Van Gogh Evolucional

1st Miranda Arias Andrés
2017075170
Student
Costa Rica Institute of Technology
Heredia, Costa Rica
Email: anmiranda@ic-itcr.ac.cr

2nd Sánchez Calderón Lery
2016254672
Student
Costa Rica Institute of Technology
Cartago, Costa Rica
Email: ljazminsc205@gmail.com

***Abstract*—This project consists in the implementation of genetic algorithms to be able to approximate an image from a randomly generated image, varying the pixel colors in no particular order. The algorithms are the ones responsible for the mutation and crossover of the genes (pixels) of every individual of the population, if applicable. The results are compared to the target image using algorithms to calculate distances and similarities between the population and the target until obtaining a satisfactory resulting new image similar to the target original.**

***Index Terms*—Genetic algorithms, evolution, mutation, genes, images, comparison, crossover, Euclidean distance, Manhattan distance**

I. Introduction

Genetic Algorithms are a type of algorithms that can be related to the Darwinian evolutionary theory's three main principles. These types of algorithms are inspired by the process of natural selection, which belong to a larger class of Evolutionary Algorithms. These algorithms are commonly used to solve optimization problems, which require a simulation of natural selection to be able to achieve the optimal solution. According to Shiffman [1], genetic algorithms follow the three main principles of the Evolutionary Theory: Heredity, Variation, and Selection, represented in a three-section algorithm consisting on Initialization, Selection and Reproduction. The concept of variation plays a role in the creation of the random population on n elements, given that the population follows no particular order in its generation, represented in the initialization stage. In the second step of the algorithm, Selection, the concept of Selection can be applied while calculating the fitness of each individual of the population. The third principle of Darwinian Natural Selection, Heredity, can be applied in the third stage of the algorithm: Reproduction. On this final step, the algorithm will run mutating and making crossovers of genes until getting a fit image as an optimal result.

Specifically applied to the project, the algorithms must evaluate and approximate a target image from a population of images made of randomly generated pixels. The quantity of those images is determined by a parameter entered by the user, as well as the target image which can be selected from the computer files. Afterwards, a fitness function is applied to verify which individuals fit best, using Euclidean Distance algorithm, Manhattan Distance algorithm and a properly invented algorithm. A process of genetic mutation and crossovers is made to generate newer generations of the previous population, and so on until a final result is obtained.

Finally, experiments will be made to determine under which parameters the program works optimally, and which distance algorithm is more effective than the others, if applicable. As well, a comparison of theoretical versus experimental time will be made.

II. Related Work

The creation of an algorithm is based on the necessity to solve a specific computational problem, or any problem that can be computationally solved. Emphasizing on genetic algorithms, their main purpose is to solve problems in the most optimal way, using Darwinian Natural Selection principles to evolve and find a solution quicker than another type of algorithm. Such types of algorithms can be implemented not only to solve optimization or search problems, but also to solve problems of routing, traffic and shipment, encryption and code breaking, image analysis, as well as computer gaming. In [3], Deepa and Sivanandam dedicate a chapter of their book *Introduction to Genetic Algorithms* to explaining the uses and applications of these algorithms. As well, in [2], Sonka et al, explain the uses of genetic algorithms in image processing, such as the main goal of this project.

III. Methodology

A. Genetic Algorithm

1) Step 1. Generating the population: The genetic algorithm programmed consists on four different steps or functions to be able to obtain the target image. The first step consists on generating the random population of images, according to the amount of images required by the user, as an input. It also requires the target image, to be able to begin comparing it in step 2. All these images with randomly generated pixels are inserted into an ArrayList of DNA type objects, which in turn is a Class that contains the BufferedImage which will be the individual to mutate and crossover (genes).

2) Step 2. Mutating the population: The second step in the process is the mutation of the population. Mutating a population is an algorithm that takes by parameter the percentage of genes (images) that will mutate. Basically the algorithm consists on generating a random number between 0 and 100, and if that number is less than the mutating percentage, the individual will mutate that pixel, meaning that a newly randomly generated color will be inserted in that specific (i, j) position of the image.

3) Step 3. Calculate fitness: Third step of the genetic algorithm is calculating the fitness or distance of each individual (image) in the population. Here, the distance algorithms are applied depending on the selection of the user. Regardless of the distance algorithm, the fitness will be how similar or different is each individual compared to the target image; the greater the fitness of one individual, the most similar to the final target image it will be, thus the higher the probability it will have of being chosen to pass its genetic information further down to the next generation.

4) Step 4. Crossover: The final step of the algorithm is to make a crossover of the individuals to create a new generation. This new generation will take the most fit individuals (those with greater fitness values), and will be filled up with other random individuals of the older population. The crossover will be made according to a value entered as an input by the user, and the individuals that do not make the fitness cut to go through to the next generating, but still do so, will be selected according to the last parameter entered by the user which is the percentage of the individuals unfit to go through.

B. Distance Algorithms

The distance algorithms will be implemented as a fitness function to calculate how accurate or not are the individuals of the population from the target image. Three different algorithms will be used. First the Euclidean Distance algorithm, second the Manhattan Distance algorithm, and lastly a proper algorithm implemented to calculate the fitness, or distance, of the images regarding the target image.

1) Euclidean Distance Algorithm: The Euclidean Distance algorithm is an algorithm to determine the Euclidean distance. This distance is described as the difference between two points in a metric space, which can be interpreted as the difference between two points in the same position of two different vectors. The Euclidean distance can be seen as a variation of the Pythagoras Theorem, given that the formula for Pythagoras can be seen in (1)

$$a^2 + b^2 = c^2 \tag{1}$$

$$c = \sqrt{a^2 + b^2} \tag{2}$$

Given that a point in a vectorial space can be seen as the (x, y) coordinates, and the difference between two points can be seen as the difference between the x and y coordinates, (3) is developed from (2), leading to the final Euclidean Distance formula in (4), with p and q being two vectors, n being the length of such vectors.

$$d = \sqrt{(x_2 - x_1)^2 + y_2 - y_1)^2} \quad (3)$$

$$D_{Euclidean}(p, q) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2} \quad (4)$$

2) Manhattan Distance Algorithm: The Manhattan Distance algorithm is another type of algorithm, such as the Euclidean Distance, used to determine the distance between two points in a Cartesian coordinates system. Also called Taxicab Geometry, the Manhattan Distance is a variation of the Euclidean Distance which calculates the distance between two different points as the sum of the absolute differences of their coordinates. In a vectorial space, that would be interpreted as the sum of the differences between p_i and q_i, with p and q being two n-dimensional vectors. The formula for the Manhattan Distance can be seen in (5), with n being the length of the vector.

$$D_{Manhattan}(p, q) = ||p - q||_i = \sum_{i=1}^{n} |p_i - q_i| \quad (5)$$

The main difference between the Euclidean and Manhattan Distance can be illustrated in Figure 1, since the red, yellow and blue lines all have the exact same minimum distance applying Manhattan Distance. However, Euclidean Distance, since it is derived from Pythagoras, it can calculate a straight distance from one point to the other, resulting in the green line being the one with the minimum distance.

Algorithm 1 Manhattan Distance

```
Require: BuffImg
Ensure: fitness
    $result = 0$
 2: for $i = 0$ to $n$ do do
        for $j = 0$ to $m$ do do
 4:         12 variable declarations and initialization
            $result + sum$
 6:     end for
    end for
 8: $fitness = result$
```

Analyzing the algorithm in [Algorithm 1], it requires a BufferedImage object which will be $n * m$, n being the width and m being the height. The resulting output will be the assignation of the fitness of the specific individual.
Line 1: variable assignation = 1
Line 2: *for* cycle = (n+1)
Line 3: *for* cycle = (m+1)
Line 4: 12 variable assignations, each = 1
Line 5: variable addition = 1
Line 8: variable assignation = 1
Given that the instructions executed the most times are both *for* cycles, the order analysis will leave the barometer in these two cycles, executing themselves $(n + 1)$ and $(m + 1)$ times respectively. This would lead to the following analysis for $O(f(n))$

$$O(f(n)) = O((n + 1) * (m + 1)) =$$

$$O((n * m) + n + m + 1) =$$

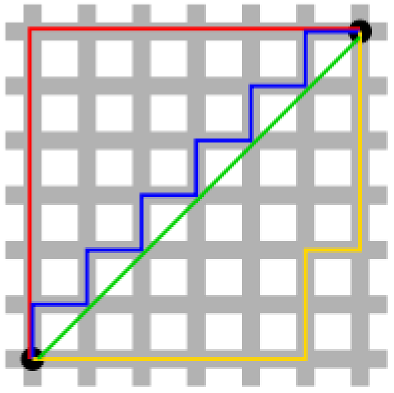
Removing the constants

$$O((n * m) + n + m) =$$

Finally, by applying the Maximum Order Law

$$O(n * m)$$

Fig. 1. Manhattan vs Euclidean Distance



3) Own Distance Algorithm - AL Distance: Given that what the distance algorithms previously discussed do is calculate the distance, fitness, or difference between two selected points, or in this case two different pixels (genes), an algorithm was created to calculate such distance between the pixels. The basic functionality of the algorithm consists on verifying, pixel by pixel, the RGB component of it and comparing the target image and the individual (randomly generated image). If their RGB component is exactly the same, the fitness is added a 1 value, and so on for every pixel. This means that the greater the result obtained, the more similar the images will be. In the algorithm, n represents the width of the image, while m represents the height of the image.

Algorithm 2 AL Distance Algorithm

```
Require: BuffImg
Ensure: fitness
    $result = 0$
 2: for $i = 0$ to $n$ do do
        for $j = 0$ to $m$ do do
 4:         if $target.getRGB == individual.getRGB$ then
                $result++$
 6:         end if
        end for
 8: end for
    $fitness = (result)^2$
```

Analyzing the algorithm in [Algorithm 2], it requires a BufferedImage object which will be $n*m$, n being the width and m being the height. The resulting output of the algorithm will be the assignation of the fitness value of the image related to the target.
Line 1: variable assignation = 1
Line 2: variable assignation, variable increment = 1, *for* cycle = (n+1)
Line 3: variable assignation, variable increment = 1, *for* cycle = (m+1)
Line 4: conditional = (m+1)
Line 5: variable increment = 1
Line 9: variable assignation = 1
Given that the instructions that execute themselves the most amount of times and both for cycles, first one executing itself $(n+1)$ times, the second one $(m+1)$, the barometer is inside these cycles, and the analysis for the order $O(f(n))$ would be the following:

$$O(f(n)) = O((n+1)*(m+1)) =$$

$$O((n*m)+n+m+1) =$$

Removing the constants

$$O((n*m)+n+m) =$$

Finally, by applying the Maximum Order Law

$$O(n*m)$$

IV. Experiments

A. Experiment 1: Initial Population

The first experiment made was to determine the overall best value for the initial population for the genetic algorithm. After performing a series of tryouts, the initial population was set to 100 individuals. This is given that with such population, and after nearly 1000 generations, the algorithm will almost achieve to recreate the target image in approximately 35 minutes. If the population is increased, the algorithm will take less generations to approximate the image, nonetheless, it will consume more time and memory to handle bigger populations. The results can be seen in the table below, with a fitness of 982844, the image was almost the same as the target.

Generation	Fitness
10	956408
100	960468
300	967976
500	973682
1000	982844

B. Experiment 2: Percentage Crossover Rate

After the first experiment made to determine an optimum initial population, an experiment was made to determine a good percentage crossover rate. The crossover rate, as explained in the genetic algorithm, is used to determine the the rate in which two different individuals (images/genes) will be mixed to produce an offspring individual that will go through to the next generation. The experiment showed results stating that the higher the percentage crossover rate, the better for the algorithm. A good percentage crossover rate ranges between 80-90%, depending on the user and the need to solve the problem faster.

Generation	Fitness
10	956124
100	959030
300	964576
500	969570
1000	977874

C. Experiment 3: Probability of Mutation

The third experiment made was to determine a good probability of mutation for the algorithm. The best values determined for the probability of mutation are those ranging between 0-15%, since the lower the values, the less random the production of newer generations.

Generation	Fitness
10	956046
100	957036
300	958926
500	960632
1000	964952

D. Experiment 4: Less Fit Individuals

The last experiment made was to determine an adequate value for the less fit individuals in the population. Once again, the lower the value, the best for the genetic algorithm. This means that with lower values, the individuals with the least fitness will not go through to the next generation.

Generation	Fitness
10	956014
100	956348
300	957554
500	958804
1000	961082

V. Results and Discussion

The overall results obtained from the experiments made to the genetic algorithm show that there is a specific type of combination of parameters to obtain the target image in a relative low time. With a high crossover rate, a medium-high initial population, combined with a low-null mutation rate and a low acceptance of less fit individuals, will result in a genetic algorithm running with optimal conditions for the creation of the image. The values determined for such parameters were:

1) Initial Population = [100, 200]
2) Crossover Rate = [80, 90]
3) Mutation Rate = [0, 15]
4) Less Fit Individual Acceptance = [0, 15]

With these values entered by parameters to the genetic algorithm, it will run and generate the target image, or at least begin to approximate the image, in 1000 generations or less. This process can be seen in figures (2) to (8), with the process showed after x amount of generations, stated in the description of each figure.

Regarding the distance algorithms, it was determined that the Manhattan Distance algorithm and Euclidean Distance algorithm are similar, though not completely the same. The fitness results with the Manhattan Distance grow in a linear way, as well as with the Euclidean Distance. As well, it was determined that with the ”AL Distance” algorithm, since it was adapted to return a fitness value $fitness^2$, it will grow in an exponential way, thus increasing the probability of the individual to pass on its genetic information.

Given that there are so many algorithms to calculate a distance, and they all work in a relatively similar way, another analysis could be made to determine if there is another better algorithm to determine distances between pixel color values. There were issues in getting these values and making the comparison between the individual color values and the target image color values. Nonetheless, the comparison was made by evaluating the values pixel by pixel, and obtaining their RGB component.

Fig. 2. Target image

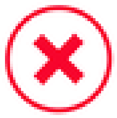


Fig. 3. First generation

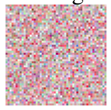


Fig. 4. Result after 10 generations

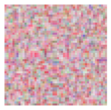


Fig. 5. Result after 100 generations

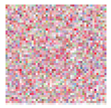


Fig. 6. Result after 300 generations

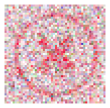


Fig. 7. Result after 500 generations

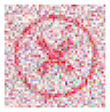
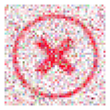


Fig. 8. Result after 1000 generations



VI. Conclusions

The conclusions of the program and the experiments show that there is a combination of parameters that work best and improve the functionality of the genetic algorithm. As well, there are other combinations or ranges of values that rather than benefiting the algorithm, they tamper with the mutation and crossover of genetic material to produce newer populations. If separated by importance, and importance being the benefit it gives the algorithm, the two most important parameters are the initial population and the crossover percentage rate; the initial population though must not be significantly high, since it will generate a cost in time running and memory of the computer.

The other two parameters could both increment or decrease the algorithms functionality. If the mutation rate is high, it will mutate more and more individuals. This means that more individuals will mutate a pixel and substitute it for another pixel with a completely random color. This implies that the higher the mutation probability, the more random the genetic process will be. Less fit individuals also generate a cost on the algorithm, since the higher the percentage entered by the user, the more individuals with lower or unacceptable fitness values will be able to pass down their genetic information to newer generations.

In terms of the distance algorithms, it was concluded that the Euclidean Distance and Manhattan Distance work relatively the same and give similar values. The own algorithms gave values that suggest an improvement, given that they return values in an exponential way, implying that individuals with a higher fitness will be more likely to be chosen to perform crossovers and pass down their genetic information.

VII. Future Work

The analysis of the algorithms could be improved to benefit the functionality of the distance algorithms. The genetic algorithm could be modified to be able to always chose the most fit "parents" of the population, thus leaving the less fit individuals behind and improving the results; a probability distribution could be made to solve this, however, the "randomness" of the algorithm could be lost.

Another option could be to try and implement different distance algorithms to verify if there is another one in existence that could work better. As well, the comparison of pixel RGB values could be improved, given that the RGB values work with the bit information of each color component, and the comparison is not always easy to do.

References

[1] Shiffman, D. (2016). 9.2: Genetic Algorithm: How it works - The Nature of Code. [online] YouTube. Available at: https://www.youtube.com/watch?v=RxTfc4JLYKs&t=1246s [Accessed 15 Sep. 2018].

[2] M. Sonka, R. Boyle and V. Hlavac, Image processing, analysis, and machine vision, 4th ed. [Andover]: Cengage Learning, 2015.

[3] S. Deepa and S. Sivanandam, Introduction to genetic algorithms, 1st ed. Berlin: Springer, 2010.

[4] G. Brassard and P. Bratley, Fundamentals of algorithmics, 1st ed. Pearson, 1995.

[5] E. Krause, Taxicab geometry, 1st ed. Dover Publications, 1986.