Chapter 5
Edge Detection Using Cellular Automata

Paul L. Rosin and Xianfang Sun

Abstract. Edge detection has been a long standing topic in image processing, generating hundreds of papers and algorithms over the last 50 years. Likewise, the topic has had a fascination for researchers in cellular automata, who have also developed a variety of solutions, particularly over the last ten years. CA based edge detection has potential benefits over traditional approaches since it is computationally efficient, and can be tuned for specific applications by appropriate selection or learning of rules. This chapter will provide an overview of CA based edge detection techniques, and assess their relative merits and weaknesses. Several CA based edge detection methods are implemented and tested to enable an initial comparison between competing approaches.

5.1 Introduction

Edge detection is a fundamental tool for computer vision systems. The original use of boundaries detected in a scene was for object detection, as they provide a set of features suitable for model matching. However, edge maps are nowadays applied to a host of different ways, such as preventing removal of significant structures in anisotropic blurring [23], indicating salient regions [33], selecting redundant seams in image resizing [1], image registration [25], depth from focus [8], extended depth of field, etc.

From Roberts' early work [31] in 1963 through to Canny's influential paper [4] in 1986, most methods have used first-order derivatives to estimate edge magnitude and orientation. Subsequently, the edgels (i.e. edge pixels) are often thresholded, linked, and thinned. Edge detection is normally carried out using local operators, and since edges may not be locally distinct in the image this means that edge maps are prone to fragmentation. Thus, research into new methods of edge

Paul L. Rosin · Xianfang Sun
School of Computer Science & Informatics, Cardiff University, Cardiff, CF24 3AA, UK
e-mail: {Paul.Rosin,Xianfang.Sun}@cs.cf.ac.uk

P. Rosin et al. (eds.), *Cellular Automata in Image Processing and Geometry*,
Emergence, Complexity and Computation 10,
DOI: 10.1007/978-3-319-06431-4_5,

detection continues to be an active area. For instance, more recent approaches try to combine changes in brightness, colour, and texture [18], and use machine learning to determine how these multiple cues are combined.

During the last decade or so there has been considerable interest in using cellular automata to perform edge detection. They have several potential benefits, such as:

- **Efficiency** – Cellular automata are both inherently parallel and computationally simple, which means that they can implemented very efficiently in hardware.
- **Global properties** – Since multiple iterations of cellular automata rules can lead to emergent global behaviour it is feasible that a cellular automaton approach could provide benefits regarding the tendency of existing methods to produce fragmented output where there is locally inconsistent data.
- **Application specificity** – Rules can be selected (e.g. learnt from training data) to operate better than general purpose edge detectors for demanding applications, e.g. noisy modalities such as ultrasound and SAR.

There are many applications of CA based edge detection. For instance, in medical image processing some examples are: detection of tumours [7], identification of the pectoral muscle in mammograms [43] and diagnosis of lung cancer [30]. Other applications include: analysis of antibiotic images [19], detection of fabric defects [42], detection of grain boundaries in rocks [11], horizon tracking [9] and content based image retrieval (see chapter 8).

However, it is difficult for the general reader to gain an understanding of the state of the art in cellular automata based edge detection as the relevant papers are dispersed over many conferences and journals, which are often not specific to either cellular automata or image processing/computer vision. This chapter aims to collect together descriptions of these methods, and to provide a critical assessment of their merits.

Another obstacle in the path of the reader is the lack of performance evaluation. In the general area of computer vision there has been considerable work on comparison of edge detectors, using various methodologies, e.g. human assessment, comparison to (human) ground truth [12, 13, 40]. However, the majority of papers on developing cellular automata methods for edge detection simply show a few examples of their results alongside the Sobel or Canny outputs. Ideally this should be replaced by quantitative evaluation scores. However, this process is generally complicated by the need to process the raw outputs of the edge detectors before comparison, and so would depend on parameters for thresholding, thinning, etc.

5.2 Boundary Detection

The classical or the most popular cellular automata (CA) are binary, so it is natural to use CA for binary image processing. Edge detection of binary images can be considered as finding the boundaries of objects or regions in an image, and thus it is also called boundary detection.

When CA are used for image edge detection, the image to be detected is usually considered as a cellular automaton, each pixel of which is taken as a cell that

connects to its neighbouring pixels (cells), and the pixel values (0 or 1 in binary images) are the state values. Basically, the process of edge detection takes several iterations of state transition from the initial states (the input image pixel values) to the final states (the output image with 1 representing edge pixels and 0 the others). It is critical to find good state transition rules for the CA.

The state transition rule proposed by Wongthanavasu and Sadananda [45] is probably the simplest one, where a cell changes its state from 1 to 0 only if all its von Neumann neighbours have state value 1, otherwise it retains its original state (either 1 or 0). Though simple, this rule can produce reasonable results. It is intuitively understandable why this simple rule works. Suppose that we have a contiguous region of 1 as the foreground, then one iteration of CA evolution using this rule will result in all the inner pixels (which have all 1 in their neighbourhoods) changing to 0 (non-edge), while the boundary pixels (those have at least one 0 in their neighbourhoods) to remain as 1 (i.e. edge). More iterations will not change the pattern. The weakness of this rule is also obvious: if noisy pixels (value 0) exist inside a foreground region, the neighbouring pixels of a noisy pixel will remain unchanged, and wrongly classified as edges. Thus, this approach is not able to work effectively in the presence of noise.

To find optimal state transition rules, several researchers resorted to genetic algorithms. As early as in 1994, Sahota *et al.* [35] used a genetic algorithm to train CA to find optimal rules for edge detection. They used three simple original/edge image pairs as the training samples, and the obtained rules were then used in edge detection. They did not describe the rules they obtained, and only showed the detection results of two manually designed toy images.

Selvapeter and Hordijk [37] adopted the same method as in [35] to train CA, where they used similar original/edge image pairs as the training samples. The contribution of their work was to deal with real images with noise. They simply used a CA-based image noise filter [38] as the first step to denoise the image, and then used the trained CA rules for edge detection. They showed that the CA-based method can produce comparable results to the other methods such as Canny, Prewitt, Sobel, and Laplacian of Gaussian operators when the image is noiseless, and it can also produce reasonable results when the image is noisy, while the other methods failed to produce meaningful edge detection results. However, their comparisons in noisy cases are unfair because they had first denoised the image before using the CA to detect the edges, while they used the other methods to directly detect the edges in the noisy image.

Batouche *et al.* [3] also applied genetic algorithms to train CA for binary image edge detection. Instead of training rules for all the $2^9 = 512$ pixel patterns in a Moore neighbourhood, they assigned those rotational symmetric patterns (that rotate 0°, 90°, 180°, 270°, or flip horizontally or vertically) the same state transition rule. A weak matching criterion was introduced, so that some patterns with a difference less than a similarity threshold were further merged into a single rule. They assembled 15 state transition rules into a packet, which is represented by a chromosome, and trained to produce optimal rules. Experiments showed that edges are successfully detected, but are a little bit thick.

Slatnia *et al.* [41] assigned rotational symmetric patterns the same rule, but they were not training symmetric state transition rules as was done in [3]. Inspired by Rosin's work [32], they instead used genetic algorithms to train only a single powerful rule, that is, the central cell changes its state only when its Moore neighbourhood matches a specific pixel pattern. Interestingly, they got an optimal rule which was the same as that proposed by Wongthanavasu and Sadananda [45]. Again, they compared their results with Canny's, and claimed that they had better results, although this is not obvious from the figures they provided.

Yang *et al.* [46] proposed a CA approach for a specialised form of edge detection. Instead of using binary state transition rules to evolve the CA, they complicated the algorithm by introducing more state values. Their method consists of two steps. The first step sets the state value of each pixel. It uses the Prewitt operator, rotating in eight angles (from 0° with interval 45°, numbered as direction 1 to 8), to compute eight direction values for each pixel, and takes the direction number (called the location coordinate in their paper) with maximal direction value as the state value (called the characteristic vector). The state value is changed to 16 if the maximal direction value is less than 3. The second step evolves the CA based on the states (represented by values in $1 \sim 8$, or 16) in the neighbourhood of each cell. The differences of the state values between the central cell and its neighbours are calculated, and the number of neighbours with differences 0, 1, or 7 is counted. If the number is not equal to 2, the state of the central cell is changed to 16. After several iterations, the pixels with state values other than 16 are classified as edge pixels. One iteration of the algorithm will result in reasonable edge detection results. More iterations will delete irregular edges, and only the edges of the objects that are strictly rectangular survive.

In fact, we can simply describe Yang *et al.*'s algorithm by binary state transition rules also in two steps. The first step performs only one iteration with the rule that if three contiguous neighbours of a central pixel have the value 1 and the other three contiguous neighbours symmetric to them about the central pixel have the value 0, then the central pixel keeps its state (either as edge or non-edge), otherwise it is non-edge. The second step performs several iterations based on the rule that an edge pixel remains as edge if exactly two pixels in its von Neumann neighbourhood are edge pixels, and all the other pixels in its Moore neighbourhood are non-edge.

It should be mentioned that most of the proposed CA-based binary image edge detection methods were compared against Canny's method. This is inappropriate because Canny's method was not designed specifically for binary image edge detection, but rather for grey-scale image edge detection. In fact, Canny's method is not poorer than the above-mentioned methods even when dealing with binary images.

Another issue is that, from an image processing perspective, detecting boundaries in binary images is a relatively trivial task, and is not generally considered to be a research problem. In comparison, detecting edges in intensity images (which often also involves estimating edge magnitude and orientation as well) still regularly generates many papers in high ranking journals and conferences in the field of image processing/computer vision.

5.3 Edge Detection in Intensity Images

For edge detection of intensity images, some approaches first convert the intensity images into binary ones, and then evolve two-state cellular automata using specific state transition rules to determine edge pixels, while the others directly update pixel states based on the relationship of the central pixel with its neighbourhood, mostly a 1-ring von Neumann or Moore neighbourhood.

Popovici and Popovici [26] proposed an edge detection approach based on the state differences between the central pixel and the pixels in its von Neumann neighbourhood. If all the absolute state differences are less than a threshold ε, then the state of the central pixel becomes 0, otherwise it remains unaltered. The rule can be formulated as:

$$v_c^+ = \begin{cases} 0 & \text{, if } |v_i - v_c| \leq \varepsilon,\ \forall i \in N_c \\ v_c & \text{, otherwise.} \end{cases}$$

where v_c and v_c^+ are the current and the updated states of the central cell c, N_c is the von Neumann neighbourhood of the cell c, and v_i is the current state value of the cell i in N_c.

Gorsevski *et al.* [11] used Popovici and Popovici's approach to detect the grain boundaries in deformed rocks, but they did not cite [26].

Wongthanavasu and Sadananda [45] proposed a conditional rule to update the cellular state as:

$$v_c^+ = \begin{cases} v_c & \text{, if } v_c \leq v_{\max} - v_{\min} \\ v_{\max} - v_{\min} & \text{, otherwise.} \end{cases}$$

where $v_{\max}$ and $v_{\min}$ are the maximum and minimum states, respectively, in the von Neumann neighbourhood of the central cell c. The above rule can be described theoretically by a state transition table. However, it is hard to construct and use such a table in practice, because there are 256^5 entries in the table for a 256 greyscale image with the von Neumann neighbourhood. Wongthanavasu and Sadananda provided a partial transition table for illustration. Our experiments show that only a single iteration of this state transition will produce reasonable results. Multiple iterations actually degrade the results.

Wongthanavasu and Lursinsap [44] also extended the above-described conditional rule into 3D image edge detection with the only difference that the neighbourhood is now 3D von Neumann. Their experiments showed that the CA approach exhibited better performance than the Sobel and the Laplacian detection algorithms on average.

Kumar and Sahoo's method [15] also directly utilizes the intensities of the central pixel and its neighbourhood to detect edges, but the algorithm description is unclear, and it is hard to figure out how the algorithm is actually implemented.

Diwakar *et al.* [7] presented an approach that first convert an intensity image to a binary image through thresholding, and then use rules similar to Conway's Game

of Life to detect the edges. Unfortunately, their description is unclear.[1] Essentially they are using the thresholding to do the majority of the work, and then the CA just finds the boundaries in the binary image. Otsu's method [21] is used for thresholding. The resulting pixel value 0 represents background, and 1 represents a potential edge pixel. The cellular automaton rule used is totalistic, and (to the best of our understanding given the inconsistencies of their description) is:

$$v_c^+ = \begin{cases} 1 & \text{, if } |M_c| = 5 \\ v_c & \text{, if } |M_c| = 3,4,6,7 \\ 0 & \text{, otherwise.} \end{cases}$$

where $|M_c|$ is the number of edge pixels (value 1) in the Moore neighbourhood of the central cell c (including c itself). Since the thresholding is global, it will miss many edges and also find spurious edges!

Another method that first converts an intensity image to a binary image and then uses CA to detect edges, was proposed by Qadir and Khan [29]. Although 2^{512} possible rule sets exist for a CA with a Moore neighbourhood, there are only 512 linear rule sets among them. Qadir and Khan tested all the 512 linear rule sets, and found that some of them showed no edge detection, some showed strong edge detection, while the others showed weak detection. They compared their results with Sobel's and Canny's, and claimed that their results are better, but this is not obviously clear from their example images provided.

The state transition rules of the above-mentioned approaches are all specified manually, and are not necessarily optimal. Some researchers tried to find optimal rules for edge detection using genetic algorithms or evolutionary algorithms.

Kazar and Slatnia [14] used genetic algorithms to construct CA rules for edge detection of intensity images. One novelty in their approach is that they do not use pixel intensities as state values, but rather label different intensities in the Moore neighbourhood of a central pixel, and take the labels as the state values. In this way, they significantly reduced the possible number of neighbourhood patterns from $256^8/5$ to $8^8/5$ for 256 greyscale images, and thus constructing state transition tables becomes computationally affordable.

Sato and Kanoh [36] introduced rule-changing cellular automata for edge detection, and used a form of genetic programming, namely gene expression programming (GEP) – an evolutionary algorithm to optimise the CA rules. The idea of rule-changing CA is to execute an array of transition rules R_i for M_i iterations in sequence. Each rule R_i is represented by a binary expression tree with the leaf nodes being the pixel states in the Moore neighbourhood of a central pixel or the constants 0, 127, and -128, and the non-leaf nodes functions max(), min(), saturated addition, or saturated subtraction. GEP were used to optimise both R_i and M_i. Experiments showed that better results are obtained using two rules (the rule-changing CA) than

[1] Diwakar *et al.*'s neighbourhood appears to contain the central pixel, which is not consistent with the standard descriptions of Conway's Game of Life. Moreover, they describe their system as implementing Wolfram's rule 124, which is however normally used to describe a one dimensional (rather than two dimensional) cellular automaton.

using only one rule (the ordinary CA). The paper provided their optimal rules and iterations R_1, M_1 and R_2, M_2. Unfortunately, it seems that errors or typos existed in R_1 because it is not a binary expression tree with two arguments for each function as stated in their algorithm specification.

Priego *et al.* [27] describe an approach for edge detection from hyperspectral (i.e. multi-band) images. Rather than input the hyperspectral values directly into the CA, they first extract a set of features from each pixel's neighbourhood (first the eight spectral angles are computed, and these are then described by their mean, directional gradients and standard deviation). A genetic algorithm is then employed to learn $M = 20$ rules, each consisting of a tuple mapping an instance of the six feature elements to a real valued output value in the range $[0, 1]$. Once the rules have been learnt, the CA is run by applying at each pixel the closest matching rule (in terms of its Euclidean distance to the six feature elements). The output at each pixel is thresholded at 0.5 to produce a binary edge map.

Beside traditional CA approaches, cellular automata combined with other techniques were also proposed for edge detection of intensity images.

Mirzaei *et al.* [20] used fuzzy cellular automata for edge detection. They designed eight masks, each of which separates the Moore neighbourhood into two groups. The average of the absolute state differences between the central pixel and the pixels within each group is calculated, and then fuzzified into a fuzzy description of 'High' and 'Low'. Thirty-two fuzzy rules are used for fuzzy state transition, and the resulting fuzzy description is defuzzified to produce an updated numerical state. The authors claim that their method has better efficiency than the Robert and Sobel edge detectors, and moreover that they have largely overcome detection errors (missed edges and false edges). However, their published results do not support this claim.

Chen and Yan [6] proposed an approach for edge detection which first uses a CA-based diffusion model for image smoothing, and then detects the image edges by finding the zero-crossings of the second order derivative of the image defined as the difference between the smoothed and the original images divided by the diffusion time. Experiments showed better results were obtained by the proposed method than that by Laplacian of Gaussian, Laplacian, Canny, and Sobel operators. However, the final results are heavily dependent on the suitable choice of the number of iterations in the smoothing stage.

Other methods, including cellular neural networks [2, 17], fuzzy cellular neural network [39], cellular learning automata [10], and cellular automata transformation methods [24], all produced reasonable results, which shows that CA combined with other techniques are promising tools for image edge detection.

Finally, we describe a method developed by one of this chapter's authors. In an attempt to retain the simplicity of binary CA with the ability to process intensity (i.e. non-binary) images, Rosin [34] applied threshold decomposition, a technique often used in image processing. This involves decomposing a gray level image into the set of binary images obtained by thresholding it at all possible gray levels. A single set of two-state CA rules is learnt which is applied independently to each binary image.

The resulting binary output images are then combined as a simple summation to produce the edge magnitudes.

To find suitable rules a deterministic approach was used, namely sequential floating forward search (SFFS) [28], which starts from the empty set and iteratively adds the best performing rule. The objective function was such that the edge magnitudes constructed from the CA with threshold decomposition should provide a good match – in terms of root mean square (RMS) – to the gray level target ground truth edge map.

The final rule set that was learnt from the training data simply consisted of a single rule. It specified that any white pixel in a 3×3 homogeneous (i.e. all white) neighbourhood is set to black (inverted). For each of the binary images that the input is decomposed into, this causes all white pixels to be replaced by black except for pixels adjacent to black pixels in the input image. Thus, a black image is formed containing a one pixel wide white strip along the original black/white transitions, and these images are summed at the reconstruction stage of threshold decomposition.

5.4 Post-processing of Edges

In the introduction we stated how, after an initial edge detection, the results are often thresholded, linked, and thinned. Clearly, cellular automata have been designed for thinning – see chapter 3. They are also suitable for performing linking, although this is less common.

Lee and Bruce [16] describe a method for edge detection that is initialised with computing a gradient angle and edge magnitude for each pixel using an algorithm mimicking the Prewitt edge detector. They then use CA to perform adaptive thresholding based on the gradient angle and edge magnitude information. The edges are typically overestimated after adaptive thresholding, and each edge pixel is tested, and removed if it has the lowest edge magnitude in its neighbourhood and removal of this pixel will not bisect the edge. Further post-processing is performed using CA to thin wide edges, remove short edges, and delete the edges which enclose small regions.

Chang *et al.*'s [5] approach to edge detection involves using an "orientation information measure" that essentially estimates edge strength, and is used to provide an under-thresholded sparse binary edge map (the mark matrix). CA rules are then applied to link the edge pixels. A set of 3×3 edge (line) patterns are defined based on the assumption that the width of edges is one pixel, and the CA updates a pixel state to edge if it matches an edge pattern when taking as edge the pixel with maximum orientation information measure in its neighbourhood or semi-neighbourhood.

Like Chang *et al.*, the method of Peer *et al.* [22] starts with a mark matrix, which is however undefined. Therefore, it is not possible to replicate their algorithm. The next stage of their method is to apply Conway's Game of Life to the mark matrix to generate a binary edge map.

Cloud models are the extension of fuzzy models, which combine fuzziness and randomness into the transformation between the qualitative linguistic description and the quantitative numerical values. Zhang *et al.* [47] combined a cloud model and cellular automata in edge detection. They used the direction information, the neighbour edge intensity, and the width of the neighbour edge isolation as the input to a cloud reasoning system, which produces a binary edge map as the mark matrix. The same CA rules as Chang *et al.*'s [5] are then used to link the edge pixels. Their experiments showed that the proposed method can detect edges appropriately, but the resulting edges are wide.

5.4.1 A Simple Edge Linking Scheme

To link disconnected thin edges using cellular automata, we propose a simple four-step method whose input is a thresholded edge map. Some of the rules are based on the 3×3 Moore neighbourhood, while other rules use just a 1×3 neighbourhood. In addition to the two states of the initial image, two more states are introduced, to represent the ends of open curves and 'T'-junctions. These extra states enable us to use a 1D rather than 2D neighbourhood at times. The rules use X, the crossing number at a pixel which is the number of transitions from white to black and vice versa when the pixel's eight neighbours are traversed in a circular fashion.

Essentially, each iteration of the CA extends open end points of edge curves by one pixel. Any further extension is terminated if the end point touches an existing edge (i.e. the point becomes a junction). After completing the extension of edges, those that did not reach an existing edge, i.e. end points at the end of open curves, are contracted back to their initial length.

Using the 3×3 neighbourhood, edges can be extended in the eight principal directions (0°, 45°, ...). If a larger neighbourhood were used, then not only could more orientations be used, but curved extensions could also be accommodated.

1. **detect end points (one iteration):**
   if p = `edge` and $X = 2$ then p = `end`
2. **extend open ends (n iterations):**
   a. **detect junctions (one iteration):**
      if p = `end` and $X = 6$ then p = `junction`
   b. **extend open ends (one iteration):**
      if p = `end` and number of non-background = 1
      and any of the 1D patterns in figure 5.1 match then update
3. **contract open ends (n iterations):**
   if p = `end` or `junction` and number of non-background = 1
   then p = `non-edge`
4. **relabel pixels:**
   if $p \neq$ `non-edge` then p = `edge`

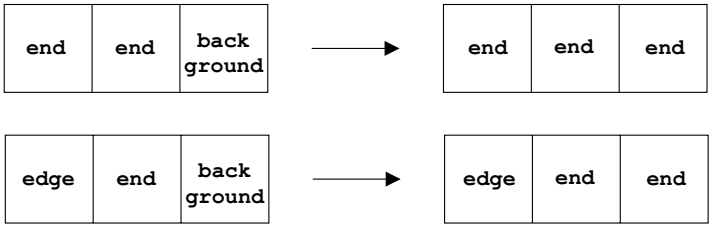
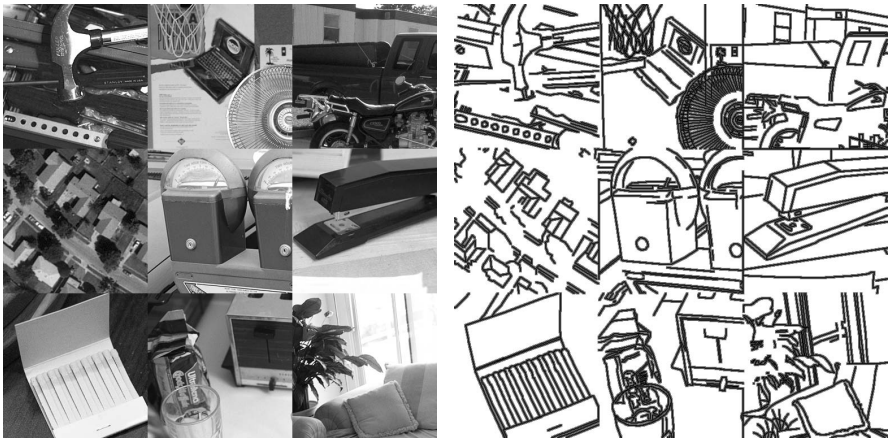


Fig. 5.1 1D CA rules for extending open ends for edge linking. The rules use a 1×3 pixel neighbourhood containing pixel labels: `edge`, `end`, `background`. All 45° rotated versions are also required.

5.5 Experiments

In this section we show results of applying several cellular automata edge detection methods. We start with Rosin's [34] method; the original training data was a 750×750 image mosaic containing sub-images from the University of South Florida data set which contains images along with manually generated ground truth edges; see figure 5.2. Note that in this chapter all edge maps are inverted for display purposes. Since there is likely to be some positional error in the ground truth edges (which are one pixel wide) the target edge map was dilated twice, with the new edges set each time to an increasingly lower intensity.

The CA rules were tested on four independent images, not included in the training data (figure 5.3), and produced the results shown in figure 5.4. The cellular automaton converged after a single iteration of the single rule. It can be seen that the results are fairly similar to the Sobel edge maps.

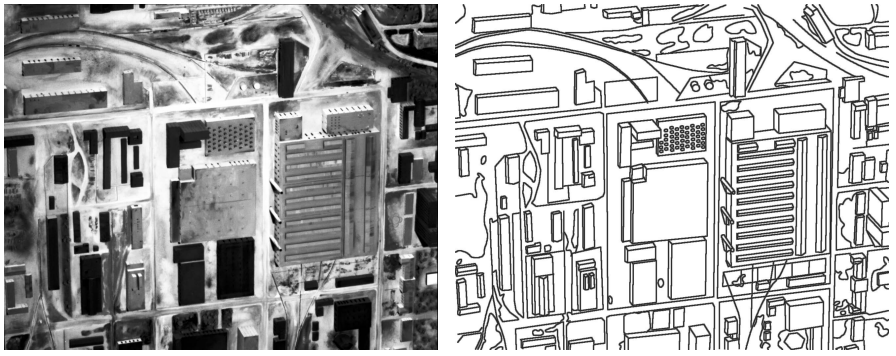


a b

Fig. 5.2 USF training data: (a) input image, (b) target ground truth image (inverted for display purposes)

Fig. 5.3 Sample test images containing indoor, outdoor, man-made and natural scenes. Also are shown the Sobel edge maps for comparison.

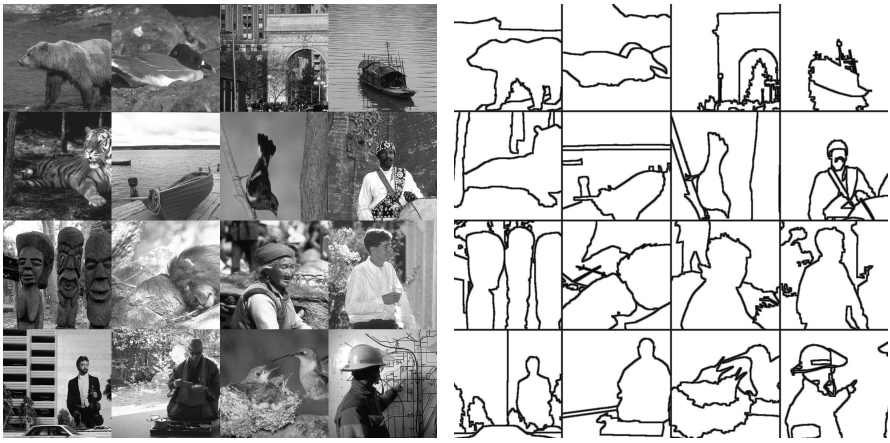
Fig. 5.4 Edges detected using Rosin's [34] method trained on data in figure 5.2



a b

Fig. 5.5 RADIUS training data: (a) input image, (b) target ground truth image

For comparison, several other training sets were used to learn rule sets for edge detection. Figure 5.5 shows the 1314×1044 image `J25` of a 40×40 inch model board especially created for the RADIUS project, along with its associated manually generated ground truth edges. After training a single rule was learnt that was identical to that learnt for the USF data set.



a b

Fig. 5.6 BSDS training data; (a) input image, (b) target ground truth image

Fig. 5.7 Edges detected using Rosin's [34] method trained on data in figure 5.6

Figure 5.6 shows a 1200×1200 image mosaic created from a subset of sub-images from the BSDS300 Berkeley Segmentation Dataset and Benchmark. Whereas the previous two sets of ground truth were explicitly made up of edges, the BSDS300 contains object boundaries. Since objects are defined at a higher semantic level than low level edges, it can be seen that there is often a poor correspondence between the two, and thus it poses a greater challenge to the cellular automaton. A larger set of rules is learnt from this data compared to the previous training data. The new rules consist of the single rule learnt previously plus another eight. The effect (see figure 5.7) is to emphasise the corners more than previously, and to reduce the response at some edges. Since there was a less direct match between the

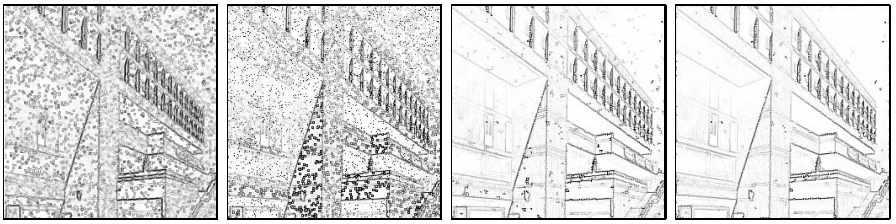


Fig. 5.8 Edge detection applied to an image which has had salt and pepper noise added. a) Sobel, b) CA rules learnt from training set in figure 5.2, c) CA rules learnt from a version of the training set in figure 5.2 with added salt and pepper noise, d) CA rules learnt from noisy data applied to both the image and an inverted version.

Fig. 5.9 Edges detected using Wongthanavasu and Lursinsap's [45] method (upper row: single iteration; lower row: iterated until convergence)

training source data and the ground truth, the CA requires more iterations to achieve convergence (typically about four iterations).

A final example of Rosin's method [34] is given where the USF training data had salt and pepper noise with probability of 0.1. This enables a set of seven rules to be learnt that are robust to similar noise. Results are shown in figure 5.8 for edge detection on a noisy version of the MIT image. Application of the Sobel produces a very noisy edge map (figure 5.8a), while applying a CA with the original rule learnt from the clean USF data also fares poorly (figure 5.8b). Using the set of rules learnt from the noisy training data produces a much better result (figure 5.8c), requiring on average about fifteen iterations of the rules. However, since the rule learnt for edge detection is restricted to inverting white pixels then the 'salt' (white noise) is effectively removed, but the 'pepper' (black noise) remains. The solution taken in Rosin [34] is to also apply the rules to the inverted image so as to remove the

Fig. 5.10 Edges detected using Popovici and Popovici's [26] method (upper row threshold = 4; middle row threshold = 16; lower row threshold = 64)

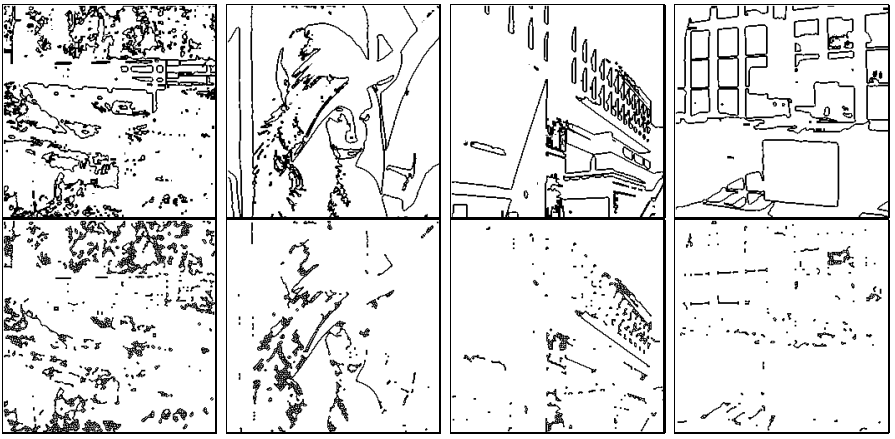


Fig. 5.11 Edges detected using Diwakar *et al.*'s [7] method (upper row: single iteration; lower row: iterated until convergence)

Fig. 5.12 Binary edge maps (left) post-processed to link fragmented edges by running five iterations of the rules described in section 5.4.1 (right); inserted edgels are coloured red. The bottom row shows two close ups from the second example.

inverted pepper. The logical AND operation is applied to the two edge maps which effectively eliminates both salt and pepper (figure 5.8d).[2]

[2] There can also be a single pixel translation between the edge responses of the two outputs, and so better results were achieved by translating one of the images by $\{-1,0,1\}$ in X and Y before the logical AND operation. The outputs were then combined using a logical OR operation.

We now show results for a variety of other CA based edge detectors. In figure 5.9 is shown Wongthanavasu and Lursinsap's [45] method. For a single iteration the results look reasonable. Additional iterations degrade the results.

Figure 5.10 shows results from Popovici and Popovici's [26] method (which converged after three iterations).[3] They require an application specific threshold parameter to be specified, and it can be seen to alter the density of the detected edges. The best results in figure 5.10 might be considered to be with the threshold set to 16, although even then the results are generally inferior, containing thick edge regions whilst also retaining many scattered and disconnected single pixel edges.

Figure 5.11 shows results from Diwakar *et al.*'s [7]. As expected, since the method is based on global thresholding, it has missed many edges and found many spurious ones. If more than a single iteration of the CA is run, the results degrade even further.

The results of applying a post-processing step of edge linking are shown in figure 5.12. The input image is edge detected using Rosin's method [34], and then thresholded to create a binary edge map. In addition, isolated edge pixels were removed. The linking method described in section 5.4.1 was applied for five iterations. It can be seen that many fragmented edges have been successfully linked (as shown by the red edgels).

5.6 Conclusions

As stated in the introduction, it is difficult for the general reader to gain an understanding of the state of the art in cellular automata based edge detection since papers are dispersed over many conferences and journals. Our brief survey shows that there exists a relatively large number of relevant papers, although a number of them were not clearly written, with details missing or occasionally inconsistent. Moreover, a number of papers are misleading, in that, according to common usage within image processing, they actually perform (binary image) boundary detection rather than (intensity image) edge detection.

The papers and the results of the experiments included in this chapter demonstrate that cellular automata are indeed capable of performing edge detection, i.e. process a grey level input image to produce an edge magnitude image (either binary or e.g. 256 values) as output. The results were of variable quality, but in order to be able to confidently evaluate and compare edge detectors a more systematic and quantitative analysis should be carried out. This has not been done to date.

The experiments revealed that for all the CA only a very few iterations were necessary to achieve their optimal results (such details were often missing from the descriptions of the methods in their original publications). Specifically, the methods of Rosin [34] (when trained on the USF dataset), Wongthanavasu and Lursinsap [45] and Diwakar *et al.* [7] should generally run for only a single iteration.

[3] Popovici and Popovici's [26] paper described a von Neumann neighbourhood, but we found better results (those shown in figure 5.10) to be obtained using a Moore neighbourhood.

Although for Rosin's method the CA converged after that iteration, this was not the case for the other methods, whose results steadily degraded when further iterations were applied. Popovici and Popovici's [26] method converged after three iterations, and Rosin's [34] method also converged after a similar number of iterations when trained on the BSDS300 dataset. This suggests that none of the above approaches are using the full power of CA to capture more global image structure by propagating information across the image via a larger number of iterations. The version of Rosin's method [34] trained on the noisy version of the USF training data required more (typically fifteen) iterations, and this is consistent with the denoising rules in [32]. The latter were found to be competitive with alternative denoising methods, and also required several tens of iterations of the rules (depending on the level of noise).

Nevertheless, cellular automata based edge detection holds promise since it is computationally efficient, and can moreover be tuned to specific domains (i.e. applications and/or image types) by appropriate selection/learning of rules. Not only that, but pre-processing and post-processing stages such as noise filtering, thinning and edge linking can also be easily included in the cellular automata framework.

References

1. Avidan, S., Shamir, A.: Seam carving for content-aware image resizing. ACM Trans. Graph. 26(3), 10 (2007)
2. Baştürk, A., Günay, E.: Efficient edge detection in digital images using a cellular neural network optimized by differential evolution algorithm. Expert Syst. Appl. 36(2), 2645–2650 (2009)
3. Batouche, M., Meshoul, S., Abbassene, A.: On solving edge detection by emergence. In: Ali, M., Dapoigny, R. (eds.) IEA/AIE 2006. LNCS (LNAI), vol. 4031, pp. 800–808. Springer, Heidelberg (2006)
4. Canny, J.: A computational approach to edge detection. IEEE Trans. Pattern Analysis and Machine Intelligence 8, 679–698 (1986)
5. Chang, C., Zhang, Y., Gdong, Y.: Cellular automata for edge detection of images. Int. Conf. on Machine Learning and Cybernetics 6, 3830–3834 (2004)
6. Chen, Y., Yan, Z.: A cellular automatic method for the edge detection of images. In: Huang, D.-S., Wunsch II, D.C., Levine, D.S., Jo, K.-H. (eds.) ICIC 2008. LNCS (LNAI), vol. 5227, pp. 935–942. Springer, Heidelberg (2008)
7. Diwakar, M., Patel, P., Gupta, K.: Cellular automata based edge-detection for brain tumor. In: Advances in Computing, Communications and Informatics, pp. 53–59 (2013)
8. Ens, J., Lawrence, P.: An investigation of methods for determining depth from focus. IEEE Trans. Pattern Analysis and Machine Intelligence 15(2), 97–108 (1993)
9. Georgilas, I., Gale, E., Adamatzky, A., Melhuish, C.: UAV horizon tracking using memristors and cellular automata visual processing (2013)
10. Gharehchopogh, F., Ebrahimi, S.: A novel approach for edge detection in images based on cellular learning automata. Int. J. Computer Vision and Image Processing 2(4), 51–61 (2012)
11. Gorsevski, P., Onasch, C., Farver, J., Ye, X.: Detecting grain boundaries in deformed rocks using a cellular automata approach. Computers & Geosciences 42, 136–142 (2012)

12. Heath, M., Sarkar, S., Sanocki, T., Bowyer, K.: Robust visual method for assessing the relative performance of edge detection algorithms. IEEE Trans. Pattern Analysis and Machine Intelligence 19(12), 1338–1359 (1997)
13. Heath, M.D., Sarkar, S., Sanocki, T.A., Bowyer, K.W.: Comparison of edge detectors: A methodology and initial study. Computer Vision and Image Understanding 69(1), 38–54 (1998)
14. Kazar, O., Slatnia, S.: Evolutionary cellular automata for image segmentation and noise filtering using genetic algorithms. Journal of Applied Computer Science and Mathematics 5(10), 33–40 (2011)
15. Kumar, T., Sahoo, G.: A novel method of edge detection using cellular automata. International Journal of Computer Applications 9(4), 38–44 (2010)
16. Lee, M., Bruce, L.: Applying cellular automata to hyperspectral edge detection. In: Int. Geoscience and Remote Sensing Symposium, pp. 2202–2205 (2010)
17. Li, H., Liao, X., Li, C., Huang, H., Li, C.: Edge detection of noisy images based on cellular neural networks. Communications in Nonlinear Science and Numerical Simulation 16(9), 3746–3759 (2011)
18. Martin, D., Fowlkes, C., Malik, J.: Learning to detect natural image boundaries using local brightness, color, and texture cues. IEEE Trans. Pattern Analysis and Machine Intelligence 26(5), 530–549 (2004)
19. Men, H., Zhang, J., Wang, C.: Measurement of inhibition zone based on cellular automata edge detection method. In: Int. Workshop on Education Technology and Computer Science, vol. 2, pp. 357–360 (2009)
20. Mirzaei, K., Motameni, H., Enayatifar, R.: New method for edge detection and denoising via fuzzy cellular automata. Int. J. Phy. Sci. 6(13), 3175–3180 (2011)
21. Otsu, N.: A threshold selection method from gray-level histograms. IEEE Trans. SMC 9, 62–66 (1979)
22. Peer, M., Qadir, F., Khan, K.: Investigations of cellular automata game of life rules for noise filtering and edge detection. Int. J. Information Engineering and Electronic Business 4(2), 22–28 (2012)
23. Perona, P., Malik, J.: Scale-space and edge detection using anisotropic diffusion. IEEE Trans. Pattern Analysis and Machine Intelligence 12(7), 629–639 (1990)
24. Piao, Y., Kim, S., Cho, S.J.: Two-dimensional cellular automata transforms for a novel edge detection. In: IComputability in Europe 2008, Logic and Theory of Algorithms (2008)
25. Pluim, J.P.W., Maintz, J.B.A., Viergever, M.A.: Image registration by maximization of combined mutual information and gradient information. IEEE Trans. Med. Imaging 19(8), 809–814 (2000)
26. Popovici, A., Popovici, D.: Cellular automata in image processing. In: Int. Symp. on the Mathematical Theory of Networks and Systems (2002)
27. Priego, B., Bellas, F., Souto, D., López-Peña, F., Duro, R.: Evolving cellular automata for detecting edges in hyperspectral images. In: Int. Conf. on Fuzzy Systems, pp. 1–6 (2012)
28. Pudil, P., Novovicova, J., Kittler, J.: Floating search methods in feature-selection. Pattern Recognition Letters 15(11), 1119–1125 (1994)
29. Qadir, F., Khan, K.: Investigations of cellular automata linear rules for edge detection. Int. J. Computer Network and Information Security 3, 47–53 (2013)
30. Qadir, F., Peer, M., Khan, K.: Efficient edge detection methods for diagnosis of lung cancer based on two-dimensional cellular automata. Advances in Applied Science Research 3(4), 2050–2058 (2012)

31. Roberts, L.: Machine Perception of Three-Dimensional Solids. In: Outstanding Dissertations in the Computer Sciences. Garland Publishing, New York (1963)
32. Rosin, P.: Training cellular automata for image processing. IEEE Trans. on Image Processing 15(7), 2076–2087 (2006)
33. Rosin, P.: A simple method for detecting salient regions. Pattern Recognition 42(11), 2363–2371 (2009)
34. Rosin, P.: Image processing using 3-state cellular automata. Computer Vision and Image Understanding 114(7), 790–802 (2010)
35. Sahota, P., Daemi, M., Elliman, D.: Training genetically evolving cellular automata for image processing. In: Int. Symp. Speech, Image Processing and Neural Networks, pp. 753–756 (1994)
36. Sato, S., Kanoh, H.: Evolutionary design of edge detector using rule-changing cellular automata. In: Nature & Biologically Inspired Computing, pp. 60–65 (2010)
37. Selvapeter, J., Hordijk, W.: Genetically evolved cellular automata for image edge detection. In: Proceedings of the International Conference on Signal, Image Processing and Pattern Recognition, SIPP 2013 (2013)
38. Selvapeter, P.J., Hordijk, W.: Cellular automata for image noise filtering. In: Nature & Biologically Inspired Computing, pp. 193–197 (2009)
39. Senthilkumar, S., Piah, A.R.M.: An improved fuzzy cellular neural network (IFCNN) for an edge detection based on parallel RK(5,6) approach. International Journal of Computational Systems Engineering 1(1), 70–78 (2012)
40. Shin, M.C., Goldgof, D.B., Bowyer, K.W.: Comparison of edge detector performance through use in an object recognition task. Computer Vision and Image Understanding 84(1), 160–178 (2001)
41. Slatnia, S., Batouche, M., Melkemi, K.E.: Evolutionary cellular automata based-approach for edge detection. In: Masulli, F., Mitra, S., Pasi, G. (eds.) WILF 2007. LNCS (LNAI), vol. 4578, pp. 404–411. Springer, Heidelberg (2007)
42. Suyi, L., Qian, W., Heng, Z.: Edge detection of fabric defect based on fuzzy cellular automata. In: Int. Workshop on Intelligent Systems and Applications, pp. 1–3 (2009)
43. Wongthanavasu, S.: Cellular automata for medical image processing. In: Salcido, A. (ed.) Cellular Automata – Innovative Modelling for Science and Engineering, pp. 395–410 (2011)
44. Wongthanavasu, S., Lursinsap, C.: A 3-D CA-based edge operator for 3-D images. In: Int. Conf. Image Processing, pp. 235–238 (2004)
45. Wongthanavasu, S., Sadananda, R.: A CA-based edge operator and its performance evaluation. J. Visual Communication and Image Representation 14(2), 83–96 (2003)
46. Yang, C., Ye, H., Wang, G.: Cellular automata modeling in edge recognition. In: 7th Int. Symp. on Artificial Life and Robotics, pp. 128–132 (2002)
47. Zhang, K., Zhang, W., Yuan, J.: Edge detection of images based on cloud model cellular automata. In: Chinese Control Conference, pp. 249–253 (2008)