Edge Detection Method based on Cellular Automata

Jyoti Swarup, Dr. Indu S
Dept. of Computer Science and Engineering, Dept. of Electronics & Communication Engineering, Delhi Technological University

Abstract: Edge Detection is a vital pre-processing phase in image processing and computer vision which detects boundaries of foreground and background objects in an image. Discrimination between significant and spurious edges plays an important role in accuracy of edge detection. This paper introduces a new approach for edge detection in images based on cellular computing. Existing edge detection methods are complex to implement and fail to produce satisfactory results in the case of noisy images. Some methods tend to give spurious edges and some tend to miss true edges in the image. The purpose of using cellular computing approach is to reduce complexity and processing time as the method is computationally simple and fast due to parallel processing. The results of the proposed algorithm are compared with results of existing edge detection techniques by evaluating MSE and PSNR values which indicates promising performance of the proposed algorithm. Visually, proposed method tend to produce better results which discriminate objects and interpret the edges more clearly even for cluttered and complex images.

Index Terms: cellular automata, edge detection, linear rules, parallel processing

I. INTRODUCTION

Edge detection is a procedure to detect contour of objects by finding the discontinuities or change in brightness within an image. Edge detection is an important step in digital image processing and computer vision by preserving the important structural properties in an image. There are several edge detection techniques [1]-[3] and can be broadly grouped into two categories. The gradient based method detects the edges by computing the maximum and the minimum in the first derivative of an image. In Laplacian method, edges are traced by locating zero crossings in the second derivative of the image. There are problems of false edge detection and missing true edges which can significantly affect the result of object recognition, pattern recognition and feature extraction processes.

Cellular Automata finds its wide applications in the area of Image processing and computer vision [14]. Theory of self-reproducing Automata (CA) was initiated by J. Von Neumann and Stan Ulam [4]-[5] in 1950’s. Stephen Wolfram extended the concept of automata by developing CA Theory [6]-[8]. Digital image is represented by a 2-D array for a grayscale image and a collection of three 2-D arrays for color image. Two dimensional Cellular Automata [10] can be implemented on an image with an ease. Various possible applications of CA in image processing ranges from edge detection algorithms, translation of images, rotation through an angle, scaling operations like thinning and zooming, finding contour and edges for image segmentation and other NP-complete problems, such as graph coloring or satisfiability, designing a controlled random number generator with smaller aliasing rate than a linear counter based on shift register and XOR gates and pattern generation [12].

II. CONCEPT OF CELLULAR AUTOMATA

A. Structure of cellular automata

Cellular Automata (CA) is a finite state machine having multiple cells. One dimensional CA is a linear array of cells and two dimensional CA [10] is a grid of cells where each cell is influenced by its neighboring cells. There is a finite range of possible states of a cell. State of a cell is updated simultaneously depending upon previous states of its neighboring cells. Cellular Automata can be represented as

![Figure 1: (a) Von-neumann (b) Moore (c) Extended Moore neighborhood](image)

There are two neighborhood structure in Cellular Automata: Von-neumann and Moore as shown in the fig. 1. Von-neumann neighborhood has four neighbors surrounding a cell and in Moore neighborhood, there are eight neighbors. The radius of neighborhood is 1. In extended Moore
neighborhood, radius is increased to 2 having 24 neighbors and one center cell [15].

Four boundary conditions in cellular automata are:

### Null boundary condition.

<table>
<thead>
<tr>
<th>0</th>
<th>x1 x2 x3 x4 x5 x6 x7</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>x8</td>
<td></td>
</tr>
</tbody>
</table>

### Fixed boundary condition.

<table>
<thead>
<tr>
<th>0/1</th>
<th>x1 x2 x3 x4 x5 x6 x7</th>
<th>0/1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>x7 x8</td>
<td></td>
</tr>
</tbody>
</table>

### Periodic boundary condition.

<table>
<thead>
<tr>
<th>x8</th>
<th>x1 x2 x3 x4 x5 x6 x7</th>
<th>x1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>x8</td>
<td></td>
</tr>
</tbody>
</table>

### Adiabatic boundary condition.

<table>
<thead>
<tr>
<th>x1</th>
<th>x1 x2 x3 x4 x5 x6 x7</th>
<th>x8</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>x8</td>
<td></td>
</tr>
</tbody>
</table>

### Reflexive boundary condition.

<table>
<thead>
<tr>
<th>x2</th>
<th>x1 x2 x3 x4 x5 x6 x7</th>
<th>x7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>x8</td>
<td></td>
</tr>
</tbody>
</table>

### Rule formation

Elementary CA has two states, 0 and 1, for every cell. For a combination of three neighbors there can be 8 (=2^3) possible combinations i.e. 000,001,……,111. There are total of 2^k rules, each rule is represented by an 8-bit binary number i.e Rule 0 to Rule 255. For a two state nine neighborhood CA, there exist 2^{29} possible rules. Among these, 2^9 rules are linear and can be determined by fig. 2. Remaining 2^{29} - 29 (= 502) are non-linear rules [16],[17].

\[
\begin{array}{ccc}
64 & 128 & 256 \\
32 & 1 & 2 \\
16 & 8 & 4 \\
\end{array}
\]

Figure 2: 2-Dimensional CA rule convention

Cellular automata have several advantages over other methods of computation. Simplicity of implementation makes it appropriate for solving complex problem in less computational time complexity. CA is comparatively faster than other methods [13].

### III. RELATED WORK

According to literature so far, CA roots itself for more than two decades in image processing [19].

In [16], Choudhury et al. applied eight basic 2-Dimensional Cellular Automata rules (Rule 2, 4, 8, 16, 32, 64, 128 and 256) to an image for its translation in all directions. Various rules were applied to obtain various operations on images like scaling and thinning horizontally as well as vertically, zooming of symmetric images. Qadir et al. [20] extended the concept of translation of the image by using twenty five neighborhoods instead of nine neighborhoods. This method for translation was used in gaming applications. In [21], Khan proposed that hybrid CA is the possible solution for rotation of images through an arbitrary angle. According to him, 2-D CA rules are applied to rotate an image by an angle \( \pi \) about x and y axis respectively.

Determination of rule set is a crucial step in CA. Specifying and selecting rules manually is a slow and laborious process, also it may not scale well to larger problems. The Fuzzy Cellular Automata is employed with fuzzy logic, having fuzzy states of a cell and fuzzy functions for transition rules. Fuzzy CA (FCA) is a special class of CA which is employed to design the pattern classifier [22]. Wang Hong et al. [23] suggested a novel method for image segmentation based on fuzzy cellular automata. In [24], More and Patel used the property of Cellular Learning Automata to enhance the edges detected by fuzzy logic. In [15], Nayak, Sahu and Mohammed compared the performance of existing edge detection techniques with their proposed method based on extended neighborhood CA and null boundary conditions.

### IV. PROPOSED ALGORITHM

In the proposed algorithm, all input images are grayscale. This method highlights the contribution made to overall appearance of an image by significant bits. Considering the fact that each pixel is represented by 8 bits, Higher-order bits i.e. first four most significant bits of binary representation of intensity depict maximum image information.

Each cell represents an image pixel with certain intensity or pixel value. According to Moore neighborhood, four linear rules are identified which can efficiently result in identification of boundary of a region. These composite rules are as given below and are calculated with some basic rules and XOR function. These rules are computed as follow:

\[
\text{Rule 29} = \text{Rule16} \oplus \text{Rule8} \oplus \text{Rule4} \oplus \text{Rule1} \\
\text{Rule 113} = \text{Rule64} \oplus \text{Rule32} \oplus \text{Rule16} \oplus \text{Rule1} \\
\text{Rule 263} = \text{Rule256} \oplus \text{Rule4} \oplus \text{Rule2} \oplus \text{Rule1}
\]
\[ \text{Rule 449} = \text{Rule256} \oplus \text{Rule128} \oplus \text{Rule64} \oplus \text{Rule1} \]

Integration of these rules result in edges present in the image.

\[ \text{Rule29} \mid \text{Rule113} \mid \text{Rule263} \mid \text{Rule449} \]

The image is first divided into its bit planes which is called bit plane slicing, then these transition rules are applied to every binary bit plane in parallel. The resultant successor matrix bit planes are merged into gray image followed by binarization of image done according to Otsu’s threshold technique. Finally, morphological operations are performed to enhance the results by removing noise and obtain true edges in the given input image.

The state of a cell in the next generation is determined by the previous state of its neighboring cells and all cells are updated synchronously resulting in unit time complexity. Every cell can have two states, 0 or 1. In a 1x3 neighborhood structure, state of d pixel is updated by considering previous states of pixel a, b and c. Fig. 3 illustrates the method to apply identified composite rules by taking different set of 1x3 neighbors to update value of pixel d. These four rules can be represented as four borders which result in edge detection when applied by sliding a window of 3x3 pixels over an image of size \( mxn \).

![Figure 3: Conceptual representation for identification of four rules.](image)

\[ \begin{array}{ccc}
  a & b & c \\
  d & & \\
\end{array} \]

(a) Rule 449

\[ \begin{array}{ccc}
  a & b & c \\
  & d & \\
\end{array} \]

(b) Rule 113

\[ \begin{array}{ccc}
  & d & \\
  b & & c \\
\end{array} \]

(c) Rule 263

\[ \begin{array}{ccc}
  a & b & c \\
  & & d \\
\end{array} \]

(d) Rule 29

Figure 4: Illustration (a) region in an image (b) Contour identified by applying these four rules.

![Image](image)

Figure 5: Flowchart of proposed method

V. RESULTS AND CONCLUSION

In this paper, a comparison of proposed method is carried out against the commonly deployed Gradient and Laplacian based Edge Detection techniques. These techniques suffer the problems of inaccurate edge detection, missing true edges, producing thin or thick lines and extra edges due to noise etc.

The results of Mendeley Dataset [25] as shown in Fig. 6-8, are compared with existing edge detection techniques and indicate promising performance of the proposed algorithm by evaluating MSE and PSNR values as given by Table 1. Mean square error (MSE) and Peak signal to noise ratio (PSNR) are used to compare quality of reconstructed image with its ground truth image. If an operator gives resultant image with less PSNR and high MSE then operator has high edge detection capability.

Evaluation of test images show that proposed method exhibit better performance even for noisy and cluttered images. Visually, proposed method produced promising results of edge detection when compared to canny, sobel and prewitt edge detectors. Canny consist of several spurious edges whereas sobel and prewitt lack some of the strong edges.

![Image](image)

Figure 6: Illustration of results for test images.

Table 1: Experimental results for test images

<table>
<thead>
<tr>
<th>Original image</th>
<th>Proposed method</th>
<th>Canny MSE</th>
<th>Sobel MSE</th>
<th>Prewitt MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image 1</td>
<td>MSE 0.8810</td>
<td>0.9172</td>
<td>0.9741</td>
<td>0.9745</td>
</tr>
<tr>
<td>PSNR</td>
<td>0.5500</td>
<td>0.3754</td>
<td>0.1138</td>
<td>0.1122</td>
</tr>
<tr>
<td>Image 2</td>
<td>MSE 0.9156</td>
<td>0.9201</td>
<td>0.9720</td>
<td>0.9718</td>
</tr>
<tr>
<td>PSNR</td>
<td>0.3831</td>
<td>0.3616</td>
<td>0.1232</td>
<td>0.1242</td>
</tr>
<tr>
<td>Image 3</td>
<td>MSE 0.8855</td>
<td>0.9258</td>
<td>0.9740</td>
<td>0.9741</td>
</tr>
<tr>
<td>PSNR</td>
<td>0.5283</td>
<td>0.3346</td>
<td>0.1146</td>
<td>0.1142</td>
</tr>
<tr>
<td>Image 4</td>
<td>MSE 0.8738</td>
<td>0.9046</td>
<td>0.9586</td>
<td>0.9585</td>
</tr>
<tr>
<td>PSNR</td>
<td>0.5858</td>
<td>0.4355</td>
<td>0.1837</td>
<td>0.1840</td>
</tr>
</tbody>
</table>


84
<table>
<thead>
<tr>
<th>Image</th>
<th>MSE</th>
<th>0.8825</th>
<th>0.9313</th>
<th>0.9671</th>
<th>0.9670</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSNR</td>
<td>0.5427</td>
<td>0.3092</td>
<td>0.1453</td>
<td>0.1459</td>
<td></td>
</tr>
</tbody>
</table>

Figure 6: Results of test image1

Figure 7: Results of test image2

Figure 8: Results of test image3

Figure 9: Bar chart of PSNR values of results of different edge detectors and proposed method

REFERENCES


[25] http://dx.doi.org/10.17632/hvtmfvbtj.1#file-7dd2224d-0376-40ca-9a0d-27455a10e503