V.N. Markov

CREATING ART IMAGES VIA THE DELAUNAY TRIANGULATION METHOD

(Innopolis University)

Introduction

Currently, the field of digital image processing is actively developing all over the world and it is one of the promising technologies in creating video surveillance systems, technical and medical diagnostic systems, development of digital photography, as well as remote monitoring of various processes [1,2]. The issues of effective functioning of image processing systems are relevant in many researches of scientists. The article presents one of the possible ways of computer processing of art images using the Delaunay triangulation method.

The program uses a genetic algorithm to draw a new picture based on a given one. Each image is constructed by triangulation and colorization of N points. You can see the examples of the program output below (figure 1):

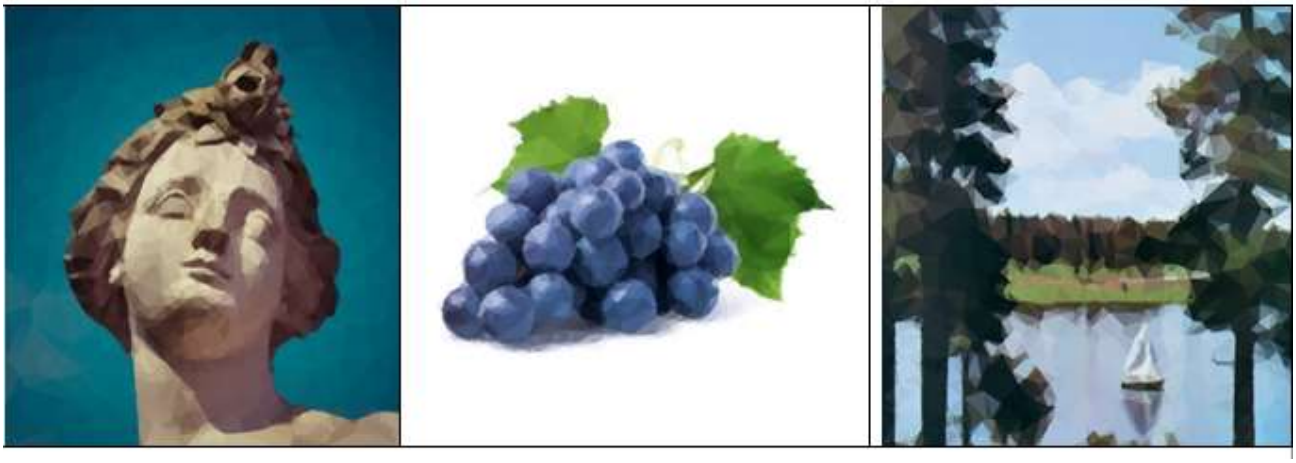

Fig. 1. Examples of art images after using the Delaunay triangulation method

Algorithm description

Chromosome representation.

Each bot has one chromosome. Each chromosome consists of N pairs of natural numbers in the range [0; RAND_MAX]. Each pair represents coordinates of points on the image plane.

Estimation of the fitness.

To estimate the fitness of bots we need to pass each bot through the pipeline consisting of several stages (figure 2).

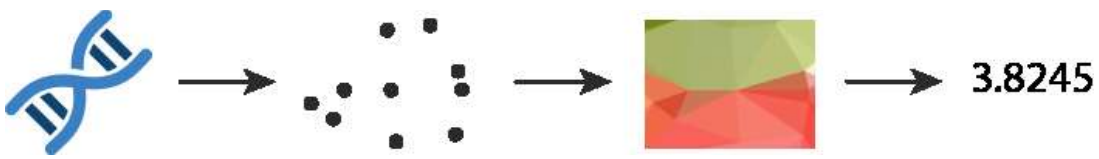

Fig. 2. Stages of bot «compilation»

The first stage is bot "compilation". At this stage, we extract information from the bot's chromosomes and transform it into collections of points. Basically, it is just a conversion and normalization of a one-dimension chromosome to get a 2d array of coordinates. The second stage is image rendering. At this step, we produce images with colored triangles based on points from the previous stage. We will take a closer look at this stage further. The last stage is loss estimation: we compare the rendered image with the target one and calculate the number that will show how similar the images are. For this, we use the standard L1 function. Also, in cases where small errors do not matter, it is better to use the L2 loss function (figure 3):

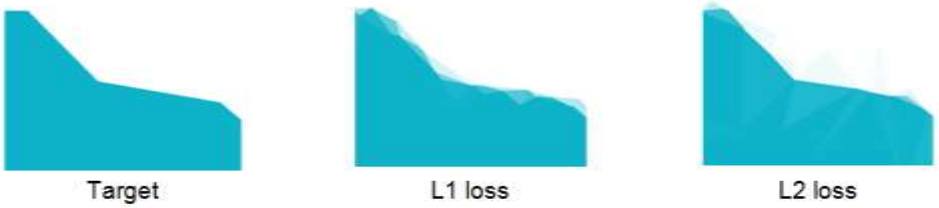

Fig. 3. Loss estimation

Sometimes some parts of images are more important than other ones. For instance, the background is not as important as the objects in the foreground. We decided to use grayscale masks to tell the program if there is any special place it should pay attention to. In this case, a pixel-wise error will be multiplied by mask, and only after that errors will be summed up. Here you can see the difference between using the mask and not using it (figure 4):

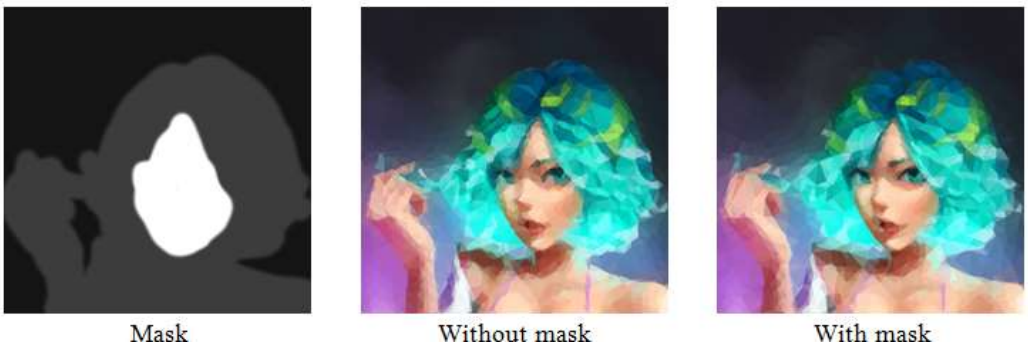

Fig. 4. Using the masks

Rendering the image.

To build and render an image from a given set of points we need to pass these points and the target image through the pipeline of three stages: triangulation, colorization, render. In the first stage, we apply Delaunay triangulation to a given set of points to obtain triangles. In the next stage, we use these triangles and the target image to estimate the color of each triangle. The color of a triangle is the mean of all pixels this triangle covers. In the last stage, we render all obtained triangles with computed colors (figure 5).

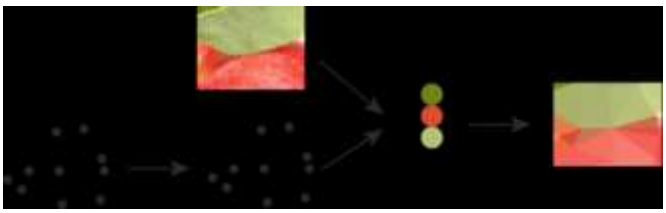

Fig. 5. Stages of rendering the image

Population size

To determine the appropriate population size M we run the program several times with different population sizes and such amount of iterations so that overall execution time will be the same for all tests. We run the program 3 times for each population size with different seeds and take the mean of results.

Selection technique

The evaluation process assigns each bots label with the score. Then all bots are sorted by an assigned score from the best bot to the worst one. Then there is a choice of 2*M partners for crossing and creating a new generation.

Evolution process (mutation, crossover)

When two bots are selected for producing offspring, we cross these bots to create a new bot. The chromosome of a new bot is produced by crossing the parent's chromosomes. First, we apply uniform crossover ("uniform" because it is simpler to implement and it is acceptable for our task). Second, we add some mutations to already created new bots. Let us notice that in the case of crossover one point (pair of coordinates) is counted as a single element but in the case of mutation, they are considered as two elements. It is so because we do not want the crossover to take y coordinate of one point and concatenate it with x coordinate of another point (because these points are not connected in any way), or even with y coordinate of another point. But we want to have the ability to change x and y by mutation separately.

Conclusion

To improve the performance of the algorithm, we write a multithreading program on C++. Then we notice that colorization in the rendering pipeline is the slowest part of the whole program, and we pay a lot of effort into increasing the speed of this part. If we compare our implementation with the same algorithm but written on Python, our program is around 80 times more efficient (we try to implement this algorithm on Python as well).

References

1. Ilyichev V. Y. Studying the application of Delaunay triangulation method with CSV files for image formation in Python // Notes of a Scientist. – 2021. – №. 9-1. – P. 301-305

2. Lawonn, K. and Günther, T. (2019), Stylized Image Triangulation. Computer Graphics Forum, 38: 221-234. https://doi.org/10.1111/cgf.13526