ASSIGNMENT II

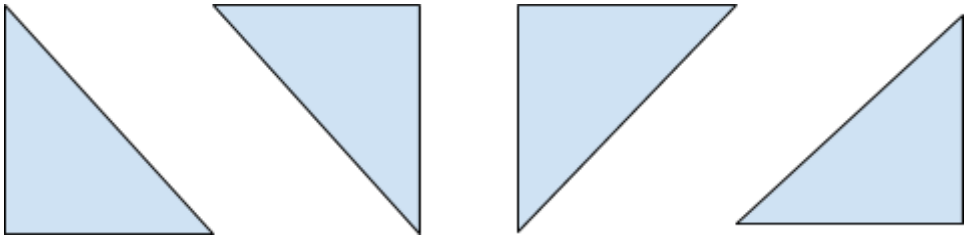
How to run

1. Install python and pip
2. Install the required libraries from requirements.txt [see the requirements folder]
    a. If you have problems with pycairo, install the version of the pycairo package (which will run on your system) found in the same folder.
3. Setup the evolution algorithm properties in the first few lines of the script
4. Run the script which corresponding to your OS [windows/linux]
    a. A prompt to choose a file should appear
    b. Messages about 16 parts of the images being processed will appear
    c. The output image will open on its own
    d. The ‘result.png’ will be saved in the same folder where the script is located

Basic evolution strategy

Reconstruction is attempted using (different) right triangles with equal sides;

As a result, there are a total of 4 possible positions for such a triangle:



As for the colour of each triangle, there two ways for it to be chosen:

[the mode can be specified by the user]

1. Letting each tringle slowly evolve its colour over many generations along with the proper position/size.

    [takes **A LOT** of generation to get good-looking output]

2. Allowing the algorithm to cheat a bit by getting the colour directly from the input image.

    [the meaningful output is obtained in the first few generations]

Genome

- The DNA for each creature is encoded as an array containing a fixed amount of ‘triangles’ each carrying information about its size (all 3 points) and colour in the (RGBA) format [the number of triangles can be changed by the user]
- The resulting genome looks something like this:

  [{ [point 1, point 2, point 3], [R, G, B] }, { [point 1, point 2, point 3], [R, G, B] }, …]
- The function that creates such an ‘attribute’ can vary the triangle’s side size in the range [1,20] and the colour for each of the [R G B A] values in %.
- Since the genome is a bit ‘*complex*’ it takes **A BIG NUMBER** of generations to get some meaningful output

  [only if the creatures have to evolve both the colour and the size/position of each triangle].

Fitness function

Fitness is evaluated as the 'Mean Squared Error' between the two images. It is basically a sum of the squared difference between each pixel of the image. Each individual in the population tries it’s best to minimise it.

Artisticness of the output

I believe that the output can be considered artistic because the output image is made up out of small right triangles, which, in my opinion, makes it look like it was repainted using very crude brush strokes. So from some distance, it looks as if it is almost the same image, but if one zooms in, it will be obvious that the image is made from triangles, not pixels.

Additional note

It was tuned to best work with 512x512 images. Other sizes will give bad results or crash the system.

The ‘deap’ library is used solely for cool statistics and generation output. All the evolution-related functions are defined by me.

There are two versions of the genetic algorithm script available, which produce slightly different results (Windows version has a grid on the image). The reason being that NumPy is somehow unable to process images bigger than 256x256 on Windows. As a result, the images have to be broken down into smaller ones and processed separately. There is no such problem on Linux. So, the images can be processed as one

Sample output can be seen throughout the rest of the report