International Journal of Computer Applications (0975 – 8887)
Volume 9– No.4, November 2010

A Novel Method of Edge Detection using Cellular Automata

Tapas Kumar
Dept. of Information Technology
Lingaya's University, Faridabad

G. Sahoo
Dept. of Information Technology
B.I.T. Mesra, Ranchi

ABSTRACT

Edge detection is one of the most commonly used operations in image analysis. Several edge detectors have been proposed in literature for enhancing and detecting of edges. In this paper a new and optimal approach of edge detection based on Cellular Automata (CA) has been proposed. The idea is simple but effective technique for edge detection that greatly improves the performances of complicated images. The comparative analysis of various image edge detection methods is presented and shown that cellular automata based algorithm performs better than all these operators under almost all scenarios.

General Terms:

Edge Detection, Image Processing, Computation Time.

Keywords:

Cellular Automata, Neighborhood Function, Gradient Method, Mask Operator.

1. INTRODUCTION

Because of its significant importance in many research areas, edge detection has received much attention during the past two decades. Since, the edge is a prominent feature of an image; it is a vital foundation for image processing, computer vision, image understanding system and pattern recognition. The detection results benefit applications such as image enhancement, recognition, morphing, restoration, registration, compression, retrieval, hiding etc. [1]. Edge detection constitutes a crucial step in most of the computer vision applications. It typically occurs on the boundary between two different regions in an image. An edge in an image is a significant local change in the image intensity [2].

Although many edge-detection evaluation methods have been developed in the past years, however this is still a challenging and unsolved problem. Canny first presented the well-known three criteria of edge detectors: good detection, good localization, low spurious response and showed that the optimal detector for an isolated step edges [5]. There are many ways to perform edge detection. However, the most may be grouped into three categories, Gradient (Approximations of the first derivative), Laplacian (Zero crossing detectors) and Image approximation algorithms are usually used for edge detection [12].

2. VARIOUS EDGE DETECTION TECHNIQUES

The early days of works on edge detection are done by Sobel and Roberts [7]. Their detection methods are based on simple intensity gradient operators. Later on, much of the research works have been devoted to the development of detectors with good detection performance as well as good localization performances. A different edge detection method i.e. Prewitt, Laplacian, Roberts and Sobel uses different discrete approximations of the derivative function. For comparison purposes, we have used here four most frequently used edge detection methods namely Sobel edge detection, Prewitt edge detection, Canny edge detection and Roberts edge detection. The details of methods are as follows:

2.1 The prewitt edge detector

The prewitt edge detector is an appropriate way to estimate the magnitude and orientation of an edge. The prewitt operator is limited to 8 possible orientations; however experience shows that most direct orientation estimates are not much more accurate [6]. This gradient based edge detector is estimated in the 3x3 neighborhood pixels edge gradient operator described by the convolution masks as shown in Figure 1.

-1	0	+1
-1	0	+1
-1	0	+1

+1	+1	+1
0	0	0
-1	-1	-1

Figure 1. Masks use for gradient operation on prewitt operations

The prewitt square root edge gradient is given by

$$G(x, y) = \{ \mid G_R (x, y) \mid^2 + \mid G_C (x, y) \mid^2 \}^2$$

with

$$G_R (x, y) = 1/ (K+2) [(A_2 + KA_3 + A_4) - A_0 + KA_7 + A_6]$$

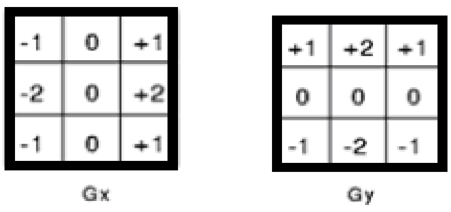
and

$$G_C (x, y) = 1/ (K+2) [(A_2 + KA_1 + A_2) - A_0 + KA_5 + A_4]$$

where K=1. In this formulation the row and column gradients are normalized to provide unit gain positive weighted and unit negative weighted about a separated edge positions. This mask is more sensitive to horizontal and vertical edges than diagonal edges.

2.2. The Sobel Edge Detection

The Sobel operator performs a 2-D spatial gradient measurement on an image and highlighting regions of high spatial frequency that correspond to edges. Typically it is used to find the approximate absolute gradient magnitude at each point in an input grayscale image [6]. In theory at least, the operator consists of a pair of 3x3 convolution kernels as shown in Figure 2.



-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

Figure 2. Masks use for gradient operation on sobel operations

Sobel edge detectors masks detect vertical and horizontal edges separately and these directional edges are combined finally. Like the other gradient operator, G_x and G_y can be implemented using convolution masks. Two masks are convoluted with image separately. The magnitude and direction of edge is calculated by using convolution results of two masks. These are as follows

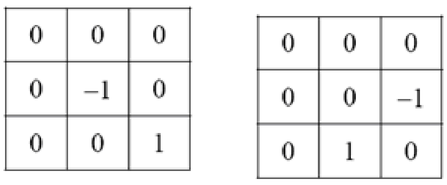
$$G = \sqrt{G^2 x + G^2 y} \quad ,$$

$$\mathrm{Arg}(G) = \tan^{-1} (|Gy| / |Gx|).$$

Note that this operator stress on pixels that are closer to the centre of the mask. The sobel operator is one of the most commonly used edge detector. This method can detect diagonal edges better as compare to Prewitt method [7].

2.3. The Roberts Edge Detection

The Roberts operator performs a 2-D spatial gradient measurement on an image and emphasizes regions of high spatial frequency that correspond to edges. Typically it is used to find the approximate absolute gradient magnitude at each point of an input grayscale image [8]. In theory at least, the operator consists of a pair of 3x3 convolution kernels as shown in Figure 3.



0	0	0
0	−1	0
0	0	1

0	0	0
0	0	−1
0	1	0

Figure 3. Masks use for gradient operation on Robert operations

Diagonal edge gradients can be obtained by forming running difference of diagonal pairs of pixels. The Robert crosses difference operators are defined in magnitude form which is as follows:

$G(x, y) = | G_1 (x, y) + G_2 (x, y) |$ and in square root means form it can be defined as:

$G(x, y) = \{| G_1 (x, y) |^2 + | G_2 (x, y) |^2\}^2$

where

$G_1 (x, y) = F (x+1, y+1) – F(x, y)$

and

$G_2 (x, y) = F (x, y+1) – F(x+1, y)$.

2.4 The Canny Edge Detection

The Canny edge detection algorithm is known to many as the optimal edge detector. Canny's intention was to enhance the edge detectors that already existed at the time he started his work. The first and most obvious criterion is low error rate. It is important that edges occurring in images should not be missed and that there be no responses to non-edges. The second criterion is that the edge points be well localized. In other words, the distance between the edge pixels as found by the detector and the actual edge is to be at a minimum. A third criterion is to have only one response to a single edge. Based on these criteria, the canny edge detector first smoothes the image to eliminate and noise. It then finds the image gradient to highlight regions with high spatial derivatives. The algorithm then tracks along these regions and suppresses any pixel that is not at the maximum (non-maximum suppression). Edges will occur at points the where the gradient is at a maximum. Therefore, all points not at a maximum should be suppressed. In order to do this, the magnitude and direction of the gradient is computed at each pixel. Then for each pixel check if the magnitude of the gradient is greater at one pixel's distance away in either the positive or the negative direction perpendicular to the gradient. If the pixel is not greater than both, suppress it. The gradient array is now further reduced by hysteresis. Hysteresis is used to track along the remaining pixels that have not been suppressed. Hysteresis uses two thresholds and if the magnitude is below the first threshold T1, it is set to zero (made a non edge). If the magnitude is above the high threshold, it is made an edge. And if the magnitude is between the thresholds T1 & T2, then it is set to zero unless there is a path from this pixel to a pixel with a gradient above T2[9].

Normally, in the above discussed algorithm the image is convolved with 4 masks, calculating horizontal, vertical and both diagonal gradients (the masks used are similar to the Prewitt or Sobel masks). The direction producing the largest result at each pixel is used to determine magnitude and direction of the gradient. After studying all traditional methods of edge detection, it has been analyzed that for these situations, a new algorithm is needed which is optimal and meets the following three criteria:

- Good Detection: The algorithm should mark as many real edges in the image as possible.
- Good Localization: Marked edges should be as close as possible to the edge in the real scene.
- Minimal Response: A given edge in the image should only be marked once, and where possible, image noises should not create false edges.

Cellular Automata can cater to this need and fulfill the desired conditions as discussed previously.

3. CELLULAR AUTOMATA

A cellular automaton is used in computer science as a discrete model for implementing algorithms. The use of cellular automata (CA) in image processing and graphical applications has received some attention over the past few years [10]. Cellular automata techniques appear as a natural tool for image processing due to their local nature and simple parallel computing implementation. The main cellular automata algorithm for k gray levels of digital images is on the basis of bi-dimensional cellular automata, which is discrete dynamical system formed by a finite number of identical objects are called cells, and arranged uniformly in a two-dimensional space. Each cell is endowed with a state, belonging to a finite state set, which changes at every discrete step of time according to a rule, called local transition function. More precisely, a CA can be defined as a 4 tuples:

$$CA = (I, N, V, f),$$

where I is the cellular space formed by a two-dimensional array of r × c cells:

$$I = \{(a, b), 1 \leq a \leq r, 1 \leq b \leq c\}.$$

Let I denote the set of integer, A 2-D cellular space is a 4-tuple (I× I is a set of cartesian product of two integer sets , V is a set of cellular states ,N is the type of neighborhood ,and f is the local transition function from V^n to V [11]. The relevant neighborhood function is a function from $I \times I \rightarrow 2^{I \times I}$ defined by

$$g(\alpha) = \{ \alpha + \delta_1 ,\ \alpha + \delta_2 , \alpha + \delta_3 , \ldots\ldots, \alpha + \delta_n \},$$

for all $\alpha \in I \times I$, where δ_i (i= 1,2,3 ……………n) € I×I is fixed. The neighborhood state function of a cell α at time t is defined by [13]:

$$h^t(\alpha) = (v^t(\alpha + \delta_1) , v^t(\alpha + \delta_2), v^t(\alpha + \delta_3), \ldots\ldots v^t(\alpha + \delta_n)).$$

The neighborhood of a cell (a, b) is the set of cells whose states at time t determine the state of the cell (a, b) at time t + 1, by means of the local transition function. Depending on the process to be modeled, one can choose an appropriate neighborhood. Nevertheless, the traditional neighborhoods considered are the Von Neumann neighborhood (see Figure 5), and the Moore neighborhood (see Figure 6). Note that the main cell is also considered in its neighborhood. A neighborhood is defined by means of a finite set of indices $V = \{(\alpha i, \beta i), 1 \leq i \leq m\} \subset Z \times Z$, such that for every cell (a, b), its neighborhood, V (a, b), is the set of m cells given by

$$V (a, b) = \{(a + \alpha 1, b + \beta 1). . . (a + \alpha m, b + \beta m): (\alpha k, \beta k) \in V\}.$$

Note that for Moore neighborhood, we have $V = \{(0, 0) , (-1, 0) , (-1, 1) , (0, 1) , (1, 1) , (1, 0) , (1,-1) , (0,-1) , (-1,-1)\}$.

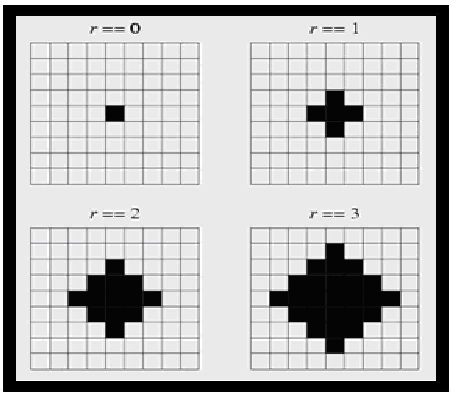


Figure 5 Von Neumann neighborhood

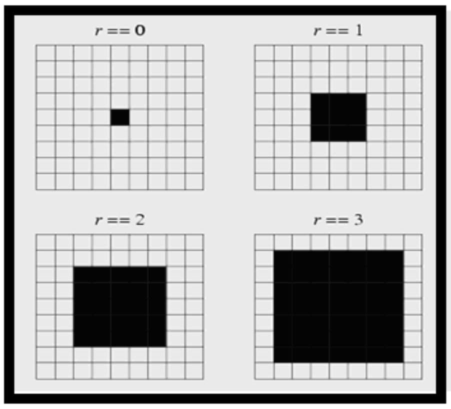


Figure 6. Moore neighborhood

A cellular automaton (CA) C is a k-dimensional array of automata. Each of the individual automata in the CA is said to occupy a cell in the CA. In the initial configuration of C, each automata is in its initial state and typically referred to as time step t = 0.

4. CELLULAR AUTOMATA BASED ALGORITHM

As stated previously, cellular automata techniques appear as a natural tool for image processing due to their local nature and simple parallel computing implementation. In this section, we present one algorithm and investigate its variation and effectiveness for processing of gray images. The algorithm will correspond to edge detection for grayscale images. The application of cellular automata techniques to real images will be presented, which together with the results will show the performance characteristics and comparison.

The main cellular automata algorithm for k gray levels of digital images is on the basis of a bi-dimensional cellular automata (IxI, V, N, f) with V = {0, 1, 2, …, k-1}, where k is a number of states, N is the type of neighborhood (e.g. n is total number of neighbors varies from 0 to 8), while the local transition function f is from V^n into V [10]. Let I denote the set of integers. V is the finite nonempty state set, N = (x1. . . xn) is the neighborhood, and f: $V^n \rightarrow V$ is the transition rule. Given a configuration c of the cells in the CA at a certain time t, the configuration c′ at time t + 1 for each cell x can be calculated as c′(x) = f(c (x1. . . xn)).
In such a 2D CA, specific neighborhoods can be defined. For example, the so-called Von Neumann neighborhood (4-neighborhood) and Moore neighborhood (8- neighborhood) for a cell $x_{i,j}$ is defined as:
$x_{i-1,j}$, $x_{i,j-1}$, $x_{i,j+1}$, $x_{i+1,j}$, and $x_{i-1,j}$, $x_{i,j-1}$, $x_{i,j+1}$, $x_{i+1,j}$, $x_{i-1,j+1}$, $x_{i+1,j-1}$, $x_{i+1,j+1}$, $x_{i-1,j-1}$, respectively.

The proposed hypothesis of cellular automata is shown in equations (1.1) and (1.2) which are as given below.

$$f((v^t(\alpha+\delta_1), v^t(\alpha+\delta_2), \ldots\ldots v^t(\alpha+\delta_n) = E(\alpha). \qquad \ldots(1.1)$$

$$\text{If } \sum_{j=0}^{k-1} N(C_J) = C_{target} \text{ and sum } (v^t(C_{target})) > k-1 \qquad \ldots(1.2)$$

$$= B(\alpha), \text{ Otherwise}$$

where

C_j is the class of the pixel values (states) in its neighborhood ($h^t(\alpha)$) for j=0, 1, 2,…………, m and $v^t(\alpha+\delta_1)$ € C_j.

N (Cj) is a number of neighbors of α which fall into class Cj
($v^t(C_{target})$) is a summation of $v^t(C_{target})$.

C_{target} is the majority class containing maximal number of neighbors.

($v^t(C_{target})$) denotes all of $v^t(\alpha+\delta_i)$ € C_{target}.

E (α) is the edge pixel value.

B (α) is the background pixel value.

k is a number of states.

The definition has been summarized in the viewing of the necessary requirement here. In our case, CA is used to enhance the images. Hence, only two-dimensional CA is considered where each cell represents one pixel in the image plane. Furthermore, it is assumed that the individual automata in each cell are identical, and hence one transition function can be defined for the CA as a whole.

4.1 CELLULAR AUTOMATA ALGORITHM FOR GRAY LEVEL EDGE DETECTION.

In digital image processing, each image is quantized into pixels. With gray-scale images, each pixel indicates the level of brightness of the image in a particular spot: 0 represents black, 255 represent white. An edge is an abrupt change in the brightness (gray scale level) of the pixels. Edge information for a particular pixel is obtained by exploring the brightness of pixels in the neighborhood of that pixel. If all of the pixels in the neighborhood have almost the same brightness, then there is probably no edge at that point. However, if some of the neighbors are much brighter than the others, then there is probably an edge at that point. Measuring the relative brightness of pixels in a neighborhood is mathematically analogous to calculating the derivative of brightness. Brightness values are discrete, not continuous, so we approximate the derivative function. The proposed conception of cellular automata for gray level images is given in equations (1.3) and (1.4) derived from equations (1.1) and (1.2) respectively, which are as follows:

$$f((v^t(\alpha+\delta_1), v^t(\alpha+\delta_2), \ldots\ldots\ldots v^t(\alpha+\delta_n) = 255 \qquad \ldots(1.3)$$

where E (α) = 255.

$$\text{If } \sum_{j=0}^{k-1} N(C_J) = C_{target} \text{ and sum } (v^t(C_{target})) > 255 \qquad \ldots(1.4)$$

$$= 0, \quad \text{otherwise}$$

The Pseudo-code of the cellular automata algorithm for edge reduction is as follows:

```
begin CA {Gray level edge detection}

    image size = I [M × N]

- Count the no. of set classes of pixels.

- Select the target pixel C target

   max = 0;
   sum = 0;
   radius=1;        // The radius of neighborhood//
   x=2;
   y=2;

     s= y1(x, y);

   for i= - radius:  radius

         for j= -radius:  radius

                  g= y1(x+i, y+j)

              if (g == n_data (C target ))

                 sum= sum + g
     end

     for k=1:5
                 if (g== n_data (k))

                  n (k) = n (k) + 1
      end

           begin
                 Max=0
                     for k=1:5
                          if (n (k) >= max)

                          Max=k

                        end

    if   ((max == C target) && (sum > 255))

                 I(x, y) =255

                 else

                 I(x, y) =0
   end.
```

5. EXPERIMENTAL RESULTS

Experiments were carried out over several 256×256 sizes of standard test images. All above said edge detection methods like Roberts operator, Sobel operator, Prewitt operator, Canny operator and the proposed edge detection method have been implemented on some standard test images using latest software technology viz, **.Net** and **MATLAB.** All the above mentioned methods have also been compared in terms of computing time.

Test Image -I:

The visual results of Lena image are shown in Figures 7 (b) - 7(f)

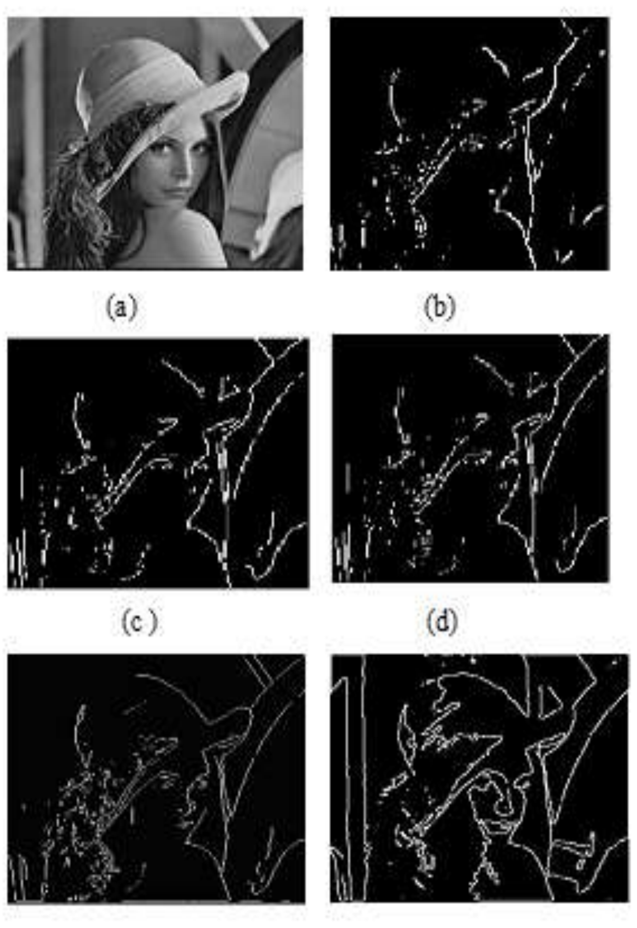


Figure 7. Edge detection results (a) Original image (b) Roberts method (c) Prewitt method (d) Sobel method (e) Canny method. (f) Cellular Automata method

The results clearly demonstrate that the Canny method has a better effect than Prewitt, Roberts and Sobel. A result produced by Cellular Automata has a better contour outline of edge and good detection effects in the top part of the hat as well as mouth as compared to Canny method. Roberts, Prewitt and Sobel methods give very weak and discontinuous edges. It also includes false edges whereas CA gives clean and almost continuous and true edges. From the above figures we can clearly see that the proposed method detects edges those are of very low intensity. Canny and Sobel both failed to detect these edges even though their methods were implemented on optimal threshold.

The visual results of Cameraman image are shown in Figures 8(b) –8(f)

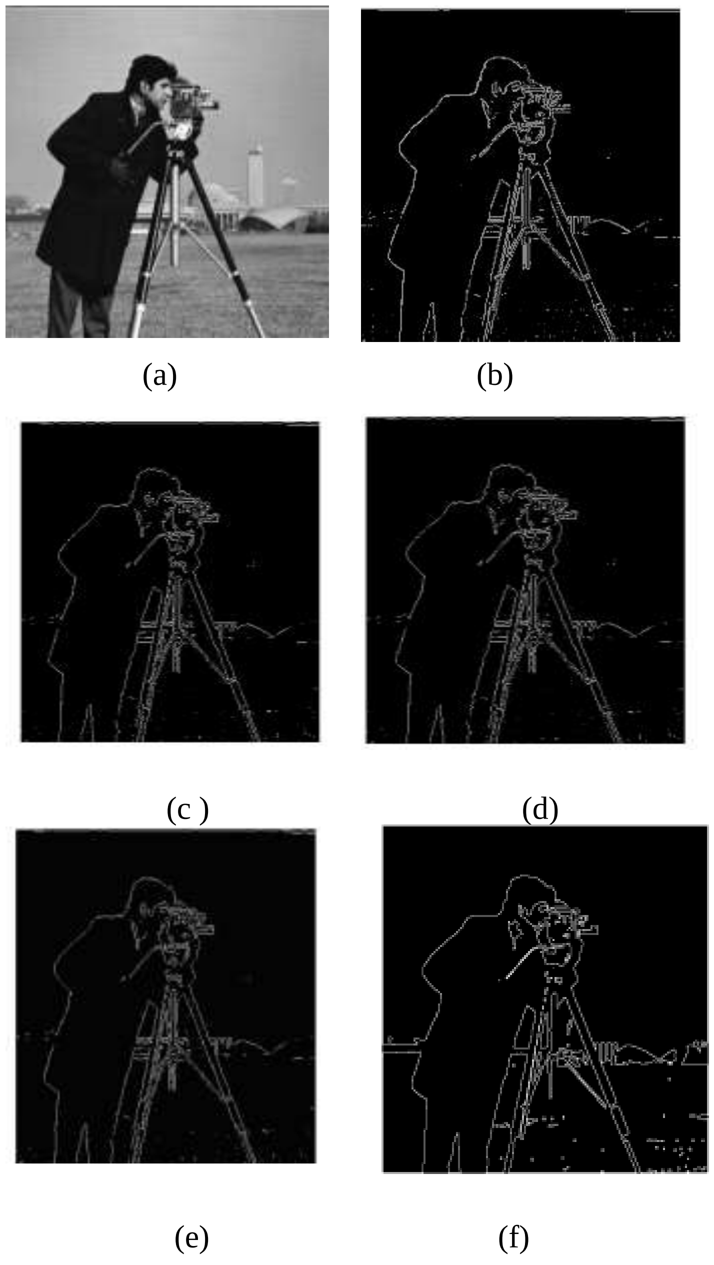


Figure 8. Edge detection results: (a) Original image (b) Roberts method (c) Prewitt method (d) Sobel method (e) Canny method (f) Cellular Automata method.

Canny method has a better effect than Prewitt, Roberts, Sobel method and continuity of edges are strong. CA has better curve outline of edges and have good detection effects on whole camera as well as body features as compared to Canny. Roberts, Prewitt and Sobel methods give very poor and discontinuous results and also include false edges whereas CA gives good, clean and almost continuous true edges. It is observed that the edges detected due to Canny method were not true. Further Sobel and Prewitt method failed to detect edges in the background.

Test Image -III:

The visual results of Rose image are shown in Figures 9(b) -9(f).

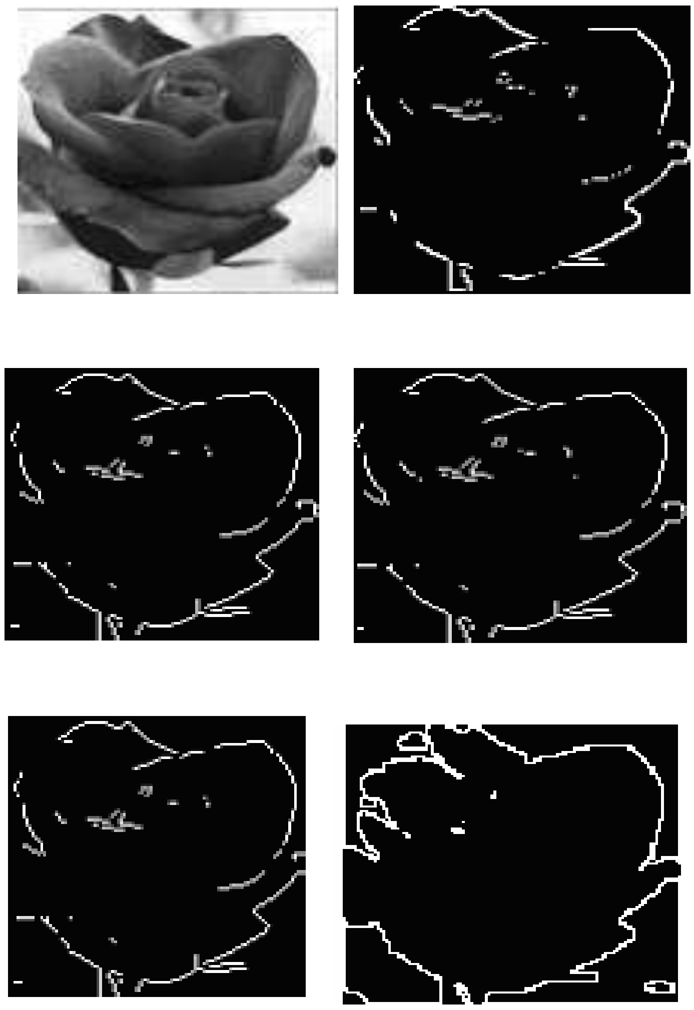


Figure 9. Edge detection results: (a) Original image (b) Roberts method (c) Prewitt method (d) Sobel method (e) Canny method (f) Cellular Automata method.

The visual results of Bird image are shown in Figures 10 (b) – 10 (f).

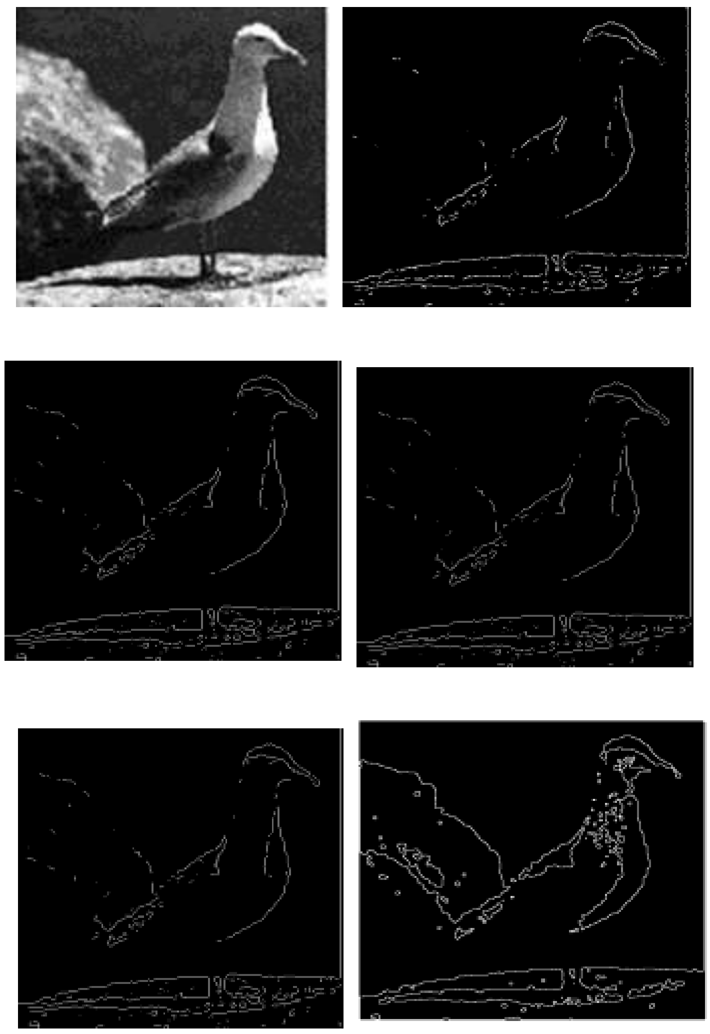


Figure 10. Edge detection results: (a) Original image (b) Roberts method (c) Prewitt method (d) Sobel method (e) Canny method (f) Cellular Automata method.

Cellular Automata has a better contour outline of edge and good detection effects as compared to Canny. It can be seen that Roberts and Prewitt methods give less accuracy and discontinuous results and also include false edges while CA gives clean and continuous edges. The new proposed method has provided better results. For time comparison point of view, the computing time has been computed and shown in table1.

Table 1. Computational time of various edge detection methods.

Computing Time						
Test Image(s)	**Size of Image**	**Edge Detection Methods**				
		Roberts	**Prewitt**	**Sobel**	**Canny**	**Cellular Automata**
I. Lena Image	256 × 256	**0.1804**	**0.2149**	**0.2306**	**0.2803**	**0.1406**
II. Cameraman Image	256 × 256	**0.2006**	**0.2105**	**0.2505**	**0.2906**	**0.1506**
III. Rose Image	256 × 256	**0.1406**	**.01703**	**0.2310**	**0.2508**	**0.1306**
IV. Bird Image	256 × 256	**0.2108**	**0.2204**	**0.2305**	**0.2508**	**0.1608**

All test results come out at computing time from range of 0.14 to 0.30 sec per image on a PC with a Pentium-IV CPU (400 MHZ), excluding I/O operations. Sobel based edge detection takes minimum time as compared to existing gradient method specifically Prewitt and Roberts method. The cellular automata based method has higher computational efficiency in minimum time. The graphical analysis of computation time has been depicted in Figures 10~13. Although Canny method is still good at detecting edges but still proposed method gave better results than Canny. The computation time of canny method is high. Finally we can say so new proposed technique has provided better results.

6. CONCLUSION

In this paper we have proposed a novel method of edge detection using cellular automata. By experimental comparison of different methods of edge detection on gray scale image we observe that the proposed method leads to a better performance. The experimental results also demonstrated that it works satisfactorily for different gray level images. This method has potential future in the field of digital image processing. The work is under further progress to examine the performance of the proposed edge detector for different gray level images affected with different kinds of noise. Here it is restricted to gray scale images but can be extended to color images also.

7. REFERENCES

[1] Ziou, D. and Tabbone, S., 1998. Edge detection techniques—an overview, Pattern Recognition and Image Analysis 8 (4), pp. 537–559.

[2] Renyan Zhang, Guoliang Zhao and Li Su, 2005. A New Edge Detection Method in Image Processing. In: IEEE Proceedings of the ISCIT'05 Oct 12-14, 1: pp.445-448.

[3] Kumar, Tapas., Sahoo, G., Lamba, I.M.S., Bhatia, C.M, 2008.Celllar automata based thresolding for edge detection in binary images. , Journal of Computer Science & its Application, vol.15. No.2. pp. 148-155.

[4] Gillavata, J., Ewerth, R. and Freisleben, B., 2003. Finding text in images via local thresholding, pp.539-542.

[5] Davis, L. S., 1995. Edge detection techniques: Computer Graphics Image Process. (4), pp. 248-270.

[6] Sobel., E., 1970. Camera Models and Machine Perception. PhD thesis. Stanford University, Stanford, California.

[7] Marr. D and Hilderth. E, 1980. Theory of Edge Detection," Proc.R.Soc. London, vol. B 207, pp 187-217.

[8] Roberts, L. G., 1965. Machine perception of three-dimensional solids," Optical and Electro-Optical Information Processing, MIT Press Cambridge, Massachusetts, pp. 159-197.

[9] S. Price, 1996. Edges: The Canny Edge Detector http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/MARBLE/low/edges/canny.htm.

[10] Popovici, A. and Popovici, D., 2002.Cellular automata in image processing: in Proceedings of the 15th International Symposium on the Mathematical Theory of Networks and Systems.

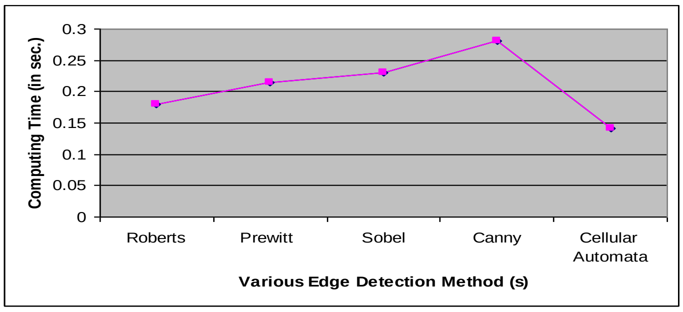


Figure 10. Time analysis of Rose image using various edge detection methods.

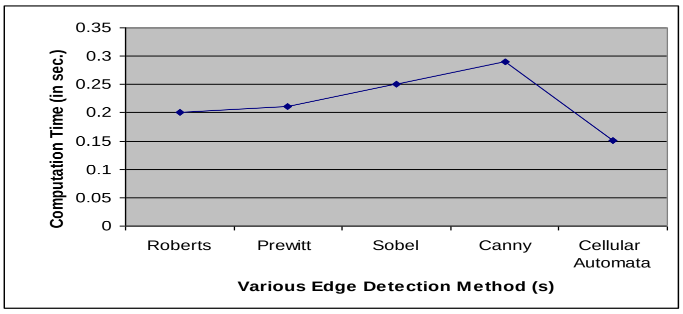


Figure 11. Time analysis of Rose image using various edge detection methods.

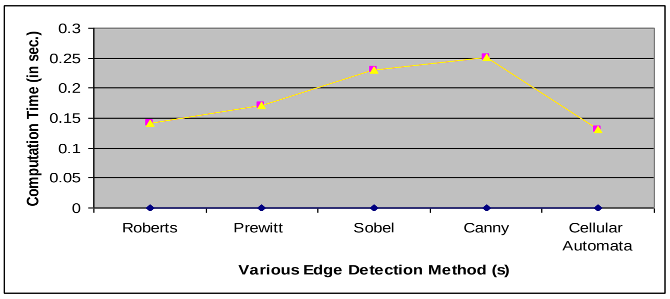


Figure 12. Time analysis of Rose image using various edge detection methods.

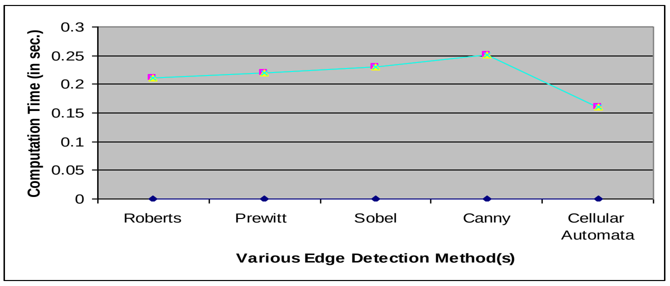


Figure 13. Time analysis of Rose image using various edge detection methods.

[11] Nie, Harald. 2006. Introduction to Cellular Automata: Organic Computing, USA.

[12] Pratt., William. K., 2001. Digital image processing, 2nd Edition.John Willey & Sons Inc., pp. 150-157.

[13] Wongthanavasu, S. and Lursinap, C., 2004. A 3-D CA –based Edge Method for 3-D images, The Proceedings of the 11th IEEE Int. Conference on Image Processing, pp. 235-238.