Evolutionary Design of Edge Detector Using Rule-Changing Cellular Automata

Shohei Sato and Hitoshi Kanoh
Department of Computer Science, Graduate School of Systems and Information Engineering
University of Tsukuba
Tsukuba, Ibaraki, 305-8573, Japan
s_shohei@kslab.cs.tsukuba.ac.jp, kanoh@cs.tsukuba.ac.jp

Abstract—A new design method for Cellular automata (CA) rules are described. We have already proposed a method for designing the transition rules of two-dimensional 256-state CA for grayscale image denoising. The gene expression programming was employed as the learning algorithm, in which the chromosome encodes the transition rule as the expression. The CA designed by the method ran faster than previous methods. In this paper, an improved method for designing the CA based edge detector is proposed. The ground truth for training CA is generated by the Canny edge detector, from which two objective functions are calculated. Both objective functions are optimized by a multi-objective evolutionary algorithm. The rule-changing CA is used to improve the performance. The experimental results showed that rule-changing CA designed by the proposed method have higher performance for edge detection than the ordinary CA.

Keywords—image processing; edge detection; cellular automata; evolutionary computation

I. INTRODUCTION

A. Cellular Automaton

The cellular automaton we address in this paper consists of a two-dimensional grid of cells, each in one of 256 states. Let \( s_{ij}(t) \) denote the state of the cell at position \((i, j)\) at time \( t \). The next state of the cell, \( s_{ij}(t+1) \), depends on the current state, \( s_{ij}(t) \), and the states of its neighbors at time \( t \), in accordance with a transition rule. The rule is the same for each cell, does not change over time, and is applied to the whole grid simultaneously. Fig. 1 shows an example of a much simpler binary CA in which each cell takes one of two possible states, black and white. The neighbors of a cell described a method for dealing with grayscale images using binary CA [5]; however, the calculation time is impractically long.

We previously presented a more practical method for dealing with 8-bit grayscale images (where every pixel takes a value from 0 to 255) in which 256-state CA are used [10]. A transition rule is represented as an expression encoded as a linear string of fixed length, and chromosomes evolve using an evolutionary algorithm. The resulting rules performed more than 100 times faster than those obtained with Rosin's method.

We have now improved this method to construct a CA-based edge detector. We adopted the rule-changing CA proposed by Kanoh et al. in order to improve the performance [11, 12, 13]. The Canny edge detector [14] is used as the learning supervisor. The fitness of each CA is inversely proportional to the error probability, which is calculated using the ground truth generated by the Canny edge detector. A multi-objective optimization technique is used to optimize both the detection rate and the false alarm rate at the same time.

After first giving an overview of image processing using CA and the edge detection problem to be addressed, we describe our method for designing rule-changing CA. We then present and discuss our experimental results. Finally, we conclude with a brief summary of the key points and a mention of possible future work.

II. OVERVIEW

A. Cellular Automaton

The cellular automaton we address in this paper consists of a two-dimensional grid of cells, each in one of 256 states. Let \( s_{ij}(t) \) denote the state of the cell at position \((i, j)\) at time \( t \). The next state of the cell, \( s_{ij}(t+1) \), depends on the current state, \( s_{ij}(t) \), and the states of its neighbors at time \( t \), in accordance with a transition rule. The rule is the same for each cell, does not change over time, and is applied to the whole grid simultaneously. Fig. 1 shows an example of a much simpler binary CA in which each cell takes one of two possible states, black and white. The neighbors of a cell...
located at \((i, j)\) are those in the region enclosed by the solid black line; this is known as a Moore-type neighborhood. An example of a transition rule is illustrated in Fig. 2. In accordance with this rule, \(s_{ij}(t+1)\) is determined to be in the white state.

Let \(S(t)\) denote the configuration for the state of all cells; specifically, \(S(0)\) means the initial configuration. A CA is a deterministic system; its temporal dynamics depends only on the initial configuration and the transition rule. Therefore, \(S(t)\) is determined uniquely by \(S(0)\) at an arbitrarily chosen \(t\). Boundary conditions are required for a CA with a finite-size grid in order to apply the transition rule to the cells at the edge. We adopt a condition under which each of the cells on the outside of the two-dimensional grid takes the same state as that of the nearest neighbor cell on the inside. The information for these states is updated at each time step.

**B. Image Processing using Cellular Automata**

CA have been applied to several basic image processing tasks due to their lattice structure and parallelism. Another advantage of using CA is their simplicity, which enables fast implementation on various kinds of devices, such as VLSI [15], FPGA [16], and GPU [17] devices. Thus, CA should be well suited for real-time image processing applications.

The standard image processing algorithm using two-dimensional CA, shown in Fig. 3, has been used for several applications in addition to edge detection, including noise suppression, thinning, and convex hull construction [4, 5, 6, 7].

**C. Edge Detection**

Detecting the edge component of the image, one of the most important image processing tasks, is used for object-background separation, 3-D interpretation of a 2-D image, and pre-processing in image understanding and recognition algorithms [8, 9]. There are many methods for edge detection, and most of them use the computed gradient magnitude of the pixel value as the measure of edge strength [18]. The Canny edge detector is widely used in computer vision due to its good edge quality [14].

The Canny edge detector is a multi-stage algorithm comprising four main steps.

- **Step 1**: Noise reduction using Gaussian filter.
- **Step 2**: Calculation of vertical and horizontal gradient magnitudes of image.
- **Step 3**: Non-maximum suppression to create thin edge.
- **Step 4**: Hysteresis thresholding.

We use the Sobel operator for calculating the gradient magnitudes in step 2. Three configuration parameters have to be determined in this algorithm: the standard deviation, \(\sigma\), of the Gaussian filter (step 1) and the two thresholds, \(TL\) and \(TH\) (step 4). To optimize these parameters, we carried out several experiments.

**D. Related Works**

A method for the evolutionary design of CA rules for edge detection was proposed by Batouche et al. [4, 7]. Two-dimensional binary CA are used for binary-image edge detection, and transition rule evolution is guided by a genetic algorithm. Another application of this method was done by Rosin and the method was extended to handle gray-scale images [5]. The images are decomposed into a number of binary images with all possible thresholds. Each of the binary images is processed using binary CA rules. Rosin later applied an improved version of this method to edge detection [6]. However, this method has a very long calculation time due to the iterative process used for hundreds of binary images.

Kanoh et al. proposed a rule-changing CA in which multiple rules are applied sequentially [11]. This reduces the complexity of a given task by dividing the task into sub-tasks and assigning a distinct rule to each one. They applied their rule-changing CA to benchmark problems and confirmed its effectiveness experimentally [11, 12, 13].
III. PROPOSED METHOD

A. Gene Expression Programming

Our proposed method for designing rule-changing CA uses multi-state CA to deal directly with grayscale images [10]. While our previous method directly encodes a transition rule sequenced as shown in Fig. 2, the proposed method represents a transition rule in the form of a mathematical expression, like those used in genetic programming, and encodes it as an array. This coding scheme enables the number of states to be increased. In this paper, we evaluate the effectiveness of our proposed method for edge detection.

We use the gene expression programming (GEP) algorithm proposed by Ferreira [19], the same as in our previous method. The GEP algorithm is based on genetic programming, in which chromosomes are represented as expressions. The GEP algorithm is almost the same as a simple genetic algorithm in that genetic operators such as selection, crossover, and mutation are applied to each chromosome string. In addition, genetic operators called transposition and recombination are used. In the transposition process, fixed length fragments of chromosome are randomly chosen and moved to other loci. Two kinds of transpositions are used: root insert sequence transposition (RIS-transposition) moves transposable elements to the root of the expression tree, and insertion sequence transposition (IS-transposition) inserts them into the gene head (see III.B) except the root. The recombination is not used in this paper.

B. Codings

The coding scheme in the GEP algorithm transfers rules from expression trees, which are tree structures corresponding to expressions (phenotypes), to linear arrays (genotypes) in accordance with certain rules. GEP genes are composed of a head and a tail. The head contains symbols that represent both functions and terminals (variables and constants), whereas the tail contains only terminals. The length of the head and the tail are determined by the following equation, where Head and Tail denote the length of the head and tail, and \( n \) is the maximum number of arguments of the functions used in the expression. In the proposed method, \( n \) equals two (see Table I). This constraint is applied throughout the search period to prevent the appearance of lethal genes.

\[
Tail = Head(n-1) + 1 \tag{1}
\]

The functions and terminals used are the same as in our previous method [10] (Table I). Fig. 4 shows an example of a chromosome identical to a transition rule in the form of a phenotype, an expression tree, and a genotype (from above). The encoding scheme can be seen from this example. Each element of the expression tree is sequenced in accordance with the numbers on the right side.

We use the rule-changing CA design method proposed by Kanoh et al. in which an array of transition rules \( R_i \) and the number of rule iterations \( M_i \) are encoded as a chromosome, as illustrated in Fig. 5. Each \( R_i \) corresponds to an array illustrated in Fig. 4. In addition to each genetic operation, the number of iterations mutates with a probability of 0.5, which is driven by the substitution of random numbers. The \( M_i \) are subject to \( M_i \geq 1 \) for all \( i \), and the sum of \( M_i \) equals \( t_{\text{max}} \).

C. Fitness Function

The previous method uses the peak signal-to-noise ratio (PSNR) as a fitness function to evaluate the effectiveness of noise removal. While the PSNR is a suitable measure for the quality of restored images, the error probability is commonly used to evaluate the performance of an edge detector [20]. We define two objective functions based on the error probability:

\[
\begin{align*}
\text{Object1} &= \frac{N_{TP}}{N_{TP} + N_{FN}} + \frac{N_{TN}}{N_{FP} + N_{TN}} \\
\text{Object2} &= \frac{N_{TP}}{N_{TP} + N_{FP}} + \frac{N_{TN}}{N_{FN} + N_{TN}}
\end{align*} \tag{2}
\]
where $N_{TP}$ is the number of true-positive results, $N_{FP}$ is the number of false-positive results, $N_{TN}$ is the number of true-negative results, and $N_{FN}$ is the number of false-negative results.

These values are calculated by comparison between an edge map generated by the Canny edge detector (true edge) and an edge map generated by the CA. The edge map is created by thresholding output images of the CA at a gray level of 127. The pixels adjacent to the true edge pixel are ignored during fitness calculation to reduce the penalty for incorrectly positioned edge pixels. Object1 is intended to increase the number of true-positives, and Object2 is intended to prevent false alarms. We considered other objective functions and chose these two on the basis of preliminary experimental results. To maximize Object1 and Object2 simultaneously, we use a multi-objective evolutionary algorithm based on the NSGA-II algorithm [21]. Pareto rank-based selection and elitism are incorporated into our improved method.

IV. Experiments

To evaluate the effectiveness of our improved method for edge detection, we conducted several experiments. The specifications of the CA and GEP environment (Table II) were set on the basis of preliminary test results [10]. Cameraman (Fig. 6(a)) was used as the training image. It contains a significant edge separating the object from the background. The ground truth image (Fig. 6 (b)) was generated by the Canny edge detector and isolated edge pixels were removed. As mentioned in section II. C, we optimized the parameters of the Canny edge detector by carrying out several experiments: $\sigma = 1.0$, $TL = 20$, and $TH = 50$. The evaluation set used in the experiments is shown in Fig. 7.

Fig. 8 shows a comparison of the ordinary CA (thin line) with the rule-changing CA (thick line), which uses two rules. Each point in the figure is the average objective function value of the whole population. Both objective functions of the latter for the last generation (Object1 = 0.76, Object2 = 0.88) were higher than those of the former (Object1 = 0.74, Object2 = 0.84). In the last generation, over 85% of the CAs were Pareto rank 1, indicating search convergence.

Fig. 9 shows the distribution of the rules obtained with Pareto rank 1. The horizontal and vertical axes respectively represent Object1 and Object2. The rule-changing CA performed better than the ordinary CA.

TABLE II. EXPERIMENTAL CONDITIONS

<table>
<thead>
<tr>
<th>Mutation</th>
<th>Randomly chosen two points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selection</td>
<td>4-tournament</td>
</tr>
<tr>
<td>Crossover</td>
<td>One point crossover</td>
</tr>
<tr>
<td>Chromosome length</td>
<td>61</td>
</tr>
<tr>
<td>Number of individuals</td>
<td>200</td>
</tr>
<tr>
<td>Number of generations</td>
<td>400</td>
</tr>
<tr>
<td>Transposition length</td>
<td>3</td>
</tr>
<tr>
<td>IS-Transposition rate</td>
<td>10%</td>
</tr>
<tr>
<td>RIS-Transposition rate</td>
<td>10%</td>
</tr>
<tr>
<td>Number of rule iterations: $t_{max}$</td>
<td>8</td>
</tr>
</tbody>
</table>

The execution examples shown in Figs. 10 (a) and (b) correspond to rules A and B in Fig. 9. The number of detected edge pixels in (b) was more than that in (a), and (a) has thinner edges than (b). Apparently, multi-objective
From these equations, we see that $R_1$ and $R_2$ perform different tasks although they are too complicated to interpret. In both $R_1$ and $R_2$, the states of all nine neighbors are used, and they have non-equivalent roles in the transition rule. The execution results of this rule are illustrated in Fig. 11. Figs. 11(a) and (d) show the results of Gaussian ($\sigma = 1.0$) and Sobel filtering (first and second steps of the Canny edge detector). The other figures show the results of the proposed method. Figs. 11(b) and (e) are output images of the transition rule. Figs. 11(c) and (f) are images after thresholding (threshold = 127). Significant edges, such as object contours and borderlines between the two surface areas, were accurately detected. Compared with the results using the Sobel filter, the transition rule preserves more details (see, for example, the window shutters in Houses). We consider that this difference was due to the presence or absence of Gaussian filter smoothing.

V. CONCLUSIONS

We have improved our evolutionary cellular automata method to enable it to detect edges more effectively than the method developed by Rosin. It uses the gene expression programming algorithm proposed by Ferreira, and each chromosome is trained on the basis of the ground truth generated by the Canny edge detector. The experimental results showed that the rule-changing CA designed using our improved method can detect edges with better accuracy than ordinary CA. We plan to further improve our method to make detection more accurate and robust, possibly by using multiple training images. We may also address the issue of edge detection in a noisy environment.

REFERENCES

Figure 11. Execution example for Houses and Lighthouse. (a, d) Results of Gaussian ($\sigma = 1.0$) and Sobel filtering, (b, e) Output images of rule-changing CA, (c, f) Output images after thresholding.


