Detection of Continuous, Smooth and Thin Edges in Noisy Images Using Constrained Particle Swarm Optimisation

Mahdi Setayesh
School of Engineering and Computer Science

Mengjie Zhang
School of Engineering and Computer Science

Mark Johnston
School of Mathematics, Statistics and Operations Research

Victoria University of Wellington, New Zealand
{mahdi.setayesh, mengjie.zhang}@ecs.vuw.ac.nz and mark.johnston@msor.vuw.ac.nz

ABSTRACT

Detecting continuous edges is a hard problem especially in noisy images. We propose an algorithm based on particle swarm optimisation (PSO) to detect continuous and smooth edges in such images. A constrained PSO-based algorithm with a new penalised objective function and two constraints is proposed to overcome noise and reduce broken edges. The new algorithm is examined and compared with a modified version of the Canny algorithm, the robust rank order (RRO)-based algorithm, and an existing PSO-based algorithm on two sets of images with different types and levels of noise. The results suggest that the new algorithm detect edges more accurately than these three algorithms and the detected edges are smoother than those detected by the previous PSO algorithm and thinner than those detected by RRO.

Categories and Subject Descriptors

I.4 [**Image Processing and Computer Vision**]: Miscellaneous; G.1.6 [**Optimization**]: Constrained optimization

General Terms

Algorithms

Keywords

Image processing, edge detection, particle swarm optimisation

1. INTRODUCTION

The initial input of a computer vision system is usually an entire image, which typically includes a large number of pixels to be processed. Therefore, meaningful features need to be extracted from the image to reduce the amount of the data. These features could be in low and high levels. Low level features are extracted directly from the image, e.g., edges, corners, blobs and ridges [30], while high level features are usually extracted from low level ones. The detection of exact edges is an essential part of such image processing algorithms [11]. In theory, the output of an ideal edge detection algorithm is the continuous contours of the object boundaries. Accurately detecting continuous contours is very hard and time consuming, and the task is even harder with noisy images [30].

The commonly used algorithms for detecting edges in noisy images are Gaussian-based [4], statistical-based [17], and scale space-based [29] edge detectors. The Gaussian-based algorithms malfunction at corners and curves [28] and establish double edges in the area with high frequencies of information. They also displace and produce false edges [4]. These methods use a Gaussian filter as a smoothing technique to reduce noise, which often causes edges to be weak and broken as a side effect [9].

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 [17]. 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.

Another group of algorithms use the scale space theory [18] to generate different scales of an image and produce the 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 applications. These methods also suffer from producing broken edges [4].

Several techniques have been proposed to compensate for broken edges, such as sequential edge linking (SEL) [10], multi-resolution SEL (M-SEL) [8] and the Hough transform [12]. Their simplicity and high speed are the main advantages, but they are not 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, circles), it cannot deal with objects with complicated shapes [19].

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 [15]. Compared with some heuristic methods such as genetic algorithms, the most important advantages of PSO are ease of its implementation, fewer operators, a limited memory for each particle and high speed of convergence [3]. As PSO has a high capability to optimise noisy functions [14][22], it has been success-

GECCO'11, July 12–16, 2011, Dublin, Ireland.

fully applied to many problems in noisy environments, such as image segmentation [21] and vision tracking [32]. Surprisingly, PSO has not been sufficiently analysed for tackling the edge detection problem.

This paper aims to develop a new PSO based approach to edge detection in noisy images with the goal of extracting continuous edges and reducing broken and jagged edges. The new algorithm will be examined and compared to the modified version of Canny proposed in [9], RRO proposed in [17] and a previous PSO algorithm [27] on two sets of noisy images. The localisation accuracy will be used to measure the performance of different algorithms.

The rest of the paper is organised as follows. Section 2 provides a brief overview of the edge detection and background information on PSO. The new PSO-based edge detection algorithm will be described in Section 3. Sections 4 and 5 presents the experiments and the discussions of the results, respectively. Section 6 draws concluding remarks.

2. BACKGROUND

2.1 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 [17]. Although many algorithms have been proposed to detect edges in noisy images, this section only briefly reviews a modified version of Canny [9] and RRO [17] as they are well-known and will be used in this paper. 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 of having a complete process of edge detection and its good localisation. This edge detector has been revised many times since it was 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 non-maxima suppression (NMS) algorithm to suppress non-maxima edges, and applying a hysteresis thresholding technique to identify edges and linking broken edges.

Many edge detection algorithms have been proposed to deal with noise in the framework of statistics. An algorithm was recently developed based on the RRO test [17]. This algorithm operates better than other statistical-based edge detectors such as Wilcoxon and t test-based edge detectors. The RRO algorithm considers eight different edge patterns for each pixel, each of which partitions the neighbourhood of the pixel into two regions. Then the RRO edge detector tests whether there is a significant difference between the average intensities of the two regions. If a significant difference occurs, the centered pixel is considered an edge pixel; otherwise a non-edge pixel.

2.2 Particle Swarm Optimisation

PSO is a branch of swarm intelligence and inspired by the social behavior of animals and biological populations and simulates a simplified social model such as flocking of birds and schooling 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 [31].

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 $\vec{X}_i(t) = (x_{i1}(t), x_{i2}(t), ..., x_{in}(t))$, where i is the index of the particle in the population. Its location is updated according to its own experience (particle and memory influence) and that of its neighbours (swarm influence). Each particle has a velocity represented by $\vec{V}_i(t)$ that is added to $\vec{X}_i(t)$ at each iteration of PSO as in Equation (1).

$$\vec{X}_i(t+1) = \vec{X}_i(t) + \vec{V}_i(t+1). \quad (1)$$

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

$$\vec{V}_{i,j}(t+1) = w\vec{V}_{i,j}(t) + C_1 r_1(\vec{X}_{pbest_i,j} - \vec{X}_{i,j}(t)) \quad (2)$$
$$+C_2 r_2(\vec{X}_{leader,j} - \vec{X}_{i,j}(t))$$

where j shows the index of jth element of the corresponding vector; r_1, and r_2 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; $\vec{X}_{pbest_i}$ denotes the best position of ith particle so far; and $\vec{X}_{leader}$ 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 can fall within a feasible search space. These methods typically utilise a particular operator to preserve new solutions to violate existing constraints [13]. In the second group, the algorithms give a penalty to the fitness of the particles which violate constraints [24]. The third group (partitioning methods) divides all particles into a feasible set and an infeasible set that are operated differently. Some of them manipulate and mend infeasible solutions or prioritise solutions based on their feasibilities [5]. In the last category, the optimisation problem is transferred to another one such that either the constraints can be handled by an easier way, or they can be eliminated. An example is using homomorphous mappings in a problem with linear equality constraints [20].

3. THE NEW APPROACH

Two PSO-based algorithms with different encoding schemes and fitness functions were previously applied to noisy binary images containing simple shapes, such as rectangles, squares, circles, crosses and triangles [26][25]. Their performances were acceptable in the binary images but they were inefficient and did not operate well on non-binary images. Another recent PSO-based algorithm could detect continuous edges in noisy images [27], but it produced jagged edges and its overall performance was worse than RRO in noisy images. This section describes a new constrained PSO-based algorithm with a new penalised fitness function to detect continuous edges in grey-level noisy images.

3.1 Encoding Scheme

An edge detection algorithm needs to extract the global structure of edges to compensate for broken edges. It also requires consideration of a large area to deal with noise. We developed an encoding scheme for the particles where each particle represents the global structure of a continuous edge. This edge divides an area into two regions, the light and dark regions as shown in Figure 1(b), with the goal of maximising **interset** distance between the average pixel intensities of two regions and minimises **intraset** distances within both regions.

In the proposed algorithm, a continuous edge is encoded into a particle as $\langle\langle o_1, o_2\rangle, \langle m_1, m_2, \ldots, m_{max/2}\rangle, \langle m_{max/2+1}, \ldots, m_{max}\rangle\rangle$, where $max + 1$ is the number of pixels on the edge

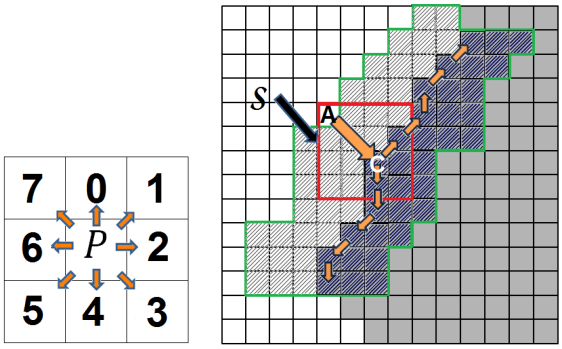


Figure 1: The particle encoding scheme. (a) Eight movement directions from a pixel P; (b) an example of a curve with two regions; (c) the particle representing the curve with $max = 10$.

curve. The encoding scheme has three parts: the offsets of the closest edge to each pixel of the image ($\langle o_1, o_2 \rangle$) and two sets of movement direction sequences from the pixel ($\langle m_1, m_2, \ldots, m_{max/2} \rangle$ and $\langle m_{max/2+1}, m_{max/2+2}, \ldots, m_{max} \rangle$). The values of two offsets (o_1 and o_2) are integers ranging from 0 to $RectSize - 1$ and m_i ranging from 0 to 7. m_i shows the direction of the movement from a pixel to one of the eight possible adjacent pixels in its neighbourhood along the continuous edge as shown in Figure 1(a). For example, the edge passing through pixel C in the neighborhood of the pixel A in Figure 1(b) is encoded as shown in Figure 1(c). (o_1, o_2) shows the offsets of pixel C from the pixel A. $RectSize$ is a parameter to show the size of the neighbourhood as represented by the square S in Figure 1(b). The value of $RectSize$ depends on the image resolution. In this example, $RectSize = 4$ and m_1, m_2, $\ldots$, m_5 show the movement directions from the point C toward top and $m_6, m_7, \ldots, m_{10}$, to the down side.

During the evolutionary process, a number of $RectSize \times RectSize$ pixels will be processed. If the best curve is found by the process, the pixels on the curve are marked as edges and the pixels within the $RectSize \times RectSize$ square (as shown by the square S in Figure 1(b)) will be marked as unprocessed pixels and will be considered in the next iteration; otherwise all pixels in this area will be marked as the processed pixels and will not be considered any more.

3.2 Truncation Method for Discrete PSO

As the search space explored by the new PSO-based algorithm is discrete, the particle positions must be truncated to integer numbers after they are updated by Equation (1). Many discrete versions of PSO use a simple truncation method to convert real numbers to integers. Instead of using a simple truncation method, we use the following method to truncate the values of particle positions to integer numbers:

$$o_i = \begin{cases} (\lfloor o_i \rfloor + 1) & , \quad o_i - \lfloor o_i \rfloor > R \\ \lfloor o_i \rfloor & , \quad otherwise \end{cases} \tag{3}$$

$$m_i = \begin{cases} (m_i + 1) \bmod 8 & , \quad m_i - \lfloor m_i \rfloor > R \\ m_i \bmod 8 & , \quad otherwise \end{cases}$$

where R is a uniform random number ranging from 0 to 1. 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.

3.3 Penalised Fitness Function

In this section, a new fitness function with a curvature factor and two different constraints are proposed to detect edges accurately, smoothly and thinly. A penalising method is used to handle these constraints in PSO.

3.3.1 Edge Magnitude Measure

As 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, there could be eight ways of dividing the neighborhood of each pixel of an image into two regions according to the eight possible movement directions. As shown in Figure 2, two of them for horizontal and vertical movements are equivalent.

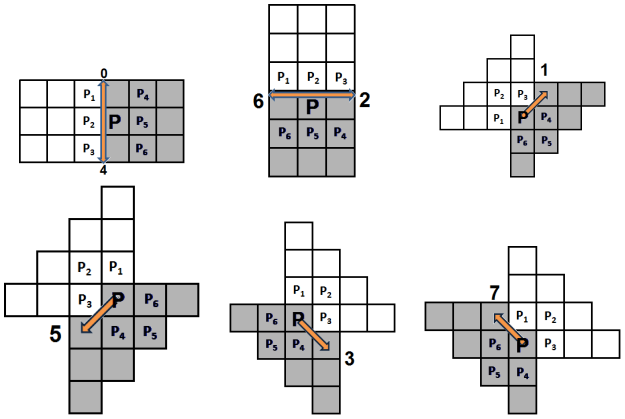


Figure 2: Eight moving ways from pixel P to its neighbours.

The algorithm calculates the edge magnitude, $EdgeMag_m(P)$, for pixel P in movement direction m. We expect that the pixels of each region are close in intensity (intraset distance), and the two sides have the highest possible difference in average intensity (interset distance). Therefore, we formulate $EdgeMag_m(P)$ as Equation (4) to maximise interset distance ($InterDis_m(P)$) between the regions and minimise intraset distance ($IntraDis_m(P)$) within the regions. To avoid dividing by zero, its numerator is added by 1.

$$EdgeMag_m(P) = \frac{InterDis_m(P)}{1 + IntraDis_m(P)} \tag{4}$$

where P is a pixel on the continuous curve represented by each particle, m is the movement direction from the pixel P to the next neighbor, ranging from 0 to 7, as shown in Figure 2. These distances are calculated as Equations 5 and 6.

$$InterDis_m(P) = min\left(1, |avg_{m_d(P)} - avg_{m_l(P)}|/w_1\right) \tag{5}$$

where $avg_{m_d}(P)$ and $avg_{m_l}(P)$ are the average intensities of the dark and light regions in the movement direction m for the pixel P, as calculated in $avg_{m_d(P)} = \frac{1}{9}\sum_{\forall P_i \in D} I_{P_i}$ and $avg_{m_l(P)} = \frac{1}{9}\sum_{\forall P_i \in L} I_{P_i}$, where D and L are the sets of the pixels in the dark and light regions respectively, m is the movement direction corresponding to the encoding scheme, and $w_1 = 90$ which is chosen based on empirical search.

The intraset distance $IntraDis_m(P)$ is the total sum of pairwise subtractions of pixel intensities of each region:

$$IntraDis_m(P) = \frac{1}{\binom{9}{2}}\left(\sum_{\substack{\forall P_i, P_j \in D \\ i > j}} min\left(1, |I_{P_i} - I_{P_j}|/w_2\right) \right.$$

$$+\sum_{\substack{\forall P_i, P_j \in L \\ i>j}} min\left(1, |I_{P_i} - I_{P_j}|/w_2\right)\Big) \quad (6)$$

where I_P is the intensity of the pixel P and $w_2 = 40$ which are chosen based on initial search.

3.3.2 NMS Factor For Edge Thinning

Non-maxima suppression (NMS) is one of the most important edge thinning techniques. It extracts local maxima of the gradient magnitude along the direction of the gradient vector and suppresses non-maximal edges. For each direction m, the $EdgeMag_m$ of the each pixel of a continuous edge is compared to the $EdgeMag_m$ of pixels P_1, P_2, ..., P_6 (as shown in Figure 2) placed on the both sides of the edge. The NMS factor of each pixel in each direction is the number of pixels which have a lower $EdgeMag_m$:

$$NMS_m(P) = |\{P_i | i \in 1..6, EdgeMag_m(P_i) < EdgeMag_m(P)\}| \quad (7)$$

where $|.|$ is the cardinality of a set and P_i is a neighbour of the pixel P as shown in Figure 2. The value of NMS is an integer number ranging from 0 to 6.

3.3.3 Possibility Score of Curve on Continuous Edge

The value of NMS factor in conjunction with $EdgeMag_m(P)$ is used to indicate a possibility score of a pixel lying on an edge in the direction m.

$$PScore_m(P) = \frac{1}{1+e^{-10(EdgeMag_m(P)-TH)}} \times \frac{1}{1+e^{-2(NMS_d(P)-4)}} \quad (8)$$

where TH is the threshold value between 0 and 1 that is defined by the user; $PScore_m(P)$ is the possibility score of the pixel P lying on an edge in the direction of m. Since thresholding techniques often cause broken edges in edge detection, we use the above equation to minimise the effect of using these techniques and improve the detection of the weak edges.

The uniformity measure of a curve introduced in [27] measures the similarity of pixel intensities along the curve:

$$U_C = \frac{1}{255 * max} \sum_{i=1}^{max} |I_{P_{i+1}} - I_{P_i}| \quad (9)$$

where I_{P_i} is the intensity of the i^{th} pixel on the curve represented by each particle. U_C is a real number between 0 and 1 and a lower value of this factor for a curve implies a better fit on the actual edge, as pixel intensities are similar along the curve. 255 in the denominator of Equation (9) is the maximum distance between two pixels in an image with a resolution with 8 bits per pixel.

The possibility score of the curve C being a continuous edge in the image is calculated as

$$PScore(C) = \frac{\sum_{\forall P_i \in C} PScore_{m_i}(P_i)/(max+1)}{1+U_C} \quad (10)$$

This possibility score measure, as a part of the new fitness function, is formulated such that it maximises the possibilities of the pixels on the curve to be placed on an edge and minimises the uniformity factor of the curve.

3.3.4 Curvature Cost of Contiuous Edges

To reduce the effect of producing jagged edges, we introduce a curvature cost of a continuous edge. Firstly, the curvature cost (CC) of each edge pixel is introduced to show a local measure of curvature for edge pixels. This measure for a particular pixel is defined based on the movement directions from the pixel to its next adjacent pixels, as shown in Equation (11).

$$CC(m_i, m_{i+1}) = \begin{cases} |m_i - m_{i+1}|/w_3 & , \quad |m_i - m_{i+1}| \leq 4 \\ (8 - |m_i - m_{i+1}|)/w_3 & , \quad otherwise \end{cases} \quad (11)$$

where m_i is the ith movement direction in a particle and $w_3 = 40$, which is chosen based on initial search.

The curvature cost of the curve C represented by a particle can be estimated by the Equation (12).

$$CCost(C) = \frac{1}{max-2}\Big(\sum_{i=1}^{max/2-1} CC(m_i, m_{i+1}) + \sum_{i=max/2+1}^{max-1} CC(m_i, m_{i+1})\Big) \quad (12)$$

3.3.5 New Penalised Fitness Function

The fitness of a particle encoding curve C is defined as follows:

$$Fitness(C) = PScore(C) - CCost(C) \quad (13)$$

subject to two constraints:

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

where $Cross(C)$ shows 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 particles, 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 avoid/reduce false alarms.

The selection of constraint handling methods is very problem dependent. Although penalising methods should be tuned for any constrained optimisation problem, their rapid convergence characteristic makes them attractive in many constrained optimisation problems [7]. Accordingly, we define a non-stationary, multi-stage penalty fitness function for edge detection to handle those two constraints, as shown in Equation (14).

$$PenFit(C) = Fitness(C) - h(K)(Cross(C) + \theta(q(C))q(C)) \quad (14)$$

where $h(K) = \sqrt{K}$, K is the current iteration number of the PSO algorithm and $q(C) = max(0, HP - PScore(C))$. $\theta(q(C))$ is calculated as Equation (15):

$$\theta(q(C)) = \begin{cases} 1 & \text{q(C)<0.001} \\ 2 & \text{q(C)<0.1} \\ 10 & \text{otherwise} \end{cases} \quad (15)$$

3.4 Summary of the Proposed Algorithm

The new constrained discrete PSO-based algorithm is outlined in Algorithm 1. In the first step, the edge possibility scores in eight different directions are calculated for each pixel of an image (line 1). Then PSO is applied to each unprocessed pixel to detect continuous edges. In each iteration (lines 4–18), the uniformity factor, edge possibility, curvature cost of each curve presented by each particle are calculated. The fitness values of each particle is computed, and the best and the worst particles are found. The worst particle is replaced with a new random one. After updating the velocities and the positions of the particles, the stopping criteria are checked whether the maximum number of iterations is exceeded or minimum error criterion is attained. Once the best continuous curve, C^* is found, its penalised fitness value is checked. If the value is not negative, all pixels on the curve C^* are marked as edges; otherwise all pixels in the neighbourhood of the pixel are marked as processed pixels (lines 19–22).

4. EXPERIMENT DESIGN

To investigate the effectiveness of the new algorithm, we compare it with a modified version of Canny [9], RRO [17], and a previous PSO algorithm [27] on two sets of benchmark images at different types and levels of noise. We applied the hysteresis thresholding and NMS techniques to Canny to improve the performance.

Algorithm 1 Constrained PSO-based edge detection algorithm

Calculating $EdgeMag_m$ for each pixel *P* on an image
For each pixel P on an image do
  If *P* is unprocessed and not marked as an edge then
    Initialize PSO population randomly for pixel *P*
    $K = 0$
    Repeat
      Increment K and calculate $h(K)$
      For each particle (decoded as curve C) do
        Evaluate $U(C)$, $PScore(C)$ and $CCost(C)$
        Evaluate $q(C)$ and $\theta(q(C))$
        Evaluate $Fitness(C)$ and $PenFit(C)$
        If $PenFit(C)$ is better than best fitness value
          Assign C to best particle
      Replace the worst particle with a new random one
      For each particle (decoded as curve C) do
        Calculate particle velocity
        Update particle position and apply update rule (3)
    Until stopping criteria are attained
    Select best particle and decode it as curve C^*
    If $PenFit(C^*) \geq 0$ then
      Mark all pixels on curve C^* as an edge
    Else mark all pixels within the square S as processed

Note that these techniques are not applicable for the RRO detector. This section describes the image sets, performance measure, and parameter settings, which are used in the experiments.

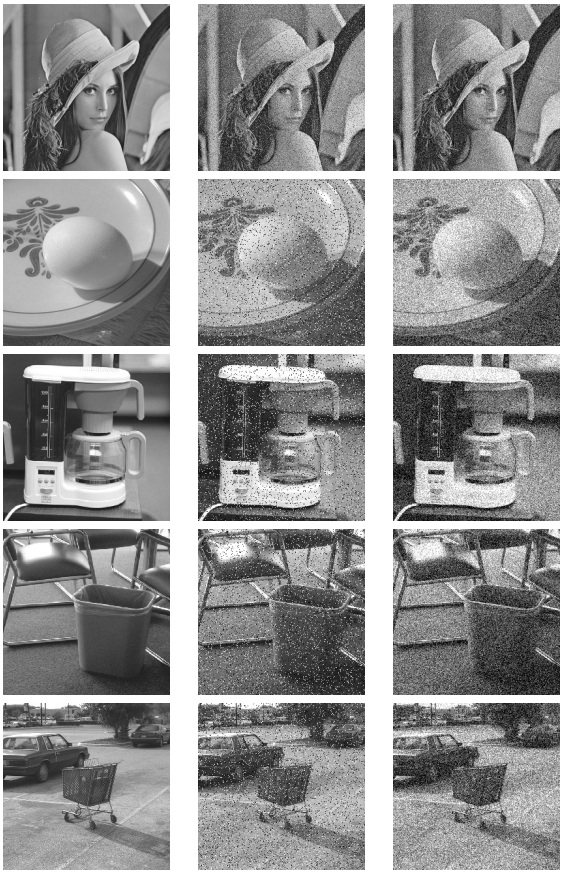
4.1 Image Sets

Two different images sets are used in our experiments. The first image set includes five natural images, Lena, egg, coffee maker, rubbish bin and car (see Figure 3), which are commonly used as benchmarks for edge detection. These images were downloaded from the South Florida University database [2]. To explore the performance of the new algorithm in noisy environments, these images are corrupted by two different types of noise: impulsive and Gaussian noises (see the images in Figure 3 in columns (b) and (c)). The probability of the impulsive noise is 0.1 and the peak-signal to noise ratio (PSNR) is $16dB$ for the Gaussian noise in these noisy images. As the ground truth of these images are not available, we will use them in a subjective (qualitative) comparison.

The second set includes two synthetic images (circle, triangle) and one real image (street/road). The real image has been provided by the University of Cordoba (Spain) and its ground truth edge map is available from [1]. The size of each image is 256×256 pixels and the resolution of each is 8 bits per pixel. These images are shown in Figure 4. To investigate the performance of the new algorithm 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 $22dB$ with a step size of $2dB$. As the ground truth of these images are available, we will use them for an objective (quantitative) comparison.

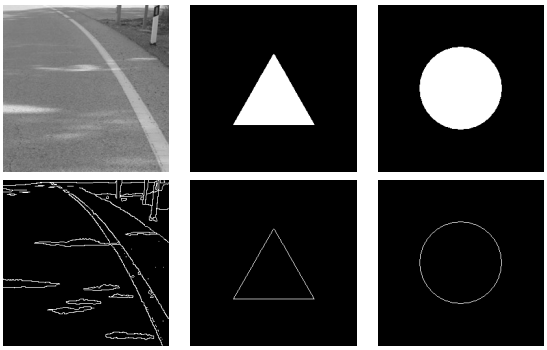
4.2 Quantitative Performance Measure

To evaluate the performance of the new algorithm, we used Pratt's Figure of Merit (PFOM) measure that is commonly used as a quantitative measure for the objective comparison of the *localisation accuracy* of edge detection algorithms [23]. This measure is defined by equation (16).



(a) (b) (c)

Figure 3: Example images for subjective comparison. (a) Original images Lena, egg, coffee maker, rubbish bin and car; (b) images with *impulssive noise* (noise probability=0.1); (c) images with *Gaussian noise* (PSNR=16dB).



(a) (b) (c)

Figure 4: Example images for objective comparision. (a) One real image from the UCO university and its manual ground truth images [1]; (b)–(c) two synthetic triangle and circle images and their ground truth.

$$R_{PFOM} = \frac{1}{max(I_I, I_A)} \sum_{i=1}^{I_A} \frac{1}{1 + \beta d(i)^2} \qquad (16)$$

where 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 near to zero.

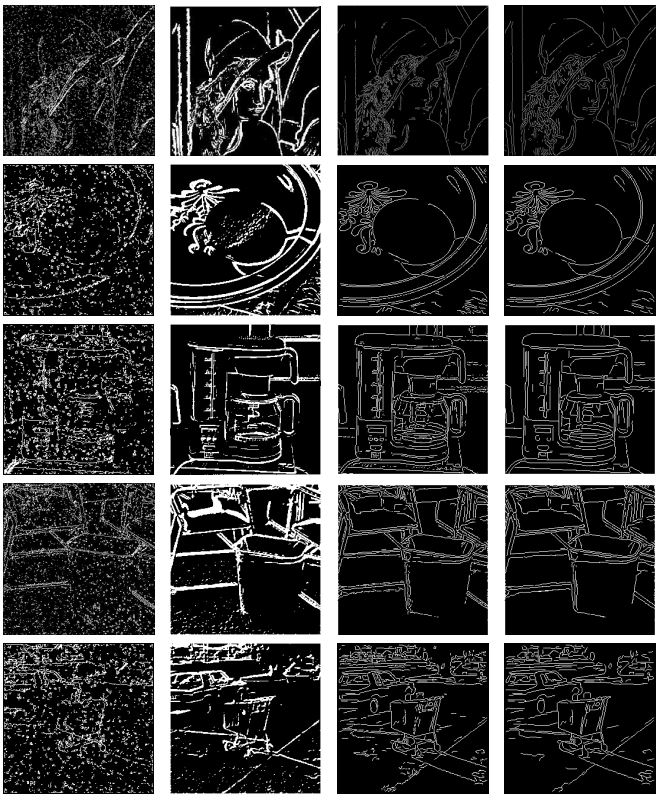
4.3 Parameter Values

In the new PSO algorithm, the population size is 50 and the maximum number of iterations is 200. The minimum length of a continuous edge, max was set at 19, $RectSize$ at 3, TH at 0.07, and HP at 0.5. We used the values $w = 0.7298$, $c_1 = 1.4962$, $c_2 = 1.4962$ for the parameters in Equation (2). We used the following parameters for the Canny edge detector: $high\ threshold = 100$, $low\ threshold = 20$, $delta = 1$, $size\ of\ the\ Gaussian filter = 5$. The edge-height parameter of the RRO detector, which defines the minimum gray-level differential across an edge, was set to 15. These values were chosen based on common settings [16] and previous experiments in [27, 9, 17] in order to make consistent and fair comparisons.

5. RESULTS

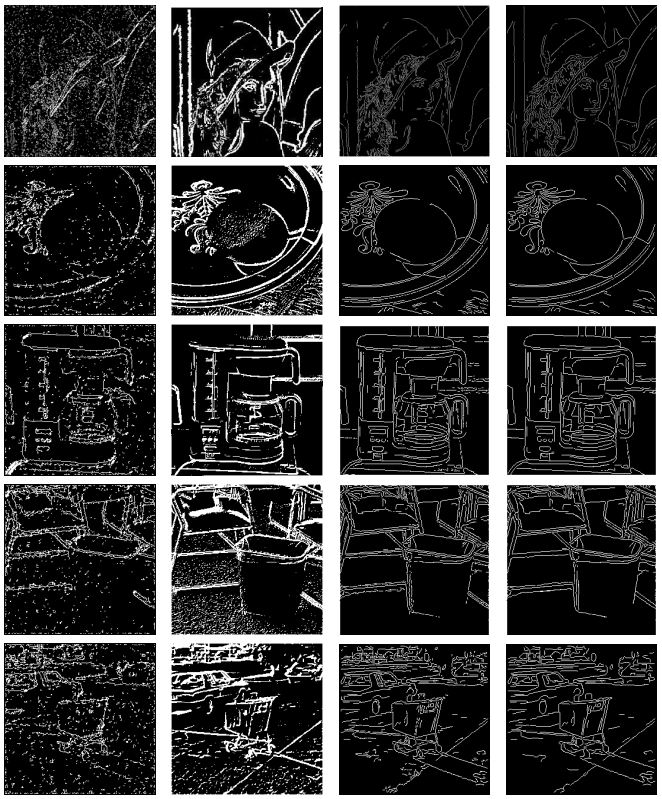
5.1 Subjective/Qualitative Comparison

The resulting images are shown in Figures 5 and 6 after applying the Canny [9], the RRO [17], the previous PSO-based algorithm [27] (PSO-1), and the new algorithm (PSO-2) on the images in the first set corrupted by impulsive and Gaussian noises respectively.



Canny [9] RRO [17] PSO-1 [27] PSO-2

Figure 5: Subjective results of edge detection produced by four algorithms on the five images corrupted by *impulsive noise*.



Canny [9] RRO [17] PSO-1 [27] PSO-2

Figure 6: Subjective results of edge detection produced by the four algorithms on the five images with *the Gaussian noise*.

The resulting images in Figure 5 show that the new algorithm (PSO-2) performed better than all the other three algorithms on the five images with *impulsive noise* at a noise probability level of 0.1. The Canny algorithm even with post-processing did not work well for these noisy images and there are many noise spots in the resulting images. This suggests that the Canny algorithm is not suitable for detecting edges for the images corrupted by impulsive noise and very sensitive to this kind of noise. The RRO detector operated better than Canny, however the detected edges are thicker than those detected by Canny. The two PSO-based algorithms detected edges much thinner than the RRO detector and PSO-2 detects edges in a more continuous format than PSO-1. As can be seen from Figure 5, for the Lena image, there are some broken edges on Lena's hat in the resulting image by RRO and PSO-1, while PSO-2 improved the detection of the edges in this area and reduced the broken edges. The edges detected by PSO-1 on the bar in the upper left corner of the image are broken and some missing, while PSO-2 significantly improved the detection of the edges in this region. 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. PSO-1 operated better than RRO on the egg but there are still several broken edges on the boundary of the egg. PSO-2 reduced the broken edges in the egg's boundary especially in the bottom of the egg. PSO-1 and 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 side of the image. PSO-2 reduced jagged edges in this region. PSO-1 and RRO did not operate well on the rubbish bin especially on its righ and bottom side of the bin; there are many broken edges in the image resulted by RRO and jagged edges in the image resulted by PSO-1, while PSO-2 improved the detection

of the edges in this area. PSO-2 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 PSO-1 and RRO did not work well in these areas.

Figure 6 shows the resulting images after applying the four algorithms on the five noisy images corrupted by Gaussian noise (PSNR=16dB). The comparison of these results with those for 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, PSO-1 and PSO-2 for these images have very similar quality to those with impulsive noise, although PSO-1 produced more jagged edges in the images with *impulsive noise* than with *Gaussian noise*. PSO-2 recognised more continuous and smoother edges than PSO-1 and Canny, and also detected edges much thinner than RRO.

5.2 Quantitative Comparison

For the quantitative/objective comparison between the new algorithm and the other three algorithms, the localisation accuracy (PFOM) was calculated from the resulting images after applying the four algorithms to the second set of images in different noise levels. The PFOM values are plotted at 11 different Gaussian noise level and 9 impulsive noise levels (see Figure 7). PSNR ranges from 0 to 22dB with step of $2dB$ and the noise probability ranges from 0.1 to 0.5 with step of 0.05.

As can be seen from the resulting plots in Figure 7, PSO-2 generally outperformed the other three algorithms especially when a high level of noise is present in the images. Canny operated reasonably well on the images with low level of Gaussian noise, but it did not work well in the images even with low level of impulsive noise in most cases. These resulting plots also show that RRO outperformed PSO-1 in the images with impulsive noise in most cases, while its performance is often lower than PSO-1 when the images were corrupted by Gaussian noise. This suggests that PSO-1 is less sensitive to Gaussian noise but more sensitive to impulsive noise than RRO. The accuracy of most algorithms is decreased when noise level is increased. However, PSO-2 can overcome high level of noise and it is less sensitive to noise than other methods in most cases.

6. CONCLUSIONS

The main goal of this paper was the reduction of the broken and jagged edges in noisy images by the use of PSO. The goal was successfully achieved by developing a new PSO-based approach to detecting edges in such images. To overcome noise and reduce the broken and jagged edges, a new penalised fitness function was developed based on the possibility score of a curve being fitted on an edge and the curvature cost of the curve with two constraints. The performance of the new algorithm was examined and compared with three existing algorithms on two different sets of images with two different types of noise at different levels. The qualitative and quantitative results showed that the new algorithm generally performed better than the Canny and RRO edge detectors and also a previous PSO-based algorithm on the images corrupted by the Gaussian and impulsive noises. In addition, the new algorithm does not use any extra preprocessing and post processing techniques. These results also show that Canny could perform reasonable well for some images with low level of noise. PSO-1 was also found to be less sensitive to Gaussian noise and more sensitive to impulsive noise than RRO.

The execution time of the new algorithm was longer (40–50 seconds depending on the noise level) than Canny (2–3 seconds including preprocessing and post processing) and RRO (3 seconds). In the future research, we will investigate ways to improve the efficiency of the new algorithm through developing a novel topology and a different method of the initialisation of the PSO population.

7. REFERENCES

[1] *Images from automatic generation of consensus ground truth for comparison of edge detection techniques*. Available from `http://www.uco.es/~ma1fegan/investigacion/imagenes/ground-truth.html`.

[2] *South Florida University for edge detector comparison*. Available from `http://marathon.csee.usf.edu/edge/edge_detection`.

[3] M. R. Al Rashidi and M. E. El-Hawary. A survey of particle swarm optimization applications in electric power systems. *IEEE Transaction on Evol. Comp.*, 13(4):913–918, 2009.

[4] M. Basu. Gaussian-based edge-detection methods: a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 32(3):252–260, 2002.

[5] M. Breaban, M. Ionita, and C. Croitoru. A new PSO approach to constraint satisfaction. In *IEEE Congress on Evolutionary Computation*, pages 1948–1954, 2007.

[6] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698, 1986.

[7] G. Coath and S. Halgamuge. A comparison of constraint-handling methods for the application of particle swarm optimization to constrained nonlinear optimization problems. In *IEEE Congress on Evolutionary Computation*, volume 4, pages 2419–2425, 2003.

[8] G. W. Cook and E. J. Delp. Multiresolution sequential edge linking. In *ICIP '95: Proceedings of the 1995 International Conference on Image Processing-Volume 1*, pages 782–791, Washington, DC, USA, 1995. IEEE Computer Society.

[9] L. Ding and A. Goshtasby. On the Canny edge detector. *Pattern Recognition*, 34(3):721–725, 2001.

[10] A. A. Farag and E. J. Delp. Edge linking by sequential search. *Pattern Recognition*, 28(5):611–633, 1995.

[11] R. C. González and R. E. W. *Digital Image Processing, Third Edition*. Prentice Hall, 2008.

[12] P. E. Hart. How the Hough transform was invented. *IEEE Signal Processing Magazine*, 26(6):18–22, 2009.

[13] X. Hu and R. Eberhart. Solving constrained nonlinear optimization problems with particle swarm optimization. In *6th World Multiconference on Systemics, Cybernetics and Informatics*, pages 203–206, 2002.

[14] S. Janson and M. Middendorf. A hierarchical particle swarm optimizer for noisy and dynamic environments. *Genetic Programming and Evolvable Machines*, 7(4):329–354, 2006.

[15] F. Kennedy, R. Eberhart, and Y. Shi. *Swarm Intelligence*. Morgan Kaufmann, San Francisco, CA, 2001.

[16] E. Laskari, K. Parsopoulos, and M. Vrahatis. Particle swarm optimization for integer programming. In *Proceedings of the IEEE Congress on Evolutionary Computation*, volume 2, pages 1582–1587, 2002.

[17] D. H. Lim. Robust edge detection in noisy images. *Comput. Stat. Data Anal.*, 50(3):803–812, 2006.

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

[19] D. S. Lu and C. C. Chen. Edge detection improvement by ant

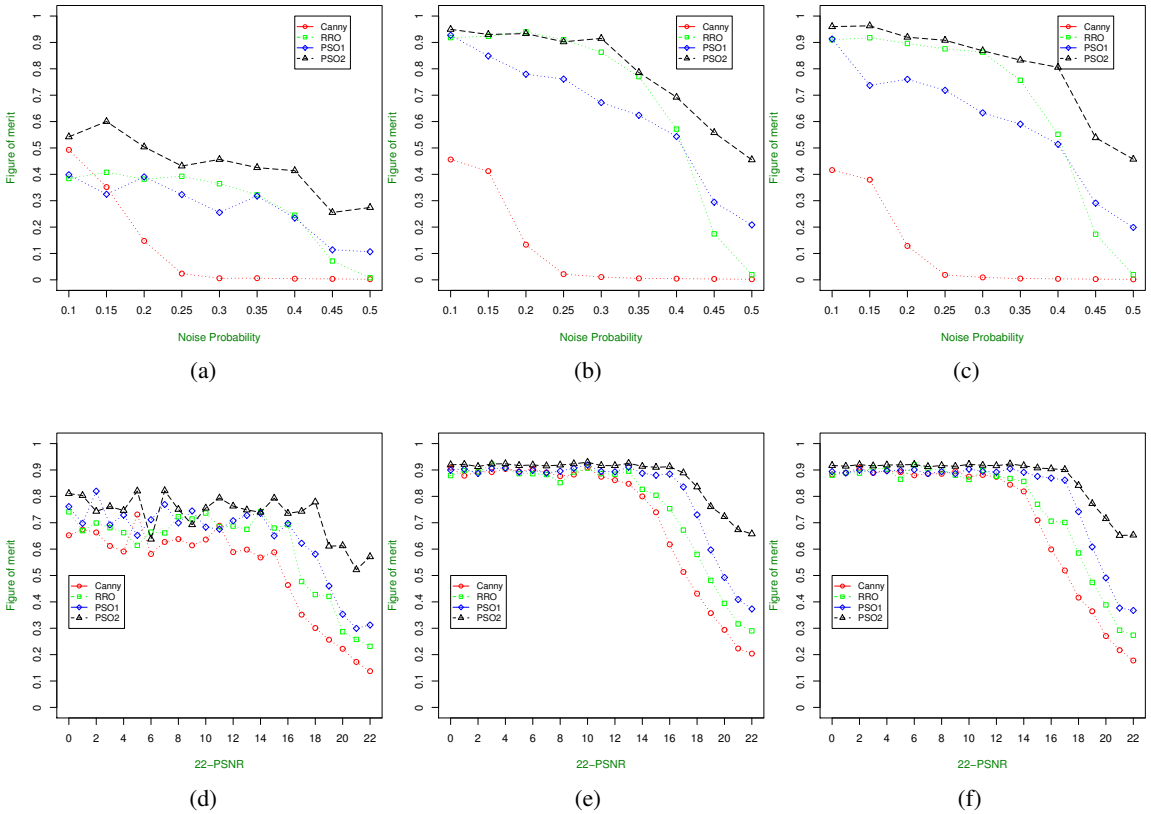


Figure 7: PFOM for the street, triangle, and circle images in the second data set. (a)–(c): with different impulsive noise levels (the noise probability ranging from 0.1 to 0.5); and (d)–(f) with different Gaussian noise levels (PSNR ranging from 0 to 22dB).

colony optimization. *Pattern Recognition Letters*, 29(4):416–425, 2008.

[20] C. Monson and K. Seppi. Linear equality constraints and homomorphous mappings in PSO. In *IEEE Congress on Evolutionary Computation*, volume 1, pages 73–80, 2005.

[21] A. Nakib, H. Oulhadj, and P. Siarry. Fractional differentiation and non-pareto multiobjective optimization for image thresholding. *Engineering Applications of Artificial Intelligence*, 22(2):236–249, 2009.

[22] H. Pan, L. Wang, and B. Liu. Particle swarm optimization for function optimization in noisy environment. *Applied Mathematics and Computation*, 181(2):908–919, 2006.

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

[24] K. Sedlaczek and P. Eberhard. Optimization of nonlinear mechanical systems under constraints with the particle swarm method. *Proceedings in Applied Mathematics and Mechanics (PAMM)*, 4(1):169–170, 2004.

[25] M. Setayesh, M. Johnston, and M. Zhang. Edge and corner extraction using particle swarm optimisation. In J. Li, editor, *AI 2010: Advances in Artificial Intelligence*, volume 6464 of *LNCS*, pages 323–333. Springer, 2011.

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

[27] M. Setayesh, M. Zhang, and M. Johnston. Improving edge detection using particle swarm optimisation. In *Proceedings of the 25th International Conference on Image and Vision Computing, New Zealand*. IEEE Press, 2010.

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

[29] B. Tremblais and B. Augereau. A fast multiscale edge detection algorithm based on a new edge preserving pde resolution scheme. *International Conference on Pattern Recognition*, 2:811–814, 2004.

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

[31] J. Wang, Z. Kuang, X. Xu, and Y. Zhou. Discrete particle swarm optimization based on estimation of distribution for polygonal approximation problems. *Expert Systems with Applications*, 36(5):9398–9408, 2009.

[32] J. Zhao and Z. Li. Particle filter based on particle swarm optimization resampling for vision tracking. *Expert Systems with Applications*, 37(12):8910–8914, 2010.