Assignment 5

Implementation

In this assignment, genetic algorithm was implemented to replicate a given gray scale image with only triangles. The program was implemented in Python with the use of opencv, and it was done from scratch.

To use genetic algorithm, I defined each component as follows:

1. **Gtype** is defined as a list of triangles. Each triangle defined by the vertices' coordinate and its alpha, grey scale color with a value from 0 to 1. The number of triangles in each Gtype is fixed according to a given parameter.
2. **Ptype** is the image generated according to the triangles specified by Gtype. The triangles can overlay each other and additively darken the overlapped region. Ptype is also normalized so that every pixel has a value from 0 to 1.
3. **Evaluation function** is the sum of squared difference of each pixel in the generated Ptype to a given goal image with the same size. Because the evaluation function is the distance of an image from the goal, a Gtype with lower evaluation has a better fitness than ones with higher evaluation, meaning that as the algorithm progress, the evaluation of Gtypes must be lower.
4. **Mutation** is done by randomly pick Gtypes from a generation according to a given mutation probability, and one random triangle from the list in each picked Gtype are slightly altered. The alteration could be either relocating a random vertex of the triangle or recoloring the triangle, with an equal chance.
5. **Crossing over** is defined with two methods. The first method is taking a number of the best Gtypes of the generation, mixing their triangles together and randomly selecting a certain number of triangles to form a new generation Gtype. The second method is forming a new generation Gtype by using the corresponding triangles from the current generation Gtypes with a given probability distribution. For example, let say the crossing over probability is *p*, the new generation Gtype's first triangle is selected from one of the first triangle of the current generation Gtypes, and the probability of selecting the first triangle from the best Gtype is (1-*p)*, probability for selecting the first triangle for the second best Gtype is *(1-p)p* and so on.

The summary of the adjustable parameters is as follow:

1. **numTri**: The number of triangles in each Gtype.
2. **generationSize**: The number of Gtypes in each generation.
3. **selectionSize**: The number of the best Gtypes to be selected and mixed their triangles to form a new generation. (only for the first crossing over method)
4. **crossoverProb**: The crossing over probability which corresponding to *p* in the example given above. (only for the second crossing over method)
5. **mutationProb**: The probability that a Gtype is picked to be mutated.
6. **maxGeneration**: The number of generations the algorithm will generate.

The goal image of this algorithm is set to 512 by 512 pixels *goldhill.png*.

Figure 1. *goldhill.png*

Result and Discussion

The first experiment that I did was to find the best method for the crossing over process. The first and the second trial run of the program utilized the first and the second crossing over method respectively. I set the parameter for each trial as follows in the Table 3.

Table 3. Parameters setting for the first and second trial runs

Parameters	First trial	Second trial
numTri	50	50
generationSize	20	20
selectionSize	3	-
crossoverProb	-	0.5
mutationProb	0.5	0.5
maxGeneration	100	100

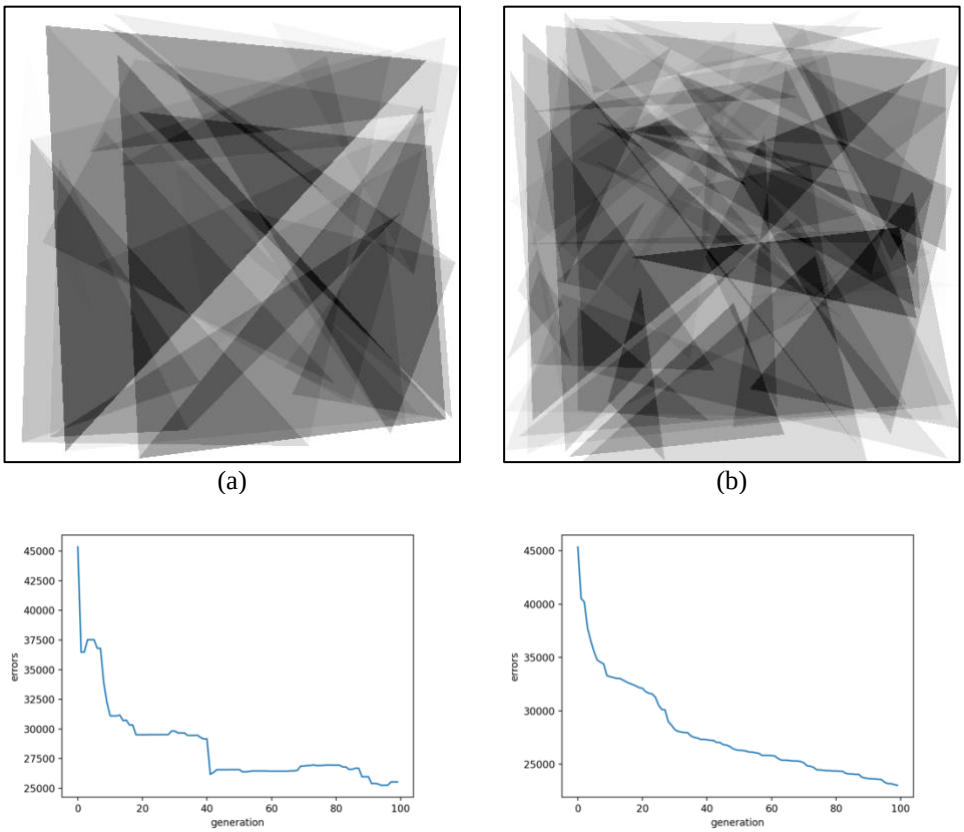


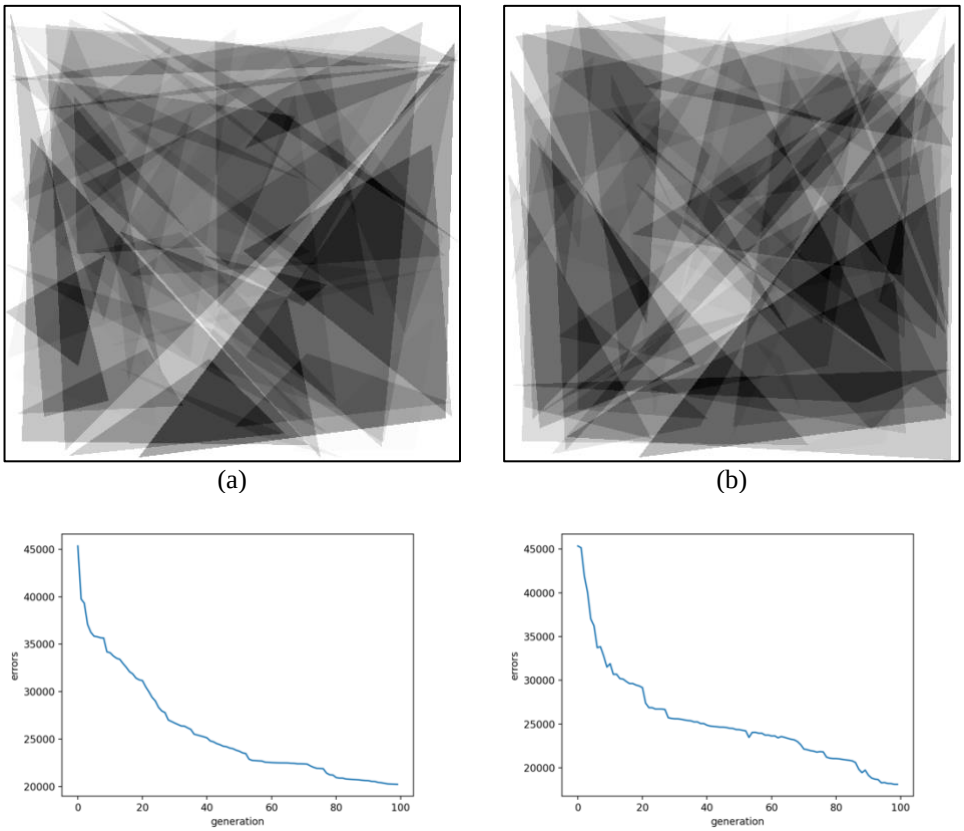
Figure 2. The best Ptypes of the last generation and error by generation plots of the first (left column) and the second (right column) trials.

For the result of the first experiment, I displayed the best Ptypes of the last generation together with the plots of the errors by generations for both program in the Figure 2 above. As you can see in Figure 2 (a) and (b), the second method of crossing over yielded a much more complex image. The reason is that the first method of crossing over is done by forming a new generation from a very limited pool of triangles. After a few generations, the pool of triangles will only contain only similar triangles and as a result, the generated image is just similar triangles overlaying each other. On the other hand, the second method of crossing over did provide a separate development of each triangle. For example, from what I have observed, a triangle that tries to represent a darker part of the image will keep getting darker as the generation progress. Furthermore, considering the plots in Figure 2 (c) and (d), the errors of figure (d) continually declined which suggested that the second method of crossing over has more potential to yield a better result,

The second experiment that I did was to see how altering crossing over probability and altering mutation probability could affect the result.

The third and the fourth trial runs of the program have the same parameter setting as the second trial except the crossing over probability. The former and the later has the crossing over probability of 0.3 and 0.8 instead of 0.5. The result is shown in the Figure 3 below.

Similarly, the fifth and the sixth trial runs of the program have the same parameter setting as the second trial except the mutation probability which was 0.3 and 0.8 respectively. The result is shown in the Figure 4 below.



(a) (b)

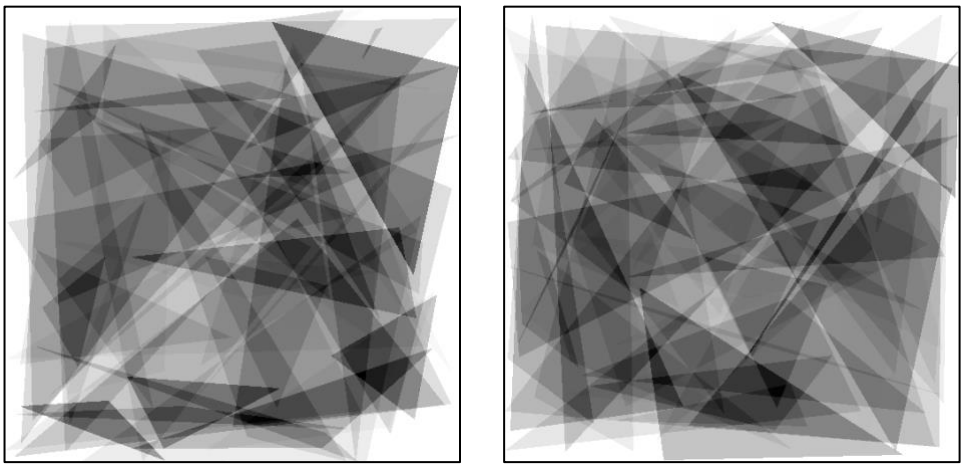
(c) (d)

Figure 3. The best Ptypes of the last generation and error by generation plots of the third (left column) and the fourth (right column) trials.

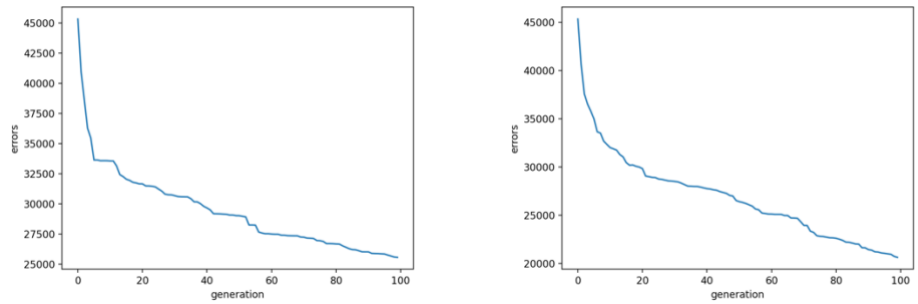
As shown in Figure 3, the result suggested that the fourth trial, which was set to have 0.8 crossing over probability, is performing slightly better. However, the third trial with 0.3 crossing over probability has more smooth errors plot as you can see in Figure 3 (c) and (d). For the Ptypes in Figure 3 (a) and (b), there are many similar triangles yielded from the two trials.

Next, in Figure 4 (c) and (d), although the slopes of the errors plots look the same, the sixth trial has much faster progression between each generation. In other word, the 100^{th} generation of the fifth trial has a similar error to the 60^{th} generation of sixth trial. This suggested that both settings' potential is about the same but the sixth trial with higher mutation probability is just faster.

For the third experiment, I tried to increase the generation size. The seventh trial run of the program has the same setting as the second trial run but with the generation size of 40. The result is shown in Figure 5.

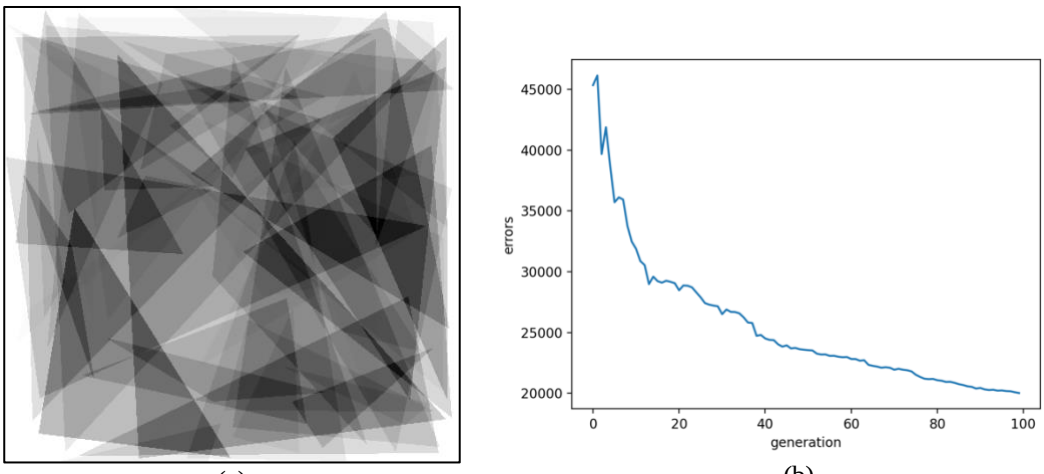


(a) (b)



(c) (d)

Figure 4. The best Ptypes of the last generation and error by generation plots of the fifth (left column) and the sixth (right column) trials.



(a) (b)

Figure 5. The best Ptype of the last generation and error by generations plot of the seventh trial.

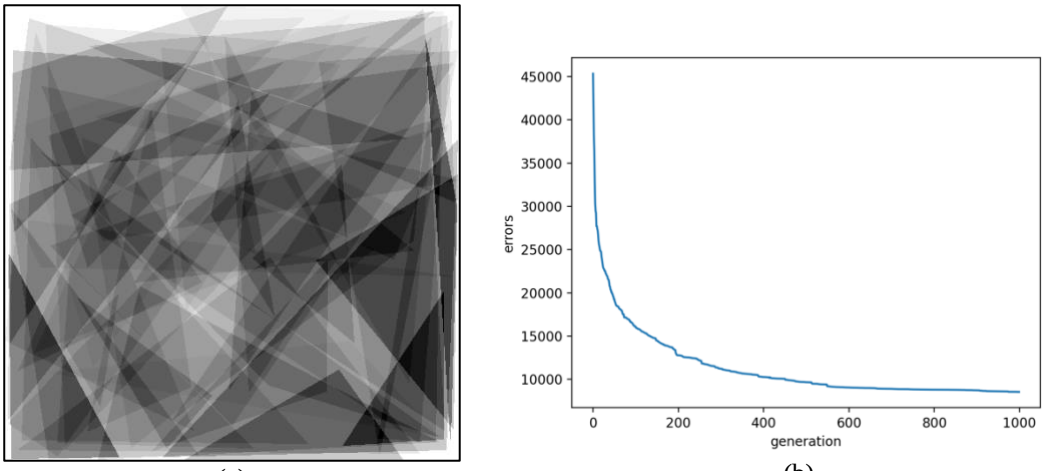
Comparing the result of the seventh trial in Figure 5 to the second trial in Figure 2 (b) and (d), the seventh trial with higher generation size perform much better than the second trial. The reason might be that the bigger generation size generates more unique triangles and there is a lot more mutation happened which allow the next generation to have many more combination to

yield a better result. However, the time taken for each progression of a generation is linearly increased as the generation size increased. It took twice as much time to train the seventh trial than the second trial.

Lastly, I just combine every good result together and let the program run for 1000 generations for my eighth trial run. The parameters setting is shown below in Table 4. The result is shown in Figure 6.

Table 4. Parameters setting for the eighth trial run

Parameters	Eighth trial
numTri	50
generationSize	40
crossoverProb	0.8
mutationProb	0.8
maxGeneration	1000



(a) (b)

Figure 6. The best Ptype of the last generation and error by generations plot of the eighth trial.

As shown in Figure 6 above, the error plot seems to be converged and the best Ptype of the last generation is displayed in Figure 6 (a). Although the program is converged, the generated image is not quite the same with our goal, *goldhill.png* in Figure 1. However, it did capture the feature of our goal, dark and light color regions or the roofs outline for instance. This could be that the number of triangles is too small or our goal has too high resolution which either way, further investigation is needed.