Cellular Automata Based Genetic Algorithm for Edge Detection in Binary Images

Rishabh Singh Thakur
Dept Electrical and Computer Engineering
Concordia University

Abstract—Edge detection is important in many computer vision applications. Traditional methods such as canny edge detection etc are limited in nature as they are hand crafted. In this paper, a flexible and powerful approach for detecting edges with cellular automata using genetic algorithms is explored. The implementation Is covered along with an evaluation on a few test images. Quality metric PSNR and edge detection performance metric MS-SSIM (Multi Scale Structural Similarity Index) are used to test the results obtained. A subjective assessment is also made to discuss the visual aspects of the test images.

Index Terms— Cellular Automata, Edge Detection, Genetic Algorithm

1 Introduction

The advances in information technology has led to what is now known as the digital age. This is an era where practically most of the people around the world posses some sort of a gadget which can capture images. Images convey a lot of information as they aid a human being's visual system in capturing information quickly. This information can literally be anything imaginable as they are synthesized by digital devices which support huge amount of information embedding in a very tiny area. It can be considered a super power to have one of these devices. But, coming back to reality, the information that is available in these images are only accessible to human beings. A computing device on the other hand has no clue as to what an image is representing. For this device it is simply a huge meaningless permutation of 0s and 1s that too in 1D. Fortunately, the information that is represented is graphical which implies it has a defined boundary. The process of extracting these prominent boundaries in an image is called as edge detection. This unlocks a whole new world of possibilities allowing computing devices to make sophisticated decisions based on the information conveyed by these edges in areas such are object detection, tracking etc.

But there is a problem. It is not a trivial task to extract these edges as most real world scenarios have very complicated edge patterns which cannot be hard coded for the machine to execute. Of course, this was the case when the computers were at their infancy. Now, we have some remarkable methods to perform edge detection such as sobel filters, canny edge detection. And there are the modern approaches involving deep convolutional neural networks.

But there is another problem here. The approach involving hand crafted filters such as sobel are limited in the sense they apply only for one single task viz edge detection and the edge detection performance is fixed i.e it cannot be tweaked to adjust the edge detection performance for a wide range of images. These problems were addressed by the modern deep convolutional neural networks which are trained on an extremely large dataset on a high end computing platform to learn such filters automatically for a given application. But, the demand of a large dataset and high end computing resource turned to be one of the biggest draw backs of these networks.

There is another way of achieving the above mentioned task of edge detection by combining the pros of the two popular methods discussed which I would like to discuss in this paper. The concept of two dimensional cellular automata combined with a genetic algorithm provides us the best of both worlds. Therefore, my areas of investigation will be the following

- The performance of a cellular automata based genetic algorithm for edge detection on binary

images.
- Different genetic algorithm strategies to boost efficiency.
- Testing the algorithm on "unseen" images.

Before proceeding further with the task it is important to understand what a cellular automata and genetic algorithm is which will be discussed in the following sections.

2 Cellular Automata

Cellular automata are rule based systems which change the global behavior of a system based on the local behavior. Let's say we have a two dimensional grid as shown in fig 1. The numbers indicate cells. Each cell can have a state which is defined within a range. For example, a cell can be either in a 0 or 1 state for an integer range 0-1. Each cell has a neighbourhood which can be of two kinds as shown in fig 2. Now, if we consider the neighbourhood in fig 2 (a) and a binary state for cells, then there are a total of 2^9 = 512 combinations possible in the 3x3 region including the cell and its neighbourhood. If we assign a rule of 0 or 1 to each of these combinations and change the value of a cell according to the rule defined by its 3x3 region we get cellular automata.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

Fig 1 Grid of cells [6]

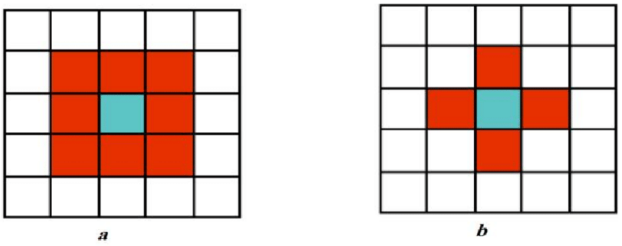

Fig 2 (a) Moore neighbourhood (b) von Neumann neighbourhood [7]

3 Genetic Algorithm

A genetic algorithm is inspired from nature's principles of variation and selection. Let's say we have a single variable function optimization problem for the function f(x) = sin(x) in the range 0 to 2pi [1]. Let's select 10 floating point numbers at random from the given range as possible solutions. This is called solution representation. Each solution is called a chromosome and each chromosome can have genes (in our case we have only one gene viz x). Our task is now to "evolve" these chromosomes called as population so that at least one of them gives maxima for the function f(x). The whole process is shown graphically in fig 3 and the pseudo code is given in fig 4. The population fitness is evaluated initially based on a fitness function which in our case is f(x). Next, we select a set of individuals from the population for mating which can be based on fitness directly or indirectly depending on the selection strategy employed. This process is called as parent selection. Next, we mutate and cross the individuals using various strategies discussed later to obtain the offsprings which in our case are again a set of floating point numbers. Finally, we select the best individuals based on fitness directly or indirectly. This step is called survivor selection.

[1] Of course, finding the maxima is trivial in this case but the procedure being explained applies to general scenarios

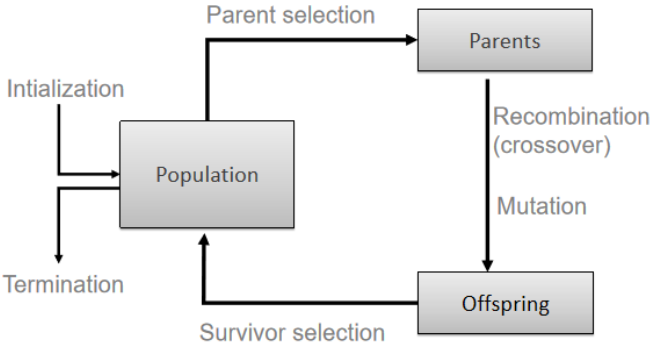

Fig 3 Genetic algorithm general process

```
Initialize of population of rule tables with random rules

While (CurrentGenerationNo<= MaxNoOfGenerations)
Do
    Evaluate fitness of the population

    Select parents for mating from the population

    Perform crossover by selecting two random parents
    from mating pool without replacement.

    Perform mutation on the crossed population

    Evaluate fitness of the offspring population

    Select survivors from the parent and offspring pool
    based on deterministic crowding

    Replace the entire population with the survivors to
    be passed on to the next generation.
End
```

Fig 4 Psuedo code of a typical genetic algorithm

The whole process is repeated from the population fitness evaluation step for a number of iterations which are called as generations until we meet the desired criteria (max fitness, max no of generations etc). This process is the general working of a genetic algorithm. It can be noted that there are two extremas for f(x) which implies two best solutions.

In general scenarios where f(x) is unknown it is very interesting to explore the search space to find various extremas. In those cases, it is important to maintain the diversity of the population to prevent a local extrema convergence.

4 Tackling Edge Detection

Now that we have learnt about cellular automata and genetic algorithm let's use them to tackle our edge detection problem. An 8-bit grayscale image consists of pixel values ranging from 0 to 255. This input image which we will refer to as start image from now on is converted to binary easily using

$$b(x,y) = \begin{cases} 0 & for\ I(x,y) < 128 \\ 1 & for\ I(x,y) \geq 128 \end{cases}$$

Where b(x,y) is the pixel at location (x,y) in the binary image and I(x,y) is the pixel at location (x,y) in the grayscale image. We take each pixel as a cell. The population of the genetic algorithm here is n rule tables each with 512 rows and with each row having either a 0 or 1 as the rule which is randomly initialized for each row. Each rule is defined as follows

$$b(x_c,y_c) = \begin{cases} 0 & for\ rule = 0 \\ 1 & for\ rule = 1 \end{cases}$$

Where (x_c, y_c) are center coordinates of the 3x3 region. This 3x3 region processing similar to that of filters used in edge detection with the difference being the rule dependent processing instead of convolution. The edges of image are handled by wrapping around i.e crossing the right edge would result wrapping the window from the right to left and crossing bottom edge would result in wrapping the window from the bottom to top. In this way the entire image is processed and one complete iteration of this process is known as a pass.

The population of rule tables evolves until the best rule table in the population is capable of performing edge detection fulfilling certain performance criteria on a given binary image. The process from here is explained in detail in the following sub sections

4.1 Representation

A rule table in the population consists of rules defined for each of the 512 rows. Each rule can be a 0 or 1. Therefore, a binary representation of chromosome containing 512 genes with each gene being a 0 or 1 is an appropriate representation of a solution in the population. In other words, a 512 length bit string represents a solution for the given task. An example of 8-bit string is 01011101.

4.2 Fitness Evaluation

For fitness evaluation canny edge detection is performed on the gray scale image to obtain a goal image. A rule table is evaluated on the start image to obtain final image. Rule table is evaluated by taking a 3x3 sliding window on the binary image and performing the rule operations described previously. The hamming distance is calculated between the final image and the goal image to obtain the fitness of the solution/individual. Our goal is to minimize this fitness function.

4.3 Parent Selection Strategy

This step decides which individuals of the population must be selected for mating to generate offsprings. This selection process can either be random or fitness based.

A random strategy based on uniform distribution might help in producing a diverse offspring population but it could also hamper the progress of evolution because of lot of weak individuals created. This strategy is given as

$$P_{uniform}(i) = \frac{1}{\mu}$$

Where μ is the population size and P is the probability of selecting i-th individual.

A fitness based strategy on the other hand can help in boosting the progress of evolution but it could also lead to premature convergence to local minima if the fittest individual dominates the selection process. For this particular reason we explore two methods. The first one is called fitness proportionate selection which implies the probability of selecting an individual for mating is directly proportional to its fitness given as

$$P_{FPS}(i) = \frac{f_i}{\sum_{j=1}^{\mu} f_j}$$

Where f_i is the fitness of i-th individual. The second one is called exponential ranking which is given as

$$P_{exp-rank}(i) = \frac{1 - e^{-i}}{c}$$

Where i is the rank of an individual and c is a constant to ensure total probability is 1. A higher fitness individual is given a larger rank value. Notice that the probability is proportional to the rank and not the absolute fitness value. This prevents the probability values from skewing towards a larger fitness value.

The number of offsprings between two parents can vary. Two offsprings are produced per two parents for simplicity.

4.4 Crossover Strategy

The mating population has to be crossed in some way to get the offspring population. Chromosomes in living beings cross their genes with others at certain points along the length to produce hybrid characteristics in offsprings.

Similarly, we can get a hybrid solution from two parents by exchanging the rules between the corresponding rows in the rule tables. The point(s) of crossing is what we are supposed to determine. Even though A one-point cross exchanges significant amount of genes, it is limited as it has a drawback known as positional bias which keeps together genes that are near each other. Instead, an n-point cross provides much flexibility in exchanging the genes thereby increasing the probability of producing a better solution. The n-point strategy involves selecting n points along the chromosome of two parents and exchanging alternative sections between them. An example is shown in fig 5. Still there exists some positional bias in this strategy. Therefore, another strategy called uniform crossover can be explored. This strategy takes a gene from the first parent and exchanges it with the corresponding gene of the second parent based on some probability. Thus, each position is independent of the others during the crossover. An example is shown in fig 6.

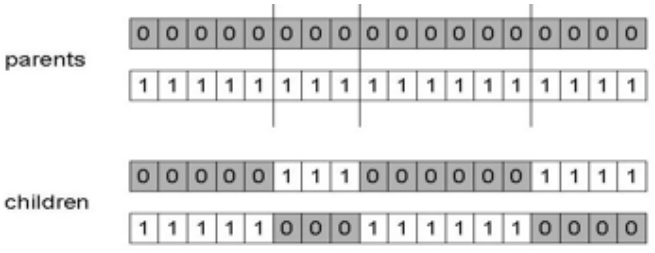

Fig 5 A 3-point crossover

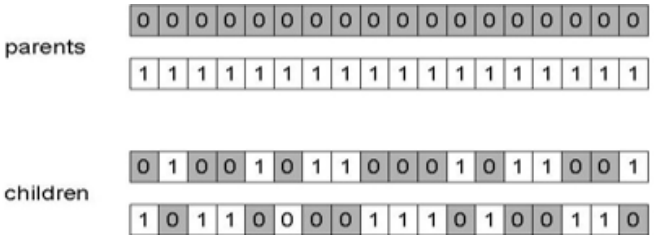

Fig 6 Uniform Crossover

4.5 Mutation Strategy

Mutation is an important step for any genetic algorithm as it introduces new values for the genes. There are only two values which a gene can take viz 0 or 1. A uniform mutation strategy is employed which takes each gene and assigns it a 0 or 1 with equal probability. It is essential to control this mutation process or it can destroy good solutions if done too much and hamper progress. Therefore, a mutation rate p_m is used which is generally in between $1/\mu$ and 1/chromosome_length.

4.6 Survivor Selection Strategy

Now that we have a parent and offspring population, it is time to decide who goes to the next generation. One approach is to select the top μ individuals based on fitness from both populations combined. However, this strategy has a drawback. It skews our search in one direction of the search space leading to what is known as genetic drift. In our case the search space representing the rule tables needed to perform edge detection may not have one global optima. Even if it has one, it may be filled with many local optimas. To avoid these successfully we must preserve the diversity of our population. This can be done by implementing a survivor selection strategy called deterministic crowding. This strategy is given as

- Every pair of parents produces a pair of offsprings.
- The fitness values are calculated for these pairs
- For each pair of parents and offsprings distances are calculated
- Four sets of distances are obtained $d(p1, o1), d(p2, o2), d(p1, o2), d(p2, o1)$.
- The distance metric in our case is the sum of absolute differences of all the corresponding rules in the rule tables divided by the number of rows of the rule table.
- These distances are then used as per the pseudo code shown in fig 7 for the selection process. It is to be remembered that our goal is to minimize the fitness. Looking at the algorithm it is clear that only individuals having similar rule patterns are being compared and the individuals with lower fitness values are passed to the next generation. This ensures preservation of diversity.

5 Measuring Diversity

It has been mentioned in the previous sections about preserving diversity. But, how do we know that it is being preserved. A metric can provide us a way to quantify it so that we can estimate how diverse a generation of population is.

```
If (d(p1, o1)+ d(p2, o2)< d(p1, o2),d(p2, o1))
Do
    If (fp1>fo1)
    Do
        Select o1 for next generation
    Else
    Do
        Select p1 for next generation
    If (fp2>fo2)
    Do
        Select o2 for next generation
    Else
    Do
        Select p2 for next generation
Else
Do
     If (fp1>fo2)
    Do
        Select o2 for next generation
    Else
    Do
        Select p1 for next generation
    If (fp2>fo1)
    Do
        Select o1 for next generation
    Else
```

```
    Do
        Select p2 for next generation
End
```

Fig 7 Deterministic crowding algorithm pseudo code. f represents fitness of individual.

This is estimated as follows:

$$D = \sum_{i=1}^{N} \sqrt{\sum_{j=1}^{i-1} d(g_i, g_j)}$$

Where d(gi, gj) is the sum of absolute differences of all the corresponding rules in the rule tables of gi and gj divided by the number of rows of the rule table. A higher value of D indicates more diversity in the population.

6 Parameter Tuning and control

We have several parameters in the methods described in previous sections. Selecting the right set of values can drastically affect our genetic algorithm performance. Therefore, a set of parameter combinations are tried and the parameters giving minimum fitness is chosen as the desired genetic algorithm.

7 Implementing the Genetic Algorithm

Now that we have all the components of our genetic algorithm defined, we integrate all of them for implementation.

8 Progress of Evolution

In order to estimate whether our genetic algorithm is progressing or not we have to plot the minimum fitness of population for each generation versus the generation number. But, genetic algorithms are probabilistic in nature therefore we have to run our algorithm several times, take the mean of minimum fitnesses of population per generation over the runs and plot it versus generation number. We should plot the standard deviation as well. These plots will show us whether our algorithm is progressing or is stagnated and whether it really works or has a high variance in its performance.

9 Testing the algorithm

A true test of the algorithm is its performance on unseen examples. Therefore, a set of unseen images as shown in fig 11 are taken and the best rule table obtained is applied on them. Fitness is evaluated on the resulting images to measure performance.

10 Results

Each genetic algorithm instance is run 2 times and undergoes only 1 pass. The number of generations is 40. The population and offspring population size is 40 each. Three GA versions each with 3 GA instances are being tested here. These are summarized in table 2.

10.1 Selecting Best GAs

From table 2 it can be seen that GA version 2 instance 2 performs the best hence this is taken for testing and discussed further. The progress of evolution for this GA is shown in fig 9. The input images and results obtained by GA instance 2 are shown in fig 8.. The pseudo code of the selected GA is shown in fig 10.

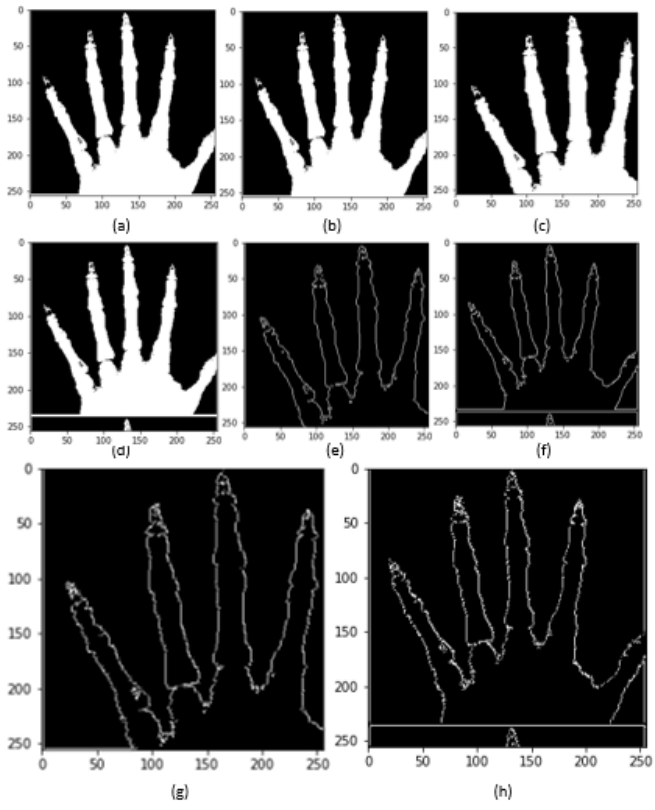

Fig 8 Results for GA version 2 instance 2 (a) initial image for run 1 (b) initial image for run 2 (c) start image for run 1 (d) start image for run 2 (e) canny image for run 1 (f) canny image for run 2 (g) Result/final image for run 1 (h) Result/final image for run 2

10.2 Testing on unseen images

The resulting rule table of GA instance 2 of version 2 is tested on the images shown in fig 11 (a) - (d). The results obtained by canny are shown in fig 11 (e) - (h). The results using GA are shown in fig 11 (i) - (l).

10.3 Objective assessment

The PSNR (peak signal to noise ratio) and MS-SSIM (Multi Scale Structural similarity index) [1] is used for assessing the results obtained. It is calculated between goal_image and final_image. The max value of MS-SSIM is 1 and min is 0. Table 1 shows the results. As we can see from the table the PSNR and MS-SSIM are decreasing as the complexity of images increase. The opposite is observed with the fitness values.

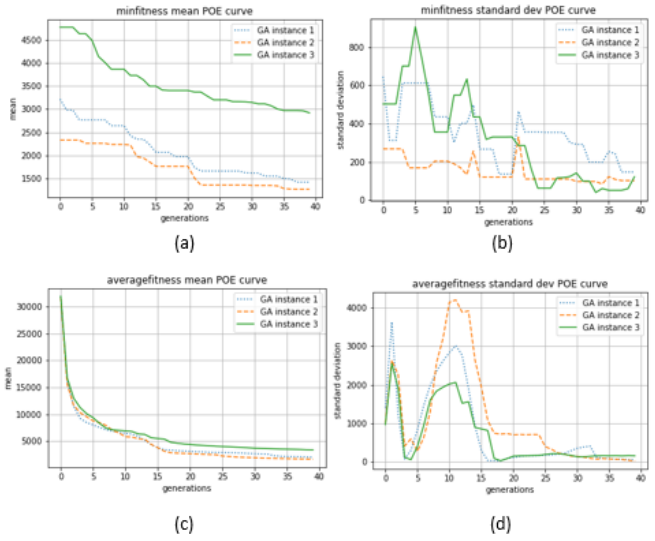

Fig 9 Progress of evolution of GA Version 2. (a) minimum fitness mean curve (b) minimum fitness standard deviation curve (c) average fitness mean curve (d) average fitness standard deviation curve

It is also seen from MS-SSIM that the edge detection performance for the set of images has values above 75%.

Images	PSNR	MS-SSIM	Fitness
Fig 11 (a)	19.507	0.9022	734
Fig 11 (b)	15.927	0.9013	1674
Fig 11 (c)	11.474	0.8484	4667
Fig 11 (d)	9.459	0.7580	7422

Table 1 Objective assessment on unseen test images from fig 11

Initialize population of 40 rule tables with random rules

While (CurrentGenerationNo<= 40)
Do
Evaluate fitness of the population

Select 40 parents for mating from the population using fitness proportionate selection

Perform n-point crossover by selecting two random parents from mating pool without replacement to get two offsprings

Perform uniform mutation on the crossed population

Evaluate fitness of the offspring population

Select survivors from the parent and offspring pool based on deterministic crowding

Replace the entire population with the survivors to be passed on to the next generation.
End

Fig 10 Psuedo code of genetic algorithm implemented

10.3 SUBJECTIVE ASSESSMENT

From the results obtained in fig 11 we can see that fig 2 (i) has horizontal edges missing from the squares/rectangles indicating loss of edges and thus poor performance. However, the images (j), (k) and (l) show that the prominent edges are maintained quite successfully enabling the identification of the image content. Although, they also suffer from loss of some of the fine edges and some random spots.

		GA version 1			GA version 2				GA version 3		
parameters	Instance 1	Instance 2	Instance 3		Instance 1	Instance 2	Instance 3		Instance 1	Instance 2	Instance 3
Generations	40	40	40		40	40	40		40	40	40
Population size	40	40	40		40	40	40		40	40	40
Offspring size	40	40	40		40	40	40		40	40	40
Mutation rate	0.1	0.3	0.5		0.1	0.3	0.5		0.1	0.3	0.5
Crossover rate	None	None	None		0.8	0.8	0.8		0.8	0.8	0.8
Parent selection	None	None	None		Fitness Proportionate	Fitness Proportionate	Fitness Proportionate		Exponential ranking	Exponential ranking	Exponential ranking
survivor selection	Deterministic Crowding	Deterministic Crowding	Deterministic Crowding		Deterministic Crowding	Deterministic Crowding	Deterministic Crowding		Deterministic Crowding	Deterministic Crowding	Deterministic Crowding
Mutation	uniform mutation	uniform mutation	uniform mutation		uniform mutation	uniform mutation	uniform mutation		uniform mutation	uniform mutation	uniform mutation
Crossover	None	None	None		n-point crossover	n-point crossover	n-point crossover		uniform crossover	uniform crossover	uniform crossover
fitness evaluation	Hamming Distance	Hamming Distance	Hamming Distance		Hamming Distance	Hamming Distance	Hamming Distance		Hamming Distance	Hamming Distance	Hamming Distance
Initial diversity	10.069	10.069	10.069		10.69	10.69	10.69		10.69	10.69	10.69
Final diversity	9.95	9.947	9.95		9.32	9.35	9.35		9.93	9.93	9.93
accuracy	1341.5	2058	1957.5		1426	1272	2922.5		1536.5	1453.5	1782.5

Table 2 parameter tuning by running three GA versions each with 3 instances and with each instance having 2 runs. GA version 2 instance 2 was found to be the best among all.

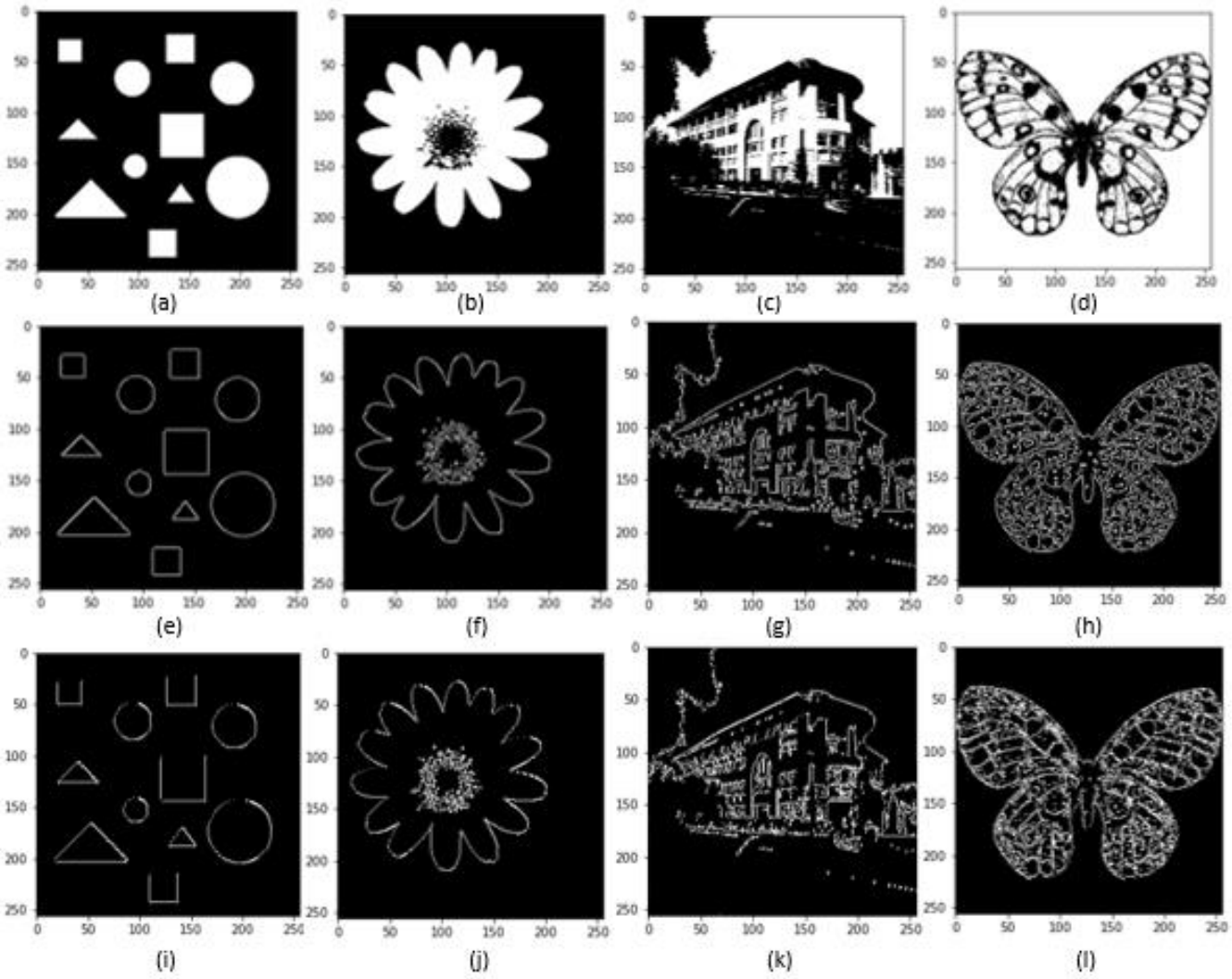

Fig 11 Results on test images using the best GA (a) simple shapes start image(b) flower start image (c) building start image (d) butterfly start image (e) simple shapes goal image (f) flower goal image (g) building goal image (h) butterfly goal image (i) simple shapes result image (j) flower result image (k) building result image (l) butterfly result image

10.4 RULE TABLE

The rule table obtained in the above process is shown in fig 12. This rule table is capable of edge detection.

[{"000000000": 0}, {"000000001": 0}, {"000000010": 0}, {"000000011": 0}, {"000000100": 0}, {"000000101": 0}, {"000000110": 0}, {"000000111": 1}, {"000001000": 1}, {"000001001": 1}, {"000001010": 1}, {"000001011": 1}, {"000001100": 0}, {"000001101": 1}, {"000001110": 0}, {"000001111": 0}, {"000010000": 1}, {"000010001": 0}, {"000010010": 0}, {"000010011": 0}, {"000010100": 0}, {"000010101": 0}, {"000010110": 1}, {"000010111": 0}, {"000011000": 0}, {"000011001": 1}, {"000011010": 0}, {"000011011": 0}, {"000011100": 1}, {"000011101": 0}, {"000011110": 1}, {"000011111": 1}, {"000100000": 1}, {"000100001": 1}, {"000100010": 1}, {"000100011": 0}, {"000100100": 1}, {"000100101": 0}, {"000100110": 0}, {"000100111": 1}, {"000101000": 0}, {"000101001": 0}, {"000101010": 1}, {"000101011": 0}, {"000101100": 1}, {"000101101": 0}, {"000101110": 1}, {"000101111": 1}, {"000110000": 0}, {"000110001": 0}, {"000110010": 1}, {"000110011": 0}, {"000110100": 1}, {"000110101": 0}, {"000110110": 1}, {"000110111": 0}, {"000111000": 0}, {"000111001": 0}, {"000111010": 0}, {"000111011": 1}, {"000111100": 0}, {"000111101": 1}, {"000111110": 1}, {"000111111": 0}, {"001000000": 1}, {"001000001": 0}, {"001000010": 0}, {"001000011": 1}, {"001000100": 0}, {"001000101": 0}, {"001000110": 0}, {"001000111": 0}, {"001001000": 1}, {"001001001": 0}, {"001001010": 0}, {"001001011": 0}, {"001001100": 0}, {"001001101": 1}, {"001001110": 1}, {"001001111": 1}, {"001010000": 0}, {"001010001": 1}, {"001010010": 1}, {"001010011": 1}, {"001010100": 1}, {"001010101": 0}, {"001010110": 0}, {"001010111": 0}, {"001011000": 1}, {"001011001": 1}, {"001011010": 1}, {"001011011": 0}, {"001011100": 0}, {"001011101": 1}, {"001011110": 0}, {"001011111": 0}, {"001100000": 0}, {"001100001": 1}, {"001100010": 1}, {"001100011": 0}, {"001100100": 0}, {"001100101": 0}, {"001100110": 1}, {"001100111": 0}, {"001101000": 1}, {"001101001": 0}, {"001101010": 0}, {"001101011": 1}, {"001101100": 1}, {"001101101": 0}, {"001101110": 0}, {"001101111": 1}, {"001110000": 1}, {"001110001": 0}, {"001110010": 1}, {"001110011": 1}, {"001110100": 1}, {"001110101": 1}, {"001110110": 1}, {"001110111": 1}, {"001111000": 1}, {"001111001": 1}, {"001111010": 0}, {"001111011": 0}, {"001111100": 1}, {"001111101": 1}, {"001111110": 1}, {"001111111": 0}, {"010000000": 0}, {"010000001": 1}, {"010000010": 0}, {"010000011": 0},

{"010000100": 1}, {"010000101": 1}, {"010000110": 0}, {"010000111": 0},
{"010001000": 1}, {"010001001": 0}, {"010001010": 1}, {"010001011": 0},
{"010001100": 0}, {"010001101": 1}, {"010001110": 1}, {"010001111": 1},
{"010010000": 1}, {"010010001": 0}, {"010010010": 1}, {"010010011": 1},
{"010010100": 0}, {"010010101": 0}, {"010010110": 1}, {"010010111": 0},
{"010011000": 0}, {"010011001": 1}, {"010011010": 1}, {"010011011": 0},
{"010011100": 1}, {"010011101": 1}, {"010011110": 0}, {"010011111": 0},
{"010100000": 1}, {"010100001": 0}, {"010100010": 0}, {"010100011": 0},
{"010100100": 1}, {"010100101": 0}, {"010100110": 1}, {"010100111": 0},
{"010101000": 1}, {"010101001": 0}, {"010101010": 1}, {"010101011": 1},
{"010101100": 0}, {"010101101": 0}, {"010101110": 0}, {"010101111": 0},
{"010110000": 0}, {"010110001": 0}, {"010110010": 0}, {"010110011": 0},
{"010110100": 1}, {"010110101": 1}, {"010110110": 1}, {"010110111": 1},
{"010111000": 0}, {"010111001": 0}, {"010111010": 1}, {"010111011": 0},
{"010111100": 0}, {"010111101": 0}, {"010111110": 0}, {"010111111": 1},
{"011000000": 0}, {"011000001": 0}, {"011000010": 1}, {"011000011": 0},
{"011000100": 1}, {"011000101": 1}, {"011000110": 0}, {"011000111": 1},
{"011001000": 1}, {"011001001": 1}, {"011001010": 1}, {"011001011": 0},
{"011001100": 1}, {"011001101": 0}, {"011001110": 1}, {"011001111": 0},
{"011010000": 1}, {"011010001": 0}, {"011010010": 1}, {"011010011": 1},
{"011010100": 0}, {"011010101": 1}, {"011010110": 0}, {"011010111": 0},
{"011011000": 1}, {"011011001": 1}, {"011011010": 1}, {"011011011": 0},
{"011011100": 1}, {"011011101": 1}, {"011011110": 0}, {"011011111": 0},
{"011100000": 1}, {"011100001": 0}, {"011100010": 1}, {"011100011": 0},
{"011100100": 0}, {"011100101": 0}, {"011100110": 1}, {"011100111": 1},
{"011101000": 1}, {"011101001": 0}, {"011101010": 0}, {"011101011": 1},
{"011101100": 0}, {"011101101": 0}, {"011101110": 0}, {"011101111": 1},
{"011110000": 0}, {"011110001": 0}, {"011110010": 1}, {"011110011": 0},
{"011110100": 1}, {"011110101": 1}, {"011110110": 0}, {"011110111": 0},
{"011111000": 0}, {"011111001": 1}, {"011111010": 1}, {"011111011": 0},
{"011111100": 1}, {"011111101": 0}, {"011111110": 1}, {"011111111": 0},
{"100000000": 0}, {"100000001": 0}, {"100000010": 1}, {"100000011": 0},
{"100000100": 0}, {"100000101": 1}, {"100000110": 0}, {"100000111": 0},
{"100001000": 1}, {"100001001": 1}, {"100001010": 0}, {"100001011": 1},
{"100001100": 0}, {"100001101": 0}, {"100001110": 1}, {"100001111": 1},
{"100010000": 1}, {"100010001": 1}, {"100010010": 1}, {"100010011": 0},
{"100010100": 0}, {"100010101": 0}, {"100010110": 0}, {"100010111": 0},
{"100011000": 0}, {"100011001": 0}, {"100011010": 0}, {"100011011": 1},
{"100011100": 1}, {"100011101": 0}, {"100011110": 0}, {"100011111": 0},
{"100100000": 1}, {"100100001": 0}, {"100100010": 1}, {"100100011": 0},
{"100100100": 0}, {"100100101": 0}, {"100100110": 1}, {"100100111": 1},
{"100101000": 0}, {"100101001": 0}, {"100101010": 0}, {"100101011": 1},
{"100101100": 1}, {"100101101": 0}, {"100101110": 1}, {"100101111": 0},
{"100110000": 0}, {"100110001": 0}, {"100110010": 0}, {"100110011": 1},
{"100110100": 1}, {"100110101": 0}, {"100110110": 0}, {"100110111": 1},
{"100111000": 1}, {"100111001": 0}, {"100111010": 1}, {"100111011": 1},
{"100111100": 1}, {"100111101": 1}, {"100111110": 0}, {"100111111": 1},
{"101000000": 1}, {"101000001": 0}, {"101000010": 1}, {"101000011": 1},
{"101000100": 0}, {"101000101": 0}, {"101000110": 1}, {"101000111": 0},
{"101001000": 0}, {"101001001": 0}, {"101001010": 0}, {"101001011": 1},
{"101001100": 1}, {"101001101": 1}, {"101001110": 1}, {"101001111": 1},
{"101010000": 0}, {"101010001": 0}, {"101010010": 0}, {"101010011": 0},
{"101010100": 1}, {"101010101": 1}, {"101010110": 1}, {"101010111": 1},
{"101011000": 0}, {"101011001": 0}, {"101011010": 0}, {"101011011": 1},
{"101011100": 0}, {"101011101": 1}, {"101011110": 0}, {"101011111": 1},
{"101100000": 1}, {"101100001": 0}, {"101100010": 0}, {"101100011": 1},
{"101100100": 1}, {"101100101": 0}, {"101100110": 1}, {"101100111": 1},
{"101101000": 0}, {"101101001": 0}, {"101101010": 0}, {"101101011": 1},
{"101101100": 0}, {"101101101": 0}, {"101101110": 1}, {"101101111": 1},
{"101110000": 1}, {"101110001": 1}, {"101110010": 1}, {"101110011": 1},
{"101110100": 0}, {"101110101": 0}, {"101110110": 0}, {"101110111": 0},
{"101111000": 1}, {"101111001": 0}, {"101111010": 1}, {"101111011": 1},
{"101111100": 1}, {"101111101": 0}, {"101111110": 0}, {"101111111": 1},
{"110000000": 1}, {"110000001": 1}, {"110000010": 1}, {"110000011": 1},
{"110000100": 1}, {"110000101": 1}, {"110000110": 0}, {"110000111": 0},
{"110001000": 0}, {"110001001": 0}, {"110001010": 1}, {"110001011": 1},
{"110001100": 0}, {"110001101": 1}, {"110001110": 0}, {"110001111": 0},
{"110010000": 1}, {"110010001": 0}, {"110010010": 1}, {"110010011": 1},
{"110010100": 1}, {"110010101": 0}, {"110010110": 0}, {"110010111": 0},
{"110011000": 1}, {"110011001": 1}, {"110011010": 1}, {"110011011": 1},
{"110011100": 0}, {"110011101": 1}, {"110011110": 1}, {"110011111": 0},
{"110100000": 0}, {"110100001": 0}, {"110100010": 1}, {"110100011": 0},
{"110100100": 1}, {"110100101": 0}, {"110100110": 1}, {"110100111": 0},
{"110101000": 0}, {"110101001": 1}, {"110101010": 0}, {"110101011": 0},
{"110101100": 0}, {"110101101": 0}, {"110101110": 0}, {"110101111": 0},
{"110110000": 0}, {"110110001": 0}, {"110110010": 1}, {"110110011": 1},
{"110110100": 1}, {"110110101": 0}, {"110110110": 1}, {"110110111": 1},
{"110111000": 0}, {"110111001": 0}, {"110111010": 1}, {"110111011": 0},
{"110111100": 0}, {"110111101": 0}, {"110111110": 0}, {"110111111": 0},
{"111000000": 0}, {"111000001": 0}, {"111000010": 1}, {"111000011": 0},
{"111000100": 1}, {"111000101": 1}, {"111000110": 1}, {"111000111": 1},
{"111001000": 1}, {"111001001": 0}, {"111001010": 0}, {"111001011": 1},
{"111001100": 1}, {"111001101": 1}, {"111001110": 0}, {"111001111": 1},
{"111010000": 1}, {"111010001": 1}, {"111010010": 1}, {"111010011": 0},
{"111010100": 0}, {"111010101": 1}, {"111010110": 0}, {"111010111": 0},
{"111011000": 1}, {"111011001": 0}, {"111011010": 1}, {"111011011": 1},
{"111011100": 0}, {"111011101": 0}, {"111011110": 1}, {"111011111": 0},
{"111100000": 1}, {"111100001": 0}, {"111100010": 0}, {"111100011": 1},
{"111100100": 1}, {"111100101": 1}, {"111100110": 0}, {"111100111": 1},
{"111101000": 1}, {"111101001": 0}, {"111101010": 0}, {"111101011": 1},
{"111101100": 0}, {"111101101": 1}, {"111101110": 0}, {"111101111": 0},
{"111110000": 0}, {"111110001": 1}, {"111110010": 1}, {"111110011": 0},
{"111110100": 1}, {"111110101": 1}, {"111110110": 1}, {"111110111": 1},
{"111111000": 1}, {"111111001": 0}, {"111111010": 0}, {"111111011": 0},
{"111111100": 0}, {"111111101": 1}, {"111111110": 0}, {"111111111": 0}]

Fig 12 Rule table for edge detection

12 Limitations

The algorithm discussed can only process binary images. The images converted from grayscale to binary experience a significant loss of information reducing the quality of edges detected in the resulting image. It also suffers from detail loss in few cases as discussed in subjective assessment. A better implementation is required to minimize the loss. It can be seen from the table 2 that the accuracy for all instances is greater than 95% considering the worst fitness to be 65536 for a 256x256 image However, this can be misleading because it is rare to obtain a worst case fitness of 65,536. For this purpose, MS-SSIM was used for assessment. But again this won't be sufficient as it only considers a few aspects of edge detection performance.

13 Conclusion

From the above results it can be concluded that the approach of detecting edges in a binary image with cellular automata using genetic algorithm is successful. The approach also has its shortcomings with respect to the quality

of edges produced. However, considering the fact that the genetic algorithm implemented was limited in its search space exploration it can be expected that a sophisticated version employing variation and selection designed to target the fine edges as well as increase the overall execution speed can provide promising results.

It is also very interesting to note the general nature of this approach. The algorithm is not limited to just detecting edges. In fact, it is capable of learning any arbitrary set of rules provided that the images used in the learning process have clear representation of the required features.

14 Future Work

To obtain better results we can try using a meta GA for parameter tuning. EBIQA (Edge Based Images Quality Assessment) and others in [5] can be used for better interpretation of edge detection results.

15 Related Works

There are many methods proposed for solving edge detection problem using genetic algorithms and cellular automata. Sahota et al. [2] in 1994 used three original/edge image pairs for training. However, they did not mention the rules obtained and testing performed by them was also limited. Batouche et al. [3] did not train all 512 rules for a moore neighbourhood by assigning rotationally symmetric patterns the same rules. They used 15 rule tables per chromosome. They obtained successful results but with a drawback of thick edges. There has been work on edge detection of intensity images as opposed to binary images discussed so far. Kazar and Slatnia [4] developed algorithm for processing intensity images. They used a novel approach of labelling different intensities in the moore neighbourhood of a central pixel and using the labels as state values instead of using pixel values. With this they reduced the possible number of neighbourhood patterns from $256^8/5$ to $8^8/5$ for 256 greyscale images which is computationally efficient while constructing state transition tables.

16 References

[1] Wang, Zhou, Eero P. Simoncelli, and Alan C. Bovik. "Multiscale structural similarity for image quality assessment." The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003. Vol. 2. Ieee, 2003.

[2] Sahota, P., M. F. Daemi, and D. G. Elliman. "Training genetically evolving cellular automata for image processing." Proceedings of ICSIPNN'94. International Conference on Speech, Image Processing and Neural Networks. IEEE, 1994.

[3] Batouche, Mohamed, Souham Meshoul, and Ali Abbassene. "On solving edge detection by emergence." International Conference on Industrial, Engineering and other Applications of Applied Intelligent Systems. Springer, Berlin, Heidelberg, 2006.

[4] Kazar, Okba, and Sihem Slatnia. "Evolutionary Cellular Automata for Image Segmentation and Noise Filtering Using Genetic Algorithms." Journal of Applied Computer Science & Mathematics 11 (2011).

[5] Sadykova, Diana, and Alex Pappachen James. "Quality assessment metrics for edge detection and edge-aware filtering: A tutorial review." 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI). IEEE, 2017.

[6] Grid image https://topdrawer.aamt.edu.au/Patterns/Good-teaching/Concrete-to-abstract/Making-number-sequences-real/A-hundred-square

[7] neighbourhood image https://www.researchgate.net/figure/a-Moore-neighborhood-b-Von-Neumann-neighborhood_fig6_262844415