Intro to Ai Assignment 2 - Report

Mahmoud Mohamed

BS19-01

Abstract

The applications of artificial intelligence in art is an emerging research topic. In particular, evolutionary algorithms have shown promising results in various artistic fields. In this paper we examine the use of simulated evolution in visual art. A modified genetic algorithm was used in order to produce a picture similar to the input in the form of a voronoi painting. The algorithm was implemented in python and tested on various input images. The resulting pictures were recognizable for the intended evolution of the fitness function used which is minimizing the visual difference between the input and the output.

Keywords: Genetic algorithm, chromosome, genome, fitness, population, evolution, mutation, voronoi diagram.

Introduction

A genetic algorithm is a type of search and optimization based on biological concepts. The population represents the pool of possible solutions and they evolve by mutation and crossover. During each iteration, a part of the population is selected (usually the best fit) to survive. The survivors then mutate and breed to produce a new generation of chromosomes. The previous steps are repeated until the solutions converge to the desired output.

One of the trickiest parts of applying genetic algorithms in digital art is choosing the fitness function. This is mainly because art is highly subjective which makes it hard to evaluate numerically. The approach that was taken here is to evaluate the difference from the desired image. And in order to compensate for the loss of evaluating artistic values in the fitness function, voronoi diagrams were used to add a sense of art with its uniformity mixed with the randomness offered by the genetic algorithm.

Evolution of the Evolution

Before arriving at the current version of the algorithm, there were a lot of different ideas of implementation. The first and most naive was to use polygons or lines as building blocks. After experimenting with it for a while, it was realized that the potential of such a method is limited and the results weren't satisfactory. The main problem with that method was the difficulty of

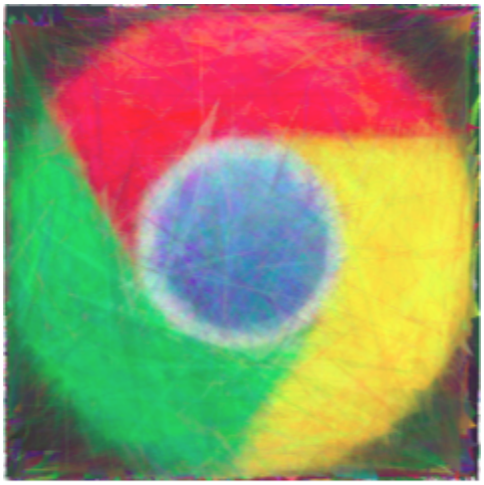


Fig.1 - Chrome using 1st algorithm (triangles)

finding an efficient blending function. There are a lot of ways of blending colors together, but to merge colors in such a way that gets the average of both parents, transparency must be used which makes the image lose its sharp colors after a lot of generations.

As a solution to this problem emerged the idea of using a combination of non-intersecting polygons, each having their own color. The most appropriate representation of such an idea was voronoi diagrams. At this point, the program utilized the aspects of a standard genetic algorithm; it started by generating a population of size ~100 chromosomes; each a voronoi diagram having ~500 points and during each iteration selection, crossover, and mutation were applied to the population to come up with the next generations. This program showed a significant improvement over the previous version. The only problem is that there was no breeding function that could make sense. The first idea that comes to mind is to randomly choose between the points of both parents but after a lot of experimentation, it was realized that it doesn't significantly improve the rate of fitness increment; instead, the mutation phase is much more important. That's when the next optimization to the algorithm was discovered.

The main idea is to completely remove the breeding phase, and only mutate the population which saves a lot of time that can be efficiently used in creating more generations and increasing the number of points in each diagram.

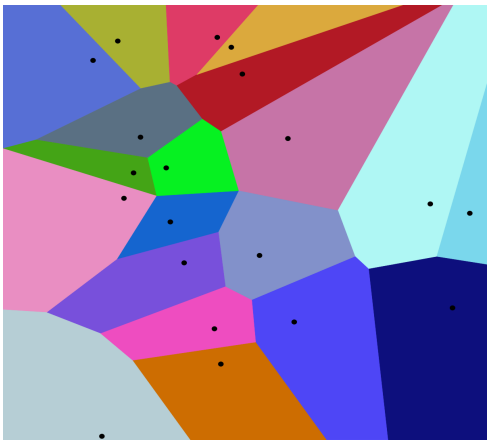
Fig. 2 - Chrome using final algorithm (100K'th generation)

The final optimization was to reduce the population size to 1. It came up after a lot of experimentation that population size doesn't matter a lot without crossover. So we can mutate only 1 genome multiple times and choose the best of them for the next generation.

Algorithm

The algorithm was written using python 3.8. The used libraries were numpy, random, datetime, scipy, and opencv (cv2).
All images were read using cv2 and drawn using numpy arrays and voronoi diagrams from scipy.

Voronoi Diagrams



Voronoi diagram is a mathematical notation of a plane which is partitioned to regions (or cells) each of which is related to a single point (which in our case is represented by *ColoredPoint* class). Each pixel is in the region of the point that is closest to it and takes this point's color. The polygons do not intersect as one pixel can't be in more than one region at a time.
In the program we use scipy library to calculate the vertices of the voronoi cells and convert them into polygons on an actual image using cv2 library.

Testing the algorithm

To run the algorithm, you need python 3.8 and pip installed. You can install required libraries by executing the command `pip install -r "requirements.txt"`. To quickly run the program execute the command `python3 main.py`. The program is initially configured to run on a sample image "`sample00.png`" which is located in the folder "`testingImages`" and produce output in the "`output`" folder. To run the program on another image you can change the value of the variable "`targetName`" at the beginning of "`main.py`" file. Multiple other samples can be found inside the folder "`testingImages`".

The output in the terminal consists of the name of the image, the current generation number, and the respective fitness and it's written every 100 generations. The other part of the output is an image produced every 1000 generations which is the drawing of the voronoi diagram of the current generation's genome.

Representation

The **solutions** (**genomes/chromosomes**) were represented as a *VoronoiDiagram* class which is made of an array of ~1300 *ColoredPoint* (the number was manually chosen after a lot of trial and error) where each point has x, y coordinates and a color.
The choice of the number of points in each diagram is a tradeoff. Higher number of points means higher accuracy but increases the running time of the algorithm. So after experimenting on a number of 512x512 images, it was concluded that diagrams with a number of points 1150-1350 will lead to equally good results in a relatively low run time.

In our case, the *ColoredPoint* class represents a **gene** and a **mutation** is a random change in the *ColoredPoint* class.

Evolution

The algorithm starts with the initially (randomly generated) population of size 1. During each iteration, the individual is mutated exactly twice and the best one out of the 3 genomes is selected depending on their fitness score.
This process is repeated roughly 150,000-170,000 times until the solution converges to the target image.

Population

As mentioned before, the decided population size in the final version of the algorithm was only 1 diagram. The comparison between this version and previous versions revealed the superiority of this version. The main reason is that the run time grows exponentially with respect to the population size and doesn't significantly improve the images because there is no crossover phase so a population of size 1 or 2 is very sufficient in this case.

Mutation

The mutation phase is applied in each iteration before the selection. During it, one random *ColoredPoint* is chosen from the diagram, and either the color is changed or the coordinates are changed (equal probability). In the color-changing mutation, each value of the three red, green, blue is changed slightly and independently by a random integer between -25, 25.
In the coordinates-changing mutation, x and y values of the point are changed slightly and independently by a random integer between -15, 15

Crossover

Even though a crossover function was not used in the final algorithm, one was implemented and used during earlier stages. The child of 2 parents was a *VoronoiDiagram* class that has approximately half of its points from each parent all randomly chosen.

Fitness

The fitness function used was the mean square error function. It takes two images and prints the sum of the squared difference between their pixels divided by the area of the image. Note that both images have to be of the same size and that the goal of the algorithm was to minimize this function.

Selection

The selection process is simple. During each generation, 2 copies of the original genome are created and subjected to a random mutation. The genome with the best fitness out of the three (the 2 copies and the original) is then selected to survive to the next generation.

What art is to me

There is not one specific agreed definition of what constitutes art. Some people say that art is a way for them to escape reality and be one with their imaginations. Others define it as something that stimulates thoughts and emotions through a wide variety of senses while *Deidara always said that “art is an explosion”. However, what I am most certain of is that if everything was considered art, then there wouldn’t be such a thing as “art”.
That’s why, for me, what comprises art are two very important characteristics:

1. Genuinity. The honest expression of emotions and thoughts must appear in any work of art. It gives the work its sense of meaning and even if it’s purpose is not to deliver any kind of meaningful message, it enables us to see through the artist.
2. Uniqueness. The unique way of expressing your inner thoughts and emotions can’t be copied. That’s why art is unique to the artist and that’s also possibly why everyone has a different sense as to what art is.

In my opinion, a lot of things are admiring about not just my work, but the use of genetic algorithms in digital art in general. It is really fascinating just thinking about how our knowledge of evolution and natural selection can be used in such a way and how a combination of random colored polygons can be assembled to create something meaningful from nothing.

What I find pretty about my work in particular is that it conveys my ideas and opinions. It simplifies the process of evolution, and steadily alleviates the randomness and irregularities of the initial voronoi diagram while embracing its differences from the target image. Afterall, if the algorithm was too focused on creating an infinitely similar duplicate of the image, it wouldn’t be considered art but a mere copy of the input.

*Deidara: A fictional character from the anime “Naruto Shippuden”. His power was creating explosive figures out of clay which can detonate at his command. Deidara always considered his bombs as art and his purpose was generating one giant masterpiece to be remembered by. One of his famous quotes was: “Art is something that blossoms for an instant before withering away. Art is beauty that lasts for just a moment. To me, the essence of art is - an explosion!”

Example Images

Fig. 3 - Deidara (Original)

Fig. 4 - Deidara (166K’th generation)

Fig. 5 - Killua (Original)

Fig. 6 - Killua (163K'th generation)

Fig. 7 - pepeSad (Original)

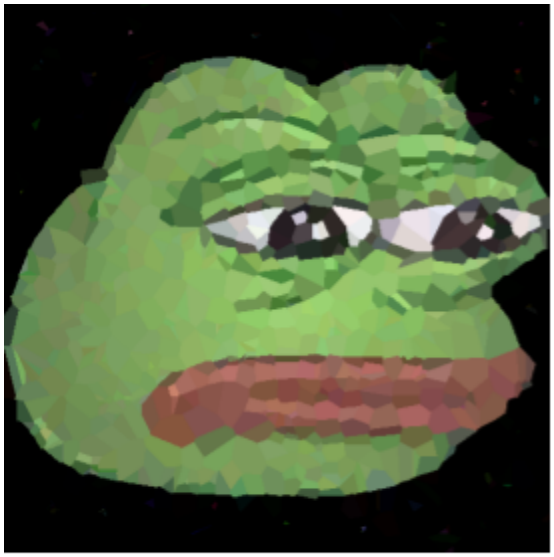


Fig. 8 - pepeSad (88K'th generation)

Fig. 9 - chrome (Original)

Fig. 10 - chrome (100K'th generation)