DIGITAL ART USING GENETIC ALGORITHM

Fadi Younes, Innopolis University 27/04/2020

Problem Description

A systematic method using genetic algorithm is implemented in Python language to perform a digital art of reproducing an image from a given input image. The goal is to produce the output image as similar to input image as possible.

Methodology

A design methodology is developed using genetic algorithm to solve the problem of reproduction of a image from the given input image. In this method, the image building input performed using circles. The opacity (alpha value) of the output image is considered to be fixed among all the pixels taking the value of 255.

The detailed design methodology is explained in the Figure 1. The drawing process continues with a outer loop where variable n takes a value from the set of N,

$$N = \{1, 2, 4, 8, \ldots, n_{max}\} \tag{1}$$

where, n_{max}, takes the value of maximum pixel number of the original image in any axis. For example, if the original image is of 512x512 pixels, the n_{max} takes the value of 512. Each element index of N corresponds to the index of an iteration. It is possible to stop at any iteration. The more iterations the programs passes, the more close the output image to the original image generated.

Iteration 0

As mentioned above, the first iteration starts with $n = 1$. The original image is divided into 1 section of area. This means, at the first iteration, the entire image area is considered. For the selected entire area of the original image, the average values of red, green, blue channels are determined. This set of red, green, and blue channels are the target value of the optimization session soon to be started. Note that the alpha channel value is ignored throughout the current work and kept to be taking a constant value of 255 in all pixel of the output image.

Genetic Algorithm session

The genetic algorithm session starts to guess the target combination of average values of red, green, and blue channels of the entire image.

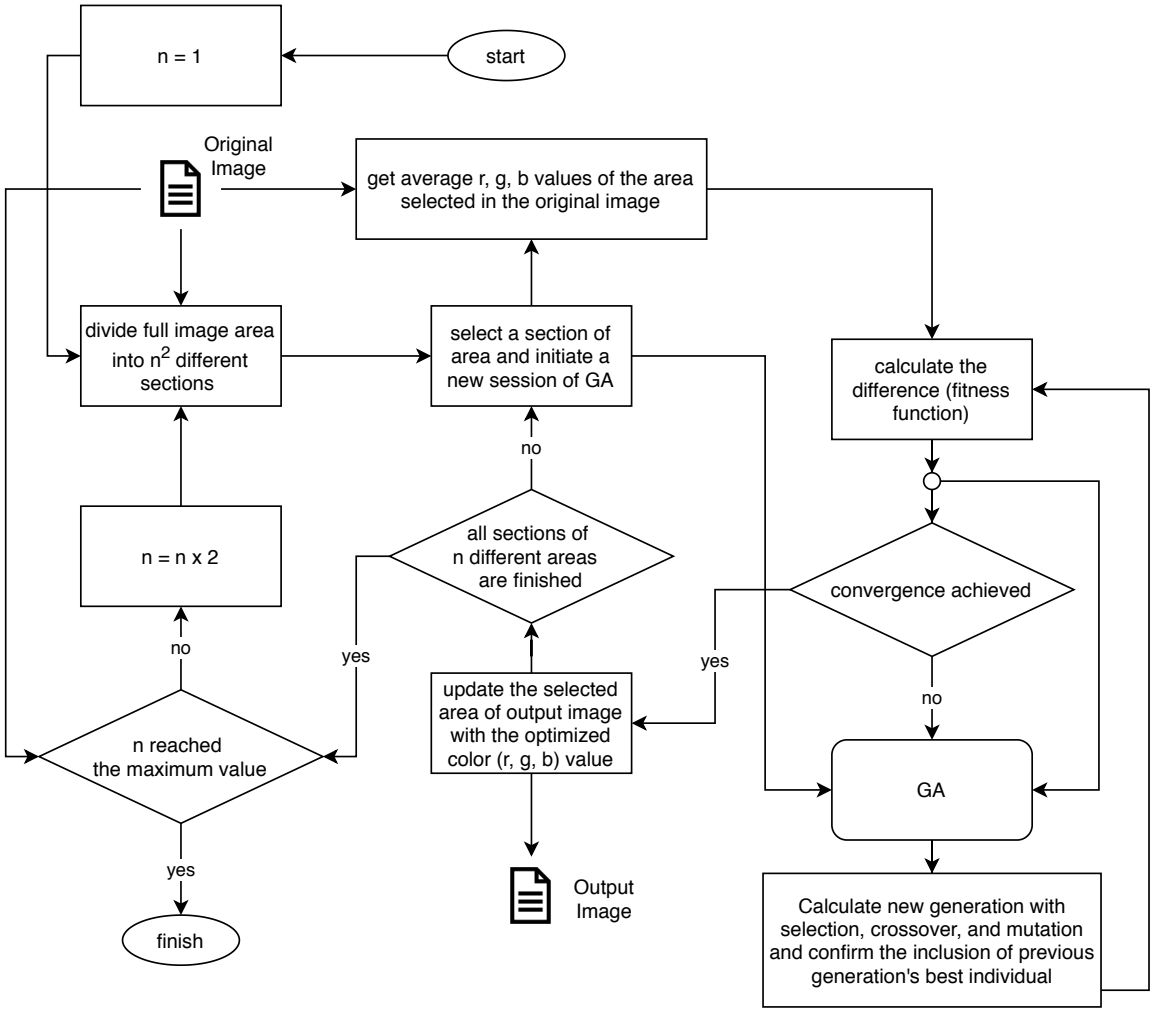


Figure 1: Methodology of redrawing the image using genetic algorithm

Design variable

The set of design variable x includes the value of red, green, and blue as described in eq.(2)

$$X = \{red, green, blue\} \tag{2}$$

Each element of the design variable set x takes an integer value i ranging from 0 to 155.

$$i \in \mathbb{Z} : 0 \leq \mathbb{Z} \leq 255 \tag{3}$$

Objective function

The objective function of the current genetic algorithm is as follows,

$$obj = \Delta red^2 + \Delta green^2 + \Delta blue^2 \tag{4}$$

The difference in the red value of the current design variable from the target red value of the original image of the selected section is defined by Δred. Similarly, for green and blue, the corresponding differences are represented by $\Delta green$ and $\Delta blue$, respectively.

The target value of red red_t is determined as follows,

$$red_t = red_{avg} + \Omega(-\min(red_{avg}, 30), \min(256 - red_{avg}, 30)) \tag{5}$$

where, red_{avg} denotes the average value of red channel in the selected section of the original image. The random value generator function Ω generates a random value within the range of its parameter arguments. Similarly, the target green $green_t$ and target blue $blue_t$ are determined.

Update output figure

When the optimized design variable x^* is determined, a circle is drawn with color defined by the optimal value of red^*, $green^*$, and $blue^*$.

$$x^* = \{red^*, green^*, blue^*\} \tag{6}$$

The center of the circle (x, y) is located at the exact middle point of the selected area,

$$x = x_{mid} + rand \tag{7}$$

$$y = y_{mid} + rand \tag{8}$$

where, x_{len} denotes the length of x dimension of the selected area and, similarly, y_{len} of y dimension. Note that, the alpha value is 255 for this circle. The random added value $rand$ is defined as,

$$rand = \Omega(-y_{len}/3, y_{len}/3). \tag{9}$$

The radius of the circle is determined as follows,

$$radius = 2x_{len}/3 + rand. \tag{10}$$

Iteration 1

The second iteration continues with the value of n taking the value of 2 second element of N mentioned in eq. (1). With this $n = 2$, the original image is divided by 2 both horizontally and vertically. This process results 4 different areas. For each of these 4 sections, a separate genetic algorithm session is run to find the average value of red, green, and blue value within the selected section of the original image. With the optimal set of red, green, and blue color the corresponding sections of the circle is drawn. Thereby, in the second iteration, four circles are drawn on the output image.

Next iterations

The above-mentioned procedure continues with iterations for the respective element in N. For any value of $n \in N$, the total n^2 number of sections of the image are obtained. Therefore, n^2 number of optimization session runs during the iteration. The iteration continues until $n \in N$ reaches the maximum value, which is either the number of pixel in a dimension of the given input image or any lower number set as a maximum iteration number parameter.

Genetic Operation

The genetic algorithm is coded and implemented in Python. The genetic algorithm consists to genetic operations, namely, selection, crossover, mutation. At the beginning of genetic algorithm session, the initial population set needed to be created.

Initialization

During iteration 0, the initial population is created randomly. However, for the other iterations, one individual x among the initial population is created from the existing color value $\{red, green, blue\}$ of the selected area to be optimized. The remaining individuals of the initial population are created randomly. The corresponding objective function values are calculated using eq.(4).

0.1 Fitness

Now the fitness of every individual in the population is calculated using eq.(4). Then the population is sorted based on the individual fitness, which helps to ease the future operations in genetic algorithm.

Carry on previous best

At this point, the previous generation's best individual (if any previous generation exists) checked to be included in case the current best is less fit compared to the previous best individual.

Selection

The selection operation keeps some individuals and removes the others according to a given factor of selection.

Crossover

Whatever number of individuals removed from the population are filled up with child individuals created from two parent individuals. These parents are selected from the only are selected to be kept in the selection process mentioned above.

Mutation

Then from the updated population after the crossover operation, individual chromosomes are randomly selected based on the rate of given mutation factor. After this mutation operation, next generation is prepared.

Results

The current digital art algorithm is implemented in Python language and tested with an input image as shown in Figure 2a. The output image, as shown in Figure 2b, is generated after eighth iteration.

(a) Input image

(b) Output image (iteration = 8)

Figure 2: Input and output images

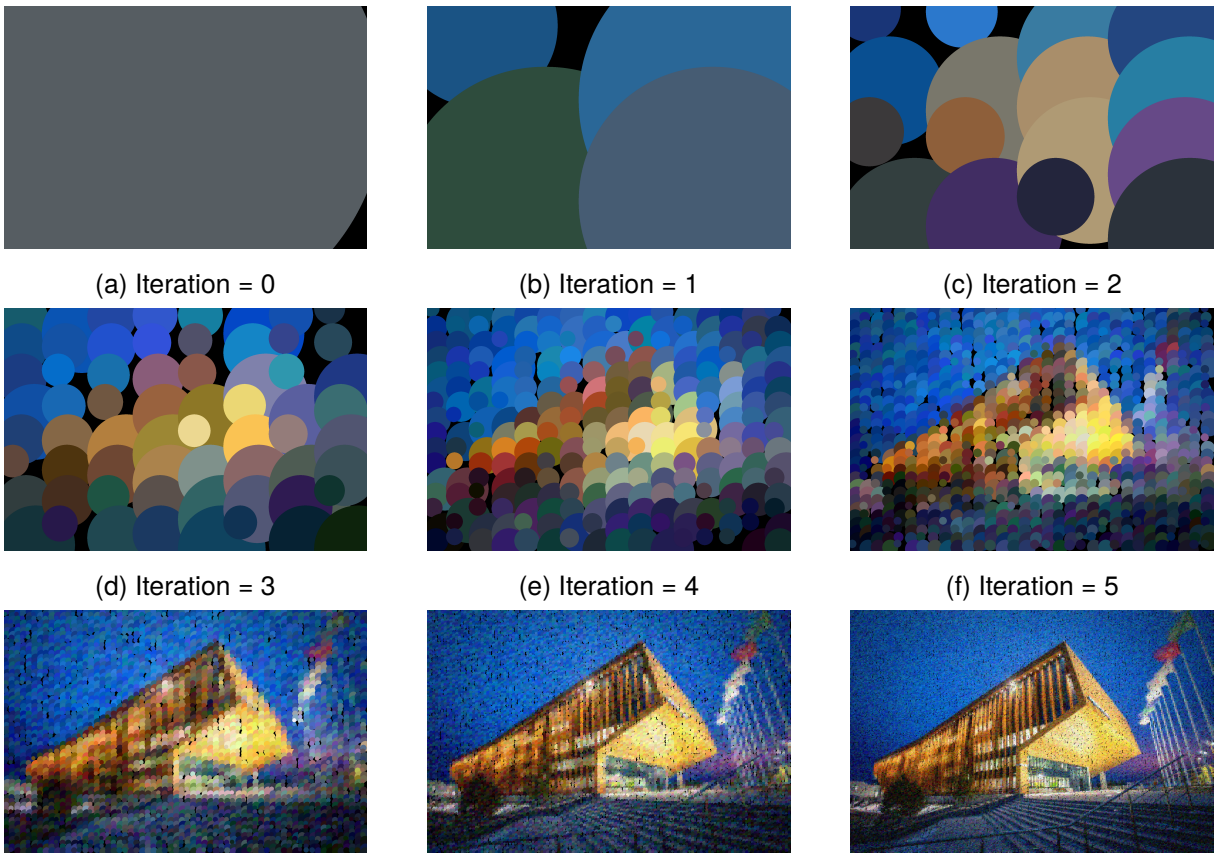
The current implementation includes only the red, green, and blue channel values, but not the alpha channel values. The alpha channel values are fixed and takes the value of 255. The genetic algorithm tested with the parameters of population size taking the value of 500, maximum generation number of 100, selection factor of 0.2, and mutation rate of 0.1. The progress of the drawing of output image is shown in Figure 3.

As can be seen from the figures, for the iteration number of zero, only one circle of drawn covering almost entire area of the image. Similarly, iteration number one includes 4 circles, iteration two includes 16 circles and so on. Note that each circle corresponds a completely independent genetic algorithm session. Note that, the import packages used in this algorithm are listed in Listing 1.

Listing 1: Digital art Python script uses some import packages

```
1 import copy
2 import random
3 import pygame
4 from PIL import Image
```

The optimization is finished after iteration number eight. The future development proposal includes inclusion of alpha values in the optimization design variable set X in eq. (2). This algorithm can address all varieties of images with varying transparency, i.e. the alpha values.



(a) Iteration = 0 (b) Iteration = 1 (c) Iteration = 2

(d) Iteration = 3 (e) Iteration = 4 (f) Iteration = 5

(g) Iteration = 6 (h) Iteration = 7 (i) Iteration = 8

Figure 3: Progress of output image optimization over the iterations

Conclusion

A digital art algorithm is designed and implemented in Python language. This program includes the development and implementation of genetic algorithm from scratch. The algorithm is tested with an example image and the produced output image has a nice match with the original input image.