ROMANIAN JOURNAL OF INFORMATION SCIENCE AND TECHNOLOGY
Volume 19, Number 4, 2016, 338–359

Training Cellular Automata for Image Edge Detection

Anand Prakash Shukla

Department of Computer Science and Engineering,
KIET Group of Institutions, Ghaziabad, INDIA
Email: anandskla@gmail.com

Abstract. Cellular automata can be significantly applied in image processing tasks. In this paper, a novel method to train two dimensional cellular automata for detection of edges in digital images has been proposed and experiments have been carried out for the same. Training of two dimensional cellular automata means selecting the optimum rule set from the given set of rules to perform a particular task. In order to train the cellular automata first, the size of rule set is reduced on the basis of symmetry. Then the sequential floating forward search method for rule selection is used to select the best rule set for edge detection. The misclassification error has been used as an objective function to train the cellular automata for edge detection. The whole experiment has been divided in two parts. First the training was performed for binary images then it is performed for gray scale images. A novel method of thresholding the image by Otsu's algorithm and then applying the cellular automata rules for the training purpose has been proposed. It has been observed that the proposed method significantly decreases the training time without affecting the results. Results are validated and compared with some standard edge detection methods both qualitatively and quantitatively and it is found better in terms of detecting the edges in digital images. Also the proposed method performs much better in corner detection as compared to the standard edge detection methods.

Keywords: Cellular Automata, Training of Cellular Automata,Sequential Floating Forward Search Algorithm, Misclassification Error, Otsu's Algorithm, Corner Detection

1 Introduction

Researches show that any complex system is a system of several elements, operating in parallel, with neighborhood relationships that as a whole exhibit emergent global behavior. In other words, the complex systems are not complicated, but these are the combinations of the many simple systems working together in parallel. Cellular automata also exhibits the same. This feature of simplicity of cellular automata and the ability to model complex systems has attracted the researchers' attention from a wide range of fields.

Concept of cellular automata was introduced by John Von Neumann in the early 1950s. The cellular automata are simple mathematical models to investigate self-organization and self-reproduction. Cellular Automata is also called system of Finite Automata, i.e. Deterministic Finite Automata (DFA) arranged in an infinite, regular lattice structure. In cellular automata, state of a cell, at the next time step is determined by the current states of the surrounding neighbors of cells along with its present state. It is updated synchronously in discrete time steps. Mathematically, the cellular automata is defined by a triplet A = (S, N, δ) where, S is a non-empty set of states, N is the neighborhood system, and $\delta : S^N \rightarrow S$ is the local transition function (rule) which defines how a cell changes its state. Commonly used neighborhood systems are the von Neumann and Moore neighborhoods.

Von Neumann Neighborhood The von Neumann neighborhood of range r is defined by equation 1.

$$N_{x_0,y_0} = \{(x,y) : |x - x_0| + |y - y_0| \leq r\} \tag{1}$$

A diamond-shaped neighborhood that can be used to define a set of cells surrounding a given cell (x_0, y_0) that affects the evolution of a two-dimensional cellular automaton on a square grid. The von Neumann neighborhood is illustrated in figure 1.

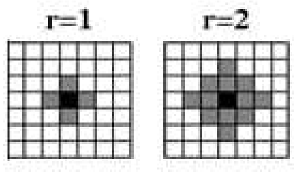


Fig. 1. von Neuman Neighborhood

Moore Neighborhood The Moore neighborhood of range r is defined by equation 2

$$N_{x_0,y_0} = \{(x,y) : |x - x_0| \leq r, |y - y_0| \leq r\} \tag{2}$$

A square-shaped neighborhood can be used to define a set of cells surrounding a given cell (x_0, y_0) that affects the evolution of a two-dimensional cellular automaton on a square grid. Moore neighborhood is illustrated in figure 2.

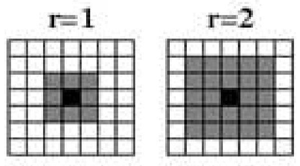


Fig. 2. Moore Neighborhood

1.1 Use of Cellular Automata in Image Processing

The digital image is considered as a two dimensional array of $M \times N$ pixels as shown in figure 3. Each pixel can be characterized by the triplet (i; j; k) where (i; j) represents its position in the array and k represents the associated color. The image may be then considered as a particular configuration of a cellular automaton that occupies the cellular space of $M \times N$ array defined by the image. Each pixel of the image represents a cell of the cellular automata and the state of the cell is defined by the value of the pixel in image. [17]

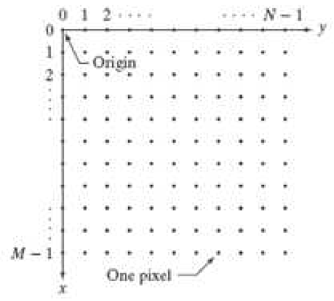


Fig. 3. Pixel representation of Digital Image.

Cellular automata has number of advantages over traditional methods of computation

- Simplicity of implementation and complexity of behavior : It is found that the cellular automata based systems can be implemented easily as each cell generally works on few simple rules, but a combination of these cells leads to a more sophisticated global behavior.
- CA are both inherently parallel and computationally simple.
- CA are extensible,that is, we can extend the simple rules by using some new computation techniques.
- One of the most important features of the CA method is that it supports n-dimensions and m-label categories where the number of labels does not increase computational time or complexity
- Allow efficient parallel processing of several tasks,
- Users are able to make corrections and modifications at any time during operations i.e. Cellular automata is more interactive.

Some of the applications in field of image processing are noise filtering, edge detection, connected set morphology, and segmentation.

2 Related Work

Training of cellular automata for an image processing task means the selection of the optimal rule set among all the possible rules which perform best for that particular task. So far most of the work on cellular automata studies the effect of applying manually designed transition functions(rules). The inverse of this i.e., obtaining appropriate rules to produce a desired effect is very hard. Even if only the binary images are considered the combinations of all possible rules are very large. Somol et al. [27] have shown that in some situations efficient feature selection can be performed using branch and bound algorithms and it is also tractable. However, Cover et al. [4] have shown that in general, without an exhaustive enumeration of all combinations an optimal selection of rules cannot be guaranteed and this is clearly impractical. One simple alternative is to measure the effect of each rule separately and then use the results for the direct construction of combinations of rules. However, practically some methods are required for revealing at least some of the inter rule combinations.

Some authors prefer the evolutionary solutions for training of cellular automata, for example, Mitchell et al. [12] have used genetic algorithm to find the solution of the density classification problem. There are some difficulties reported by the authors themselves as breaking of symmetries in early generations (for short-term gains) the training data became too easy for the cellular automata in later generations of the genetic algorithm. This problem was solved by Julle et al. [11] by using genetic algorithms with co-evolution. In this method, to perform effective training, the training set was not fixed during evolution, rather it is gradually increased as the difficulty arises. Once the cellular automata was trained for the initial solutions for simple problem, it would be improved and extended by evolving more challenging data. Instead of genetic algorithms, Andre et al. [1] used a standard genetic programming framework. But this method was computationally very expensive. Extending the density classification task to two-dimensional grids, J Morales et al. [10] applied standard genetic algorithm to train the cellular automata for density classification problem to two-dimensional grids.

Very less work has been done to perform the training of cellular automata for image processing. One of the practical problems of using cellular automata for training is the determination of the rule set. The traditional methods of specifying rules manually is a slow and tedious process because of which it is unattractive. Sahota et al. [20] used genetic algorithm for the tasks such as edge detection in binary images. In this work, a generalized system is set up that attempts to discover the precise cellular automata rules required to solve the edge detection problem. Genetic algorithm was used to locate the rules and once the system is trained it is able to process other images. Basically, this model works in two distinct stages - the training phase and the execution phase. Figure 4 shows this training process. In the training phase, the user must supply pairs of input and output images. The artificial images are used for this purpose. The model attempts to find rules for the cellular automata that will provide the desired type of processing. The input patterns are first mapped onto the rule array, after which the automaton is allowed to update for a fixed number of cycles. The cellular automata rule selector attempts to discover rapidly the rules that will produce the desired results by using genetic algorithm. The root mean square (RMS) error produced by each image pair is chosen as an objective function to drive the genetic algorithm. In the execution phase once an optimal rule set has been found for the desired problem the system is ready. New images are given as input to the trained cellular automata and it performs the task on the basis of the optimal rules found in training phase. The results of this work show that genetically evolved cellular automata can successfully learn the process of edge detection for binary noise-free images.

Slatnia and Kazar [26] have used an evolutionary process to find a rule set of cellular automata among a set of optimal rules for extracting edges in a binary image by applying genetic algorithm. The genetic algorithm was initialized on the basis of

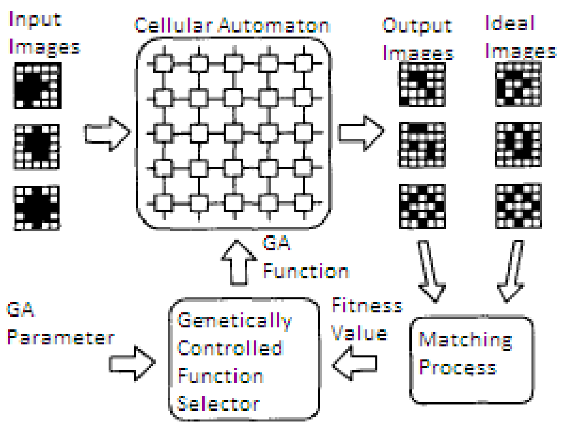


Fig. 4. Model for Training Cellular Automata by Genetic Algorithm [20].

random construction of the rule packets extracted from the neighborhood model as shown in figure 5. For each cellular automata rule packets(similar rule according to its neighborhood) are searched in the image then the found packet modifies the central pixel according to the defined transition. Thereafter, distance between the result of the edges detected by this method and the ideal edges are calculated along with the error of miss-classed pixels. Again this method generates a new population by applying selection, crossover and mutation by using the edge detection described above. This process iterates until there is no further improvement in the objective function or for a fixed maximum number of iterations.

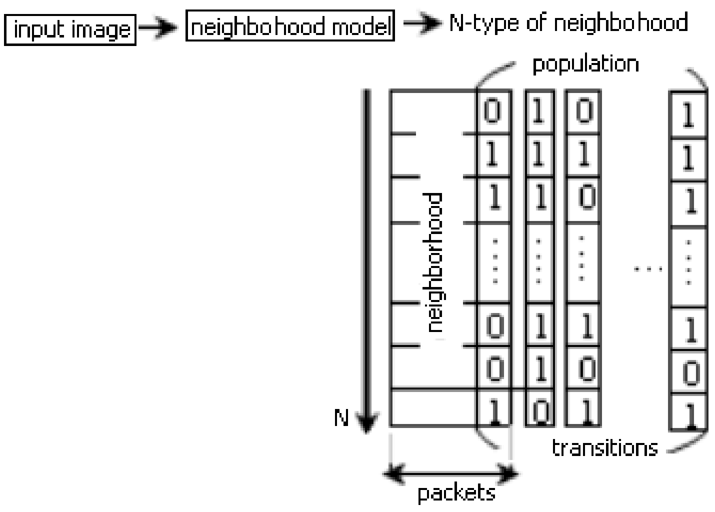


Fig. 5. Construction of Neighborhood Model [26].

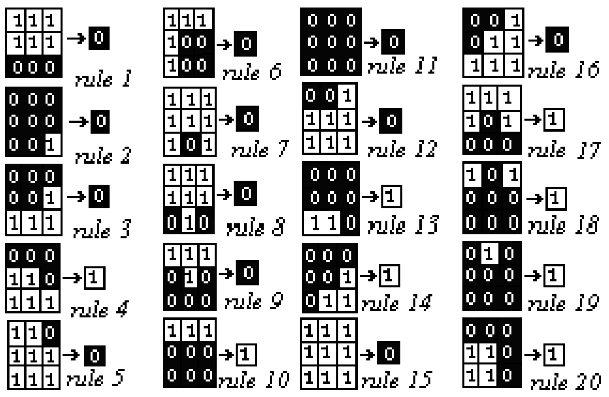


Fig. 6. Best packet of cellular automata rules found by genetic algorithm [26].

The effort of Paul L. Rosin in the field of training of cellular automata is appreciable. In his work [18, 19], the training of cellular automata for the different image processing tasks has been performed efficiently. First, the possible rule set is reduced from 256 to 51 on the basis of symmetry and reflection for central black and white pixels. Thereafter, the training based on this reduced rule set is performed. Experiments are performed for the following conditions.

1. Rule sets can be learned and applied separately for both central black and white pixels.

2. The two rule sets(with central black and white pixel) are considered equivalent

3. Just one of the black and white rule set is considered appropriate and the other is ignored.

In this work, each rule pattern is matched with the neighborhood pattern of pixels of the image. If it is matched then the color of the central pixel is inverted. Before processing, each rule is converted to an 8 bit string pattern and then matched with the converted pattern of the image pixel neighborhood. For the training purpose the Sequential Floating Forward Search (SFFS) [15] method for the feature selection has been used. To direct the SFFS method there is a requirement of the objective function. Root Mean Square (RMS) error between the input and target image is considered as the objective function.

In order to train the cellular automata for filtering of salt and pepper noise, the above mentioned algorithm is applied and the cellular automata is trained by SFFS method by using the RMS as objective function. It is found that at low level of noise only one rule is capable of performing the filtering of salt and pepper noise efficiently. As the level of noise increases more rules are added to the resulting set. The result is compared with the median filter and the mathematical morphological filter. It is found experimentally that the trained rule set performs better than other filters.

In case of filtering the gray level images, there is difficulty that the number of possible rules are too large. This is avoided by the threshold decomposition method. In this method the gray-level image is decomposed by thresholding at all possible gray levels into a set of corresponding binary images. Then the results of the trained rule set for the binary images is applied to each of the binary images obtained by threshold decomposition. Then the results are combined. In this work the results of filtering the binary images are simply added. The advantage of this method is that the result of binary image training is used directly. The disadvantage is that the decomposition process is a time taking process. The training of cellular automata for thinning of the black region has been also performed in this work [18]. As only the black portion is thinned, rules were applied in black pixels only. There are two ways to generate the training data. First, some one pixel wide curves were taken as the target output, and were dilated by varying amounts to provide the prethinned input. Along with that, some thinned binary images by applying the standard thinning algorithm are considered. Both sets of data were combined to form a composite training input and output image pair. The summed proportions of black pixel errors and white pixel errors was considered as the objective function to drive the SFFS algorithm. The resulting trained rule set contains the seven rules which are capable of performing the thinning of the black portion. The result is compared with the standard method by visual inspection and found satisfactory. Along with the above mentioned algorithms, Rosin [18] also considered two variants of the cellular automata which are the B-rule cellular automata and 2-cycle cellular automata. The aforesaid problems are also trained for these two variants of cellular automata and the results are found accordingly.

Similar experiments have also been performed for the training of cellular automata for noise filtering [21,24] and morphological operations [22,23,25]

3 Training Strategy

An image can be considered as a lattice of cells of two dimensional cellular automata where each pixel corresponds to the particular cell of the CA. In case of binary images, a pixel can acquire one of the two values, i.e. 0 and 1. If we consider the Moore neighborhood of the pixels then 2^8 i.e. 256 rules are possible for the central white pixel and the same for the central black pixel. By using PolyaBurnside counting lemma [16] this number can be reduced by determining the number of distinct patterns after removing equivalent symmetric versions. According to this lemma the number of equivalence classes are

$$Class = \frac{1}{P} \sum_{p \in P} |Fix(p)| \tag{3}$$

where Class is the number of equivalence classes, P is a set of permutations of a set A, and Fix(p) is the number of elements of A that are invariant under p.

In case of Moore neighborhood of the pixels, the number of equivalence classes in terms of the number of possible pixel values x is defined as

$$Class = \frac{x^8 + 2x^2 + x^4 + 4x^5}{8} \tag{4}$$

where the terms in the numerator are the 0^o rotation (identity) $\pm 90^o$(two rotations), 180^o(single), and four rotations corresponding to mirror symmetry through vertical, horizontal, and diagonal lines of reflection. For the binary image where the pixel values permitted are only 0 and 1. By putting the value of x as 2, equation 4 gives value 51. Hence, after removing the symmetry the reduced rule set contains 51 rules for center pixel 1 and similarly 51 rules for center pixel 0. The resulting rule set for center pixel 1 is shown in figure 7. The same set can be obtained for the center pixel 0 by replacing the center 1 by 0. Now the aim is to find the best rule set which is most suitable for the given problem(Edge detection).

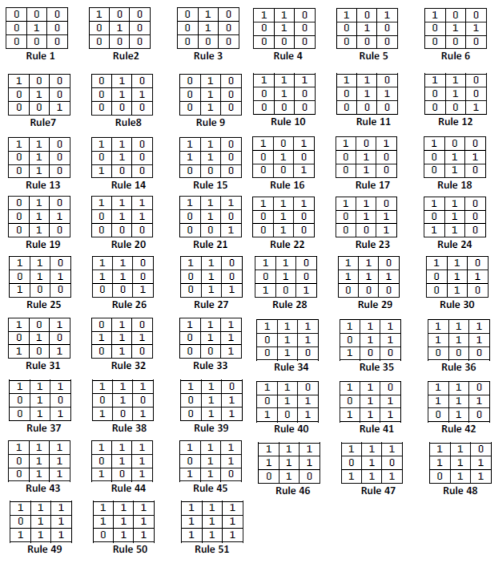


Fig. 7. Rule set with center pixel 1.

4 Training Strategy for Edge Detection

Edges are significant local changes of intensity in an image or in other words edges are points in a digital image at which the image brightness changes abruptly or has discontinuities and it typically occurs on the boundary between two different regions in an image. Edge detection is defined as a set of methods which aims to identify edges in a digital image. Edge detection is one of the important steps in image processing, computer vision techniques, image pattern recognition and image analysis because it may significantly reduce the amount of data to be processed and may therefore filter out less relevant information from the digital images but preserves the important structural properties of an image which simplifies the subsequent task of interpretation of the information content in the image.

In this section we concentrate on techniques for training CA rules to perform edge detection for the binary and gray scale images to a significant level of performance. The training process of CA takes a considerably longer time, but this is not a problem since it can be carried out offline. Once the CA is trained for the edge detection and resulting rule set is found then it can be directly applied to the images in which edge detection is required.

4.1 Training Strategy for Binary Images

In the current experiment the Moore neighborhood(eight way connected) described in the Introduction section of this paper is considered. Since the boundary pixels are not connected by eight pixels, so these pixels are neglected. This is called the fixed value boundary condition. The start state of each pixel in the cellular automata is considered as the pixel value of the input image.

4.2 Algorithm

In order to perform edge detection in the given binary image each rule set is considered among 51 rules shown in figure 7. The matching of the pattern is performed only in the white portion of the given image. Each pixel of the image is taken under consideration and if the neighborhood pattern(Moore neighborhood) is matched with the pattern of rule or set of rules under consideration then the central pixel is inverted to black, otherwise it remains unaltered. The black portion of the image remains unchanged. One of the challenges is to simulate the parallel behavior of the cellular automata in the sequential machine with a single processor and sequential operations. To resolve this issue the algorithm is designed in such a manner that in each iteration, all the image pixels are notionally processed in parallel. To perform this the processed pixels are stored in a secondary image instead, and then copied back to the original image at the end of each iteration. The pseudo code for this algorithm is given in algorithm 1.

Algorithm 1 Edge Detection Using CA Rules

```
Require: input image of size M $\times N$, Ruleset
1: $[row, column] \leftarrow$ size(A)
2: for $i = 2$ to $M - 1$ and $j = 2$ to $N - 1$ do
3:     for k = 1 to 4 do
4:         if A[i,j] = 1 and 3x3pattern(A[i,j]) = Rotate$90^o$(Rule_matrix) then
5:             $C[i,j] \leftarrow 0$
6:         else
7:             $C[i,j] \leftarrow 1$
8:         end if
9:     end for
10: end for
11: $B \leftarrow C$
```

In the algorithm 1 first, the rule set under consideration is taken into set RuleMatrix which is a set of 3x3 matrix. Then for each pixel in image with value 1, the 3x3 neighborhood of the pixel is converted into the matrix format. If this pattern is matched with the rules under consideration then the value of central pixel is inverted to zero. Otherwise it remains unchanged. The same steps are performed for rotating the 3x3 neighborhood pattern to four times each by 90^o so that the symmetric patterns can also be considered.

4.3 Computational Complexity

Even without the specialized hardware implementations that are available as [14], [28] and [7], the running time of the cellular automata is moderate. If there are N number of pixels, and a neighborhood size of M(here M=8 in case of Moore neighborhood), and K is the number of rules in the rule set, the computational complexity is O(MN). Note that the complexity is independent of the number of rules available in the Ruleset.

4.4 Sequential Floating Forward Search Method

Sequential Floating Forward Search (SFFS) [15] is a deterministic feature selection method and is very widely being used for building classifier systems. The reason behind using the SFFS method is

1. Its simplicity of implementation
2. It is deterministic(not randomize) and repeatable.
3. It provides good speed with effectiveness.
4. It does not require the many parameters as necessary for genetic algorithms. [5]

In the past some studies have compared the effectiveness of SFFS against other strategies for feature selection. For example, Jain and Zongker [9] found that the SFFS performed best among the fifteen feature selection algorithms including genetic algorithm. Some other studies such as Hao et. al [8] found, in their experiments on different data sets, that there was little difference in effectiveness between genetic algorithms and SFFS.

Algorithm 2 Sequential Floating Forward Search

```
1: Step 1:
2: $Y \leftarrow \{\phi\}$
3: Step 2:
4: Select the best feature
5: $x^+ \leftarrow argmax J(Y_k + x) | x \notin Y_k$
6: $Y_k \leftarrow Y_k + x^+$
7: k=k+1
8: Step 3:
9: Select the worst feature
10: $x^- \leftarrow argmax J(Y_k - x) | x \in Y_k$
11: Step 4:
12: if $J(Y_k - x^-) > J(Y_k)$ then
13:     $Y_k \leftarrow Y_k - x^-$
14:     $k = k + 1$
15:     Go to Step 3
16: else
17:     Go to Step 2
18: end if
```

The SFFS algorithm can be described as follows. Let Y_k denote the rule set at iteration k and its score be $J(Y_k)$. Here, $J(Y_k)$ is defined by the result of the algorithm 1 by applying the CA rule set to the input image and computing the objective function defined in the next subsection. Step 1 shows that the initial rule set is empty. In step 2 of each iteration, all rules in figure 7are considered for addition to the rule set. Only the rule giving the maximum score is added to the resulting rule set. This process is repeated until no improvement in score is gained by adding rules. In step 3, each rule in the rule set found in step 2 is removed to find the rule whose removal provides the resulting rule set with the improved value of objective function. As shown in step 4, if removal of a rule causes the better score of the objective function then it is discarded from the rule set and again the next rule is tried for the deletion and the process goes to step 3. Otherwise, the process goes to step 2 for the addition of a new rule to the rule set.

4.5 Objective Function

The objective function used to select the best rule set has a very crucial role. The simplest and most used objective function is Mean Square Error(MSE) or the Root Mean Squared (RMS) error between the processed and the reference image. However, it has some limitations [6] and they often do not capture the similarity seen by the human visual system(HVS). Wang et al. [29] proposed a method for image quality assessment which is Structural Similarity Index Measure(SSIM) which measures image similarity taking luminance, contrast and structure into account. But in case of edge detection even the SSIM index fails to detect edges since the SSIM index is more sensitive to the qualitative difference in the edge magnitude map that the CA is capable of producing. The misclassification error defined by Yasnoff et al. [30] has been considered which is best suited for the classification of the edge and non edge pixels. The misclassification error is defined as

$$error = 1 - \frac{|NE_o \cap NE_T| + |E_o \cap E_T|}{NE_o + E_o} \tag{5}$$

where E_o and NE_o show the edge and non edge pixels of the edge image obtained by the CA rule where as E_T and NE_T shows the target image which is considered as the reference image for the purpose of computation.

One of the major issues for checking the performance is to select the target image with which the comparison of results is to be made. Since the effectiveness of the algorithm depends on the target image hence its role is very important. In the case of binary images, the images generated by the Canny edge detector [3] are used as it is a well known optimal edge detector having the property of low error rate, good localization(The distance between edge pixels detected and real edge pixels is minimum) and minimal response(only one detector response per edge).

So, to find the objective function defined in equation 5 first, the edges of the given image are obtained by the Canny edge detector and the resulting image is considered as the reference image. Then the rule set is applied to the same image to obtain the resulting edge image. Then misclassification error between the reference image and the edge image obtained by the rule set is calculated. This error is used to drive the SFFS algorithm to find the best rule set for edge detection.

5 Training Strategy for Gray Scale Images

In case of gray scale images the training process is very much complicated. In case of Moore neighborhood where cells can have 256 possible intensities, 256^8 neighborhood patterns are possible for the central pixel with value 0. Also the same number of patterns for the central pixel with other values (1 to 255). If we reduce the number of rules from the rule set by using equation 4, still the number of classes are greater than 2×10^{18}, which is obviously too large to enumerate practically. One idea is that if the given gray scale image is converted to the binary level by threshold decomposition then the result of the previous section can be applied for the resulting image. Rosin [18] used the threshold decomposition for all possible gray level values and then combined the results of each decomposition to find the final image. This method works for the denoising and other methods but its effectiveness in case of edge detection is suspicious. Also, it is a very time taking process.

To over come the aforesaid problem, we propose a novel method to decompose the gray level image by Otsu's threshold decomposition method [13]. The motivation behind using the Otsu's method is

1. It is based on a very simple idea: Find the threshold that minimizes the weighted within-class variance.
2. This turns out to be the same as maximizing the between-class variance.
3. It is fast
4. It operates directly on the gray level histogram.

Due to the property of minimizing the within class variance and maximizing the between class variance Otsu's method is able to show the maximum objects present in the image into the thresholded image.

5.1 Otsu's Thresholding Method

Otsu's thresholding method is used to automatically perform clustering-based image thresholding in order to convert the gray scale image into a binary image. The algorithm assumes that the image contains two classes of pixels following a bi-modal histogram (foreground pixels and background pixels), it then calculates the optimum threshold separating the two classes so that their combined spread (intra-class variance) is minimal, or equivalently their inter-class variance is maximal.

Let the weighted sum of variances of the two classes be defined as:

$$\sigma_w^2(t) = w_0(t)\sigma_0^2(t) + w_1(t)\sigma_1^2(t) \tag{6}$$

where, weights w_0 and w_1 are the probabilities of the two classes separated by a threshold t and σ_0^2 and σ_1^2 are the variances of these two classes.

The class probability $w_0(t)$ and $w_1(t)$ is computed from the L histograms as:

$$w_0(t) = \sum_{i=0}^{t-1} p(i)$$

$$w_1(t) = \sum_{i=t}^{L-1} p(i)$$

Otsu shows that minimizing the intra-class variance is the same as maximizing the inter-class variance which is expressed in terms of class probabilities w and class means μ .

$$\sigma_b^2(t) = \sigma^2 - \sigma_w^2(t) = w_0(\mu_0 - \mu_T)^2 + w_1(\mu_1 - \mu_T)^2 = w_0(t)w_1(t)\left[\mu_0(t) - \mu_1(t)\right]^2 \tag{7}$$

where, the class mean $\mu_0(t)$, $\mu_1(t)$ and $\mu_T(t)$ is defined as:

$$\mu_0(t) = \sum_{i=0}^{t-1} i\frac{p(i)}{w_0}$$

$$\mu_1(t) = \sum_{i=t}^{L-1} i\frac{p(i)}{w_1}$$

$$\mu_T = \sum_{i=0}^{L-1} ip(i)$$

The class means and class probabilities can be computed iteratively. The steps of Otsu's thresholding is shown in algorithm 3

Algorithm 3 Otsu's Thresholding Algorithm

```
1: Compute histogram and probabilities of each intensity level
2: Set up initial $w_i(0)$ and $\mu_i(0)$
3: For all possible thresholds t=1,... maximum intensity do
        Update $w_i$ and $\mu_i$
        Compute $\sigma_b^2(t)$
4: Desired threshold corresponds to the maximum $\sigma_b^2(t)$
```

To select the reference image for the calculation of the misclassification error which is used as the objective function for the gray scale images, images from the University of South Florida data set which contains images along with manually generated ground truth edges [2] are considered. The reference images are dilated first because there may be a possibility of some positional error in the ground truth edges (which are one pixel wide).

6 Experiments and Results

In order to perform the experimental work and the validation of the results, the whole process is divided into two major parts. First, the training of cellular automata was done for the binary images and the best rule set for edge detection was learnt. Second, the training was performed for the gray scale images. Then the training process is applied to the test images and results are verified. To perform the experiment MATLAB 2014a environment has been used for the simulation purpose.

6.1 Training Rule Set for Binary Image

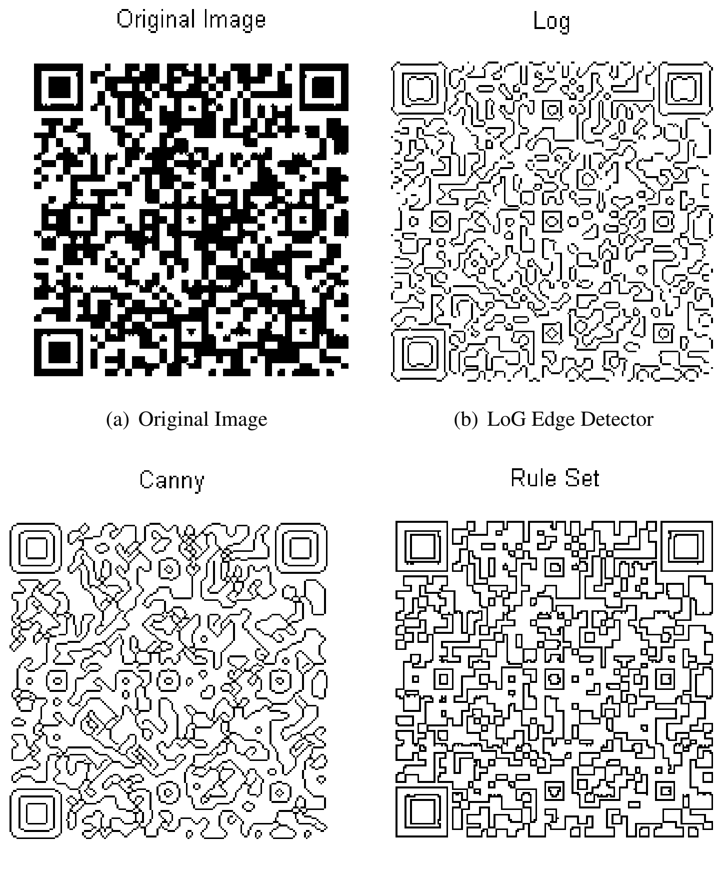
In order to train the rule set 20 binary sample images are carefully selected in such a way that the images contain all kind of shapes such as lines, circles, cones and some complicated structures. Edges of all selected images are obtained by Canny's edge detector and these edge images are considered as reference images. Then the rules are selected by using the objective function misclassification error with reference to these images. The Sequential Floating Forward Search(SFFS) method as described in the previous section was used for the training purpose. In all the images the trained rule set which gives the best edge detection result is rule number 51 as shown in figure 8. It should be noted that the rule set that was generated in figure 8 is simple, consisting of only one rule. The careful analysis of the rule shows that in the white portion of the image any pixel in a 3 $\times$ 3 neighborhood has all pixels white then it inverts the central pixel to black. Since in our algorithm of the CA the rule is only applied to white pixels, then all white pixels are replaced by black except for pixels adjacent to black pixels in the input image. This leaves a black image with a one pixel wide white edges.

1	1	1
1	1	1
1	1	1

Rule 51

Fig. 8. Rule set learned for edge detection

The learned rule is applied to 20 binary images and the results are analyzed. The result is also compared with some standard edge detector operators such as Roberts, Sobel, Prewitt, Laplacian of Gaussian(LoG) and with Canny edge detectors. The results are shown in figure 9. Note that in the figures shown all edge maps are inverted and figures are scaled down for display purposes. In the current section only results of two images are shown.



(a) Original Image (b) LoG Edge Detector

(c) Canny Edge Detector (d) Rule Set Edge Detector

Fig. 9. Results of Edge Detectors

The careful observation of the results by visual inspection shows that the trained rule set which contains only one rule is capable of detecting edges very efficiently. Also, its result is superior than other edge detectors. It performs well in junction while other edge detectors like Canny edge detector fail to show the sharp edges at junctions. This is shown in figure 9.

It is clearly depicted in figure 9 that the performance of rule set at the corner is superior than the canny and LoG operator. Both of these operators are not able to show the junction properly. So, the learnt rule set is capable of detecting the edges whereas most of the traditional methods fail.

6.2 Training Rule Set for Gray Scale Images

In order to train the cellular automata for the gray scale images, 12 images from the University of South Florida data set which contains images along with manually generated ground truth edges are taken as reference images. As described in the previous section (training strategy for gray scale image) the selected image is thresholded to convert in binary image by the Otsu's threshold decomposition algorithm. For the display purpose, the given edge image is inverted before using it as the reference image since it is the inverted version of original image. As we know that after threshold decomposition the image is converted to binary and we may directly use the result of the training result of binary image. But for satisfaction, the same experiment is again performed with the ground truth images as reference image and as expected, the learned rule set is same i.e., the rule number 51 as shown in figure 8.

The learned rule is applied to 10 ground truth images and the objective function (misclassification error) is calculated. The results of edge detection using the various standard edge detectors are found. Figure 10 shows the original sample images. Figure 11 shows the ground truth images showing the edges of the images shown in figure 10. Images of figure 11 are used as reference images and the misclassification errors are calculated with reference to these images.

(a) 101.pgm

(b) 103.pgm

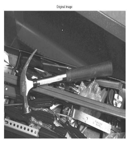
(c) 104.pgm

(d) 106.pgm

(e) 108.pgm

(f) 109.pgm

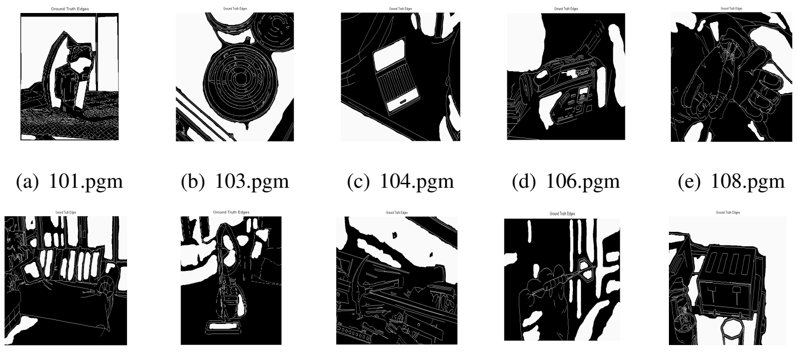
(g) 110.pgm


(h) 125.pgm

(i) 126.pgm

(j) 131.pgm

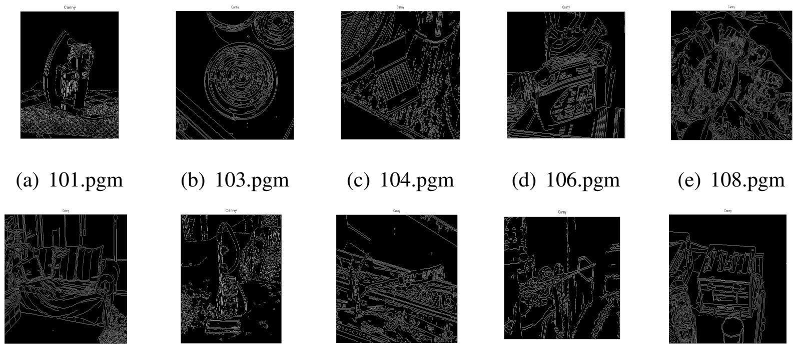
Fig. 10. Original Images



(a) 101.pgm (b) 103.pgm (c) 104.pgm (d) 106.pgm (e) 108.pgm

(f) 109.pgm (g) 110.pgm (h) 125.pgm (i) 126.pgm (j) 131.pgm

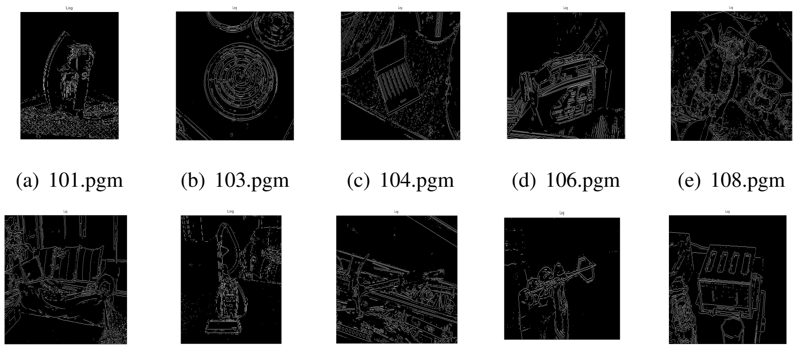
Fig. 11. Ground Truth Images



(a) 101.pgm (b) 103.pgm (c) 104.pgm (d) 106.pgm (e) 108.pgm

(f) 109.pgm (g) 110.pgm (h) 125.pgm (i) 126.pgm (j) 131.pgm

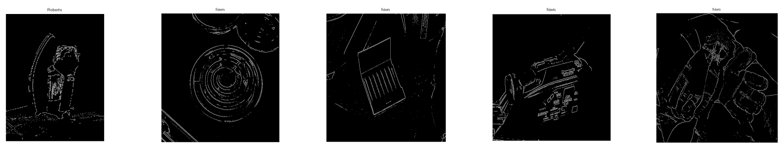
Fig. 12. Edge Images Using Canny Edge Detector



(a) 101.pgm (b) 103.pgm (c) 104.pgm (d) 106.pgm (e) 108.pgm

(f) 109.pgm (g) 110.pgm (h) 125.pgm (i) 126.pgm (j) 131.pgm

Fig. 13. Edge Images Using Laplacian of Gaussian(LOG) Edge Detector


(a) 101.pgm (b) 103.pgm (c) 104.pgm (d) 106.pgm (e) 108.pgm

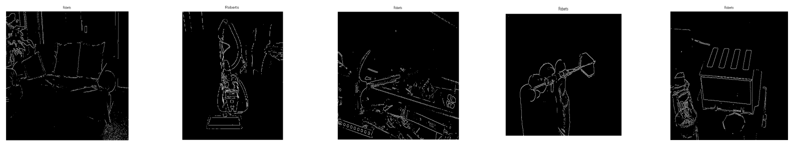
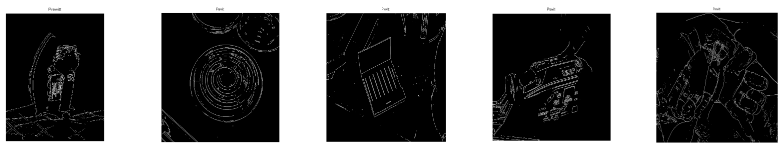

(f) 109.pgm (g) 110.pgm (h) 125.pgm (i) 126.pgm (j) 131.pgm

Fig. 14. Edge Images Using Robert Edge Detector


(a) 101.pgm (b) 103.pgm (c) 104.pgm (d) 106.pgm (e) 108.pgm

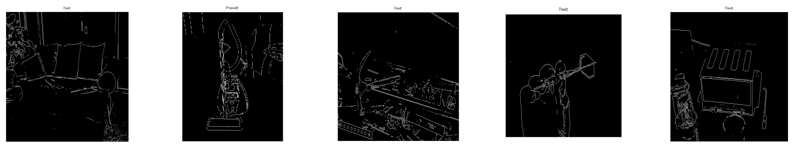
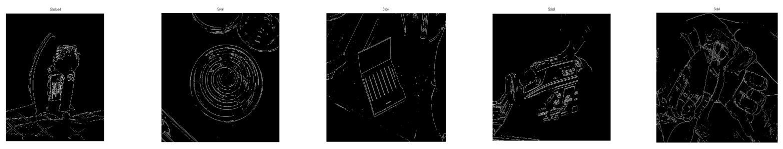

(f) 109.pgm (g) 110.pgm (h) 125.pgm (i) 126.pgm (j) 131.pgm

Fig. 15. Edge Images Using Prewitt Edge Detector


(a) 101.pgm (b) 103.pgm (c) 104.pgm (d) 106.pgm (e) 108.pgm

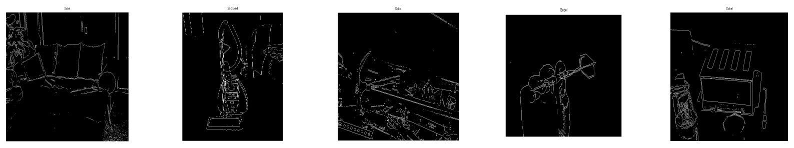
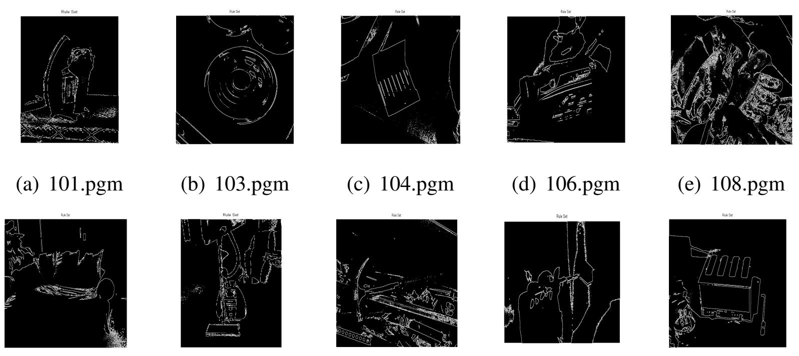

(f) 109.pgm (g) 110.pgm (h) 125.pgm (i) 126.pgm (j) 131.pgm

Fig. 16. Edge Images Using Sobel Edge Detector



(a) 101.pgm (b) 103.pgm (c) 104.pgm (d) 106.pgm (e) 108.pgm

(f) 109.pgm (g) 110.pgm (h) 125.pgm (i) 126.pgm (j) 131.pgm

Fig. 17. Edge Images Using Rule Set

Figure 12, 13, 14, 15 and 16 show the results of Canny, Laplacian of Gaussian(LOG), Robert, Prewitt and Sobel edge detectors respectively. Figure 17 shows the results of rule set which is obtained by the training of cellular automata. As it is clear from the results, shown in the figures 14, 15 and 16 that the Roberts, Sobel and Prewitt edge detectors are less powerful in terms of detecting the edges. Whereas, figures 12, 13 and 17 show that Canny, LOG edge detectors and Rule Set are more powerful and are capable to show more edges. So the results of LoG, Canny and Rule Set are compared quantitatively in terms of misclassification error. The misclassification error is shown in table 1 and graph in figure 18

Table 1. Table showing the misclassification error with ground truth edge images for the edge detectors

Ground Truth Images	**LoG Operator**	**Canny Edge Detector**	**Rule Set**
101.pgm	0.5301	0.534779	0.533581
103.pgm	0.523633	0.5271	0.505559
104.pgm	0.281957	0.304769	0.262361
106.pgm	0.327524	0.341254	0.309014
108.pgm	0.244441	0.266889	0.300519
109.pgm	0.295612	0.298177	0.28176
110.pgm	0.247319	0.269671	0.246837
125.pgm	0.338195	0.346726	0.33845
126.pgm	0.276484	0.296864	0.285505
131.pgm	0.534582	0.539122	0.524503

It is clear from table 1 and the graph in figure 18 that the misclassification error for the rule set is lesser in most of the cases. It clearly indicates that the performance of the rule set is better than Canny and LoG operator(also from others such as Roberts, Prewitt and Sobel). Also, visual inspection of resulting images shows that the quality of edges detected from the rule set is much better than others as it shows thick and clear edges. Also the rule set is capable of showing the weak edges.

In table 1, which shows the misclassification errors obtained, for image 108.pgm the error using rule set is more than Canny and LoG operators. Considering this as a

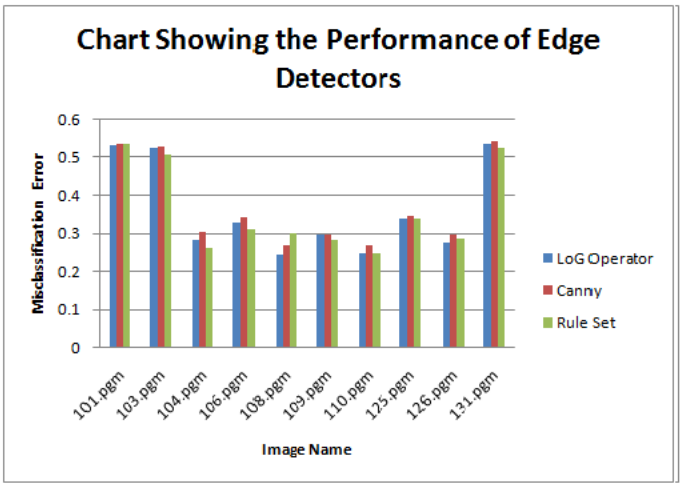
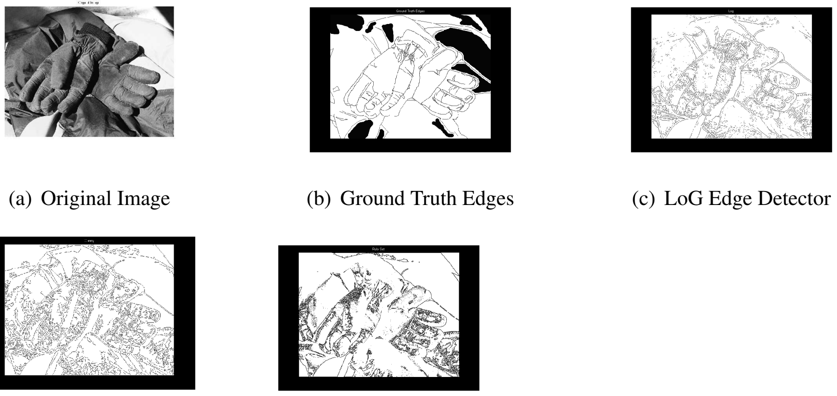


Fig. 18. Misclassification Errors for Various Images.

special case some discussion is required.



(a) Original Image (b) Ground Truth Edges (c) LoG Edge Detector

(d) Canny Edge Detector (e) Rule Set Edge Detector

Fig. 19. Results of Edge Detectors for 108.pgm

The results of this image are shown in figure 19 and the justification of the result is as follows: The ground truth edge images have limitations, as these images are generated manually, only strong edges are shown. By the visual inspection of the results shown in figure 19(e) it is clear that the misclassification error is because of the weak edges obtained by the rule set edge detector while they are missing from the manually generated ground truth image. Though the Canny operator also shows the weak edges as shown in figure 19(d), but it is the property of the the Canny operator that it shows only those weak edges which are connected with the strong edges whereas the rule set along with the above mentioned edges, also shows the weak edges that are not connected with the strong edges. This is the reason of getting more misclassification error in the case of edges obtained by the cellular automata rule set.

7 Conclusions

In this paper the training of cellular automata for edge detection has been performed. The results are encouraging. It has been observed that it is possible to learn good rule sets to perform some complicated image processing tasks such as edge detection. In the case of edge detection the learned rule set is extremely simple as it contains only a single rule with the performance and efficiency up to the mark. The result of the learned rule set has also been compared with some standard edge detectors and it has been found that in most of the cases its performance is better than the standard edge detectors, specially in corner detection. It is important to mention that the aim of cellular automata is to provide simplicity to perform the complex tasks, which is clearly depicted in this paper. As the detection of edges depends on the sensitivity of Otsu's algorithm, in future some other algorithms may be used in place of Otsu's algorithm in which there is a provision of sensitivity control which can control the generation of weak edges as per requirements.

References

[1] Andre, D., Bennett III, F. H., and Koza, J. R. Discovery by genetic programming of a cellular automata rule that is better than any known rule for the majority classification problem. In *Proceedings of the First Annual Conference on Genetic Programming* (1996), MIT Press, pp. 3–11.

[2] Bowyer, K., Kranenburg, C., and Dougherty, S. Edge detector evaluation using empirical roc curves. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.* (1999), vol. 1, IEEE.

[3] Canny, J. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 6 (1986), 679–698.

[4] Cover, T. M., and Van Campenhout, J. M. On the possible orderings in the measurement selection problem. *Systems, Man and Cybernetics, IEEE Transactions on 7*, 9 (1977), 657–661.

[5] Goldberg, D. E., and Deb, K. A comparative analysis of selection schemes used in genetic algorithms. *Foundations of genetic algorithms 1* (1991), 69–93.

[6] Guo, L., and Meng, Y. What is wrong and right with mse. In *Eighth IASTED International Conference on Signal and Image Processing* (2006), pp. 212–215.

[7] Halbach, M., and Hoffmann, R. Implementing cellular automata in fpga logic. In *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International* (2004), IEEE, p. 258.

[8] Hao, H., Liu, C.-L., and Sako, H. Comparison of genetic algorithm and sequential search methods for classifier subset selection. In *ICDAR* (2003), Citeseer, pp. 765–769.

[9] Jain, A., and Zongker, D. Feature selection: Evaluation, application, and small sample performance. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 19*, 2 (1997), 153–158.

[10] JIMÉNEZ MORALES, F., CRUTCHFIELD, J. P., AND MITCHELL, M. Evolving two-dimensional cellular automata to perform density classification: A report on work in progress. *Parallel Computing 27*, 5 (2001), 571–585.

[11] JUILLE, H., AND POLLACK, J. B. Coevolving the" ideal" trainer: Application to the discovery of cellular automata rules. In *University of Wisconsin* (1998), Citeseer.

[12] MITCHELL, M., CRUTCHFIELD, J. P., AND HRABER, P. T. Evolving cellular automata to perform computations: Mechanisms and impediments. *Physica D: Nonlinear Phenomena 75*, 1 (1994), 361–391.

[13] OTSU, N. A threshold selection method from gray-level histograms. *Automatica 11*, 285-296 (1975), 23–27.

[14] PRESTON, K., AND DUFF, M. J. *Modern cellular automata: theory and applications*, vol. 198. Plenum Press New York, 1984.

[15] PUDIL, P., NOVOVIČOVÁ, J., AND KITTLER, J. Floating search methods in feature selection. *Pattern recognition letters 15*, 11 (1994), 1119–1125.

[16] ROBERTS, F., AND TESMAN, B. *Applied combinatorics*. CRC Press, 2011.

[17] ROSIN, P., ADAMATZKY, A., AND SUN, X. *Cellular Automata in Image Processing and Geometry*, vol. 10. Springer, 2014.

[18] ROSIN, P. L. Training cellular automata for image processing. *Image Processing, IEEE Transactions on 15*, 7 (2006), 2076–2087.

[19] ROSIN, P. L. Image processing using 3-state cellular automata. *Computer vision and image understanding 114*, 7 (2010), 790–802.

[20] SAHOTA, P., DAEMI, M., AND ELLIMAN, D. Training genetically evolving cellular automata for image processing. In *Speech, Image Processing and Neural Networks, 1994. Proceedings, ISSIPNN'94., 1994 International Symposium on* (1994), IEEE, pp. 753–756.

[21] SHUKLA, A. P., AND AGARWAL, S. Training cellular automata for salt and pepper noise filtering. In *Computational Intelligence on Power, Energy and Controls with their impact on Humanity (CIPECH), 2014 Innovative Applications of* (2014), IEEE, pp. 519–524.

[22] SHUKLA, A. P., AND AGARWAL, S. Training two dimensional cellular automata for some morphological operations. In *Computational Intelligence on Power, Energy and Controls with their impact on Humanity (CIPECH), 2014 Innovative Applications of* (2014), IEEE, pp. 121–126.

[23] SHUKLA, A. P., AND AGARWAL, S. Selection of optimum rule set of two dimensional cellular automata for some morphological operations.

[24] SHUKLA, A. P., CHAUHAN, S., AND AGARWAL, S. Training of cellular automata for image filtering. In *Proc. Second International Conference on Advances in Computer Science and Application-CSA 2013* (2013), pp. 86–95.

[25] SHUKLA, A. P., CHAUHAN, S., AGARWAL, S., AND GARG, H. Training cellular automata for image thinning and thickening. In *Confluence 2013: The Next Generation Information Technology Summit (4th International Conference)* (2013), IET, pp. 394–400.

[26] SLATNIA, S., AND KAZAR, O. Images segmentation based contour using evca approach, evolutionary cellular automata. *Courrier du Savoir*, 09 (2009).

[27] SOMOL, P., PUDIL, P., AND KITTLER, J. Fast branch & bound algorithms for optimal feature selection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 26*, 7 (2004), 900–912.

[28] TOFFOLI, T., AND MARGOLUS, N. *Cellular automata machines: a new environment for modeling*. MIT press, 1987.

[29] WANG, Z., BOVIK, A. C., SHEIKH, H. R., AND SIMONCELLI, E. P. Image quality assessment: from error visibility to structural similarity. *Image Processing, IEEE Transactions on 13*, 4 (2004), 600–612.

[30] YASNOFF, W. A., MUI, J. K., AND BACUS, J. W. Error measures for scene segmentation. *Pattern Recognition 9*, 4 (1977), 217–231.