)precision + \alpha recall} \tag{2.36}$$

where $\alpha = 0.5$.

Pratt's figure of merit (PFOM) is another objective measure proposed

by Pratt [14]. It is defined as:

$$R_{PFOM} = \frac{1}{max(I_I, I_A)} \sum_{i=1}^{I_A} \frac{1}{1 + \beta d(i)^2} \tag{2.37}$$

where I_I and I_A indicate the number of ideal and actual edge points in the ground truth and the actual images respectively, $d(i)$ is the distance of the pixel i in the actual edge map from nearest ideal edge point in the ideal edge map, and β is a constant scale factor which is equal to $\frac{1}{9}$. This measure is an index to compute the localisation accuracy of an edge detection algorithm. This measure is commonly used for an objective comparison [11].

Another approach was proposed by Shin et al. [96]. In this method, the performance of edge detection algorithms was compared when they were applied to a motion detection task. This method is called an indirect method for the assessment of edge detectors.

Moreno et al. [97] presented four other measures, namely, completeness, discrimination ability, precision, and robustness ability of an edge detector. The completeness is a measure that shows the ability of an edge detector to detect all possible edges in noiseless images. Equation (2.38) shows how to calculate this measure.

$$R = \frac{1}{m} \sum_{i=1}^{m} \phi(d_i)) \tag{2.38}$$

where ϕ is a radial decaying function ranging from 0 to 1, and m is the number of ideal edges, d_i is the distance of the pixel i in the actual edge maps from the nearest ideal edge point in the ground truth.

The discriminability is the ability of an edge detector to discriminate between important and not important edges [97]. The measure is computed as:

$$DS = \frac{\sum_{i=1}^{n} e_i \phi(d_i)}{\sum_{i=1}^{m} \phi(d_i)} - \frac{\sum_{i=1}^{n} e_i (1 - \phi(d_i))}{\sum_{i=1}^{n} (1 - \phi(d_i))} \tag{2.39}$$

where DS is the discriminability measure of an edge detector, n is the number of detected edges and e_i is the edge magnitude of edge i in the edge map.

The false alarm rejection measure (precison measure) is the ability of an edge detector to detect edges as close as possible to ideal edges [97]. This measure (FAR) is as follows:

$$FAR_{measurement} = \frac{1}{n}\sum_{i=1}^{n}\phi(d_i) \tag{2.40}$$

where n is the number of the edges in the resulted edge maps.

Finally, the robustness measure is the ability of an edge detector to reject the noise. This measure can be calculated by Equation (2.41).

$$MSE_{measurement} = \frac{1}{rc}\sum_{i=1}^{r}\sum_{j=1}^{c}(e_{ij} - e')^2 \tag{2.41}$$

where r and c are the dimensions of the edge map, and e_{ij} and e'_{ij} are the magnitude of an edge at location i, j of noisy and noiseless images respectively.

2.5 Particle Swarm Optimisation

The main goal of this section is to review particle swarm optimisation (PSO) as a global optimisation method and some of its concepts, such as position and velocity update equations, and topology as an information sharing mechanism. This section provides a brief overview of evolutionary computation algorithms, such as evolution strategy, evolutionary programming, genetic algorithms and genetic programming followed by several swarm-based algorithms, such as ant colony optimisation, stochastic diffusion search and gravitational search algorithms.

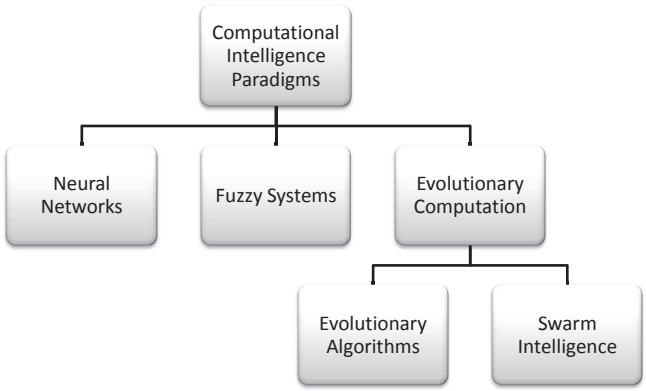


Figure 2.11: Computational intelligence paradigms [98].

2.5.1 Computational Intelligence and Evolutionary Algorithms

Computational Intelligence (CI) is a successor of artificial intelligence. CI experts mainly consider biological, psychological and evolutionary inspirations from nature for implementation [98]. Figure 2.11 shows three primary branches of CI, i.e., neural networks, fuzzy systems and evolutionary computing including swarm intelligence and evolutionary algorithms. Neural networks are developed based on biological counterparts in the human nervous system. Evolutionary computing similarly heavily draws on the principles of Darwinian evolution observed in nature. Human reasoning using fuzzy, is modelled by fuzzy systems.

Evolutionary computation (EC) encompasses a number of problem-solving methods designed to simulate evolution. These methods are all population-based and rely on a combination of random variation and selection to solve problems. Several different approaches exist within EC, including evolution strategy (ES), evolutionary programming (EP), genetic algorithms (GAs) and genetic programming (GP) [99]. These algorithms have been inspired by the theory of Charles Darwin who first popularised

the modern theory of evolution. The fundamental principle underlying evolution is optimisation, where the goal is survival of the species. However, it does not mean that the EC methods can be applied only to optimisation problems. The EC paradigms help to solve problems which were previously hard to solve.

The term of evolutionary algorithms refers to a category of algorithms that can all work in the general evolutionary framework [100]. The exact form of the representations and operators, as well as the relationship between the sizes of parent and offspring populations, define the specific instance of EA, e.g. EP, ES, GA, or GP.

Consider a population of n individuals, $P(t) = (x_1(t), x_2(t), ..., x_n(t))$ at time t, where each $x_i \in S$ represents a potential solution of a problem in the search space S. Let $f(x)$ be a function that determines the quality of a solution, called the fitness function. The fitness of the whole population can thus be expressed as $F(t) = (f(x_1(t)), f(x_2(t)), ..., f(x_n(t)))$. Given arbitrary parameters n, λ, Θ_r, Θ_m and Θ_s, the general EA framework is as Algorithm 2.1. The parameters Θ_r, Θ_m and Θ_s (called strategy parameters) are the recombination, mutation and selection operators respectively. The parameter n is the size of the parent population; $n + \lambda$ denotes the total population size (parents plus offspring) after the recombination and mutation operators are applied.

Evolutionary programming was introduced by Fogel in the context of evolving finite state-machines to use in the prediction of time series [101]. This algorithm did not use recombination; it exclusively relied on mutation. Later evolutionary programming was extended to include more general representations, including ordered lists (to solve the Travelling Salesman Problem) and real-valued vectors for continuous function optimisation [102]. Modern EPs are characterised as EAs without recombination, thus relying exclusively on mutation and selection. When EPs are applied to real-valued optimisation problems, they use normally-distributed mutations and usually evolve their strategy parameters concurrently.

Algorithm 2.1 Pseudo code for the general framework for describing EAs [98]

1: $t = 0$
2: $P(t)$=initialise(n)
3: $F(t)$=evaluate($P(t)$, n)
4: **repeat**
5: $P'(t)$=recombine($P(t)$, Θ_r)
6: $P''(t)$=mutate($P'(t)$, Θ_m)
7: $F(t)$=evaluate($P''(t)$, λ)
8: $P(t+1)$=select($P''(t)$, $F(t)$, n, Θ_s)
9: $t = t + 1$
10: **until** stopping criterion is met

Evolutionary strategy (ES), introduced in 1968, are usually applied to real-valued optimisation problems [103]. The ES individuals use both mutation and recombination operators, and search both the search space and the strategy parameter space simultaneously. The parent and offspring population sizes usually differ (the offspring population at least as large as the parent population).

The genetic algorithm (GA) was originally described by Holland [104]. It is another instance of EAs. The emphasis of GA is usually recombination, with mutation treated as a background operator. Only a brief overview of the GA will be presented here; the reader could refer to [105] for an in-depth treatment of the subject.

The canonical GA uses a binary format to represent the genotypes. The genotype is converted using a mapping function into the equivalent phenotype, which is an element in the search space. A simple example will clarify this process. Assume that GA is used to locate the minimum of function $f(x) = x^2 - 10x + 25$. It is known that the minimiser is located in the interval $[0, 10)$. Assume that a 16-bit representation is used to represent the values of x, so that the genotypes of the elements in the population are

16-bit strings. Given such a 16-bit representation, b, the equivalent phenotype can be obtained using

$$x = 10 \times \frac{1}{2^{16}} \sum_{i=0}^{15} 2^i b_i$$

where b_i denotes the value of bit i in the bit string b. The factor 10 is to scale the value from its initial range $[0, 1)$ to the range $[0, 10)$. The minimiser of this function is 5, so the genotype representation, using 16 bits, is 1000000000000000. Although the minimiser has an exact representation in this example, this is not always the case. By increasing the number of bits in the genotype greater accuracy can be obtained.

The notion of using separate genotypic and phenotypic representations used to be one of the defining differences between GAs and other type of evolutionary algorithms. This difference has blurred somewhat with the advent of non-binary coded GAs. For example, a GA could use real-valued numbers to represent population members [106], using arithmetic crossover [107] instead of binary crossover. Other possible representation include permutations and tree-based representations, usually utilised in GP. GP is another EA-based method to find a computer program that performs a desired task [108]. This method has been applied to a variety of areas from scheduling to evolving classification systems.

2.5.2 Swarm-based Algorithms

Swarm intelligence was inspired by collective behaviours which exist in nature, e.g., fish schooling, ant colonies, birds flocking and animal herding. These kinds of behaviours were introduced and employed in artificial intelligence by Beni and Wang [109] in 1989, in the context of cellular robotic systems. It is a well-known fact that an ant is not clever (and almost blind) but it can do striking things in a colony when it cooperates with other ants. An agent like an ant cannot do big jobs but if some ants

can work together, they will be able to do remarkable jobs, for example building a big colony.

Several swarm-based algorithms have been proposed in the literature. Ant colony optimisation [110], particle swarm optimisation [111], stochastic diffusion search [112] and gravitational search algorithms [113] are examples of them. A main difference between them is the method of information transfer and sharing among agents or individuals of the swarm, but all of them are inspired based on collective behaviours available in nature.

A category of swarm-based algorithms is ant colony optimisation (ACO) which is based on simulating the activities of an ant colony. This method was invented by Colorni et al. [114]. The ant colony optimisation method is suitable for problems which need to look for paths to goals. In this method, there are some finite artificial ants that as agents locate optimal solutions by moving through a search space including all possible solutions. Real ants produce pheromones to direct each other toward resources while moving in their environment. The simulated ants or agents similarly register their locations and best solutions that they find, so that more ants can locate better solutions in the next iterations [110]. ACO has been applied to many combinatorial optimisation problems, ranging from folding proteins to routing vehicles [110].

Another swarm-based method for optimisation problems is stochastic diffusion search (SDS) which is a population-based probabilistic global search [112]. This method suits problems whose objective functions can be separated into multiple independent partial functions. In this method, each agent maintains a hypothesis which is iteratively examined by evaluating a randomly selected partial objective function encoded by the agent's current hypothesis. This algorithm consists of two phases: test and diffusion. In the first phase, each agent tests its hypothesis. In the second phase, the agents exchange information about hypotheses via one-to-one communication [112].

The gravitational search algorithm (GSA) is inspired based on Newton's Gravity law and the idea of mass interactions. The GSA algorithm uses Newtonian physics theory and its agents are a collection of masses. In GSA, all agents are considered as separated and independent masses. Based on the gravitational force, each mass in the system can see the location and situation of other masses. Thus, the gravitational force is a method of transferring information among different masses [113].

Particle swarm optimisation (PSO) is a global optimisation algorithm for dealing with problems in which a best solution can be represented as a point in an n-dimensional space. All particles are randomly placed in this space and initialised with an initial velocity, as well as a communication channel between the particles [115][18]. The particles then move through the solution space and are evaluated according to some fitness criterion after each iteration. Their positions are adjusted based on their velocity to move towards those particles within their communication grouping which have better fitness values. In the next section, particle swarm optimisation will be described in more detail as it is directly used in this thesis.

2.5.3 Particle Swarm Optimisation

PSO is a global optimisation method, proposed by Kennedy and Eberhart in 1995 [116]. This method was inspired by social behaviors of animals and biological populations [116]. In fact, it is a simulation of a simplified social model like bird flocking and fish schooling. PSO was originally an optimisation method for continuous nonlinear functions, i.e., the search space is continuous and decision variables are encoded into real numbers. However, several discrete versions of the algorithm have been proposed in literature [117][118][119]. In PSO, there is a population of finite individuals which are called particles. Some advantages of PSO in comparison to other heuristic search algorithms such as GA are ease of its implementation, its fewer parameters for adjustment, its fewer operators and high

rate of its convergence [120]. PSO has been utilised in many areas, such as training neural networks [121], optimising power systems [120], fuzzy control system [120], robotics [122], radio and antenna design [123] and computer games [124].

In the basic PSO (BPSO), there is a population containing m potential solutions which are represented as m particles. These particles move through n-dimensional search space. The position of the i^{th} particle is represented as vector $\vec{X}_i = (x_{i1}, x_{i2}, ..., x_{in})$ in an n-dimensional search space. The position of each particle is changed according to its own experience and that of its neighbours. Let $\vec{X}_i(t)$ denote the position of particle P_i at time t. The position of P_i is changed in each iteration of the BPSO algorithm by adding a velocity $\vec{V}_i(t)$ to determine a new position as shown in equation (2.42).

$$\vec{X}_i(t+1) = \vec{X}_i(t) + \vec{V}_i(t+1) \tag{2.42}$$

The velocity vector is updated based on three values: the effect of current motion or velocity, particle memory influence, and swarm influence [125].

$$\vec{V}_{i,j}(t+1) = w\vec{V}_{i,j}(t) + C_1 Rand_{1j}(\vec{X}_{pbest_i,j} - \vec{X}_{i,j}(t)) + C_2 Rand_{2j}(\vec{X}_{leader,j} - \vec{X}_{i,j}(t)) \tag{2.43}$$

where $Rand_{1,j}$, and $Rand_{2,j}$ are random variables with uniform distributions. Here, w denotes inertia weight which is employed by the BPSO algorithm to control the impact of the previous velocity of particle P_i; parameters C_1 and C_2 are learning factors that represent the attraction of a particle toward either its own success or that of its neighbours respectively. Coefficients C_1 and C_2 are called self and swarm confidences respectively. In equation (2.43), $\vec{X}_{pbest_i}$ denotes the best position of the i^{th} particle. $\vec{X}_{leader}$ is the position of a particle which is used to guide other particles toward better regions of the search space. This particle called the leader. The leader of each particle is specified based on a neighbour-

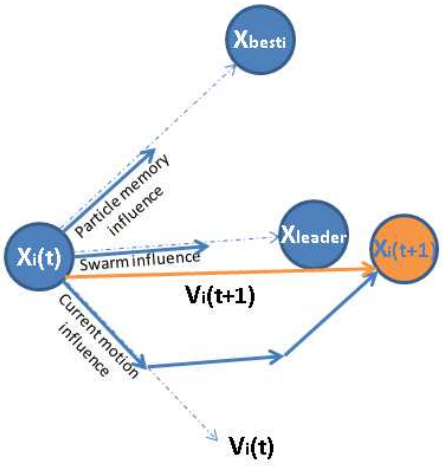


Figure 2.12: Depiction of the position and velocity in PSO algorithm adopted from [126].

hood structure [115]. In this equation, $w\vec{V}_i(t)$, $C_1 Rand_{1j}(\vec{X}_{pbest_i,j} - \vec{X}_{i,j}(t))$, and $C_2 Rand_{2j}(\vec{X}_{leader,j} - \vec{X}_{i,j}(t))$ are correspondingly called current motion, particle memory influence, and swarm influence. Figure 2.12 depicts the position and velocity in PSO algorithm.

BPSO has been designed in two steps, i.e., randomly initializing a population, and iteratively updating velocities and positions as shown in Algorithm 2.2.

2.6 Swarm Intelligence for Edge Detection

As mentioned earlier, there are four main categories in swarm intelligence, i.e., Ant Colony Optimisation (ACO) [110], Particle Swarm Optimisation (PSO) [111], Stochastic Diffusion Search (SDS) [112], and Gravitational Search Algorithms (GSA) [113]. There are a number of researches on using ACO and GSA for edge detection [7][127] but there are only a few works on edge detection by PSO and SDS. The main goal of this section is to provide a brief overview of swarm-based edge detection algorithms.

Algorithm 2.2 Pseudo code for the BPSO algorithm

```
Initialise PSO population randomly (position and velocity of each particle)
repeat
    for all particles do
        Evaluate the fitness value of the particle
        Find in the particle neighbourhood, the leader particle
        Calculate particle velocity according to (2.43)
        Update particle position according to (2.42)
    end for
    Update leader or leaders
until maximum iterations exceeded or minimum error criteria attained
Select best particle in the population and decode it as the best solution
```

2.6.1 ACO-based Edge Detectors

The ACO-based edge detectors [7] use a number of ants which move on an image to make a pheromone matrix. Each entry of the pheromone matrix in ACO shows an amount of food to be found by ants. In these methods, each entry of the pheromone represents the edge information at each pixel of the image. The ants move on the image based on local intensity values and after several iterations the final edge map is constructed based on the pheromone matrix.

Zhuang et al. [128] proposed an ACO-based feature extraction algorithm in 2008. In this algorithm, a perceptual graph is used to represent the relationship between neighbouring pixels. The algorithm utilises this graph in order to extract image features. This graph is built by an ant colony system. The results showed that the ant colony-based system can effectively extract features in simple and semi-complex digital images.

Ouadfel [129] presented an edge detector which utilises Markov Random Fields (MRF) together with an ant colony-based segmentation method

in order to detect edges on magnetic resonance images (MRI). He compared the results with genetic algorithm and a simulated annealing based method and showed that the performance of his proposed method is higher in the MRI images.

Lakehal [130] proposed a new method to detect interest points in an image. This method utilises an ACO-based system to detect the objects' centres as the interest points. This algorithm first uses an agent-based edge detection algorithm to detect edges and then applies an ACO method to extract features and classify the objects. This algorithm depends on the agent based system and accordingly, cannot be used alone for feature extraction.

Mirzayans et al. [131] utilised another ant colony-based system for feature extraction in the images containing simple shapes, such as rectangles, squares, triangles, crosses and circles. They showed that although the algorithm is efficient in noisy images, it is not applicable to extract features in real images and requires more investigation.

Wang [132] presented another method in which ants have a lifetime and reproduce. The main idea in this method is that there are a large number of ants in the areas which are close to image features. In this method, there are two classes of ants. The first class of ants are fixed at the points of interest in an image and the second class of ants are mobile on the image. The proposed method performs well in noiseless images but noise reduces its performance in comparison to the Sobel edge detector [133].

Etemad and White [133] presented a swarm-based technique for feature extraction and segmentation. In this algorithm, two different types of pheromone are used to share information among ants. They compared the performance of the algorithm with the Sobel and Canny algorithms in the images corrupted by impulse noise. The results showed that their proposed method outperforms Sobel and Canny in such images.

Wong et al. [7] improved the performance of the Canny algorithm to compensate for broken edges. This method first utilises the Canny algo-

rithm to detect edges and then applies an ant colony-based system to find the shortest path between two endpoints of broken edges. This algorithm has many parameters which must be tuned for every image. Although this algorithm can enhance weak edges very well, it is very slow and there are many false edges in the resulting edge maps [12]. Wong et al. did not investigate the performance of the algorithm in the images corrupted by Gaussian noise.

2.6.2 Gravitational Approach to Edge Detection

Lopez-Molina et al. [127] proposed a gravitational search-based approach to detecting edges. In this method, each pixel in an image is considered as a celestial body with a mass represented by its gray level. Therefore, according to the law of universal gravity, each pixel as a celestial body puts forces onto its neighbours and receives forces from its neighbours. The sums of all these forces along the vertical and horizontal directions are used to compute the edge magnitude and orientation. They examined the proposed algorithm and compared with Sobel and Canny in a few standard clean and noisy images corrupted Gaussian noise. Its performance was comparable to the Sobel and Canny in the images with Gaussian noise but there are many noise spots when the algorithm applies to the images with impulse noise.

2.6.3 PSO-based Edge Detectors

The first PSO-based edge detector has been developed during this thesis in 2009. In [22], we developed a PSO-based edge detector which utilised a homogeneity edge detection operator to estimate the magnitude of the edges along a continuous curve. In this paper, an encoding scheme and a fitness function was developed to evaluate a continuous curve lying on the edges in several real and synthetic images. We subjectively compared it with the Sobel edge detector. Although the results showed that the pro-

posed method performed better than Sobel in clean and noisy syntactic images corrupted by impulse noise, it could not perform well in real images and was very slow. In [24], we developed another fitness function and encoding scheme in order to apply PSO to the detection of edges in real noisy images. The performance of the PSO-based edge detector was good in comparison with Sobel and Canny in the images corrupted by Gaussian and impulse noise. However, the comparison showed that its performance is lower than a statistical-based edge detector proposed by Lim [11].

Alipoor et al. [134] used PSO to find an optimum edge filter. A synthetic image and its edge map are used for training. The proposed method was applied to a simple synthetic image and subjectively compared with the Sobel edge detector. The results showed that the proposed method can perform well in simple synthetic clean and noisy images. The authors did not investigate its performance on real images.

Aghamohammadi et al. [135] applied our proposed algorithm in [24] to detect the cracks on solar cell panels which convert the energy of light into electrical energy. Since the cracks on the surface of the solar cell panels cause the performance of the panels to reduce, they proposed an automated inspection system based on our proposed method [24] in order to detect the cracks.

2.7 Summary and Discussion

As described in this chapter, there are two main categories of edge detection algorithms, i.e., soft computing and non-soft computing-based algorithms. The most important non-soft computing-based algorithms include first and second derivative-based, Gaussian filter-based, statistical-based, scale space-based and transform-based edge detectors. First derivative-based edge detectors are very simple and can calculate edge orientation, but they are very sensitive to noise and are inaccurate. The main advan-

tage of the second derivative-based edge detectors is their high accuracy in edge localisation. However, they perform poorly at corners and they cannot find the orientation of the edges. Accordingly, the recognised edges are thick. The Gaussian-based edge detectors utilise Gaussian filters to reduce noise. These algorithms are rather insensitive to noise, but often displace or remove the edges. These algorithms also usually produce false edges and malfunction at corners. The statistical- based detectors are somewhat insensitive to noise, but cannot find edge orientations. The scale space based edge detectors work based on the scale space theory. They can perform well in noisy images and are rather fast. However, they have difficulty in choosing filter size and in combining edge information from different scales. Although the transform-based edge detectors, such as the wavelet-based edge detection algorithms, precisely detect edges in noisy images, they suffer from producing broken edges.

There are many edge detectors based on soft computing techniques, such as fuzzy sets, ant colony optimisation, genetic algorithms, and neural networks. Fuzzy-based algorithms perform well in noisy images and have few parameters. However, these parameters should manually be set for every image. False edges are also frequent in these algorithms. ACO-based edge detectors can enhance weak edges and work well in noisy images. However, these algorithms are very slow and require many parameters to be set manually. Artificial neural network-based detectors suit noisy images and are almost accurate, but they need to be trained for a particular domain, which means they are domain specific. Genetic algorithm-based edge detectors are accurate but sensitive to noise and very slow.

Particle swarm optimisation (PSO) is a population-based meta-heuristic method for solving global optimisation problems based on social-psychological principles. Compared with some heuristic methods such as genetic algorithms, the most important advantages of PSO are ease of its implementation, fewer operators, a limited memory for each particle and high speed of convergence. As PSO has a high capability to optimise noisy functions

[108][17], it has been successfully applied to many problems in noisy environments, such as image segmentation [21] and vision tracking [20]. Surprisingly, PSO was not applied to tackling the edge detection problem before this thesis started in 2009, and still has not suficiently analysed for edge detection. The execution time of an edge detection algorithm is very important in many applications. Since PSO does not use the gradient information of the function being optimised, it has a high capability to optimise noisy functions. These features of PSO make it to be a good candidate for edge detection in noisy images. This thesis will investigate the capacity and potential of PSO for edge detection.

Chapter 3

Image Sets

This chapter presents the image sets used in the experiments arranged during this research, and our justification for their inclusion in the experiments. This project will use two noiseless benchmark image sets along with their ground truth images (pre-defined edge map of the images) and one image set including synthetic shapes generated specially for this project. All images will be corrupted through adding noise (impulse and Gaussian) to their noiseless images. These image sets have been selected based on their frequent usage for the comparison of edge detectors in noisy and illuminated images from the literature. The first set has been provided by South Florida University [136]. The well-known Lena and rubbish-bin images are included in this set. The second set has been provided by University of Cordoba (UCO) and includes some artificial images and their artificial ground truth [137].

3.1 Real Image Set of South Florida University

This set contains 28 images that have been provided by South Florida University [136]. Some samples of this image set are shown in Figure 3.1. They are usually used for a subjective comparison of edge detectors in the literature. These images will be used in our experiments in different noise

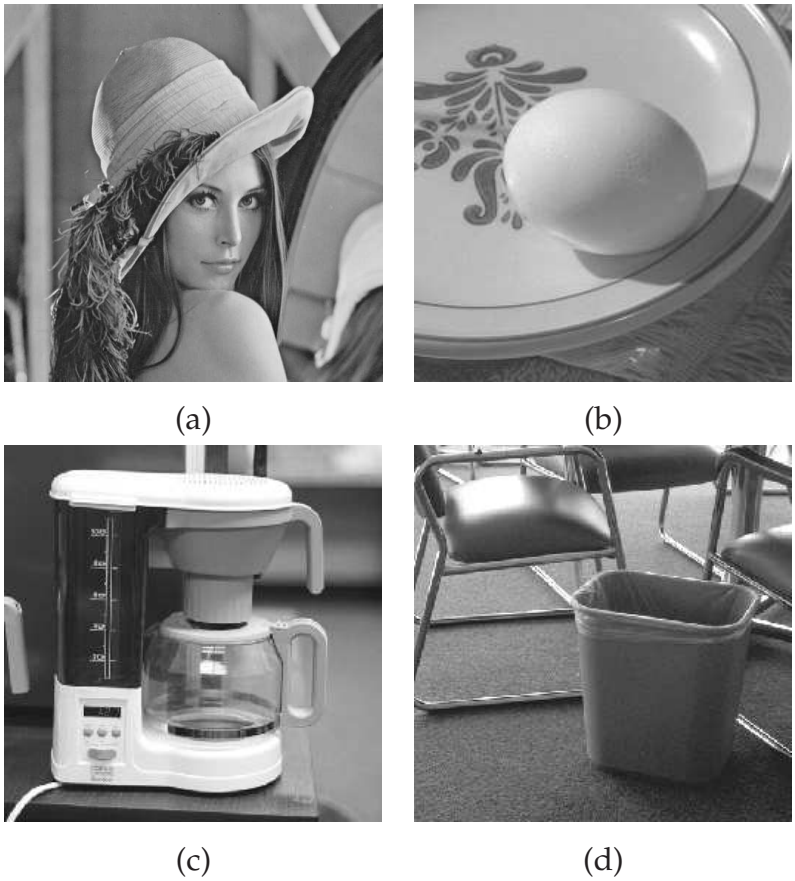
levels. All of these images are noiseless. They are corrupted by two different kinds of noise, Gaussian and impulse noise as will be described in Section 3.3. In this image set, there are several images with illuminated areas. Since the illumination phenomena causes edges to become weak, most edge detection algorithms cannot perform well in such areas. For example, the Lena image in Figure 3.1(a) has several illuminated areas on the hat and the bar in left side of the image. For the egg and rubbish-bin images as can be seen in Figure 3.1(b) and (d), there are illuminated areas around the boundary of the egg and the rubbish-bin. For the car image in Figure 3.1(e), the edges around the car are illuminated.

3.2 Standard Artificial and Real Image Set of University of Cordoba

This set contains some artificial and real images and their artificial ground truth images which have been provided by University of Cordoba (UCO) [137]. Some examples of this image set with illuminated areas are shown in Figure 3.2. These images are commonly used in literature to investigate the localisation accuracy of edge detectors. Their ground truth images contain thin and single pixel width edges. These images are usually used for objective comparison.

3.3 Images with Impulse and Gaussian Noise

An image may be contaminated by noise in its transmission or acquisition. In fact, noise is any unwanted information that corrupts an image. Noise comes into an image from different resources. There are many different types of noise which can be classified into three classes: additive noise, multiplicative noise and impulse noise. Since additive noise and impulse



(a) (b)

(c) (d)

Figure 3.1: Example images from South Florida University database for a subjective comparison [136] (a)-(d) the original Lena, egg, coffee maker and rubbish bin images which are commonly used for a subjective comparison of edge detection algorithms.

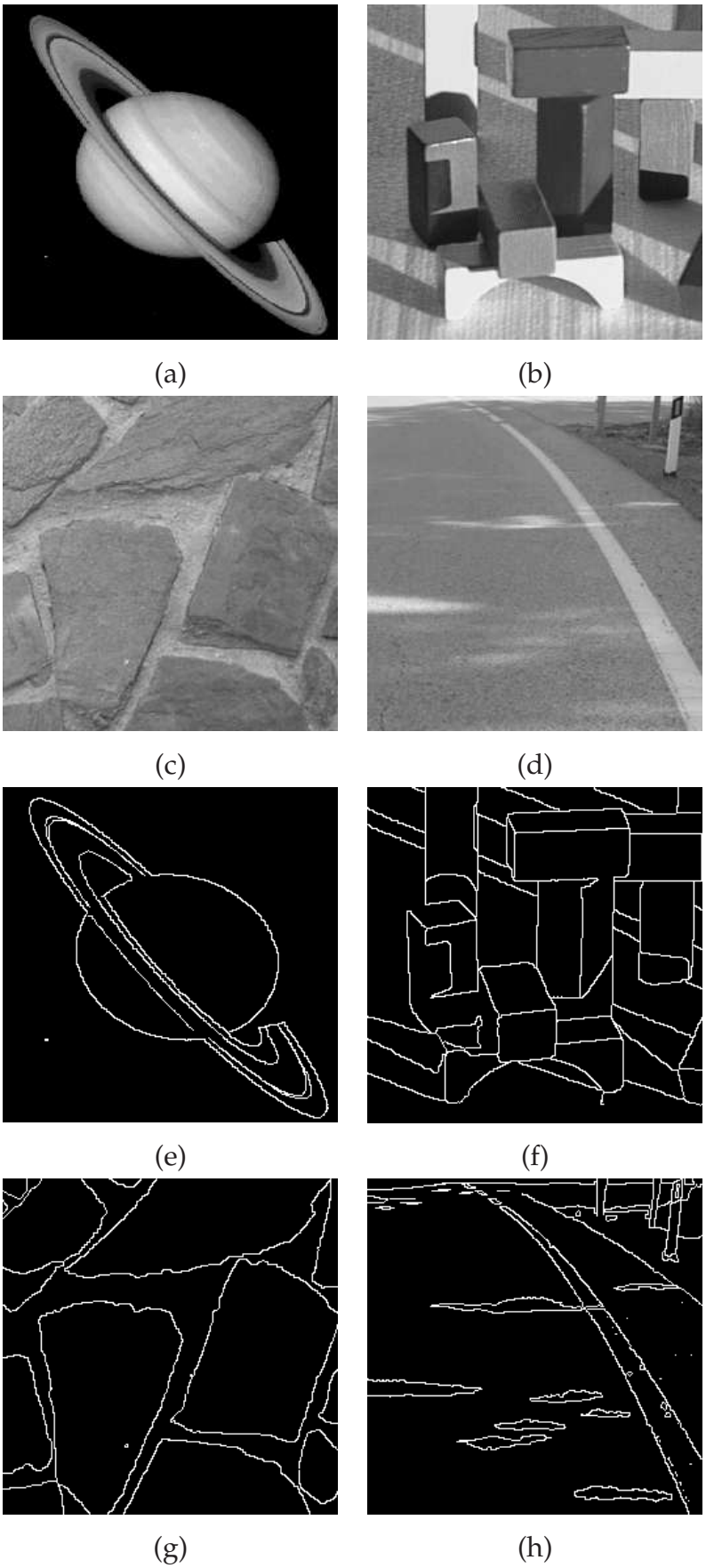


Figure 3.2: (a)-(d) Some samples from UCO university database and (e)-(h) their ground truth images provided by the university [137].

noise are more commonly used in image processing and analysis, we provide a brief overview of these two types of noise in this section. These two types of noise are also used in this thesis.

3.3.1 Additive Noise

Let $I(x, y)$ be the original intensity of pixel (x, y) in image I, and let $I_N(x, y)$ be the corrupted version of the pixel intensity. Additive noise can be modelled as:

$$I_N(i, j) = I(i, j) + \eta(i, j) \tag{3.1}$$

where $\eta(i, j)$ is a noise function which returns a random value generated by an arbitrary distribution. This function is independent of the intensity of pixels in the original image. Typically, the noise function is a symmetric function about zero, namely, this function does not alter the average brightness of the image. This model is usually used to model the thermal noise in photo-electronic sensors. For the Gaussian noise in image processing, $\eta(i, j) = \sigma\varepsilon(i, j)$ where $\varepsilon(i, j)$ is a Gaussian random variable with zero mean and unit variance, and σ^2 is the variance of the Gaussian noise.

3.3.2 Impulse Noise

In impulse noise, the intensity of a pixel is altogether replaced by a random variable (with probability P_n) or is unmodified (with probability $1-P_n$). In salt-and-pepper noise, this random variable is either 0 (black) or 1 (white) with a same probability (0.5). The impulse noise can be modelled as follows:

$$f(i, j) = \begin{cases} r_2 & r_1 < P_n \\ I(i, j) & otherwise \end{cases} \tag{3.2}$$

where P_n is the probability of noise occurrence, r_1 is a uniform random variable between 0 and 1, r_2 is a uniform random variable in the range of

pixel gray levels.

Since all images described in the last two sections are clean, we add impulse and Gaussian noise in various levels to the noiseless images in order to investigate the performance of edge detection algorithms.

3.3.3 Summary

In this chapter, we presented the image sets which will be used in our experiments. These image sets includes noiseless benchmark images which are commonly used in edge detection. Some of these images are illuminated which causes edge detection algorithms cannot perform well and broken edges are appeared in the resulted edge maps. Therefore, we will use these images in our experiments to examine the performance of our newly developed algorithms. Since these images are noiseless and our main goal in this research is to detect edges in noisy images, we will corrupt all images by two different types of noise model, i.e., Gaussian and impulse noise model which are commonly used to simulate noise in image processing.

Chapter 4

Novel Edge Detection Algorithm Robust to Noise using PSO

This chapter firstly proposes a novel constrained optimisation model for detecting continuous edges in noisy images. Then two PSO-based algorithms are developed to find good solutions. These two algorithms use two different constraint handling methods: penalising and preservation. They are compared with a revised version of Canny as a Gaussian filter-based edge detector and the robust rank order (RRO)-based algorithm as a statistical-based edge detector on two sets of images with different types and levels of noise.

4.1 Introduction

The edges of objects in an image contain important information that can be used as low-level features in image analysis and computer vision systems [2]. The main goal of an edge detection algorithm is to provide the continuous contours of the object boundaries. In practice, accurately detecting these continuous contours is very hard and time consuming especially when noise exists in the image [1].

Many algorithms have been proposed using various different paradigms

such as curve fitting [138], optimization of a criterion [10][139], image transforms [82][140] statistical testing [11] and soft computing [141][8] to detect edges for different applications. The selection of an edge detection algorithm for a particular application depends on its performance in a variety of environmental conditions (such as illumination and noise) and the requirements of the system of interest (such as real time ability, continuity of edges, thinness of edges and scale insensitivity).

The commonly used algorithms for detecting edges in noisy images include Gaussian-based [62], statistical-based [11], and scale space-based [142] edge detectors. The Gaussian-based algorithms often malfunction at corners and curves [56] and establish double edges in areas with high frequencies of information. They also displace edges and produce false edges [62]. These methods use a Gaussian filter as a smoothing technique to reduce noise, which often causes edges to be weak and broken as a side effect [61][15]. Although there are several algorithms that utilise sharpening techniques to reduce these side effects, they suffer from producing jagged edges [143].

Several statistical-based methods have been proposed to detect edges in noisy images, such as the t-detector, Wilcoxon detector, and robust rank-order (RRO) detector [11]. These methods are insensitive to noise because of considering a large neighbourhood for each pixel in comparison to other edge detection methods. They use a statistical test to check whether an $r \times r$ window can be divided into two subregions with significant differences in intensities. If there is a significant difference between them, the pixel is classified as an edge otherwise a non-edge. These algorithms are data-driven and do not function based on an edge model, thus they cannot recognise edge magnitudes which are required for edge thinning and linking. Therefore the produced edges are often thick [144].

Another group of algorithms use the scale space theory [65] to generate different scales of an image and produce an image pyramid. These methods operate on a large area of an image through generating different

scales of the image. While the operation on the low resolution images allows them to be very fast, the difficulty of choosing the size of the filters with combining edge information from different scales restricts their application. Some of these methods such as wavelet-based edge detectors utilise an image transform to detect edges. Although these methods are insensitive to noise, they suffer from producing broken and jagged edges [82] [62].

Several techniques have been proposed to compensate for broken edges, such as sequential edge linking (SEL) [90], multi-resolution SEL (M-SEL) [91] and the Hough transform. The simplicity and high speed are the main advantages, but they are not necessarily accurate due to not considering the global structure of edges. While the Hough transform can operate well on the images containing just simple shapes (such as straight lines or circles), it often does not deal well with objects having complicated shapes [3]. Snake-based methods are another type of these techniques. They utilise an active contour model to detect an object boundary [145]. These methods need to have a priori knowledge about the boundary and are very slow [146].

Most of the edge detection algorithms described above use a convolution of an image with an $n \times n$ matrix, where usually $n \leq 5$ to reduce the computation time. This means that the information from a limited area is considered in these algorithms to mark a pixel as an edge. The area size has a strong effect on accuracy such that if the area size is increased, the algorithm will be less sensitive to noise but at the same time, the localisation accuracy will be lower. If we want to increase the localisation accuracy of the algorithm, we need to consider all edge patterns. However, this will substantially increase the computation time ($t(n) = \frac{(n^2-1)^n}{2} = O(n^{2n})$) [16]. Therefore a heuristic algorithm is required to explore a large area to overcome the noise and consider the global structure of the edges to reduce broken edges in a reasonable time.

Particle Swarm Optimisation (PSO) is a population-based meta-heuristic

method for solving global optimisation problems based on social-psychological principles, introduced by Kennedy and Eberhart in 1995 [116]. Compared with some heuristic methods, such as genetic algorithms, the most important general advantages of PSO are ease of its implementation, few operators, a limited memory for each particle and high speed of convergence [120]. PSO is very stable and efficient in noisy environments [147]. Its comparison with evolutionary algorithms has shown that PSO has a high capability to optimise noisy functions [148][108][17] and it has been successfully applied to many problems in noisy environments, such as image segmentation [21] and vision tracking [20]. PSO has good potential for edge detection in noisy images, but surprisingly, it has not been sufficiently analysed for tackling edge detection problems.

4.1.1 Chapter Goals

This chapter aims to develop new PSO based approaches to edge detection in noisy images with the goal of extracting continuous edges and reducing the number of broken edges. The main goals of this chapter are as follows:

- Developing a fitness funtion and a particle encoding for PSO to detect edges in noisy images
- Exploring a large area and examine all possible edge patterns in order to increase the localisation accuracy of edge detection and extracting the global structure of edges in order to detect the edges with greater continiuity
- Comparing the new PSO-based method with the modified version of the Canny algorithm proposed in [60] and the RRO algorithm proposed in [11] on two sets of noisy images.

The rest of this chapter is organised as follows. Section 4.2 describes a revised version of Canny along with the RRO-based edge detector. The

new PSO-based algorithms are presented in Sections 4.3 and 4.4. Sections 4.5 and 4.6 presents discussion on experimental results followed by a summary in Section 4.7.

4.2 Edge Detection Algorithms

Edge detection as low-level feature detection is one of the critical elements in image processing. The main function of edge detection is to find the boundaries of image regions based on properties such as intensity and texture [11]. Although many algorithms have been proposed to detect edges in noisy images, this section only briefly reviews a modified version of Canny [61] and RRO [11] as they are very commonly used in edge detection in noisy images and will be compared with the new approaches proposed in this chapter.

4.2.1 Revised Versions of the Canny Algorithm

The Canny edge detector as a Gaussian filter-based algorithm operates as an optimisation process to find the maxima of the gradient magnitude of an image after the image is smoothed by a Gaussian filter to reduce noise [10]. This algorithm is very popular because it has a complete process of edge detection and has good localisation. This edge detector has been revised many times since it was first proposed. Its typical steps include applying a Gaussian filter to reduce noise, estimating the gradient magnitude and edge direction for each pixel of an image, using a non-maxima suppression (NMS) algorithm to suppress non-maxima edges, and applying a hysteresis thresholding technique to identify edges and link broken edges.

The size of the filter is very important in reducing noise and its size depends on the noise level. The Canny algorithm was revised by Jeong and Kim [64] by proposing an adaptive method in order to determine the

optimal filter size in noisy images. They suggested a standard and adaptive method to determine filter scale for edge detection for each area of an image. This method was extended from the optimal filter concept proposed by Poggio et al. [149] and the scale-space theory proposed by Witkin [150]. This method adaptively finds optimal filter scales for each pixel before extracting edge maps. Jeong and Kim defined an energy function as a function over continuous scale space as follows:

$$\begin{aligned} E(\sigma) = & \int\int \left((f - G \star f)^2 + \lambda \left| \nabla \frac{1}{\sigma(x,y)} \right|^2 \right) dxdy \\ = & \int\int \left((f - \left[\int\int \frac{1}{2\pi\sigma^2} e^{-\frac{\alpha^2+\beta^2}{2\sigma^2}} f(x-\alpha, y-\beta) d\alpha d\beta \right])^2 \right. \\ & \left. + \lambda((-\frac{\sigma_x}{\sigma^2})^2 + (-\frac{\sigma_y}{\sigma^2})^2) \right) dxdy \end{aligned} \quad (4.1)$$

where f is a signal, G is a Gaussian function and $\lambda = 60$ is a constant to control the smoothness ability of the Canny algorithm. It is obvious that when $\sigma \to 0$, the first term, i.e, $f - G \star f$ tends to a small value and when $\sigma \to +\infty$, it tends to a large value. This is reverse for the second term. Therefore, this energy function is minimised at somewhere in the search space, $0 < \sigma(x,y) < \infty$. The discrete form of this energy function can be estimated as follows:

$$\begin{aligned} E = & \sum_i \sum_j \left\{ \left(I(x,y) - \left[\sum_\alpha \sum_\beta \frac{1}{2\pi\sigma^2(i,j)} e^{-\frac{\alpha^2+\beta^2}{2\sigma^2(i,j)}} f(i-\alpha, j-\beta) \right] \right)^2 \right. \\ & \left. + \frac{\lambda}{\sigma^4(i,j)} \left[(\sigma(i+1,j) - \sigma(i,j))^2 + (\sigma(i,j+1) - \sigma(i,j))^2 \right] \right\} \end{aligned} \quad (4.2)$$

Jeong and Kim used a simple iterative successive over-relaxation method to obtain the optimal scale for each pixel on an image.

After applying the Gaussian filter and estimating the magnitude of the edges, the NMS technique is used as an edge thinning algorithm [151]. The NMS technique proposed by Canny chooses a pixel as an edge only

when the edge magnitude at that pixel is larger than the edge magnitude of the pixels in the direction of the gradient. Canny also proposed that it can be used as a post-processing algorithm along with any gradient operator to detect edges with a single pixel width. Most edge detection algorithms utilise a thresholding technique to identify edges and non-edges. The problem of producing broken edges is very common when a single global threshold value is used for edge thresholding [60]. Canny proposed the hysteresis thresholding technique, inspired by biological mechanisms, for detecting edges with greater continuity [152]. This technique usually utilises two threshold values (high and low) to tackle the problem of broken edges [10]. The hysteresis thresholding technique includes two main steps. In the first step, only the pixels whose gradient magnitudes are greater than the high threshold value are chosen as edges. In the second step, the pixels are detected whose gradient magnitudes are greater than the low threshold value and are adjacent to other edge pixels [55]. Manual determination of these two threshold values is very time consuming. Therefore, many unsupervised techniques have been proposed to determine these values [60]. Sen and Pal [60] proposed an automatic way in order to estimate these two threshold values. The low and high threshold values are computed as follows:

$$Threshold_{High} = 2\sigma\sqrt{f_u ln2} \tag{4.3}$$

$$Threshold_{Low} = \frac{1}{2} Threshold_{High} \tag{4.4}$$

where $f_u = s - l$, $s = \sum_{i=1}^{N}\sum_{j=1}^{N} a_{ij}^2$ and $l = \sum_{i=1}^{N}\sum_{j=3}^{N} a_{ij}a_{ij-2}$ where the coefficients a_{ij} correspond to the kernel of the Gaussian filter.

4.2.2 Robust Rank Order-based Edge Detector

Many edge detection algorithms have been proposed to deal with noise within the framework of statistics. These algorithms utilise a statistical test to detect an edge. An algorithm was recently developed by Lim based on

the robust rank-order (RRO) test [11]. This algorithm performs better than other statistical-based edge detectors such as Wilcoxon and t-test-based edge detectors [153][72] in terms of accuracy. The RRO algorithm considers eight different edge patterns for each pixel, each of which partitions the neighbourhood of the pixel into two sub-regions (gray and white) as can be seen in Figure 4.1. Lim considered the intensity of neighbours of each pixel on an image as 24 independent observations which are partitioned into $\mathfrak{G} = \{\mathfrak{g}_1, \mathfrak{g}_2, \ldots, \mathfrak{g}_{12}\}$ and $\mathfrak{W} = \{\mathfrak{w}_1, \mathfrak{w}_2, \ldots, \mathfrak{w}_{12}\}$ corresponding with the gray ($\mathfrak{G}$) and the white ($\mathfrak{W}$) subregions as shown in Figure 4.1. At least one of the window partitions shown in Figure 4.1 will be matched on an edge if there is an edge passing from the central pixel. The samples in $\mathfrak{G}$ and $\mathfrak{W}$ come from two continuous distributions, $A(\mathfrak{g} - \mu_{\mathfrak{g}})$ and $B(\mathfrak{w} - \mu_{\mathfrak{w}})$ with shifted parameters $\mu_{\mathfrak{g}}$ and $\mu_{\mathfrak{w}}$. Lim did not make any assumption about the nature of these two distribution. He defined the modified observations, α_i and β_i as follows:

$$\alpha_i = \begin{cases} \mathfrak{g}_i + \delta & \mathfrak{g}_i \in \mathfrak{G} \\ \mathfrak{w}_i & \mathfrak{w}_i \in \mathfrak{W} \end{cases} \tag{4.5}$$

$$\beta_i = \begin{cases} \mathfrak{g}_i - \delta & \mathfrak{g}_i \in \mathfrak{G} \\ \mathfrak{w}_i & \mathfrak{w}_i \in \mathfrak{W} \end{cases} \tag{4.6}$$

where δ is a parameter to define the minimum gray-level differential for the detection of an edge. In this method, the following hypothesises are tested:

$$H_0^{\uparrow} : \mu_{\mathfrak{g}} + \delta \geq \mu_{\mathfrak{w}} \;\; versus \;\; H_1^{\uparrow} : \mu_{\mathfrak{g}} + \delta < \mu_{\mathfrak{w}} \tag{4.7}$$

and

$$H_0^{\downarrow} : \mu_{\mathfrak{g}} - \delta \leq \mu_{\mathfrak{w}} \;\; versus \;\; H_1^{\downarrow} : \mu_{\mathfrak{g}} - \delta > \mu_{\mathfrak{w}} \tag{4.8}$$

Lim showed that since the distributions A and B are not identical in real world images, the Wilcoxon test is not an appropriate test. Accordingly, he considered the the RRO test on the modified observations, α_i and β_i to determine the existence of a significant difference in gray level

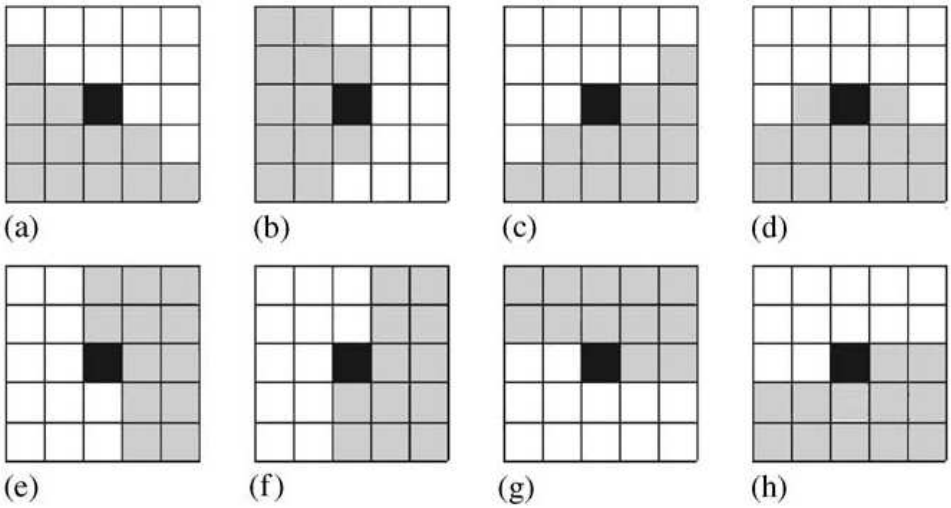


Figure 4.1: Partitioning the neighbourhood of a pixel in eight different ways where a gray sub-region represents partition $\mathfrak{G}$ and a white sub-region represents partition $\mathfrak{W}$ [11].

between them. Lim's method first considers the diagonal edge shown in Figure 4.1(a) and then the RRO statistic is obtained for testing $H_0^{\uparrow}$ against $H_1^{\uparrow}$ on α_i. In order to obtain the statistic, for each $\mathfrak{g}_i+\delta$, the number of $\mathfrak{w}_i$ in $\mathfrak{W}$, which is smaller than $\mathfrak{g}_i+\delta$, is counted. This number shows the position of $\mathfrak{g}_i+\delta$ and is denoted by $U(\mathfrak{W},\mathfrak{g}_i+\delta)$. Similarly, the position of each $\mathfrak{w}_i$ in $\mathfrak{W}$, $U(\mathfrak{G}+\delta,\mathfrak{w}_i)$ is found. Let $U(\mathfrak{W},\mathfrak{G}+\delta)$ be the mean of $U(\mathfrak{W},\mathfrak{g}_i+\delta)$, $U(\mathfrak{G}+\delta,\mathfrak{W})$ be the mean of $U(\mathfrak{G}+\delta,\mathfrak{w}_i)$, $V_{\mathfrak{G}+\delta}=\sum_{\mathfrak{g}_i\in\mathfrak{G}}(U(\mathfrak{W},\mathfrak{g}_i+\delta)-U(\mathfrak{W},\mathfrak{G}+\delta))$ and $V_{\mathfrak{W}+\delta}=\sum_{\mathfrak{w}_i\in\mathfrak{W}}(U(\mathfrak{G}+\delta,\mathfrak{w}_i)-U(\mathfrak{G}+\delta,\mathfrak{W}))$. So, the RRO statistic can easily be computed as:

$$U_\alpha=\frac{12(U(\mathfrak{G}+\delta,\mathfrak{W})-U(\mathfrak{W},\mathfrak{G}+\delta))}{2\sqrt{V_{\mathfrak{G}+\delta}+V_{\mathfrak{W}+\delta}+U(\mathfrak{G}+\delta,\mathfrak{W})U(\mathfrak{W},\mathfrak{G}+\delta)}} \tag{4.9}$$

Similarly, the RRO statistic can be obtained for testing $H_0^{\downarrow}$ against $H_1^{\downarrow}$ on β_i. So,

$$U_\beta=\frac{12(U(\mathfrak{W},\mathfrak{G}-\delta)-U(\mathfrak{G}-\delta,\mathfrak{W}))}{2\sqrt{V_{\mathfrak{G}-\delta}+V_{\mathfrak{W}-\delta}+U(\mathfrak{G}-\delta,\mathfrak{W})U(\mathfrak{W},\mathfrak{G}-\delta)}} \tag{4.10}$$

The null hypothesis, $H_0^{\uparrow}$ (or $H_0^{\downarrow}$) is rejected if $U^*=max(U_\alpha,U_\beta)$ has a large value. In the RRO-edge detector, a pixel is recognised as an edge when U^* is larger than a predefined threshold value, T_{sl} at a specified significant level sl.

The number of edge patterns used in this algorithm is more than that of the Canny edge detector which usually uses two or four edge patterns. Therefore, the localisation accuracy of these algorithms is often higher than that of Gaussian-based edge detectors in the images corrupted by noise. This algorithm also has only a few parameters that can be easily tuned by the user in order to detect edges in noisy images.

4.3 The New PSO-Based Approaches

The new methods proposed here are based on heuristically solving an optimisation problem. We wish to search for the best curve segment of a

given length which can be fitted on a continuous edge. This curve separates a region of an image into two subregions. All possible edge patterns would need to be examined in order to find the best curve such that it maximises the dissimilarity of pixel intensities of two subregions and maximises the similarities of the pixel intensities inside of each subregion. Therefore the search space in this optimisation problem is all possible curves which partition this region into two subregions. An encoding scheme is developed to represent these curves in this search space. To evaluate each curve, a fitness function is formulated to measure the dissimilarity between two subregions and the similarities of the pixels within each subregion. In this formulation, there are two simple constraints which should be satisfied. Two different PSO-based algorithms are proposed to handle the constraints. As will be shown, these algorithms have different efficiency in speed and effectiveness in accuracy. This section provides the details about the encoding scheme and the fitness function with the two constraints, followed by two PSO-based algorithms proposed in Section 4.4.

4.3.1 Encoding Scheme

Most edge detection algorithms convolve a convolution matrix on an image to calculate the edge magnitude only for a single pixel at a time and then utilise a thresholding technique to classify the pixel as an edge or non-edge. Therefore, a large number of pixels which have weak magnitudes may be falsely classified as non-edges or a few pixels which have high magnitudes may be falsely recognised as edges. It may cause a real continuous edge to be broken or some speckles to appear on a resulting edge map especially in noisy images. For that reason, the proposed method processes a collection of pixels at a time instead of a single pixel in order to extract the global structure of the real edge and considers a large area rather than a small one in order to attempt to overcome noise.

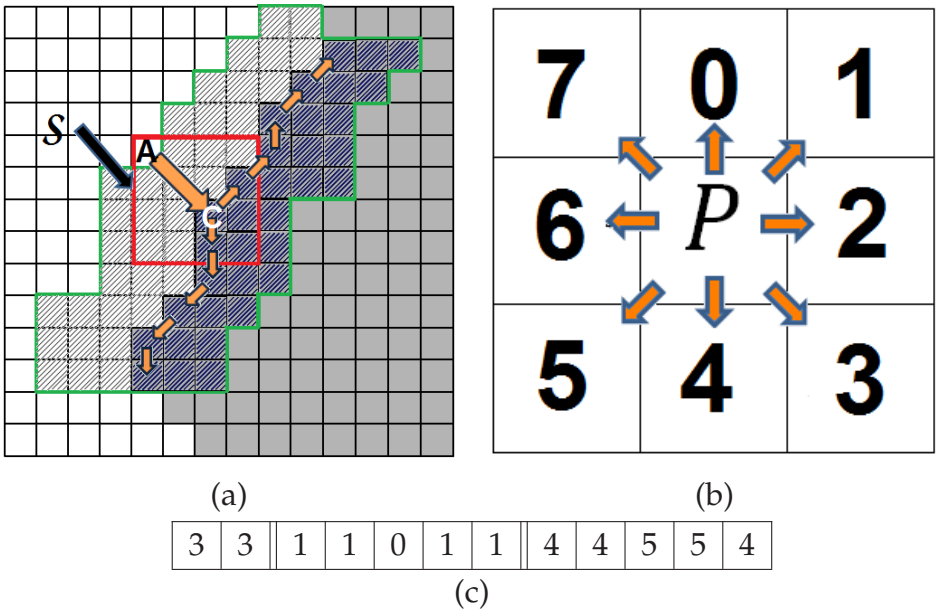


Figure 4.2: The particle encoding scheme. (a) An example of a curve with two regions; (b) eight movement directions from a pixel P; (c) the particle representing the curve with $L = 5$.

A continuous edge is a collection of consecutive pixels which divide an area of an image into two regions: the light and dark regions in Figure 4.2(a). The goal is to maximise the **interset** distance between the pixel intensities of the two regions and minimise the **intraset** distances within both regions. These consecutive pixels can be represented by a group of directional arrows (see the arrows in Figure 4.2(a)). Let pixel C be the middle pixel of the consecutive pixels on the continuous edge and $2L + 1$ be the number of the pixels on it. The red square in Figure 4.2(a) is the area which pixel C can be located in. The number of pixels along one side of this square is $SqrSize$. The relative position of pixel C with respect to pixel A (the upper left pixel of the red square) shows the offset of pixel C. With regard to the points explained above, a continuous edge can be represented by three components: the offset of pixel C and two sequences of movement direction sequences from pixel C representing the consecutive pixels. Let $\langle o_1, o_2 \rangle$ be the offset where o_1 and o_2 are integers ranging from 0 to $SqrSize - 1$, and $\langle m_1, m_2, \ldots, m_L \rangle$ and $\langle m_{L+1}, m_{L+2}, \ldots, m_{2L} \rangle$ be two sequences of movement direction sequences away from pixel C where m_i are integers ranging from 0 to 7. Each m_i shows the direction of movement from a pixel to one of the eight possible adjacent pixels in its neighbourhood along the continuous edge as shown in Figure 4.2(b). By changing the values of these components in the range of interest, all possible continuous edges sited inside of a region with the area of $(2L + SqrSize)^2$ can be represented by this encoding.

For example, the edge passing through pixel C, which is located inside the square in Figure 4.2(a), is encoded as shown in Figure 4.2(c). In this example, $SqrSize = 4$, $L = 5$, and $\langle m_1, m_2, \ldots, m_5 \rangle$ show the movement directions from the point C towards the top and $\langle m_6, m_7, \ldots, m_{10} \rangle$ towards the bottom. In this example, all striped pixels enclosed by the green lines are used to evaluate the curve, as follows.

4.3.2 A Fitness Function

The encoding scheme can represent all possible continuous edges with a minimum specified length ($2L + 1$) located in a specified area of an image. To evaluate each edge in this search space, a fitness function is introduced in this subsection. As illustrated later, the fitness value of each continuous edge is based on the average edge magnitude of all pixels along the edge. In this subsection, an edge magnitude measure and a curvature cost measure are also formulated followed by two constraints to detect continuous, smooth and thin edges in the images corrupted by noise.

Edge Magnitude Measure

Most edge detection algorithms use variant edge operators which have been developed based on different order derivatives to calculate edge magnitude, such as the first [10], second [57] and fourth derivatives [154]. These algorithms are often very sensitive to noise. However some of these operators work well in clean images. For this reason, we introduce a new approach to calculating edge magnitude in noisy images. The main idea is the optimisation of the interset distance between the regions separated by a continuous edge, and the intraset distances within the regions.

We propose eight ways of dividing the neighbourhood of each pixel of an image into two regions according to the eight possible movement directions, as shown in Figure 4.3. In each edge pattern in Figure 4.3, let D and L be the two sets of pixels corresponding to the dark and light regions respectively. It is obvious that if the interset distance between these two sets is great and their intraset distances are small, the edge magnitude will be great; and also if the interset distance is small and their intraset distances are great, the edge magnitude will be small. Hence, edge magnitude can be modelled as a function of these distances. We expect that the pixels of each region are close in intensity (low intraset distance), and the pixels of these two regions have the highest possible difference in intensity (high in-

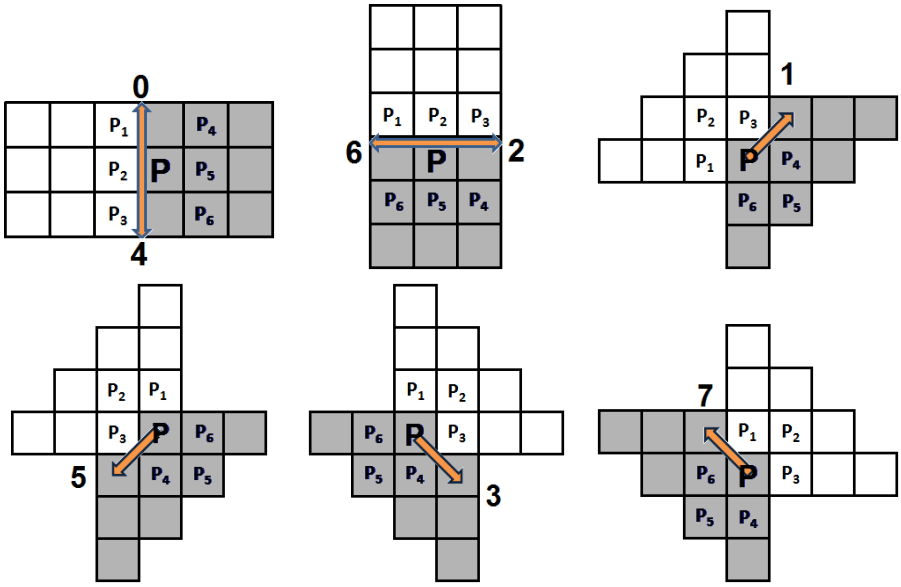


Figure 4.3: Eight ways of moving from pixel P to a neighbouring pixel.

terset distance). Therefore, we formulate the edge magnitude at pixel P in movement direction m, $EdgeMag_m(P)$ as Equation (4.11) to maximise the interset distance ($InterDis_m(P)$) between the regions and minimise the intraset distance ($IntraDis_m(P)$) within the regions. To avoid dividing by zero, the denominator is increased by 1.

$$EdgeMag_m(P) = \frac{InterDis_m(P)}{1 + IntraDis_m(P)} \tag{4.11}$$

Here P is a single pixel on a continuous edge and m is the movement direction from the pixel P to the next adjacent pixel on the edge. The interset distance is calculated based on Equation (4.12).

$$InterDis_m(P) = min\,(1, |avg_{m,d}(P) - avg_{m,l}(P)|/w_1) \tag{4.12}$$

Here $avg_{m,d}(P)$ and $avg_{m,l}(P)$ are the average intensities of the dark and

light regions corresponding to movement direction m for pixel P (see Figure 4.3), as calculated in $avg_{m,d}(P) = \frac{1}{n}\sum_{P_i \in D} I_{P_i}$ and $avg_{m,l}(P) = \frac{1}{n}\sum_{P_i \in L} I_{P_i}$; $n = |L| = |D|$ ($n = 9$ in Figure 4.3); I_{P_i} is the intensity of the ith pixel in the corresponding set; and w_1 is a weight factor.

The intraset distance $IntraDis_m(P)$ is a sum of pairwise subtractions of pixel intensities in a region as shown in Equation (4.13):

$$IntraDis_m(P) = \frac{1}{\binom{n}{2}}\left(\sum_{\substack{P_i,P_j \in D \\ i>j}} min\left(1, |I_{P_i} - I_{P_j}|/w_2\right) + \sum_{\substack{P_i,P_j \in L \\ i>j}} min\left(1, |I_{P_i} - I_{P_j}|/w_2\right) \right) \tag{4.13}$$

where w_2 is a weight factor.

NMS Factor for Edge Thinning

Non-maxima suppression (NMS) is one of the most important edge thinning techniques [2]. It extracts a local maximum of the edge magnitude along the direction of the gradient vector and suppresses non-maximal edges. Here, the $EdgeMag_m$ of a pixel on a continuous edge for each direction m is compared to the $EdgeMag_m$ of pixels P_1, P_2, ..., P_6 (as shown in Figure 4.3) on both sides of the edge. The NMS factor in each direction is the number of these neighbouring pixels whose edge magnitudes in the same direction are lower than the edge magnitude of the pixel:

$$NMS_m(P) = |\{P_i | i \in \{1, \ldots, 6\}, EdgeMag_m(P_i) < EdgeMag_m(P)\}| \tag{4.14}$$

where $|\cdot|$ is the cardinality of a set and $\{P_1, \ldots, P_6\}$ are the particular neighbours of the pixel P as shown in Figure 4.3. The value of NMS is an integer ranging from 0 to 6. The NMS factor in direction m is larger when P is a local maxima in that direction.

The NMS factor in conjunction with $EdgeMag_m(P)$ is used to indicate the total edge magnitude of a pixel lying on a thin edge in direction m. If

the edge direction is not estimated accurately, it may cause some real edge pixels to be removed by the NMS algorithm and broken edges to appear on the edge map. Therefore, a non-maxima edge should not necessarily be removed. The proposed method does not remove the non-maxima edge, but reduces the edge magnitude of the non-maxima edge by multiplying by a number less than 1; for the edges with high NMS factor values, this is close to 1 and for those with the low values, this is close to zero. Therefore we use a sigmoid function to scale a NMS factor value between 0 and 1 and generate this number. Thus the total edge magnitude of each pixel in direction m is calculated as Equation (4.15).

$$TotalEdgeMag_m(P) = EdgeMag_m(P) \times \frac{1}{1 + e^{-2(NMS_m(P)-4)}} \tag{4.15}$$

Most edge detection algorithms utilise thresholding techniques to identify edges after calculation of edge magnitudes. These techniques use one or more threshold values to decide whether or not a pixel is an edge according to its edge magnitude. An edge pixel with an edge magnitude less than the threshold values may be wrongly recognised as a non-edge. For this reason, thresholding techniques often cause broken edges in the edge detection. Therefore, we use another sigmoid function to minimise the side effect of using these techniques. The total edge magnitude of each pixel is scaled by the sigmoid function between 0 and 1 in order to estimate a possibility score of the pixel P lying on an edge, as can be seen in Equation (4.16):

$$PScore_m(P) = \frac{1}{1 + e^{-\frac{3.317}{TH}(TotalEdgeMag_m(P)-0.6229TH)}} \tag{4.16}$$

where $PScore_m(P)$ is the possibility score of the pixel P lying on an edge in the direction of m; and TH is a threshold value between 0 and 1 which can be estimated by Otsu's method for image segmentation [155] as will be described in subsection 4.3.3. We use Equation (4.16) to minimise the

side effect of using the thresholding techniques and improve the detection of the weak edges. This equation is formulated such that TH is the threshold point of the sigmoid function at which the third derivative of the sigmoid function is zero, so its saturation point is $0.246TH$ at which the third derivative of the function is also zero. By this way, the possibility scores of the strongest edges will be higher than the threshold point TH and close to 1, those of the weak edges will be about 0.5, and those of weakest edges will be lower than the value corresponding to the saturation point. We aim that all weak edges are given a chance to be detected as edges if they are located on a continuous edge along with some strong edges. More information about the sigmoid function is available in the appendix.

Possibility Score of a Curve on a Continuous Edge

The proposed model considers a collection of pixels located on a continuous edge instead of considering only a single pixel as most edge detection algorithms operate. Since the pixels along a continuous edge have similar intensities, the pixel intensity of the broken edges are very similar to the intensities of their adjacent edge pixels. Therefore, in addition to the edge magnitude of the pixels on the continuous edge, the intensities of the pixels are also used to evaluate each curve. We introduce the uniformity factor of curve C fitting on a continuous edge in order to calculate the similarity of the pixels on the curve in intensity:

$$U(C) = \frac{1}{255 \times 2L} \sum_{i=1}^{2L} |I_{P_{i+1}} - I_{P_i}| \tag{4.17}$$

where I_{P_i} is the intensity of the ith pixel on curve C. Here $U(C)$ is a real number between 0 and 1 and a low value of this factor for a curve implies a better fit on the actual edge, as pixel intensities are similar along the curve. In the denominator of Equation (4.17), 255 is the maximum distance

between two pixels in an image with a resolution of 8 bits per pixel.

The average of the possibility scores of the pixels on a continuous edge in conjunction with the uniformity factor of the edge are used to estimate the total possibility score of the curve being on a continuous edge. The possibility score of the curve C is formulated as Equation (4.18) such that it maximises the possibility of the pixels on the curve and minimises its uniformity factor.

$$PScore(C) = \frac{\sum_{P_i \in C} PScore_{m_i}(P_i)/(2L+1)}{1 + U_C} \tag{4.18}$$

Curvature Cost of Continuous Edges

All adjacent pixels on a smooth edge usually have almost the same edge orientation, i.e., the difference between the edge orientation of two adjacent pixels is low and their edge directions are similar. Therefore, we propose a curvature cost of a continuous edge in order to reduce the effect of producing jagged edges. The curvature cost (CC) of an edge pixel is introduced here to show a local measure of curvature followed by the curvature cost of a continuous edge. The local curvature measure is defined based on a movement direction from a pixel to its adjacent pixels, as shown in Equation (4.19).

$$CC(m_i, m_{i+1}) = \begin{cases} |m_i - m_{i+1}|/w_3 & |m_i - m_{i+1}| \leq 4 \\ (8 - |m_i - m_{i+1}|)/w_3 & otherwise \end{cases} \tag{4.19}$$

Here m_i is the ith movement direction according to the encoding and w_3 is a weight factor.

The curvature cost of curve C is calculated by Equation (4.20) which is the average of the curvature cost of all single pixels on the curve.

$$CCost(C) = \frac{1}{2L-2}\left(\sum_{i=1}^{L-1} CC(m_i, m_{i+1}) + \sum_{i=L+1}^{2L-1} CC(m_i, m_{i+1})\right) \quad (4.20)$$

Fitness Function with Two Constraints

Since the possibility score of a curve should be maximised to fit more accurately on a continuous edge and its curvature cost should be minimised to be smooth, we propose the following fitness function to evaluate curve C:

$$Fitness(C) = PScore(C) - CCost(C) \quad (4.21)$$

subject to two constraints:

$$Cross(C) = 0 \quad \text{and} \quad PScore(C) > HP$$

where $Cross(C)$ counts how many times the curve C crosses itself and HP is a threshold value that is defined by the user. The curves, represented by the encoding, may sometimes intersect themselves, so we set a constraint $Cross(C) = 0$. On the other hand, $PScore(C) > HP$ as another constraint should be satisfied to reduce false alarms.

4.3.3 Otsu's Method for Estimation of TH

Otsu's method [155] is a very common nonparametric approach to determining a global threshold value for binarisation of the resulting image after applying an edge operator. It works in an optimum way to divide a set of pixels into two subsets (edge and non-edge) where it maximises the discriminating criteria of interset variance between the pixel intensities in

these subsets [155]. The edge magnitudes of the pixels belonging to the first subset are less than or equal to t and those of the pixels in the second subset are greater than t. Let $\mu_1(t)$ and $\mu_2(t)$ be the average edge magnitude of the pixels in the first and second subsets, and $N_1(t)$ and $N_2(t)$ be the number of the pixels in these subsets respectively. The average edge magnitude of all pixels ($\mu_A(t)$) can be calculated as follows.

$$\mu_A(t) = \frac{N_1(t)\mu_1(t) + N_2(t)\mu_2(t)}{N_1(t) + N_2(t)}$$

The interset variance between these two subsets $\delta_A(t)$ is

$$\delta_A(t) = N_1(t)[\mu_1(t) - \mu_A(t)]^2 + N_2(t)[\mu_2(t) - \mu_A(t)]^2$$

To estimate TH in Equation (4.16), the local maximum of the edge magnitude of each pixel $LocalEdgeMag(P)$ is first calculated using Equation (4.22).

$$LocalEdgeMag(P) = \max_{i=1}^{8}(TotalEdgeMag_i(P)) \tag{4.22}$$

Applying this equation results in an edge magnitude map which can be used as the input to the Otsu's method in order to estimate the value of parameter TH in Equation (4.16). Figure 4.4 shows how Otsu's method can be used in the proposed method for determination of the value of this parameter. The value t corresponding to the maximum of $\delta_A(t)$ is considered as TH. As shown in Figure 4.4, the edge magnitudes of each pixel in 8 different direction are first calculated and then the local maximum edge magnitude of each pixel is computed. The resulted local edge magnitude maps is considered as the input of Otsu's method to estimate TH.

4.4 Two Proposed PSO-based Algorithms

Two PSO-based algorithms are developed for the optimisation of the proposed model to detect edges in noisy images. As can be seen in the overall

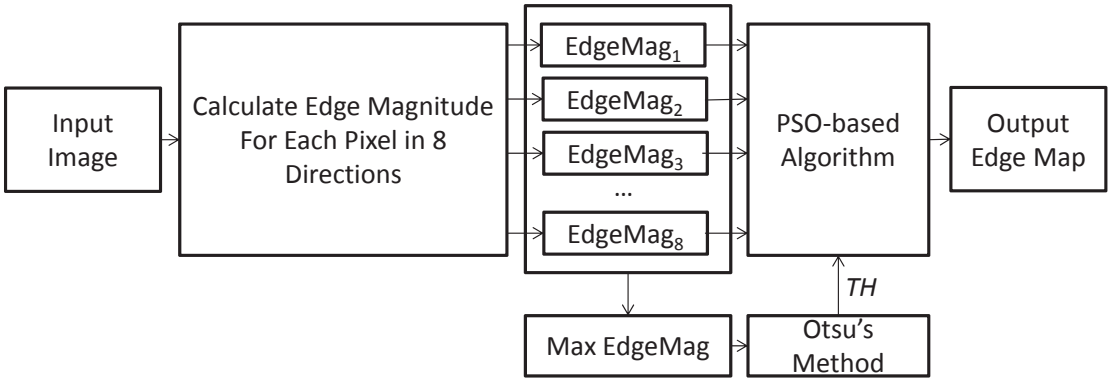


Figure 4.4: Paradigm of PSO-based Edge Detector.

flowchart of the PSO-based algorithm in Figure 4.5, the red square in Figure 4.2 (a) first moves on the area in which exists at least one pixel with a high edge magnitude. Then one of the PSO-based algorithms is applied to the chosen area and find the best curve which can be fitted on a continuous edge passing inside the red square. After applying the PSO-based algorithms, the red square moves to the next block.

Each particle in the PSO-based algorithms represents a curve in an area of an image using the developed encoding scheme. In the proposed algorithms, we move the red square (as shown in Figure 4.2(a)) over the image from top left to bottom right. After each movement, we apply a PSO-based algorithm to find the best curve which can be fitted on a real continuous edge. In each run of PSO, all possible curves, whose centres (pixel C) are located inside of the $SqrSize \times SqrSize$ red square, are processed. If the best curve is found by the PSO algorithm, the pixels on the curve are marked as edges and the pixels within the rectangle are not marked as processed pixels; otherwise all pixels within the square are marked as processed pixels. Those pixels which are not marked as processed should be considered in the next iteration of the main loop because the algorithm may find another curve in this area. If $SqrSize$ is set to a large value, the speed of the algorithm will be increased because of processing a large

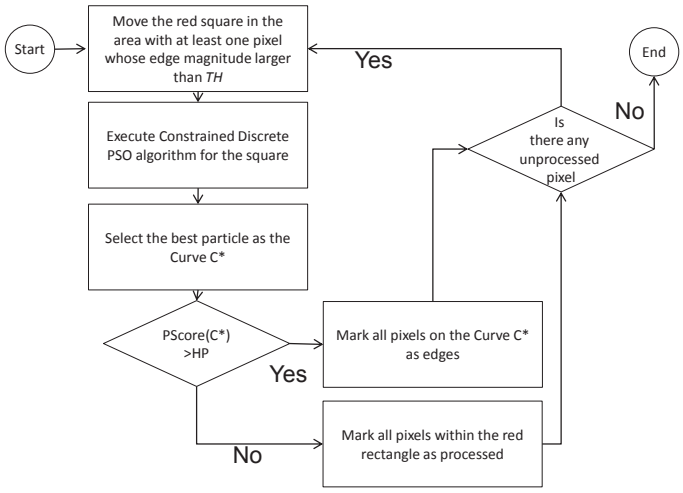


Figure 4.5: The flowchart of the PSO-based algorithm

number of pixels by the PSO algorithm at a time. However, it may cause that some details in the edge map are removed as will be shown later. Also, if $2L + 1$ is set to a small value, the execution time of the algorithm will be decreased due to the reduction of the length of the particle encoding. However, it may cause that the number of the broken edges will be increased. Therefore, these parameters should be adjusted by the user carefully. We will further discuss about these parameters in Section 4.6.4.

As described Section 4.3.2, the PSO-based edge detector should optimise a function with two constraints. The selection of constraint handling methods is very problem dependent. Several methods have been proposed to handle constraints in PSO. These methods can be categorised into four main groups. In the first group, all particles are initialised such that the potential solutions fall within a feasible search space. These methods typically utilise a particular operator to preserve new solutions to not violate existing constraints [156]. In the second group, the algorithms add a penalty to the fitness of the particles which violate constraints [157]. The third group (partitioning methods) divide all particles into a feasible set

and an infeasible set that are operated on differently. Some of them manipulate and mend infeasible solutions or prioritise solutions based on their feasibility [158][159]. In the last category, the optimisation problem is transformed to another one such that either the constraints can be handled in an easier way, or they can be eliminated. An example is using homomorphous mappings on a problem with linear equality constraints [160].

Two different but more commonly used methods are applied here. The first is based on a preservation method and the second is based on a penalising method. This section describes these two algorithms after explaining the truncation method to convert the real values to integers in the PSO-based algorithms.

4.4.1 Truncation Method for Discrete PSO

As the search space explored by the new PSO-based algorithms is discrete, the particle positions must be truncated to integers after they are updated by Equation (2.42). Many discrete versions of PSO use a simple truncation method to convert real numbers to integers [119]. Instead of using a simple truncation method, the following method is used to truncate the values of particle positions to integers:

$$o_i = \begin{cases} (\lfloor o_i \rfloor + 1) & \text{if } o_i - \lfloor o_i \rfloor > R \\ \lfloor o_i \rfloor & \text{otherwise} \end{cases} \tag{4.23}$$

$$m_i = \begin{cases} (m_i + 1) \; mod \; 8 & \text{if } m_i - \lfloor m_i \rfloor > R \\ m_i \; mod \; 8 & \text{otherwise} \end{cases}$$

where R is a uniform random number ranging from 0 to 1. We expect that this simple truncation method increases the diversity of the particles in the population in the described discrete search space to avoid being trapped in a local optima. Note that this rule is only applied to convert the real values of the particle positions to integers but not used to update the

particle velocities. In this equation, the decimal parts of the numbers m_i and o_i show the probability of truncating to the largest integer numbers which are smaller than them and the complementary probability shows the probability of truncating to the smallest integer numbers which is at least as large.

4.4.2 Preservation of Feasible Continuous Edges

Algorithm 4.1 summarises the first PSO-based algorithm (PSO1) which aims to detect edges in noisy images using the optimisation method described in the previous section. This algorithm utilises a preservation method to handle the constraints.

We expect that this algorithm based on preservation can effectively maximise the distances between pixel intensities in the two regions (interset distance) separated by a continuous edge and minimise the distance between the pixel intensities within each region (intraset distance), and accordingly accurately detect continuous, thin and smooth edges in complex images. The PSO algorithm could be initialised only once for all runs. This increases the speed of the algorithm; however, it may prematurely converge to local optima and reduce the accuracy of the algorithm. In the proposed algorithm, the PSO-algorithm is initialised for each iteration of the main loop. Since preservation methods suffer from low diversity of particles, the constraints are examined in line 16 after updating the position of each particle and applying the update rule in line 15.

4.4.3 Penalising Infeasible Continuous Edges

The second PSO-based algorithm (PSO2) uses a penalising method to handle constraints. Although penalising methods require tuning for any constrained optimisation problem, their rapid convergence characteristic makes them attractive [161]. We define a non-stationary and multi-stage penalty fitness function adopted from [162] for edge detection to handle the two

Algorithm 4.1 PSO-based edge detection algorithm based on a preservation method to handle the constraints (PSO1)

for all pixel P on an image with local edge magnitude larger than TH **do**
  if P is unprocessed and not marked as an edge **then**
    Initialize PSO population in feasible search space randomly for pixel P
    repeat
      for all Particle decoded as curve C in Population **do**
        Evaluate $U(C)$ (4.17), $PScore(C)$ (4.18) and $CCost(C)$ (4.20)
        Evaluate $Fitness(C)$ (4.21)
        if $Fitness(C)$ is better than best fitness value in history and C is feasible **then**
          Update personal best position
        end if
      end for
      Assign the best particle in the population to the $leader$
      for all particle decoded as curve C in population **do**
        Calculate particle velocity (2.43)
        Update particle position (2.42) and apply update rule (4.23)
        if $Cross(C) \neq 0$ or $PScore(C) \leq HP$ **then**
          Replace the particle with a new random feasible one
        end if
      end for
    until maximum iterations exceeded or minimum error criteria attained
    Select best feasible particle in the population and decode it as curve C^*
    Mark all pixels on curve C^* as an edge
    if no feasible particle found **then**
      Mark all pixels within the red rectangle as processed
    end if
  end if
end for

constraints as shown in Equation (4.24). Since any optimisation problem can be optimised by an easier way when it does not have any constraints, we expect that the penalised PSO algorithm operates more efficiently than the previous algorithm.

$$PenFit(C) = Fitness(C) - \sqrt{K}(Cross(C) + \theta(q(C))q(C)) \tag{4.24}$$

Here K is the current iteration number of the PSO algorithm, $q(C) = max(0, HP - PScore(C))$, and $\theta(q(C))$ is calculated as Equation (4.25):

$$\theta(q(C)) = \begin{cases} 1 & \text{if } q(c) < 0.001 \\ 2 & \text{if } q(c) < 0.1 \\ 10 & \text{otherwise} \end{cases} \tag{4.25}$$

The second constrained discrete PSO-based algorithm utilising a penalising method is outlined in Algorithm 4.2. In each iteration of this algorithm (lines 5–19), the uniformity factor, edge possibility and curvature cost of each curve presented by each particle are calculated. The fitness value of each particle is computed and then the best and the worst particles are found. The worst particle is replaced with a new random one in line 15 in order to increase the diversity of the particles. After updating the velocities and the positions of the particles, the stopping criteria are checked, i.e., whether the maximum number of iterations is exceeded or minimum error criterion is attained. Since the best continuous curve C^* may violate the constraints, its penalised fitness value is checked not to be less than HP. If the value is less than HP, all pixels on the curve C^* are marked as edges; otherwise all pixels inside of the red square are marked as processed pixels (lines 21–25).

Algorithm 4.2 Constrained PSO-based edge detection algorithm based on a penalising method to handle the constraints (PSO2)

for all pixel P on an image with a local edge magnitude larger than TH **do**
  if P is unprocessed and not marked as an edge **then**
    Initialize PSO population randomly for pixel P
    $K = 0$
    repeat
      Increment K
      for all particle (decoded as curve C) **do**
        Evaluate $U(C)$, $PScore(C)$ and $CCost(C)$
        Evaluate $q(C)$ and $\theta(q(C))$
        Evaluate $Fitness(C)$ and $PenFit(C)$
        if $PenFit(C)$ is better than best fitness value **then**
          Assign C to best particle
        end if
      end for
      Replace the worst particle with a new random one
      for all Particle decoded as curve C **do**
        Calculate particle velocity
        Update particle position and apply update rule (4.23)
      end for
    until maximum iterations exceeded or minimum error criteria attained
    Select best particle and decode it as curve C^*
    if C^* is feasible **then**
      Mark all pixels on curve C^* as an edge
    else
      Mark all pixels within red rectangle as processed
    end if
  end if
end for

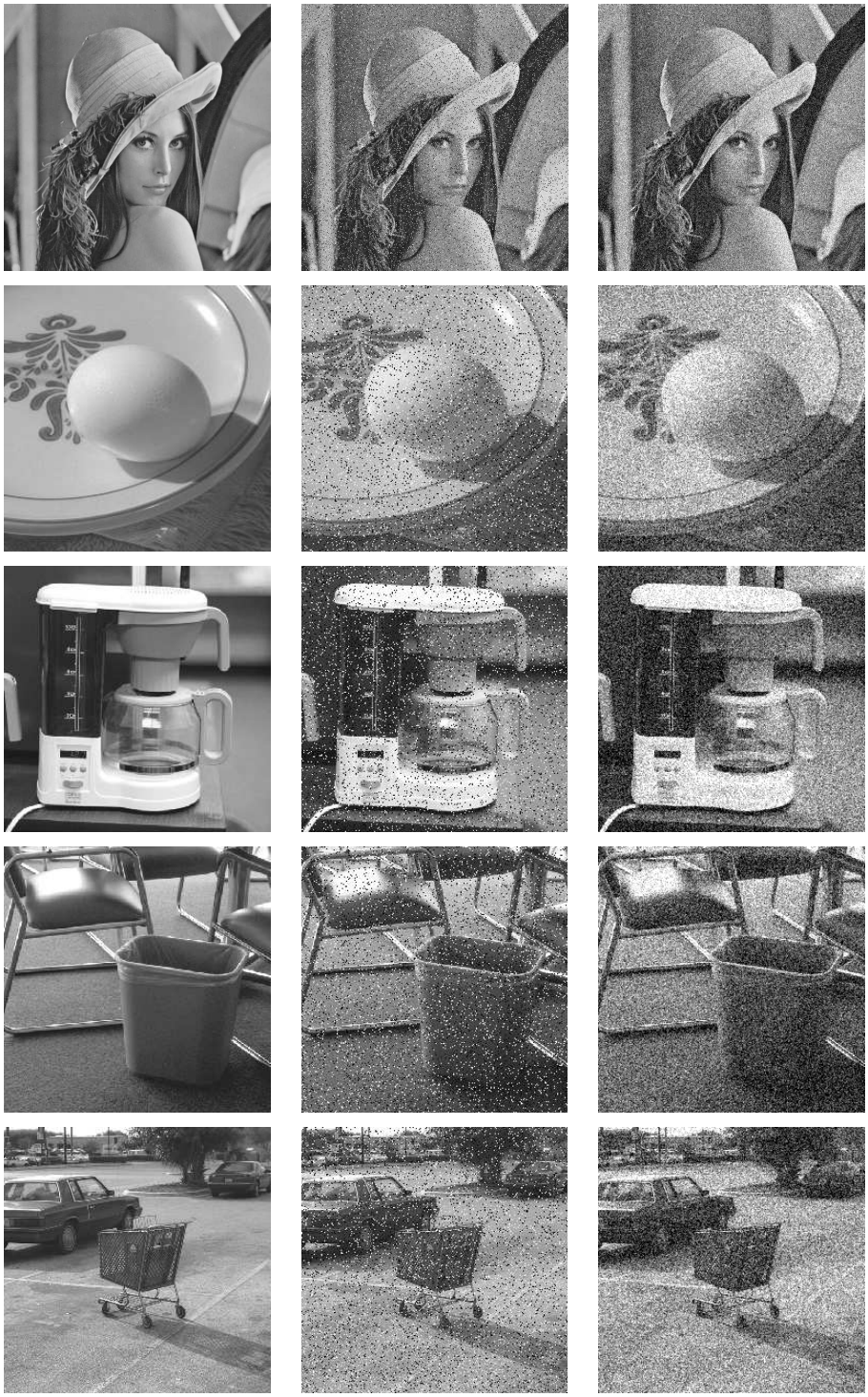
4.5 Experimental Design

To investigate the effectiveness of the new algorithms, we first compare the first algorithm (PSO1) with a modified version of Canny [61] and RRO [11] algorithms on two sets of benchmark images at different types and levels of noise. Then we compare the efficiency and effectiveness of PSO1 which is based on a preservation method with those of the second new algorithm (PSO2) which is based on a penalising method. We equip the Canny algorithm with an unsupervised hysteresis thresholding technique proposed in [163], with an adaptive method to estimate its filter scale proposed in [64] and with a NMS technique proposed in [10] to improve its performance. Note that these techniques are not applicable for the RRO detector. This section also describes the image sets, performance measure, and parameter settings, which are used in the experiments.

4.5.1 Image Sets

Two different image sets are used in our experiments. The first image set includes five natural images which are commonly used as benchmarks for edge detection: Lena, egg, coffee maker, rubbish bin and car (see Figure 4.6). As described in Chapter 3, to explore the performance of the new algorithms in noisy environments, these images are corrupted by two different types of noise: impulse and Gaussian (see the images in Figure 4.6 in columns (b) and (c)). The probability of the impulse noise is 0.1 and the peak-signal to noise ratio (PSNR) is $16dB$ for the Gaussian noise in these noisy images. The reason for choosing these values is that a PSNR below 16dB and a probability above 0.1 are effective noise based on litrature[164]. As the ground truth of these images are not available, we will use them for a subjective (qualitative) comparison.

The second image set includes one synthetic circle image and four real images (Saturn, multi-cube, wall and road). The real images have been provided by the University of Cordoba (Spain) and their ground truth



(a) (b) (c)

Figure 4.6: Example images for subjective comparison. (a) Original images Lena, egg, coffee maker, rubbish bin and car; (b) images with *impulssive noise* (noise probability=0.1); (c) images with *Gaussian noise* (PSNR=16dB).

edge maps are available from [137]. The size of each image is 256×256 pixels and the resolution of each is 8 bits per pixel. These images are shown in Figure 4.7. To investigate the performance of the new algorithms in noisy environments, we also add two different types of noise in different noise levels. For the impulse noise, the noise probability ranges from 0.1 to 0.5 with a step size of 0.05. For the Gaussian noise, the PSNR value ranges from 0 to $22dB$ with a step size of $2dB$. As the ground truth of these images are available, we used them for an objective (quantitative) comparison.

4.5.2 Quantitative Performance Measure

To evaluate the performance of the new algorithm, we use Pratt's Figure of Merit (PFOM) which is commonly used as a quantitative measure for the objective comparison of the *localisation accuracy* of edge detection algorithms [14]. This measure is defined by Equation (4.26).

$$R_{PFOM} = \frac{1}{max(I_I, I_A)} \sum_{i=1}^{I_A} \frac{1}{1 + \beta d(i)^2} \tag{4.26}$$

Here I_I and I_A indicate the number of ideal and actual edge points in the ground truth and the generated edge map images, $d(i)$ is the distance between the pixel i in the generated edge map and the nearest ideal edge point in the ideal edge map, and β is a constant scale factor which is typically set to $\frac{1}{9}$. This measure is an index to compute the localisation accuracy of edge detection algorithms. The ideal value of R_{PFOM} is 1.0 and the minimum could be very small. A larger value indicates stronger performance.

4.5.3 Parameter Settings

The selection of a population size and the maximum number of iterations is problem dependent. In PSO, the population size usually ranges from 20 to 50. In comparison to other evolutionary algorithms, PSO needs a

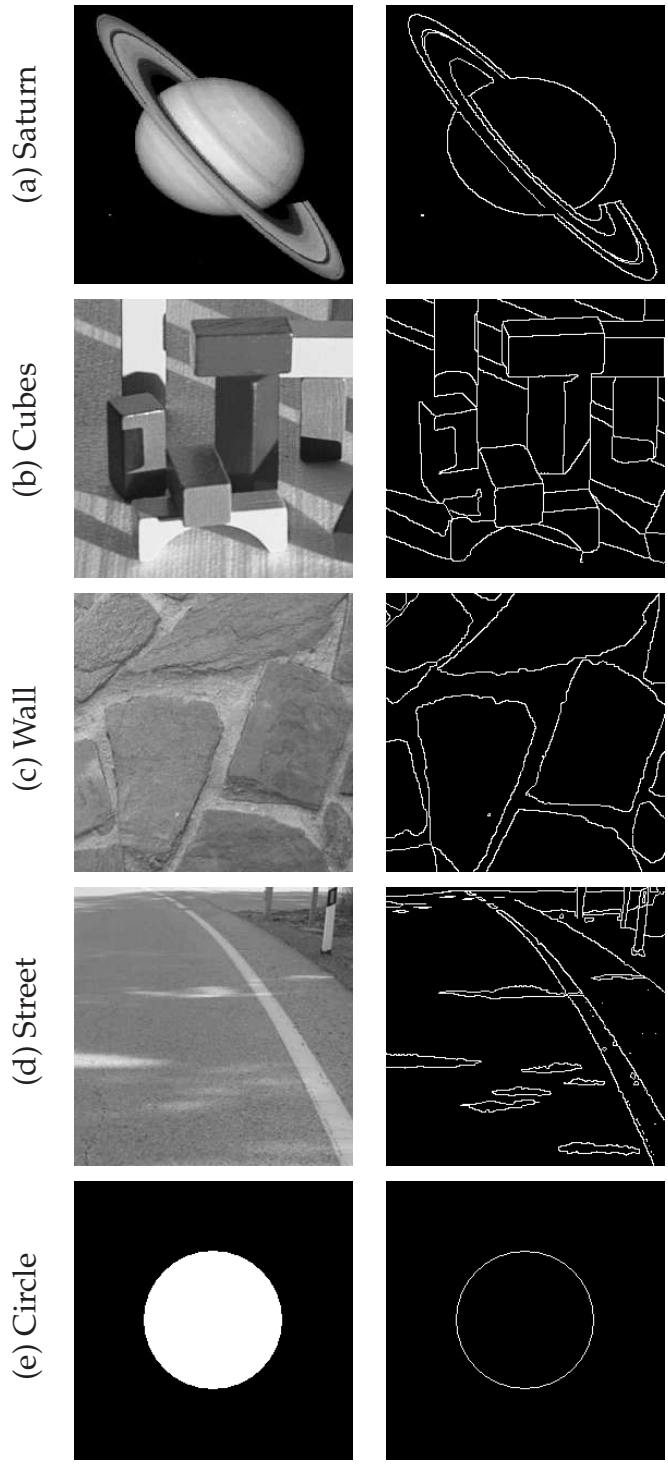


Figure 4.7: Example images for objective comparision. (a)–(d) four real image from the UCO university and their manual ground truth images [137]; (e) one synthetic circle image and its ground truth.

smaller population to find a high quality solution [165]. In the proposed PSO algorithms, the population size is 50 and the maximum number of iterations is 200 according to the chosen particle length [118]. The minimum length of a continuous edge, $2L + 1$ was set at 21, $SqrSize$ at 6, n at 9, $w_1 = 90$, $w_2 = 40$, $w_3 = 40$, TH at the value estimated by the Ostu's method [155], and HP at 0.5. We chose the value of the weight factors (w_1, w_2 and w_3) based on emprical search. We changed the values of w_1 and w_2 and then calculated the local edge magnitude of each pixel on several images. We subjectively compared the resulting edge magnitude images with those provided by the Sobel edge detector [2] used in the Canny algorithm to estimate edge magnitudes. The value of these two factors (w_1, w_2) were adjusted such that the resulting edge magnitude images are sufficiently similar. To estimate weight factor w_3, we applied PSO1 on several noisy images including simple shapes such as (circles, ellipses and rectangles). We chose the value of this factor such that PSO1 could detect smooth edges in this images. These three weight factors can be ideally determined through a brute-force search on a large number of images with ground truth. We used the values $w = 0.7298$, $c_1 = 1.4962$, $c_2 = 1.4962$ for the parameters in Equation (2.43). These values were chosen based on common settings [166].

In order to make consistent and fair comparison of the proposed algorithms with Canny and RRO, we used the following adaptive parameters for the Canny edge detector: $high\ threshold = 2\sigma\sqrt{f_u ln2}$, $low\ threshold = \frac{1}{2} high\ threshold$ with $f_u = s - l$, $s = \sum_{i=1}^{N}\sum_{j=1}^{N} a_{ij}^2$ and $l = \sum_{i=1}^{N}\sum_{j=3}^{N} a_{ij}a_{ij-2}$ where the coefficients a_{ij} correspond to the kernel of the filter with which the image has been smoothed [163], the filter size (σ) was set at the value estimated from the approach proposed by Jeong and Kim [64]. The edge-height parameter of the RRO detector, which defines the minimum gray-level differential across an edge, was set to the value which gave a highest PFOM value.

4.6 Results and Discussion

This section presents the results of the subjective comparison of four algorithms (Canny, RRO, PSO1 and PSO2) followed by the results of the objective comparison. This section also provides a short discussion on the parameters of the PSO-based edge detection algorithms.

4.6.1 Subjective/Qualitative Comparison

For a qualitative comparison of PSO1 and PSO2 with Canny and RRO, we first applied PSO1 and PSO2 to the images in the first set. The resulting images are shown in Figures 4.8 and 4.9 after applying the Canny [61], the RRO [11] and PSO1 on the images in the first set (Figure 4.6) corrupted by impulse and Gaussian noise respectively. Since the edge maps resulted from PSO1 and PSO2 were very similar to each other, the edge maps resulted only from PSO are shown in these figures.

The resulting images in Figure 4.8 show that PSO1 performed better than the other two algorithms on the five images with *impulse noise* at a noise probability level of 0.1. The Canny algorithm, even with post-processing, did not work well for these noisy images and there are many noise spots in the resulting images. This suggests that the Canny algorithm is not suitable for detecting edges for the images corrupted by impulse noise and is sensitive to this kind of noise. The RRO detector operated better than Canny, however the detected edges are thicker than those detected by Canny. PSO1 detected edges much thinner than the RRO detector and found edges with greater continuity. As can be seen from Figure 4.8, for the Lena image, there are some broken edges on Lena's hat in the resulting image by RRO, while PSO1 improved the detection of the edges in this area and reduced the broken edges. The edges detected by PSO1 on the bar in the upper left corner of the image have been significantly improved in comparison with other two algorithms. For the egg image, RRO detected some false edges on the surface of the egg and also there are

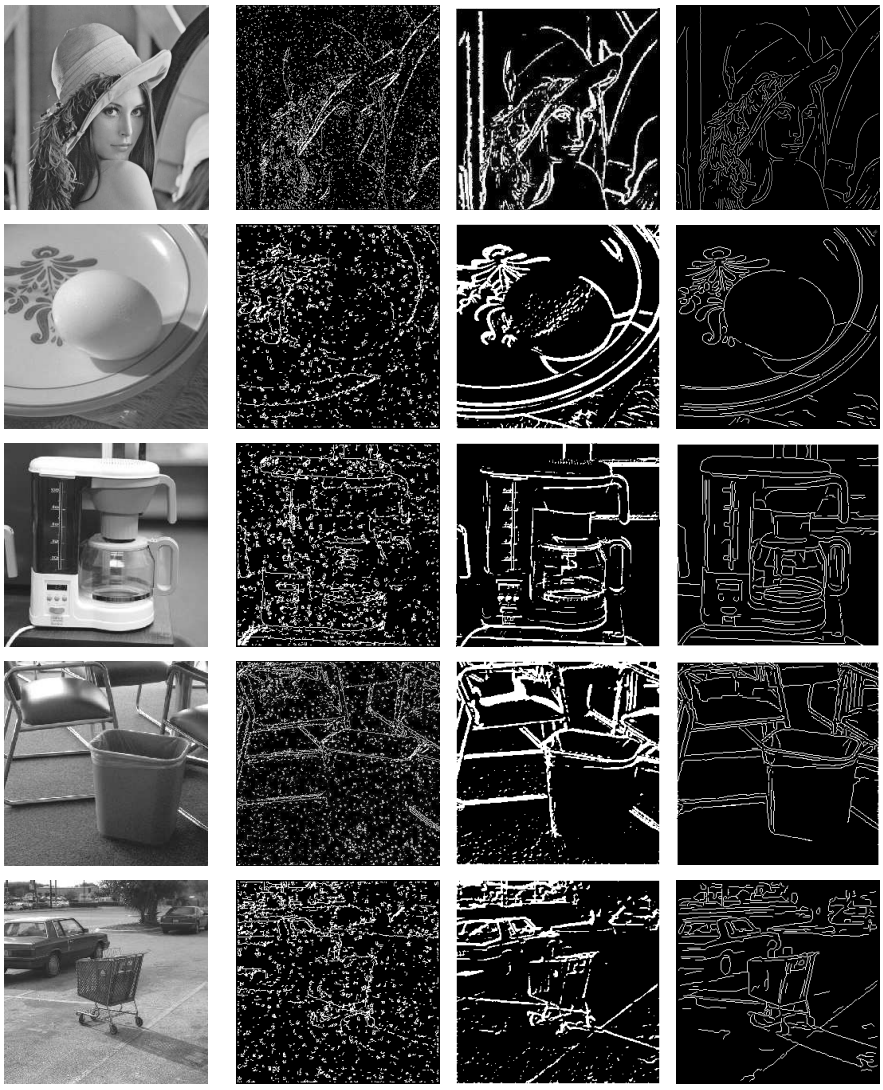


Figure 4.8: Subjective results of edge detection produced by three algorithms on the five images corrupted by *impulse noise* (noise probability=0.1).

several broken edges on the egg's boundary. PSO1 operated much better than RRO on the boundary of the egg. PSO1 reduced the broken edges on the egg's boundary especially at the bottom of the egg. RRO detected edges well in the coffee maker image, however there are still some problems in the detection of the edges in the middle-right of the image. PSO1 detected more continuous and smoother edges in this region. RRO did not operate well on the rubbish bin especially on the left and bottom sides of the bin and there are many broken edges in the image produced by RRO, while PSO1 improved the detection of the edges in this area. PSO1 also improved the detection of the edges for the car image on the surface of the street, the back wheel of the car and the trolley, while RRO did not work well in these areas.

The resulting images are shown in Figure 4.9 after applying the three algorithms on the five noisy images corrupted by Gaussian noise (PSNR= 16dB). The comparison of these results with those for the impulse noise shows that Canny detected edges much better on the egg and coffee maker images and the noise was almost removed, but there were still many noise spots and broken edges on the Lena, car, and rubbish bin images. This implies that Canny is less sensitive to Gaussian noise than to impulse noise. The edges detected by RRO and PSO1 for these images have very similar quality to those with the impulse noise, although PSO1 recognised edges with greater continuity and smoother than Canny, and also detected edges much thinner than RRO.

4.6.2 Objective/Quantitative Comparison

To objectively compare the first new algorithm (PSO1) with the other two algorithms (RRO and Canny), the localisation accuracy (PFOM) was calculated from the resulting images after applying the three algorithms to the second set of images (Figure 4.7) at different noise levels. The PFOM values are plotted at 11 different Gaussian noise levels and 9 impulse noise

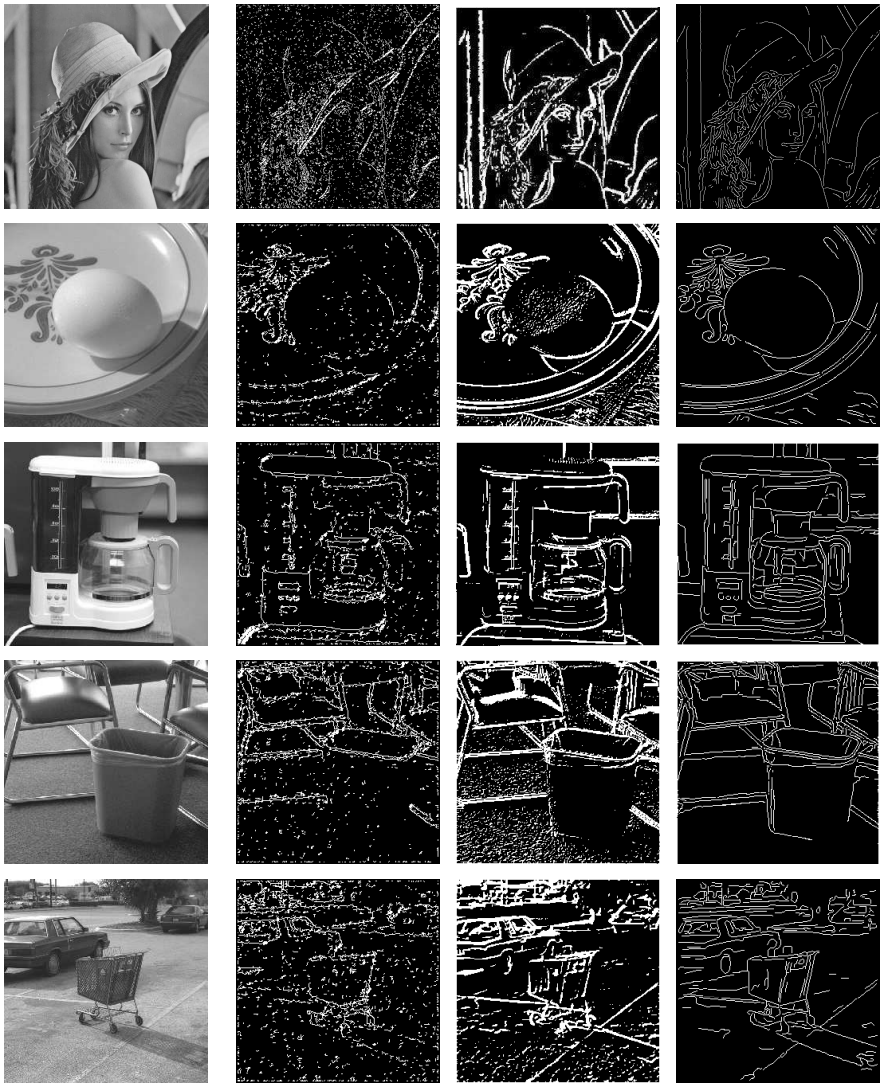


Figure 4.9: Subjective results of edge detection produced by the three algorithms on the five images with *Gaussian noise* (PSNR=16dB).

levels as depicted in Figure 4.10. PSNR ranges from 0 to 22dB with step of $2dB$ and the noise probability ranges from 0.1 to 0.5 with step of 0.05. The number of pixels in each set in Equations (4.12) and (4.13), parameter n, was set at 9 in all experiments in this subsection. The average of the localisation accuracy of PSO was plotted after 30 runs for each image in each noise level.

As can be seen from the resulting plots in Figure 4.10, PSO1 generally outperformed the other two algorithms especially when a high level of noise is present in the images. The Canny algorithm operated reasonably well on the images with a low-level of Gaussian noise, but it did not work well in the images even with a low-level of impulse noise in most cases (see Figures 4.10(a), (b), (c) and (d)). These resulting plots also illustrate that PSO1 outperformed RRO in the images with impulse and Gaussian noise in most cases, however its performance is lower than RRO in a few cases in the images corrupted by Gaussian noise (see Figure 4.10(e)). This suggests that PSO1 is less sensitive to Gaussian noise and impulse noise than RRO. Canny is more sensitive to impulse noise and also more sensitive to high-levels of Gaussian noise than RRO. As expected, the accuracy of most algorithms is decreased when noise level is increased. However, PSO1 can overcome high-levels of noise and it is less sensitive to noise than other methods in most cases.

4.6.3 Discussion on Parameter Values

As already described, PSO1 has several additional parameters in comparison with Canny and RRO. The parameters are the weight of different factors (w_1, w_2 and w_3), n, $SqrSize$ and L. Variation of the values of the last three parameters may have different influences on efficiency and effectiveness of PSO1 and PSO2. This subsection considers the influences of these parameters on the performance of PSO1.

Figure 4.11 shows the resulting images after applying PSO1 with differ-

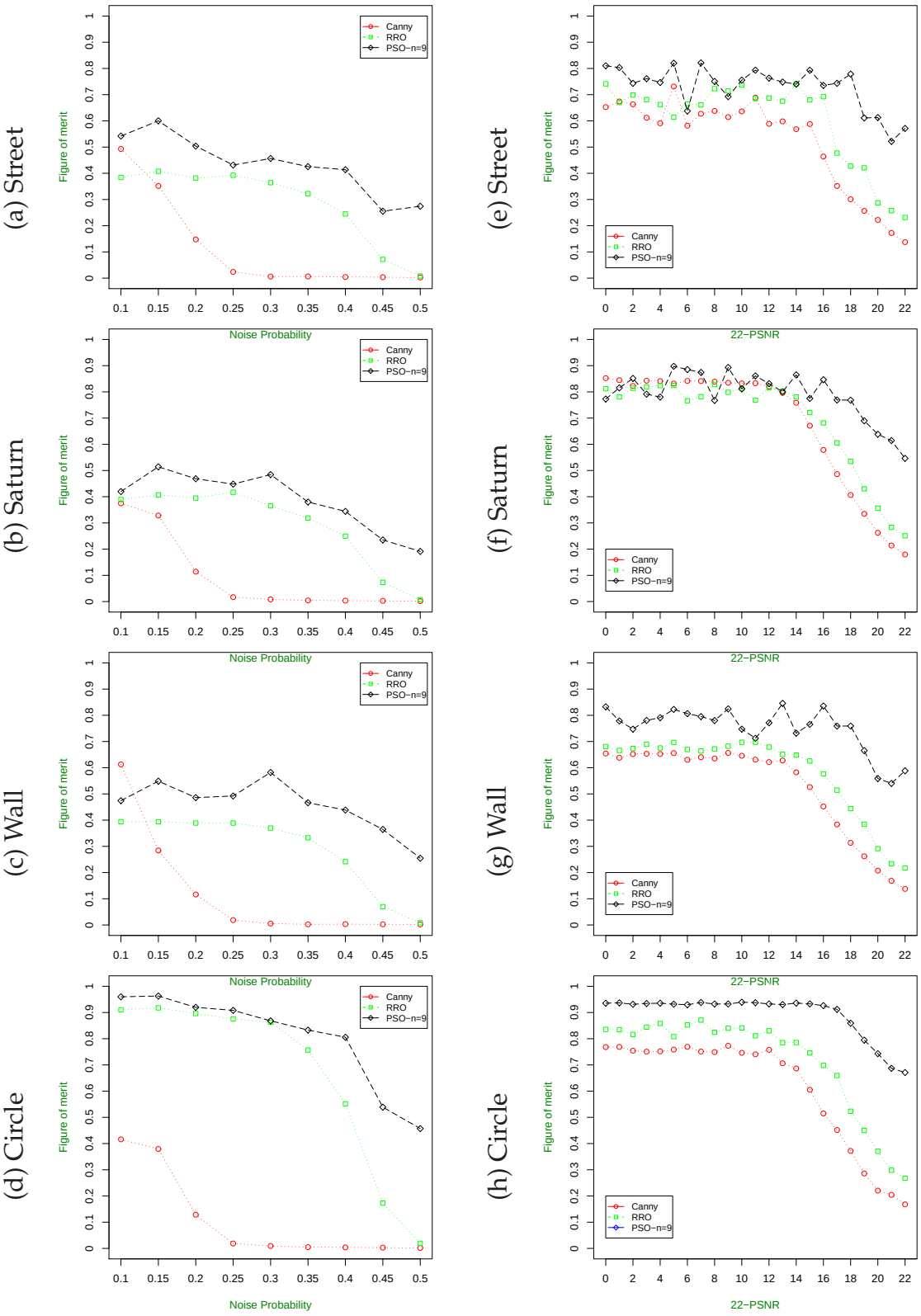
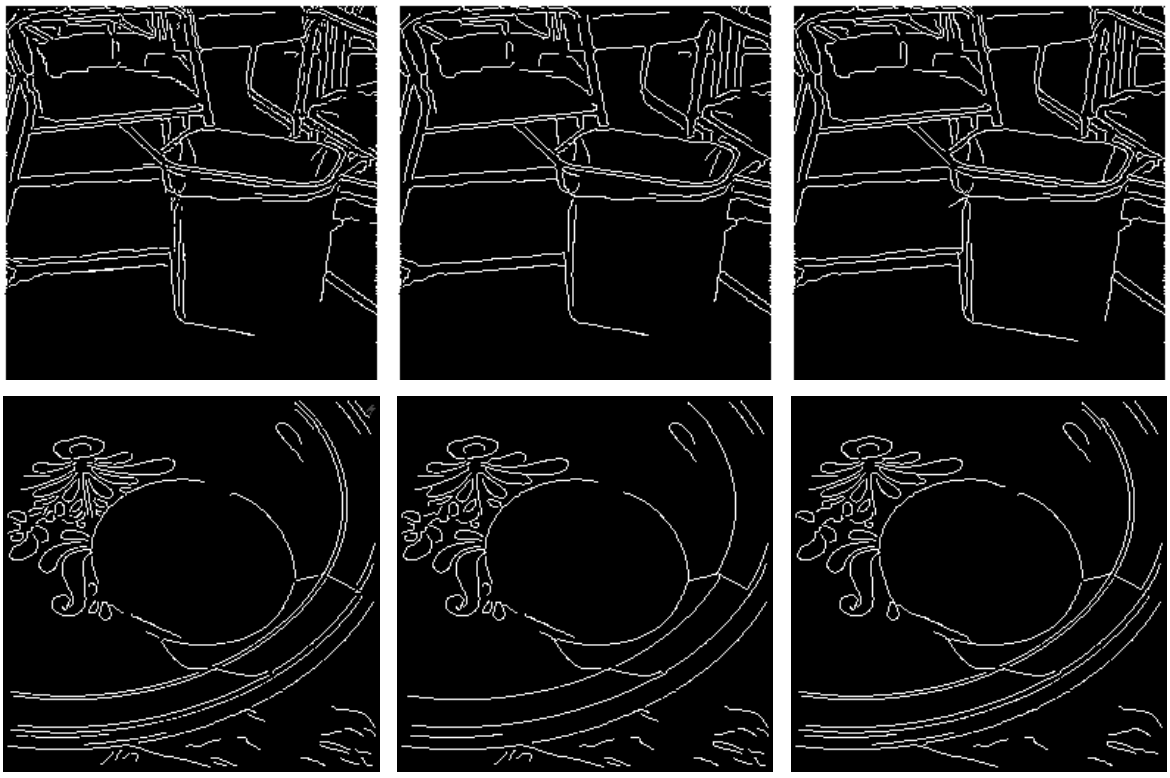


Figure 4.10: PFOM for the street, Saturn, wall and circle images in the second image set. (a)–(d): with different impulse noise levels (the noise probability ranging from 0.1 to 0.5); and (e)–(h) with different Gaussian noise levels (PSNR ranging from 0 to 22dB).

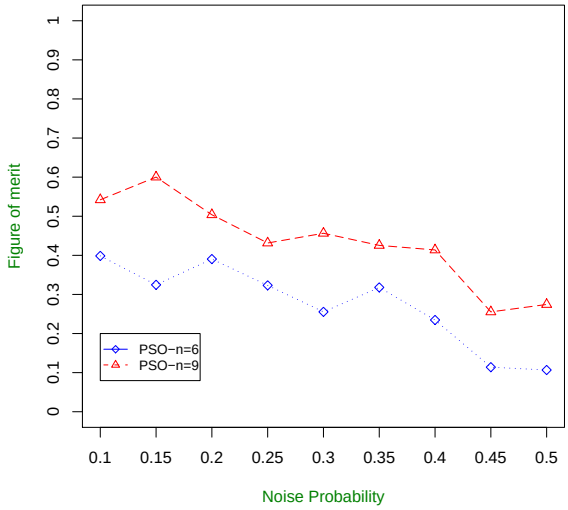


(a) (b) (c)

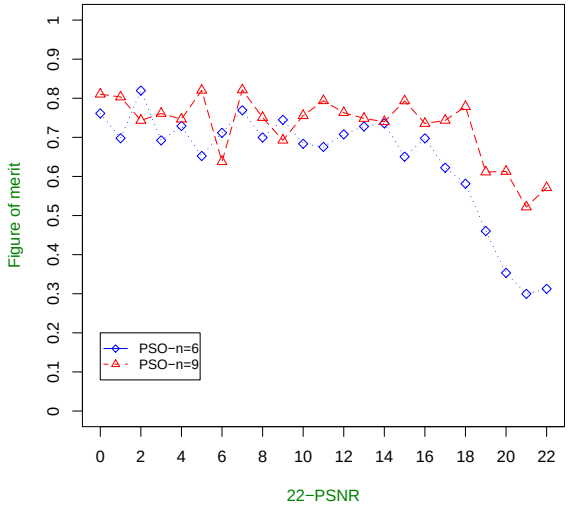
Figure 4.11: Resulting images after applying PSO1 with different parameter values (a) $L = 10$ and $SqrSize = 6$ (b) $L = 10$ and $SqrSize = 10$ (c) $L = 15$ and $SqrSize = 6$

ent parameters on the rubbish-bin and egg images corrupted by impulse noise whose probability is 0.1. As can be seen in the original images in Figure 4.6, there are some illumination areas in the egg and rubbish-bin images. The illumination phenomena makes it difficult for an edge detection algorithm to work well in these areas. The images shown in Figure 4.11(a) are the output of PSO1 with $L = 10$ and $SqrSize = 6$. The resulting images in Figure 4.11(b) depict that when $SqrSize$ is increased, some details of the objects in the images are lost. This happens especially when a weak continuous edge is close to a strong edge and is almost parallel with it such as the edges of the chairs and rubbish-bin in the first image and the edges around the plate in the egg image. The images in Figure 4.11(c) are the resulting images when $L = 15$ and $SqrSize = 6$. In these images, the edges around the egg and rubbish-bin have been improved. In this case, when the length of the curve represented by a particle is increased, the length of broken edges is reduced but the algorithm is slower. Therefore, the selection of a suitable particle length and the size of the square can affect the accuracy and speed of the algorithm.

To illustrate the influence of parameter n (the number of the neighbour pixels in Equations (4.12) and (4.13)) on the algorithm's accuracy, the localization accuracy (PFOM) at different noise levels was calculated from the resulting images after applying PSO1. The value of parameter n was set at 6 and 9 and then the algorithm was applied to the street image in the second image set which was corrupted by different types and levels of noise. The average of the localisation accuracy of PSO1 was plotted after 30 runs in each noise level as can be seen in Figure 4.12. The plots shows that when $n = 9$, the algorithm's accuracy is higher than when $n = 6$ in most cases.



(a)



(b)

Figure 4.12: Comparing PFOM for the street image in the second image set when $n = 6$ or $n = 9$ (a) with different impulse noise levels (the noise probability ranging from 0.1 to 0.5the impulse noise) (b) with different Gaussian noise levels (PSNR ranging from 0 to 22dB). Note horizontal axis is $22-\text{PSNR}$

4.6.4 Comparison of the Two Proposed Algorithms

For the objective comparision of the two proposed algorithms (PSO1 and PSO2), we use PFOM as a measure of localisation accuracy and the number of fitness function evaluations as a measure of time.

Table 4.1 shows PFOM estimated from the resulting images after applying Canny, RRO, PSO1 and PSO2. G6, G10, G14, G18 and G22 represent PSNR from 6dB to 22dB for Gaussian noise and N0.1, N0.2, N0.3, N0.4 and N0.5 represent noise probability from 0.1 to 0.5 for impulse noise. The columns "PSO1" and "PSO2" show the 95% confidence intervals for the localisation accuracy of the two new algorithms after 30 runs for each image in each noise level. To compare the accuracy means of PSO1 and PSO2, t-tests were used (the alternative hypothesis was inequality of the means). The statistical analysis showed that null hypothesis is accepted in almost all cases, i.e, there is no significant difference between their localisation accuracy, except for a few cases as shown in bold in Table 4.1 columns PSO1 and PSO2. However, the accuracy variance of the second PSO algorithm is lower than that of the first one; this implies that the second algorithm is more stable than the first one. Table 4.1 also shows that both PSO1 and PSO2 are significantly better than Canny and RRO in most cases except for a few cases with bold fonts.

Table 4.2 depicts the 95% confidence intervals for the number of fitness function evaluations of the two new algorithms after 30 runs for each image in each noise level. The alternative hypothesis was that the number of fitness function evaluations of PSO2 is greater than that of PSO1. Statistical analysis showed that null hypothesis is rejected in all cases. Therefore, the number of fitness function evaluations of PSO1 is 16% to 41% greater than PSO2. It implies that PSO1 is slower than PSO2. The results in this table also shows that the number of fitness function evaluation for the both algorithms is reduced when the level of the noise is increased.

The execution time of PSO1 is usually between 50 and 70 seconds and that of PSO2 is between 40 and 50 seconds for these images depending

Table 4.1: Comparison of accuracy of two proposed algorithms 4.1 and 4.2 with Canny and RRO

Image	Noise Level	95% Confidence Interval for Accuracy PSO1	95% Confidence Interval for Accuracy PSO2	Standard deviation PSO1	Standard deviation PSO2	Canny	RRO
Circle	G22	0.935478 ± 0.008350	0.931228 ± 0.001180	0.023335	0.003298	0.768116	0.835660
Circle	G18	0.931387 ± 0.008350	0.929149 ± 0.002917	0.023335	0.008151	0.754445	0.816217
Cicle	G14	0.935589 ± 0.009186	0.935650 ± 0.003156	0.025669	0.008821	0.751867	0.858205
Circle	G10	0.929730 ± 0.009603	0.928105 ± 0.003227	0.026836	0.009017	0.769119	0.852699
Circle	G6	0.932547 ± 0.010021	0.931485 ± 0.002688	0.028003	0.007511	0.749058	0.824651
Saturn	G22	0.772790 ± 0.007515	0.772799 ± 0.003151	0.021002	0.008807	**0.852345**	0.812005
Saturn	G18	0.851451 ± 0.008768	0.853361 ± 0.002623	0.024502	0.007330	0.822624	0.813742
Saturn	G14	0.780282 ± 0.010021	0.784602 ± 0.002808	0.028003	0.007847	**0.840852**	0.824204
Saturn	G10	0.885272 ± 0.010438	0.883163 ± 0.003215	0.029169	0.008985	0.841772	0.766146
Saturn	G6	0.767548 ± 0.011273	0.767398 ± 0.002800	0.031503	0.007826	**0.838813**	0.828318
Cube	G22	0.617938 ± 0.005845	0.618242 ± 0.003151	0.016335	0.008807	0.187473	0.401117
Cube	G18	0.644613 ± 0.006680	0.646577 ± 0.002529	0.018668	0.007068	0.222038	0.360251
Cube	G14	0.517665 ± 0.007933	0.516603 ± 0.002968	0.022169	0.008295	0.207251	0.406565
Cube	G10	0.632640 ± 0.008768	0.633265 ± 0.002672	0.024502	0.007467	0.194032	0.399380
Cube	G6	0.589924 ± 0.010021	0.589206 ± 0.002687	0.028003	0.007508	0.203340	0.395320
Wall	G22	0.832500 ± 0.005428	0.746579 ± 0.002871	0.015168	0.008024	0.654331	0.680672
Wall	G18	**0.747364 ± 0.006263**	0.746972 ± 0.003032	0.017502	0.008473	0.652029	0.673407
Wall	G14	0.791053 ± 0.020088	0.791264 ± 0.003442	0.056135	0.009618	0.652318	0.675357
Wall	G10	0.806529 ± 0.009269	0.806284 ± 0.002958	0.025902	0.008265	0.630157	0.669606
Wall	G6	0.780092 ± 0.010605	0.780462 ± 0.002816	0.029636	0.007870	0.635119	0.671810
Street	G22	0.810434 ± 0.004175	0.809075 ± 0.002740	0.011668	0.007656	0.652529	0.741287
Street	G18	0.743296 ± 0.005845	0.743951 ± 0.003119	0.016335	0.008716	0.663258	0.698976
Street	G14	0.746577 ± 0.007098	0.746826 ± 0.002906	0.019835	0.008120	0.591021	0.662430
Street	G10	0.637710 ± 0.007933	0.641211 ± 0.003197	0.022169	0.008934	0.581495	**0.664391**
Street	G6	0.750670 ± 0.008852	0.750181 ± 0.003453	0.024736	0.009649	0.638049	0.722894
Circle	N0.1	0.959942 ± 0.012526	0.959575 ± 0.002985	0.035003	0.008342	0.416076	0.910307
Circle	N0.2	0.919787 ± 0.014196	0.922108 ± 0.003108	0.039670	0.008684	0.128539	0.896144
Cicle	N0.3	0.868376 ± 0.015448	0.872434 ± 0.002940	0.043171	0.008216	0.009316	0.863126
Circle	N0.4	0.805718 ± 0.016283	0.805952 ± 0.002973	0.045504	0.008307	0.003783	0.551743
Circle	N0.5	0.457309 ± 0.017536	0.453197 ± 0.002472	0.049004	0.006908	0.001893	0.019552
Saturn	N0.1	0.419754 ± 0.011273	0.421777 ± 0.002676	0.031503	0.007479	0.374629	0.389246
Saturn	N0.2	0.468760 ± 0.012943	0.470071 ± 0.002718	0.036170	0.007596	0.114122	0.394962
Saturn	N0.3	0.484417 ± 0.014613	0.483590 ± 0.002852	0.040837	0.007969	0.008486	0.365243
Saturn	N0.4	0.344146 ± 0.016283	0.191153 ± 0.003083	0.045504	0.008615	0.003533	0.249544
Saturn	N0.5	0.191539 ± 0.016283	**0.192462 ± 0.002698**	0.045504	0.007539	0.001750	0.007544
Cube	N0.1	0.570007 ± 0.012108	0.569811 ± 0.002997	0.033836	0.008375	0.238533	0.451012
Cube	N0.2	0.534157 ± 0.012943	0.535551 ± 0.002924	0.036170	0.008171	0.066385	0.430736
Cube	N0.3	0.535441 ± 0.013778	0.534368 ± 0.002878	0.038503	0.008043	0.005257	0.393973
Cube	N0.4	0.406655 ± 0.015448	0.406561 ± 0.002501	0.043171	0.006988	0.002173	0.263651
Cube	N0.5	0.291388 ± 0.016283	0.291420 ± 0.003122	0.045504	0.008724	0.001052	0.009412
Wall	N0.1	0.474320 ± 0.006263	0.477228 ± 0.002531	0.017502	0.007074	**0.612840**	0.394115
Wall	N0.2	0.485948 ± 0.007933	0.488712 ± 0.002757	0.022169	0.007705	0.116294	0.388994
Wall	N0.3	0.581962 ± 0.008768	0.582185 ± 0.002954	0.024502	0.008256	0.005233	0.369926
Wall	N0.4	0.438475 ± 0.009603	0.440016 ± 0.002716	0.026836	0.007589	0.003154	0.241493
Wall	N0.5	0.254742 ± 0.010021	0.256399 ± 0.002995	0.028003	0.008369	0.002110	0.008363
Street	N0.1	0.542130 ± 0.009186	0.542094 ± 0.002988	0.025669	0.008349	0.492784	0.384163
Street	N0.2	0.503819 ± 0.012641	0.381383 ± 0.003045	0.035327	0.008508	0.147636	0.381919
Street	N0.3	0.456459 ± 0.010922	**0.456501 ± 0.002825**	0.030521	0.007895	0.005876	0.364330
Street	N0.4	0.413806 ± 0.013200	0.413252 ± 0.002178	0.036886	0.006086	0.004350	0.244949
Street	N0.5	0.274330 ± 0.019822	0.275494 ± 0.003303	0.055392	0.009229	0.002438	0.008015

Table 4.2: Comparison of number of fitness function evaluations of two proposed algorithms

Image	Gaussian Noise	Number of Fitness Evaluations		Impulse Noise	Number of Fitness Evaluations	
		PSO1	PSO2		PSO1	PSO2
Circle	G22	323030 ± 1312	207800 ± 920	N0.1	400715 ± 991	219534 ± 432
Circle	G18	327334 ± 1521	229185 ± 1212	N0.2	450759 ± 1235	252709 ± 672
Circle	G14	353063 ± 1715	244257 ± 1589	N0.3	490668 ± 1331	304466 ± 1005
Circle	G10	366924 ± 1777	243988 ± 2123	N0.4	539773 ± 1465	346415 ± 1226
Circle	G6	373982 ± 1796	253840 ± 2546	N0.5	581010 ± 1544	386201 ± 1364
Saturn	G22	476004 ± 902	345226 ± 414	N0.1	541637 ± 1102	354948 ± 589
Saturn	G18	484546 ± 1117	346531 ± 715	N0.2	584619 ± 1375	385045 ± 1069
Saturn	G14	524227 ± 1489	359331 ± 1002	N0.3	620979 ± 1629	419322 ± 1217
Saturn	G10	532034 ± 1822	369737 ± 1135	N0.4	667164 ± 1934	471066 ± 1340
Saturn	G6	545284 ± 2155	376747 ± 1319	N0.5	707622 ± 1822	528268 ± 1474
Cube	G22	469166 ± 1512	340213 ± 1132	N0.1	508684 ± 512	345031 ± 361
Cube	G18	496997 ± 1802	363181 ± 1345	N0.2	544439 ± 1013	374948 ± 612
Cube	G14	509903 ± 1937	380805 ± 1717	N0.3	585330 ± 1977	421404 ± 1023
Cube	G10	540693 ± 2158	381208 ± 1677	N0.4	631625 ± 2742	466059 ± 1408
Cube	G6	556178 ± 2282	397107 ± 1753	N0.5	670429 ± 3363	502716 ± 2027
Wall	G22	519792 ± 604	354213 ± 918	N0.1	562815 ± 1472	365010 ± 1239
Wall	G18	523865 ± 1287	367963 ± 1392	N0.2	607663 ± 1813	394998 ± 1472
Wall	G14	538217 ± 2380	386111 ± 1751	N0.3	651921 ± 2399	441756 ± 1851
Wall	G10	554694 ± 2605	407519 ± 2082	N0.4	689938 ± 3311	504399 ± 1936
Wall	G6	572290 ± 2757	407763 ± 2503	N0.5	734026 ± 4646	549808 ± 2066
Street	G22	461291 ± 812	287938 ± 376	N0.1	450404 ± 771	294990 ± 503
Street	G18	487904 ± 1172	282458 ± 710	N0.2	496485 ± 1326	324903 ± 693
Street	G14	512448 ± 1518	295158 ± 1002	N0.3	535521 ± 1589	384105 ± 792
Street	G10	526171 ± 1779	312752 ± 1627	N0.4	580484 ± 1989	428221 ± 947
Street	G6	565901 ± 2002	336680 ± 2317	N0.5	614524 ± 2280	471332 ± 1148

on the noise level. This is consistent with the number of fitness function evaluations in Table 4.2.

4.7 Summary

Detection of continuous edges is a hard problem and most edge detection algorithms produce broken edges in noisy images. This chapter firstly presented a novel constrained optimisation model for detecting continuous, thin and smooth edges in such images. Then two particle swarm optimisation-based algorithms were applied to search for good solutions. These two algorithms utilised two different constraint handling methods: penalising and preservation. The algorithms were examined and compared with a modified version of the Canny algorithm as a Gaussian filter-based edge detector and the robust rank order (RRO)-based algorithm as a statistical-based edge detector on two sets of images with different types and levels of noise. Pratt's figure of merit as a measure of localisation accuracy was used for the comparison of these algorithms. Experimental results showed that the proposed edge detectors are more robust under noisy conditions and their performances are higher than the Canny and RRO algorithms for the images corrupted by impulse and Gaussian noise. The proposed algorithm based on the penalising method is faster than the algorithm using the preservation method to handle the constraints.

In this chapter, Otsu's method as a global thresholding technique was used to estimate a threshold value required for the PSO-based algorithm. In the next chapter, we will equip the PSO-based edge detector with a local thresholding technique in order to improve its performance in the illuminated noisy images.

Chapter 5

A Local Thresholding Technique in PSO

In the previous chapter, Otsu's method as a global binarisation method was used to estimate a parameter of the PSO-based edge detector. This method is considered as a state-of-the art global binarisation technique and commonly used for thresholding edge magnitude images. Since it extracts global features from the whole of an image to estimate a global threshold value for the image binarisation, it often cannot perform well in illuminated noisy images. Therefore, several local binarisation methods have been proposed in the literature to binarise the illuminated images. In this chapter, the PSO-based edge detector will be equipped with a local binarisation technique to extract local features from the neighbourhood of each pixel in order to estimate a local threshold value for each pixel.

5.1 Introduction

In chapter 4, we developed a new encoding scheme and a fitness function for a PSO-based algorithm in order to detect edges in noisy images and compared its performance on real images corrupted by two different types of noise (Gaussian and impulse). We compared its localisation

accuracy with a modified version of the Canny algorithm as a Gaussian filter-based edge detector equipped with an adaptive hysteresis thresholding technique to detect continuous edges and applying a non-maxima suppression (NMS) technique to detect thin edges. We demonstrated that the PSO-based algorithm can work better than Canny and RRO while Canny produced many speckles and broken edges in noisy images and RRO recognised edges more thickly than the others. However Canny and RRO operated better than PSO in a few cases. We equipped the PSO-based algorithm with an adaptive method by use of Otsu's method [155] to estimate one of its parameters. For a fair comparison, we used a dynamic hysteresis thresholding proposed in [163] in order to have better connected edges and an adaptive filter size proposed in [64] in order to overcome noise for the Canny algorithm. We also set the edge-height parameter of the RRO detector to the value which gave the highest localisation accuracy. The results showed that the PSO-based algorithm generally outperforms RRO and Canny in noisy images but there were still broken edges in the noisy image with illuminated areas.

Since most edge detection algorithms consider the thresholding step as a simple binarisation method, they utilise a binarisation method, such as Otsu's method to suppress false edges. Although Otsu's method is better than other global binarisation methods [167], it cannot work well in illuminated images [168]. Many local binarisation methods have been proposed to solve this problem by extracting local features for each pixel. Sauvola and Peitikain proposed an adaptive local binarisation method for document images [169]. Even though the performance of this method is higher than other global and local binarisation methods for the illuminated noisy images [168], it is not applicable for binarisation of edge magnitude images resulted by edge detection algorithms.

5.1.1 Chapter Goals

The main goals of this chapter are as follows:

- Increasing the localisation accuracy of the PSO-based edge detector in the illuminated noisy images
- Reducing the number of broken edges in such images
- Introducing a novel local thresholding technique by use of the Sauvola-Pietkinen method for the PSO-based edge detection algorithm.

The rest of this chapter is organised as follows. Section 5.2 provides background information on thresholding techniques including local and global binarisation methods.The new local thresholding technique will be introduced in Section 5.3. Sections 5.4 and 5.5 presents discussion on experimental results followed by a summary in Section 5.6.

5.2 Thresholding Techniques

Thresholding techniques in edge detection can be categorised into two main groups: (a) global thresholding techniques, which apply the features extracted from the whole of an image; and (b) local thresholding techniques, which use local features to choose a threshold value for each pixel on an image in order to identify edges. This section provides a brief overview of global and local thresholding techniques in edge detection.

5.2.1 Global Thresholding Techniques

Most thresholding techniques operate as image binarisation methods in the field of image segmentation. In these techniques, edge magnitude images are used as inputs to binarise. Global binarisation techniques use

global features extracted from the whole image to estimate a single threshold value for the image binarisation. Local thresholding techniques utilise local features extracted from a small area around each pixel.

Many different methods have been developed to binarise greyscale images. Otsu [155] proposed a global binarisation method which first estimates a single global threshold value for the binarisation of a greyscale image and then uses this value to assign each pixel on the image either to background or foreground. As described in Section 4.3.3, Otsu's method is a method based on maximisation of between-class variance. This method chooses a threshold value for the binarisation of an image such that it maximises the interset variance between the intensity of background and foreground pixels [155]. Ridler and Calvard [170] developed another method based on an iterative selection method to minimise inter-class variance. A global entropy-based thresholding technique was proposed by Kapur [171]. This method uses the entropy of the greylevel histogram of an image to estimate the global threshold value to binarise the image. Tsai [172] presented a method to automatically select a global threshold value based on the moment-preserving principle. In this method, the threshold value is chosen such that the moments of the image of interest are preserved in the binarised image. Rosin [173] proposed a thresholding algorithm based on finding a corner in the greylevel histogram. Rosin compared his proposed method with the methods proposed by Otsu [155], Ridler and Calvard [170], and showed that his algorithm performs better than those methods. However, the comparison was not comprehensive because of not considering real images in his objective experiments. Medina-Carnicer et al. [174] comprehensively evaluated the performance of seven different state-of-the art global thresholding techniques, such as the methods proposed by Otsu [155], Ridler and Calvard [170], Rosin [173] and Tsai [172] in synthetic and real images. They showed that Otsu's method generally outperforms the other methods. Otsu's method is more commonly used for thresholding edge magnitude images resulting from edge detectors.

5.2.2 Local Thresholding Techniques

Although global thresholding techniques are computationally fast and their performance is good in greylevel images, they cannot perform well in illuminated noisy images. Therefore, several local binarisation methods have been proposed to binarise the illuminated greylevel images.

In this category of thresholding algorithms, a threshold value is estimated for each pixel. The local threshold value depends on some local statistical features extracted from the neighbourhood of each pixel. These statistical features can be range or variance. Nakagawa and Rosenfeld [175] and Deravi and Pal [176] proposed two local thresholding techniques for the first time. However, Niblack [177] showed that their performance was not as good as the global shareholding techniques. He presented a method which calculated a local threshold value for each pixel on an image based on the local mean and variance inside a sliding window:

$$T(i,j) = m(i,j) + ks(i,j) \tag{5.1}$$

where (i,j) is the pixel located on the centre of the sliding window, $m(i,j)$ and $s(i,j)$ are the mean and the variance of intensity of all pixels in the window, and k is a constant between 0 and 1. Niblack's method does not perform well in the images with the background containing light texture as the intensity of these undesirable details easily exceed the estimated threshold values. Sauvola and Pietikinen [169] added a control factor to Equation (5.1) in order to solve this problem. This factor controls the dynamic range of variance in order to reduce the sensitivity of the method in such images. Bukhari et al. [168] showed that the Sauvola-Pietkinen method can enhance the illuminated areas better than other local and global binarisation methods.

5.3 New Local Thresholding Technique

The comparison of local binarisation methods experienced in [168] shows that the Sauvola-Pietkinen method proposed in [169] operates better than other types of local and global binarisation methods especially in images with illuminated areas. This method considers a grey scale document image as an array in which $I_{P'_{x,y}} \in [0, 255]$ is the intensity of pixel P' at position (x, y). In this method, the local threshold $TH_{P'_{x,y}}$ is estimated using the local mean $m_{P'_{x,y}}$ and standard deviation $s_{P'_{x,y}}$ of the pixel intensities in a $W \times W$ window centred around pixel $P'_{x,y}$:

$$TH_{P'_{x,y}} = m_{P'_{x,y}} \left[1 + k \left(\frac{s_{P'_{x,y}}}{R} - 1\right)\right] \tag{5.2}$$

Here, R is the maximum value of the standard deviation whose value for a 8-bit grey level image is 128; k is a real parameter ranging from 0 to 1. This parameter controls the threshold value for pixel $P'_{x,y}$ such that the lower the value of k, the higher the threshold value from the local mean $m_{P'_{x,y}}$.

To the best of our knowledge, the Sauvola-Pietkinen method has never been applied for the binarisation of edge magnitude images. This is because of the different nature of document images and edge magnitude images. Document images are always positive and their background is white whereas edge magnitude images are negative and their background is black. To solve this problem, we first invert the local edge magnitude ($LocalEdgeMag(P_{x,y})$) and calculate its inverse as $1 - LocalEdgeMag(P_{x,y})$, and then scale the result in range $[0, 255]$ by multiplying by 255. Let $I_{P'_{x,y}} = (1 - LocalEdgeMag(P_{x,y})) \times 255$. Therefore, $m_{P'_{x,y}} = (1 - m_{P_{x,y}}) \times 255$ and $s_{P'_{x,y}} = 255 \times s_{P_{x,y}}$ where $m_{P_{x,y}}$ and $s_{P_{x,y}}$ are the local features at pixel $P_{x,y}$ in the edge magnitude image. Thus,

$$TH_{P_{x,y}} = (1 - m_{P_{x,y}}) \left[1 + k \left(\frac{255 s_{P_{x,y}}}{R} - 1\right)\right] \tag{5.3}$$

where $TH_{P_{x,y}}$ is in the range between 0 and 1. We use the threshold value estimated by equation (5.3) in equation (4.16) to calculate the possibility

score of each pixel. In this equation, the local mean and standard deviation are used to adapt the value of the threshold according to the magnitude of the edges inside of the local neighbourhood of each pixel. In the case of the neighbourhoods with high edge magnitudes, the threshold $TH_{P_{x,y}}$ is almost equal to $m_{P_{x,y}}$ and in the case of the neighbourhoods with low edge magnitudes, the threshold value is less than the local mean in order to relatively increase the possibility score of the weak edges in these regions.

5.3.1 Different Degrees of Integral Images

In order to calculate $TH_{P_{x,y}}$, the local mean and standard deviation should be computed for each pixel. We use the concept of integral image which is an intermediate representation for an image in order to compute rectangle features in computer vision using an efficient way [178]. We generalise this concept to the different integral degrees of an image and show integral degree i (Ω^i) of an edge magnitude image as equation (5.4):

$$\Omega^i(x,y) = \sum_{p=0}^{x}\sum_{q=0}^{y} LocalEdgeMag^i(P_{p,q}) \tag{5.4}$$

The integral image can be calculated in an efficient manner by equation (5.5).

$$\begin{aligned}\Omega^i(x,y) &= \Omega^i(x-1,y)+\Omega^i(x,y-1)-\Omega^i(x-1,y-1)\\ &\quad +LocalEdgeMag^i(P_{x,y})\end{aligned} \tag{5.5}$$

5.3.2 Calculation of Local Features using Different Degrees of Integral Images

The local mean and standard deviation can be easily calculated as equations (5.6) and (5.7) once the integral images are computed.

$$\begin{aligned} m_{P_{x,y}} = \quad & (\Omega^1(x+(W-1)/2, y+(W-1)/2) + \\ & \Omega^1(x-(W-1)/2, y-(W-1)/2) - \\ & \Omega^1(x-(W-1)/2, y+(W-1)/2) - \\ & \Omega^1(x+(W-1)/2, y-(W-1)/2))/W^2 \end{aligned} \tag{5.6}$$

Since the variance of a variable is always equal to the expectation of the square of the variable minus the square of the mean of the variable,

$$\begin{aligned} s^2_{P_{x,y}} \quad = \quad & (\Omega^2(x+(W-1)/2, y+(W-1)/2) + \\ & \Omega^2(x-(W-1)/2, y-(W-1)/2) - \\ & \Omega^2(x-(W-1)/2, y+(W-1)/2) - \\ & \Omega^2(x+(W-1)/2, y-(W-1)/2))/W^2 \\ & -m^2_{P_{x,y}} \end{aligned} \tag{5.7}$$

5.4 Experimental Design

Although the proposed local thresholding technique can be applied to any edge detector which can estimate edge magnitudes of the pixels of an image such as different order derivatives [10][154], we apply this technique to the PSO-based edge detector which is more insensitive to noise than other edge detectors such as Canny and RRO. To examine the performance of the new local thresholding technique, we compare our previous PSO-based algorithm (PSO2) proposed in Chapter 4, utilising the sigmoid function with parameter TH estimated by Otsu's method, with the improved PSO-based algorithm (PSO3) utilising the same function with parameter

$TH_{P_{x,y}}$ estimated by the proposed method. This section provides the details on the two image sets, objective performance measure, and parameter settings, which are used in the experiments.

5.4.1 Image Sets

We use two different image sets in the experiments. The images in the first set are clean and easily accessible through the South Florida University database as described in Chapter 3. Since the main goal of the algorithm is the detection of continuous edges in noisy and illuminated images, we first chose three images from this database with illuminated areas which are commonly used as benchmarks for edge detection: Lena, egg and rubbish bin. Then these images are corrupted by two different types of noise: impulse and Gaussian. The second image set includes the images which were used for an objective (quantitative) comparison. This image set contains four images (Saturn, multi-cube, wall and road) as shown in Chapter 3. In addition to those four images, we added two images (rubbish-bin and egg) to the second image set. The ground truth images of these two images are recently available to download from [179]. For the images corrupted by the impulse noise, the noise probability ranges from 0.1 to 0.5 with a step size of 0.05. For the Gaussian noise, PSNR ranges from 0 to $22dB$ with a step size of $1dB$.

5.4.2 Parameter Settings

In PSO2 and PSO3, the population size is 50 and the maximum number of iterations was set at 200 according to the chosen particle length. We used the values $w = 0.7298$, $c_1 = 1.4962$, $c_2 = 1.4962$ for the parameters in Equation (2.43) [166]. The minimum length of a continuous edge, $2L + 1$ was set at 21 and $SqrSize$ at 6 [28]. The parameters of the novel thresholding techniques, W and k, were respectively set at 21 and 0.05 [168]. The experiments in [168] showed that a small value of k like 0.05 gives better results

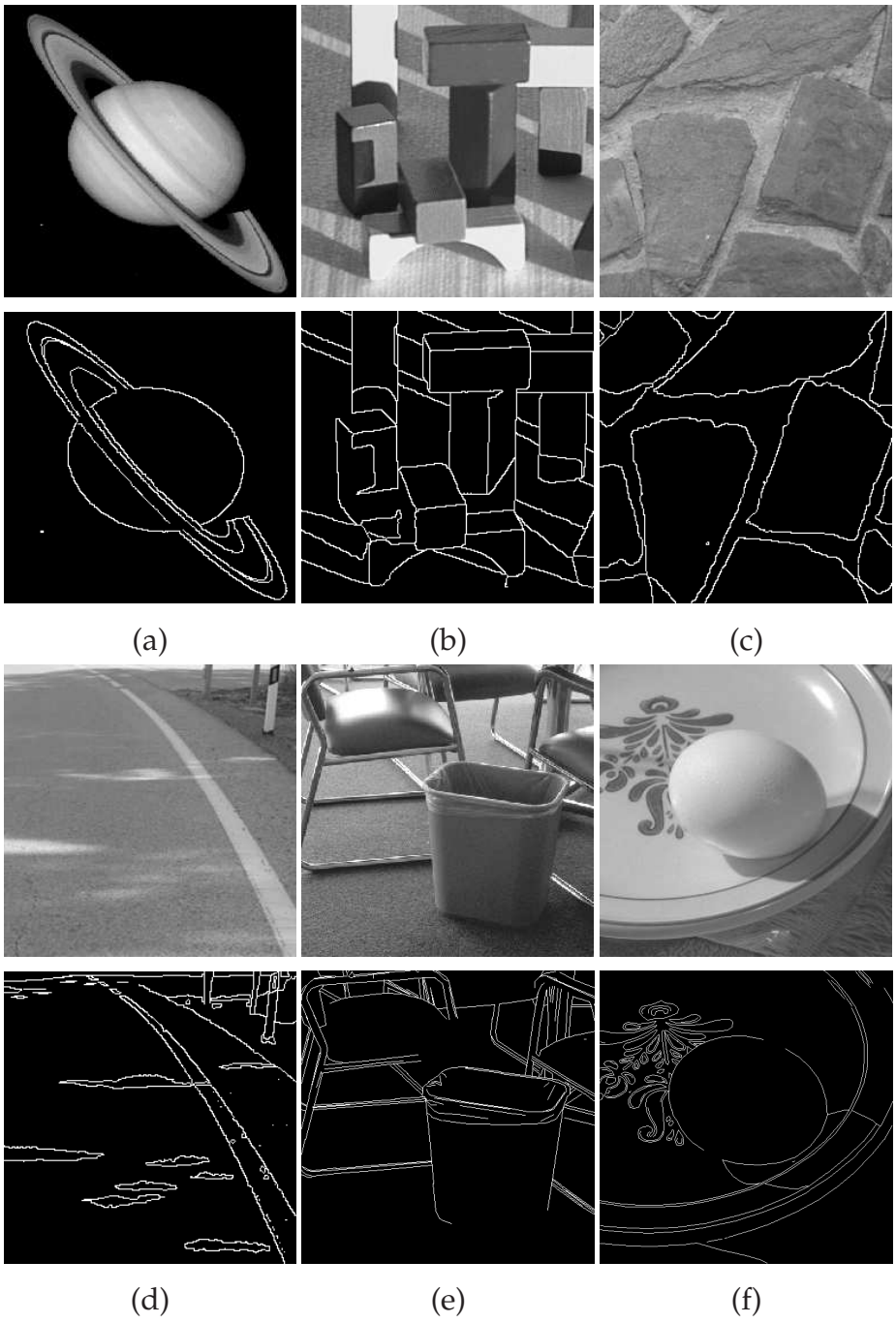


Figure 5.1: Example images for objective comparision. (a)–(d) four real image from the UCO university and their manual ground truth images [137], (e) and (f) two real images from the South Florida University and their ground truth images [179].

with fewer broken edges.

5.5 Results and Discussion

This section presents the results of the subjective and objective comparison of PSO2 and PSO3 followed by a discussion.

5.5.1 Subjective/Qualitative Comparison

The resulting images are shown in Figures 5.2 and 5.3 after applying PSO2 and PSO3 in the first set (Figure 5.1) corrupted by impulse and Gaussian noises respectively. As can be seen in Figure 5.2, the edges detected by PSO3 were improved on Lena's hat especially on the top which is an illuminated area and they are more connected than the edges recognised by PSO2. For the egg image, PSO3 could operate better than PSO2 around the egg. However, there are still broken edges in this area. This problem may be solved by increasing parameter L. Although there are several false edges and still broken edges, PSO3 improved the edges in the rubbish-bin image particularly on its bottom-right corner. The results suggest that the new local thresholding technique performs better than Otsu's method in the illuminated noisy images.

5.5.2 Objective/Quantitative Comparison

For an objective comparison of PSO2 and PSO3, the localisation accuracy (PFOM) was calculated from the resulting images after applying both algorithms to the images in the second set (Figure 5.1) at different noise levels. Figure 5.4 depicts the graphs in which the average of the resulting PFOM values after 30 runs are plotted versus different noise levels for each image corrupted by impulse noise. The noise probability ranges from 0.1 to 0.5 with step of 0.05. The plots in Figure 5.4 indicate that PSO3 generally performed better than PSO2. Statistical analysis showed that PSO3

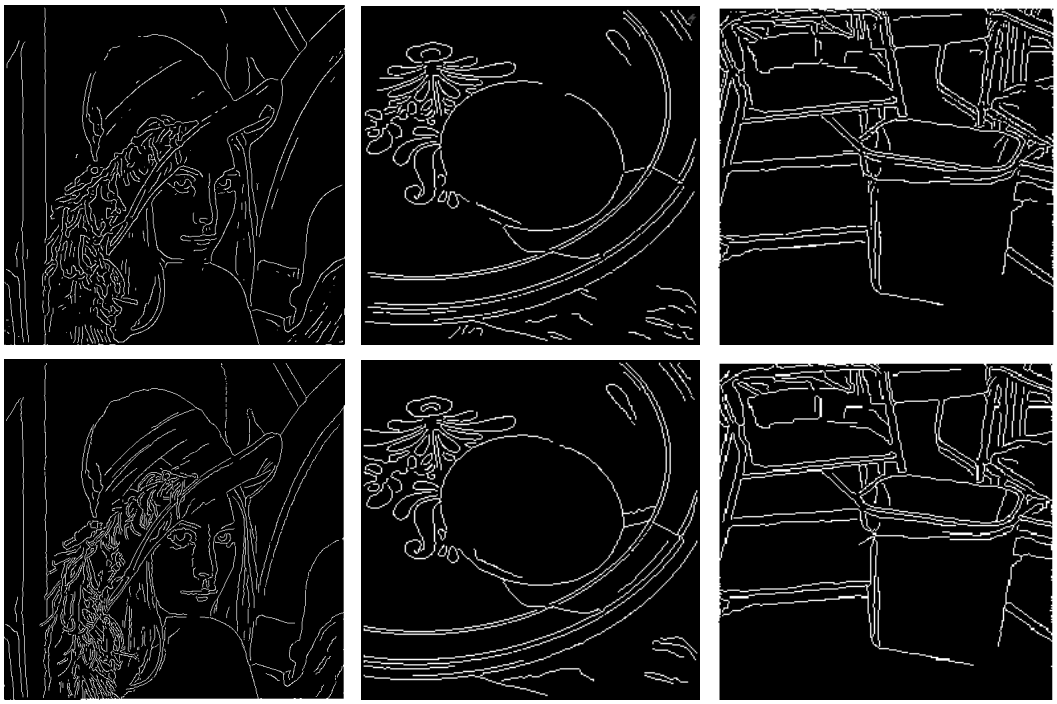


Figure 5.2: Subjective results of edge detection produced by PSO2 (top row) and PSO3 (bottom row) on the three images corrupted by *impulse noise* (noise probability=0.1).

has a higher accuracy in 34 cases out of 54. For the first four images, PSO3 could detect the edges more accurately than PSO2 in 31 cases out of 36 (see Figures 5.4(a), (b), (c) and (d)). For the last two images, its accuracy is lower in 15 cases out of 18 (see Figures 5.4(e) and (f)). A reason is that there are a few "false positive" edges which were recognised by PSO3 as edges whereas they were actually *incorrectly* labelled as non-edges in their ground truth images. This caused that the real improvements of PSO3 were considered "false".

For each image corrupted by Gaussian noise, the average of the resulting PFOM values after 30 runs are plotted versus different noise levels in Figure 5.5. PSNR ranges from 0 to 22dB with step of 1dB. The plots in

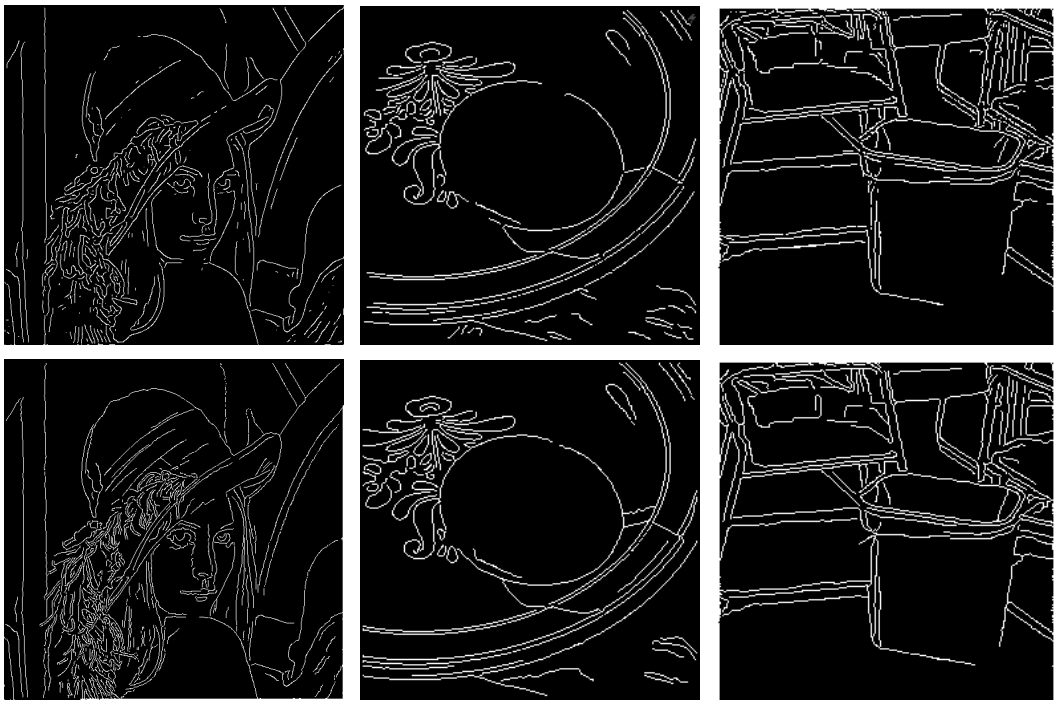
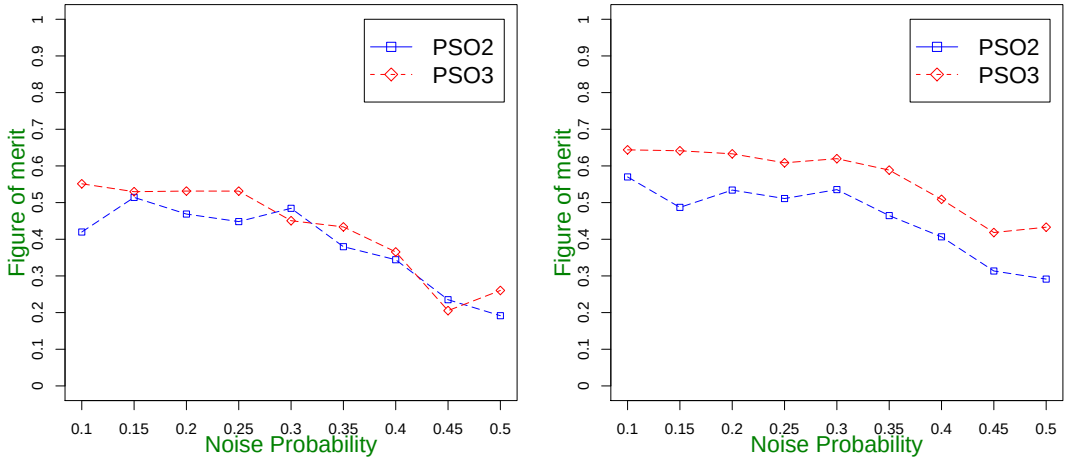
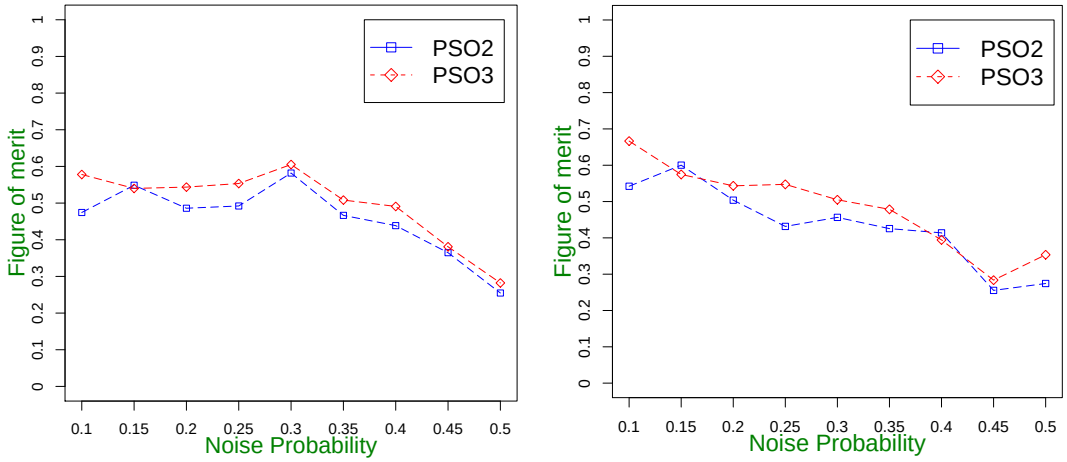


Figure 5.3: Subjective results of edge detection produced by PSO2 (top row) and PSO3 (bottom row) on the three images with *Gaussian noise* (PSNR=16dB).

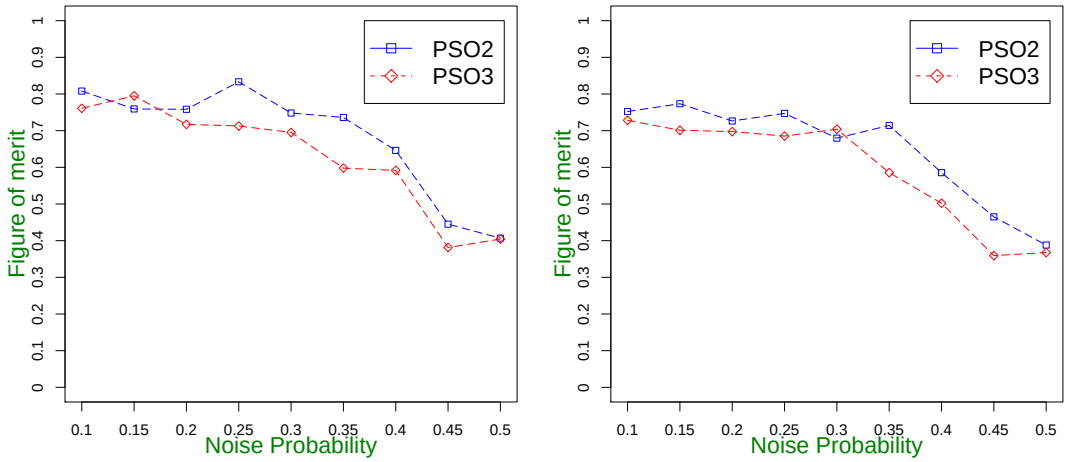
Figure 5.5 indicate that PSO3 has a higher accuracy than PSO2 in the most images corrupted by Gaussian noise. Statistical analysis showed that its accuracy is higher in 83 cases out of 138. For the first four images, PSO3 could detect the edges more accurately than PSO2 in 79 cases out of 92 (see Figures 5.5(a), (b), (c) and (d)). For the last two images, its accuracy is lower in 42 cases out of 46 (see Figures 5.5(e) and (f)). A reason is again that there are a few "false positive" edges which were recognised by PSO3. As the objective results show, PSO3 generally performs better than PSO2. Figure 5.2 and 5.3 clearly show that PSO3 can even detect the difficult edges in the boundary of the egg and rubbish-bin that were not labelled in the ground truth images.



(a) Saturn (b) Cube

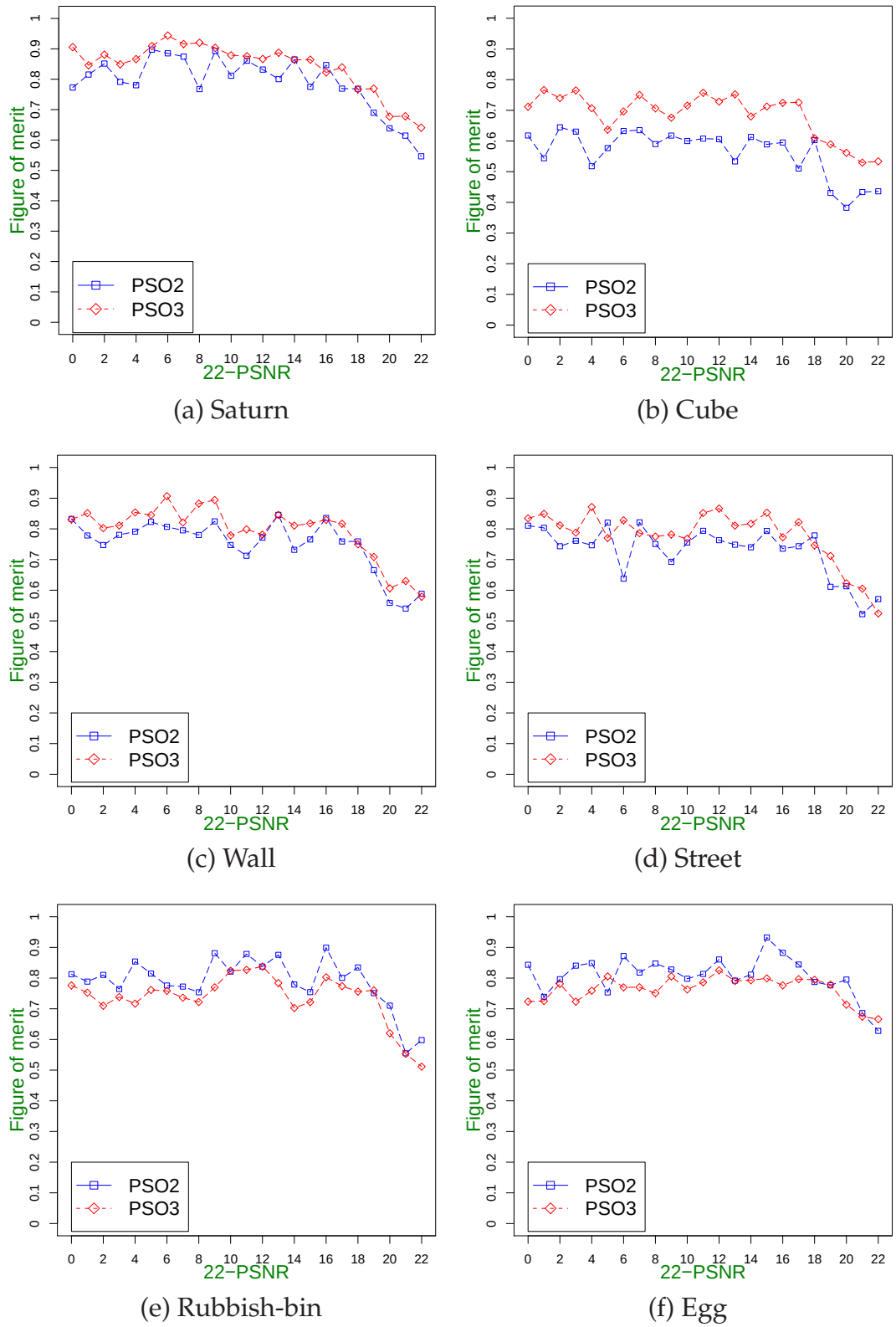


(c) Wall (d) Street



(e) Rubbish-bin (f) Egg

Figure 5.4: PFOM vs. noise level for Saturn, cube, wall, street, rubbish-bin and egg images in the second image set with different impulse noise levels (the noise probability ranging from 0.1 to 0.5).



(a) Saturn (b) Cube

(c) Wall (d) Street

(e) Rubbish-bin (f) Egg

Figure 5.5: PFOM vs. noise level for Saturn, cube, wall, street, rubbish-bin and egg images in the second image set with different Gaussian noise levels (PSNR ranging from 0 to 22dB).

5.6 Summary

In this chapter, a novel local thresholding technique was proposed for a particle swarm optimisation (PSO)-based algorithm to detect edges with greater continuity. The new technique was based on the Sauvola-Pietkinen method which is often used for binarising the illuminated document images but normally cannot be applied to edge magnitude images. This method was equipped by an integral imaging technique for more efficiency and adopted into the PSO-based algorithm to detect edges in grey level illuminated noisy images. We compared the performance of the new algorithm with our previous PSO-based edge detector utilising Otsu's method which is commonly used as a thresholding technique in edge detection. Experimental results showed that the PSO-based algorithm utilising the new local thresholding technique performs better than the one that uses Otsu's method.

Since in all PSO-based edge detectors which have been proposed so far, the fully connected graph has been chosen as a neighbourhood structure, the effects of static topologies in the PSO-based edge detector will be investigated in the next chapter.

Chapter 6

Effects of Static Topologies in PSO-based Edge Detector

In the previous chapter, we revised the PSO-based algorithm with a novel local thresholding technique based on the Sauvola-Pietkinen method [169] and improved the performance of the PSO-based algorithm in the illuminated areas. In all PSO-based edge detectors which have been proposed in this thesis so far, the fully connected graph has been chosen as a neighbourhood structure. The effects of other static topologies in the PSO-based edge detector will be investigated in this chapter.

6.1 Introduction

Since the basic PSO algorithm was developed by Kennedy and Eberhart in 1995 [116], several versions of PSO have been proposed to improve the performance of the basic algorithm [115, 180]. It has been shown that the performance of the basic PSO algorithm strongly depends on the values of the self and swarm confidence coefficients and the inertia weight [181, 182]. So other methods have been proposed to reduce the sensitivity of the basic PSO algorithm to such parameters [180]. The most recent research has also shown that the chosen neighbourhood topology affects

the performance of the PSO algorithm and its effect depends on the function (task) being optimised. A neighbourhood topology may perform well for one function but could not work well for another function. Kennedy [183] and Bratton and Kennedy [184] demonstrated that choosing an ideal topology requires a complete experimentation for a particular problem.

There are two major types of topology in PSO: static and dynamic. Since static topologies are very popular because of their speed and the ease of their implementation, in this chapter, we concentrate on the static topologies and in the next chapter, we will focus on dynamic topologies.

In order to widely study the performance of using different static topologies within PSO, six different well-known topologies, i.e., fully connected, ring, star, tree-based, von Neumann (mesh) and toroidal topologies have been selected to investigate their performances for the PSO-based edge detectors. These topologies will be utilised within three PSO versions: the Canonical or basic PSO algorithm (CanPSO) [181], the Bare Bones PSO (BBPSO) [185] and the Fully Informed Particle Swarm (FIPS) [165].

6.1.1 Chapter Goals

Since the performance of static topologies is different in various applications and in various versions of PSO, in this chapter, we aim to:

- investigate the performance of three well-known versions of PSO (Canonical PSO, Bare Bones PSO and Fully informed PSO) on the detection of edges in noisy images;
- compare the accuracy of these three algorithms equipped with six well-known static topologies (fully connected, ring, star, tree-based, von Neumann and toroidal topologies);

The rest of this chapter is organised as follows. Sections 6.2 and 6.3 provide a detailed description of the three well-known versions of PSO and the six neighbourhood topologies. Sections 6.5 and 6.6 present discussion on the experimental results followed by a summary in Section 6.8.

6.2 Three Well-Known PSO Methods

In the literature, several variant versions of PSO have been proposed. Three of them, i.e., the canonical PSO, bare bones PSO and fully informed particle swarm are very popular. Since the performance of these three versions are different for different problems when they are equipped with different topologies, we briefly describe them in this section.

6.2.1 Canonical PSO (CanPSO)

The canonical (constricted) PSO was proposed by Clerc and Kennedy [181]. This method introduced the constriction factor, χ, in order to guarantee the convergence of the PSO algorithm. This factor is calculated from the existing parameters of the PSO algorithm, i.e, the self (C_1) and swarm (C_2) confidence learning factors as follows:

$$\chi = \frac{2}{|2 - \phi - \sqrt{\phi^2 - 4\phi}|} \tag{6.1}$$

where $\phi = C_1 + C_2 > 4$ to guarantee convergence. Clerc and Kennedy showed that when $\phi < 4$, the particles in the PSO algorithm would slowly spiral toward the position of the best particle and there is no guarantee to converge, while for $\phi > 4$, the particles would quickly converge. The constricted PSO algorithm uses the equal values for the confidence factors for the sake of simplicity. The velocity in the canonical PSO with the constriction factor is computed as:

$$\begin{aligned} V_{i,j}(t+1) &= \chi[V_{i,j}(t) + C_1 r_{1j}(X_{pbest_i,j} - X_{i,j}(t)) \\ &\quad + C_2 r_{2j}(X_{leader,j} - X_{i,j}(t))] \end{aligned} \tag{6.2}$$

When $\phi = 4.1$, $\chi = 0.72984$ and $C_1 = C_2 = 2.05$. With this parameter setting, the canonical PSO is equivalent with the standard PSO regarding to the values of their parameters. In fact, the canonical PSO is a special case of the standard PSO whose parameters have been chosen analytically [186].

6.2.2 Bare Bones PSO (BBPSO)

In 2002, Clerc and Kennedy [181] proved that all particles in the PSO population converge to a weighted average of their neighbourhood and personal best positions. According to this idea, Kennedy [185] proposed an almost parameter free PSO algorithm which omits the influence of current motion of each particle (velocity) and uses a simple Gaussian distribution to compute the new position of each particle as in Equation (6.3). The distribution mean is the average of the personal best and leader positions, and its standard deviation is the difference of their positions. This was called Bare Bones PSO (BBPSO).

$$\begin{aligned} \vec{X}_{i,j}(t+1) \;\sim\; & N\Big(\mu_{i,j} = (\vec{X}_{leader,j} + \vec{X}_{pbest_i,j})/2, \\ & \sigma^2_{i,j} = (\vec{X}_{leader,j} - \vec{X}_{pbest_i,j})^2\Big) \end{aligned} \tag{6.3}$$

In BBPSO, the particles move with a larger step towards their leader's position if their personal best position is far from the position of their leader. This may cause the personal best position to quickly move towards the leader's position. In such cases, the step size becomes small and accordingly the exploration ability of PSO reduces in support of exploitation. We expect that this feature of BBPSO along with different topologies would affect its performance on the detection of edges in noisy images.

6.2.3 Fully Informed Particle Swarm (FIPS)

In CanPSO and BBPSO, each particle shares information just with the best neighbour whereas in FIPS, each particle is influenced by all of its neighbours specified by a neighbourhood topology. Therefore, in FIPS there is a stronger swarm influence than the other versions [165]. The new velocity

of each particle in this version is calculated as:

$$\vec{V}_{i,j}(t+1) = \chi\left(\vec{V}_{i,j}(t) + \sum_{n=1}^{N} \frac{U(0,\phi)(\vec{X}_{nbr(n),j} - \vec{X}_{i,j})}{N}\right) \quad (6.4)$$

Here, N is the number of particles in the neighbourhood of each particle; $U(0,\phi)$ is a uniform random variable between 0 and ϕ; and $\vec{X}_{nbr(n)}$ is the position of its n-th neighbour. In FIPS, none of the particles in the population is not influenced by their personal best position. Since each particle in FIPS is usually influenced by a more local neighbourhood than the other versions, its population usually has a higher diversity. Since the velocity of each particle is influenced by the average between its neighbours' position and its current position, we hypothesise that this version of PSO can deal with noisy images better than other versions.

6.3 A Classification of PSO Neighbourhood Topologies

An important feature of the PSO algorithm is a topology that defines how particles are connected to each other and how they exchange or share the information that they have found so far [115]. The neighbourhood topology influences the speed of information flow among particles. Since the exploration and exploitation abilities of the PSO algorithm can be controlled by adjusting the speed of information flow, the topology can be used as a mechanism to tune these abilities of the PSO algorithm.

A neighbourhood topology can be directed or undirected, static or dynamic, and regular, random or spatial (see Figure 6.1). A topology is directed if the flow of information among particles is one way; otherwise it is undirected. In a static topology, the neighbourhood structure among particles remains fixed throughout the PSO iterations whereas in a dynamic

<table>
<tr><td></td><td colspan="2">Information Flow</td></tr>
<tr><td>Structure</td><td>one-way</td><td>two-way</td></tr>
<tr><td rowspan="2">Static</td><td colspan="2">Regular</td></tr>
<tr><td rowspan="2">Dynamic</td><td colspan="2" rowspan="2">Random</td></tr>
<tr><td colspan="2">Spatial</td></tr>
</table>

Figure 6.1: A classification of PSO neighbourhood topologies.

one, it potentially changes at certain iterations. The neighbours of each particle in the PSO population can be specified through a regular or a random way. In the regular way, each particle is assigned a node in the graph representing the structure of a neighbourhood topology according to its index in the population. But in the random way, the particle is assigned by chance. In a spatial topology, the particles use the spatial information from the function or position space to choose their neighbours. Figure 6.1 shows a general classification of PSO topologies based on their characteristics. Since the major goal of this chapter is to investigate the influence of using static topologies within the PSO-based edge detector on its accuracy, we focus on the static topologies here and leave the dynamic topologies to the next chapter.

6.4 Static Topologies

Most static topologies are defined by a one-way (directed) or two-way (undirected) graph in which each node represents a particle in a PSO population and edges depict which particles are connected to each other. After initialisation of all particles, each node in the graph is assigned by a regular or a random way to a particle in the population and the neighbours of each particle are defined based on the edge structure of the graph. In the

literature, several typical neighbourhood topologies have been proposed as follows:

- Fully connected graph (FCG). This topology is commonly used as a neighbourhood structure in PSO because of its strong ability at exploitation and its high speed in convergence. However, it suffers from being trapped in local optima in most applications. In this topology, each particle is fully connected to the other particles [115] and is influenced by the best particle of the entire swarm (gbest), as well as its own past experience (pbest) in CanPSO and BBPSO and by all connected particles in FIPS. In CanPSO and BBPSO, the leader is global best particle ($leader = gbest$ in equations (6.2) and (6.3)). This topology is shown in Figure 6.2(a).

- Ring topology (RT). There are two immediate neighbours for each particle in the graph [115] as shown in Figure 6.2(b). Therefore each particle has a local best particle among two particles within its neighbourhood. In this topology, each particle is influenced by a leader in its local neighbourhood plus its own past experience (pbest). In this case, the leader is called the local best (lbest) particle. The most important features of the ring topology are its low speed of information sharing among particles and high exploration ability.

- Star graph (SG). In this case, one particle, which is called the focal particle, is just connected to all other particles [187] as shown in Figure 6.2(c). In this topology, particles are isolated from each other and they communicate through the focal particle. This topology is sometimes called the wheel topology. In this topology, $leader = focal$. The focal particle is selected randomly before PSO iterations start. All particles in the population move toward the focal particle at each iteration and its position is influenced by the best particle of its neighbourhood. Since the focal particle adjusts its flight direction toward its best neighbour particle in an iteration and the other particles tune

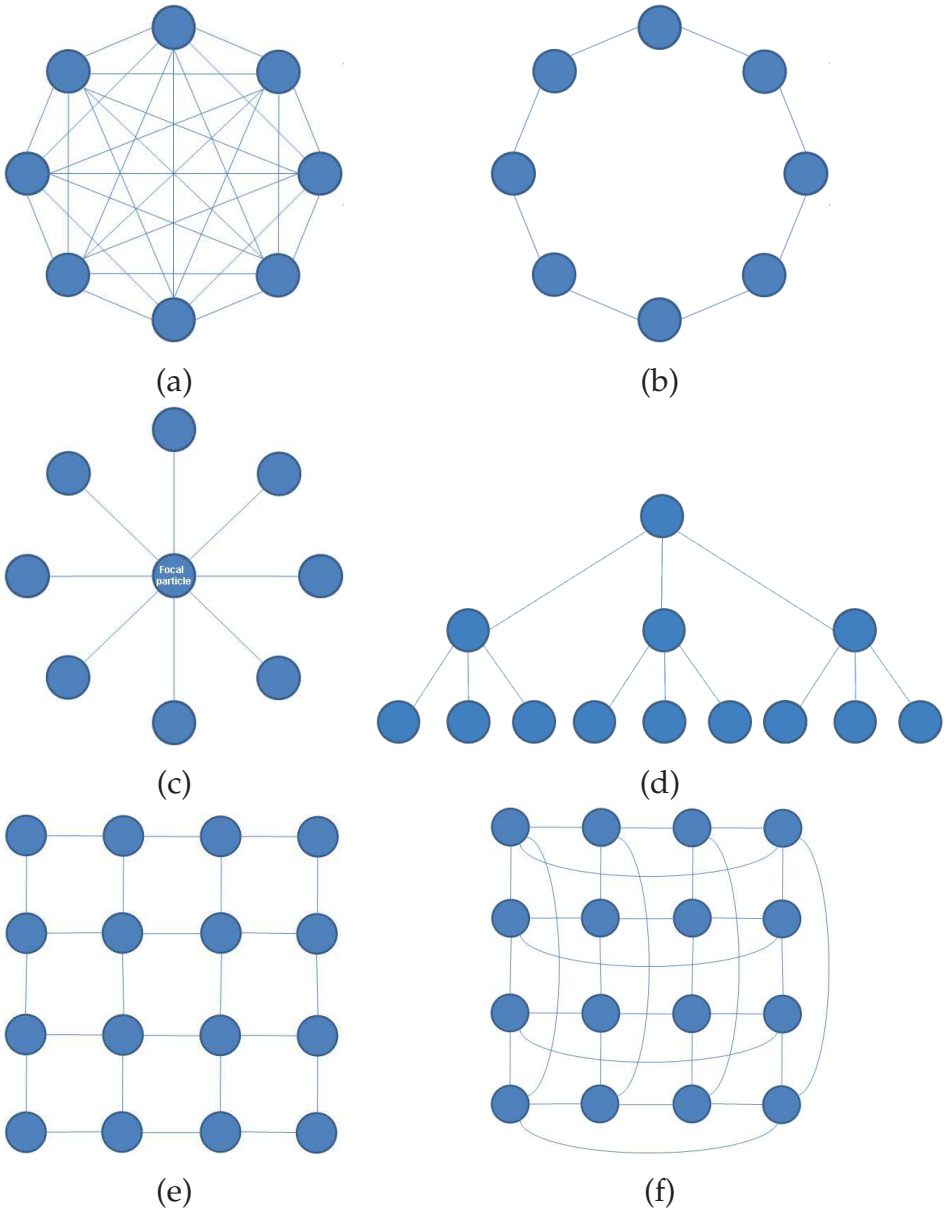


Figure 6.2: Some well-known topologies used in PSO: (a) FCG (b) RT, (c) ST, (d) TBG, (e) VNT and (f) TRO.

their trajectories toward the focal particle and their personal best positions in the next iteration, the focal particle acts as a buffer or a filter and slows down the transmission of information among particles. Thus the star topology can prevent PSO from becoming trapped in a local optima. Another important feature of the topology is to preserve the diversity of potential solutions but it may reduce the information sharing ability among the particles.

- Tree-based graph (TBG). In this topology which is also known as a hierarchical topology, there is a root particle at the top level of the hierarchy and all particles in the second level of the tree are connected to the root. These particles in the second level can also have several children in their neighbourhoods [187] (see Figure 6.2(d)). The leader of each particle is its parent in the tree in this topology. Whenever each child particle finds a solution which is better than the best particle found by its parent, the child and parent particles exchange their best personal information. In this topology, $leader = pbest_{parent}$ in equation (6.2). Note that the tree representing this topology may be incomplete according to the number of particles in the population. In this topology, each non-leaf particle behaves like the focal particle in the star topology and filters out information. A good solution can be propagated through the whole population at most in $\lceil 2\log_m(N) - 1 \rceil$ iterations where m is the branching factor (the maximum number of children for each node) of the tree topology.

- The von Neumann topology (VNT). In this case, each particle has up to four neighbours within its neighbourhood and exchanges the information with them [187]. The graph representing the corresponding topology can be 2-dimensional or generally n-dimensional. Each node in the 2-dimensional graph has up to four neighbour nodes in its four different directions (left, right, up and down). An example of the 2-dimensional graph is shown in Figure 6.2(e). In the 2D

graph which is most popular, each node located in the corner is just connected to two nodes. Since each particle in the PSO population is usually assigned to one node in the graph in a random way, the particles corresponding to the nodes at the corners only have two particles in their neighbourhoods. The particles, which are assigned to the nodes at the boundaries of the graph, only have three neighbouring particles.

- The toroidal topology (TRO). There are similarities between this topology and VNT. All particles in this topology have four adjacent particles. The particles at the boundaries of the graph representing the structure of VNT, have less than four particles in their neighbourhood. But these particles in the toroidal topology are connected to the particles on the other side of the graph in order to have four neighbours like other particles, as shown in Figure 6.2(f).

Mendes et al. [165] indicated that if the neighbourhood size (the number of neighbours) of a particle increases, the performance of PSO may become worse. On the other hand, if it decreases, the run time of the algorithm may be increased. Montes de Oca and Stützle [188] investigated the convergence behaviour of different versions of PSO and showed that their behaviour may be different from each other on different problems. They also showed that there is a strong relationship between the chosen topology and their robustness to premature convergence to optimise some benchmark non-linear functions. Kennedy [183] presented that one of the main causes of premature convergence in PSO is the kind of chosen topology. Kennedy [183] showed that the effect of using different topologies in FIPS is completely different from the two other versions of PSO. For example, FCG in the CanPSO means that each particle is influenced by the best information found by all particles, whereas in FIPS, each particle is influenced by all best solutions found by all other particles. Therefore, each particle in FIPS is impacted by numerous and diverse particles rather than

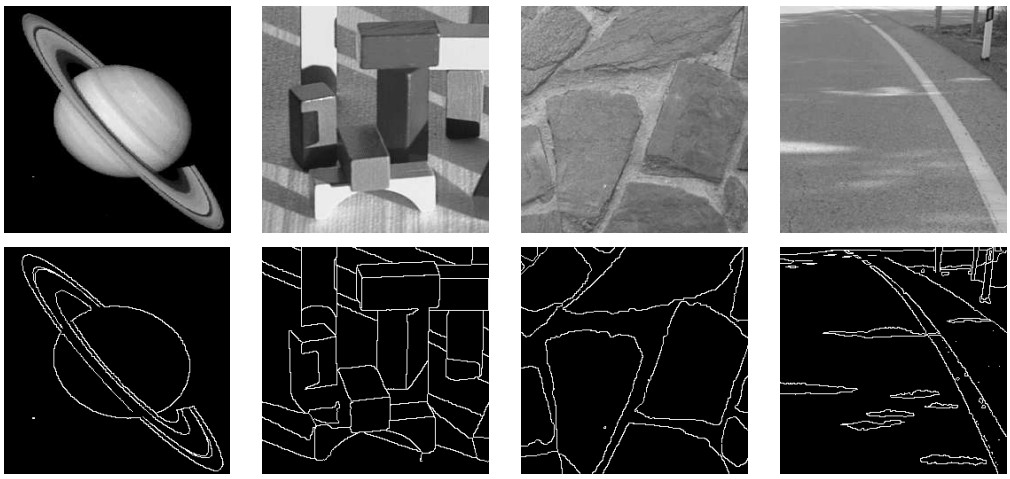
only one or two particles in CanPSO or BBPSO.

6.5 Experiment Design

This section presents the image set used in the experiments and parameter settings.

6.5.1 Image Set

To compare the performance of CanPSO, BBPSO and FIPS with different static topologies, we will apply these algorithms on a set of benchmark images from chapter 3. The image set includes four real images (Saturn, multi-cube, wall and street) with their ground trouth as shown in Figure 6.3.



(a) (b) (c) (d)

Figure 6.3: (a)-(d) Saturn, multi-cube, wall, street images and their ground truth images from UCO university [137].

6.5.2 Parameter Values

We use the values $\chi = 0.7298$, $C_1 = 2.05$, $C_2 = 2.05$ for the PSO parameters in Equations (6.2) and (6.3). The number of particles in the PSO was set at 50 and the maximum number of iterations was 200. These parameters were adjusted based on common settings [184]. The branching factor of the tree-based topology was set at 3. The parameters of our PSO-based edge detection algorithm were set at the values regarding to our experiments in Chapters 4 and 5 for comparison purposes. In our experiments, $max + 1 = 21$, $SqrSize = 6$, and $HP = 0.5$.

6.6 Results and Quantitative Comparison

This section discusses statistical and quantitative results obtained from applying different static topologies in the three well-known versions of PSO. In this section, we first present the results on using static topologies in the CanPSO-based edge detector followed by the results on using them in the BBPSO and FIPS-based edge detection algorithms. We then choose the best topology for each algorithm and compare their performances with each other when they are equipped with their own best topology. The detailed results are available in the appendix.

6.6.1 Results on Static Topologies within CanPSO

For an objective comparison, we first carried out our experiments with the CanPSO-based edge detection algorithm equipped with different static topologies. Figures 6.4 and 6.5 show the box plots resulted from the 30 independent runs of CanPSO with each topology in the images corrupted by Gaussian and impulse noise respectively. In these plots, the horizontal axis is the chosen topology and the vertical axis is the accuracy of the CanPSO-based edge detector. Here, G6, G10, G14, G18 and G22 represent PSNR from 6dB to 22dB with step 4 for Gaussian noise and N0.1, N0.2,

N0.3, N0.4 and N0.5 represent noise probability from 0.1 to 0.5 with step 0.1 for impulse noise. As can be seen in Figure 6.4, the accuracy of CanPSO with the ring topology is equal or higher than other topologies in images corrupted by Gaussian noise in most cases. Its accuracy is lower than other topologies only in the Saturn image corrupted by G10 and G22, and in the cube image corrupted by G10.

As can be seen in Figure 6.5, the accuracy of the CanPSO-based edge detector with the ring topology is not lower than other topologies when the images are corrupted by N0.4 and 0.5 but its accuracy is lower only in the Saturn image with N0.1 and N0.2, the cube image with N0.2 and 0.3 and the street image with N0.1. This shows that CanPSO with RT (CanPSO-RT) works better than other topologies when the images are corrupted by a higher level of noise and its accuracy becomes lower in the images with a lower level of impulse noise.

Table 6.1 shows the summary of the comparison of the accuracy of the CanPSO-based algorithm with FCG, RT, SG, TBG, VNT and TRO. For a fair and comprehensive comparison, we used a simple multiple comparison procedure proposed by Holm [189] to adjust p-values resulted from the Student two paired T-test to compare the pairwise accuracy means. As we expected, CanPSO with RT can outperform CanPSO with other static topologies in most cases and its accuracy with RT is higher or equal in 92% of the cases (184 out of 200). The summary in Table 6.1 shows the number of cases where each topology is statistically significantly same/better/worse than CanPSO with the other topologies. The results also show that the static topologies can be approximately ranked from highest accuracy to lowest accuracy as RT, {TRO, VNT}, {TBG, SG} and FCG.

6.6.2 Results on Static Topologies within BBPSO

We also examined the performance of BBPSO with different static topologies. Figure 6.6 shows the box plots from the results of its 30 independent

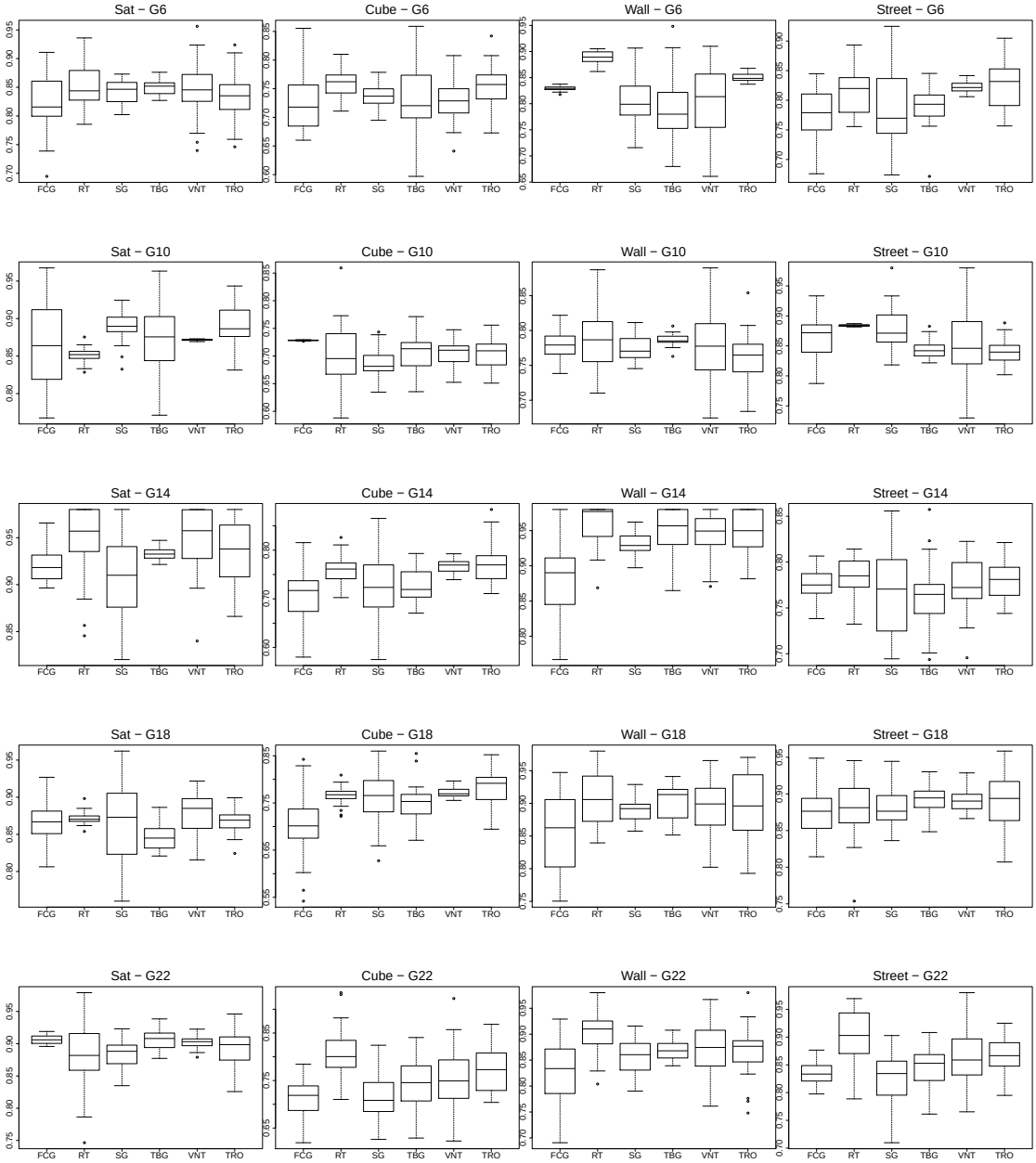


Figure 6.4: The box-plots drawn from the results of 30 independent runs of **CanPSO** with static topologies in noisy images corrupted by *Gaussian* noise.

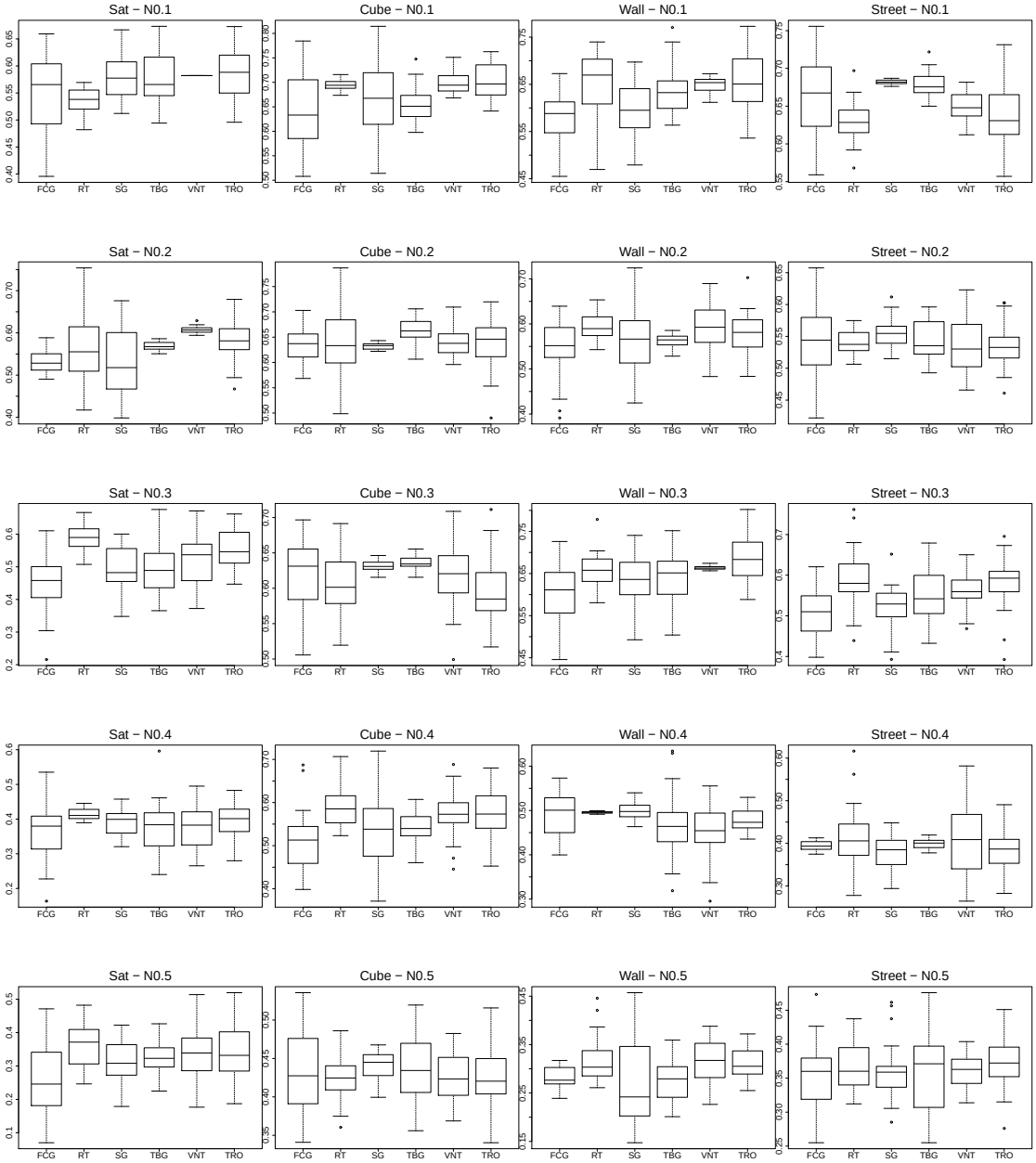


Figure 6.5: The box-plots drawn from the results of 30 independent runs of **CanPSO** with static topologies in noisy images corrupted by *impulse* noise.

Table 6.1: The performance of the **CanPSO**-based algorithm with different static topologies

Topology		FCG	RT	SG	TBG	VNT	TRO	Worse	Same	Better
	Worse	–	22	3	10	17	18	70		
FCG vs	Same	–	15	35	29	22	20		121	
	Better	–	3	2	1	1	2			9
	Worse	3	–	3	5	3	2	16		
RT vs	Same	15	–	23	25	30	33		126	
	Better	22	–	14	10	7	5			58
	Worse	2	14	–	3	7	11	37		
SG vs	Same	35	23	–	35	30	26		149	
	Better	3	3	–	2	3	3			14
	Worse	1	10	2	–	6	7	26		
TBG vs	Same	29	25	35	–	33	31		153	
	Better	10	5	3	–	1	2			21
	Worse	1	7	3	1	–	1	13		
VNT vs	Same	22	30	30	33	–	39		154	
	Better	17	3	7	6	–	0			33
	Worse	2	5	3	2	0	–	12		
TRO vs	Same	20	33	26	31	39	–		149	
	Better	18	2	11	7	1	–			39

runs with static topologies in noisy images corrupted by Gaussian noise. As can be seen in this figure, the BBPSO-based edge detector with RT performs better than other topologies in images with a high level of Gaussian noise (G6 and G10) in all cases whereas its accuracy decreases in images with a low level of the noise in a few cases (see Sat-G14, Sat-G22, Cube-G14, Street-G14 and Street G22 in Figure 6.6).

Figure 6.7 shows the box plots from the results of 30 independent runs of BBPSO with static topologies in noisy images corrupted by impulse noise. These plots indicate that the performance of BBPSO with RT (BBPSO-RT) is equal or higher than other static topologies in the images with a low level of impulse noise whereas its accuracy is lower in a few cases in the images with a high level of noise (see Sat-N0.4, Cube-N0.3, Wall-N0.4 and Street-N0.3 in Figure 6.7).

For a fair comparison, we again used Holm's method to adjust p-values resulted from the Student two paired T-test to compare the pairwise accuracy means. The results show that BBPSO with RT (BBPSO-RT) is more accurate than BBPSO with other static topologies in most cases. The performance of BBPSO with RT is statistically better or same in 86% of the cases (172 out of 200). Table 6.2 show the number of the cases where BBPSO-RT with each topology is same/better/worse than the other topologies. For the BBPSO-based edge detection algorithm, the static topologies can be approximately ranked as RT, TRO, {VNT, TBG} and {SG, FCG} from highest accuracy to lowest accuracy.

6.6.3 Results on Static Topologies within FIPS

For FIPS, the box plots are drawn from the results of 30 independent runs for each static topology in noisy images corrupted by Gaussian and impulse noise as can be seen in Figures 6.8 and 6.9. The FIPS-based edge detector performs better than other topologies in all noise level in all cases except the wall image corrupted by impulse noise with the noise probabil-

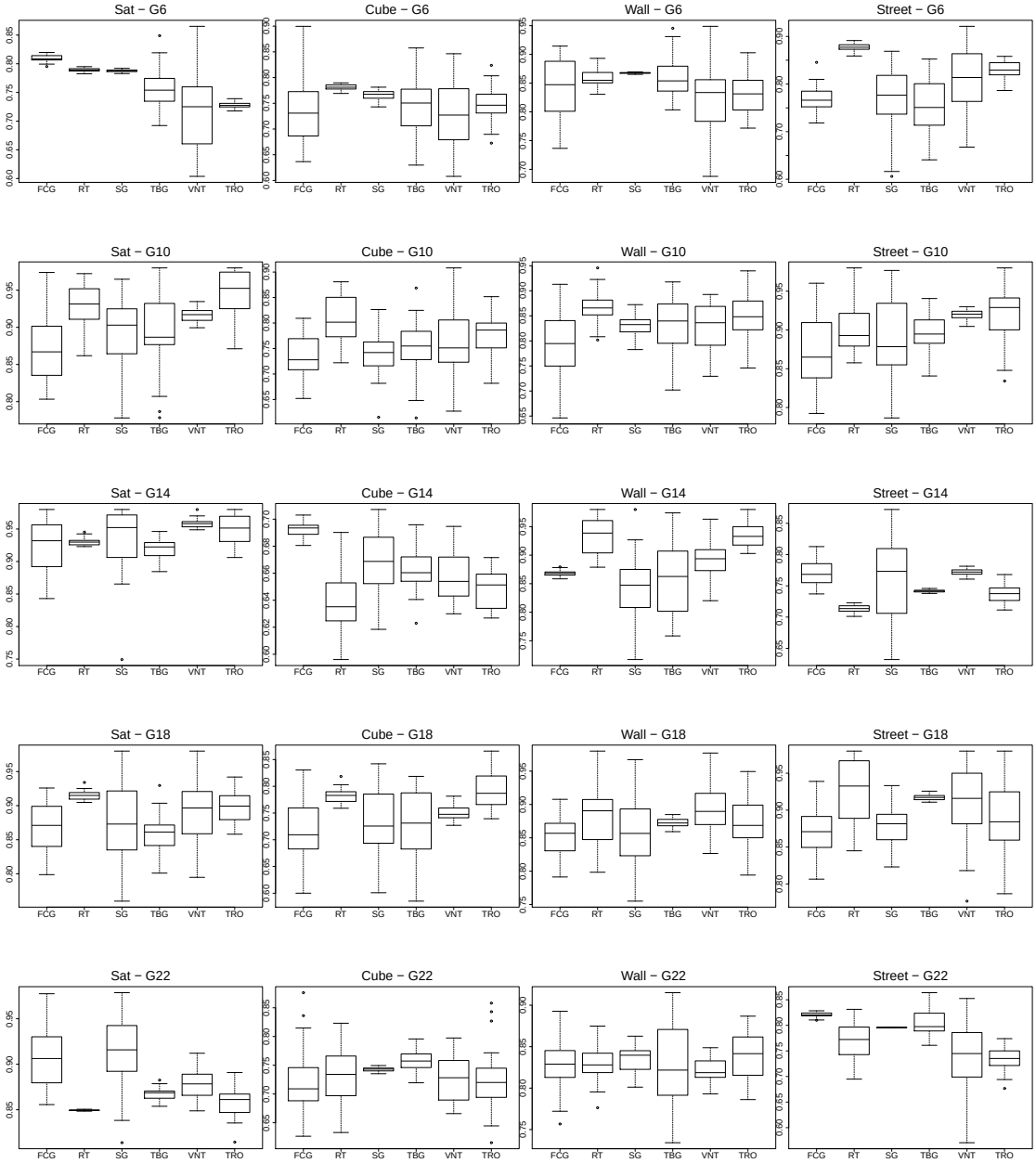


Figure 6.6: The box-plots drawn from the results of 30 independent runs of **BBPSO** with static topologies in noisy images corrupted by *Gaussian* noise.

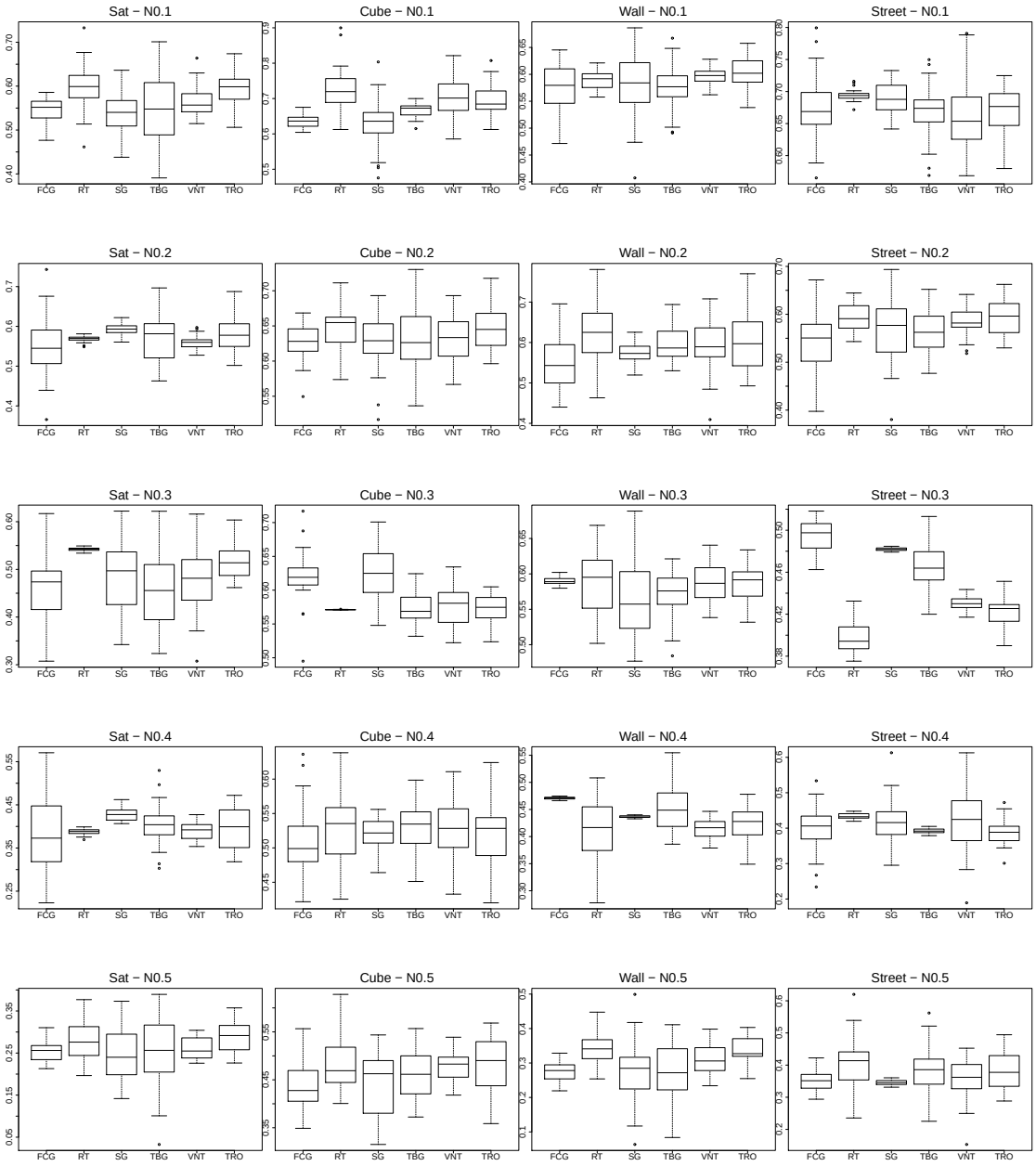


Figure 6.7: The box-plots drawn from the results of 30 independent runs of **BBPSO** with static topologies in noisy images corrupted by *impulse* noise.

Table 6.2: The performance of the **BBPSO**-based algorithm with different static topologies

Topology		FCG	RT	SG	TBG	VNT	TRO	Worse	Same	Better
FCG vs	Worse	–	19	3	4	12	16	54		
	Same	–	14	32	30	21	16		113	
	Better	–	7	5	6	7	8			33
RT vs	Worse	7	–	8	6	5	2	28		
	Same	14	–	18	21	26	33		112	
	Better	19	–	14	13	9	5			60
SG vs	Worse	5	14	–	2	7	11	39		
	Same	32	18	–	33	25	21		129	
	Better	3	8	–	5	8	8			32
TBG vs	Worse	6	13	5	–	6	10	40		
	Same	30	21	33	–	29	24		137	
	Better	4	6	2	–	5	6			23
VNT vs	Worse	7	9	8	5	–	3	32		
	Same	21	26	25	29	–	34		135	
	Better	12	5	7	6	–	3			33
TRO vs	Worse	8	5	8	6	3	–	30		
	Same	16	33	21	24	34	–		128	
	Better	16	2	11	10	3	–			42

ity of 0.1.

The statistical results of applying the FIPS-based edge detection algorithm with the static topologies are shown in Table 6.3. Unlike CanPSO and BBPSO whose accuracies become higher when the ring topology is chosen, the accuracy of FIPS is higher in most cases when the toroidal topology is chosen as a neighbourhood structure. Table 6.3 shows the number of the cases where FIPS with each topology is same/better/worse than the other topologies. The performance of FIPS with TRO (FIPS-TRO) is statistically better or equal in 99.5% of the cases (199 out of 200). The static topologies for the FIPS-based edge detection algorithm can be approximately ranked as TRO, {VNT, RT}, {FCG, TBG} and SG from highest accuracy to lowest accuracy.

6.6.4 Comparison of CanPSO, BBPSO and FIPS with the Best Topology

Figures 6.10 and 6.11 shows the box plots from the results of 30 independent runs of CanPSO-RT, BBPSO-RT and FIPS-TRO in images corrupted by Gaussian and impulse noise. Generally, FIPS with the toroidal topology has a higher accuracy than other methods except a few cases as can be seen in these figures.

Table 6.4 shows the statistical results of the comparison of FIPS-TRO with CanPSO-RT and BBPSO-RT. As we expected, the performance of the PSO-based edge detector is improved when the FIPS model is used to update the velocity of the particles and the chosen topology is TRO. The main reason is that the particles in FIPS are influenced by the average of its neighbours unlike two other versions and accordingly FIPS works better than the canonical PSO and Bare Bones PSO. The performance of FIPS with TRO is higher than or equal with that of CanPSO-RT and BBPSO-RT for 61 out of 80 cases (76%).

Our comparison also shows that the FIPS model with the toroidal topol-

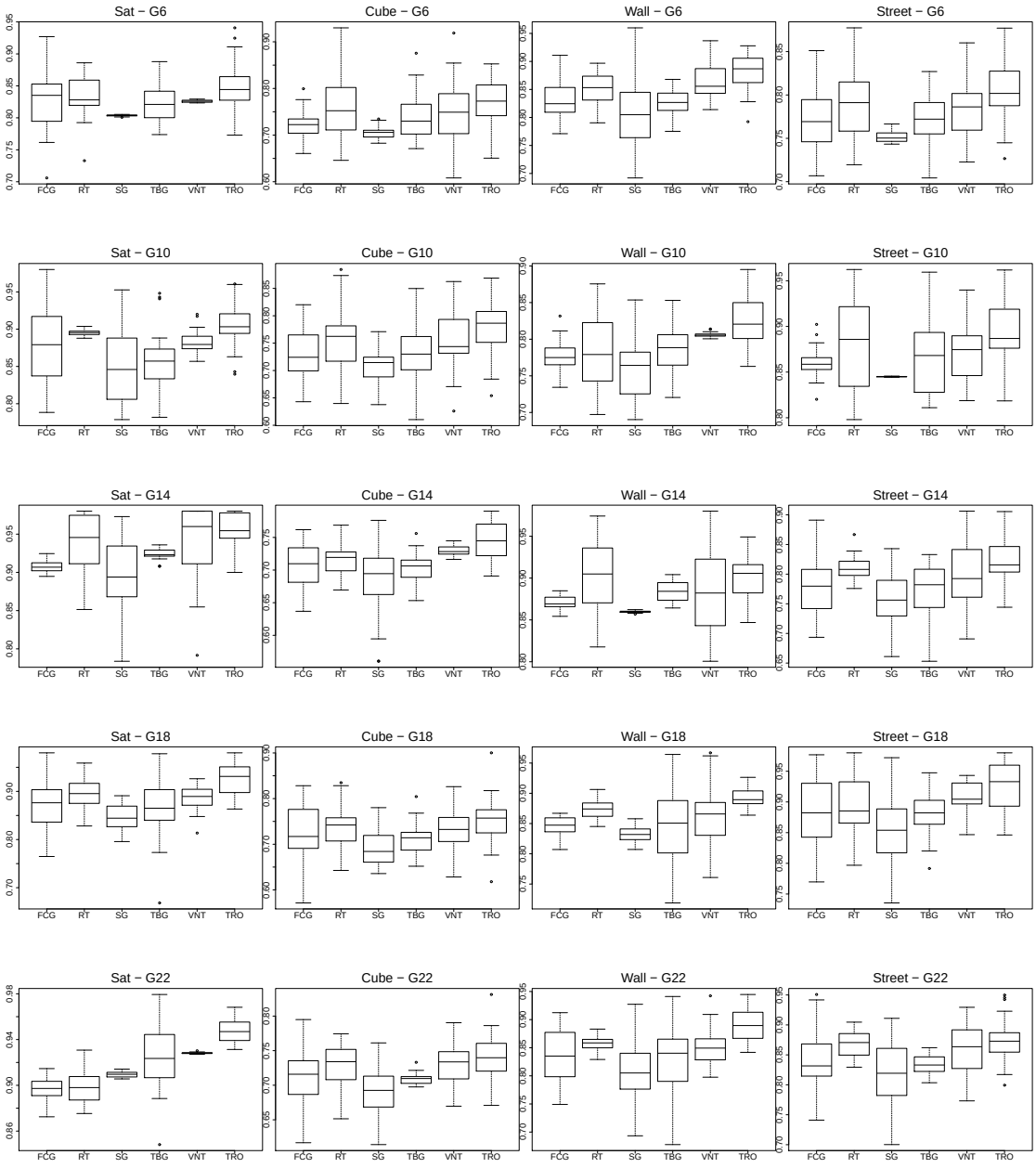


Figure 6.8: The box-plots drawn from the results of 30 independent runs of **FIPS** with static topologies in noisy images corrupted by *Gaussian* noise.

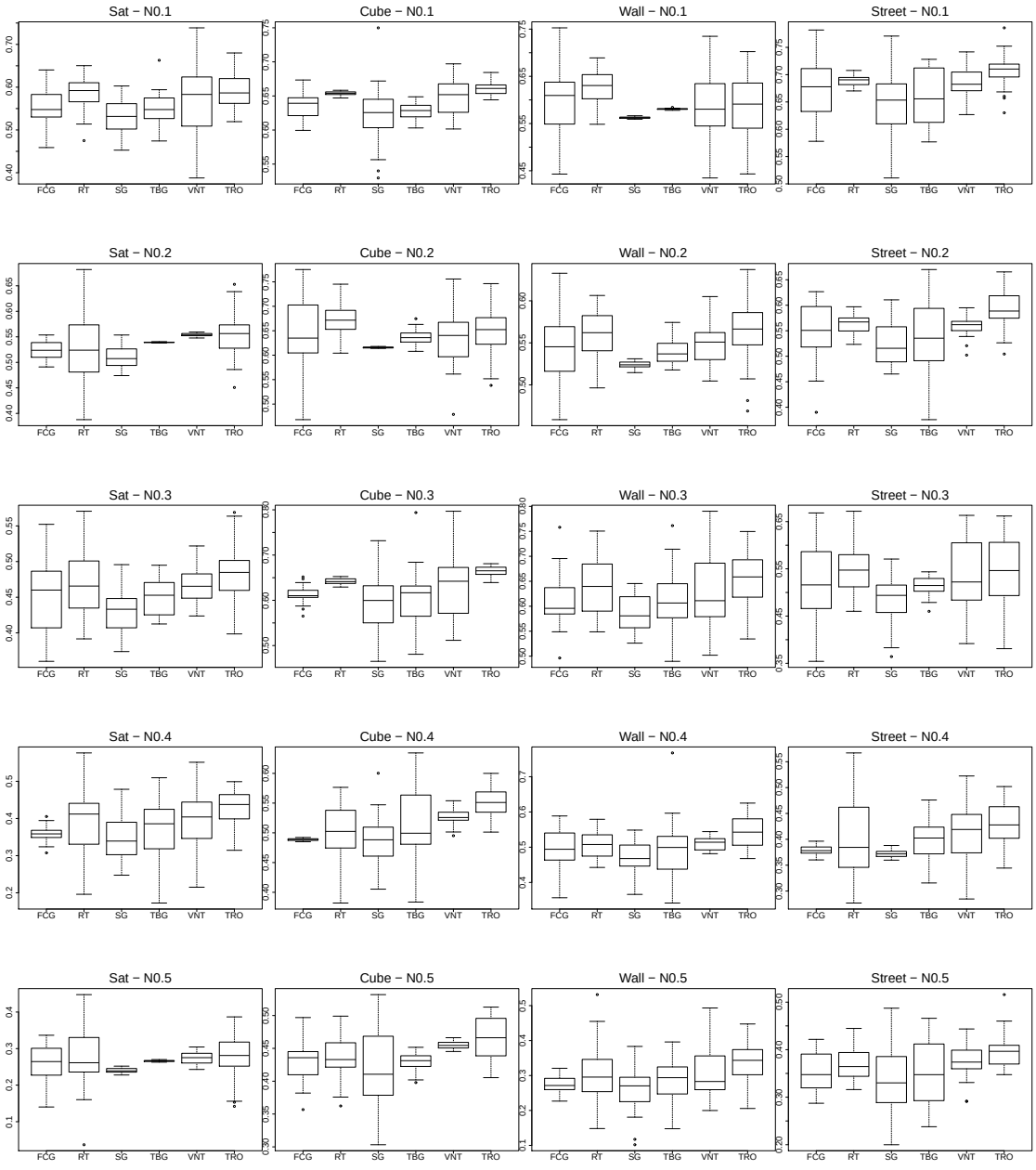


Figure 6.9: The box-plots drawn from the results of 30 independent runs of **FIPS** with static topologies in noisy images corrupted by *impulse* noise.

Table 6.3: The performance of the **FIPS**-based algorithm with different static topologies

Topology		FCG	RT	SG	TBG	VNT	TRO	Worse	Same	Better
	Worse	–	5	1	1	7	32	46		
FCG vs	Same	–	35	36	39	33	8		151	
	Better	–	0	3	0	0	0			3
	Worse	0	–	1	1	1	8	11		
RT vs	Same	35	–	8	29	37	31		140	
	Better	5	–	31	10	2	1			49
	Worse	3	31	–	5	36	39	114		
SG vs	Same	36	8	–	35	4	1		84	
	Better	1	1	–	0	0	0			2
	Worse	0	10	0	–	3	28	41		
TBG vs	Same	39	29	35	–	37	12		152	
	Better	1	1	5	–	0	0			7
	Worse	0	2	0	0	–	5	7		
VNT vs	Same	33	37	4	37	–	35		146	
	Better	7	1	36	3	–	0			47
	Worse	0	1	0	0	0	–	1		
TRO vs	Same	8	31	1	12	35	–		87	
	Better	32	8	39	28	5	–			112

Table 6.4: Comparison of FIPS-TRO with CanPSO-RT, BBPSO-RT

Topology		CanPSO-RT	BBPSO-RT	FIPS-TRO	Worse	Same	Better
	Worse	–	14	13	27		
CanPSO-RT vs	Same	–	6	17		23	
	Better	–	20	10			30
	Worse	20	–	16	36		
BBPSO-RT vs	Same	6	–	15		21	
	Better	14	–	9			23
	Worse	10	9	–	19		
FIPS-TRO vs	Same	17	15	–		32	
	Better	13	16	–			29

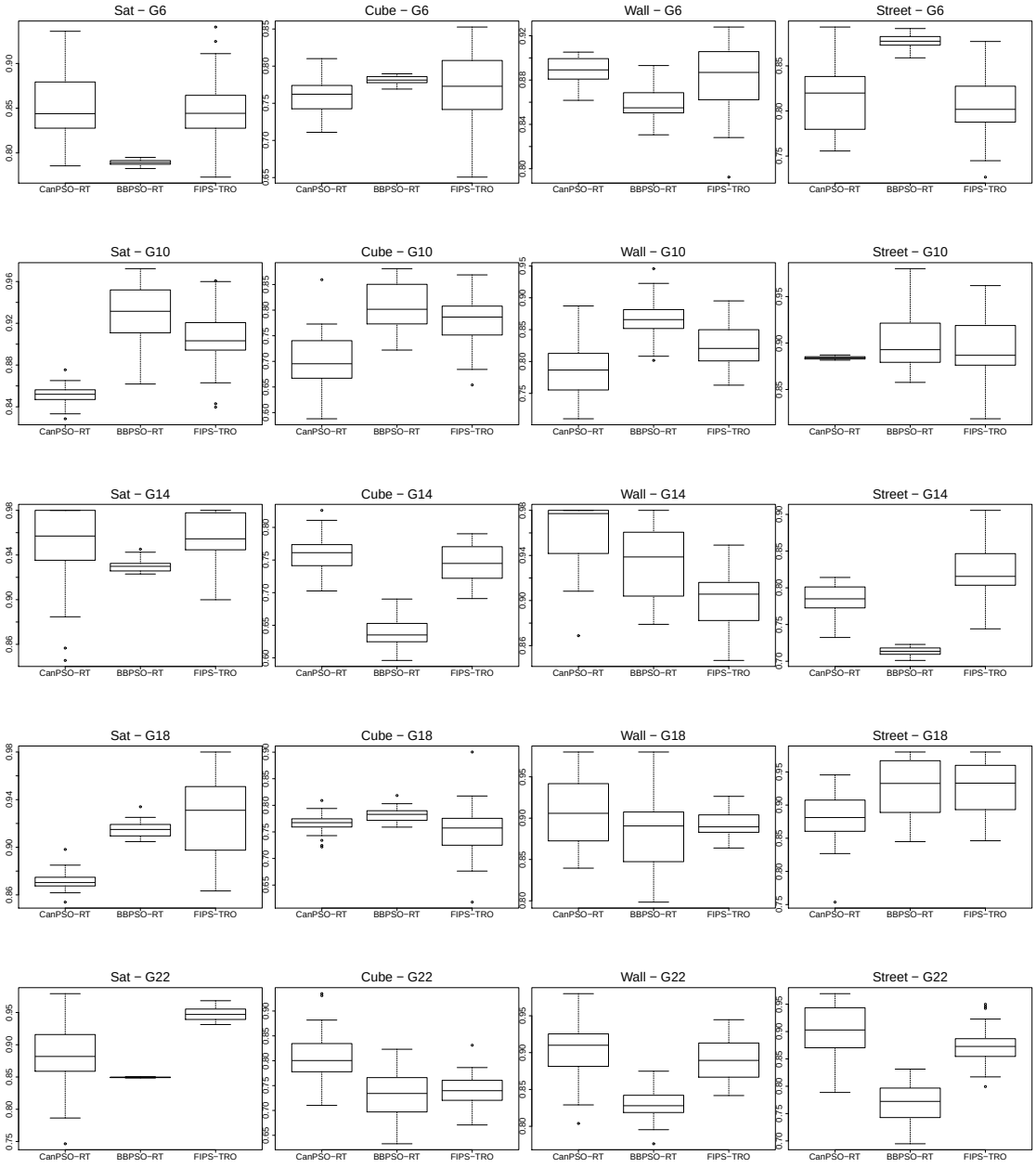


Figure 6.10: The box-plots drawn from the results of 30 independent runs of CanPSO, BBPSO and FIPS with static topologies in noisy images corrupted by *Gaussian* noise.

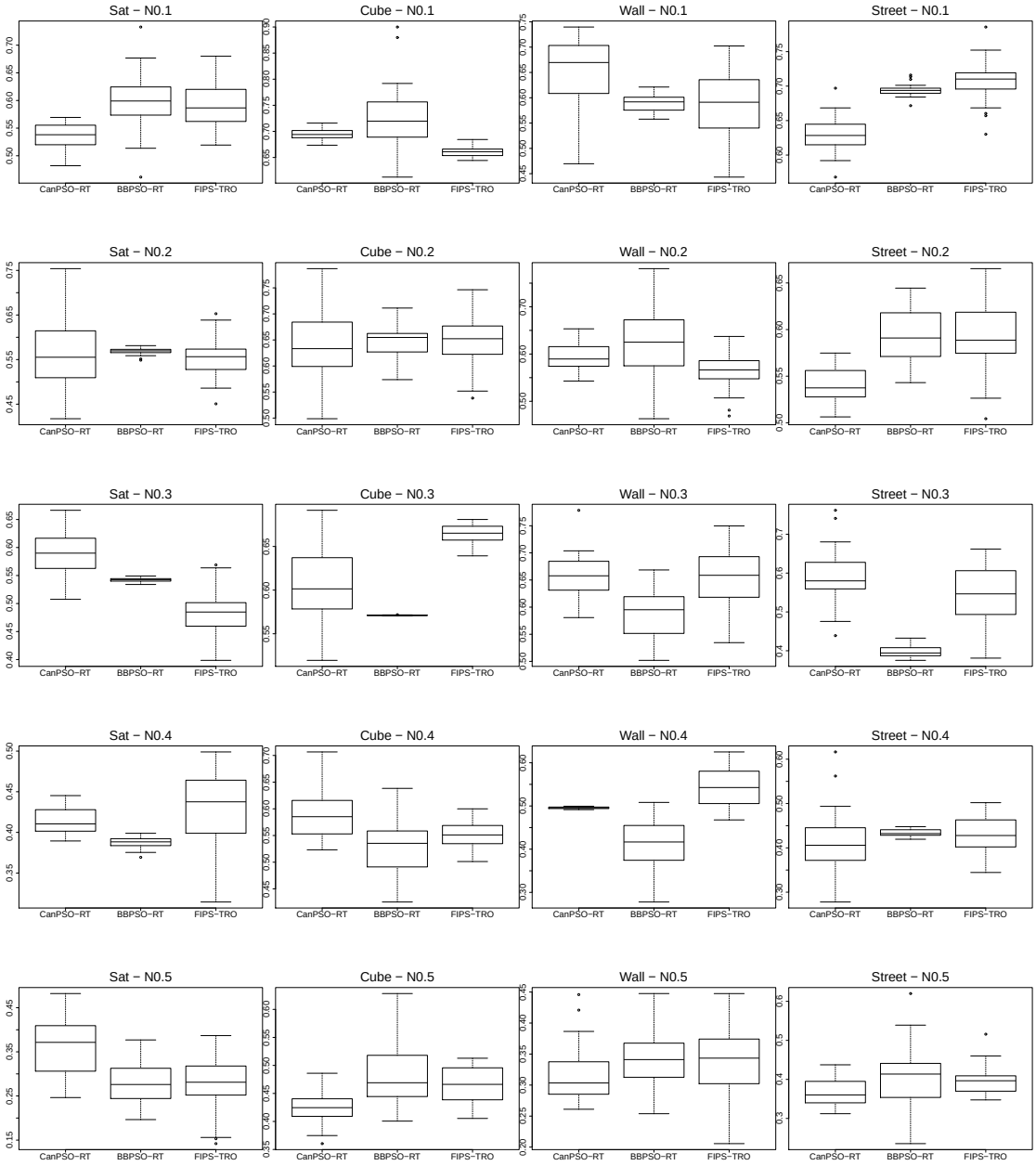


Figure 6.11: The box-plots drawn from the results of 30 independent runs of CanPSO, BBPSO and FIPS with static topologies in noisy images corrupted by *impulse* noise.

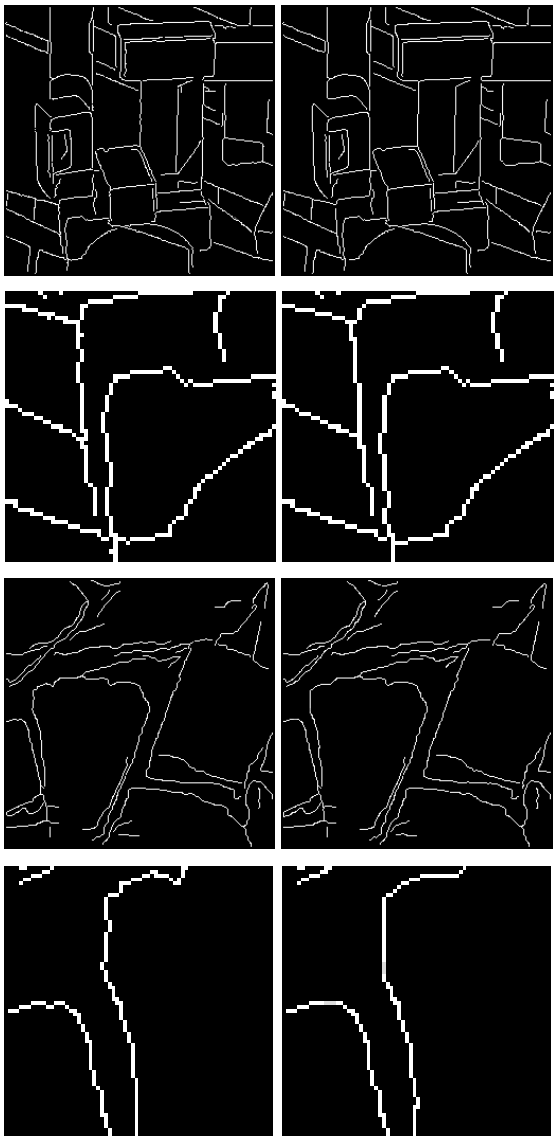
ogy increases the accuracy over our previous CanPSO edge detection algorithm equipped with FCG proposed in the previous chapter by approximately 5% on average.

6.7 Example of Detected Edge Maps

Figure 6.12 shows the edge maps resulting from applying CanPSO-FCG and FIPS-TRO based edge detectors to the Cube and Wall images. We enlarge them to be able to properly view their differences. As can be seen in Figure 6.12, FIPS-TRO localises the edges better than CanPSO-FCG which in the edges are displaced because of becoming trapped in local optima.

6.8 Summary

In this chapter, six different static topologies were implemented within the three versions of PSO and their effects were investigated in the PSO-based edge detector in noisy images. We arranged a statistical analysis to compare the effectiveness of each topology in the three versions of PSO and investigated their performance in images corrupted by Gaussian and impulse noise. Computational experiments showed that FIPS with the toroidal topology outperforms the canonical and bare bones PSO with various static topologies in most cases and is more robust to noise.



CanPSO-FCG FIPS-TRO

Figure 6.12: The resulting images from applying CanPSO-FCG and FIPS-TRO (the second and fourth rows show an enlarged version of a small region of the images in the first and the third rows respectively)

Chapter 7

A Spatial Random-Meaningful Neighbourhood Topology in PSO for Edge Detection in Noisy Images

As described in the previous chapter, there are two main categories of neighbourhood topologies in PSO: static and dynamic. In the previous chapter, the effects of using static topologies in the PSO-based edge detection algorithm were investigated. The results showed that the fully informed particle swarm with the toroidal topology performs better than other versions and other static topologies. In this chapter, different versions of PSO will be equipped with different dynamic topologies and their performance will be investigated, and then a novel dynamic topology will be introduced.

7.1 Introduction

The connections between particles, defined by a neighbourhood topology, is an important unique feature of PSO. A topology can control the speed of information flow among particles, which can affect the exploration and exploitation abilities of PSO. Therefore, a dynamic topology, which changes the connection structure between particles over the PSO iterations, can control the exploration and exploitation abilities of PSO [190]. In the fully connected topology, all particles are connected to each other and accordingly they share their acquired information among all others very quickly in one iteration. The canonical PSO with the fully connected topology has a strong exploitation ability but a weak exploration ability. Therefore, it could encounter the premature convergence problem and be likely trapped into local optima. As for the ring topology, each particle has an information flow with its two neighbouring particles and therefore every particle slowly shares its acquired information with other particles in the PSO population. The ring topology could improve the exploration ability of the PSO algorithm but it might reduce the the convergence speed of the algorithm [115]. Therefore, a few dynamic topologies have been proposed to effectively control the exploration and exploitation abilities of PSO over its iterations [190].

7.1.1 Chapter Goals

Since the performance of dynamic topologies has not been investigated in different versions of PSO, in this chapter, we aim to:

- investigate the performance of CanPSO, BBPSO and FIPS on the detection of edges in noisy images when they are equipped with different dynamic topologies (gradually increasing directed neighbourhood (GIDN) [190] and random dynamic topology [191]), and
- improve the performance of the PSO-based edge detector through

developing a novel dynamic topology which uses spatial-meaningful information.

The rest of this chapter is organised as follows. Section 7.2 describes two state-of-the art dynamic topologies, i.e., the random topology and gradually increasing directed neighbourhood topology followed by introducing a novel dynamic topology in Section 7.3. Sections 7.4 and 7.5 presents a discussion on experimental results followed by the summary in Section 7.6.

7.2 Dynamic Neighbourhood Topologies

Different methods have been used to develop various dynamic topologies such as restructuring a neighbourhood topology or dynamically changing the number of neighbours. In 1999, Suganthan [192] developed a dynamic neighbourhood by which exploration and exploitation abilities of PSO were controlled. In this topology, a specified number of the closest particles are chosen as the neighbours of each particle at each iteration. The number of connections is gradually increased over iterations, i.e., at the early iterations a small number of particles are selected as the neighbours of each particle while at the final iterations the entire population is chosen.

In this section, the dynamic topology proposed by Akat and Gazi [191] is reviewed followed by the gradually increasing directed neighbourhood topology proposed by Liu et al. [190].

7.2.1 Random Topology

Akat and Gazi [191] proposed a method to dynamically determine the particle neighbours in a random way. In this method, a threshold value, τ should be defined before the PSO algorithm is initialised. In the first step

of this topology, at each iteration, for every pair of particles (a, b), a uniform random variable, ϵ_{ab} between 0 and 1 is generated. A random number for (b, a) is chosen independently from the random number for (a, b). In this method, at each iteration, a total of $N(N-1)$ random numbers are generated where N is the size of the PSO population. In the next step, the neighbours of each particle a at iteration t are chosen as follows:

$$\mathcal{N}_a(t) = \{b | a \neq b \text{ and } \epsilon_{ab} < \tau\} \tag{7.1}$$

If $\epsilon_{ab} < \tau$, particle b is chosen as a neighbour of particle a at iteration t. This topology adds one more random feature to the PSO algorithm in addition to its other random features, such as the random uniform variables, r_1 and r_2 in Equation (2.43) which allow particles to move toward local or global best particles with a step of random length. This topology allows particles to share information with a random subset of neighbours which may be a different subset at another iteration. Therefore, the particles can search different directions in the search space which may increase the performance of the PSO algorithm.

7.2.2 Gradually Increasing Directed Neighbourhood (GIDN)

Liu et al. [190] improved the method proposed by Suganthan by developing a mechanism that could more effectively balance between the exploration and the exploitation ability of PSO than other topologies. This method was based on a gradually increasing directed neighbourhood (GIDN). Algorithm 7.1 shows the outline of PSO with GIDN.

In GIDN, the number of connections are gradually increased as calculated in Equation (7.2) and these directed connections are randomly selected. The number of the neighbours for particle P_i is calculated as:

$$|H_K(P_i)| = \left\lfloor N\left(\frac{t}{MaxIter_i}\right)^{\alpha} + \beta \right\rfloor \tag{7.2}$$

where $H_K(P_i)$ is the set of the neighbours of particle P_i at iteration K, $\lfloor . \rfloor$

Algorithm 7.1 PSO with GIDN [190]

Initialize PSO population
for all Particle P_i in Population **do**
    $H_{K=0}(P_i) \leftarrow \emptyset$
end for
repeat
    $K \leftarrow 1$
    for all Particle P_i in Population **do**
        Calculate desired $|H_K(P_i)|$ using Equation (7.2)
        if $|H_K(P_i)| > |H_{K-1}(P_i)|$ **then**
            Randomly Select $|H_K(P_i)| - |H_{K-1}(P_i)|$ distinct particles that do not have any connection with P_i and add them to its neighbourhood
        end if
        Evaluate fitness of P_i
        if $Fitness(P_i)$ is better than personal best **then**
            Update personal best position
        end if
    end for
    Assign the best particle in the population to the $leader$
    for all Particle P_i in population **do**
        Calculate particle velocity (6.2)
        Update particle position (2.42)
    end for
    Increase K
until stopping criteria attained

is the floor function, $MaxIter$ is the maximum number of iterations, α is a parameter to control the speed of information flow through increasing the neighbourhood size, t is the iteration number, and β is the initial neighbourhood size at the first iteration. In this model, each particle starts with β neighbours and randomly adds $|H_K(P_i)| - |H_{K-1}(P_i)|$ particles to its neighbourhood without taking their spatial information into account. To validate the GIDN model, Liu et al. [190] compared it with the dynamic topologies developed by Suganthan [192], Kennedy [193] and Mohais et al. [194]. The results were promising and the proposed method outperformed the other dynamic approaches.

7.3 Novel Dynamic Topology for a PSO-based Edge Detector

In this section, we introduce a novel dynamic topology based on GIDN with the outline of our novel algorithm in order to improve the accuracy of edge detection in noisy images.

7.3.1 New Spatial Random-Meaningful Topology (SRMT)

There are several versions of PSO that utilise spatial information to update the velocity and position of the particles. For example, in fitness distance ratio (FDR)-PSO [195] and fitness Euclidean ratio (FER)-PSO [196], spatial information is used to update the velocity of particles and guide them towards fitter points in their neighbourhood. Accordingly, they can more effectively locate the global optima. Both versions of PSO use the position of another particle in addition to the position of the personal best particle and the leader. That is to say that there are three different terms in their velocity update equations as follows:

$$\begin{aligned} V_{i,j}(t+1) &= \chi[V_{i,j}(t) + C_1 r_{1j}(X_{pbest_i,j} - X_{i,j}(t)) \\ &\quad + C_2 r_{2j}(X_{leader,j} - X_{i,j}(t)) + C_3 r_{3j}(X_{m,j} - X_{i,j}(t))] \end{aligned}$$

where m is a particle that the value of its j-th dimention, $X_{m,j}$ is the value of the j-th dimension of the a-th particle's personal best whose $FDR(a,i,j)$ in FDR-PSO or $FER(a,i)$ in FER-PSO is the largest among all particles. The fitness distance ratio for a maximisation problem is calculated as:

$$FDR(a,i,j) = \frac{Fitness(P_a) - Fitness(P_i)}{|X_{aj} - X_{ij}|}$$

and the Euclidean distance ratio is computed as:

$$FER(a,i) = \left(\frac{N}{Fitness(leader) - Fitness(P_w)}\right)\left(\frac{Fitness(P_a) - Fitness(P_i)}{\|\vec{X}_a - \vec{X}_i\|}\right)$$

where P_w is the worst particle in the PSO population and $\|.\|$ is the Euclidean distance of two particles in the search space.

To use spatial-meaningful information in order to more effectively select the neighbours of each particle in a random way and hopefully increase the performance of the PSO-based edge detection algoritim, we propose the spatial random-meaningful topology (SRMT). To meaningfully choose the neighbours of a particle (P), we first assign a neighbourhood score to each particle (P_n) in the PSO population at iteration K. We then select $|H_K(P)| - |H_{K-1}(P)|$ distinct particles which still do not currently have any connection with P and add them to its neighbourhood. Since we want the closest particles to a particle to have a higher score to be a neighbour of the particle, we define this score as:

$$NScore_K(P_n \; is \; a \; neighbour \; of \; P) = 1 - \frac{Dist_K(P,P_n)}{\sum_{P_i \notin H_{K-1}(P)} Dist_K(P,P_i)} \quad (7.3)$$

where $Dist_K(P,P_i)$ can be either the fitness or the Euclidean distance between particles P and P_i in either fitness or search space at iteration K.

Since the calculation of a Euclidean distance in the search space is more complex and time consuming than that of a fitness distance, we use the fitness distance to calculate the space between particles. So, $Dist_K(P, P_i) = |Fitness(P) - Fitness(P_i)|$. In Equation (7.3), if particle P_n is closer to P in the fitness space, its score of being a neighbour of particle P is higher and if their distance is larger, the score is lower. The particles with a lower score have a lower chance to be chosen as a neighbour of a particle and the particles with a higher score have a higher chance.

Algorithm 7.2 shows the outline of our PSO-based edge detector with SRMT. In this algorithm, the number of connections between the particles is increased at each iteration. Their new neighbours are randomly added based on their neighbourhood score as in Line 12 if the number of the connections in the previous iteration is less than that of the current iteration. Since the random selection of the new neighbours is based on spatial meaningful information, we expect that the novel method can guide the particles towards better areas in the search space and accordingly increase the accuracy of the PSO-based edge detection algorithm.

7.4 Experiment Design

7.4.1 Image Set

To compare the performance of CanPSO, BBPSO and FIPS with different dynamic topologies and validate the performance of the novel topology (SRMT) in these algorithms, we will apply these algorithms on a set of benchmark images from Chapter 3. The image set includes four real grey level images (Saturn, multi-cube, wall and street) as can be seen in Figure 3.2. To compare the performance of the PSO-based edge detector with different dynamic topologies, we use Pratt's Figure of Merit.

Algorithm 7.2 PSO-based edge detection algorithm with the spatial random-meaningful topology

for all pixel P on an image with a local edge magnitude larger than a predefined threshold **do**
  if P is unprocessed and not marked as an edge **then**
    Initialize PSO population randomly for pixel P
    for all Particle P_i in population **do**
      $H_{K=0}(P_i) \leftarrow \emptyset$
    end for
    repeat
      Increment K
      for all Particle P_i decoded as curve C **do**
        Calculate $|H_K(P_i)|$ using Equation (7.2)
        if $|H_K(P_i)| > |H_{K-1}(P_i)|$ **then**
          According to their scores, randomly select $|H_K(P_i)| - |H_{K-1}(P_i)|$ particles that do not have any connection with P_i and add them to its neighbours
        end if
        Evaluate $Fitness(C)$ and $PenFit(C)$
        if $PenFit(C)$ is better than personal best **then**
          Update personal best position for P_i
        end if
      end for
      for all Particle P_i decoded as curve C **do**
        For CanPSO or FIPS, Calculate particle velocity according to (2.43)
        After Updating particle position according (2.42) for CanPSO and FIPS, or (6.3) for BBPSO, apply update rule (4.23)
      end for
    until stopping criteria attained
    Select best particle and decode it as curve C^*
    if C^* is feasible **then**
      Mark all pixels on curve C^* as an edge
    else
      Mark all pixels within red rectangle as processed
    end if
  end if
end for

7.4.2 Parameter Values

We use the values $\chi = 0.7298$, $C_1 = C_2 = 2.05$ for CanPSO and FIPS parameters in Equations (6.2) and (6.4). The number of particles in the PSO is set at 50 and the maximum number of iterations at 200. These parameters are adjusted based on common settings [166][184]. The branching factor of the tree-based topology is set at 3 [197]. For GIDN, both α and β are set at 2 [190]. The parameters of our PSO-based edge detection algorithm, $2L + 1 = 21$, $SqrSize = 6$, and $HP = 0.5$ as recommended the previous chapters.

7.5 Results and Discussion

This section shows quantitative and qualitative results obtained from applying different dynamic topologies in the three well-known versions of PSO. In this section, we first present the quantitative results on using dynamic topologies in the three versions of PSO and then compare the best dynamic topology with other static topologies. We only present the overall results here, and the detailed results are available in the appendix.

7.5.1 Results on Dynamic Topologies

For an objective comparison, we first carried out our experiments with the three versions of PSO equipped with different dynamic topologies. Figures 7.1 and 7.2 show the box plots resulting from 30 independent runs of CanPSO, BBPSO and FIPS with the random topology, GIDN and SRMT in the images corrupted by Gaussian and impulse noise correspondingly. In these plots, the horizontal axis is the chosen topology and the vertical axis is the localisation accuracy (PFOM) of the PSO-based edge detector. Here, G6, G10, G14, G18 and G22 represent PSNR from 6dB to 22dB with step 4 for Gaussian noise and N0.1, N0.2, N0.3, N0.4 and N0.5 represent noise probability from 0.1 to 0.5 with step 0.1 for impulse noise. As can

be seen in Figure 7.1, the accuracy of CanPSO with the novel topology is equal or higher than other dynamic topologies in images corrupted by Gaussian noise in most cases. Its accuracy is lower than at least one of other topologies only in the Saturn image corrupted by G10, G18 and G22, in the cube image corrupted by G10, in the wall image corrupted by G6 and G10, and in the street image corrupted by G6, G10, G4 and G18.

As can be seen in Figure 7.2, the accuracy of the CanPSO-based edge detector with SRMT is lower than at least one of other topologies in the Saturn image with N0.1, in the cube image with N0.1, N0.3 and 0.5, in the wall image with N0.2, N0.4 and N0.5, and in the street image with N0.1, N0.2 and N0.5. In other cases, its accuracy is equal or higher than other topologies.

Table 7.1 summarises the results of applying CanPSO, BBPSO and FIPS with the Random, GIDN and SRMT dynamic topologies. For a fair comparison, we used Holm's method to adjust p-values resulted from the Student two paired T-test to compare the pairwise accuracy means. This table shows the number of the cases where each dynamic topology is same/better/worse than the other dynamic topologies. The results show that CanPSO with SRMT performs better than the other algorithms with the dynamic topologies. The comparison shows that the accuracy of CanPSO-SRMT is statistically higher or equal in 81.8% of the cases (262 out of 320). The dynamic topologies can be approximately ranked as CanPSO-SRMT, CanPSO-GIDN, FIPS-SRMT, CanPSO-Random, FIPS-GIDN, BBPSO-SRMT, FIPS-Random, BBPSO-GIDN and BBPSO-Random from highest accuracy to lowest accuracy. CanPSO with SRMT increases the accuracy over CanPSO-Random, CanPSO-GIDN, BBPSO-Random, BBPSO-GIDN, BBPSO-SRMT, FIPS-Random, FIPS-GIDN, FIPS-SRMT by approximatly 2.4%, 1.5%, 7.0%, 5.1%, 4.2%, 3.6%, 2.9% and 1.3%, on average respectively.

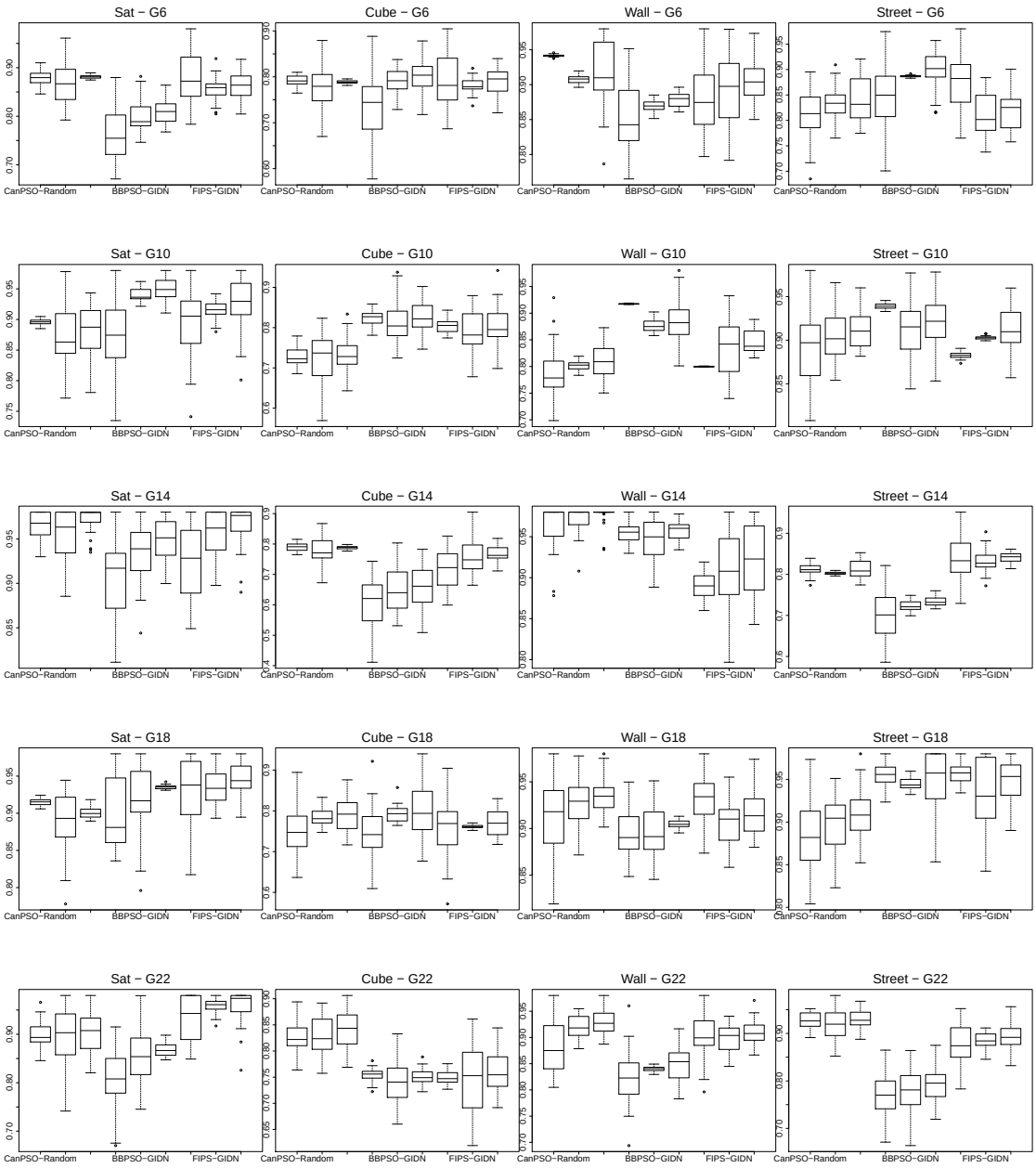


Figure 7.1: The box-plots drawn from the results of 30 independent runs of CanPSO, BBPSO, FIPS with dynamic topologies (CanPSO-Random, CanPSO-GIDN, CanPSO-SRMT, BBPSO-Random, BBPSO-GIDN, BBPSO-SRMT, FIPS-Random, FIPS-GIDN and FIPS-SRMT, from left to right correspondingly in each plot) in noisy images corrupted by *Gaussian* noise.

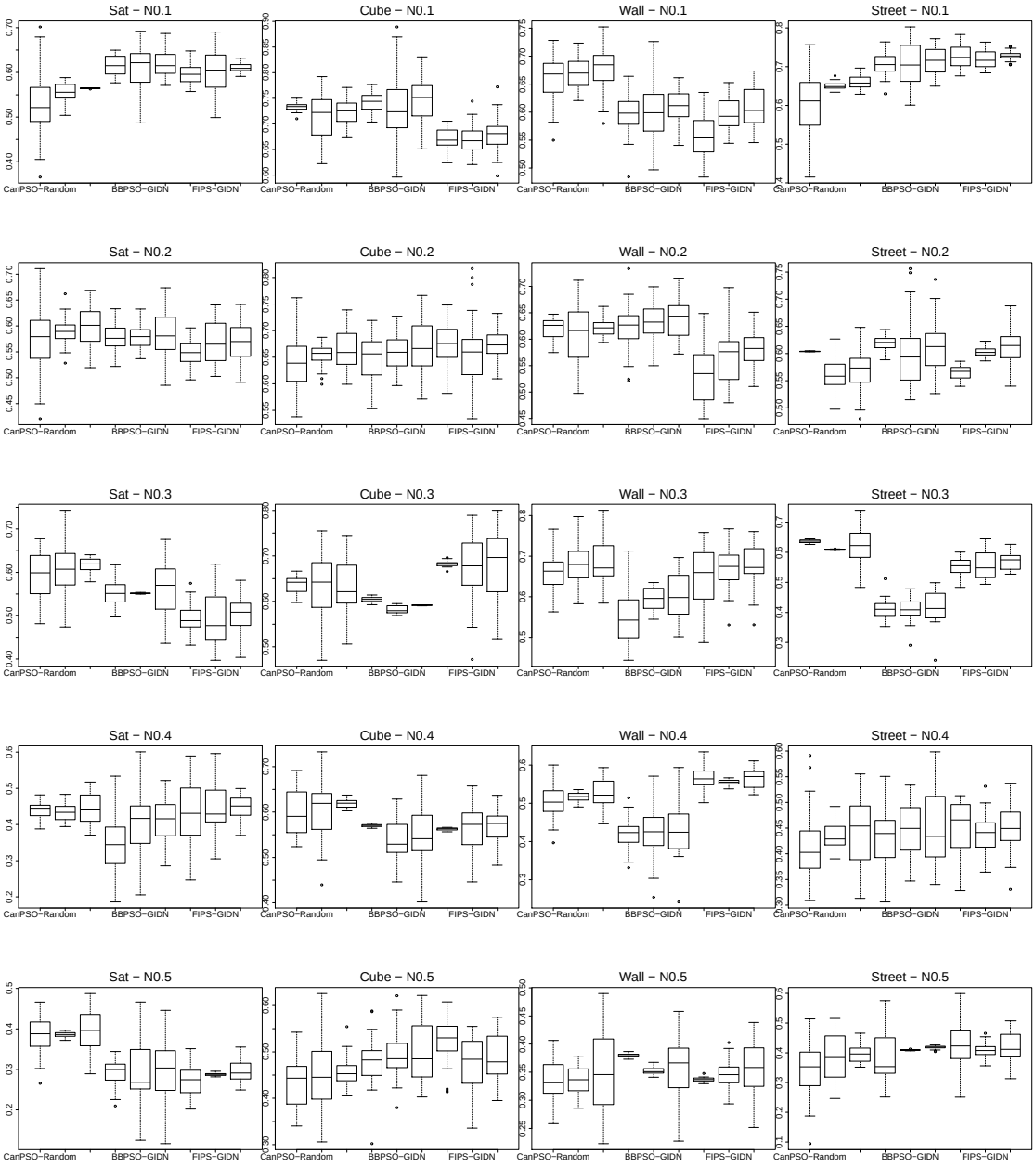


Figure 7.2: The box-plots drawn from the results of 30 independent runs of CanPSO, BBPSO and FIPS with dynamic topologies (CanPSO-Random, CanPSO-GIDN, CanPSO-SRMT, BBPSO-Random, BBPSO-GIDN, BBPSO-SRMT, FIPS-Random, FIPS-GIDN and FIPS-SRMT, from left to right correspondingly in each plot) in noisy images corrupted by *impulse* noise.

Table 7.1: The performance of the PSO-based algorithm with dynamic topologies

		CanPSO			BBPSO			FIPS			Total		
Topology		Rand	GIDN	SRMT	Rand	GIDN	SRMT	Rand	GIDN	SRMT	Worse	Same	Better
	Worse	–	2	10	7	9	11	11	9	13	72		
Random vs	Same	–	36	29	16	14	14	16	20	17		162	
	Better	–	2	1	17	17	15	13	11	10			86
	Worse	2	–	0	8	10	12	12	12	12	68		
GIDN vs	Same	36	–	40	12	11	11	13	17	18		158	
	Better	2	–	0	20	19	17	15	11	10			94
	Worse	1	0	–	8	8	10	9	10	12	58		
SRMT vs	Same	29	40	–	11	13	11	16	17	17		154	
	Better	10	0	–	21	19	19	15	13	11			108
	Worse	17	20	21	–	9	14	17	16	17	131		
Random vs	Same	16	12	11	–	29	25	13	16	17		139	
	Better	7	8	8	–	2	1	10	8	6			50
	Worse	17	19	19	2	–	0	11	10	12	90		
GIDN vs	Same	14	11	13	29	–	40	20	24	22		173	
	Better	9	10	9	0	–	9	6	6	0			49
	Worse	15	17	19	1	0	–	10	10	10	82		
SRMT vs	Same	14	11	11	25	40	–	19	21	24		165	
	Better	11	12	10	14	0	–	11	9	6			73
	Worse	13	15	15	10	9	11	–	6	8	87		
Random vs	Same	16	13	16	13	20	19	–	31	31		159	
	Better	11	12	9	17	11	10	–	3	1			74
	Worse	11	11	13	8	6	9	3	–		61		
GIDN vs	Same	20	17	17	16	24	21	31	–	40		186	
	Better	9	12	10	16	10	10	6	–	0			73
	Worse	10	10	11	6	6	6	1	0	–	50		
SRMT vs	Same	17	18	17	17	22	24	31	40	–		186	
	Better	13	12	12	17	12	10	8	0	–			84

Table 7.2: Comparison of CanPSO-SRMT with CanPSO-RT, BBPSO-RT and FIPS-TRO

Statistically	CanPSO-SRMT vs		
	CanPSO-RT	BBPSO-RT	FIPS-TRO
Worse	0	7	7
Same	39	12	13
Better	1	21	20

7.5.2 Comparison Between Static and Dynamic Topologies

Since CanPSO and BBPSO-based edge detectors work well with the ring topology and FIPS performs well with the toroidal topology, we compared the accuracy of CanPSO-RT, BBPSO-RT, FIPS-TRO and CanPSO-SRMT with each other. Table 7.2 shows the number of the cases where CanPSO-SRMT is same/better/worse than CanPSO-RT, BBPSO-RT and FIPS-TRO. The results show that CanPSO-SRMT performs better than CanPSO-RT, BBPSO-RT and FIPS-TRO. The comparison shows that the accuracy of CanPSO-SRMT is statistically higher or equal with CanPSO-RT in 100% of the cases (40 out of 40) and is higher or equal with BBPSO-RT and FIPS-TRO in 82.5% of the cases (33 out of 40). This implies that our novel dynamic PSO-based edge detection algorithm outperforms CanPSO and FIPS with FCG, RT, TRO, Random and GIDN.

Our comparison also shows that the novel dynamic topology increases the accuracy by 11.3% on average over our previous CanPSO edge detection algorithm equipped with FCG proposed in the previous chapter and by 1.5% over CanPSO equipped with GIDN as a dynamic topology.

7.5.3 Examples of Detected Edge Maps

Figure 7.3 shows several images resulting from CanPSO-FCG, FIPS-TRO, FIPS-GIDN and CanPSO-SRMT for a subjective comparison. The original

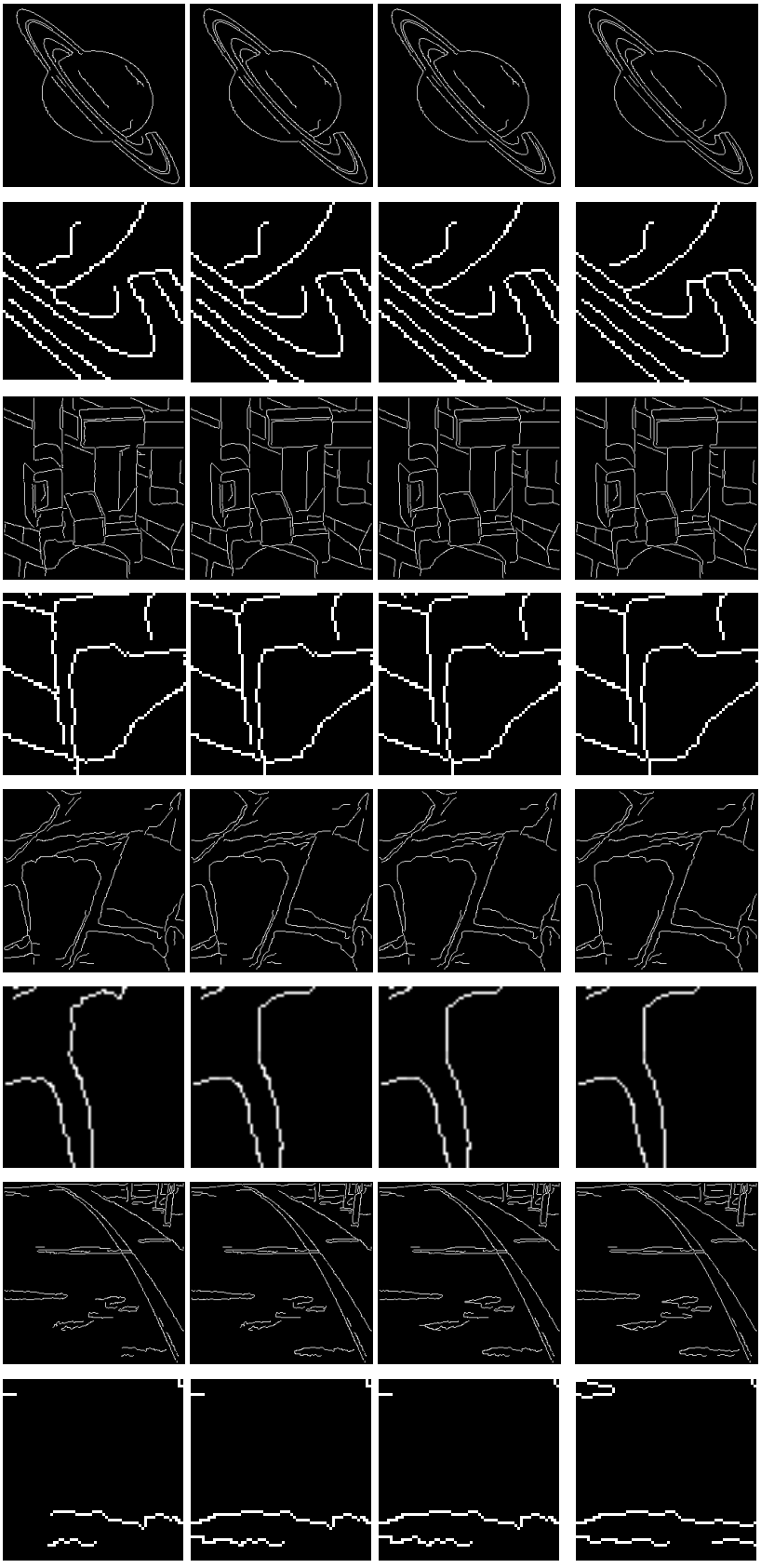


Figure 7.3: The resulting images from applying CapPSO-FCG, FIPS-TRO, FIPS-GIDN and CanPSO-SRMT.

Saturn, cube and wall and street images were first corrupted by Gaussian noise (peak signal to noise ratio (PSNR) is 16db) and then these algorithms are applied to these images. We enlarge them to be able to properly view their differences. The second, fourth, sixth and eighth rows show an enlarged version of a small region of the resulting images in the first, third, fifth and seventh rows respectively. For the Saturn image, the performance of FIPS-TRO, CanPSO-RT and FIPS-GIDN are similar and there are only a few displaced edges while CanPSO-SRMT performs better in terms of continuity and curvature of the edges (see the right middle of the enlarged version). In the edge maps of the cube and wall images, there are a few displaced edges. The edges detected by CanPSO-SRMT are also smoother than the edges detected by the other algorithms (see the centre of the enlarged version of the cube image and the middle bottom area of the wall image). For the street image, CanPSO-FCG does not work well on the areas in shadow on the road; the dynamic topologies perform better. These areas are very cluttered. CanPSO-SRMT can deal with the detection of edges in such areas better than the other two dynamic topologies (see the bottom of enlarged versions for the street image and the edges in the areas in shadow).

7.6 Summary

In this chapter, the performance of CanPSO, BBPSO and FIPS were investigated for detecting edges in noisy images when they utilise different well-known dynamic topologies. A novel spatial random-meaningful topology was also developed and utilised within the PSO-based edge detection algorithm. We arranged quantitative and qualitative experiments in order to investigate the effects of dynamic topologies in the accuracy of PSO-based edge detection algorithms. Experimental results indicated that the localisation accuracy of the PSO-based edge detector with the novel topology is higher than other static and dynamic topologies in most cases.

Chapter 8

Conclusions and Future Work

The major goal of this thesis was to develop a PSO-based approach to detecting edges with greater continuity in noisy images. The focus was on using the PSO algorithm to detect edges better than traditional vision approaches in noisy images through considering a large area and extracting the global structure of the edges. This was achieved by developing the PSO-based edge detectors without utilising any preprocessing or post processing techniques. The thesis demonstrated a set of novel ideas and approaches which use PSO to extract the global structure of the edges in order to compensate for the broken edges and exploring a large area in order to overcome noise.

8.1 Achieved Objectives

In this thesis, the following objectives have been achieved:

1. A novel PSO-based method was proposed to detect a continuous edge as a sequence of connected pixels while traditional edge detectors are usually applied to one pixel at a single run without considering the global structure of the edges. The proposed algorithm improved the localisation accuracy of edge detection in noisy images

in comparison to the traditional methods.

2. A constrained PSO-based algorithm was developed to increase the speed of the PSO-based algorithm. The performance of two well-known constraint handling methods were compared with each other in terms of efficiency and effectiveness. The results demonstrated that the PSO-based edge detector with a penalising method is more efficient than when it is equipped with a preservation method.

3. The thesis proposed a novel local thresholding technique which was used inside the PSO-based edge detector. The novel technique was based on an existing image binarisation method with a high performance in illuminated noisy images. Our results showed that the performance of the PSO-based edge detector equipped with the local thresholding technique is higher than when it uses a global thresholding technique.

4. The thesis investigated the effect of using different static topologies in three versions of PSO when they are applied to edge detection in noisy images. The results showed that different topologies have different effects on the accuracy of three versions of PSO which use different information sharing mechanisms among particles.

5. The thesis finally proposed a novel dynamic topology as an information sharing mechanism in order to increase the accuracy of the PSO-based edge detector. The novel topology was compared with several existing static and dynamic topologies. The results showed that the novel topology can share information among particles in a more effective way in comparison to other existing topologies and accordingly increase the accuracy of the PSO-based edge detector.

8.2 PSO for Edge Detection

The main goal of the PSO-based edge detector was to reduce the number of broken edges and to increase the localisation accuracy of edge detection in noisy images. An optimisation solution was proposed in Chapter 4 in order to detect edges in such images. The goal was successfully achieved by developing a new approach in the framework of PSO to optimise the solution. To overcome noise and reduce brokenness, an encoding scheme and a fitness function were developed based on the possibility score of a curve being fitted on an edge and the curvature cost of the curve with two constraints. Two different methods were proposed to handle the constraints. In addition to the position, orientation and magnitude of edges, the proposed methods utilised other characteristics of the edge pixels, such as the global structure of the edges, the smoothness of the edges, and the pixel intensities to detect the edges.

The efficiency and effectiveness of the new PSO-based algorithms were examined on two benchmark image sets corrupted by two different types of noise at different levels and compared with Canny as a Gaussian filter-based edge detector and RRO as a statistical-based edge detector based on PFOM as a measure of accuracy. The objective and subjective results showed that the new algorithms generally performed better than the Canny and RRO edge detectors in the images corrupted by Gaussian and impulse noise. Although the execution time of the new algorithms is longer than Canny and RRO, the edges detected by them are more connected and more accurate than Canny and RRO. Furthermore, the new algorithms do not use any extra preprocessing or post processing techniques. These results also showed that Canny could perform reasonably well for some images with a low level of noise; however, its performance is worse than RRO in most cases. The results also demonstrated that the localisation accuracy of edge detection can be increased while a large area is considered in edge detection to overcome noise and mark a pixel of an image as an edge. This

was obtained in the PSO-based edge detectors by considering more edge patterns in comparison to traditional edge detection algorithms.

8.3 A Local Thresholding Technique for the PSO-based Edge Detector

The main goal of Chapter 5 was the development of a local thresholding technique for the PSO-based algorithms to overcome the problem of detection of edges in noisy images with illuminated area. Since the edges in the illuminated areas are weaker than the edges in other areas, most edge detection algorithms cannot perform well in the illuminated areas and the resulting edges are usually broken after applying thresholding techniques.

To address the problem of detection of edges in the illuminated areas, the threshold value estimated by thresholding techniques should be adjusted according to the edge magnitudes of the pixels in these areas. We borrowed the main idea of the new local thresholding technique from the Sauvola-Pietkinen method as a way of binarisation of illuminated document images and adapting this method to the PSO-based algorithms. In this method, a threshold value for each pixel is estimated based on the mean and variance of local edge magnitudes of its neighbours. The local threshold value for each pixel is higher when there are many strong edges in its neighbourhood and it is lower when there are many weak edges in its neighbourhood.

The experiments in Chapter 5 showed that the performance of the PSO-based algorithm equipped by the proposed local thresholding technique is better than the PSO-based algorithms utilising Otsu's method in noisy images with illuminated area. The results showed that the local thresholding technique reduces broken edges and accordingly increases the localisation accuracy of edge detection in the noisy illuminated image.

8.4 Static Topologies for the PSO-based Edge Detector

The main goal of Chapter 6 was the investigation of the performance of different static topologies in three versions of PSO, namely CanPSO, BBPSO and FIPS to detect edges in noisy images. These versions of PSO use different velocity and position equations to compute the new position of each particle in the PSO population. The particles in CanPSO are influenced by their current velocities, their personal best positions and the position of their leaders based on a chosen topology. BBPSO is a parameter free version which omits the influence of velocity of each particle and uses a simple Gaussian distribution to compute the new position of each particle. In this version of PSO, each particle is affected by its personal best position and its leaders. In FIPS, the velocity of each particle is affected by the average between the positions of its neighbours. The position of each particle in this version is influenced by its current velocity and the positions of its neighbours based on a chosen topology. In this chapter, the effects of these features of CanPSO, BBPSO and FIPS with different neighbourhood topologies were investigated on their performance to detect edges in noisy images.

The results showed that different topologies have different effects on the accuracy of the various versions of PSO. The comparison of CanPSO and BBPSO with different static topologies demonstrated that their behaviours are very similar with different topologies. CanPSO and BBPSO perform better when they use the ring topology as a neighbourhood topology and worse when they are equipped by the fully connected topology in comparison to the other static topologies. The static topologies can be approximately ranked as RT, TRO, VNT, TBG, SG and FCG. The results showed that FIPS behaves differently when it is equipped with different static topologies. Unlike CanPSO and BBPSO, the FIPS-based edge detection algorithm with the toroidal topology has a higher accuracy than

the other topologies. For FIPS, the static topologies can be approximately ranked as TRO, VNT, RT, FCG and SG. The results showed that FIPS with TRO has the highest accuracy among three versions of PSO with different static topologies.

8.5 A Novel Dynamic Spatial Random-Meaningful Neighbourhood Topology

In Chapter 7, the main goal was to investigate the performance of CanPSO, BBPSO and FIPS on the detection of edges in noisy images when they use different dynamic neighbourhood topologies, namely, the random and the gradually increasing dynamic topologies. Since a topology can change the speed of information flow among particles, it can affect the exploration and exploitation abilities of PSO. Dynamic topologies change the connection structure between particles over the PSO iterations. Accordingly, they can control the exploration and exploitation abilities of PSO and help it to likely avoid being trapped into a local optima. In this chapter, the influence of dynamic topologies on the performance of the PSO-based edge detector were investigated and a novel dynamic topology was also proposed to improve its performance in the detection of edges in noisy images.

The results showed that different dynamic topologies have different effects on the accuracy of the various versions of PSO. The experiments showed that CanPSO, BBPSO and FIPS perform better when they use the novel topology as a neighbourhood topology in comparison with the other dynamic topologies and they perform worse when they use the random topology. The dynamic topologies can be approximately ranked as CanPSO-SRMT, CanPSO-GIDN, FIPS-SRMT, CanPSO-Random, FIPS-GIDN, BBPSO-SRMT, FIPS-Random, BBPSO-GIDN and BBPSO-Random from highest accuracy to lowest accuracy.

In Chapter 7, a novel dynamic topology named SRMT was developed

which was an improved version of a gradually increasing directed neighbourhood (GIDN). Spatial meaningful information are used to compute the neighbourhood probability of each particle to be a neighbour of another particle. This probability was used to randomly choose the neighbours of each particle at each iteration. This was the reason that the novel topology was called the spatial random-meaningful topology (SRMT). Our experiments showed that using SRMT improves the accuracy of the three versions of PSO-based edge detection algorithms in comparison to other static and dynamic topologies. Although the accuracy of CanPSO, BBPSO and FIPS-based edge detection algorithms are improved when they utilise a dynamic topology, that of FIPS-TRO is comparable with CanPSO-SRMT in most cases and FIPS-TRO is computationally less expensive than CanPSO with SRMT as a dynamic topology.

8.6 Limitations

This section briefly provides some important limitations of the PSO-based edge detector.

8.6.1 Edge Detection in Coloured Images

Nowadays, coloured images are widely used in many applications, such as remote sensing and medical images. Accordingly, edge detection in coloured images is very important. Since the main goal of this thesis was to develop a PSO-based edge detector for gray level images, we focused on detecting edges in such images and did not consider multidimensional data, such as color and multispectral images. However, the PSO-based edge detector could easily be applied to coloured images by use of vector order statistics.

8.6.2 Low Speed of the PSO-based Edge Detector

The main disadvantage of the PSO-based edge detector is its low speed in comparison to traditional edge detection algorithms. Although the edges detected by the PSO-based edge detector are more connected and more accurate than the edges detected by the tradition edge detectors in noisy images, its execution time is much longer. The execution time of the PSO-based edge detector is usually between 40 and 50 seconds depending on the noise level in the images used in the experiments while the tradition methods detect edges in a few seconds. Although the speed of the PSO-based edge detection algorithm can be increased by the use of concurrency or parallel programming, its efficiency can be also improved by decreasing the number of times which PSO is applied to an image. For example, in the proposed algorithm, we used a very simple condition which should be satisfied before PSO is applied to a pixel. In Line 1 of Algorithms 4.1 and 4.2, PSO is applied to the pixels which have a larger local edge magnitude than the estimated local threshold value. By improving this condition and analysing the local edge magnitudes of the neighbours of each pixel of an image, the execution time of the algorithm could be improved.

8.7 Future Research Directions

This section provides some possible future research directions in three aspects, developing a fitness function and an encoding scheme for coloured images, constraint handling in the PSO-based edge detector, investigating the performance of weighted neighbourhood topologies and applying the new methods to more images to further investigate the performance of the new methods.

8.7.1 PSO-based Edge Detector for Coloured Images

One possible extension of this research is to develop a fitness function and an encoding scheme for the PSO-based edge detector to apply to coloured images. Since coloured images contain more information than gray level images, more edge information is expected from colour edge detection. It has been shown that 90% of the edges are about the same in gray level and in coloured images [198]. Accordingly, 10% of the edges may not be detected by edge detection algorithms in gray level images. These algorithms have to adapt to multidimensional data, such as multispectral and coloured images. The PSO-based edge detection algorithm cannot be currently applied to multispectral and coloured images and consequently a fitness function and an encoding scheme are required to adapt the proposed algorithm to such images. A new fitness function and an encoding scheme may be developed by borrowing some ideas from vector order statistics and their applications in traditional colour edge detectors.

8.7.2 Handling the Constraints in the Proposed Optimisation Solution

The constraint handling methods in PSO are categorised into four main groups, namely, preservation, penalising, repairing and transferring. In Chapter 4, the performance of a preservation and a penalising method were investigated and shown that the penalising method has almost the same accuracy as the preservation method but its speed is higher. Recently, a few repairing and transferring methods have been proposed for PSO to handle constraints [160]. It has been shown that the efficiency and effectiveness of these methods are better than preservation and penalising methods but they are very problem dependent and should be customised for a particular problem. As an extension of this research, the performance of the repairing and transferring methods can be investigated in the PSO-based edge detector. For example, when a curve crosses itself, the particle

representing the curve can be repaired such that it does not crosses itself. This repairing technique may improve the efficiency and effectiveness of the PSO-based edge detector in terms of the execution time and the localisation accuracy.

8.7.3 A Topology with Weighted Connections

In Chapters 6 and 7, the effects of static dynamic topologies were investigated in the PSO-based edge detector. In all topologies, the connections between particles were not weighted while assigning values as weights to the connections would give a degree of importance to the neighbours of the particles based on their positions and improve the performance of the PSO-based edge detector. The weights can be changed over time depending on how successful the particles are or have been in the past, e.g., a particle may increase the weight of a connection with a neighbour who has found a better solution or may decrease it if the neighbour has not found a better one. A dynamic change in the weights as a part of learning/evolution may allow the particles to improve the knowledge which they obtain from their neighbours and accordingly improve the accuracy of the PSO-based edge detectors.

8.7.4 Further Investigation with More Images

This thesis only used a small number of images from the selected image sets as the test bed. A future work is to apply the new methods developed in this thesis to more images to further investigate and confirm the performance of these methods. The source code is available to generate results for all the remaining images from the chosen standard image sets to demonstrate the extend to which the proposed method significantly exceeds the ground truth results on these image sets.

Bibliography

[1] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. Prentice Hall, 2007.

[2] S. E. Umbaugh, *Computer Imaging: Digital Image Analysis and Processing*. CRC Press, 2005.

[3] D.-S. Lu and C.-C. Chen, "Edge detection improvement by ant colony optimization," *Pattern Recognition Letters*, vol. 29, no. 4, pp. 416–425, 2008.

[4] G. Srivastava, R. Verma, R. Mahrishi, and S. Rajesh, "A novel wavelet edge detection algorithm for noisy images," in *Proceedings of the International Conference on Ultra Modern Telecommunications Workshops (ICUMT)*, 2009, pp. 1–8.

[5] W. Cao, R. Che, and D. Ye, "An illumination-independent edge detection and fuzzy enhancement algorithm based on wavelet transform for non-uniform weak illumination images," *Pattern Recognition Letters*, vol. 29, no. 3, pp. 192–199, 2008.

[6] L. R. Liang and C. G. Looney, "Competitive fuzzy edge detection," *Applied Soft Computing*, vol. 3, no. 2, pp. 123–137, 2003.

[7] Y.-P. Wong, V. C.-M. Soh, K.-W. Ban, and Y.-T. Bau, "Improved Canny edges using ant colony optimization," in *CGIV '08: Proceedings of the 2008 Fifth International Conference on Computer Graphics,*

Imaging and Visualisation. Washington, DC, USA: IEEE Computer Society, 2008, pp. 197–202.

[8] A. Baştürk and E. Günay, "Efficient edge detection in digital images using a cellular neural network optimized by differential evolution algorithm," *Expert Systems with Applications*, vol. 36, no. 2, pp. 2645–2650, 2009.

[9] S. M. Bhandarkar, Y. Zhang, and W. D. Potter, "An edge detection technique using genetic algorithm-based optimization," *Pattern Recognition*, vol. 27, no. 9, pp. 1159–1180, 1994.

[10] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679–698, 1986.

[11] D. H. Lim, "Robust edge detection in noisy images," *Computational Statistics and Data Analysis*, vol. 50, no. 3, pp. 803–812, 2006.

[12] A. A. Abarghouei, "Advances of soft computing methods in edge detection," *International Journal of Advances in Soft Computing and Its Applications*, vol. 1, no. 2, pp. 162–203, 2010.

[13] P. Azad, T. Gockel, and R. Dillmann, *Computer Vision: Principles and Practice*. Elektor International Media BV, 2008.

[14] W. Pratt, *Digital Image Processing: PIKS Scientific Inside*. Wiley Interscience, 2007.

[15] A. Jevtic, I. Melgar, and D. Andina, "Ant based edge linking algorithm," in *Proceedings of the 35th Annual Conference of IEEE Industrial Electronics (IECON)*, 2009, pp. 3353–3358.

[16] Q. Zhu, "Efficient evaluations of edge connectivity and width uniformity," *Image and Vision Computing*, vol. 14, no. 1, pp. 21–34, 1996.

[17] H. Pan, L. Wang, and B. Liu, "Particle swarm optimization for function optimization in noisy environment," *Applied Mathematics and Computation*, vol. 181, no. 2, pp. 908–919, 2006.

[18] K. E. Parsopoulos and M. N. Vrahatis, "Recent approaches to global optimization problems through particle swarm optimization," *Natural Computing*, vol. 1, pp. 235–306, 2002.

[19] Y. Chen, H. Wang, and J. Su, "Particle swarm optimization for image noise cancellation," in *Proceedings of the First International Conference on Innovative Computing, Information and Control*. Washington DC, USA: IEEE Computer Society, 2006, pp. 587–590.

[20] J. Zhao and Z. Li, "Particle filter based on particle swarm optimization resampling for vision tracking," *Expert Systems with Applications*, vol. 37, no. 12, pp. 8910–8914, 2010.

[21] A. Nakib, H. Oulhadj, and P. Siarry, "Fractional differentiation and non-pareto multiobjective optimization for image thresholding," *Engineering Applications of Artificial Intelligence*, vol. 22, no. 2, pp. 236–249, 2009.

[22] M. Setayesh, M. Zhang, and M. Johnston, "A new homogeneity-based approach to edge detection using PSO," in *Proceedings of the 24th International Conference on Image and Vision Computing, New Zealand*. IEEE Press, 2009, pp. 231–236.

[23] M. Setayesh, M. Johnston, and M. Zhang, "Edge and corner extraction using particle swarm optimisation," in *AI 2010: Advances in Artificial Intelligence*, ser. Lecture Notes in Computer Science, J. Li, Ed. Springer Berlin / Heidelberg, 2011, vol. 6464, pp. 323–333.

[24] M. Setayesh, M. Zhang, and M. Johnston, "Improving edge detection using particle swarm optimisation," in *Proceedings of the 25th*

International Conference on Image and Vision Computing, New Zealand. IEEE Press, 2010, pp. 1–8.

[25] ——, "Edge detection using constrained discrete particle swarm optimisation in noisy images," in *Proceedings of the 2011 IEEE Congress on Evolutionary Computation*. IEEE Press, 2011, pp. 246–253.

[26] ——, "Detection of continuous, smooth and thin edges in noisy images using constrained particle swarm optimisation," in *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, 2011, pp. 45–52.

[27] ——, "A novel local thresholding technique in PSO for detecting continuous edges in noisy images," in *Proceedings of the 26th International Conference on Image and Vision Computing, New Zealand*. IEEE Press, 2011, pp. 333–339.

[28] ——, "Investigating particle swarm optimisation topologies for edge detection in noisy images," in *Proceedings of the 24th Australasian Joint Conference on Artificial Intelligence*, ser. Lecture Notes in Computer Science, vol. 7106. Springer, 2011, pp. 609–618.

[29] ——, "Effects of static and dynamic topologies in particle swarm optimisation for edge detection in noisy images," in *Proceedings of the 2012 IEEE Congress on Evolutionary Computation*. IEEE Press, 2012, pp. 8–15.

[30] ——, "A spatial random-meaningful neighbourhood topology in pso for edge detection in noisy images," in *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation*, 2012, pp. 1403–1404.

[31] S. Cagnon, E. Lutto, and G. Olagu, *Genetic and Evolutionary Computation for Image Processing and Analysis*. Hindawi Publishing Corporation, 2008.

[32] M. Seul, *Practical Algorithms for Image Analysis: Description, Examples, and Code*. Cambridge University Press, 2000.

[33] N. Kilic, P. Gorgel, O. Ucan, and A. Kala, "Multifont ottoman character recognition using support vector machine," in *Proceedings of the Third International Symposium on Communications, Control and Signal Processing*, 2008, pp. 328 –333.

[34] T. Shuang-tong and L. Wen-ju, "Number and letter character recognition of vehicle license plate based on edge hausdorff distance," in *Proceedings of the Sixth International Conference on Parallel and Distributed Computing, Applications and Technologies*, 2005, pp. 850–852.

[35] M. Pfister, S. Behnke, and R. Rojas, "Recognition of handwritten zip codes in a real-world non-standard-letter sorting system," *Applied Intelligence*, vol. 12, no. 1-2, pp. 95–115, 2000.

[36] D. Mery and D. Filbert, "Automated flaw detection in aluminum castings based on the tracking of potential defects in a radioscopic image sequence," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 6, pp. 890–901, 2002.

[37] V. Sherrow, *Medical Imaging*. Marshal Cavendish Press, 2006.

[38] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. N. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, M. Gittleman, S. Harbaugh, M. Hebert, T. M. Howard, S. Kolski, A. Kelly, M. Likhachev, M. McNaughton, N. Miller, K. Peterson, B. Pilnick, R. Rajkumar, P. Rybski, B. Salesky, Y.-W. Seo, S. Singh, J. Snider, A. Stentz, W. R. Whittaker, Z. Wolkowicki, J. Ziglar, H. Bae, T. Brown, D. Demitrish, B. Litkouhi, J. Nickolaou, V. Sadekar, W. Zhang, J. Struble, M. Taylor, M. Darms, and D. Ferguson, "Autonomous driving in urban environments: Boss and the urban challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 425–466, 2008.

[39] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, D. Johnston, S. Klumpp, D. Langer, A. Levandowski, J. Levinson, J. Marcil, D. Orenstein, J. Paefgen, I. Penny, A. Petrovskaya, M. Pflueger, G. Stanek, D. Stavens, A. Vogt, and S. Thrun, "Junior: The stanford entry in the urban challenge," *Journal of Field Robotics*, vol. 25, no. 9, pp. 569–597, 2008.

[40] C. Jian-nan, Z. Chuang, Z. Han, L. Yang, and Y. Yan-tao, "Approach of moving objects detection in active video surveillance," in *Proceedings of the 48th IEEE Conference on Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009*, 2009, pp. 3130–3136.

[41] M. Verma, A. Majumdar, and B. Chatterjee, "Edge detection in fingerprints," *Pattern Recognition*, vol. 20, no. 5, pp. 513–523, 1987.

[42] M. Nixon and A. S. Aguado, *Feature Extraction and Image Processing, Second Edition*. Academic Press, 2008.

[43] B. Heisele, T. Serre, and T. Poggio, "A component-based framework for face detection and identification," *International Journal of Computer Vision*, vol. 74, no. 2, pp. 167–181, 2007.

[44] B. Leung, "Component-based car detection in street scene images," Master's thesis, School of Computer Science and Electrical Engineering, MIT University, 2004.

[45] H. Chidiac and D. Ziou, "Classification of image edges," in *Proceedings of the Conference on Vision Interface, Canada*, 1999, pp. 17–24.

[46] P. A. Mlsna and J. J. Rodriguez, in *Handbook of Image and Video Processing (2nd ed.)*, A. Bovik, Ed. San Diego, CA, USA: Elsevier Academic Press, 2005, ch. Gradient and Laplacian edge detection, pp. 535–553.

[47] A. J. Pinho and L. B. Almeida, "A review on edge detection based on filtering and differentiation," *REVISTA DO DETUA*, vol. 2, no. 1, pp. 113–126, 1997.

[48] M. Roushdy, "Comparative study of edge detection algorithms applying on the grayscale noisy image using morphological filter," *ICGST International Journal on Graphics, Vision and Image Processing*, vol. 6, pp. 17–23, 2007.

[49] M. Heath, S. Sarkar, T. Sanocki, and K. Bowyer, "Comparison of edge detectors: a methodology and initial study," *Computer Vision and Image Understanding*, vol. 69, no. 1, pp. 38–54, 1998.

[50] E. Clavier, S. Clavier, and J. Labiche, "Image sorting - image classification: A global approach," in *ICDAR '99: Proceedings of the Fifth International Conference on Document Analysis and Recognition*. Washington, DC, USA: IEEE Computer Society, 1999, pp. 123–129.

[51] R. A. Kirsch, "Computer determination of the constituent structure of biological images," *Computers and Biomedical Research*, vol. 4, no. 3, pp. 315–328, 1971.

[52] M. Pharr and G. Humphreys, *Physically based Rendering from Theory to Implementation*. Burlington, MA, USA: Morgan Kaufmann, 2010.

[53] J. Canny, "Finding edges and lines in images," Massachusetts Institute of Technology, Cambridge, MA, USA, Tech. Rep., 1983.

[54] J. S. Chen and G. Medioni, "Detection, localization, and estimation of edges," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 2, pp. 191–198, 1989.

[55] N. Hautiere, J.-P. Tarel, and R. Bremond, "Perceptual hysteresis thresholding: Towards driver visibility descriptors," in *Proceedings*

of the IEEE International Conference on Intelligent Computer Communication and Processing, 2007, pp. 89–96.

[56] M. Sharifi, M. Fathy, and M. T. Mahmoudi, "A classified and comparative study of edge detection algorithms," in *Proceedings of the International Conference on Information Technology: Coding and Computing*, 2002, pp. 117–120.

[57] D. Marr and E. Hildreth, "Theory of edge detection," *Proceedings of the Royal Society of London Series B*, vol. 207, pp. 187–217, 1980.

[58] J. Shen and S. Castan, "Towards the unification of band-limited derivative operators for edge detection," *Signal Processing*, vol. 31, no. 2, pp. 103–119, 1993.

[59] J. J. Clark, "Authenticating edges produced by zero-crossing algorithms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 1, pp. 43–57, 1989.

[60] D. Sen and S. K. Pal, "Gradient histogram: Thresholding in a region of interest for edge detection," *Image and Vision Computing*, vol. 28, no. 4, pp. 677–695, 2010.

[61] L. Ding and A. Goshtasby, "On the Canny edge detector," *Pattern Recognition*, vol. 34, no. 3, pp. 721–725, 2001.

[62] M. Basu, "Gaussian-based edge-detection methods: a survey," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 32, no. 3, pp. 252–260, 2002.

[63] C.-C. Kang and W.-J. Wang, "A novel edge detection method based on the maximizing objective function," *Pattern Recognition*, vol. 40, no. 2, pp. 609–618, 2007.

[64] H. Jeong and C. Kim, "Adaptive determination of filter scales for edge detection," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 14, pp. 579–585, 1992.

[65] T. Lindeberg, *Scale-Space Theory in Computer Vision*. Norwell, MA, USA: Kluwer Academic Publishers, 1994.

[66] ——, "Edge detection and ridge detection with automatic scale selection," *International Journal of Computer Vision*, vol. 30, no. 2, pp. 117–156, 1998.

[67] P. Bao, L. Zhang, and X. Wu, "Canny edge detection enhancement by scale multiplication," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 9, pp. 1485–1490, 2005.

[68] B. G. Schunck, "Edge detection with gaussian filters at multiple scales of resolution," in *Advances in Image Analysis*. McGraw Hill, 1992, pp. 75–105.

[69] F. Bergholm, "Edge focusing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 9, no. 6, pp. 726–741, 1987.

[70] V. Lacroix, "The primary raster: a multiresolution image description," in *Proceedings of the 10th International Conference on Pattern Recognition*, vol. 1, 1990, pp. 903–907.

[71] A. Goshtasby, "On edge focusing," *Image and Vision Computing*, vol. 12, no. 4, pp. 247–256, 1994.

[72] A. C. Bovik, T. S. Huang, and D. C. J. Munson, "Nonparametric tests for edge detection in noise," *Pattern Recognition*, vol. 19, no. 3, pp. 209–219, 1986.

[73] J. S. Huang and D. H. Tseng, "Statistical theory of edge detection," *Computer Vision, Graphics, and Image Processing*, vol. 43, no. 3, pp. 337–346, 1988.

[74] J. Aron and L. Kurz, "Edge detection using anova techniques," in *Proceedings of the International Symposium on Information Theory*, Israel, 1973.

[75] Z. Hou and T. S. Koh, "Robust edge detection," *Pattern Recognition*, vol. 36, no. 9, pp. 2083 – 2091, 2003.

[76] D. H. Lim and S. J. Jang, "Comparison of two-sample tests for edge detection in noisy images," *Journal of the Royal Statistical Society. Series D (The Statistician)*, vol. 51, no. 1, pp. 21–30, 2002.

[77] S. Mallat, *A Wavelet Tour of Signal Processing*, 2nd ed. London: Academic Press, 1999.

[78] S. Mallat and S. Zhong, "Characterization of signals from multiscale edges," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 7, pp. 710–732, 1992.

[79] D. Heric and D. Zazula, "Combined edge detection using wavelet transform and signal registration," *Image and Vision Computing*, vol. 25, no. 5, pp. 652–662, 2007.

[80] M.-Y. Shih and D.-C. Tseng, "A wavelet-based multiresolution edge detection and tracking," *Image and Vision Computing*, vol. 23, no. 4, pp. 441–451, 2005.

[81] S. Yi, D. Labate, G. Easley, and H. Krim, "A shearlet approach to edge analysis and detection," *IEEE Transactions on Image Processing*, vol. 18, no. 5, pp. 929–941, 2009.

[82] W. Jiang, K.-M. Lam, and T.-Z. Shen, "Efficient edge detection using simplified gabor wavelets," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 39, no. 4, pp. 1036–1047, 2009.

[83] D.-S. Kim, W.-H. Lee, and I.-S. Kweon, "Automatic edge detection using 33 ideal binary pixel patterns and fuzzy-based edge thresholding," *Pattern Recognition Letters*, vol. 25, no. 1, pp. 101–106, 2004.

[84] A. J. Pinho and L. B. Almeida, "Edge detection filters based on artificial neural networks," in *ICIAP '95: Proceedings of the 8th International Conference on Image Analysis and Processing*. London, UK: Springer-Verlag, 1995, pp. 159–164.

[85] S. Di Zenzo, "A note on the gradient of a multi-image," *Computer Vision, Graphics, and Image Processing*, vol. 33, no. 1, pp. 116–125, 1986.

[86] P. Trahanias and A. Venetsanopoulos, "Vector order statistics operators as color edge detectors," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 26, no. 1, pp. 135–143, 1996.

[87] A. Hajjar and T. Chen, "VLSI architecture for real-time edge linking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 1, pp. 89–94, 1999.

[88] J. R. Parker, *Algorithms for Image Processing and Computer Vision*. New York, USA: John Wiley & Sons, Inc., 1996.

[89] P. E. Hart, "How the Hough transform was invented," *IEEE Signal Processing Magazine*, vol. 26, no. 6, pp. 18–22, 2009.

[90] A. A. Farag and E. J. Delp, "Edge linking by sequential search," *Pattern Recognition*, vol. 28, no. 5, pp. 611–633, 1995.

[91] G. W. Cook and E. J. Delp, "Multiresolution sequential edge linking," in *Proceedings of the 1995 International Conference on Image Processing*, vol. 1, 1995, pp. 41–44.

[92] M. C. Shin, D. Goldgof, and K. W. Bowyer, "Comparison of edge detectors using an object recognition task," *Proceesings of the IEEE*

Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 1360–1365, 1999.

[93] D. Huttenlocher, G. Klanderman, and W. Rucklidge, "Comparing images using the Hausdorff distance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, pp. 850–863, 1993.

[94] K. Bowyer, C. Kranenburg, and S. Dougherty, "Edge detector evaluation using empirical ROC curves," *Computer Vision and Image Understanding*, vol. 84, no. 1, pp. 77–103, 2001.

[95] D. R. Martin, C. C. Fowlkes, and J. Malik, "Learning to detect natural image boundaries using local brightness, color, and texture cues," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 5, pp. 530–549, 2004.

[96] M. Shin, D. Goldgof, K. Bowyer, and S. Nikiforou, "Comparison of edge detection algorithms using a structure from motion task," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 31, no. 4, pp. 589–601, 2001.

[97] R. Moreno, D. Puig, C. Julia, and M. Garcia, "A new methodology for evaluation of edge detectors," in *Proceedings of the 16th IEEE International Conference on Image Processing (ICIP), 2009*, 2009, pp. 2157–2160.

[98] S. Sumathi and P. Surekha, *Computational Intelligence Paradigms*. Florida, USA: CRC Press, 2010.

[99] D. Fogel and M. T., *Advanced Algorithms and Operators*. Bristol and Philadelphia: Institute of Physics Publishing, 1999.

[100] T. Back, *Evolutionary Algorithms in Theory and Practice*. New York, USA: Oxford University Press, 1996.

[101] L. J. Fogel, "On the organisation of intellect," Ph.D. dissertation, University of California at Los Angeles, 1964.

[102] L. J. Fogel and D. B. Fogel, "Artificial intelligence through evolutionary programming," US Army Research Institute, Tech. Rep., 1986.

[103] H. P. Schwefel, "Experimentelle optimierung einer zweiphasendse teil," AEG Research Institute, Berlin, Technical Report No. 35 of the Project MHDStaustrahlrohr 11.034/68, 1968.

[104] J. H. Holland, "Outline for a logical theory of adaptive systems," *J. ACM*, vol. 9, no. 3, pp. 297–314, 1962.

[105] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.

[106] L. J. Eshelman and J. D. Schaffer, "Real-coded genetic algorithms and interval-schemata," in *Foundation of Genetic Algorithms 2*, D. L. Whitley, Ed. San Mateo, CA: Morgan Kaufmann., 1993, pp. 187–202.

[107] K. Deb and R. B. Agrawal, "Simulated binary crossover for continuous search space," Department of Mechanical Engineering, Indian Institute of Technology, Kanpur, India, Tech. Rep., 1994.

[108] S. Janson and M. Middendorf, "A hierarchical particle swarm optimizer for noisy and dynamic environments," *Genetic Programming and Evolvable Machines*, vol. 7, no. 4, pp. 329–354, 2006.

[109] G. Beni and J. Wang, "Swarm intelligence in cellular robotic systems," in *Proceedings of NATO Advanced Workshop on Robots and Biological Systems*, Tuscany, Italy, June 1989.

[110] M. Dorigo and T. Stutzle, *Ant Colony Optimization*. MIT Press, 2004.

[111] M. Clerc, *Particle Swarm Optimization*. London, UK: ISTE Publishing Company, 2006.

[112] G.Beni and J. Wang, "Stochastic searching networks," in *Proceedings of the 1st IEE International Conference on ANNs*, London, UK, 1989, pp. 329–331.

[113] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi, "GSA: A gravitational search algorithm," *Information Sciences*, vol. 179, no. 13, pp. 2232–2248, 2009.

[114] A. Colorni, M. Dorigo, and V. Maniezzo, "Distributed optimization by ant colonies," in *European Conference on Artificial Life*, 1991, pp. 134–142.

[115] F. Kennedy, R. Eberhart, and Y. Shi, *Swarm Intelligence*. San Francisco, CA: Morgan Kaufmann, 2001.

[116] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, 1995, pp. 1942–1948.

[117] M. F. Tasgetiren, P. N. Suganthan, and Q.-Q. Pan, "A discrete particle swarm optimization algorithm for the generalized traveling salesman problem," in *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*. New York, NY, USA: ACM, 2007, pp. 158–167.

[118] J. Wang, Z. Kuang, X. Xu, and Y. Zhou, "Discrete particle swarm optimization based on estimation of distribution for polygonal approximation problems," *Expert Systems with Applications*, vol. 36, no. 5, pp. 9398–9408, 2009.

[119] A. H. Kashan and B. Karimi, "A discrete particle swarm optimization algorithm for scheduling parallel machines," *Computers and Industrial Engineering*, vol. 56, no. 1, pp. 216–223, 2009.

[120] M. R. Al Rashidi and M. E. El-Hawary, "A survey of particle swarm optimization applications in electric power systems," *Transactions on Evolutionary Computaion*, vol. 13, no. 4, pp. 913–918, 2009.

[121] A. Rakitianskaia and A. P. Engelbrecht, "Training neural networks with PSO in dynamic environments," in *Proceedings of the Eleventh Conference on Congress on Evolutionary Computation*. Piscataway, NJ, USA: IEEE Press, 2009, pp. 667–673.

[122] J. Pugh and A. Martinoli, "Multi-robot learning with particle swarm optimization," in *AAMAS '06: Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*. New York, NY, USA: ACM, 2006, pp. 441–448.

[123] N. Jin and Y. Rahmat-Samii, "Particle swarm optimization for antenna designs in engineering electromagnetics," *Journal of Artificial Evolution and Applications*, vol. 2008, pp. 1–10, 2008.

[124] J. Duro and J. de Oliveira, "Particle swarm optimization applied to the chess game," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2008, pp. 3702–3709.

[125] Y. Shi and R. Eberhart, "Empirical study of particle swarm optimization," in *Proceedings of the 1999 Congress on Evolutionary Computation*, vol. 3, 1999, pp. 1945–1950.

[126] R. Hassan, B. Cohanim, and O. de Weck, "Comparison of particle swarm optimisation and the genetic algorithm," in *Proceedings of the Structures, Structural Dynamics and Materials Conference*, American Institute of Aeronautics and Astronautics, 2005, pp. 1–13.

[127] C. Lopez-Molina, H. Bustince, J. Fernandez, P. Couto, and B. De Baets, "A gravitational approach to edge detection based on triangular norms," *Pattern Recognition*, vol. 43, no. 11, pp. 3730–3741, 2010.

[128] X. Zhuang, G. Yang, and H. Zhu, "A model of image feature extraction inspired by ant swarm system," in *Proceedings of the 2008 Fourth International Conference on Natural Computation*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 553–557.

[129] S. B. M. Ouadfel, "Ant colony system with local search for Markov random field image segmentation," vol. 1, 2003, pp. 133–136.

[130] E. Lakehal, "A swarm intelligence based approach for image feature extraction," in *Proceedings of the International Conference on Multimedia Computing and Systems*, 2009, pp. 31–35.

[131] T. Mirzayans, N. Parimi, P. Pilarski, C. Backhouse, L. Wyard-Scott, and P. Musilek, "A swarm-based system for object recognition," *Neural Network World*, vol. 15, pp. 351–365, 2005.

[132] Y. Z. M. Wang, "Joint transform correlator based on joint image feature extraction using swarm intelligence method," in *Proceedings of the IEEE International Conference on Mechatronics and Automation*, 2009, pp. 4964–4969.

[133] A. Etemad and T. White, "An ant-inspired algorithm for detection of image edge features," *Applied Soft Computing*, vol. 11, no. 8, pp. 4883–4893, 2011.

[134] M. Alipoor, S. Imandoost, and J. Haddadnia, "Designing edge detection filters using particle swarm optimization," in *Proceedings of the 18th Iranian Conference on Electrical Engineering (ICEE)*, 2010, pp. 548–552.

[135] A. Aghamohammadi, A. Prabuwono, S. Sahran, and M. Mogharrebi, "Solar cell panel crack detection using particle swarm optimization algorithm," in *Proceedings of the International Conference on Pattern Analysis and Intelligent Robotics (ICPAIR)*, vol. 1, 2011, pp. 160–164.

[136] M. Heath, S. Sarkar, T. Sanocki, and K. Bowyer, *Images from South Florida University for edge detector comparison*, 1998, available from http://marathon.csee.usf.edu/edge/edge_detection.

[137] N. L. Fernández-García, A. Carmona-Poyato, R. Medina-Carnicer, and F. J. Madrid-Cuevas, *Images from automatic generation of consensus ground truth for comparison of edge detection techniques*, 2008, available from http://www.uco.es/~ma1fegan/Comunes/investigacion/imagenes/ground-truth.html.

[138] G. Chen and Y. Hong Yang, "Edge detection by regularized cubic b-spline fitting," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 25, no. 4, pp. 636–643, 1995.

[139] J. Nascimento and J. Marques, "Adaptive snakes using the EM algorithm," *IEEE Transactions on Image Processing*, vol. 14, no. 11, pp. 1678–1686, 2005.

[140] K. S. Shanmugam, F. M. Dickey, and J. A. Green, "An optimal frequency domain filter for edge detection in digital pictures," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 1, no. 1, pp. 37–49, 1979.

[141] L. Hu, H. Cheng, and M. Zhang, "A high performance edge detector based on fuzzy inference rules," *Information Sciences*, vol. 177, no. 21, pp. 4768–4784, 2007.

[142] B. Tremblais and B. Augereau, "A fast multiscale edge detection algorithm based on a new edge preserving pde resolution scheme," in *Proceedings of the 17th International Conference on Pattern Recognition*, vol. 2. Los Alamitos, CA, USA: IEEE Computer Society, 2004, pp. 811–814.

[143] J.-W. Han, J.-H. Kim, S.-H. Cheon, J.-O. Kim, and S.-J. Ko, "A novel

image interpolation method using the bilateral filter," *IEEE Transactions on Consumer Electronics*, vol. 56, no. 1, pp. 175–181, 2010.

[144] S. Konishi, A. Yuille, J. Coughlan, and S. C. Zhu, "Statistical edge detection: learning and evaluating edge cues," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 1, pp. 57–74, 2003.

[145] R. Lee and J. Lin, "An elastic contour matching model for tropical cyclone pattern recognition," *IEEE Transactions on Systems, Man, and Cybernetics: Part B*, vol. 31, no. 3, pp. 413–417, 2001.

[146] L. MacEachern and T. Manku, "Genetic algorithms for active contour optimization," in *Proceedings of the 1998 IEEE International Symposium on Circuits and Systems*, vol. 4, 1998, pp. 229–232.

[147] K. E. Parsopoulos and M. N. Vrahatis, "Particle swarm optimizer in noisy and continuously changing environments," in *Artificial Intelligence and Soft Computing*. IASTED/ACTA Press, 2001, pp. 289–294.

[148] J. Vesterstrom and R. Thomsen, "A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems," in *Proceedings of the Congress on Evolutionary Computation*, vol. 2, 2004, pp. 1980–1987.

[149] T. Poggio, H. Voorhees, and A. Yuille, "A regularized solution to edge detection," *Journal of Complexity*, vol. 4, no. 2, pp. 106–123, 1988.

[150] A. P. Witkin, "Scale-space filtering," in *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1983, pp. 1019–1022.

[151] R. Deriche, "Using Canny's criteria to derive a recursively implemented optimal edge detector," *International Journal of Computer Vision*, vol. 1, no. 2, pp. 167–187, 1987.

[152] J. Ferwerda, "Elements of early vision for computer graphics," *IEEE Computer Graphics and Applications*, vol. 21, no. 5, pp. 22–33, 2001.

[153] M. Fesharaki and G. Hellestrand, "A new edge detection algorithm based on a statistical approach," in *Proceedings of the International Symposium on Speech, Image Processing and Neural Networks (ISSIPNN)*, 1994, pp. 21–24.

[154] K. Ghosh, S. Sarkar, and K. Bhaumik, "A possible mechanism of zero crossing detection using the concept of the extended classical receptive field of retinal ganglion cells," *Biological Cybernetics*, vol. 93, no. 1, pp. 1–5, 2005.

[155] N. Otsu, "A threshold selection method for gray level histograms," *IEEE Transaction on Systems, Man, Cybernetics*, vol. 9, pp. 62–66, 1976.

[156] X. Hu and R. Eberhart, "Solving constrained nonlinear optimization problems with particle swarm optimization," in *Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI)*, 2002, pp. 203–206.

[157] K. Sedlaczek and P. Eberhard, "Optimization of nonlinear mechanical systems under constraints with the particle swarm method," in *Proceedings in Applied Mathematics and Mechanics*, 2004, pp. 169–170.

[158] M. Breaban, M. Ionita, and C. Croitoru, "A new PSO approach to constraint satisfaction," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2007, pp. 1948–1954.

[159] J. Fuentes Cabrera and C. Coello Coello, "Handling constraints in particle swarm optimization using a small population size," in *MICAI 2007: Advances in Artificial Intelligence*, ser. Lecture Notes in Computer Science, vol. 4827. Springer Berlin/Heidelberg, 2007, pp. 41–51.

[160] C. Monson and K. Seppi, "Linear equality constraints and homomorphous mappings in PSO," in *Proceedings of the IEEE Congress on Evolutionary Computation*, vol. 1, 2005, pp. 73–80.

[161] G. Coath and S. Halgamuge, "A comparison of constraint-handling methods for the application of particle swarm optimization to constrained nonlinear optimization problems," in *Proceedings of the IEEE Congress on Evolutionary Computation*, vol. 4, 2003, pp. 2419–2425.

[162] K. E. Parsopoulos and M. N. Vrahatis, "Particle swarm optimization method for constrained optimization problems," in *Proceedings of the Euro-International Symposium on Computational Intelligence*. IOS Press, 2002, pp. 214–220.

[163] R. Medina-Carnicer, F. Madrid-Cuevas, A. Carmona-Poyato, and R. Muoz-Salinas, "On candidates selection for hysteresis thresholds in edge detection," *Pattern Recognition*, vol. 42, no. 7, pp. 1284–1296, 2009.

[164] S. Aditya and S. Katti, "Flexcast: Graceful wireless video streaming," in *Proceedings of the 17th annual international conference on Mobile computing and networking*. ACM, 2011, pp. 277–288.

[165] R. Mendes, J. Kennedy, and J. Neves, "The fully informed particle swarm: Simpler, maybe better," *IEEE Transactions on Evolutionary Computation*, vol. 8, pp. 204–210, 2004.

[166] E. Laskari, K. Parsopoulos, and M. Vrahatis, "Particle swarm optimization for integer programming," in *Proceedings of the Congress on Evolutionary Computation*, vol. 2, 2002, pp. 1582–1587.

[167] E. Badekas and N. Papamarkos, "Automatic evaluation of document binarization results," in *Proceedings of the 10th Iberoamerican Congress conference on Progress in Pattern Recognition, Image Analysis and Applications*. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 1005–1014.

[168] S. S. Bukhari, F. Shafait, and T. M. Breuel, "Adaptive binarization of unconstrained hand-held camera-captured document images," *Journal of Universal Computer Science*, vol. 15, no. 18, pp. 3343–3363, 2009.

[169] J. Sauvola and M. Pietikinen, "Adaptive document image binarization," *Pattern Recognition*, vol. 33, pp. 225–236, 2000.

[170] T. W. Ridler and S. Calvard, "Picture thresholding using an iterative selection method," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 8, no. 8, pp. 630–632, 1978.

[171] J. S. P. W. A. Kapur, "A new method for gray-level picture thresholding using the entropy of the histogram," *Computer Vision, Graphics, and Image Processing*, vol. 29, no. 3, pp. 273–285, 1985.

[172] W.-H. Tsai, "Moment-preserving thresholding: a new approach," *Computer Vision, Graphics and Image Processing*, vol. 29, no. 3, pp. 377–393, 1985.

[173] P. Rosin, "Unimodal thresholding," *Pattern Recognition*, vol. 34, no. 11, pp. 2083–2096, 2001.

[174] R. Medina-Carnicer, F. Madrid-Cuevas, N. Fernndez-Garca, and A. Carmona-Poyato, "Evaluation of global thresholding techniques in non-contextual edge detection," *Pattern Recognition Letters*, vol. 26, no. 10, pp. 1423–1434, 2005.

[175] Y. Nakagawa and A. Rosenfeld, "Some experiments on variable thresholding," *Pattern Recognition*, vol. 11, no. 3, pp. 191–204, 1979.

[176] F. Deravi and S. Pal, "Grey level thresholding using second-order statistics," *Pattern Recognition Letters*, vol. 1, no. 56, pp. 417–422, 1983.

[177] W. Niblack, *An Introduction to Digital Image Processing*. Prentice Hall, 1986.

[178] P. Viola and M. Jones, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, pp. 137–154, 2004.

[179] K. Bowyer, C. Kranenburg, and S. Dougherty, "Ground truth images from edge detector evaluation using empirical ROC curves," available from http://figment.csee.usf.edu/edge/roc/, accessed in 2011.

[180] M. G. Omran, A. P. Engelbrecht, and A. Salman, "Bare bones differential evolution," *European Journal of Operational Research*, vol. 196, no. 1, pp. 128–139, 2009.

[181] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[182] F. van den Bergh and A. Engelbrecht, "A study of particle swarm optimization particle trajectories," *Information Sciences*, vol. 176, no. 8, pp. 937–971, 2006.

[183] J. Kennedy, "Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance," in *Proceedings of the 1999 Congress on Evolutionary Computation*, vol. 3, 1999, pp. 1931–1938.

[184] D. Bratton and J. Kennedy, "Defining a standard for particle swarm optimization," in *Proceedings of the IEEE Swarm Intelligence Symposium*, 2007, pp. 120–127.

[185] J. Kennedy, "Bare bones particle swarms," in *Proceedings of the 2003 IEEE Swarm Intelligence Symposium*, 2003, pp. 80–87.

[186] Y. Shi and R. Eberhart, "Comparing inertia weights and constriction factor in particle swarm optimisation," in *Proceedings of Evolutionary Computing*, 2000, pp. 84–89.

[187] M. Reyes-Sierra and C. A. Coello Coello, "Multi-objective particle swarm optimizers: A survey of the state-of-the-art," *International Journal of Computational Intelligence Research*, vol. 2, no. 3, pp. 287–308, 2006.

[188] M. A. Montes de Oca and T. Stützle, "Convergence behavior of the fully informed particle swarm optimization algorithm," in *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*. New York, NY, USA: ACM, 2008, pp. 71–78.

[189] S. Holm, "A simple sequentially rejective multiple test procedure," *Scandinavian Journal of Statistics*, vol. 6, no. 2, pp. 65–70, 1979.

[190] H. Liu, E. Howely, and J. Duggan, "Particle swarm optimisation with gradually increasing directed neighbourhoods," in *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*. New York, NY, USA: ACM, 2011, pp. 29–36.

[191] S. Akat and V. Gazi, "Particle swarm optimization with dynamic neighborhood topology: Three neighborhood strategies and preliminary results," in *Proceedings of IEEE Swarm Intelligence Symposium*, 2008, pp. 1–8.

[192] P. Suganthan, "Particle swarm optimiser with neighbourhood operator," in *Proceedings of the IEEE Congress on Evolutionary Computation*, vol. 3, 1999, pp. 1958–1962.

[193] J. Kennedy, "Stereotyping: improving particle swarm performance with cluster analysis," in *Proceedings of the IEEE Congress on Evolutionary Computation*, vol. 2, 2001, pp. 1507–1512.

[194] A. Mohais, R. Mendes, C. Ward, and C. Posthoff, "Neighborhood re-structuring in particle swarm optimization," in *AI 2005: Advances in Artificial Intelligence*, ser. Lecture Notes in Computer Sci-

ence, S. Zhang and R. Jarvis, Eds. Springer Berlin/Heidelberg, 2005, vol. 3809, pp. 776–785.

[195] K. Veeramachaneni, T. Peram, C. Mohan, and L. A. Osadciw, "Optimization using particle swarms with near neighbor interactions," in *Lecture Notes Computer Science*. Springer Verlag, 2003, pp. 110–121.

[196] X. Li, "A multi-modal particle swarm optimizer based on fitness Euclidean-distance ratio," in *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*. New York, NY, USA: ACM, 2007, pp. 78–85.

[197] D. Schor, W. Kinsner, and J. Anderson, "A study of optimal topologies in swarm intelligence," in *Proceedings of the 23rd Canadian Conference on Electrical and Computer Engineering (CCECE)*, 2010, pp. 1–8.

[198] C. Novak and S. Shafe, "Color edge detection," in *Proceedings of the DARPA Image Understanding Workshop*, Los Angeles, CA, USA, 1987, pp. 35–37.

Appendix A

Sigmoid Function

The general form of the sigmoid function is formulated as:

$$Y = \frac{\alpha_1}{1 + e^{-\alpha_2(x-\alpha_3)}}$$

where x is the input variable, α_1 is the range of Y, α_2 is the gain coefficient, and α_3 is the point of maximum gain. The maximum gain is the point at which the slope of the sigmoid function is maximum or at which the value of the second derivative is zero. Therefore, the maximum gain equals $\alpha_1\alpha_2/4$. The threshold and saturation points for the sigmoid function are defined as the values of x at which the third derivative of the sigmoid function are zero, i.e.,

$$\begin{aligned} TH &= \alpha_3 + 1.317/\alpha_2 \\ SAT &= \alpha_3 - 1.317/\alpha_2 \end{aligned}$$

where TH and SAT are the threshold and saturation points for the sigmoid function. Therefore,

$$\begin{aligned} \alpha_2 &= \frac{2.634}{TH - SAT} \\ \alpha_3 &= \frac{SAT + TH}{2} \end{aligned}$$

Appendix B

Detailed Results from Chapters 6 and 7

In Tables B.1, B.2 and B.3, the columns FCG, RT, SG, TBG, VNT and TRO show the 95% confidence intervals for the localisation accuracy of the CanPSO-based algorithm with different static topologies after 30 independent runs for each image at each noise level.

In Table B.4, the columns Random, GIDN and SRMT show the 95% confidence intervals for the localisation accuracy of CanPSO and FIPS-based edge detection algorithms after 30 independent runs for each image at each noise level.

Table B.1: The performance of the CanPSO-based algorithm with different static topologies

Image	Noise Level	95% Confidence Interval for Accuracy					
		FCG	RT	SG	TBG	VNT	TRO
Sat	G22	0.9056±0.0020	0.8787±0.0184	0.8842±0.0070	0.9076±0.0056	0.9030±0.0032	0.8949±0.0101
Sat	G18	0.8664±0.0085	0.8714±0.0029	0.8650±0.0199	0.8473±0.0059	0.8756±0.0107	0.8676±0.0057
Sat	G14	0.9202±0.0062	0.9525±0.0159	0.9141±0.0173	0.9331±0.0022	0.9546±0.0156	0.9342±0.0112
Sat	G10	0.8668±0.0186	0.8518±0.0034	0.8900±0.0078	0.8743±0.0167	0.8717±0.0004	0.8896±0.0108
Sat	G6	0.8225±0.0180	0.8515±0.0135	0.8418±0.0073	0.8512±0.0047	0.8467±0.0165	0.8353±0.0157
Cube	G22	0.7117±0.0131	0.8097±0.0189	0.7171±0.0171	0.7386±0.0193	0.7523±0.0237	0.7717±0.0160
Cube	G18	0.7069±0.0256	0.7651±0.0066	0.7589±0.0194	0.7514±0.0139	0.7719±0.0040	0.7798±0.0153
Cube	G14	0.7064±0.0197	0.7580±0.0106	0.7256±0.0246	0.7270±0.0110	0.7685±0.0048	0.7701±0.0146
Cube	G10	0.7284±0.0003	0.7007±0.0205	0.6867±0.0085	0.7079±0.0111	0.7037±0.0083	0.7051±0.0104
Cube	G6	0.7245±0.0169	0.7591±0.0088	0.7380±0.0067	0.7356±0.0220	0.7279±0.0120	0.7555±0.0127
Wall	G22	0.8306±0.0196	0.8997±0.0143	0.8552±0.0118	0.8678±0.0063	0.8736±0.0207	0.8667±0.0168
Wall	G18	0.8537±0.0216	0.9076±0.0148	0.8902±0.0053	0.9045±0.0089	0.8953±0.0141	0.8956±0.0179
Wall	G14	0.8823±0.0220	0.9767±0.0169	0.9315±0.0061	0.9552±0.0131	0.9488±0.0151	0.9602±0.0188
Wall	G10	0.7813±0.0069	0.7814±0.0154	0.7748±0.0071	0.7865±0.0028	0.7804±0.0188	0.7615±0.0113
Wall	G6	0.8293±0.0015	0.8877±0.0038	0.8036±0.0174	0.7896±0.0235	0.8067±0.0227	0.8495±0.0027
Street	G22	0.8343±0.0064	0.9002±0.0169	0.8272±0.0162	0.8468±0.0113	0.8617±0.0211	0.8662±0.0118
Street	G18	0.8714±0.0127	0.8798±0.0145	0.8796±0.0088	0.8922±0.0073	0.8917±0.0057	0.8879±0.0135
Street	G14	0.7748±0.0063	0.7819±0.0080	0.7667±0.0154	0.7628±0.0129	0.7741±0.0107	0.7802±0.0071
Street	G10	0.8664±0.0137	0.8842±0.0006	0.8766±0.0121	0.8435±0.0054	0.8504±0.0224	0.8408±0.0073
Street	G6	0.7727±0.0166	0.8115±0.0117	0.7868±0.0221	0.7914±0.0121	0.8232±0.0033	0.8283±0.0137
Sat	N0.1	0.5513±0.0256	0.5351±0.0092	0.5800±0.0151	0.5754±0.0166	0.5821±0.0000	0.5913±0.0169
Sat	N0.2	0.5315±0.0094	0.5682±0.0276	0.5271±0.0282	0.5685±0.0032	0.6080±0.0028	0.5888±0.0181
Sat	N0.3	0.4506±0.0299	0.5882±0.0147	0.4919±0.0261	0.4980±0.0272	0.5250±0.0269	0.5561±0.0215
Sat	N0.4	0.3655±0.0280	0.4141±0.0057	0.3918±0.0131	0.3783±0.0243	0.3786±0.0235	0.3970±0.0166
Sat	N0.5	0.2604±0.0343	0.3659±0.0220	0.3096±0.0213	0.3236±0.0170	0.3352±0.0281	0.3406±0.0275
Cube	N0.1	0.6439±0.0236	0.6951±0.0034	0.6716±0.0260	0.6541±0.0118	0.6991±0.0076	0.7010±0.0130
Cube	N0.2	0.6330±0.0112	0.6345±0.0248	0.6334±0.0021	0.6648±0.0084	0.6414±0.0098	0.6367±0.0179
Cube	N0.3	0.6198±0.0173	0.6076±0.0146	0.6312±0.0026	0.6374±0.0037	0.6160±0.0157	0.5992±0.0161
Cube	N0.4	0.5089±0.0250	0.5893±0.0161	0.5337±0.0268	0.5422±0.0121	0.5746±0.0192	0.5753±0.0201
Cube	N0.5	0.4329±0.0183	0.4252±0.0099	0.4423±0.0063	0.4375±0.0138	0.4258±0.0110	0.4279±0.0141
Wall	N0.1	0.5780±0.0204	0.6491±0.0256	0.5945±0.0198	0.6317±0.0172	0.6474±0.0062	0.6589±0.0223
Wall	N0.2	0.5436±0.0226	0.5919±0.0100	0.5638±0.0259	0.5613±0.0052	0.5921±0.0185	0.5787±0.0161
Wall	N0.3	0.6052±0.0244	0.6560±0.0155	0.6330±0.0219	0.6419±0.0207	0.6638±0.0015	0.6830±0.0191
Wall	N0.4	0.4909±0.0172	0.4957±0.0007	0.4994±0.0069	0.4643±0.0259	0.4544±0.0221	0.4795±0.0090
Wall	N0.5	0.2822±0.0070	0.3157±0.0163	0.2707±0.0329	0.2749±0.0139	0.3133±0.0151	0.3134±0.0116
Street	N0.1	0.6666±0.0188	0.6303±0.0091	0.6824±0.0012	0.6787±0.0057	0.6502±0.0065	0.6394±0.0141
Street	N0.2	0.5433±0.0183	0.5395±0.0063	0.5549±0.0076	0.5442±0.0106	0.5365±0.0146	0.5346±0.0124
Street	N0.3	0.5049±0.0216	0.5907±0.0246	0.5218±0.0187	0.5497±0.0230	0.5614±0.0161	0.5827±0.0226
Street	N0.4	0.3943±0.0040	0.4116±0.0258	0.3763±0.0138	0.3991±0.0042	0.4004±0.0282	0.3828±0.0152
Street	N0.5	0.3535±0.0168	0.3656±0.0124	0.3589±0.0144	0.3579±0.0217	0.3598±0.0087	0.3731±0.0145

Table B.2: The performance of the BBPSO-based algorithm with different static topologies

Image	Noise Level	95% Confidence Interval for Accuracy					
		FCG	RT	SG	TBG	VNT	TRO
Sat	G22	0.9057±0.0110	0.8494±0.0002	0.9108±0.0140	0.8677±0.0020	0.8800±0.0056	0.8580±0.0058
Sat	G18	0.8681±0.0129	0.9151±0.0026	0.8767±0.0214	0.8582±0.0097	0.8966±0.0161	0.8979±0.0076
Sat	G14	0.9246±0.0144	0.9296±0.0019	0.9383±0.0203	0.9201±0.0058	0.9590±0.0026	0.9515±0.0079
Sat	G10	0.8740±0.0172	0.9314±0.0100	0.8935±0.0171	0.8969±0.0199	0.9168±0.0031	0.9475±0.0130
Sat	G6	0.8092±0.0022	0.7893±0.0009	0.7875±0.0006	0.7572±0.0120	0.7232±0.0255	0.7274±0.0017
Cube	G22	0.7203±0.0207	0.7308±0.0169	0.7427±0.0015	0.7564±0.0069	0.7303±0.0138	0.7236±0.0188
Cube	G18	0.7150±0.0204	0.7827±0.0045	0.7363±0.0218	0.7235±0.0250	0.7503±0.0047	0.7954±0.0118
Cube	G14	0.6924±0.0021	0.6394±0.0082	0.6693±0.0084	0.6638±0.0059	0.6592±0.0063	0.6488±0.0051
Cube	G10	0.7302±0.0146	0.8058±0.0152	0.7389±0.0156	0.7508±0.0187	0.7618±0.0210	0.7803±0.0147
Cube	G6	0.7368±0.0225	0.7811±0.0019	0.7665±0.0033	0.7432±0.0206	0.7258±0.0232	0.7491±0.0121
Wall	G22	0.8303±0.0115	0.8301±0.0077	0.8348±0.0054	0.8310±0.0176	0.8214±0.0053	0.8401±0.0095
Wall	G18	0.8531±0.0109	0.8852±0.0153	0.8568±0.0186	0.8715±0.0022	0.8943±0.0141	0.8731±0.0131
Wall	G14	0.8684±0.0019	0.9383±0.0141	0.8457±0.0213	0.8592±0.0221	0.8929±0.0105	0.9346±0.0080
Wall	G10	0.7959±0.0223	0.8664±0.0110	0.8301±0.0074	0.8343±0.0186	0.8297±0.0162	0.8514±0.0162
Wall	G6	0.8405±0.0201	0.8585±0.0058	0.8677±0.0004	0.8596±0.0130	0.8232±0.0203	0.8316±0.0131
Street	G22	0.8197±0.0017	0.7716±0.0137	0.7957±0.0000	0.8035±0.0085	0.7377±0.0238	0.7353±0.0080
Street	G18	0.8718±0.0116	0.9353±0.0203	0.8776±0.0082	0.9166±0.0014	0.9146±0.0213	0.8891±0.0163
Street	G14	0.7696±0.0074	0.7133±0.0022	0.7642±0.0222	0.7421±0.0006	0.7724±0.0018	0.7371±0.0049
Street	G10	0.8729±0.0167	0.9014±0.0106	0.8909±0.0187	0.8951±0.0094	0.9204±0.0022	0.9216±0.0132
Street	G6	0.7700±0.0097	0.8772±0.0030	0.7696±0.0245	0.7526±0.0192	0.8098±0.0234	0.8296±0.0062
Sat	N0.1	0.5466±0.0093	0.5980±0.0189	0.5422±0.0166	0.5562±0.0260	0.5646±0.0130	0.5953±0.0139
Sat	N0.2	0.5504±0.0282	0.5678±0.0028	0.5934±0.0054	0.5724±0.0217	0.5599±0.0060	0.5791±0.0152
Sat	N0.3	0.4598±0.0253	0.5418±0.0014	0.4833±0.0256	0.4548±0.0294	0.4801±0.0249	0.5185±0.0129
Sat	N0.4	0.3850±0.0315	0.3873±0.0021	0.4291±0.0053	0.4038±0.0178	0.3897±0.0072	0.3975±0.0167
Sat	N0.5	0.2535±0.0086	0.2789±0.0178	0.2449±0.0212	0.2507±0.0299	0.2610±0.0089	0.2870±0.0131
Cube	N0.1	0.6370±0.0067	0.7247±0.0244	0.6282±0.0260	0.6677±0.0068	0.7054±0.0225	0.6973±0.0157
Cube	N0.2	0.6290±0.0094	0.6472±0.0122	0.6260±0.0141	0.6285±0.0172	0.6342±0.0118	0.6457±0.0108
Cube	N0.3	0.6200±0.0141	0.5712±0.0001	0.6256±0.0132	0.5746±0.0084	0.5785±0.0106	0.5727±0.0073
Cube	N0.4	0.5107±0.0170	0.5301±0.0183	0.5192±0.0078	0.5304±0.0122	0.5226±0.0159	0.5232±0.0176
Cube	N0.5	0.4348±0.0179	0.4810±0.0212	0.4434±0.0225	0.4587±0.0176	0.4756±0.0104	0.4842±0.0192
Wall	N0.1	0.5784±0.0147	0.5906±0.0067	0.5842±0.0225	0.5750±0.0152	0.5963±0.0054	0.6030±0.0105
Wall	N0.2	0.5509±0.0222	0.6232±0.0234	0.5703±0.0096	0.5961±0.0143	0.5926±0.0238	0.5980±0.0227
Wall	N0.3	0.5897±0.0017	0.5855±0.0161	0.5636±0.0190	0.5721±0.0125	0.5855±0.0103	0.5874±0.0089
Wall	N0.4	0.4713±0.0008	0.4097±0.0214	0.4372±0.0007	0.4512±0.0152	0.4161±0.0068	0.4206±0.0110
Wall	N0.5	0.2767±0.0100	0.3414±0.0161	0.2707±0.0332	0.2750±0.0283	0.3151±0.0156	0.3383±0.0129
Street	N0.1	0.6726±0.0193	0.6941±0.0031	0.6896±0.0077	0.6708±0.0150	0.6651±0.0209	0.6733±0.0112
Street	N0.2	0.5488±0.0205	0.5934±0.0098	0.5646±0.0236	0.5619±0.0162	0.5854±0.0113	0.5936±0.0148
Street	N0.3	0.4954±0.0052	0.3984±0.0050	0.4815±0.0006	0.4663±0.0078	0.4305±0.0023	0.4223±0.0052
Street	N0.4	0.3994±0.0222	0.4336±0.0027	0.4146±0.0251	0.3924±0.0023	0.4136±0.0309	0.3893±0.0126
Street	N0.5	0.3489±0.0107	0.3995±0.0303	0.3447±0.0025	0.3821±0.0270	0.3585±0.0237	0.3828±0.0206

Table B.3: The performance of the FIPS-based algorithm with different static topologies

Image	Noise Level	95% Confidence Interval for Accuracy					
		FCG	RT	SG	TBG	VNT	TRO
Sat	G22	0.8960±0.0037	0.8975±0.0047	0.9098±0.0007	0.9235±0.0101	0.9285±0.0002	0.9478±0.0040
Sat	G18	0.8765±0.0190	0.8945±0.0108	0.8484±0.0091	0.8682±0.0226	0.8861±0.0090	0.9282±0.0143
Sat	G14	0.9079±0.0024	0.9450±0.0161	0.8978±0.0165	0.9235±0.0021	0.9489±0.0207	0.9585±0.0091
Sat	G10	0.8819±0.0209	0.8961±0.0014	0.8498±0.0175	0.8577±0.0144	0.8834±0.0052	0.9061±0.0109
Sat	G6	0.8281±0.0167	0.8335±0.0112	0.8036±0.0004	0.8232±0.0103	0.8261±0.0008	0.8449±0.0138
Cube	G22	0.7124±0.0136	0.7304±0.0096	0.6918±0.0125	0.7093±0.0025	0.7289±0.0102	0.7385±0.0114
Cube	G18	0.7199±0.0231	0.7420±0.0155	0.6895±0.0136	0.7119±0.0113	0.7322±0.0153	0.7510±0.0190
Cube	G14	0.7060±0.0125	0.7165±0.0088	0.6863±0.0210	0.7031±0.0080	0.7298±0.0030	0.7458±0.0104
Cube	G10	0.7320±0.0162	0.7586±0.0207	0.7091±0.0100	0.7369±0.0189	0.7546±0.0198	0.7799±0.0180
Cube	G6	0.7226±0.0108	0.7624±0.0233	0.7041±0.0042	0.7368±0.0167	0.7489±0.0249	0.7722±0.0168
Wall	G22	0.8375±0.0176	0.8581±0.0041	0.8120±0.0188	0.8246±0.0232	0.8511±0.0121	0.8889±0.0104
Wall	G18	0.8463±0.0053	0.8726±0.0056	0.8322±0.0048	0.8469±0.0222	0.8602±0.0179	0.8945±0.0053
Wall	G14	0.8707±0.0028	0.9018±0.0147	0.8600±0.0004	0.8835±0.0043	0.8869±0.0204	0.8999±0.0087
Wall	G10	0.7769±0.0080	0.7836±0.0175	0.7604±0.0140	0.7851±0.0109	0.8064±0.0012	0.8254±0.0123
Wall	G6	0.8302±0.0125	0.8496±0.0098	0.8095±0.0235	0.8280±0.0071	0.8657±0.0107	0.8816±0.0107
Street	G22	0.8442±0.0194	0.8675±0.0078	0.8162±0.0187	0.8331±0.0061	0.8595±0.0151	0.8744±0.0133
Street	G18	0.8818±0.0190	0.8994±0.0178	0.8535±0.0187	0.8805±0.0124	0.9077±0.0083	0.9287±0.0178
Street	G14	0.7837±0.0197	0.8096±0.0073	0.7566±0.0180	0.7726±0.0162	0.7973±0.0185	0.8206±0.0132
Street	G10	0.8595±0.0056	0.8791±0.0176	0.8450±0.0001	0.8713±0.0154	0.8692±0.0108	0.8930±0.0114
Street	G6	0.7720±0.0116	0.7921±0.0143	0.7525±0.0022	0.7744±0.0107	0.7837±0.0123	0.8024±0.0128
Sat	N0.1	0.5517±0.0150	0.5858±0.0135	0.5314±0.0147	0.5509±0.0133	0.5709±0.0275	0.5901±0.0140
Sat	N0.2	0.5238±0.0066	0.5292±0.0258	0.5099±0.0074	0.5392±0.0002	0.5537±0.0010	0.5547±0.0159
Sat	N0.3	0.4523±0.0175	0.4682±0.0157	0.4309±0.0109	0.4512±0.0089	0.4681±0.0094	0.4836±0.0164
Sat	N0.4	0.3576±0.0071	0.3920±0.0297	0.3439±0.0207	0.3753±0.0294	0.3978±0.0309	0.4283±0.0179
Sat	N0.5	0.2603±0.0173	0.2725±0.0267	0.2403±0.0024	0.2663±0.0007	0.2745±0.0057	0.2780±0.0219
Cube	N0.1	0.6373±0.0069	0.6542±0.0009	0.6226±0.0151	0.6278±0.0042	0.6488±0.0094	0.6606±0.0039
Cube	N0.2	0.6467±0.0248	0.6732±0.0133	0.6158±0.0004	0.6358±0.0052	0.6385±0.0215	0.6515±0.0191
Cube	N0.3	0.6125±0.0064	0.6417±0.0022	0.5983±0.0237	0.6071±0.0236	0.6352±0.0239	0.6635±0.0042
Cube	N0.4	0.4891±0.0007	0.5012±0.0162	0.4849±0.0151	0.5057±0.0220	0.5257±0.0048	0.5507±0.0082
Cube	N0.5	0.4296±0.0116	0.4372±0.0117	0.4123±0.0224	0.4297±0.0045	0.4540±0.0019	0.4656±0.0114
Wall	N0.1	0.5986±0.0275	0.6267±0.0118	0.5621±0.0007	0.5807±0.0005	0.5849±0.0239	0.5883±0.0199
Wall	N0.2	0.5450±0.0161	0.5601±0.0103	0.5239±0.0013	0.5410±0.0053	0.5474±0.0079	0.5601±0.0131
Wall	N0.3	0.6088±0.0179	0.6408±0.0209	0.5859±0.0125	0.6118±0.0219	0.6306±0.0255	0.6571±0.0189
Wall	N0.4	0.4956±0.0205	0.5094±0.0126	0.4718±0.0159	0.4903±0.0292	0.5106±0.0065	0.5453±0.0160
Wall	N0.5	0.2747±0.0079	0.3013±0.0307	0.2607±0.0228	0.2910±0.0201	0.3143±0.0269	0.3361±0.0188
Street	N0.1	0.6728±0.0195	0.6902±0.0034	0.6478±0.0212	0.6587±0.0175	0.6850±0.0098	0.7081±0.0111
Street	N0.2	0.5492±0.0209	0.5634±0.0061	0.5245±0.0155	0.5381±0.0281	0.5586±0.0068	0.5928±0.0131
Street	N0.3	0.5227±0.0284	0.5530±0.0205	0.4885±0.0187	0.5129±0.0076	0.5353±0.0264	0.5500±0.0242
Street	N0.4	0.3805±0.0027	0.4046±0.0285	0.3716±0.0028	0.4015±0.0139	0.4105±0.0221	0.4292±0.0153
Street	N0.5	0.3518±0.0143	0.3698±0.0122	0.3331±0.0236	0.3506±0.0229	0.3750±0.0129	0.3990±0.0130

Table B.4: The performance of the PSO-based algorithm with dynamic topologies

Image	Noise Level	95% Confidence Interval for Accuracy					
		CanPSO		BBPSO		FIPS	
		GIDN	SRMT	GIDN	SRMT	GIDN	SRMT
Sat	G22	0.8989±0.0207	0.9088±0.0167	0.8594±0.0208	0.8694±0.0050	0.9580±0.0056	0.9680±0.0176
Sat	G18	0.8913±0.0148	0.9012±0.0027	0.9250±0.0212	0.9350±0.0010	0.9385±0.0101	0.9485±0.0083
Sat	G14	0.9726±0.0173	0.9826±0.0096	0.9396±0.0152	0.9497±0.0087	0.9685±0.0146	0.9786±0.0134
Sat	G10	0.8719±0.0171	0.8819±0.0159	0.9414±0.0039	0.9513±0.0071	0.9161±0.0055	0.9260±0.0158
Sat	G6	0.8714±0.0159	0.8814±0.0015	0.7992±0.0125	0.8091±0.0086	0.8549±0.0085	0.8648±0.0087
Cube	G22	0.8297±0.0119	0.8397±0.0140	0.7408±0.0144	0.7509±0.0055	0.7485±0.0218	0.7584±0.0138
Cube	G18	0.7851±0.0084	0.7953±0.0154	0.7927±0.0072	0.8025±0.0231	0.7609±0.0015	0.7709±0.0121
Cube	G14	0.7779±0.0162	0.7881±0.0019	0.6493±0.0249	0.6593±0.0261	0.7560±0.0198	0.7659±0.0092
Cube	G10	0.7209±0.0247	0.7310±0.0164	0.8158±0.0179	0.8259±0.0142	0.7900±0.0195	0.7998±0.0193
Cube	G6	0.7790±0.0177	0.7890±0.0014	0.7910±0.0100	0.8009±0.0127	0.7822±0.0061	0.7922±0.0094
Wall	G22	0.9197±0.0072	0.9297±0.0089	0.8401±0.0017	0.8500±0.0119	0.8988±0.0092	0.9088±0.0081
Wall	G18	0.9276±0.0091	0.9374±0.0070	0.8952±0.0103	0.9051±0.0018	0.9045±0.0079	0.9146±0.0082
Wall	G14	0.9968±0.0133	1.0067±0.0144	0.9483±0.0109	0.9584±0.0042	0.9099±0.0164	0.9199±0.0150
Wall	G10	0.8013±0.0032	0.8113±0.0112	0.8764±0.0038	0.8865±0.0159	0.8353±0.0174	0.8453±0.0070
Wall	G6	0.9077±0.0018	0.9176±0.0193	0.8686±0.0032	0.8787±0.0038	0.8915±0.0166	0.9016±0.0107
Street	G22	0.9202±0.0135	0.9302±0.0063	0.7816±0.0169	0.7915±0.0141	0.8845±0.0057	0.8945±0.0102
Street	G18	0.8998±0.0116	0.9099±0.0108	0.9453±0.0030	0.9554±0.0173	0.9386±0.0188	0.9486±0.0108
Street	G14	0.8017±0.0011	0.8116±0.0077	0.7232±0.0045	0.7332±0.0039	0.8305±0.0101	0.8405±0.0043
Street	G10	0.9041±0.0098	0.9142±0.0075	0.9114±0.0104	0.9212±0.0111	0.9030±0.0006	0.9128±0.0087
Street	G6	0.8315±0.0112	0.8416±0.0154	0.8872±0.0007	0.8972±0.0139	0.8122±0.0159	0.8222±0.0135
Sat	N0.1	0.5547±0.0077	0.5647±0.0002	0.6078±0.0182	0.6178±0.0110	0.6001±0.0163	0.6102±0.0038
Sat	N0.2	0.5882±0.0096	0.5982±0.0147	0.5778±0.0077	0.5878±0.0167	0.5646±0.0139	0.5744±0.0124
Sat	N0.3	0.6082±0.0229	0.6183±0.0052	0.5517±0.0004	0.5618±0.0228	0.4934±0.0225	0.5034±0.0151
Sat	N0.4	0.4340±0.0082	0.4441±0.0151	0.3973±0.0319	0.4074±0.0233	0.4385±0.0242	0.4484±0.0116
Sat	N0.5	0.3857±0.0021	0.3958±0.0176	0.2889±0.0291	0.2987±0.0254	0.2881±0.0010	0.2980±0.0104
Cube	N0.1	0.7150±0.0153	0.7251±0.0089	0.7346±0.0245	0.7446±0.0159	0.6708±0.0100	0.6807±0.0125
Cube	N0.2	0.6545±0.0079	0.6645±0.0126	0.6572±0.0114	0.6673±0.0197	0.6614±0.0233	0.6716±0.0102
Cube	N0.3	0.6279±0.0250	0.6376±0.0244	0.5813±0.0028	0.5915±0.0002	0.6733±0.0259	0.6834±0.0238
Cube	N0.4	0.6092±0.0220	0.6193±0.0031	0.5400±0.0172	0.5500±0.0246	0.5607±0.0189	0.5707±0.0129
Cube	N0.5	0.4453±0.0283	0.4553±0.0113	0.4909±0.0180	0.5010±0.0233	0.4756±0.0196	0.4856±0.0193
Wall	N0.1	0.6691±0.0104	0.6790±0.0138	0.6006±0.0182	0.6105±0.0113	0.5983±0.0106	0.6082±0.0127
Wall	N0.2	0.6118±0.0192	0.6218±0.0054	0.6331±0.0125	0.6432±0.0134	0.5701±0.0183	0.5800±0.0126
Wall	N0.3	0.6761±0.0177	0.6862±0.0198	0.5955±0.0097	0.6056±0.0196	0.6670±0.0175	0.6771±0.0188
Wall	N0.4	0.5156±0.0043	0.5257±0.0144	0.4197±0.0245	0.4295±0.0251	0.5554±0.0026	0.5653±0.0091
Wall	N0.5	0.3357±0.0088	0.3457±0.0251	0.3515±0.0024	0.3614±0.0197	0.3459±0.0083	0.3558±0.0168
Street	N0.1	0.6501±0.0035	0.6601±0.0056	0.7040±0.0206	0.7140±0.0121	0.7182±0.0090	0.7283±0.0043
Street	N0.2	0.5595±0.0104	0.5695±0.0130	0.6034±0.0240	0.6133±0.0167	0.6029±0.0032	0.6130±0.0113
Street	N0.3	0.6105±0.0001	0.6205±0.0192	0.4085±0.0137	0.4185±0.0189	0.5600±0.0163	0.5700±0.0100
Street	N0.4	0.4316±0.0085	0.4417±0.0238	0.4437±0.0197	0.4536±0.0290	0.4392±0.0140	0.4492±0.0161
Street	N0.5	0.3856±0.0287	0.3955±0.0103	0.4095±0.0005	0.4194±0.0023	0.4088±0.0085	0.4188±0.0191