Eye Illusion Enhancement Using Interactive Differential Evolution

Zaid Salami Mohamad, Arman Darvish, Shahryar Rahnamayan
Faculty of Engineering and Applied Science
University of Ontario Institute of Technology (UOIT), Oshawa, Canada
Zaid.Mohamad@uoit.ca, Arman.Darvish@uoit.ca, Shahryar.Rahnamayan@uoit.ca

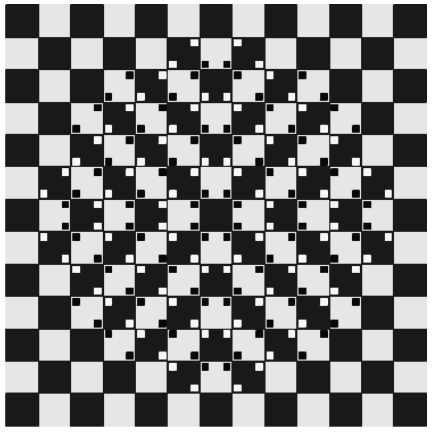
Abstract— **Eye illusion is one of the most interesting topics which attracts majority of the people. In this paper, an interactive evolutionary technique has been proposed to improve the illusion factor in eye illusion images. Eye illusion or vision illusion is subjective and can differ from person to person. This technique utilizes an evolutionary algorithm, namely Differential Evolution (DE), to improve the vision deceiving factor for an Eye Illusion image. Modeling of human vision perception is impossible or at least very complicated even for a specific person. So, proposing a general fitness function as an optimization objective would not be an easy task. That is why an interactive optimization approach seems a reasonable approach in this regard. To the best of our knowledge, the current work is the first attempt which utilizes an interactive optimization technique to enhance vision illusion images. Performance of the proposed approach is verified on two eye illusion test cases, but that is applicable to other eye illusion image enhancements and also image processing tasks, such as image filtering.**

Keywords: Interactive Evolutionary Algorithm (IEA); Eye Illusion; Differential Evolution (DE); Image Enhancement

I. Introduction

Eyes are one of the most important organs of the human being and we are all the reflection of what we have seen throughout our life. Eye illusion or vision illusion is an interesting topic which has been studied for many years and lots of illusive designs have been created by different artists. Everybody has seen at least one deceiving scene in his/her life. Following figure shows one of the well-known eye illusion images, Bulging Checkerboard. In this checkerboard, a central bulge is visible, but that is illusionary and the checkerboard is fully regular.

Level of eye illusion is subjective because it is related to human's visual perception and it can vary from person to person. This characteristic makes the design hard because the final result should be illusive enough to deceive majority of people. A good procedure to achieve this goal is to collect viewers' opinion and incorporate their feedbacks in the design to enhance the result. This needs an interactive method; so, the feedback from the user can be utilized in the design process. This paper investigates the using of an interactive evolutionary algorithm for improving eye illusion in two case studies. The



Bulging Checkerboard, the central bulge is illusionary.

image quality is not good at first and proposed approach tries to improve image quality. This method can be applied to different kinds of image processing tasks with variant applications. The variables' search space involved in the design of an illusive image is complex and brute-force searching of all the possible combinations (parameter levels) is impossible or at least time consuming, so desirable method is to use an optimizer to enhance the image. Because of the subjective nature of this problem, using an interactive evolutionary algorithm seems to be meaningful. The absence of a mathematical model to be used as objective function makes users' feedback is being used to select the best candidate in the population. There have been some other researches in image processing field who utilized interactive evolutionary algorithms (IEA) [2], [3]; they have mainly used a genetic algorithm as an optimizer. One of the topics in image processing is about the sequence of filters. Filter sequence can affect the enhanced image, so optimizing this sequence can improve the final result. IEA has been used to optimize the filter sequence in [6]. There are many IEC researches in the field of evolutionary graphic art and computer graphic animation [4], [9-12]. Unlike most of the researches which use genetic algorithm; our approach is based on DE algorithm.

The rest of this paper is organized as follows. Section II describes the Interactive Evolutionary algorithm. Section III is a short review of Differential Evolutionary Algorithm. Section IV describes the proposed approach. Section V presents the experimental verifications. Section VI concludes the paper

978-1-61284-072-7/11/$26.00 ©2011 IEEE

II. Interactive Evolutionary Algorithm (IEA)

When there is not any form of fitness function for the problem and no mathematical model can be defined then using interactive evolutionary algorithm would be a proper option.

In this case, the user chooses the best candidate or ranks the individuals in the current population. Based on the user's feedback, the optimizer generates the next generation. This method is widely used for problems which are based on human's perception, such as art related designs.

The main problem with this method is user fatigue which can affect final result. If the problem consists of too many parameters and computation time and iterations are too long then it makes user tired and the user's response is degraded. In order to overcome this problem, the interactive evolutionary algorithm should be designed in a way that it tries to solve a low dimensional problem which needs small population size and leads small number of function evaluations (i.e., number of user feedbacks).

In this paper, the function which is used for changing the picture parameters has only four and three variables for first and second case studies, respectively, which make the optimization problem easier and so the convergence will occur with less number of iterations.

The flowchart of the IEA is shown in Fig.1. As seen, the selection is being done by the user; unlike other optimization methods which use an explicit fitness function in evolutionary search process.

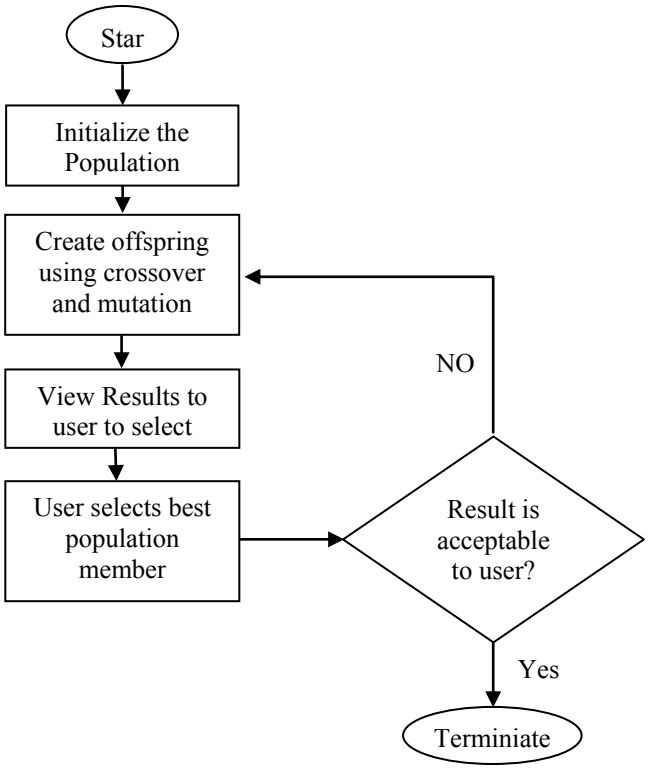


Fig. 1: A general flowchart of Interactive Evolutionary Algorithm (IEA)

III. A Short Overview of DE Algorithm

DE was introduced by Price and Storn. It was the result of his work to solve the Chebychev Polynomial fitting Problem that had been posed to him by Rainer Storn [1]. Price came up with the idea of using vector differences for perturbing population individuals. By this way, DE was born. DE works on multi-dimensional real-valued problems which are not necessarily continuous or differentiable.

DE is a simple effective and robust population-based optimization algorithm. The main idea behind DE is a scheme for generating trial parameter vectors. Basically, for every vector in the population (called target vector), DE selects randomly two other vectors, then subtracts them and adds the weighted difference to a randomly chosen third vector (called the base vector) to produce a mutant vector. Then, for every vector in the mutant population, it uses a user defined value called Crossover rate (Cr) to control the fraction of the parameter values which are copied from the mutant and target vector to the trial vector. Finally, for the selection step, if the trial vector has an equal or lower fitness value (for minimization problem) than that of its target vector, it replaces the target vector in the next generation; otherwise, the target retains its place in the population for the next generation. These steps are repeated for every vector in the population to produce the next new population. Algorithm 1 presents pseudocode of the classical Differential Evolution (DE) algorithm [1]. Three main operators (mutation, crossover, and selection) are given in lines 5-6, 7-13, and 15-19, respectively.

Algorithm 1: **Pseudocode of DE Algorithm**. P_0: Initial population, N_p: Population size, V: Noise vector, U: Trial vector, D: Problem dimension, BFV: Best fitness value so far, VTR: Value-to-reach, NFC: Number of function calls, MAX_{NFC}: Maximum number of function calls, F: Mutation constant, rand(0,1): Uniformly generated random number, C_r: Crossover rate, f(.): Objective function, P': Population of the next generation.

```
1: Generate uniformly distributed random population P0
2: while (BFV>VTR and NFC<MAXNFC) do
3:// Generate-and-Test-Loop
4: for i=0 to Np do
5:    Select three parents Xa, Xb, and Xc randomly from current population
       where i≠a≠b≠c
                    //Mutation
6:        Vi←Xa + F * (Xc-Xb)
                    //Crossover
7:        for j= 0 to D do
8:           if rand(0,1)<Cr then
9:                 Ui,j ← Vi,j
10:          else
11:                Ui,j ← Xi,j
12:          end if
13:       end for
           // Selection
14:       Evaluate Ui
15:   if (f(Ui)≤f(Xi)) then
16:      X'i← Ui
17:   else
18:      X'i←Xi
19:   end if
20: end for
21: X←X'
22: end while
```

IV. Proposed Approach

Differential Evolution Algorithm is a well-known effective robust algorithm which has been utilized as an optimizer in this paper. Every Eye illusion image has its unique features which should be addressed and integrated with DE to produce functioning solution. This paper addresses two different eye illusion case studies which will be discussed in the following subsections, A and B.

A. Case study 1: Checker-Shadow illusion image

The proposed approach for this type of eye illusion depends on manipulating input image intensity values by mapping input intensity range [P1, P2] to output range [P3, P4] and saturating to output range boundaries, all input values those are outside the output range. The proposed approach uses DE to search a four-dimension solution space (two variables for input range and two for output range those will form the intensity values of the output image). The corresponding four variables have boundaries and variable criteria that should be met as follows:

A. P1and P3 ∈ [0, 1).
B. P2 and P4 ∈ [0, 1].
C. P2 > P1
D. P4 > P3

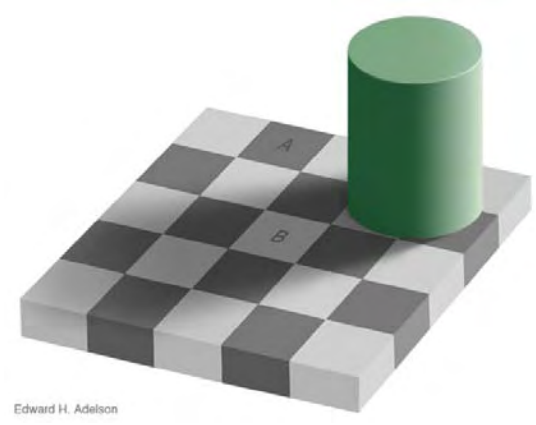


Fig. 2: Checker-Shadow Illusion (main image is the courtesy of MIT university http://web.mit.edu/persci/people/adelson/)

Having these conditions will require correcting parameters' values after every mutation and crossover, before new generation is utilized to generate images to be exposed to the user for selection. A special user interface (Fig. 3) has been developed to realize this approach and allow user selection.

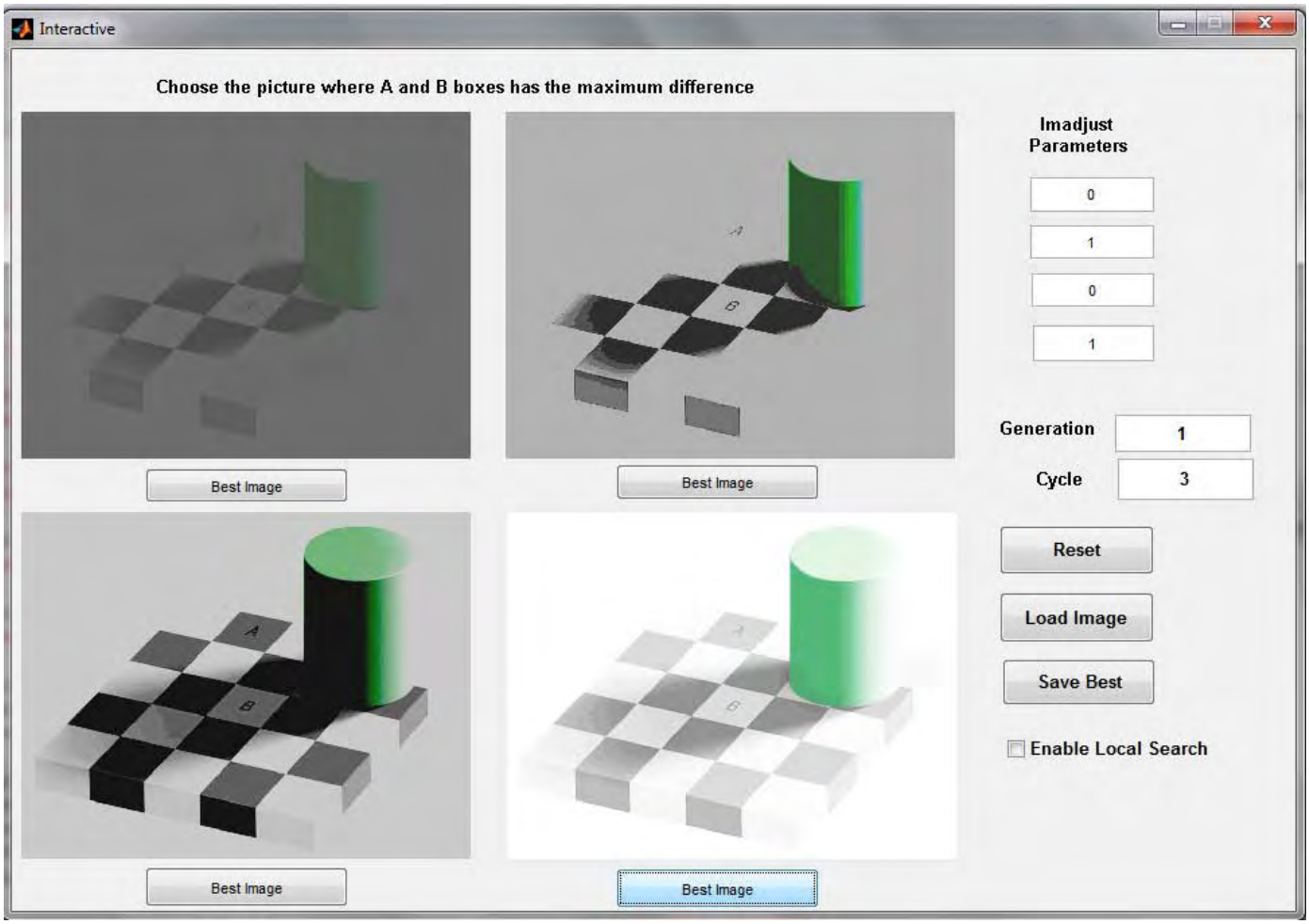


Fig. 3: The snapshot of GUI for the first case study, Checker-Shadow illusion image.

The following is our proposed approach:

1: Generate NP = 40 uniformly distributed random population
2: Correct populations to comply with criteria
3: Program start with Best member = original picture. It will be replaced later when new best member is found
4: While (user did not terminate)
5: While (end of population not reached)
6: View three picture of the population plus last best member
7: User choose a new best member or keep the previous one
8: End (inner while loop)
9: Generate new population using DE
10: Correct newly generated population:
if (P1 == 1) then regenerate P1 till it is less than one
If (P3 == 1) then regenerate P3 till it is less than one
If (P2 < P1) then regenerate P2 to a value in the range [P1,1]
If (P3 < P4) then regenerate P4 to a value in the range [P3,1]
11: End (outer while loop)
12: Save best member to disk.

The program will generate 40 images in every generation to be evaluated by user. This is to comply with DE algorithm recommendations which states the population size should be 8D-10D, where D indicates the problem's dimension.

B. Case study 2: Checkerboard illusion image

Proposed approach for this type of eye illusion problem depends on changing the thickness of the separator line between rows, gray level of this line, and finally the amount of applied shifts in pixel on rows. The algorithm to enhance this image is similar to previous one except for the dimension which is three for the mentioned design, population size is NP=30, and the correction criteria used for the Checkerboard (Fig 4) illusion problem is as the follows:

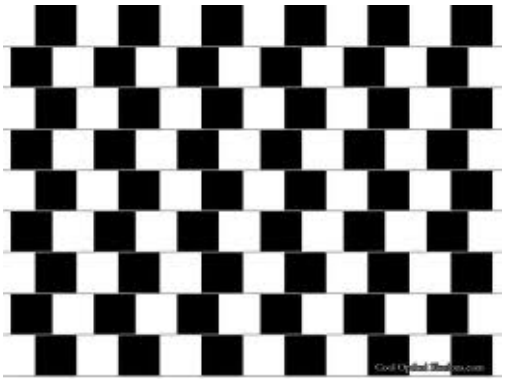


Fig. 4: Checkerboard illusion sample image.

- Separator line thickness is in the range [1,3]
- Separator line Gray level is in the range [0,255]
- Shift between rows varies in the range [0,Tile size in Pixel]

Fig. 5, the new interface that is designed for this problem which incorporates new features to enable users to change rows' number and pixels numbers in every tile.

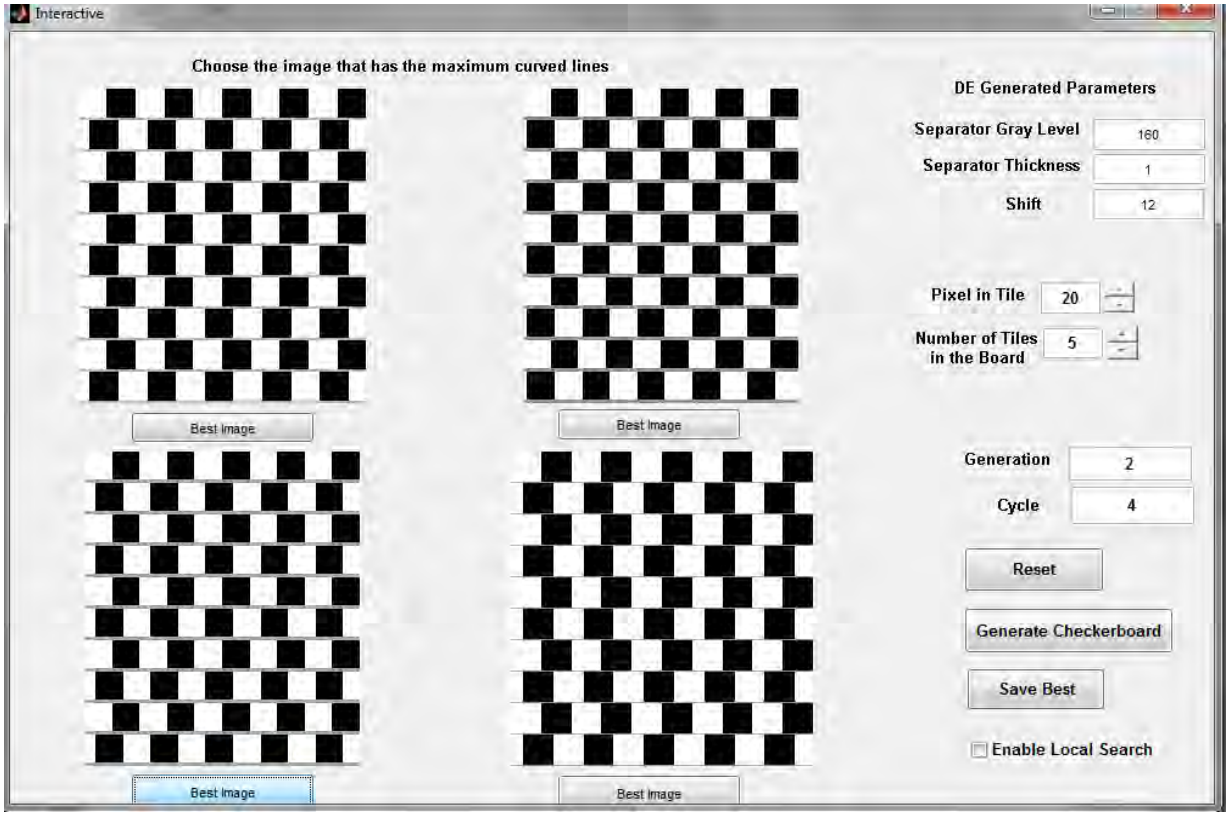


Fig. 5: The snapshot of GUI for the second case study, Checkerboard illusion image.

V. Experimental Verification

DE algorithm parameters are set similar to literature cited right after each parameter

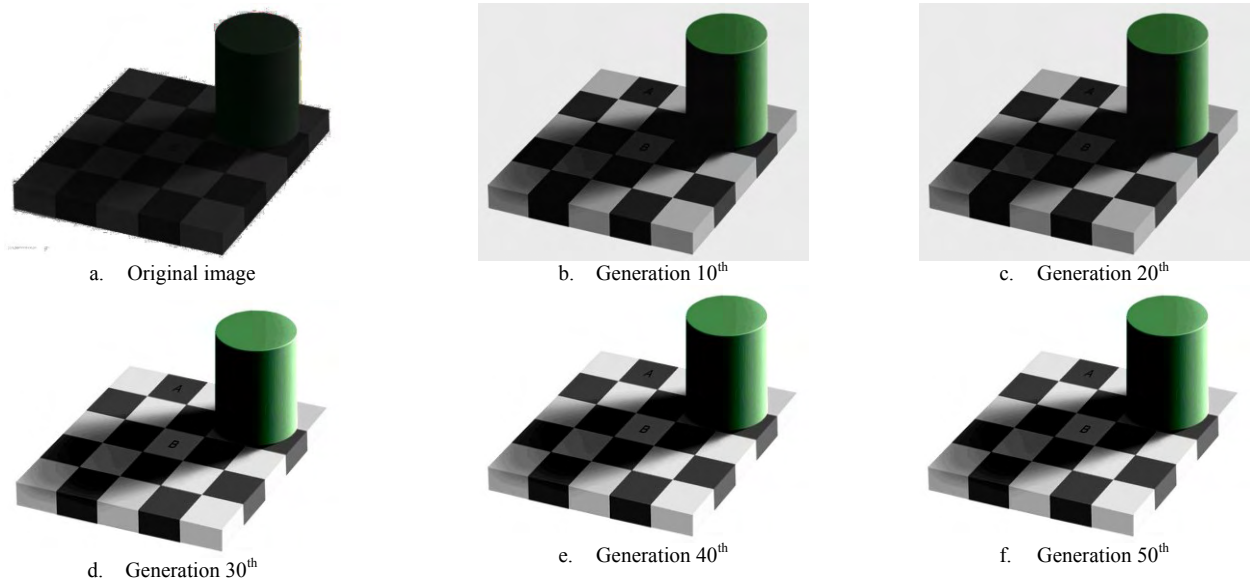
- Cr parameter was set to 0.9 [15-19]
- F parameter was set to 0.5 for both problems discussed in this paper [15-19]

A. Results for Checker-Shadow illusion problem

We used three distorted images with a bad quality to test the proposed algorithm. The best image is captured every ten generations and results are shown below. The illusion is not clear in the input images, but as it is shown, after some generations the illusion can be seen in the images and it gets better and better generation by generation.

The illusion in this image happens in the black and white squares marked as A and B. These two squares have the same color but because of the surrounding squares and the shadow of the cylindrical the viewer cannot see that and believes that A is darker than B

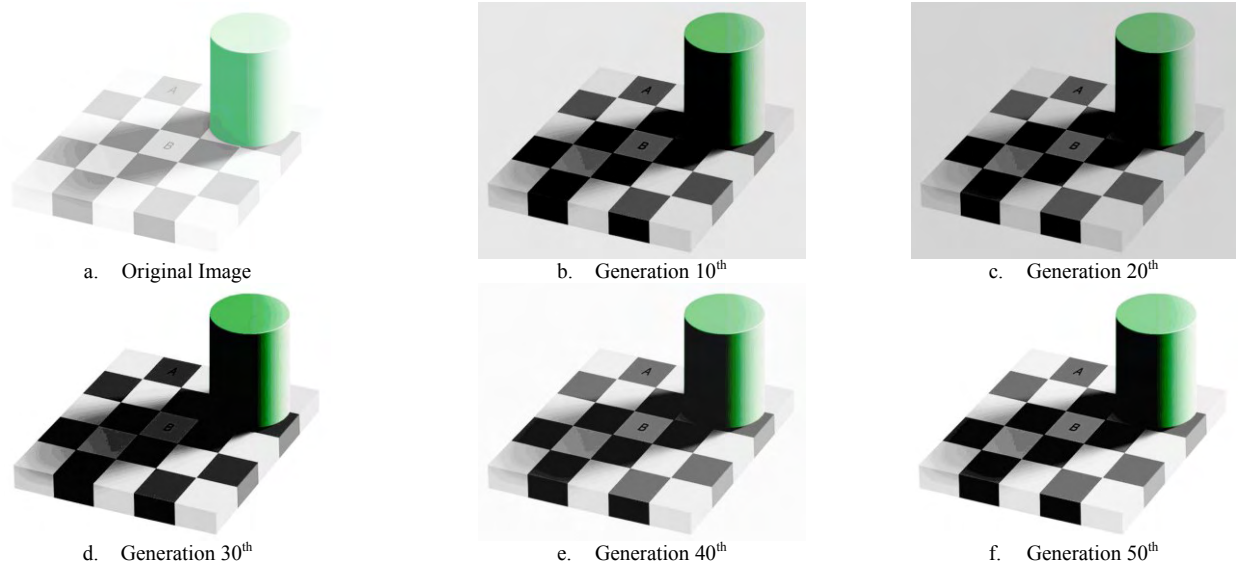
In Fig. 6, the first sample is a completely poor dark image and even the green cylindrical object on the right hand side of the image seems to be grey. After 20 generations the image is clear and the illusion can be seen obviously. In Fig. 7, the second sample starts with a very bright image with a poor eye illusion. In generation 40th, DE produces images with higher illusion quality. As the user continues to choose better images DE keeps improving and produces more good quality images in the next generations. And this can be seen clearly in the generation 50th



a. Original image b. Generation 10th c. Generation 20th

d. Generation 30th e. Generation 40th f. Generation 50th

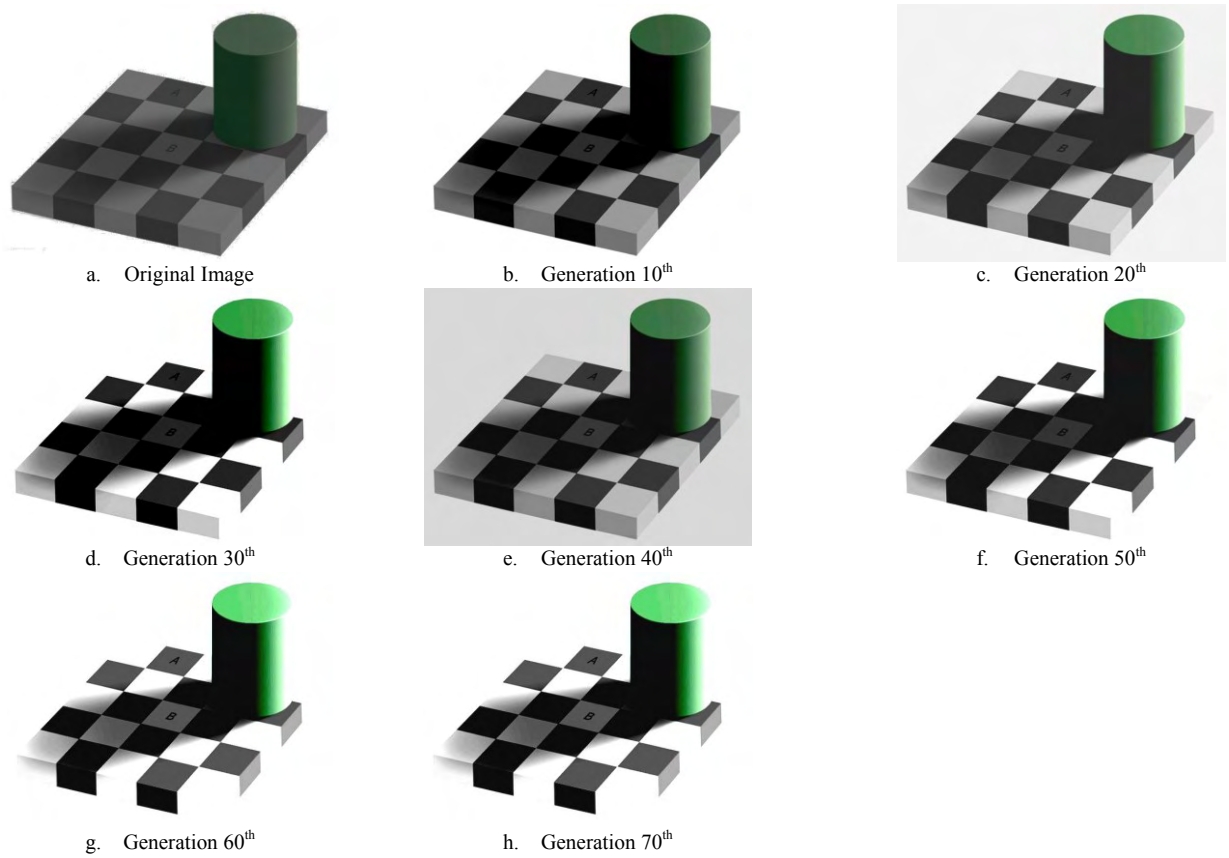
Fig. 6: First experiment, input image and the best results after 10, 20, 30, 40, and 50 generations.



a. Original Image b. Generation 10th c. Generation 20th

d. Generation 30th e. Generation 40th f. Generation 50th

Fig. 7: Second experiment, input image and the best results after 10, 20, 30, 40, and 50 generations.



a. Original Image b. Generation 10th c. Generation 20th

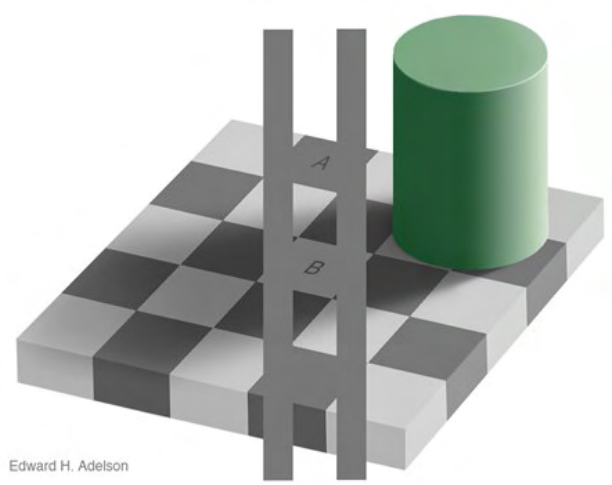
d. Generation 30th e. Generation 40th f. Generation 50th

g. Generation 60th h. Generation 70th

Fig. 8: The third experiment, input image and the best results after 10, 20, 30, 40, 50, 60, and 70 generations.

In Fig. 8, the third sample is dark and the image is not clear but as it can be seen in after 10 generations the image is clear and sharp and after 30 generations the result is very clear and the illusion can be clearly noticed.

Images produced in previous experiments for Checker Shadow Illusion clearly shows that, for human eyes, the squares A and B are different in color. The following image can be accepted as a proof in this regard.



B. Results for Checkerboard Illusion problem

The Checkerboard illusion image will lead the human viewer to think that the rows in the image are curved and non-horizontal while the reality is the opposite; they are straight and horizontal.

Fig. 10, the first generated images do not present a noticeable eye illusion quality.

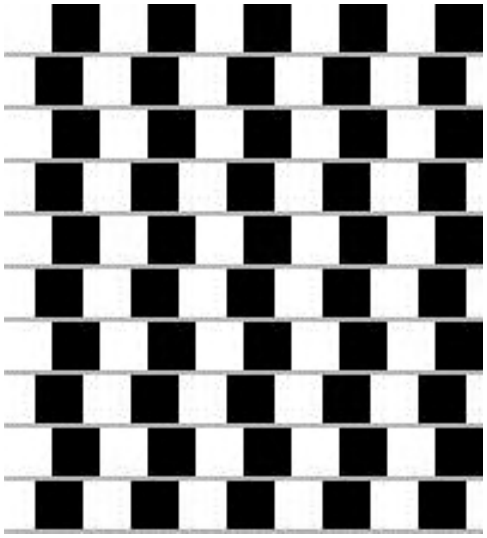


Figure 10: First generation

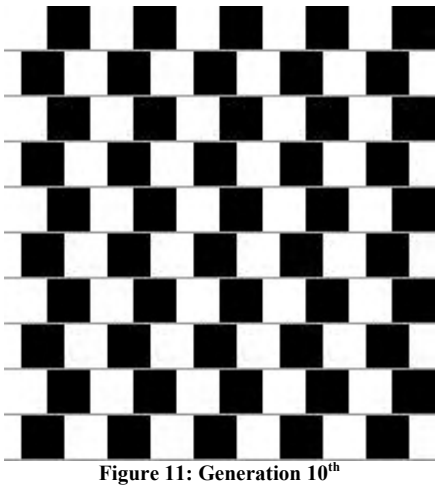


Figure 11: Generation 10th

But in Fig. 11, the best member of generation 10th clearly shows high illusion quality.

VI. Conclusion Remarks

A limited number of evaluations are feasible in situations where we have to use human-based evaluations, as it is the case in eye illusion designs. Eye fatigue becomes a limiting factor in complex problems where huge interactions are required.

Problems with mathematical fitness functions can be easily evaluated for thousands, or even millions with no deterioration in evaluation quality, but that is not the case for any interactive optimization approach.

These points make it mandatory to find faster converging algorithms for problems that need high number of human-based evaluations. Investigating hybrid metahuristics, such as combining DE and a local search algorithm, could be a reasonable direction for our future research. Different DE algorithms can be applied to solve IEA problems and investigating performance of these various algorithms

The proposed approach in this paper can be applied to other image processing fields subject to customizing the proposed algorithm to a targeted application such as image contrast enhancement or filtering. It builds another research direction in this area. This work can be assumed as a first step in computerized designing of eye illusion images with higher qualities and amazing characteristics. In fact, user can define the type of the illusion which he/she is interested; then computers can design images which include the targeted illusion with a highest deceptiveness.

References

[1] K.V. Price, R.M. Storn, J.A. Lampinen, Differential Evolution A Practical Approach to Global Optimization, Springer, 2005, ISBN:3540209506.

[2] J. Graf and W. Banzhaf, "Interactive evolution of images" in 4th Annual Conf. on Evolutionary Programming, San Diego, CA, USA, pp. 53-56, MIT Press, Mar. 1995.

[3] K. Otoba, K.Tanaka and M. Hitafuji, " Image Processing and Interactive Selection with Java Based on Genetic Algorithm" in 3rd IFAC/CIGR Workshop on Artifical Intelligence in Agriculture, Makuhari, Japan, pp.83-88, Apr. 1996.

[4] S. Baluja, D. Pomerleau, and T.Jochem, "simulating user's preferences: Towards automated artificial evolution for computer generated images" Tech. Rep. CMU-CS-93-198, CMU Computer Science Technical Reports, Oct. 1993.

[5] S.-B. Cho, "Applying interactive genetic algorithm to content based image retrieval: A preliminary result," in Workshop on Interactive Evolutionary Computation, Fukuoka, Japan, pp. 19-24, Mar. 1998.

[6] T.Mutoh, N.Komagata, and K.Ueda, "An experimental study for automatically generating image filter sequence by using simulated breeding" in Workshop on Interactive Evolutionary Computation, Fukuoka, Japan, pp. 7-12, Mar.1998.

[7] J. Graf, "Interactive evolutionary algorithms in design" in Int. Conf. on Artificial Neural Nets and Genetic Algorithms, Ales France, pp. 227-230, Apr. 1995

[8] J.-Y. Lee and S.-B. Cho, "Interactive genetic algorithm for content based image retrieval using interactive genetic algorithm," in Proc. Of FUZZ-IEEE'99, pp. II-998-II1003, Aug. 1999.

[9] P. J Angeline. "Evolving fractal movies" in 1st Annual Conf. on Genetic Programming, Stanford, CA, USA, pp. 503-511, July 1996.

[10] S. Baluja, D. Pomerleau, and T.Jochem, "Towards automated artificial evolution for computer generated images" Connection Science, vol. 6, no. 2&3, pp. 325-354, 1994.

[11] G. R. Greenfield, "Evolving expressions and art by choice" Leonard, vol. 33, no. 2, pp.93-99, 2000.

[12] G. R. Greenfield, "New directions for evolving expressions," in 1st annual Conf. of BRIDGES: Mathematical Connections in Art, Music, and Science, Winfield, Kansas, USA, July 1998.

[13] S. Das and P. N. Suganthan, "Differential Evolution: A Survey of the State-of-the-Art", IEEE Trans. On Evolutionary Computation, DOI: 10.1109/TEVC.2010.2059031, April 2011.

[14] Rahnamayan, S., H.R.Tizhoosh and M.M.A Salama,. "Opposition-based Differential Evolution" IEEE Trans. Evolutionary Computation, 12: 64-79, 2008

[15] R. Storn, K Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," in Journal of Global Optimization 11. Norwell, MA: Kluwer, 1997, pp. 341–359.

[16] J. Vesterstroem and R. Thomsen, "A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems," Proc. Congr. Evol. Comput., vol. 2, pp. 1980–1987, 2004.

[17] J. Liu and J. Lampinen, "A fuzzy adaptive differential evolution algorithm," Soft Computing-A Fusion of Foundations, Methodologies and Applications, vol. 9, no. 6, pp. 448–462, 2005

[18] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, "Self adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," IEEE Trans. Evol. Comput., vol. 10, no. 6, pp. 646–657, Dec. 2006.

[19] M. M. Ali and A. Törn, "Population set-based global optimization algorithms: Some modifications and numerical studies," Comput. Oper.Res., vol. 31, no. 10, pp. 1703–1725, 2004.