Umair Ali Khan, Alireza Fasih, Kyandoghere Kyamakya, Jean Chamberlain Chedjou

Transportation Informatics Group, Alpen Adria University, Klagenfurt, Austria.

Benchmarking of Traditional Method of Genetic Algorithm with the Real Coded Method with a modified type of 2-Point Crossover *(F-Crossover)*

Abstract *–A benchmarking using cellular neural networks is performed between the traditional method of Genetic algorithm (using binary population of random chromosomes) with a real coded approach of genetic algorithm. The benchmarking was done with various image processing operations and it is shown that in most of the image processing operations, real coded appraoch converges faster. Real numbers population prevents the repeated encoding and decoding of chromosomes. Also the sizes of chromosomes are relatively smaller. Moreover, a modified type of 2-point crossover (F-Crossover) is introduced which decreases the convergence time of the genetic algorithm and eliminates the need of mutation.*

Keywords*: Benchmarking, Genetic algorithm, Real number population, Chromosomes, Encoding, Decoding.*

1. Introduction

Genetic algorithm is inspired by the natural genetic process. It is effectively used for search and optimization problems to find the exact or approximate solutions. It finds applications in automatic programming, machine learning, economics, ecology and social systems etc. The traditional approach of genetic algorithm works with a binary encoding of the parameter set, (not the parameters themselves) [1]. The initial population of binary chromosomes is produced by using some suitable encoding method [2]. These binary chromosomes have very large sizes that increase with the problem parameters. Using standard IEEE-754 encoding method [3], the size of a chromosome used for templates optimization of size $n \times n$ is 32(n^2+1) (4-byte real number that is equal to 32 bits; 32n^2 bits for templates and 32 bits for Bias). Moreover, these binary coded chromosomes involve repeated encoding and decoding mechanisms as well.

In this paper, we have performed a benchmarking of binary coded genetic algorithm with real coded approach for image processing operations and have shown that the later decreases the sizes of the chromosomes and eliminates repeated encoding and decoding functions. The benchmarking consists of comparing the results of our approach based on real coded population with the resuls from the traditional genetic algorithm using Cellular Neural Networks [4]. Using real numbers, the size of a chromosome used for templates optimization of size $n \times n$ is 2n^2+1, which is much smaller than that of a binary chromosome. The output results of the two methods differ by no means. For most of the image processing tasks, real numbers approach converges faster than the binary numbers approach.

A modified type of 2-Point crossover *(called F-Crossover in this paper)* is also introduced that significantly decreases the convergence time of the genetic algorithm. F-crossover improves the performance of 2-point crossover by performing a circular shift between the two sites and then exchanging the bits / numbers between the sites. This crossover eliminates the need of mutation as well.

This paper discusses a brief introduction of cellular neural network, genetic algorithm and F-crossover. In the last, we present some benchmarking results of binary coded and real coded methods and we compare them with F-crossover. It is shown that the real coded method converges earlier than the binary coded method. Also, it is shown that F-crossover significantly decreases the convergence time of the genetic algorithm. Our results use F-crossover with real coded method without mutation.

2. Cellular Neural Networks

Cellular neural network is a n-dimensional array of mainly identical dynamical systems, called cells in which most interactions are local within a finite radius r [5]. Cellular neural networks take some of the properties of *Cellular Automata* and *Artificial Neural Networks*. Like cellular automata, the cells can have discrete states and communicate directly to their nearest neighbors only. Similarly, like artificial neural networks, the cells are simple processing elements that provide parallel computation and can also have continuous states. The most important factor in cellular neural network is the concept of neighborhood which provides fast parallel computation and makes the VLSI implementation of CNN flexible.

A *standard CNN architecture* consists of an $M \ X \ N$ rectangular array of cells $C(i,j)$ with cartesian coordinates $(i,j), \ i=1,2,......, M, j=1,2,......,N$ (Fig.1) [6].

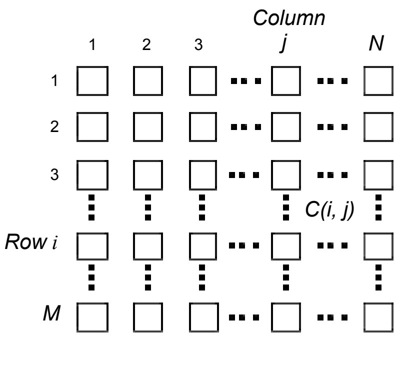


Fig.1 Basic CNN architecture

Cell $C(i,j)$ can interact with neighboring cells defined in a sphere of influence of given radius r. A cell is completely described by the variables *Input, State, Output* and *templates*.

In terms of a computing machine, inputs and intitial states are used to supply the cellular system with the input data(to be processed) while the result or output data is availabe in the form of a pattern (spatial, temporal or spatio-temporal) of the outputs [7].

The behavior of CNN can be described by the follwoing 1st order ODE:

(1) $$\dot{x}_{ij} = -x_{ij} + \sum_{C(k,l)\in N(i,j)} A(i,j;k,l)y_{kl} + \sum_{C(k,l)\in N(i,j)} B(i,j;k,l)u_{kl} + I$$

Where x_{ij} is the state of the cell $C(i,j)$ and u_{kl} is the input. $A(i,j;k,l)$ and $B(i,j;k,l)$ are called *feedback template* and *control template* respectively which represent the strength of synaptic interaction of the cell $C(i,j)$ with all the cells $C(k,l)$ in its neighborhood $N(i,j)$. *I is a bias* or *threshold*. The non-linear output function is modelled by Eq.3.

(2) $$y_{ij} = \frac{1}{2}(|x_{ij}+1| - |x_{ij}-1|)$$

The three parameters; *feedback template (A), control template (B) and Bias (I)* uniquely determine the state of CNN. Under certain conditions, the CNN will always converge to a steady state. The output is then used for evaluation [8].

3. Genetic Algorithm

Genetic algorithm takes the properties of evolution, adaptation and natural selection [9] to solve a problem. It starts with a randomly generated population of candidate solutions to a problem called *chromosomes*. The fitness value of each chromosome is calculated to start the process of evolution which happens in the form of several generations. As the generation process evolves, the quality of chromosomes is improved. The process of natural selection causes chromosomes that encode successful structures to reproduce more often than those that do not [10]. Genetic algorithm solves the problem of finding good chromosomes by manipulating in the chromosomes blindly without any knowledge about the problem it is solving [11].

At first, an initial pool of random chromosomes is created using some encoding method. Traditional genetic algorithm creates binary encoded chromosomes. The initial population can be created using real numbers which don't required any encoding. After that, the fitness of each chromosome is evaluated using a suitable fitness function. The fitness function plays a very important role as it assigns proper fitness scores to each candidate solution and the same scores are used for selecting parent chromosomes to bread new childs by mating and mutation. For image processing operations, we use a fitness function based on the Euclidean distance (Eq. 3). After all the fitness values have been calculated, the list is sorted in descending order.

(3) $$Fitness = 1 - \frac{\sum_{i=1}^{N}\sum_{j=1}^{M}(I(i,j)-T(i,j))^2}{4MN}$$

In Eq.3, I represents an $M\,X\,N$ output image generated by CNN and T represents an $M\ X\ N$ target image. Every chromosome is applied to CNN as its parameters and the output is compared with the target image which is the solution to be reached. The fitness function in Eq.3 calculates the pixel-wise difference between the CNN generated output image and the target image. The denominator $4MN$ is used to normalize the fitness between 0 and 1. Before applying the input image to CNN, it is normalized between 0 and 1. By the same rule, the CNN output is also normalized before calculating fitness.

The process of evolution starts by selecting parents based on their fitness values. We select two random parents from upper half of the population. The selected parents are then breaded according to a fixed probability to produce new childs. This process is called *crossover*. Crossover operator randomly chooses a locus and exchanges the subsequences before and after that locus between two chromosomes to create two offspring [12]. It roughly mimics biological recombination between two single-chromosomes organisms. A 2-point crossover of real coded chromosomes is shown in Fig.2. The two crossing sites C_1 and C_2 are randomly selected and the information between these two sites is exchanged between the two chromosomes resulting in two new childs.

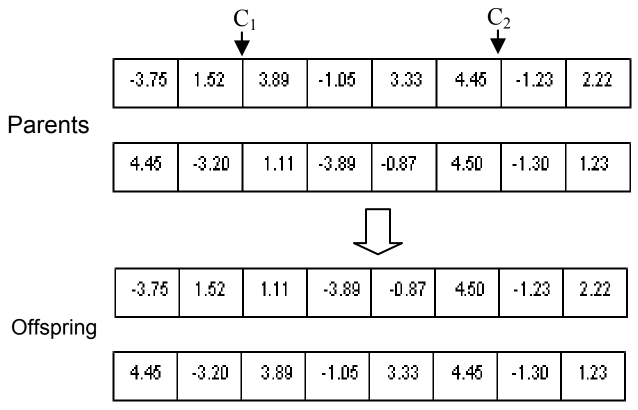


Fig.2 2-Point Crossover in genetic algorithm

After crossver, another genetic algorithm operator *mutation* introduces new information in the offspring. Mutation provides genetic diversity and avoids the algorithm to fall into local minima which is a situation in which all the chromosomes may have equal fitnesses and this stops or slows down the process of evolution. Mutation occurs at a low probability and in binary coded chromosome, it flips a random bit. In real coded population, mutation replaces a random number with another randomly selected number.

The fitness scores of the new childs are evaluated and they are then inserted in the population if their fitness is better than the two worst parents in population. The worst parents are then replaced to make room for the new childs. The population list is sorted in descending order and the process is repeated until the fitness of a chromose is found to be equal to or greater than a specified value or the algorithm completes a specified number of iterations.

4. F-crossover

we propose a modified type of 2-point cross over; the F-crossover that decreases the convergence time of genetic algorithm and eliminates mutation as well. Fig.3 shows F-cossover between two chromosomes.

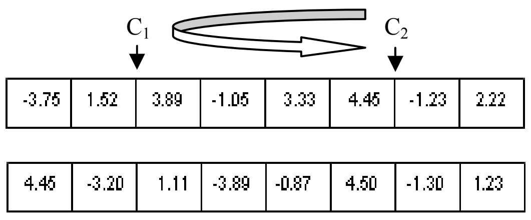


Fig.3(a) Circular Shift between 2 crossing sites

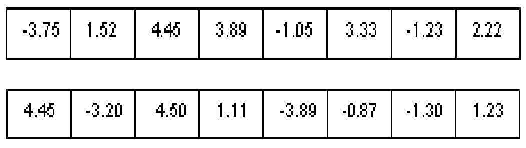


Fig.3(b) Child chromosomes after circular shift

-3.75	1.52	4.50	1.11	-3.89	-0.87	-1.23	2.22

4.45	-3.20	4.45	3.89	-1.05	3.33	-1.30	1.23

Fig.3(c) Exchange of numbers between the crossing sites

Two random crossing sites C_1 and C_2 are selected. A circular shift of the numbers between the sites is performed. We use right circular shift of the numbers, however left circular shift can also be used. The numbers/bits between these sites are then interchanged between the two chromosomes. F-crossver provides a rapid change of information in the chromosomes allowing fast convergence. The rapid change of information in the form of circular shift of the numbers also provides a sort of intrinsic genetic diversity and no more mutation is required.
Fig.4 shows an image processing operation by real coded genetic algorithm with F-crossover using a 3 x 3 CNN. An image having blurred edges is processed by genetic algorithm and a target is specified for sharpening the edges. The process was performed without mutation.

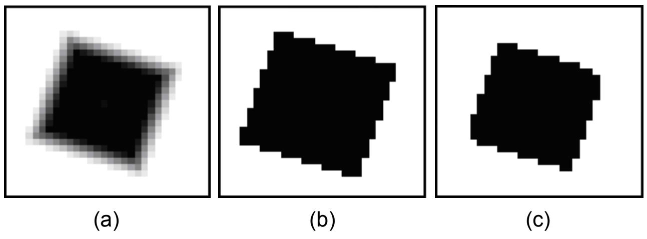


Fig.4 (a) Input Image (b) Target Image (c) Output Image

A population of 100 chromosomes was used in this image processing operation. The intial graph of random errors calculated during evaluating fitnesses of 100 chromosomes is shown in Fig.5.

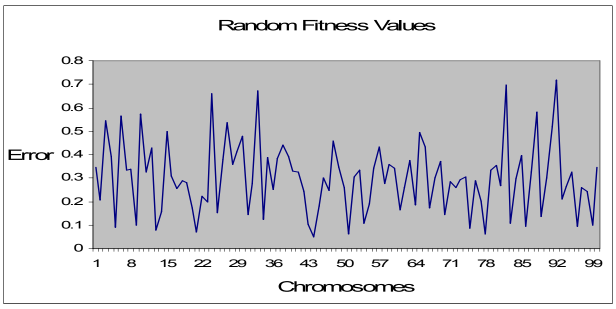
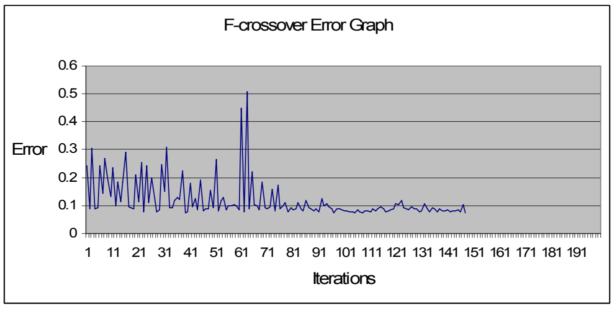


Fig.5 Error graph of 100 chromosomes calculated before evaluation process

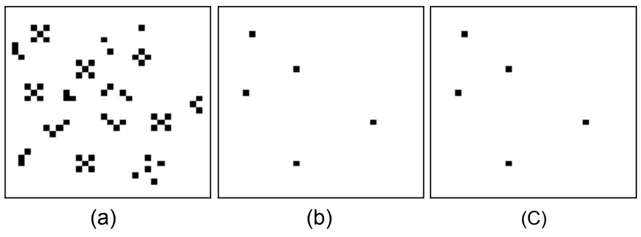


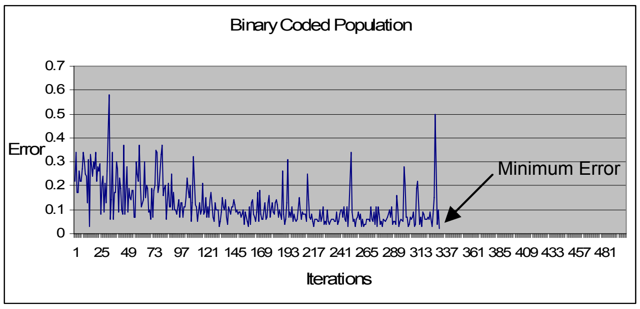
Fir.6 Error graph of evolution process

The graph in Fig.6 shows the error rate of the evolution process. It is shown that the error starts randomly and remains chaotic. However, this error comes down gradually and finally reaches a minimum value. The higher peaks are produced due to the circular shift which sometimes creates childs of bad fitness values.

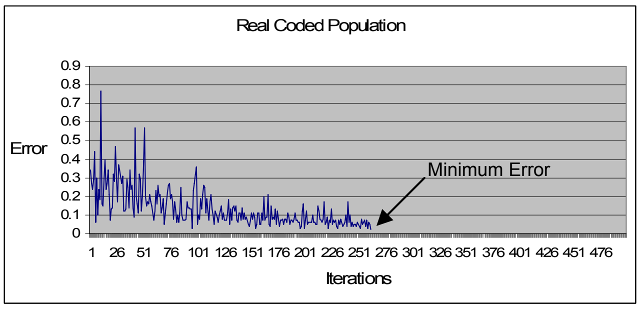
5. Benchmarking

A benchmarking is performed between binary coded genetic algorithm and real coded genetic algorithm using 2-point crossover and it is shown that real coded algorithm converges faster than the binary coded algorithm. The results of the two methods are then compared with F-crossover using real coded approach. Using F-crossover, genetic algorithm converges faster. The F-crossover was used without mutation.

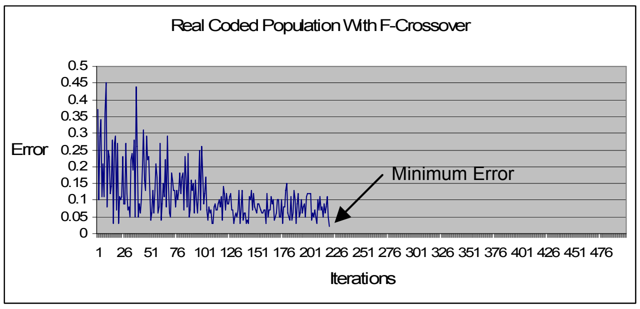




(d)



(e)

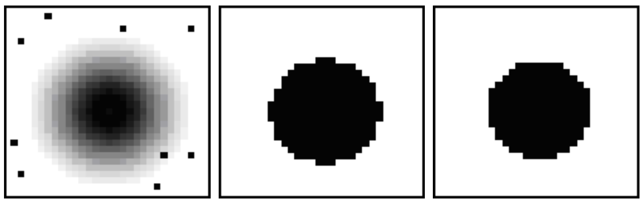


(f)

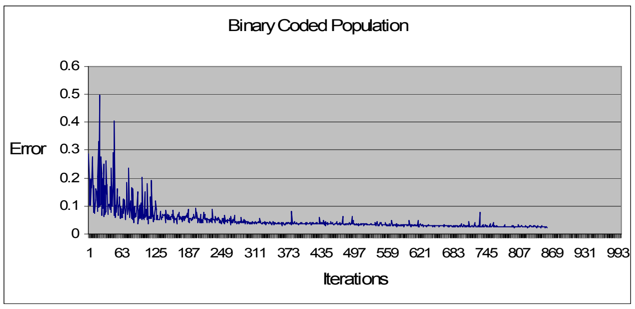
Fig.7 (a), (b), (c) Input, Target and output images respectively; (d), (e), (f) error graphs of binary coding, real coding and F-crossover

As shown in Fig.7(a), an input image was provided to genetic algorithm and the aim was to detect specifc points in the image as specified by the target (Fig.7(c)). A fitness value of 0.99 was specified as a stopping criterion. The binary coded algorithm reached the fitness value in 331 iterations. The real coded algorithm achieved the specified fitness value in 263 iterations, while the real coded algorithm using F-crossover reached the fitness value in just 220 iterations. The graph in Fig.7(f) shows that error in F-crossover is more chaotic than the one in binary coded and real coded algorithms using 2-point crossover. The chaotic error is due to the rapid change of information in chromosomes due to F-crossover.

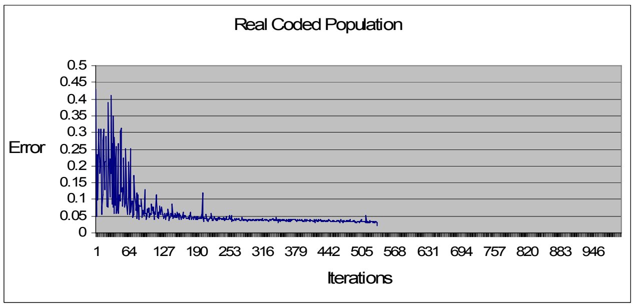
For binary coded method, we used 8-bit encoding scheme. The 8-bits are sufficient to represent values between -5 and +5. This range is suitable for VLSI implementation as well. Out of these 8 bits, MSB is used as sign bit, 3 bits are used for fixed point bits and 4 bits are used for floating point bits.



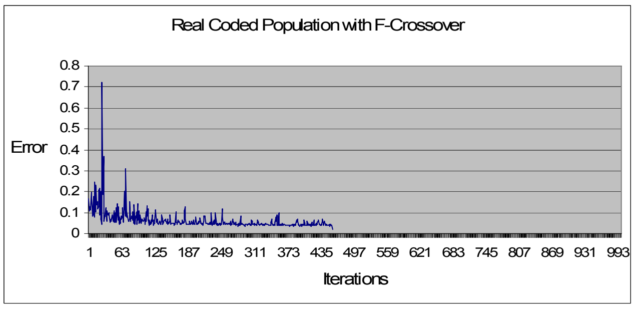
(a) (b) (c)



(d)



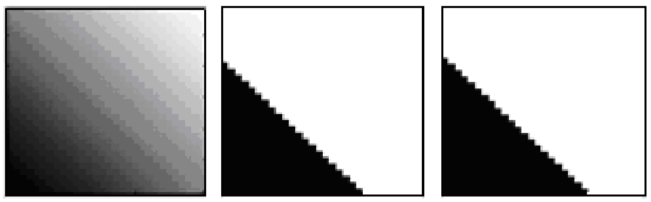
(e)



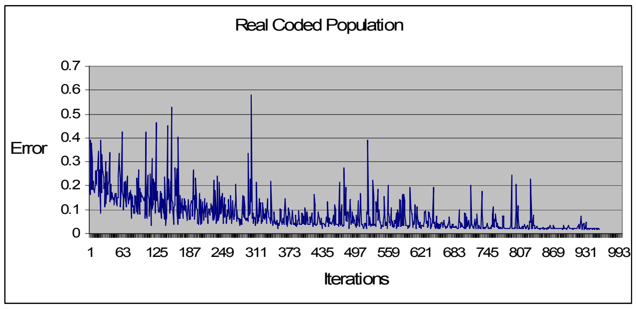
(f)

Fig.8 (a), (b), (c) Input, Target and output images respectively; (d), (e), (f) error graphs of binary coding, real coding and F-crossover

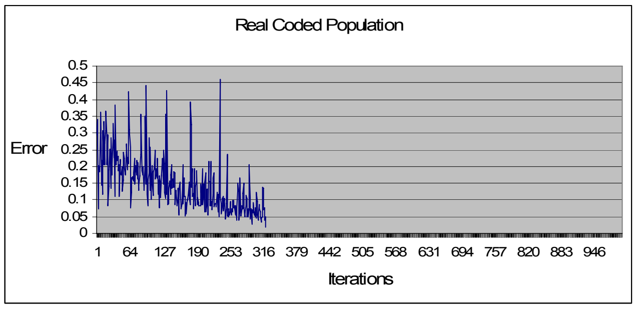
Fig.8 shows a second test performed on the image shown in Fig.8(a). The aim was to remove the noise and thresholding the image as given in target image (Fig.8(b)). Binary coded and real coded algorithm reached the minimum error in 861 and 536 iterations respectively. Whereas, the real coded population with F-cossover converged to the minimum error in 488 iterations.



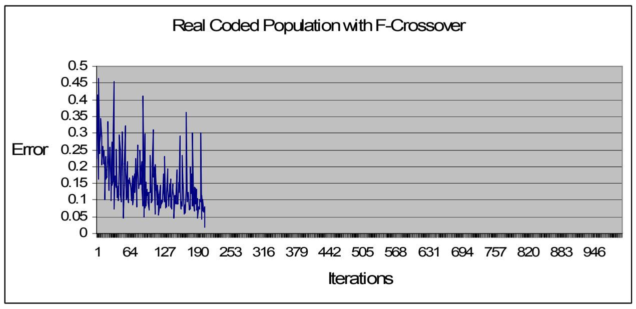
(a) (b) (c)



(d)



(e)



(f)

Fig.9 (a), (b), (c) Input, Target and output images respectively; (d), (e), (f) error graphs of binary coding, real coding and F-crossover

Another interesting task is shwn in Fig.9. The task is thresholding the input image to a specified value as defined by the target image. As shown in graphs in Fig.9(d),(e) & (f), binary coded and real coded algorithms achieved the minimum error in 957 and 323 iterations respectively. The real coded algorithm with F-crossover converged to the minimum error in 207 iterations.

Some other image processing tasks performed with F-crossvoer without using mutation are also presented in Fig.10-13 alongwith the corresponding templates.

The first image processing task consists separating a triangular part from the image (Fig.10(a)) as depicted in Fig.10(b) as the target. The output is shown in Fig.10(c).

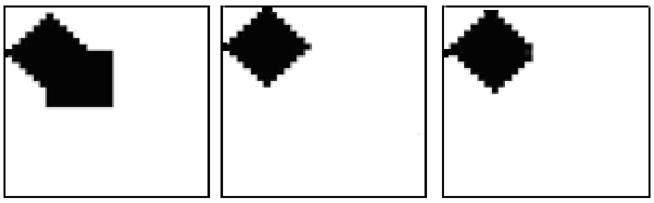

(a) (b) (c)

Fig.10 Removing the rectangle part from the figure

$$A = \begin{pmatrix} 4.6 & 3.7 & 3.2 \\ 3.3 & 4.5 & -0.8 \\ 2.1 & 0.5 & -1.5 \end{pmatrix}, B = \begin{pmatrix} 3.3 & 2.5 & 3.6 \\ -0.6 & 1.3 & 2.4 \\ 4.2 & 0.8 & -2.7 \end{pmatrix}, I = 4.6$$

A similar processing is performed with the same image (Fig.11(a)) where a rectangular part is separated from the image as specified by the target in Fig. 11(b). Fig.11(c) shows the output.

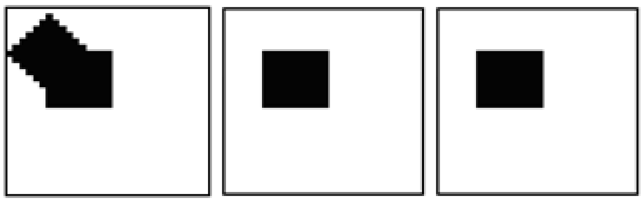

(a) (b) (c)

Fig.11 Extracting rectangular part

$$A = \begin{pmatrix} -4.4 & 0.5 & 5 \\ 2.5 & 5 & 4.5 \\ 3.3 & 4.7 & 4.6 \end{pmatrix}, B = \begin{pmatrix} -1.8 & 1.4 & 0 \\ 2.9 & 2.8 & 3.1 \\ 0.8 & -2.3 & 2 \end{pmatrix}, I = 4.7$$

Fig.12(c) shows the output of another image processing task that involves detecting the west part of the image (Fig.12(a)).

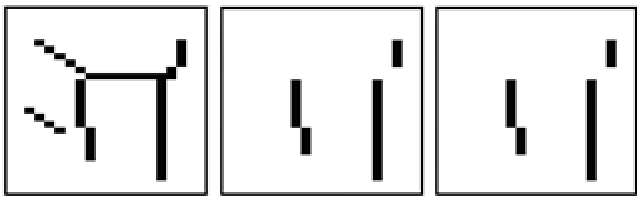

(a) (b) (c)

Fig.12 West Detecting

$$A = \begin{pmatrix} -0.1 & 3 & -1.7 \\ 4.4 & 4.6 & 1.5 \\ 0.8 & 2.4 & 4.6 \end{pmatrix}, B = \begin{pmatrix} -0.4 & 2.9 & -0.2 \\ -1.1 & 4.6 & -0.3 \\ -1.5 & 3.7 & -1.1 \end{pmatrix}, I = -0.3$$

Next image processing task is removing some objects from the image (Fig.13(a)) as specified by the target image (Fig.13(b)). Fig.13(c) shows the output.

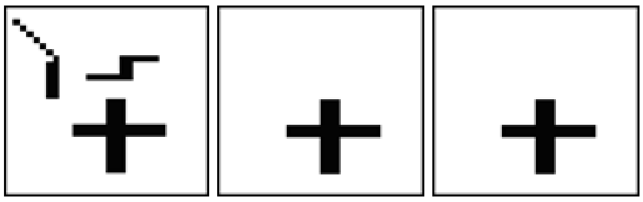

(a) (b) (c)

Fig.13 Object Removing

$$A = \begin{pmatrix} 4.5 & 3 & 1.6 \\ 1.4 & 4.8 & 3.8 \\ 4.5 & 4.9 & 2.5 \end{pmatrix}, B = \begin{pmatrix} 4.3 & 3 & 2.2 \\ 4.6 & 4.9 & 4.3 \\ 4.8 & 1 & 4.1 \end{pmatrix}, I = 4.8$$

In the following image processing operation, an image (Fig.14(a)) is thresholded according to the target image Fig.14(b)). Fig.14(c) shows the output.

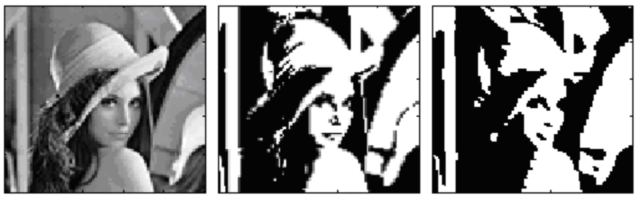

(a) (b) (c)

Fig.14 Thresholding

$$A = \begin{pmatrix} 1.1 & 0.9 & -4.3 \\ 2.4 & 5 & 3.4 \\ -0.8 & 1.6 & 3.8 \end{pmatrix}, B = \begin{pmatrix} -2.1 & 1.8 & 1.8 \\ 3.6 & 1.6 & -1.9 \\ 4 & 0.8 & 1.1 \end{pmatrix}, I = -0.2$$

REFERENCES

[1] Tibor Kozek, Tamas Roska, Leon O.Chua, *"Genetic Algorithm for CNN Template Learning," IEEE Transcations on Circuits and Systems –I: Fundamental Theory and Applications.* Vol. 40, NO. 6, June 1993.

[2] Tibor Kozek, Tamas Roska, Leon O.Chua, *"Genetic Algorithm for CNN Template Learning," IEEE Transcations on Circuits and Systems–I: Fundamental Theory and Applications.* Vol. 40, NO. 6, June 1993.

[3] R. Ubal, et al., *"Floating-Point Representation, Algorithms, and Circuits in Undergraduate Courses," IEEE Transactions on Education*, vol. 49, no. 3, 2006, pp. 321.

[4] L.O. Chua and L. Yang, *"Cellular neural networks: Applications," IEEE Transactions on Circuits and Systems*, vol. 35, no. 10, 1988, pp. 1273-1290.

[5] L.O. Chua and T. Roska, *"The CNN paradigm," IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 40, no. 3, 1993, pp. 147-156

[6] L.O. Chua and T. Roska,*"Cellular neural networks and visual computing: foundation and applications"*, Cambridge University Press, 2002, Chapter 2, pp.7.

[7] R. Dogaru, "*Universality and emergent computation in cellular neural networks*," World Scientific River Edge, NJ, 2003, Chapter 2, pp.10.

[8] M. Zamparelli, *"Genetically trained cellular neural networks,"* vol. 10, no. 6, 1997, pp. 1143-1151.

[9] C.R. Darwin and C. Darwin, *"The origin of species by means of natural selection*," Adamant Media Corporation, 2000, chapter 4, pp. 62.

[10] O.N. Ucan, et al., *"Extraction Of Facial Features Using Genetic Cellular Neural Networks," Journal of Electrical & Electronics*, vol.2, No.2, year 2002

[11] C.T. Lin and C.S.G. Lee,*"Neural fuzzy systems: a neuro-fuzzy synergism to intelligent systems*," Prentice-Hall, Inc. Upper Saddle River, NJ, USA, 1996.

[12] M. Mitchell, "*An introduction to genetic algorithms*," Bradford Books, 1996, Chapter 1, pp.10.

Authors

Umair Ali Khan, Research Assistant, Transportation Informatics Group, Alpen Adria University, Klagenfurt, Austria, Email: ukhan@edu.uni-klu.ac.at; Alireza Fasih, Research Assistant Transportation Informatics Group, Alpen Adria University, Klagenfurt, Austria, Email: alireza.fasih@uni-klu.ac.at ; Prof. Dr.-Ing. Kyandoghere Kyamakya, Head Transportation Informatics Group, Alpen Adria University, Klagenfurt, Austria, Email: kyandoghere.kyamakya@uni-klu.ac.at ; Dr.-Ing. Jean Chamberlain Chedjou, Transportation Informatic Group, Alpen Adria University, Klagenfurt, Austria, Email: jean.chedjou@uni-klu.ac.at