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/© 2015 Elsevier GmbH. All rights reserved.

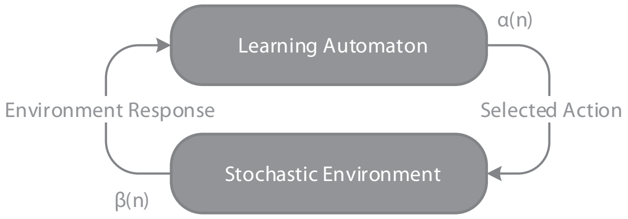

Fig. 1. The interaction sequence between learning automaton and environment where $\alpha(n)$ and $\beta(n)$ are the selected action of learning automaton and environment response of the selected action, respectively.

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.

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

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

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 $\left\{Z^D, \phi, N, F\right\}$ 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).

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

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

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.