Information Sciences 246 (2013) 28–51

Contents lists available at SciVerse ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

دانلود کننده مقالات علمی freepapers.ir FREE papers

A novel particle swarm optimisation approach to detecting continuous, thin and smooth edges in noisy images

Mahdi Setayesh [a,*], Mengjie Zhang [a], Mark Johnston [b]

[a] School of Engineering and Computer Science, Victoria University of Wellington, PO Box 600, Wellington 6140, New Zealand
[b] School of Mathematics, Statistics and Operations Research, Victoria University of Wellington, PO Box 600, Wellington 6140, New Zealand

A R T I C L E I N F O

Article history:
Received 30 December 2011
Received in revised form 11 April 2013
Accepted 16 May 2013
Available online 31 May 2013

Keywords:
Edge detection
Particle swarm optimisation
Noisy images
Impulsive and Gaussian noise

A B S T R A C T

Detection of continuous edges is a hard problem and most edge detection algorithms produce jagged and thick edges particularly in noisy images. This paper firstly presents a novel constrained optimisation model for detecting continuous, thin and smooth edges in such images. Then two particle swarm optimisation-based algorithms are applied to search for good solutions. These two algorithms utilise two different constraint handling methods: penalising and preservation. The algorithms are 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 is used for the comparison of these algorithms. Experimental results show that the proposed edge detectors are more robust under noisy conditions and their performances are better than the Canny and RRO algorithms for the images corrupted by impulsive and Gaussian noise. The proposed algorithm based on the penalising method is faster than the algorithm using the preservation method to handle the constraints.

© 2013 Elsevier Inc. All rights reserved.

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 [57]. 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 [18].

Many algorithms have been proposed using various different paradigms such as curve fitting [7], optimisation of a criterion [6,41], image transforms [26,53] statistical testing [32] and soft computing [2,22] 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 [3], statistical-based [32], and scale space-based [56] edge detectors. The Gaussian-based algorithms often malfunction at corners and curves [54] and establish double edges in areas with high frequencies of information. They also displace edges and produce false edges [3]. These methods use a Gaussian filter as a smoothing technique to reduce noise, which often causes edges to be weak

* Corresponding author. Tel.: +64 2102782961.
E-mail address: mahdi.setayesh@ecs.vuw.ac.nz (M. Setayesh).

0020-0255/$ - see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.ins.2013.05.031

دانلود کننده مقالات علمی freepapers.ir FREE papers

and broken as a side effect [11,25]. Although there are several algorithms that utilise sharpening techniques to reduce these side effects, they suffer from producing jagged edges as a side effect of these techniques [19].

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 [32]. 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 [29].

Another group of algorithms uses the scale space theory [33] 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 [3,26].

Several techniques have been proposed to compensate for broken edges, such as sequential edge linking (SEL) [12], multi-resolution SEL (M-SEL) [9] 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 on objects with complicated shapes [34]. Snake-based methods are another type of these techniques. They utilise an active contour model to detect an object boundary [31]. These methods need to have a priori knowledge about the boundary and are very slow [35].

Most of the edge detection algorithms described above use a convolution of an image with an $n \times n$ matrix, where usually $n \leqslant 5$ 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 $\left(t(n) = \frac{(n^2-1)^n}{2} = O(n^{2n})\right)$ [62]. 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 [28]. 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 [1]. PSO is very stable and efficient in noisy environments [44]. Its comparison with evolutionary algorithms has shown that PSO has a high capability to optimise noisy functions [38,43,58] and it has been successfully applied to many problems in noisy environments, such as image segmentation [40] and vision tracking [61]. PSO has a good potential for edge detection in noisy images, but surprisingly, it has not been sufficiently analysed for tackling edge detection problems.

This paper aims to develop a new PSO based approach to edge detection in noisy images with the goal of extracting continuous and thin edges, and reducing broken and jagged edges. The new approach will be examined and compared to the modified version of the Canny algorithm proposed in [50] and the RRO algorithm proposed in [32] on two sets of noisy images. The localisation accuracy will be used to measure the performance of different algorithms.

The rest of this paper is organised as follows. Section 2 provides background information on PSO and a brief overview of edge detection. The new PSO-based edge detection approach will be introduced in Section 3 followed by a summary of two new algorithms in Section 4. Sections 5 and 6 present the experiments and results with discussion. Section 7 draws concluding remarks.

2. Background

This section provides a summary of the necessary background on particle swarm optimisation and edge detection.

2.1. Particle swarm optimisation

PSO is a branch of swarm intelligence inspired by the social behaviour of animals and biological populations. It simulates a simplified social model, such as a flock of birds and a school of fish. Although it was originally developed to optimise continuous nonlinear functions, there are some discrete versions of PSO to work on discrete search spaces [59].

In PSO, there is a population of m particles. These particles move through an n-dimensional search space. The location of each particle in the search space at time t is represented as the vector $\overrightarrow{X}_i(t) = (x_{i1}(t), x_{i2}(t), \ldots, x_{in}(t))$, where i is the index of the particle in the population. Its location is updated according to its own experience (current motion and particle memory influences) and that of its neighbours (swarm influence). Each particle has a velocity represented by $\overrightarrow{V}_i(t)$ that is added to $\overrightarrow{X}_i(t)$ at each iteration of PSO as in Eq. (1).

دانلود کننده مقالات علمی freepapers.ir FREE papers

$$\overrightarrow{X}_i(t+1) = \overrightarrow{X}_i(t) + \overrightarrow{V}_i(t+1). \tag{1}$$

The velocity is changed based on three components: current motion influence, particle memory influence, and swarm influence [55]:

$$V_{i,j}(t+1) = wV_{i,j}(t) + C_1 r_{1,j}(X_{pbest_i,j} - X_{i,j}(t)) + C_2 r_{2,j}(X_{leader,j} - X_{i,j}(t)) \tag{2}$$

where j shows the index of jth element of the corresponding vector; $r_{1,j}$, and $r_{2,j}$ are uniform random variables between 0 and 1; w is an inertia weight to control the impact of the previous velocity; C_1 and C_2 are called the self and swarm confidence learning factors that represent the attraction of a particle toward its best previous location and the best particle of the population, respectively; $\overrightarrow{X}_{pbest_i}$ denotes the best position of ith particle so far; and $\overrightarrow{X}_{leader}$ is the position of a particle for guiding other particles toward better regions of the search space.

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 [23]. In the second group, the algorithms add a penalty to the fitness of the particles which violate constraints [49]. The third group (partitioning methods) divides 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 [5,16]. 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 [39].

2.2. 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 [32]. Although many algorithms have been proposed to detect edges in noisy images, this section only briefly reviews a modified version of Canny [11] and RRO [32] as they are very commonly used in edge detection in noisy images and will be compared with the new approaches proposed in this paper.

2.2.1. Revised versions of 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 [6]. This algorithm is very popular because it has a complete process of edge detection and has good localisation accuracy. 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 to 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 [24] 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. [46] and the scale-space theory proposed by Witkin [60]. This method adaptively finds optimal filter scales for each pixel before extracting edge maps. [24] defined an energy function as a function over continuous scale space as follows:

$$\begin{aligned} E(\sigma) &= \iint (f - G\bigstar f)^2 + \lambda \left| \nabla \frac{1}{\sigma(x,y)} \right|^2 dxdy \\ &= \iint \left(f - \left[\iint \frac{1}{2\pi\sigma^2} e^{-\frac{\alpha^2+\beta^2}{2\sigma^2}} f(x-\alpha, y-\beta) d\alpha d\beta \right] \right)^2 + \lambda \left(\left(-\frac{\sigma_x}{\sigma^2} \right)^2 + \left(-\frac{\sigma_y}{\sigma^2} \right)^2 \right) dxdy \end{aligned} \tag{3}$$

where ☆ is the convolution operator, σ is the size of Gaussian filter and $\lambda = 60$ is a parameter to control the smoothness ability of the Canny algorithm. It is obvious that when $\sigma \to 0$, the first term, i.e, $f - G☆f$ 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:

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

[24] 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, an NMS technique is used as an edge thinning algorithm [10]. The NMS technique proposed by Canny chooses a pixel as an edge only when the gradient 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

دانلود کننده مقالات علمی freepapers.ir FREE papers

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 [50]. Canny proposed the hysteresis thresholding technique, inspired by biological mechanisms, for detecting more continuous edges [14]. This technique usually utilises two threshold values (high and low) to tackle the problem of broken edges [6]. 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 [20]. Manual determination of these two threshold values is very time consuming. Therefore, many unsupervised techniques have been proposed to determine these values [50]. Sen and Pal [50] proposed an automatic way in order to estimate these two threshold values. The low and high threshold values are computed as follows:

$$Threshold_{High} = 2\sigma\sqrt{f_u ln2} \tag{5}$$

$$Threshold_{Low} = \frac{1}{2}Threshold_{High} \tag{6}$$

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

2.2.2. 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 developed by [32] based on the robust rank-order (RRO) test [32]. This algorithm operates better than other statistical-based edge detectors such as Wilcoxon and *t*-test-based edge detectors [4,15] 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 (grey and white) where each subregion contains 12 pixels. [32] 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 grey ($\mathfrak{G}$) and the white ($\mathfrak{W}$) subregions. At least one of the window partitions 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 $\mu_{\mathfrak{w}}$. [32] did not make any assumption about the nature of these two distributions. He defined the modified observations, α_i and β_i as follows:

$$\alpha_i = \begin{cases} \mathfrak{g}_i + \delta & \mathfrak{g}_i \in \mathfrak{G} \\ \mathfrak{w}_i & \mathfrak{w}_i \in \mathfrak{W} \end{cases} \tag{7}$$

$$\beta_i = \begin{cases} \mathfrak{g}_i - \delta & \mathfrak{g}_i \in \mathfrak{G} \\ \mathfrak{w}_i & \mathfrak{w}_i \in \mathfrak{W} \end{cases} \tag{8}$$

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

$$H_0^{\uparrow} : \mu_{\mathfrak{g}} + \delta \geqslant \mu_{\mathfrak{w}} \ versus \ H_1^{\uparrow} : \mu_{\mathfrak{g}} + \delta < \mu_{\mathfrak{w}} \tag{9}$$

and

$$H_0^{\downarrow} : \mu_{\mathfrak{g}} - \delta \leqslant \mu_{\mathfrak{w}} \ versus \ H_1^{\downarrow} : \mu_{\mathfrak{g}} - \delta > \mu_{\mathfrak{w}} \tag{10}$$

He 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 RRO test on the modified observations, α_i and β_i to determine the existence of a significant difference in grey level between them. [32]'s method first considers eight different edge patterns and then the RRO statistic is obtained for testing $H_0^{\uparrow}$ against $H_1^{\uparrow}$ on α_i for each edge pattern. 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:

$$U_{\alpha} = \frac{12(U(\mathfrak{G} + \delta, \mathfrak{W}) - U(\mathfrak{W}, \mathfrak{G} + \delta))}{2\sqrt{V_{\mathfrak{G}+\delta} + V_{\mathfrak{W}+\delta} + U(\mathfrak{G} + \delta, \mathfrak{W})U(\mathfrak{W}, \mathfrak{G} + \delta)}} \tag{11}$$

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

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

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 significance level *sl*.

دانلود کننده مقالات علمی
freepapers.ir
FREE papers

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 a few parameters that can be easily tuned by the user in order to detect edges in noisy images.

Note that PSO has not been applied to edge detection before our preliminary works [51,52]. In this paper, we extend those works by adopting the PSO-based algorithm with the Otsu's method and providing further results and analysis.

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 in intensity of the pixels inside of each subregion. Therefore the search space in this optimisation problem is all possible curves which partition this region into two subregions. A new encoding scheme is developed to represent these curves in this search space. To evaluate each curve, a new 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.

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 a 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.

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 Fig. 1a. 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 Fig. 1a). Let pixel C be the middle pixel of the consecutive pixels on the continuous edge and $2L + 1$

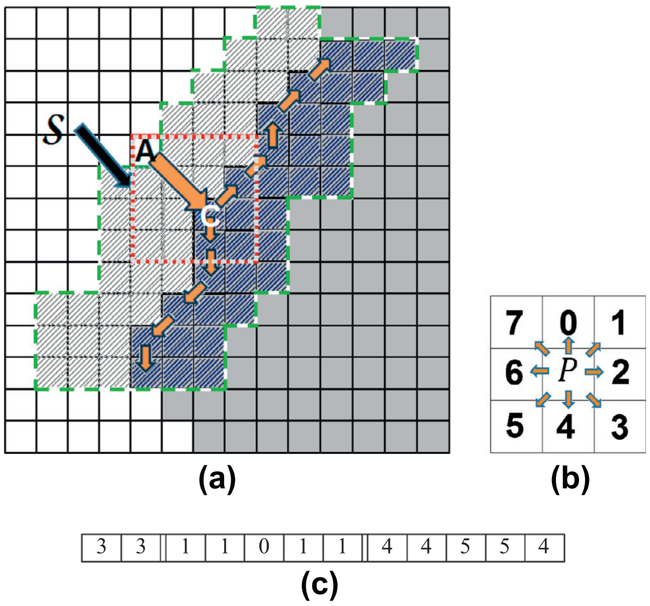

Fig. 1. The particle encoding scheme. (a) An example of a curve with two regions; (b) eight movement directions from a pixel P; (c) the particle representing the curve with $L = 5$. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

دانلود کننده مقالات علمی freepapers.ir FREE papers

be the number of the pixels on it. The dotted red square in Fig. 1a 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 dotted 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 sets 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 sets 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 Fig. 1b. By changing the values of these components in the range of interest, all possible continuous edges inside of a region with the area of $(2L + SqrSize)^2$ can be represented by this new encoding.

For example, the edge passing through pixel C, which is located inside the square in Fig. 1a, is encoded as shown in Fig. 1c. In this example, $SqrSize = 4$, $L = 5$, and $m_1, m_2, \ldots, m_5$ show the movement directions from the point C towards the top and $m_6, m_7, \ldots, m_{10}$ towards the bottom. In this example, all striped pixels enclosed by the dashed green lines are used to evaluate the curve, as follows.

3.2. A new fitness function

The new 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 new 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, a new 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.

3.2.1. New 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 [6], second [36] and fourth derivatives [17]. 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 *six* ways of dividing the neighbourhood of each pixel of an image into two regions according to the *eight* possible movement directions, as shown in Fig. 2. Note that the patterns corresponding with the movement directions 2 and 6, and 0 and 4 are same and so appear once each in Fig. 2. In each edge pattern in Fig. 2, let D and B 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 large and their intraset distances are small, the edge magnitude will be large; and also if the interset distance is small and their intraset distances are large, 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 interset distance). Therefore, we formulate the edge magnitude at pixel P in movement direction m, $EdgeMag_m(P)$ as Eq. (13) to maximise the interset distance ($InterDis_m(P)$) between

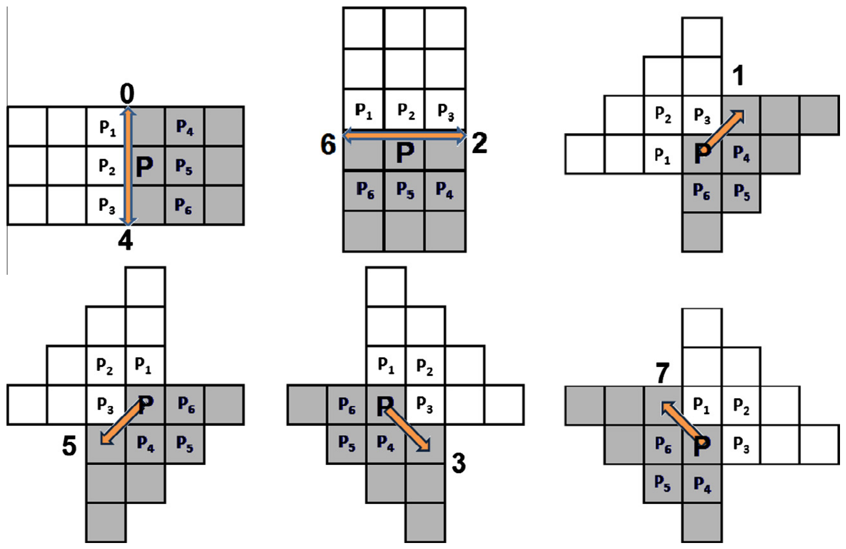

Fig. 2. Six ways of moving from pixel P to a neighbouring pixel.

دانلود کننده مقالات علمی freepapers.ir FREE papers

the regions and minimise the intraset distance ($IntraDis_m(P)$) within the regions. To avoid dividing by zero, the denominator is increased by 1.

$$EdgeMag_m(P) = \frac{InterDis_m(P)}{1 + IntraDis_m(P)} \tag{13}$$

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 intraset and interset distances are calculated as Eqs. (14) and (15).

$$InterDis_m(P) = min(1, |avg_{m,d}(P) - avg_{m,l}(P)|/w_1) \tag{14}$$

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 Fig. 2), as calculated in $avg_{m,d}(P) = \frac{1}{n}\sum_{P_i \in D} I_{P_i}$ and $avg_{m,l}(P) = \frac{1}{n}\sum_{P_i \in B} I_{P_i}$; $n = |B| = |D|$ ($n = 9$ in Fig. 2); 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:

$$IntraDis_m(P) = \frac{1}{\binom{n}{2}} \left(\sum_{\substack{P_i, P_j \in D \\ i > j}} min(1, |I_{P_i} - I_{P_j}|/w_2) + \sum_{\substack{P_i, P_j \in B \\ i > j}} min(1, |I_{P_i} - I_{P_j}|/w_2) \right) \tag{15}$$

where w_2 is a weight factor.

3.2.2. NMS factor for edge thinning

Non-maxima suppression (NMS) is one of the most important edge thinning techniques [57]. 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 $P_1, P_2, \ldots, P_6$ (as shown in Fig. 2) 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:

$$NMS_m(P) = |\{P_i | i \in \{1, \ldots, 6\}, EdgeMag_m(P_i) < EdgeMag_m(P)\}| \tag{16}$$

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 Fig. 2. 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 the 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 Eq. (17).

$$TotalEdgeMag_m(P) = EdgeMag_m(P) \times \frac{1}{1 + e^{-2(NMS_m(P) - 4)}} \tag{17}$$

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 value may be wrongly recognised as a non-edge. For this reason, thresholding techniques often cause broken edges. 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 Eq. (18):

$$PScore_m(P) = \frac{1}{1 + e^{-\frac{3.317}{TH}(TotalEdgeMag_m(P) - 0.6229TH)}} \tag{18}$$

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 [42] as will be described in Section 3.3. We use Eq. (18) 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.

دانلود کننده مقالات علمی freepapers.ir FREE papers

3.2.3. *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:

$$U(C) = \frac{1}{255 \times 2L} \sum_{i=1}^{2L} |I_{P_{i+1}} - I_{P_i}| \tag{19}$$

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 Eq. (19), 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 Eq. (20) such that it maximises the possibility of the pixels on the curve and minimises its uniformity factor.

$$PScore(C) = \frac{\sum_{P_i \in C} PScore_{m_i}(P_i)/(2L+1)}{1 + U(C)} \tag{20}$$

where m_i is the movement direction from pixel P_i on curve C.

3.2.4. *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 Eq. (21).

$$CC(m_i, m_{i+1}) = \begin{cases} |m_i - m_{i+1}|/w_3 & |m_i - m_{i+1}| \leqslant 4 \\ (8 - |m_i - m_{i+1}|)/w_3 & \textit{otherwise} \end{cases} \tag{21}$$

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 Eq. (22) which is the average of the curvature cost of all single pixels on the curve.

$$CCost(C) = \frac{1}{2L-2} \left(\sum_{i=1}^{L-1} CC(m_i, m_{i+1}) + \sum_{i=L+1}^{2L-1} CC(m_i, m_{i+1}) \right) \tag{22}$$

3.2.5. *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:

$$Fitness(C) = PScore(C) - CCost(C) \tag{23}$$

subject to two constraints:

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

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.

3.3. *Otsu's Method for Estimation of TH*

Otsu's method [42] 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 [42]. The edge magnitudes of the pixels belonging to the first subset are less than or equal with 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.

دانلود کننده مقالات علمی freepapers.ir FREE papers

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

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 Eq. (18), the local maximum of the edge magnitude of each pixel *LocalEdgeMag*(*P*) is first calculated using Eq. (24).

$$LocalEdgeMag(P) = \max_{i=0}^{7}(TotalEdgeMag_i(P)) \tag{24}$$

Applying this equation results in an edge magnitude map which can be used as the input to Otsu's method in order to estimate the value of parameter *TH* in Eq. (18). Fig. 3 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*.

4. Summary of the two proposed algorithms

Two PSO-based algorithms are used for the optimisation of the proposed model to detect edges in noisy images. As can be seen in the overall flowchart of the PSO-based algorithm in Fig. 4, the dotted red square first moves over 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 to find a best curve which can be fitted on a continuous edge passing inside the dotted red square. After applying the PSO-based algorithms, the dotted red square moves to the next block.

The selection of a constraint handling method is very problem dependent. As described earlier, there are many approaches to handling constraints in PSO. Two different methods are applied here. The first is based on a preservation method and the second is based on a penalising method. This section summarises these two algorithms after explaining the truncation method to convert the real values to integers in the PSO-based algorithms.

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 dotted red square (as shown in Fig. 1a) 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* dotted 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 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 6.3.

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 Eq. (1). Many discrete versions of PSO use a simple truncation method to convert real numbers to integers [27]. Instead of using a simple truncation method, the following method is used to truncate the values of particle positions to integers:

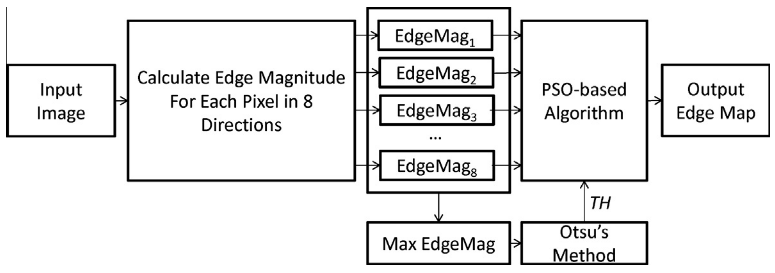

Fig. 3. Paradigm of PSO-based edge detector.

دانلود کننده مقالات علمی freepapers.ir FREE papers

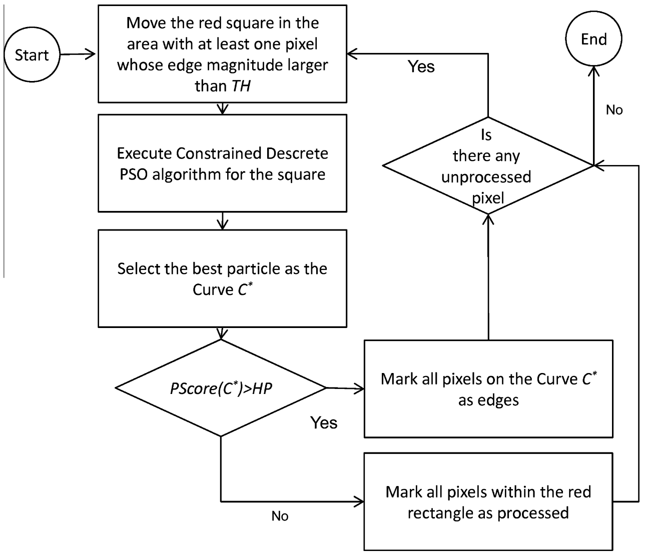
Start
Move the red square in the area with at least one pixel whose edge magnitude larger than TH
Execute Constrained Descrete PSO algorithm for the square
Select the best particle as the Curve C*
PScore(C*)>HP
Yes
No
Mark all pixels on the Curve C* as edges
Mark all pixels within the red rectangle as processed
Is there any unprocessed pixel
Yes
No
End

Fig. 4. The flowchart of the PSO-based algorithm.

$$o_i = \begin{cases} (\lfloor o_i \rfloor + 1) & \text{if } o_i - \lfloor o_i \rfloor > R \\ \lfloor o_i \rfloor & \text{otherwise} \end{cases}$$
$$m_i = \begin{cases} (m_i + 1)\ mod8 & \text{if } m_i - \lfloor m_i \rfloor > R \\ m_i\ mod8 & \text{otherwise} \end{cases} \tag{25}$$

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.

4.2. Preservation of feasible continuous edges

Algorithm 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. In this method, the feasibility of each particle is examined in each iteration. If it violates the constraints, it is replaced with a new feasible one.

Algorithm 1. PSO-based edge detection algorithm based on a preservation method to handle the constraints (PSO1)

```
1:  for all pixel $P$ on an image with local edge magnitude larger than $TH$ do
2:    if $P$ is unprocessed and not marked as an edge then
3:      Initialize PSO population in feasible search space randomly for pixel $P$
4:      repeat
5:        for all Particle decoded as curve $C$ in Population do
6:          Evaluate $U(C)$ (19), $PScore(C)$ (20) and $CCost(C)$ (22)
7:          Evaluate $Fitness(C)$ (23)
8:          if $Fitness(C)$ is better than best fitness value in history and $C$ is feasible then
9:            Update personal best position
10:         end if
11:       end for
12:       Assign the best particle in the population to the leader
13:       for all particle decoded as curve $C$ in population do
14:         Calculate particle velocity (2)
15:         Update particle position (1) and apply update rule (25)
```

(continued on next page)

دانلود کننده مقالات علمی freepapers.ir FREE papers

```
16:            **if** $Cross(C) \neq 0$ or $PScore(C) \leqslant HP$ **then**
17:              Replace the particle with a new random feasible one
18:            **end if**
19:          **end for**
20:        **until** maximum iterations exceeded or minimum error criteria attained
21:        Select best feasible particle in the population and decode it as curve $C^*$
22:        Mark all pixels on curve $C^*$ as an edge
23:        **if** no feasible particle found **then**
24:          Mark all pixels within the red rectangle as processed
25:        **end if**
26:      **end if**
27:    **end for**
```

We expect that this algorithm based on preservation can effectively maximise the distances between pixels in the two regions (interset distance) separated by a continuous edge and minimise the distance between the pixels 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 the 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. If the constraints are violated, the particle is reinitialized at random in line 17 so that the curve C, represented by the particle, does not cross itself, i.e., $Cross(C) = 0$.

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 [8]. We define a non-stationary and multi-stage penalty fitness function adopted from [45] for edge detection to handle the two constraints as shown in Eq. (26). 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.

$$PenFit(C) = Fitness(C) - \sqrt{K}(Cross(C) + \theta(q(C))q(C)) \tag{26}$$

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 Eq. (27):

$$\theta(q(C)) = \begin{cases} 1 & \text{if } q(C) < 0.001 \\ 2 & \text{if } q(C) < 0.1 \\ 10 & \text{otherwise} \end{cases} \tag{27}$$

Algorithm 2. Constrained PSO-based edge detection algorithm based on a penalising method to handle the constraints (PSO2)

```
1:     **for all** pixel $P$ on an image with a local edge magnitude larger than $TH$ **do**
2:       **if** $P$ is unprocessed and not marked as an edge **then**
3:         Initialize PSO population randomly for pixel $P$
4:         $K = 0$
5:         **repeat**
6:           Increment $K$
7:           **for all** particle (decoded as curve $C$) **do**
8:             Evaluate $U(C)$, $PScore(C)$ and $CCost(C)$
9:             Evaluate $q(C)$ and $\theta(q(C))$
10:            Evaluate $Fitness(C)$ and $PenFit(C)$
11:            **if** $PenFit(C)$ is better than best fitness value **then**
12:              Assign $C$ to best particle
13:            **end if**
14:          **end for**
```

دانلود کننده مقالات علمی
freepapers.ir
FREE papers

```
15:         Replace the worst particle with a new random one
16:         **forall** Particle decoded as curve $C$ **do**
17:           Calculate particle velocity
18:           Update particle position and apply update rule (25)
19:         **end for**
20:       **until** maximum iterations exceeded or minimum error criteria attained
21:       Select best particle and decode it as curve $C^*$
22:       **if** $C^*$ is feasible **then**
23:         Mark all pixels on curve $C^*$ as an edge
24:       **else**
25:         Mark all pixels within red rectangle as processed
26:       **end if**
27:     **end if**
28:   **end for**
```

The second constrained discrete PSO-based algorithm utilising a penalising method is outlined in Algorithm 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 dotted red square are marked as processed pixels (lines 21–25).

5. Experimental design

To investigate the effectiveness of the new algorithms, we first compare the first algorithm (PSO1) with a modified version of Canny [11] and RRO [32] 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 [37], with an adaptive method to estimate its filter scale proposed in [24] and with a NMS technique proposed in [6] 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.

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 Fig. 5). These images were downloaded from the South Florida University database [21]. To explore the performance of the new algorithms in noisy environments, these images are corrupted by two different types of noise: impulsive and Gaussian (see the images in Fig. 5 in columns (b) and (c)). The probability of the impulsive noise is 0.1 and the peak-signal to noise ratio (PSNR) is 16 dB for the Gaussian noise in these noisy images. 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 edge maps are available from [13]. The size of each image is 256×256 pixels and the resolution of each is 8 bits per pixel. These images are shown in Fig. 6. 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 impulsive 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 22 dB with a step size of 1 dB. As the ground truth of these images are available, we used them for an objective (quantitative) comparison.

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 [47]. This measure is defined by Eq. (28).

$$R_{PFOM} = \frac{1}{max(I_I, I_A)} \sum_{i=1}^{I_A} \frac{1}{1 + \beta d(i)^2} \tag{28}$$

دانلود کننده مقالات علمی
freepapers.ir
FREE papers

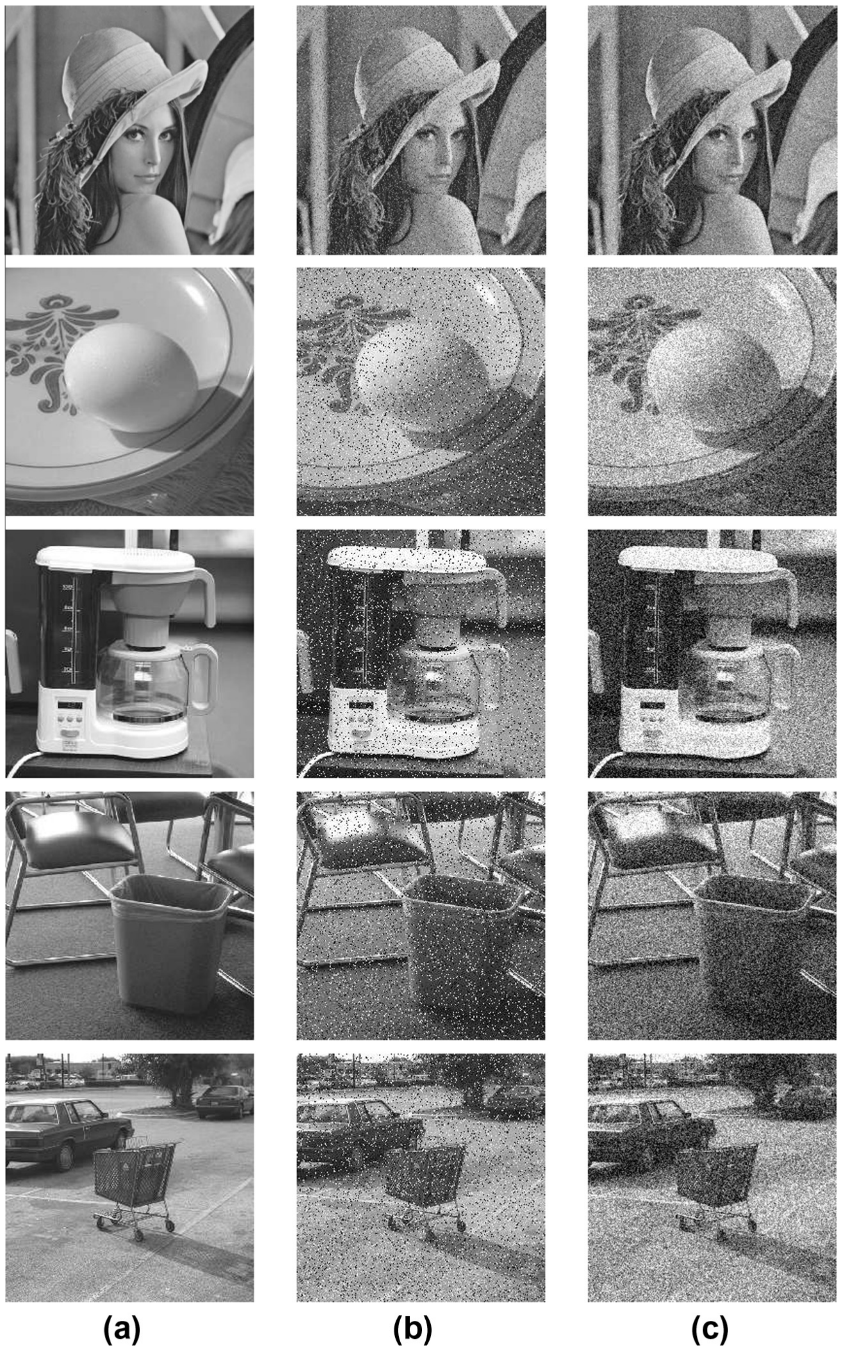

Fig. 5. Example images for subjective comparison. (a) Original images Lena, egg, coffee maker, rubbish bin and car; (b) images with *impulsive noise* (noise probability = 0.1); and (c) images with *Gaussian noise* (PSNR = 16 dB).

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.

5.3. Parameter settings

In the proposed PSO algorithms, the population size is 50 and the maximum number of iterations is 200 according to the chosen particle length. The minimum length of a continuous edge, $2L + 1$ was set at 21, *SqrSize* at 6, n at 9, $w_1 = 90$, $w_2 = 40$,

دانلود کننده مقالات علمی freepapers.ir FREE papers

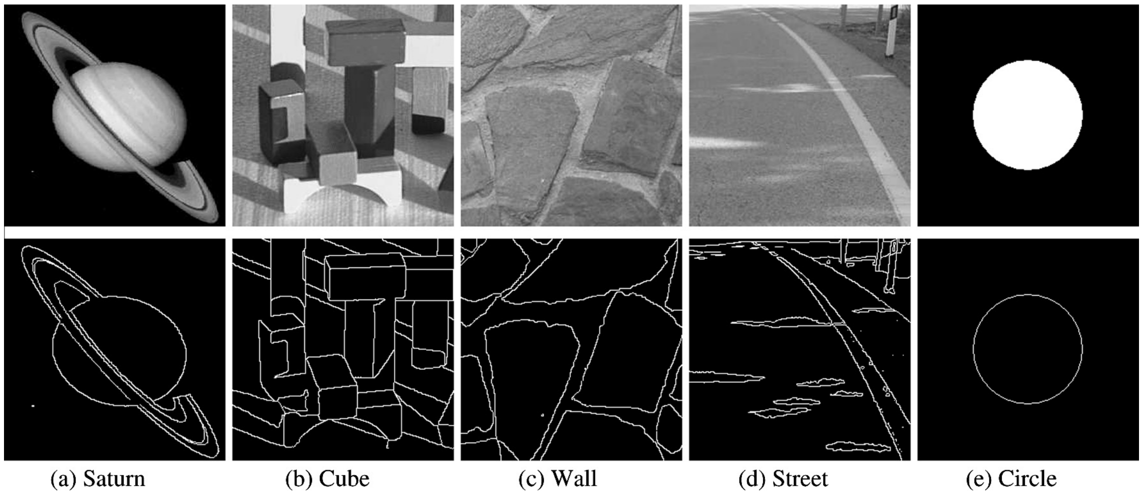

Fig. 6. Example images for objective comparision. (a)–(d) four real image from the UCO university and their manual ground truth images [13]; and (e) one synthetic circle image and its ground truth.

$w_3 = 40$, *TH* at the value estimated by the Ostu's method [42], and *HP* at 0.5. We chose the value of the weight factors (w_1, w_2 and w_3) based on the following preliminary experimentation. 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 [57] 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. Although Samal et al. [48] showed that the convergence time may be decreased in PSO and a good fitness value may also be obtained when $w = 0.6$, $c_1 = 0.103$ and $c_2 = 2.897$, we used the values $w = 0.7298$, $c_1 = 1.4962$, $c_2 = 1.4962$ for the parameters in Eq. (2) because these values are more commonly used for real and integer valued parameter optimisation problems [30,59]. We also chose the fully connected graph (gbest) as a swarm topology to select the leader particle in PSO because of its speed and the ease of its implementation.

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: $highthreshold = 2\sigma\sqrt{f_u ln2}, lowthreshold = \frac{1}{2}highthreshold$ 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 [37], the filter size (σ) was set at the value estimated from the approach proposed by Jeong and Kim [24]. The edge-height parameter of the RRO detector, which defines the minimum grey-level differential across an edge, was set to the value which gave a highest PFOM value.

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.

6.1. Subjective/qualitative comparison

The resulting images are shown in Figs. 7 and 8 after applying the Canny [11], the RRO [32] and PSO1 on the images in the first set (Fig. 5) corrupted by impulsive and Gaussian noises respectively.

The resulting images in Fig. 7 show that PSO1 performed better than the other two algorithms on the five images with *impulsive noise* at a noise probability level of 0.1. The Canny algorithm, even with preprocessing (Gaussian filter and NMS) and post-processing (hysteresis thresholding as an edge linking technique), 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 impulsive 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 Fig. 7, 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

دانلود کننده مقالات علمی
freepapers.ir
FREE papers

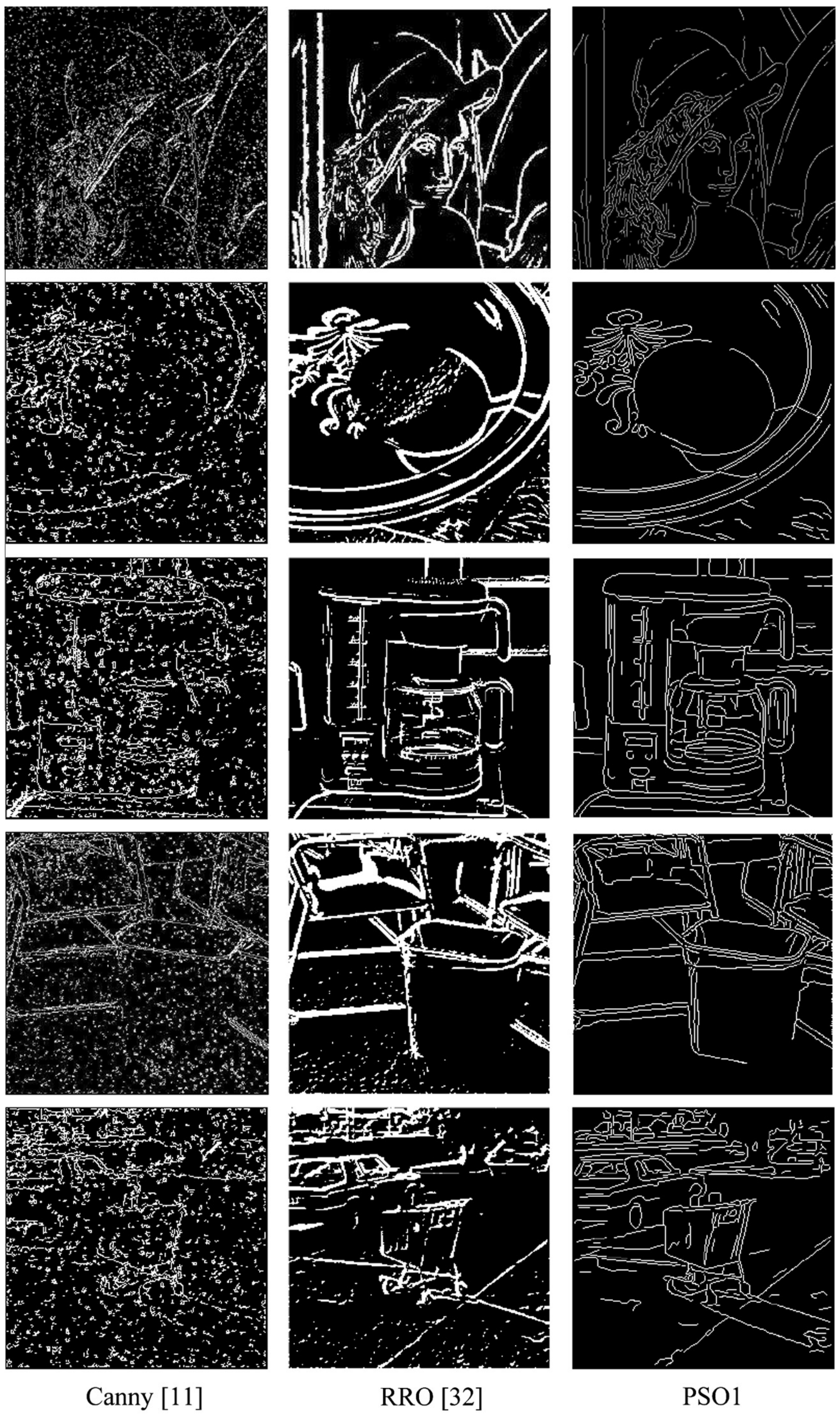

Fig. 7. Subjective results of edge detection produced by three algorithms on the five images corrupted by *impulsive noise* (noise probability = 0.1).

significantly improved in comparison with the other two algorithms. For the egg image, RRO detected some false edges on the surface of the egg and also there are 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

دانلود کننده مقالات علمی freepapers.ir FREE papers

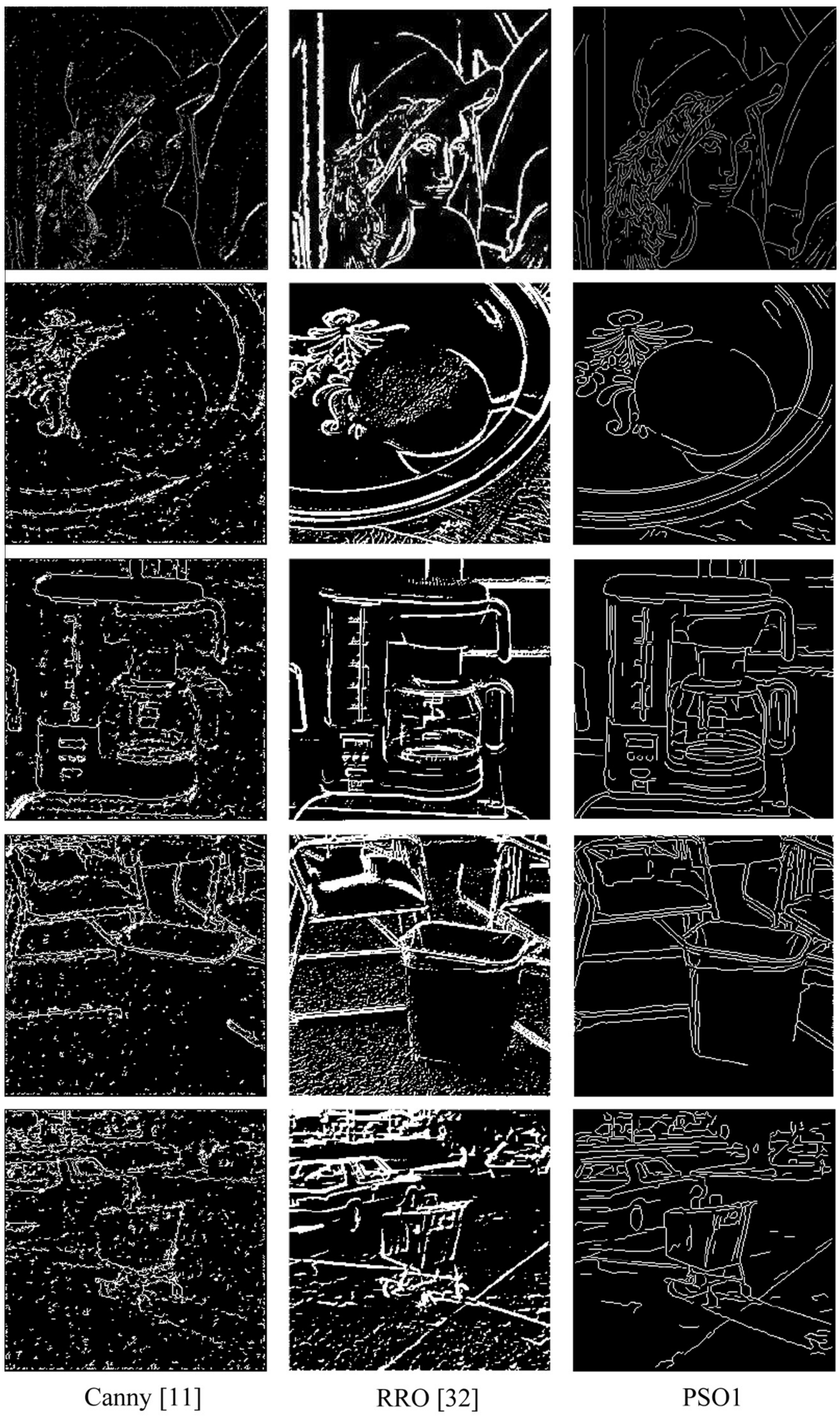

Fig. 8. Subjective results of edge detection produced by the three algorithms on the five images with *Gaussian noise* (PSNR = 16 dB).

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 Fig. 8 after applying the three algorithms on the five noisy images corrupted by Gaussian noise (PSNR = 16 dB). The comparison of these results with those for the impulsive 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 impulsive noise. The edges detected by RRO and PSO1 for these images have very similar quality to those with the impulsive noise, although PSO1 recognised edges that are more continuous and smoother than Canny, and also detected edges much thinner than RRO.

دانلود کننده مقالات علمی freepapers.ir FREE papers

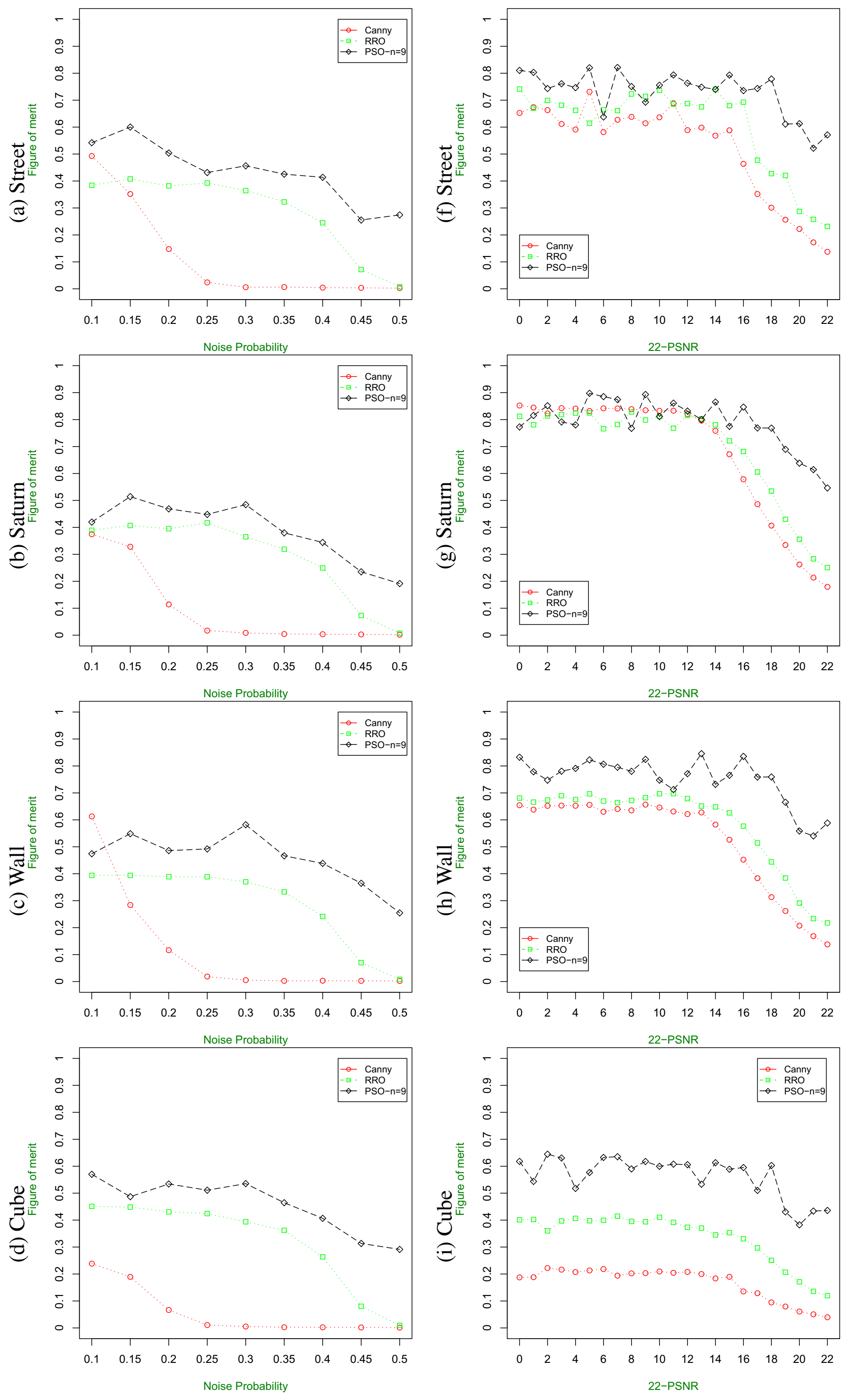

Fig. 9. PFOM for the street, Saturn, wall and circle images in the second image set. (a)–(e): with different impulsive noise levels (the noise probability ranging from 0.1 to 0.5); and (f)–(j) with different Gaussian noise levels (PSNR ranging from 0 to 22 dB) (Part I).

دانلود کننده مقالات علمی freepapers.ir FREE papers

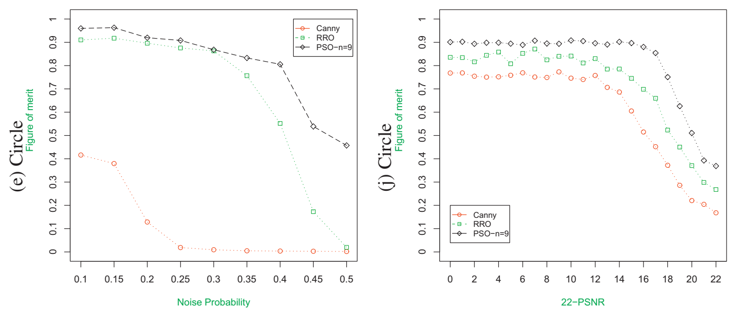

Fig. 9 (*continued*)

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 (Fig. 6) at different noise levels. The PFOM values are plotted at 11 different Gaussian noise levels and 9 impulsive noise levels as depicted in Fig. 9. PSNR ranges from 0 to 22 dB with step of 1 dB and the noise probability ranges from 0.1 to 0.5 with step of 0.05. The number of pixels in each set in Eqs. (14) and (15), 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 Fig. 9, 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 impulsive noise in most cases (see Fig. 9a–e). These resulting plots also illustrate that PSO1 outperformed RRO in the images with impulsive and Gaussian noises in most cases, however its performance is lower than RRO in a few cases in the images corrupted by Gaussian noise (see Fig. 9f). This suggests that PSO1 is less sensitive to Gaussian noise and impulsive noise than RRO. Canny is more sensitive to impulsive 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.

6.3. Discussion on parameter values

As already described, PSO1 has several additional parameters in comparison with Canny and RRO. The parameters include w_1, w_2, w_3, n, *SqrSize* and L as described in Section 5.3. Variation of the values of the parameters n, *SqrSize* and L 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.

Fig. 10 shows the resulting images after applying PSO1 with different parameters on the rubbish bin and egg images corrupted by impulsive noise whose probability is 0.1. As can be seen in the original images in Fig. 5, 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 Fig. 10a are the output of PSO1 with $L = 10$ and $SqrSize = 6$. The resulting images in Fig. 10b 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 Fig. 10c 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 which is the number of the neighbour pixels in Eqs. (14) and (15) on the algorithm's accuracy, the localisation accuracy (PFOM) at different noise levels was calculated from the resulting images after applying PSO1. The value of parameter n was set at 6 (see the edge partitions in Fig. B.1 in B) and 9 (see the edge partitions in Fig. 2) 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 Fig. 11. The plots shows that when $n = 9$, the algorithm's accuracy is higher than when $n = 6$ in most cases.

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.

دانلود کننده مقالات علمی freepapers.ir FREE papers

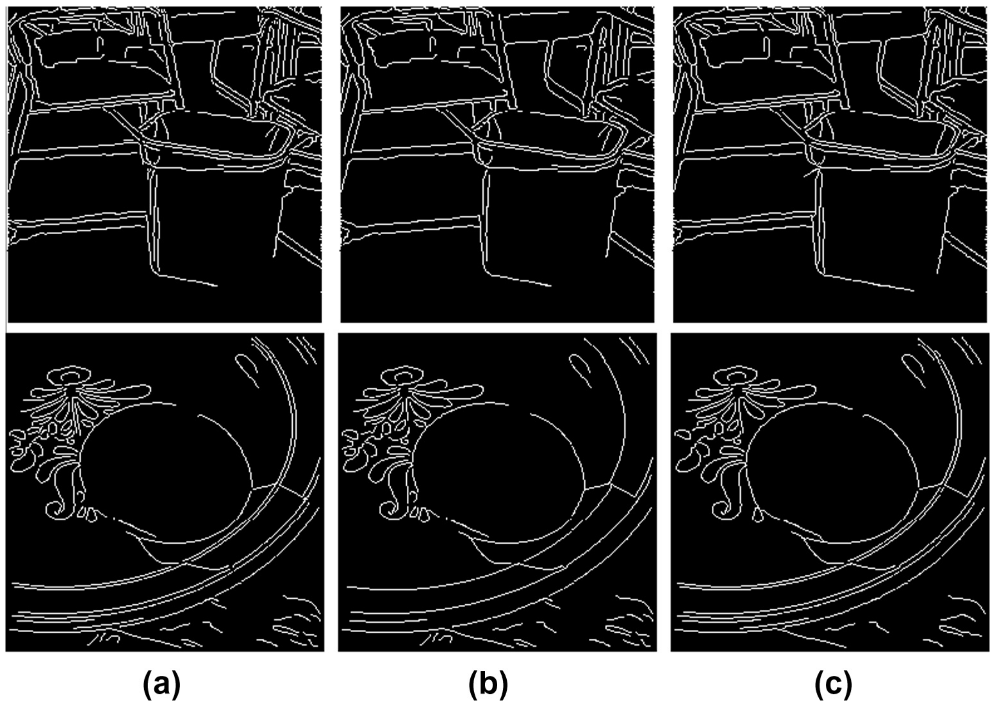

Fig. 10. Resulting images after applying PSO1 with different parameter values (a) $L = 10$ and $SqrSize = 6$, (b) $L = 10$ and $SqrSize = 10$ (c) $L = 15$ and $SqrSize = 6$.

Table 1 shows PFOM estimated from the resulting images after applying Canny, RRO, PSO1 and PSO2. G6, G10, G14, G18 and G22 represent PSNR from 6 dB to 22 dB 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 impulsive 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 the 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 by symbol $*$ in Table 1 columns PSO1 and PSO2. However, the accuracy standard deviation of the second algorithm is lower than that of the first one; this implies that the second algorithm is more stable than

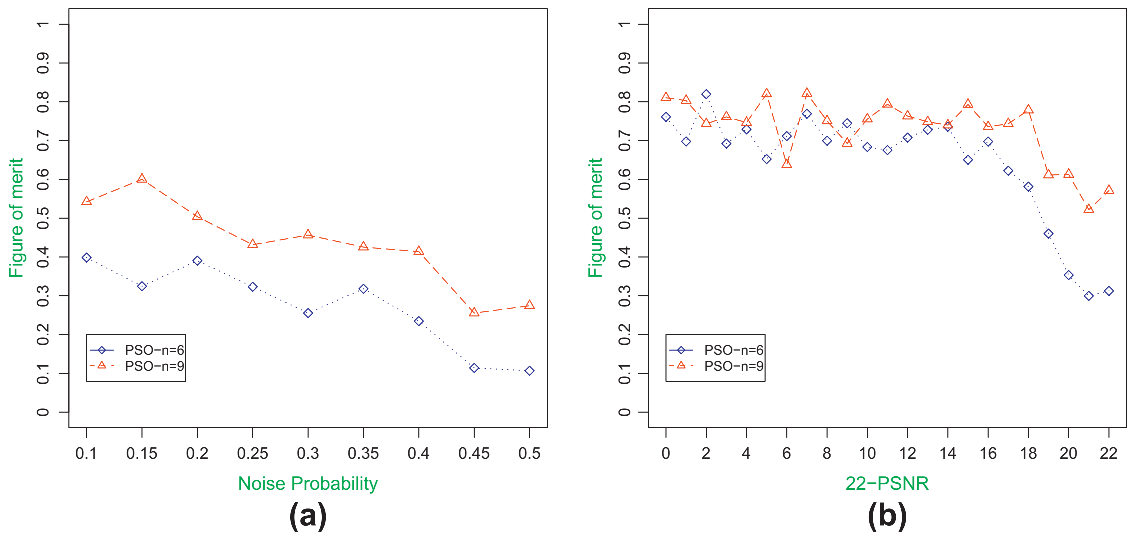

Fig. 11. Comparing PFOM for the street image in the second image set when $n = 6$ or $n = 9$ (a) with different impulsive noise levels (the noise probability ranging from 0.1 to 0.5 the impulsive noise) (b) with different Gaussian noise levels (PSNR ranging from 0 to 22 dB). Note horizontal axis is 22-PSNR.

دانلود کننده مقالات علمی freepapers.ir FREE papers

the first one. Table 1 also shows that both PSO1 and PSO2 are significantly better than Canny and RRO in most cases except a few cases as shown by symbol △ in Table 1 columns Canny and RRO.

Table 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 the null hypothesis is rejected in all cases. In fact, the number of fitness function evaluations of PSO1 is 23–45% greater than PSO2. It implies that PSO1 is slower than PSO2. The results in this table also shows that the execution time of the both algorithms is reduced when the level of the noise is increased.

Table 1
Comparison of *accuracy* of two proposed Algorithms 1 and 2 with Canny and RRO.

Image	Noise level	95% Confidence interval for accuracy		Standard deviation		Canny	RRO
		PSO1	PSO2	PSO1	PSO2		
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

دانلود کننده مقالات علمی freepapers.ir FREE papers

Table 2
Comparison of *number of fitness function evaluations* of two proposed algorithms.

Image	Gaussian Noise	Number of fitness PSO1	Function evaluation PSO2	Impulsive Noise	Number of fitness PSO1	Function evaluation 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

The execution time of PSO1 is usually between 50 and 70 s and that of PSO2 is between 40 and 50 s for these images depending on the noise level on a machine with 2.6 GHz CPU and 4 GB RAM. These are considerably longer than RRO and Canny. We will investigate new ways of reducing the computation cost in the future.

7. Conclusions

The main goal of this paper was the reduction of the broken and jagged edges in noisy images by proposing an optimisation model 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 the broken and jagged edges, a new fitness function based on the possibility score of a curve being fitted on an edge and the curvature cost of the curve with two constraints was developed. Two different methods were proposed to handle the constraints. 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 the Canny and RRO algorithms based on PFOM as a measure of efficiency.

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 impulsive noise. Although the execution time of the new algorithms are 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.

In comparison with the other deterministic algorithms, the new approach has many parameters such as the size of the square, the minimum length of the particle, and the weights of different factors in the fitness function. Our experiments show that there may be a relationship between these parameters especially between first two parameters. The size of the square and the minimum length of the particle affect the accuracy and the speed of the algorithms and they will be analysed more precisely in the future. Furthermore, as a future research direction, the initialization of particles with more diversity, which may help the PSO convergence, can be investigated. In this paper, we used a fully connected graph as the neighbourhood topology in all our experiments. Since choosing an ideal topology requires complete experimentation, we will investigate the influence of the different topologies on the accuracy and execution time of the new algorithms. The influence of using local thresholding techniques instead of using global ones will also be investigated.

Appendix A. Sigmoid function

The general form of the sigmoid function is formulated as:

دانلود کننده مقالات علمی freepapers.ir FREE papers

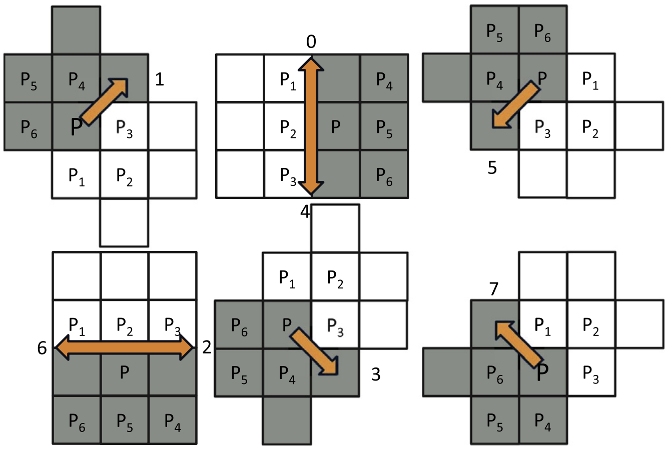

Fig. B.1. Six ways of moving from pixel P to a neighbouring pixel when $n = 6$.

$$Y = \frac{\alpha_1}{1 + e^{-\alpha_2(x-\alpha_3)}}$$

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.,

$$TH = \alpha_3 + 1.317/\alpha_2$$
$$SAT = \alpha_3 - 1.317/\alpha_2$$

where TH and SAT are the threshold and saturation points for the sigmoid function. Therefore,

$$\alpha_2 = \frac{2.634}{TH - SAT}$$
$$\alpha_3 = \frac{SAT + TH}{2}$$

Appendix B. Edge partitions when $n = 6$

For $n = 6$, we considered six ways of dividing a neighbourhood of a pixel into two sub-regions in our experiments as follows: (see Fig. B.1).

References

[1] M.R. Al Rashidi, M.E. El-Hawary, A survey of particle swarm optimization applications in electric power systems, Transactions on Evolutionary Computation 13 (2009) 913–918.
[2] A. Baştürk, E. Günay, Efficient edge detection in digital images using a cellular neural network optimized by differential evolution algorithm, Expert Systems with Applications 36 (2009) 2645–2650.
[3] M. Basu, Gaussian-based edge-detection methods: a survey, IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 32 (2002) 252–260.
[4] A.C. Bovik, T.S. Huang, D.C.J. Munson, Nonparametric tests for edge detection in noise, Pattern Recognition 19 (1986) 209–219.
[5] M. Breaban, M. Ionita, C. Croitoru, A new PSO approach to constraint satisfaction, in: Proceedings of the IEEE Congress on Evolutionary Computation, IEEE Press, 2007, pp. 1948–1954.
[6] J. Canny, A computational approach to edge detection, IEEE Transactions on Pattern Analysis and Machine Intelligence 8 (1986) 679–698.
[7] G. Chen, Y. Hong Yang, Edge detection by regularized cubic b-spline fitting, IEEE Transactions on Systems, Man and Cybernetics 25 (1995) 636–643.
[8] G. Coath, S. Halgamuge, A comparison of constraint-handling methods for the application of particle swarm optimization to constrained nonlinear optimization problems, Proceedings of the IEEE Congress on Evolutionary Computation, vol. 4, IEEE Press, 2003, pp. 2419–2425.
[9] G.W. Cook, E.J. Delp, Multiresolution sequential edge linking, Proceedings of the 1995 International Conference on Image Processing, vol. 1, IEEE Computer Society, 1995, pp. 41–44.
[10] R. Deriche, Using Canny's criteria to derive a recursively implemented optimal edge detector, International Journal of Computer Vision 1 (1987) 167–187–187.
[11] L. Ding, A. Goshtasby, On the Canny edge detector, Pattern Recognition 34 (2001) 721–725.
[12] A.A. Farag, E.J. Delp, Edge linking by sequential search, Pattern Recognition 28 (1995) 611–633.
[13] N.L. Fernández-García, A. Carmona-Poyato, R. Medina-Carnicer, F.J. Madrid-Cuevas, Images from Automatic Generation of Consensus Ground Truth for Comparison of Edge Detection Techniques, 2008. <http://www.uco.es/ma1fegan/Comunes/investigacion/imagenes/ground-truth.html>.
[14] J. Ferwerda, Elements of early vision for computer graphics, IEEE Computer Graphics and Applications 21 (2001) 22–33.

دانلود کننده مقالات علمی freepapers.ir FREE papers

[15] M. Fesharaki, 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), IEEE Press, 1994, pp. 21–24.
[16] J. Fuentes Cabrera, C. Coello Coello, Handling constraints in particle swarm optimization using a small population size, in: MICAI 2007: Advances in Artificial Intelligence, Lecture Notes in Computer Science, vol. 4827, Springer, Berlin/Heidelberg, 2007, pp. 41–51.
[17] K. Ghosh, S. Sarkar, K. Bhaumik, A possible mechanism of zero crossing detection using the concept of the extended classical receptive field of retinal ganglion cells, Biological Cybernetics 93 (2005) 1–5.
[18] R.C. Gonzalez, R.E. Woods, Digital Image Processing, Prentice Hall, 2007.
[19] J.W. Han, J.H. Kim, S.H. Cheon, J.O. Kim, S.J. Ko, A novel image interpolation method using the bilateral filter, IEEE Transactions on Consumer Electronics 56 (2010) 175–181.
[20] N. Hautiere, J.P. Tarel, R. Bremond, Perceptual hysteresis thresholding: towards driver visibility descriptors, in: Proceedings of the IEEE International Conference on Intelligent Computer Communication and Processing, IEEE Press, 2007, pp. 89–96.
[21] M. Heath, S. Sarkar, T. Sanocki, K. Bowyer, Images from South Florida University for Edge Detector Comparison, 1998. <http://marathon.csee.usf.edu/edge/edge_detection>.
[22] L. Hu, H. Cheng, M. Zhang, A high performance edge detector based on fuzzy inference rules, Information Sciences 177 (2007) 4768–4784.
[23] X. Hu, R. Eberhart, Solving constrained nonlinear optimization problems with particle swarm optimization, in: Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI), International Institute of Informatics and Systemics, 2002, pp. 203–206.
[24] H. Jeong, C. Kim, Adaptive determination of filter scales for edge detection, IEEE Transactions on Pattern Analysis and Machine Intelligence 14 (1992) 579–585.
[25] A. Jevtic, I. Melgar, D. Andina, Ant based edge linking algorithm, in: Proceedings of 35th Annual Conference of IEEE Industrial Electronics (IECON), IEEE Press, 2009, pp. 3353–3358.
[26] W. Jiang, K.M. Lam, T.Z. Shen, Efficient edge detection using simplified gabor wavelets, IEEE Transactions on Systems, Man, and Cybernetics, Part B 39 (2009) 1036–1047.
[27] A.H. Kashan, B. Karimi, A discrete particle swarm optimization algorithm for scheduling parallel machines, Computers and Industrial Engineering 56 (2009) 216–223.
[28] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of the IEEE International Conference on Neural Networks, IEEE Press, 1995, pp. 1942–1948.
[29] S. Konishi, A. Yuille, J. Coughlan, S.C. Zhu, Statistical edge detection: learning and evaluating edge cues, IEEE Transactions on Pattern Analysis and Machine Intelligence 25 (2003) 57–74.
[30] E. Laskari, K. Parsopoulos, M. Vrahatis, Particle swarm optimization for integer programming, Proceedings of the Congress on Evolutionary Computation, vol. 2, IEEE Press, 2002, pp. 1582–1587.
[31] R. Lee, J. Lin, An elastic contour matching model for tropical cyclone pattern recognition, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 31 (2001) 413–417.
[32] D.H. Lim, Robust edge detection in noisy images, Computational Statistics and Data Analysis 50 (2006) 803–812.
[33] T. Lindeberg, Scale-Space Theory in Computer Vision, Kluwer Academic Publishers., Norwell, MA, USA, 1994.
[34] D.S. Lu, C.C. Chen, Edge detection improvement by ant colony optimization, Pattern Recognition Letters 29 (2008) 416–425.
[35] L. MacEachern, T. Manku, Genetic algorithms for active contour optimization, Proceedings of the 1998 IEEE International Symposium on Circuits and Systems, vol. 4, IEEE Press, 1998, pp. 229–232.
[36] D. Marr, E. Hildreth, Theory of edge detection, Proceedings of the Royal Society of London Series B 207 (1980) 187–217.
[37] R. Medina-Carnicer, F. Madrid-Cuevas, A. Carmona-Poyato, R. Muoz-Salinas, On candidates selection for hysteresis thresholds in edge detection, Pattern Recognition 42 (2009) 1284–1296.
[38] E. Mendel, R.A. Krohling, M. Campos, Swarm algorithms with chaotic jumps applied to noisy optimization problems, Information Sciences 181 (2011) 4494–4514.
[39] C. Monson, K. Seppi, Linear equality constraints and homomorphous mappings in PSO, Proceedings of the IEEE Congress on Evolutionary Computation, vol. 1, IEEE Press, 2005, pp. 73–80.
[40] A. Nakib, H. Oulhadj, P. Siarry, Fractional differentiation and non-pareto multiobjective optimization for image thresholding, Engineering Applications of Artificial Intelligence 22 (2009) 236–249.
[41] J. Nascimento, J. Marques, Adaptive snakes using the EM algorithm, IEEE Transactions on Image Processing 14 (2005) 1678–1686.
[42] N. Otsu, A threshold selection method for gray level histograms, IEEE Transaction on Systems, Man, Cybernetics 9 (1976) 62–66.
[43] H. Pan, L. Wang, B. Liu, Particle swarm optimization for function optimization in noisy environment, Applied Mathematics and Computation 181 (2006) 908–919.
[44] K.E. Parsopoulos, 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.
[45] K.E. Parsopoulos, 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.
[46] T. Poggio, H. Voorhees, A. Yuille, A regularized solution to edge detection, Journal of Complexity 4 (1988) 106–123.
[47] W. Pratt, Digital Image Processing: PIKS Scientific Inside, Wiley Interscience, 2007.
[48] N. Samal, A. Konar, S. Das, A. Abraham, A closed loop stability analysis and parameter selection of the particle swarm optimization dynamics for faster convergence, in: IEEE Congress on Evolutionary Computation, pp. 1769–1776.
[49] K. Sedlaczek, P. Eberhard, Optimization of nonlinear mechanical systems under constraints with the particle swarm method, in: Proceeding of Applied Mathematics and Mechanics, John Wiley and Sons, 2004, pp. 169–170.
[50] D. Sen, S.K. Pal, Gradient histogram: thresholding in a region of interest for edge detection, Image and Vision Computing 28 (2010) 677–695.
[51] M. Setayesh, M. Zhang, 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, ACM Press, 2011, pp. 45–52.
[52] M. Setayesh, M. Zhang, 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.
[53] K.S. Shanmugam, F.M. Dickey, J.A. Green, An optimal frequency domain filter for edge detection in digital pictures, IEEE Transactions on Pattern Analysis and Machine Intelligence 1 (1979) 37–49.
[54] M. Sharifi, M. Fathy, M.T. Mahmoudi, A classified and comparative study of edge detection algorithms, in: Proceedings of the International Conference on Information Technology: Coding and Computing, IEEE Press, 2002, pp. 117–120.
[55] Y. Shi, R. Eberhart, Comparing inertia weights and constriction factor in particle swarm optimisation, in: Proceedings of the IEEE Congress on Evolutionary Computation, IEEE Press, 2000, pp. 84–89.
[56] B. Tremblais, B. Augereau, A fast multiscale edge detection algorithm based on a new edge preserving PDE resolution scheme, Proceedings of the 17th International Conference on Pattern Recognition, vol. 2, IEE Computer Society, 2004, pp. 811–814.
[57] S.E. Umbaugh, Computer Imaging: Digital Image Analysis and Processing, CRC Press, 2005.
[58] J. Vesterstrom, R. Thomsen, A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems, Proceedings of the IEEE Congress on Evolutionary Computation, vol. 2, IEEE Press, 2004, pp. 1980–1987.
[59] J. Wang, Z. Kuang, X. Xu, Y. Zhou, Discrete particle swarm optimization based on estimation of distribution for polygonal approximation problems, Expert Systems with Applications 36 (2009) 9398–9408.

[60] A.P. Witkin, Scale-space filtering, in: Proceedings of the Eighth International Joint Conference on Artificial Intelligence, IJCAI'83, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1983, pp. 1019–1022.
[61] J. Zhao, Z. Li, Particle filter based on particle swarm optimization resampling for vision tracking, Expert Systems with Applications 37 (2010) 8910–8914.
[62] Q. Zhu, Efficient evaluations of edge connectivity and width uniformity, Image and Vision Computing 14 (1996) 21–34.

FREE papers
freepapers.ir دانلود کننده مقالات علمی