Int. J. Electron. Commun. (AEÜ) 69 (2015) 1282–1290

Contents lists available at ScienceDirect

International Journal of Electronics and Communications (AEÜ)

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

Cellular edge detection: Combining cellular automata and cellular learning automata

Mohammad Hasanzadeh Mofrad[a,b], Sana Sadeghi[c], Alireza Rezvanian[a,d,*], Mohammad Reza Meybodi[a]

[a] Soft Computing Laboratory, Computer Engineering and Information Technology Department, Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran
[b] ICT Research Institute, Academic Center for Education, Culture and Research (ACECR), Tehran, Iran
[c] Department of Engineering, Payame Noor University, Tehran, Iran
[d] Department of Computer Engineering, Hamedan Branch, Islamic Azad University, Hamedan, Iran

ARTICLE INFO

Article history:
Received 7 November 2014
Accepted 13 May 2015

Keywords:
Image processing
Edge detection
Learning automata
Cellular automata
Cellular learning automata

ABSTRACT

In this paper, we propose a cellular edge detection (CED) algorithm which utilizes cellular automata (CA) and cellular learning automata (CLA). The CED algorithm is an adaptive, intelligent and learnable algorithm for edge detection of binary and grayscale images. Here, we introduce a new CA local rule with adaptive neighborhood type to produce the edge map of image as opposed to CA with fixed neighborhood. The proposed adaptive algorithm uses the von Neumann and Moore neighborhood types. Experimental results demonstrate that the CED algorithm has superior accuracy and performance in contrast to other edge detection methods such as Sobel, Prewitt, Robert, LoG and Canny operators. Moreover, the CED algorithm loses fewer details while extracting image edges compare to other edge detection methods.

1. Introduction

Image processing is a type of signal processing where the input image is converted to an image or a set of image features. An image defines as a 2-*D* function of $f(x,y)$ where x and y are spatial coordinates and f is the domain of the image intensity or grayscale level at each pair of (x,y).

Edge detection is one of the key processes of machine vision systems. This process reduces the computation time and storage space of images while preserving valuable information about the image boundaries. Prewitt [1] and Canny [2] are two well-known edge detection algorithms. Assuming wrong points as edges and initializing large number of parameters are two main defects of these traditional edge detection algorithms.

Edge detection has a lot of applications in image processing [3] and medical imaging [4]. In [4] a Cellular Neural Network (CNN) used for edge detection and noise removal. In [5] edge detection performed by using Particle Swarm Optimization (PSO). Moreover, in [6] ellipsoid shapes are detected by an edge detection approach.

Learning automaton is an adaptive entity which is placed in an unknown environment. The ultimate goal of a learning automaton is to learn the optimal action through the reinforcement signal which is produced by the environment as the feedback of selected actions of learning automaton. Learning automata (LA) have applications in evolutionary computing [7,8], grid computing [9,10], complex network sampling [11], solving maximum clique problem [12] and solving minimum vertex cover problem [13].

CA is a discrete framework consists of a regular grid of cells, each of which including a finite set of states. CA has various applications in image processing. In [14] a hybrid method based on CA and fuzzy logic introduced for impulse noise elimination. In [15] CA used for enhancement, smoothing and denoising of digital images. In [16,17] various CA applications in image processing are introduced such as: noise filtering, feature selection, thinning, convex hull, etc.

CLA follow a mathematical model for complex systems, which consists of tiny entities. CLA compose of CA where a learning automaton lies in each CA's cells. A segmentation algorithm is proposed in [18] that processes the color and texture of image by CLA and segment the skin-like areas of image.

Edge detection is the process of simplifying the representation of an image into a set of extracted objects boundaries. The CED algorithm composes of two key components including: (1) CA that store the boundary information of edges, (2) CLA that learn the texture, size and boundary distribution of edges. The interworking

* Corresponding author at: Soft Computing Laboratory, Computer Engineering and Information Technology Department, Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran. Tel.: +98 2164545120; fax: +98 2166495521.
E-mail address: a.rezvanian@aut.ac.ir (A. Rezvanian).

http://dx.doi.org/10.1016/j.aeue.2015.05.010
1434-8411/

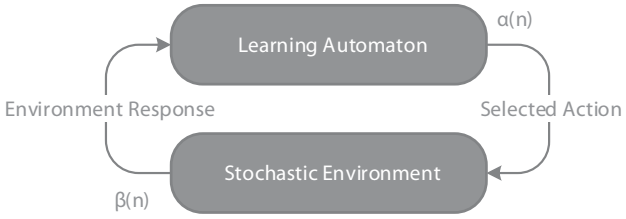

Fig. 1. The interaction sequence between learning automaton and environment where $\alpha(n)$ and $\beta(n)$ are the selected action of learning automaton and environment response of the selected action, respectively.

of these two components enables the CED algorithm to accurately extract edge information from images.

The remaining parts of this paper are organized as follows. In Section 2, we present a brief overview of LA, CA and CLA. The CED algorithm is introduced in Section 3. In Section 4, we present and discuss the qualitative and quantitative results for various sets of sample images. Finally the paper is concluded in Section 5.

2. Automata theory

2.1. Learning automata

A learning automaton [19] is a machine that can perform a finite number of actions. Each selected action of learning automaton is evaluated in a probabilistic environment and the evaluation response is applied to learning automaton with a positive or negative reinforcement signal. The aforementioned signals affect the next action that is taken by the learning automaton. The ultimate goal of a learning automaton is to learn the foremost action among all actions. The best action is the one that maximizes the probability of getting a positive signal from the environment. Fig. 1 depicts the learning automaton interactions toward the environment.

The environment can be modeled as a triplet of $E = \{\alpha, \beta, c\}$ where $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ is the input set, $\beta = \{\beta_1, \beta_2, \ldots, \beta_m\}$ is the output set and $c = \{c_1, c_2, \ldots, c_r\}$ is the set of penalty probabilities where c_i is the probability that α_i gets the penalty signal. In static environments the c_i values remain unchanged while in non-static environments these values change during the progress of learning automaton.

According to the nature of its input, the environment can be divided into three models of P, Q and S:

- In P-model environment β is a two members set ($\{0, 1\}$) where $\beta_1 = 1$ and $\beta_2 = 0$ are penalty and reward signals, respectively.
- In Q-model environment β is an arbitrary finite set of discrete symbols.
- In S-model environment β is a random variable from the interval [0,1] of real numbers.

A learning automaton is a fixed quintuple like $\{\alpha, \beta, F, G, \phi\}$ where $\alpha = \{\alpha_1, \ldots, \alpha_r\}$ is the action set, $\beta = \{\beta_1, \ldots, \beta_m\}$ is the input set, $\phi = \{\phi_1, \ldots, \phi_s\}$ is the internal states set, $F: \phi \times \beta \to \phi$ is the function that generates the new state of learning automaton and $G: \varphi \to \alpha$ is the output function that maps the current state to the next input of learning automaton.

Variable structure learning automata (VSLA) are quadruple of $\{\alpha, \beta, p, T\}$ where $\alpha = \{\alpha_1, \ldots, \alpha_r\}$ is the action set, $\beta = \{\beta_1, \ldots, \beta_m\}$ is the input set, $p = \{p_1, \ldots, p_r\}$ is the probability vector of each action and $p(n+1) = T[\alpha(n), \beta(n), p(n)]$ is the learning algorithm. The following algorithm is a typical linear learning algorithm. Assume that, the action α_i is selected in nth interval, the positive and negative responses are calculated through (1) and (2), respectively.

(-1,1)	(0,1)	(1,1)
(-1,0)	(0,0)	(1,0)
(-1,-1)	(0,-1)	(1,-1)

(a) Moore Neighborhood

	(0,1)	
(-1,0)	(0,0)	(1,0)
	(0,-1)	

(b) von Neumann Neighborhood

Fig. 2. (a) Moore neighborhood and (b) von Neumann neighborhood.

By using these two response schemes, the probability vector of learning automaton will be updated.

$$\begin{cases} p_i(n+1) = p_i(n) + a[1 - p_i(n)] \\ p_j(n+1) = (1-a)p_j(n) \quad \forall j\, j \neq i \end{cases} \tag{1}$$

$$\begin{cases} p_i(n+1) = (1-b)p_i(n) \\ p_j(n+1) = \left(\frac{b}{r-1}\right) + (1-b)p_j(n) \quad \forall j\, j \neq i \end{cases} \tag{2}$$

where in (1) and (2), a and b are the reward and penalty parameters. If a and b are equal then the learning algorithm will be called linear reward penalty algorithm (L_{RP}). If a is much larger than b then the learning algorithm will be called linear reward epsilon penalty ($L_{R\varepsilon P}$). If b is equal to zero then the learning algorithm will be called linear reward inaction (L_{RI}).

2.2. Cellular automata

CA [20] are a model for investigating the behavior of complex systems and have typical applications in image processing [14]. CA are discrete dynamic systems that their behavior are completely dependent on their local communications. In CA, the space defines as a network of cells and time advances discretely. Also in each time step, rules are deployed globally and the next state of each cell is determined through the current state of its adjacent cells. The CA rules determine that how neighboring cells influence the central cell. Two cells will be neighbors, if one of them can influence the other one through a governing CA rule.

A D-dimensional CA is a quadruple of $\{Z^D, \phi, N, F\}$ where Z^D is a network of D-tuple ordered integers that could be a finite, semi-finite or infinite set, $\phi = \{1, \ldots, m\}$ is a finite set of actions, $N = \{\bar{x}_1, \ldots, \bar{x}_{\bar{m}}\}$, $\bar{x}_i \in Z^D$ is the neighborhood vector that is a finite subset of Z^D and $F: \phi^{\bar{m}} \to \phi$ is the local rule of CA. The neighborhood vector $N(u)$ defines the relative position of neighbors for each cell u in the cell network. $N(u)$ is calculated by (3) in which the neighbor cells should meet the two constraints of (4).

$$N(u) = \{u + \bar{x}_i | i = 1, \ldots, \bar{m}\} \tag{3}$$

$$\begin{cases} \forall u \in Z^D \Rightarrow u \in N(u) \\ \forall u, v \in Z^D \Rightarrow u \in N(v) \wedge v \in N(u) \end{cases} \tag{4}$$

Moore and von Neumann are two well-known neighborhood types of CA. Fig. 2 shows a Moore neighborhood with $N = \{(-1,1), (0,1), (1,1), (-1,0), (0,0), (1,0), (-1,-1), (0,-1), (1,-1)\}$ and a von Neumann neighborhood with $N = \{(0,1), (-1,0), (0,0), (1,0), (0,-1)\}$ in a 2-D space.

The local rule F can be characterized as *general*, *totalistic* and *outer totalistic* rules:

- In general rule the next value of a cell depends on the value of its neighboring cells in the current time step.
- In totalistic rule the next value of a cell depends on the various states of its neighboring cells.
- In outer totalistic rule the next value of a cell depends on its current state and the various states of its neighboring cells.

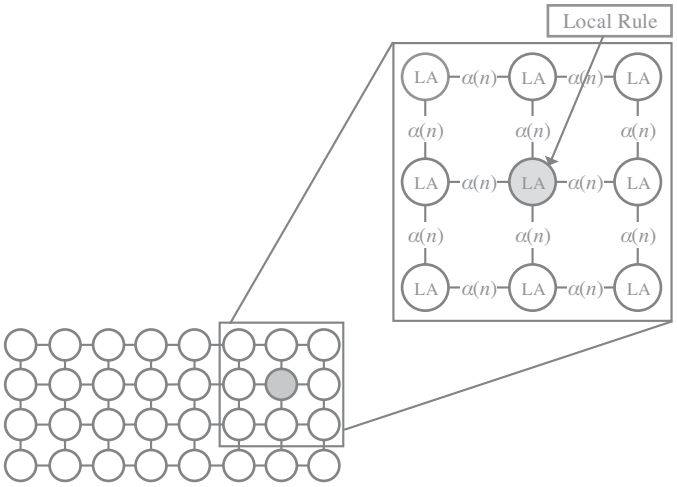

Fig. 3. A typical CLA in a cellular environment with Moore neighborhood. In *n*th time step, the local rule is determined by the selected actions of neighboring LA and is deployed to the central learning automaton as the reinforcement signal.

2.3. Cellular learning automata

CLA [21] are combinations of LA [19] and CA [20]. The CLA model combines the learning ability of LA with cooperation window of CA. The CLA model is a tool for modeling the simple component systems. In CLA, the behavior of each component is determined from the behavior of its neighbors and its previous experiments. CLA show a complex behavior through simple interactions of single components. Thus, CLA are suitable for modeling the real-world problems.

As shown in Fig. 3, CLA consist of CA that each of its cells equips with one or more LA that determines the state of the corresponding cell. Like CA, in CLA the local rule governs the environment and determines how the selected action gets the reward or penalty signal. These signals update the probability vector of the corresponding learning automaton in order to attain the optimal decision. CLA framework aims to utilize LA to calculate the probability of state transformation in stochastic CA. CLA can be divided into two categories of asynchronous and synchronous where in synchronous CLA, all cells synchronize with a global clock and cooperate simultaneously.

The *D*-dimensional CLA are quintuple of $\{Z^D, \phi, A, N, F\}$ where Z^D is a network of *D*-tuple ordered integers that could be a finite, semi-finite or infinite set, $\phi = \{1, \ldots, m\}$ is a finite set of actions, *A* is a set of LA that each of them corresponds to a specific CLA cell, $N = \{\bar{x}_1, \ldots, \bar{x}_{\bar{m}}\}$ is the neighborhood vector which is a finite subset of Z^D and $F : \phi^{\bar{m}} \to \underline{\beta}$ is the local rule of CA where β is the reinforcement signal.

3. Cellular edge detection

In this section, we propose a cellular model which utilizes a nested chain of CA and CLA for edge detection of images. The proposed algorithm can be used for edge detection of binary and grayscale images. The edges of an image are the boundaries in which an intense alteration may occur. Since the edge detection of image has various applications, it is necessary for it to run independent from the feature extraction process.

In cellular edge detection (CED) algorithm CA are assigned to the image in such a way that each cell of CA corresponds to a pixel of image. Also, the CLA determines the neighborhood type of CA. The combination of these two techniques accurately detects the edges of image and converges to the optimal edge map.

3.1. Cellular edge detection model

The proposed model utilizes a 2-*D*, symmetric and nondeterministic CA in addition to a 2-*D*, symmetric and nondeterministic CLA. The CA rules can detect relatively intense variations in brightness, color or details of image and find edges based on these extracted information. In the following of this section, we present the framework and formulation of CED algorithm.

3.1.1. Cellular automata framework

A *D*-dimensional CA is defined as follows:

- $CA = \{Z^2, \Phi, N, F\}$
- Z^2 is a 2-*D* CA identical to the input image $I_{m\times n}$.
- $\Phi = \{0, \ldots, 255\}$ is the set of finite states for a grayscale image with 256 intensity levels and $\Phi = \{0, 1\}$ is the set of finite states for a binary image with 2 intensity level.
- $N = \{Moore_{3\times3}, von\ Neumann_{3\times3}\}$ is the set of available neighborhood types for CA.
- *F* is the local rule of CA where the local rules of binary and grayscale images are calculated through (5–7), respectively. *F* acts as the relationship criterion of central cell and its neighbors.

3.1.2. Binary edge detection rule

A typical input image *I* consists of a set of pixels whose their values are zero or one. If the pixel value of *I* is one ($I = 1$), the corresponding cell will evaluate with the following constraints through (5):

1) If at least one of the cells around the central cell equals to zero, the corresponding pixel will identify as an edge in the output image *J* ($J = 1$).
2) If all of the cells around the central cell equal to one, the corresponding pixel will not identify as an edge in the output image *J* ($J = 0$).

Otherwise, if the pixel value of *I* is zero ($I = 0$), it will evaluate through (6). Based on (6), if the central cell equals to zero, the corresponding pixel of output image *J* will set to zero ($J = 0$).

$$J_{i,j} = F\left(N|I_{i,j} = 1\right) = \begin{cases} 1 & \text{if } \exists I \in N | I_N = 0 \\ 0 & \text{if } \forall I \in N | I_N = 1 \end{cases} \tag{5}$$

$$J_{i,j} = F\left(N|I_{i,j} = 0\right) = 0 \tag{6}$$

3.1.3. Grayscale edge detection rule

In (7), the local rule is derived from the states of central cell and its adjacent neighbors. If the absolute difference between the central pixel intensity and the maximum pixel intensity of neighborhood violates the threshold value θ, then the central cell will consider as edge ($J = 1$), and otherwise the central cell will not consider as edge ($J = 0$).

$$J_{i,j} = F(N) = \begin{cases} 1 & \text{if } \left|I_{i,j} - \max(N)\right| \geq \theta \\ 0 & \text{Otherwise} \end{cases} \tag{7}$$

3.1.4. Cellular learning automata framework

A *D*-dimensional CLA is defined as follow:

- $CLA = \{Z^D, \phi, A, N, F\}$
- Z^2 is a 2-*D* CLA identical to the input image $I_{m\times n}$.
- $\Phi = \{Moore_{3\times3}, von\ Neumann_{3\times3}\}$ is the set of finite states.
- *A* is the set of LA identical to the input image $I_{m\times n}$.
- $N = \{Moore_{3\times3}\}$ is the set of available neighborhood types for CLA.
- *F* is the local rule of CLA, which is defined by (8):

Algorithm 1 Cellular Edge Detection (CED)

Define
Input image $I_{m\times n}$ where m is the number of rows and n is the number of columns
Output image $J_{m\times n} = [0]_{m\times n}$
Construct a $CA_{m\times n}$ identical to $I_{m\times n}$
Initialize $\phi = \{0, 1\}$ the action set of CA for binary image and $\phi = \{0, \ldots, 255\}$ the action set of CA for grayscale image
Consider $N = \{Moore_{3\times3}, von\ Neumann_{3\times3}\}$ as a set of available neighborhood types of CLA. Construct a $CLA_{m\times n}$ identical to $I_{m\times n}$
Initialize $\phi = \{\phi_1, \phi_2\}$ the action set of CLA where $\phi_1 = Moore_{3\times3}$ is the action corresponding to select the Moore neighborhood type and $\phi_2 = von\ Neumann_{3\times3}$ is the action corresponding to select the von Neumann neighborhood type.
Initialize P the probability vector of LA with $P(r) = \left(\frac{1}{|r|}\right) | r \in \{1, 2\}$ where $|r|$ is the number of actions, $r = 1$ is the probability of selecting the Moore neighborhood type and $r = 2$ is the probability of selecting the von Neumann neighborhood type
Consider neighborhood window W as the selected neighborhood type of CA which is determined by CLA action selection
Initialize generation number $k = 0$, maximum number of generations K and threshold value θ
for $k \in [1, K]$ **do**
  Action selection: Synchronically all LA of CLA select an action from their action set ϕ based on probability vector P
  for $i \in [1, m]$ **do**
    for $j \in [1, n]$ **do**
      $W = \phi_{i,j}$ // Determine neighborhood type of CA
      $N = W(I_{i,j})$ // Calculate neighboring cells of the selected neighborhood window
      Calculate $J_{i,j}$ based on (5–7)
      Calculate reinforcement signal of $LA_{i,j}$ based on (8)
      Update probability vector of $LA_{i,j}$ based on (1) and (2)
    end for
  end for
end for

Fig. 4. Pseudo-code of CED algorithm.

$$\beta_{i,j} = F(N) = \begin{cases} 1 & \text{if } A_{i,j} = A(N) \forall A \in N \\ 0 & \text{Otherwise} \end{cases} \quad (8)$$

Alternatively, (8) calculates the reinforcement signal of each learning automaton through local interactions of LA. If all LA in the same neighborhood choose a same action, they will be rewarded, and otherwise they will be punished. This strategy leads LA to choose similar neighborhood type in analogous segments of image.

3.1.5. Cellular edge detection pseudo-code

The CED algorithm consists of six steps where the 2–5 steps are repeated till the maximum number of generations is reached. These steps are as follows:

1) Initializing the CA and CLA parameters
2) Selecting the actions of CLA
3) Calculating the output image J based on CA rules
4) Calculating the reinforcement signal β
5) Updating the probability vector P
6) Presenting the output image J

The 2–5 steps of the CED algorithm gradually lead the CLA to learn the appropriate neighborhood type in order to efficiently process the image and converge to an optimal edge map. The pseudo-code of CED algorithm is shown in Fig. 4.

4. Experimental results

The CED algorithm utilizes CA and CLA for edge detection process. The CA present the dynamicity in boundary growth and CLA demonstrate the object learning of CED. The combination of these two reinforcement learning techniques enables CED to easily process and learn the edges of images with balanced or unbalanced intensities.

A set of four standard benchmark images are used for evaluating the performance of CED algorithm including one binary image (*Circuits*) and three grayscale images (*Coins*, *Lena* and *Cameraman*). Also, the CED algorithm is compared with a set of popular edge detection algorithms such as Prewitt [1], Canny [2], etc. The general settings of CED algorithm are as follows:

1) The maximum number of generations is set to 50.
2) The learning algorithm is L_{RP} with $\alpha = \beta = 0.5$ as reward and penalty parameters.
3) The threshold value of grayscale images θ determines empirically from the range of [15,30].

Experiments are divided into five parts including: binary edge detection, grayscale edge detection, grayscale edge detection of noisy images, performance evaluation of grayscale edge detection of noisy images and quantitative comparison study of normal and noisy grayscale images. In the following, these experiments will be presented.

4.1. Experiment 1: edge detection of binary images

Fig. 5 shows the edge detection result of Circuits test image for different algorithms. The test image includes a large number of horizontal and vertical lines and a few dotted points. The outputs of Prewitt and Canny algorithms are almost the same, yet Prewitt algorithm preserves a little information about the dotted points. The CED algorithm utilizes a proper neighborhood type by efficient switching between von Neumann and Moore neighborhood types. This algorithm not only could preserve the dotted points, but it also easily detects both horizontal and vertical edges.

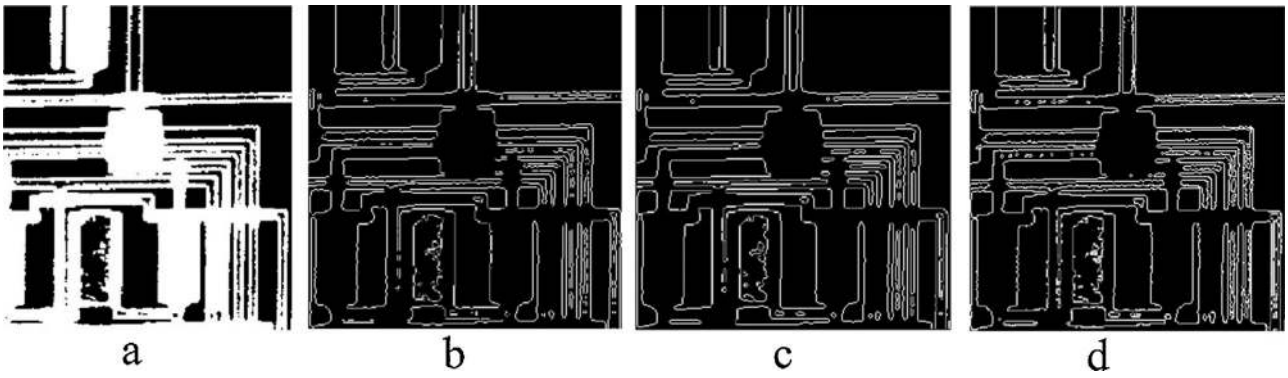

Fig. 5. (a) The Circuits test image, (b) Prewitt algorithm, (c) Canny algorithm and (d) CED algorithm.

From the window pattern view of neighborhood type, the square-like, 9-cell Moore neighborhood fits best for horizontal edges while the plus-like, 5-cell von Neumann neighborhood fits best for vertical edges. In Circuits image, if the edge is horizontal then von Neumann neighborhood type will be desirable and otherwise, if the edge is vertical then Moore neighborhood type will be desirable. In CED algorithm, a CLA is responsible for selecting the neighborhood type of CA while the gradient of image intensity may vary in different parts of the image. After selecting the neighborhood type of each cell, the CA rules will deploy on the input image and consequently the output image will be formed. Finally, the reinforcement signal will be calculated and will deploy in a way that membered LA of a same neighborhood will gradually learn a unified local neighborhood type.

4.2. Experiment 2: edge detection of grayscale images

Fig. 6 shows the detected edges of different edge detection algorithms for *Coins* test image. From Fig. 6(b), not only is Prewitt's output extremely pale and plain, but it also misses most of the key edges. Moreover from Fig. 6(c), Canny's output misses some certain edges that will lead to produce a meaningless image. Fig. 6(d) shows the output generated by the CED algorithm. We observe that a threshold value between [15,30] balances edge details and clarity. The CED algorithm produces connected, clean and meaningful edges. Moreover, there is a tradeoff between accuracy and purity of the resulted image of CED algorithm that is controlled by the threshold value. As you can see there is a little amount of noise in the edge map of Fig. 6(d). If the threshold value of the CED algorithm is increased, the amount of noise in the *coins* will be reduced; but in this condition the algorithm will miss some principal edges. The reversed process of this phenomenon is also viable. So, the threshold value should properly set prior to the execution of edge detection process.

Fig. 7 shows the output of different edge detection methods for *Lena* test image. As shown in Fig. 7(b), Prewitt's output almost loses a portion of all edges. Also in Fig. 7(c), Canny's output shows a small number of pixels with no real connectivity especially around *Lena*'s hair. In Fig. 7(d), unlike other algorithms, the CED algorithm draws continuous, dominant and significant edges around *Lena*'s hairs, eyes, nose and lips.

The applied results of Prewitt, Canny and CED algorithms on *Cameraman* test image are shown in Fig. 8(b)–(d). From Fig. 8(b), the detection results of Prewitt method are not satisfying and it eliminates most of the buildings edges. Fig. 8(c) presents the result of classical Canny method. As well as Prewitt method, Canny method loses the buildings edges that placed in depth of Fig. 8(c). Unlike other edge detection algorithms, CED clearly draws solid edges of buildings and *cameraman*'s face and coat in Fig. 8(d).

4.3. Experiment 3: edge detection of grayscale noisy images

In this section the application of CED algorithm for noisy image edge detection is presented. Figs. 9–11 are the results of edge detection process of *Coins*, *Lena* and *Cameraman* test images which are corrupted by the salt and pepper impulse noise with probability of $p = 0.005$.

The edge detection result of *Coins* image is showed in Fig. 9. The CED algorithm (Fig. 9(d)) outperforms Prewitt and Canny methods when the noise is applied to the *Coins* image. This algorithm could maintain the main structure of edge information while processing the noisy input image. However, Prewitt and Canny algorithms produce irrelevant edges and thicken the noisy parts of the output image compare to Fig. 9(a).

The resulting outputs of noisy *Lena* test image from the various algorithms are displayed in Fig. 10. It is apparent that Prewitt algorithm (Fig. 10(b)) almost produces an incomprehensible output image with the presence of salt and pepper noise in its area. Also in Fig. 10(c), Canny algorithm amplifies the noisy parts

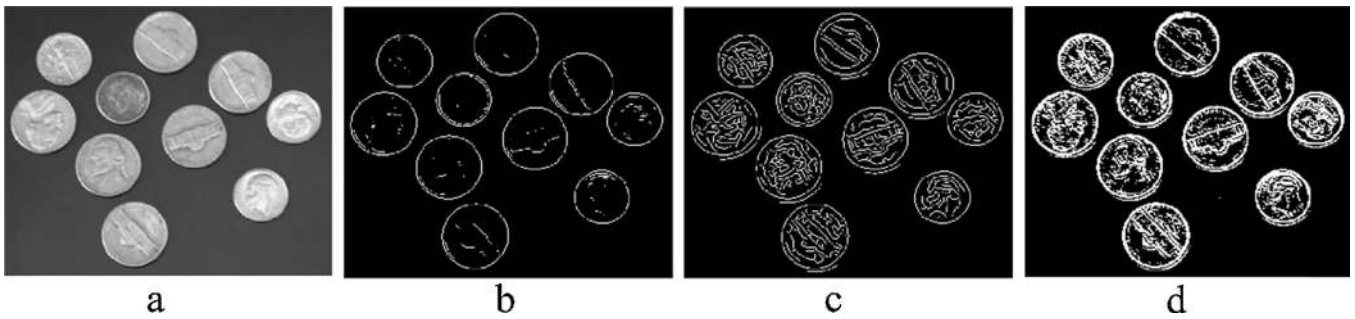

Fig. 6. (a) The *Coins* test image, (b) Prewitt algorithm, (c) Canny algorithm and (d) CED algorithm.

Fig. 7. (a) The *Lena* test image, (b) Prewitt algorithm, (c) Canny algorithm and (d) CED algorithm.

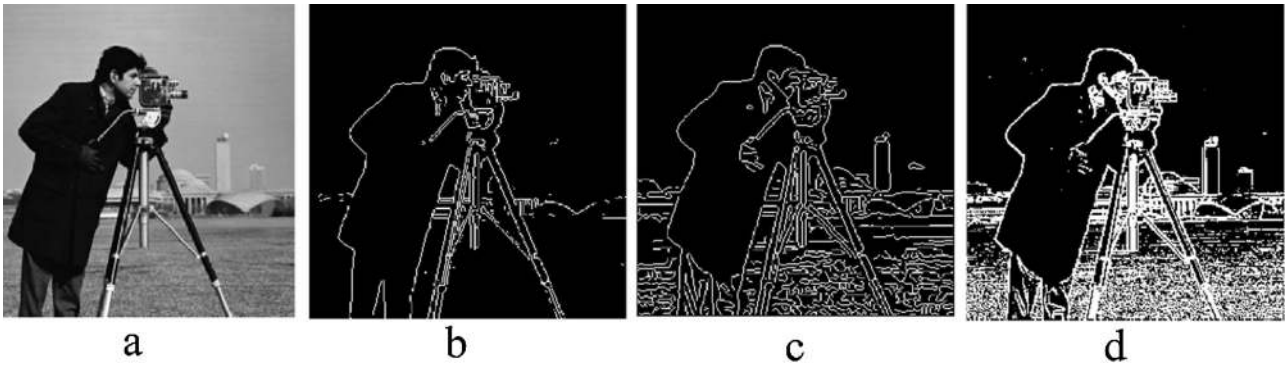

Fig. 8. (a) The *Cameraman* test image, (b) Prewitt algorithm, (c) Canny algorithm and (d) CED algorithm.

of the produced edge map. On the other hand in Fig. 10(d), since the CED algorithm could extract enough information from the CA neighborhood, there is sufficient information available for computing the image edges.

Fig. 11 shows the edge detection results of *Cameraman* test image with the presence of salt and pepper noise with probability of $p = 0.005$. Fig. 11(b) and (c) shows the result of applying Prewitt and Canny algorithm to the *Cameraman* image which corrupted by salt

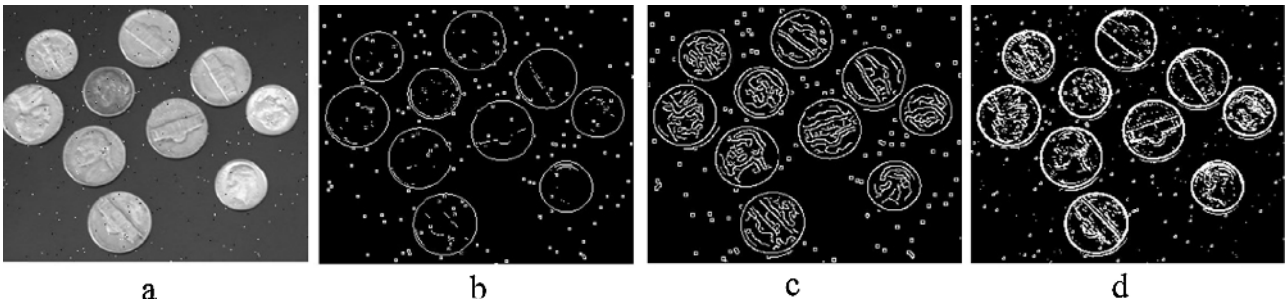

Fig. 9. (a) The *Coins* test image corrupted by salt and pepper impulse noise with probability of $p = 0.005$, (b) Prewitt algorithm, (c) Canny algorithm and (d) CED algorithm.

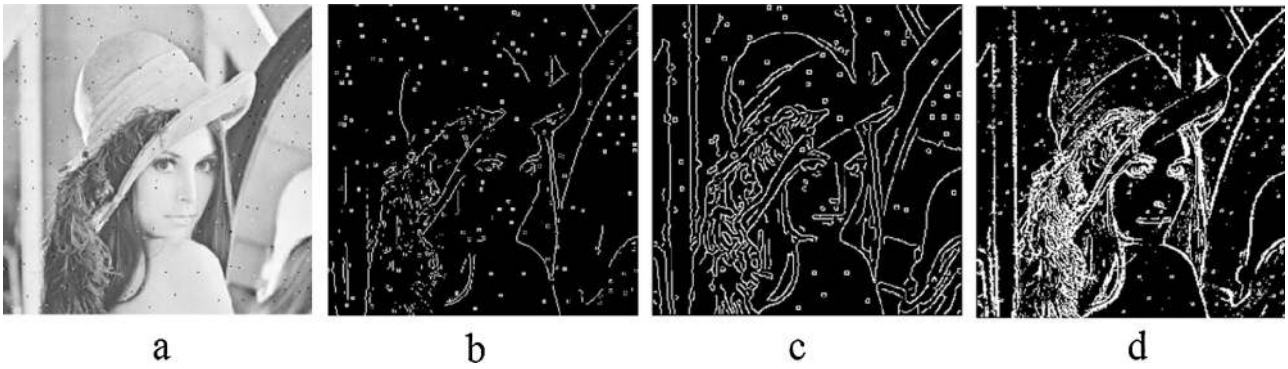

Fig. 10. (a) The *Lena* test image corrupted by salt and pepper impulse noise with probability of $p = 0.005$, (b) Prewitt algorithm, (c) Canny algorithm and (d) CED algorithm.

Fig. 11. (a) The *Cameraman* test image corrupted by salt and pepper impulse noise with probability of $p = 0.005$, (b) Prewitt algorithm, (c) Canny algorithm and (d) CED algorithm.

and pepper noise (Fig. 11(a)). In compare with Prewitt algorithm the Canny algorithm is much more successful in *Cameraman* test image. From Fig. 11(d), the CED is more effective while finding the edges of this test image. Applying the edge detection rule produces an edge map that preserves both magnitude and quality of edges. Also, CED could easily detect directed edges while minimizing the effects of noise.

4.4. Experiment 4: performance evaluation of edge detection of grayscale noisy images

4.4.1. Mean square error

Mean square error (MSE) is an image quality assessment metric. MSE calculates the average difference between original and edge detected image. So, the lower MSE means lesser error between the original image and edge detected one. For original image I and edge detected image J, MSE is calculated through (9):

$$\mathrm{MSE} = \frac{1}{m \times n} \sum_{i=1}^{m} \sum_{j=1}^{n} \left(I_{i,j} - J_{i,j}\right)^2 \tag{9}$$

where in (9), m and n are height and width of the images, respectively.

4.4.2. Peak signal to noise ratio

Peak Signal to Noise Ratio (PSNR) is another image quality assessment metric that is expressed in terms of decibel (dB) scale. PSNR is the ratio between the original image and the distortion signal in an image. A higher PSNR indicates the construction of a higher quality image. The PSNR is calculated based on MSE by (10):

$$\mathrm{PSNR} = 10\log_{10}\left(\frac{R^2}{\mathrm{MSE}}\right) \tag{10}$$

where in (10), $R = 255$ is the maximum variation for an 8-bit grayscale input image and MSE is calculated by (9).

4.4.3. Discussion of MSE and PSNR

The MSE and PSNR of noisy images of *Coins*, *Lena* and *Cameraman* corrupted by salt and pepper impulse noise with probability of $p = 0.005$ are reported for Prewitt [1], Canny [2], Sobel [22], Robert [23], Laplacian of Gaussian as LoG [24] and CED algorithms in Table 1. In edge detection algorithms, a lower MSE demonstrates that the produced edge map contains strong and even weak edge points. Moreover, a higher PSNR demonstrates a higher ratio of signal (original image) to noise (produced edge map). So, an edge detection algorithm like CED that has lower MSE and higher PSNR is a accurate and reliable choice. The inverse relation of MSE and PSNR in (9) and (10) proves this relationship.

4.5. Experiment 5: quantitative comparison study

In this section, CED algorithm is compared with several other edge detection methods. Having a quantitative comparison, Baddeley's Delta Metric (BDM) [25] is used as a numeric metric. The BDM is a parameter that is used to calculate the dissimilarity measure between two binary images. The k-BDM ($0 < k < \infty$) is defined as follows:

$$\Delta^k(I_a, I_b) = \left[\frac{1}{|P|}\sum_{p \in P}\left|g(d(p, I_a)) - g(d(p, I_b))\right|^k\right]^{1/k} \tag{11}$$

where in (11), I_a and I_b are two binary images with equal dimensions $N \times M$, $p = \{1,\ldots,N\} \times \{1,\ldots,M\}$ is the set of position. Also, $g(.)$ is a concave increasing function that is used for weighting and the distance between position p and the closest edge point of the set I_a is defined by $d(p,I_a)$.

The experiments were performed on 10 types of Berkeley Segmentation Data Set (BSDS) images [26] including human, face, airplane, tree, garden, island, building, zebra, panda and sea-sky. Also in this experiment, we set $k = 2$ and $g(x) = min(c, x)$ where $c = \sqrt{M^2 + N^2}$.

Table 1
MSE and PSNR in decibel of Prewitt, Canny, Sobel, Robert, LoG and CED algorithms while edge detection of *Coins*, *Lena* and *Cameraman* corrupted by salt and pepper impulse noise with probability of $p = 0.005$.

Algorithm	Image					
	Coins		Lena		Cameraman	
	MSE	PSNR (dB)	MSE	PSNR (dB)	MSE	PSNR (dB)
Prewitt	1380	6.7635	3493	2.7320	1804	5.6016
Canny	1379	6.7690	3491	2.7346	1802	5.6056
Sobel	1380	6.7657	3549	2.6635	1804	5.6011
Robert	1380	6.7655	3550	2.6626	1804	5.6008
LoG	1379	6.7690	3548	2.6649	1804	5.6025
CED	**1377**	**6.7737**	**3488**	**2.7382**	**1800**	**5.6120**

Table 2
The comparison of CED with other edge detection methods in normal images via BDM.

BSDS image #	Prewitt [1]	Canny [2]	Sobel [22]	Robert [23]	LoG [24]	CFED [27]	GEDT [28]	AED [29]	NGED [30]	CED
37073	8.20	10.62	7.98	7.95	8.28	32.84	16.52	15.25	14.39	**7.61**
69015	11.32	9.24	11.35	15.50	9.84	25.09	10.55	19.91	14.90	**9.14**
46076	25.23	27.03	25.22	25.47	26.53	33.36	33.11	26.94	**23.35**	26.37
102061	40.87	60.89	40.85	41.23	53.28	68.48	75.41	65.64	41.18	**40.61**
143090	20.47	24.76	20.48	20.86	23.80	44.54	31.57	36.68	34.11	**19.58**
147091	29.21	29.65	29.12	29.15	29.18	37.59	29.09	31.54	**26.27**	29.85
181079	27.23	17.82	28.05	42.54	16.53	35.36	19.44	18.50	**15.81**	17.21
189080	15.68	15.78	15.67	15.85	15.62	30.34	20.14	15.68	15.76	**15.44**
253027	34.50	22.42	34.35	35.91	27.12	**20.81**	28.23	24.79	21.23	23.96
86016	25.55	23.95	25.09	28.44	23.77	36.58	24.59	36.23	31.95	**23.40**
Sum	238.26	242.16	238.16	262.9	233.95	364.99	288.65	291.16	238.95	**213.17**

Table 3
The comparison of CED with other edge detection methods in noisy images corrupted by Gaussian noise with zero mean and variance 25 based on BDM.

BSDS image #	Prewitt [1]	Canny [2]	Sobel [22]	Robert [23]	LoG [24]	CFED [27]	GEDT [28]	AED [29]	NGED [30]	CED
37073	13.65	14.14	13.59	13.58	13.65	42.98	16.52	19.27	18.25	**13.53**
69015	9.95	9.80	9.90	10.20	9.80	24.79	10.90	18.93	14.70	**9.71**
46076	30.01	32.61	30.03	31.79	32.45	43.76	33.10	30.21	**30.04**	31.61
102061	74.71	74.85	75.72	74.68	74.67	85.18	75.36	68.07	**63.01**	74.25
143090	30.49	31.57	30.46	30.39	30.83	57.99	31.52	45.35	42.11	**30.41**
147091	28.18	28.55	28.12	29.01	28.39	49.87	29.01	**24.81**	29.22	28.92
181079	18.89	18.90	18.74	19.83	18.74	38.76	19.16	19.04	**16.32**	19.16
189080	20.06	19.68	20.15	21.38	19.52	32.76	19.94	**16.33**	20.92	19.93
253027	13.83	13.76	13.97	14.17	13.83	24.22	13.78	32.59	30.78	**13.69**
86016	25.24	25.11	25.25	25.14	24.96	41.05	24.52	36.92	33.78	**24.34**
Sum	13.65	14.14	13.59	13.58	13.65	42.98	16.52	19.27	18.25	**13.53**

In this experiment, the results of CED algorithm are compared with Prewitt [1], Canny [2], Sobel [22], Robert [23], Laplacian of Gaussian as LoG [24], competitive fuzzy edge detection as CFED [27], gravitational edge detection based on triangular norms using product *t*-norm as GEDT [28], Edge detection using ant algorithms as AED [29] and new gravitational edge detection as NGED [30]. The results of this experiment on normal images and noisy images corrupted by Gaussian noise with zero mean and variance 25 based on BDM are reported in Tables 2 and 3, respectively. The best results are highlighted in boldface. The results indicate that the CED algorithm performs better than other edge detection algorithms at least for six out of ten normal images and five out of ten noisy images while most of other edge detection methods have similar results. Also, the results of noisy images are similar to normal images for CED algorithm.

4.6. Discussion of results

From Figs. 6–8, the CED algorithm easily detects directed and undirected edges and draws continues edge map from them. Also, from Figs. 9–11, CED algorithm robustly detects the edges and successfully removes further noise without deteriorating the quality of edges. Moreover, the MSE and PSNR of the CED algorithm that are reported in Table 1 numerically justify the acute edge detection power of CED algorithm. Finally, Tables 2 and 3 demonstrate that the CED algorithm outperforms the other edge detection methods on BSDS while evaluating the BDM parameter.

The CED algorithm may produce some unnecessary points in the edge map due to the high level of transparency of test images. This algorithm proposes a dynamic edge detection strategy that utilizes CLA cells to adaptively set the neighborhood type of relative pixel intensities. The CED algorithm consists of two main components:

1) A CA which is composed of a grid of cells similar to the input image size. CA compute the edge map of image.
2) A CLA which determines the neighborhood type of the CA. Each learning automaton of CLA progressively learns from its neighbors toward creating a smooth and clear edge map.

The following are four main characteristics of the CED algorithm:

1) Calculating the image edge map with regard to intensity changes.
2) Using an image specific threshold to determine the optimal threshold value for different images.
3) Utilizing an adaptive neighborhood strategy that produces smooth edges.
4) Using the previous experiences while selecting the neighborhood type.

5. Conclusion

Edge detection is one of the key applications of machine vision. Two main weaknesses of traditional edge detection algorithms are slowness and edge loss. The popular edge detection approaches use multiple stages to detect edges of image. In this paper we propose the cellular edge detection (CED) algorithm. This algorithm works based on parallel cells of CA combined with CLA which speed up the edge detection process. Moreover, the smoothing and noise filtering properties of this algorithm flats the edges of image. Also, to prevent the edge loss, we apply a deterministic CA rule to images where the neighborhood type can change adaptively.

The fact that no learning automaton has a complete picture of edge detection process is the interesting part of CED algorithm. So, every cell in CLA operates with limited information about the image and indirectly effects other cells while calculating their corresponding reinforcement signals. This makes sense when CED algorithm simply draws connected edges from horizontal and vertical gradient beams of image.

In order to fully investigate the pros and cons of CED and other algorithms, a set of experiments for binary and grayscale images are conducted. In both binary and grayscale images, the CED algorithm accurately detects edges and merely extracts unnecessary details. Furthermore, the CED algorithm presents superior numerical results in terms of MSE, PSNR and BDM performance metrics while comparing with other edge detection methods. In this work, the optimal value for CA's threshold is choosed empirically based on the input image while designing an adaptive scheme to automate the full process of the CED algorithm can be considered as a future work.

References

[1] Prewitt JMS. Object enhancement and extraction, vol. 75. New York: Academic Press; 1970.
[2] Canny J. A computational approach to edge detection. IEEE Trans Pattern Anal Mach Intell 1986;(6):679–98.
[3] Yüksel ME. Edge detection in noisy images by neuro-fuzzy processing. AEU – Int J Electron Commun 2007;61(2):82–9.
[4] Duan S, Hu X, Wang L, Gao S, Li C. Hybrid memristor/RTD structure-based cellular neural networks with applications in image processing. Neural Comput Appl 2013;25(2):291–6.
[5] Nabizadeh S, Faez K, Tavassoli S, Rezvanian A. A novel method for multi-level image thresholding using particle swarm optimization algorithms. In: 2010 2nd international conference on computer engineering and technology (ICCET), vol. 4. 2010. p. V4-271–-280.
[6] Cuevas E, González M, Zaldívar D, Pérez-Cisneros M. Multi-ellipses detection on images inspired by collective animal behavior. Neural Comput Appl 2013;24(5):1019–33.
[7] Hasanzadeh M, Meybodi MR, Ebadzadeh MM. Adaptive cooperative particle swarm optimizer. Appl Intell 2013;39(2):397–420.
[8] Hasanzadeh M, Sadeghi S, Rezvanian A, Meybodi MR. Success rate group search optimiser. J Exp Theor Artif Intell 2014:1–17.
[9] Hasanzadeh M, Meybodi MR. Grid resource discovery based on distributed learning automata. Computing 2014;96(9):909–22.
[10] Hasanzadeh M, Meybodi MR. Distributed optimization grid resource discovery. J Supercomput 2015;71(1):87–120.
[11] Rezvanian A, Rahmati M, Meybodi MR. Sampling from complex networks using distributed learning automata. Phys A: Stat Mech Appl 2014;396:224–34.
[12] Rezvanian A, Meybodi MR. Finding maximum clique in stochastic graphs using distributed learning automata. Int J Uncertain Fuzziness Knowl Based Syst 2015;23(01):1–31.
[13] Mousavian A, Rezvanian A, Meybodi MR. Cellular learning automata based algorithm for solving minimum vertex cover problem. In: 2014 22nd Iranian conference on electrical engineering (ICEE). 2014. p. 996–1000.
[14] Sadeghi S, Rezvanian A, Kamrani E. An efficient method for impulse noise reduction from images using fuzzy cellular automata. AEU – Int J Electron Commun 2012;66(9):772–9.
[15] Hernandez G, Herrmann HJ. Cellular automata for elementary image enhancement. Graph Models Image Process 1996;58(1):82–9.
[16] Rosin PL. Training cellular automata for image processing. IEEE Trans Image Process 2006;15(7):2076–87.
[17] Rosin PL. Image processing using 3-state cellular automata. Comput Vis Image Underst 2010;114(7):790–802.
[18] Fotouhi M, Rohban MH, Kasaei S. Skin detection using contourlet-based texture analysis. In: Fourth international conference on digital telecommunications, 2009. ICDT'09. 2009. p. 59–64.
[19] Narendra KS, Thathachar M. Learning automata: a survey. IEEE Trans Syst Man Cybern 1974;(4):323–34.
[20] Wolfram S. Cellular automata and complexity: collected papers, vol. 152. Reading: Addison-Wesley; 1994.
[21] Beigy H, Meybodi M. A mathematical framework for cellular learning automata. Adv Complex Syst 2004;4(3–4):295–320.
[22] Sobel I, Feldman G. A 3×3 isotropic gradient operator for image processing. In: Presented at the Stanford artificial intelligence project (SAIL). 1968.
[23] Roberts LG, Thesis Machine perception of three-dimensional solids. Massachusetts Institute of Technology; 1963.
[24] Torre V, Poggio TA. On edge detection. IEEE Trans Pattern Anal Mach Intell 1986;8(2):147–63.
[25] Baddeley AJ. An error metric for binary images. In: Robust computer vision: quality of vision algorithms. Germany; 1992. p. 59–78.
[26] Martin D, Fowlkes C, Tal D, Malik J. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In: Eighth IEEE international conference on computer vision, 2001. ICCV 2001. Proceedings, 2001, vol. 2. 2001. p. 416–23.
[27] Liang LR, Looney CG. Competitive fuzzy edge detection. Appl Soft Comput 2003;3(2):123–37.
[28] Lopez-Molina C, Bustince H, Fernandez J, Couto P, De Baets B. A gravitational approach to edge detection based on triangular norms. Pattern Recognit 2010;43(11):3730–41.
[29] Nezamabadi-pour H, Saryazdi S, Rashedi E. Edge detection using ant algorithms. Soft Comput 2005;10(7):623–8.
[30] Deregeh F, Nezamabadi-pour H. A new gravitational image edge detection method using edge explorer agents. Nat Comput 2013;13(1):65–78.