INTRO

Intelligent Art Generation

Ada Godzisz
Mateusz Guściora
Michalina Kulas

Agenda

- ★ Goals and Motivation
- ★ Little Theory
- ★ Literature Review of Methods
- ★ Our chosen Approach, Method, Objects and Tools
- ★ References

GOALS AND MOTIVATION

GOAL:

Creating images of artistic merit along with experimenting with parameters to improve (subjectively) those images.

MOTIVATION:

Learning about the possibilities of artificial intelligence algorithms and applying them to generate images and create art.

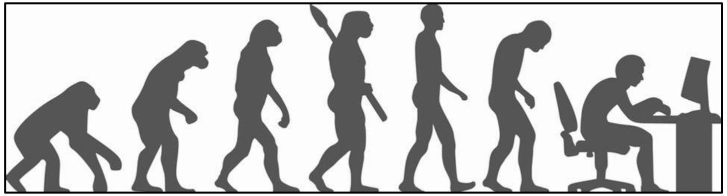
PAPER OVERVIEW, METHODS AND ALGORITHMS OVERVIEW

Evolutionary algorithm

- ★ Evolutionary algorithm (EA) is a metaheuristic optimization algorithm. The idea is inspired on a biological evolution (reproduction, mutation, recombination, and selection.) [1]

Types:

- ★ **Genetic algorithm (GA)**
- ★ Genetic programming (GP)
- ★ Evolutionary programming
- ★ Evolution strategy
- ★ Differential evolution
- ★ Neuroevolution
- ★ Learning classifier system



Genetic Algorithm (GA) – general idea

- Initial population
- Chromosome (also Genotype)
- Fitness function calculation
- Genetic operators: selection, crossover, mutation
- Generations and End evolution

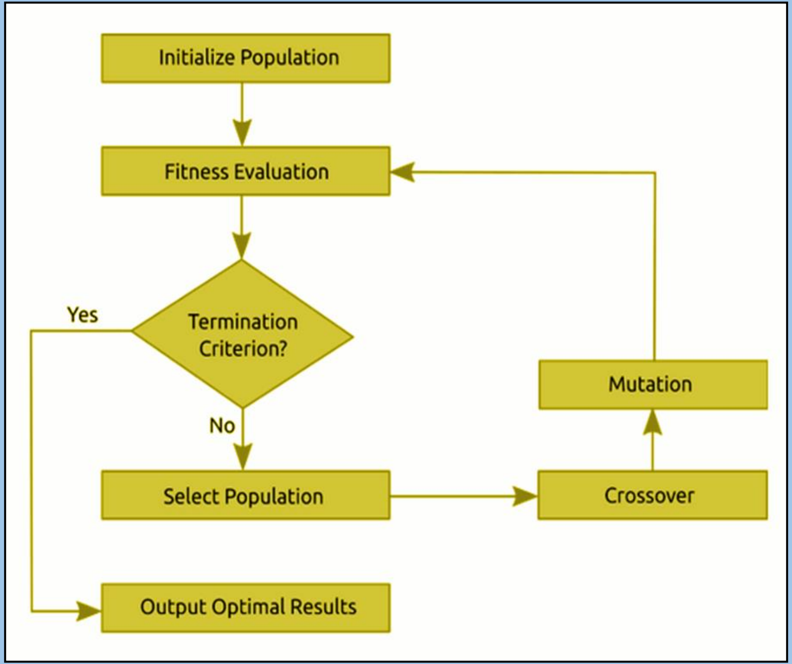
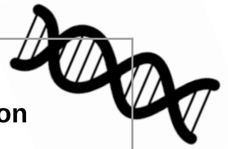


Figure 1. Flowchart illustrating the working of the GA[2]

Approaches comparison



Number of Source	Data Representation	Initial Population	Fitness Calculation	Parent Selection	Crossover (opc)	Mutation (opc)	End of evolution
1)	Chromosome = 1D row vector of pixels	Randomly generated image with RGB noise	sum(target img)-mean[abs(target img-current img)]	Half of the solutions with the highest fitness value	Single-point at the 1D row vector	Random change the value for 1% of genes	Generation 15 000
2)	Chromosome = 120 brush strokes	Randomly generated, 128 members	1-abs(target img-current img)	fit=1 to good solutions, fit=0 to bad, take fit=1 solutions	50:50 brush strokes from parents	Random change of the brush stroke, 1,5%	After executing the set number of brush strokes
3)	Genetic information 10 bytes Individual triangle data (ARGBX1Y1X2..)	Each byte randomized (0-255) from genetic information	Comparison target photo from the solution *Scaled fitness	Most fit individual from the last generation of the run	swaps every bit randomly chosen	bite mutate byte mutate triangle swap	Set number by user (~ 10 000-20 000)
4)	Chromosome = quarter of whole tile pattern 32x32 pixels	Uniformly selected in the range of 0-255 = grayscale	Cost function = Sum of dif of the values to the neighbour pixels	Ranked candidates based on Cost function. Minimal wins	Not applied	Assigning a random value to a random point	Generation 15 000

1) Reproducing Images using a GA with Python

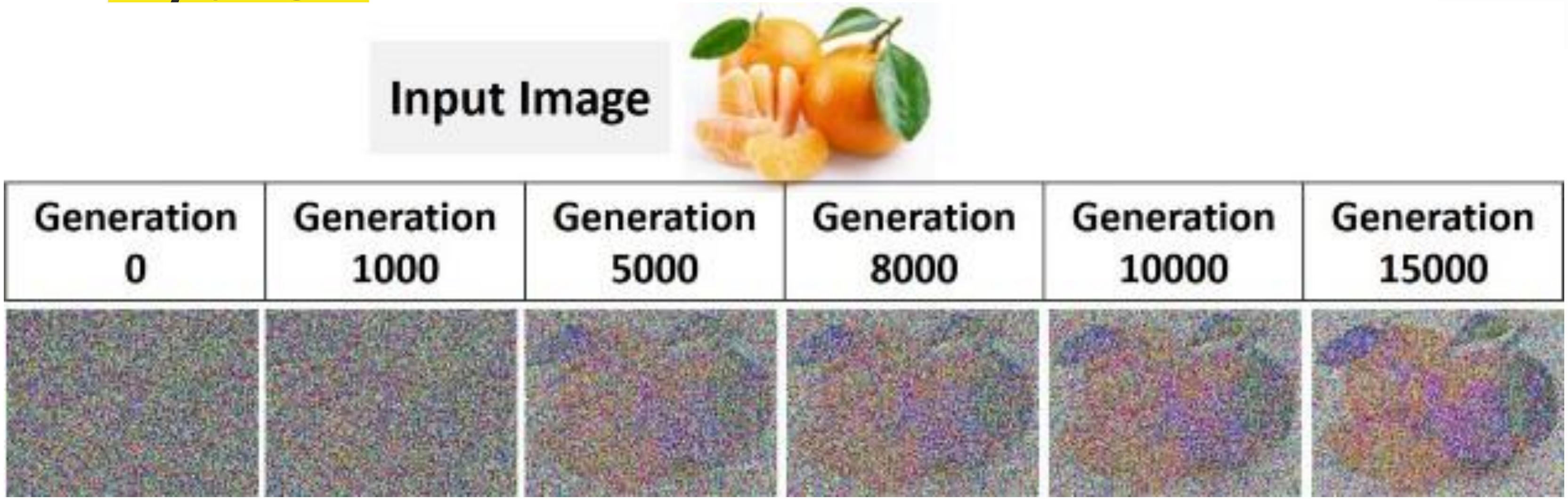


Figure 2. The progress of the algorithm when applied to an RGB image.[3]

2) Procedural Paintings with Genetic Evolution Algorithm

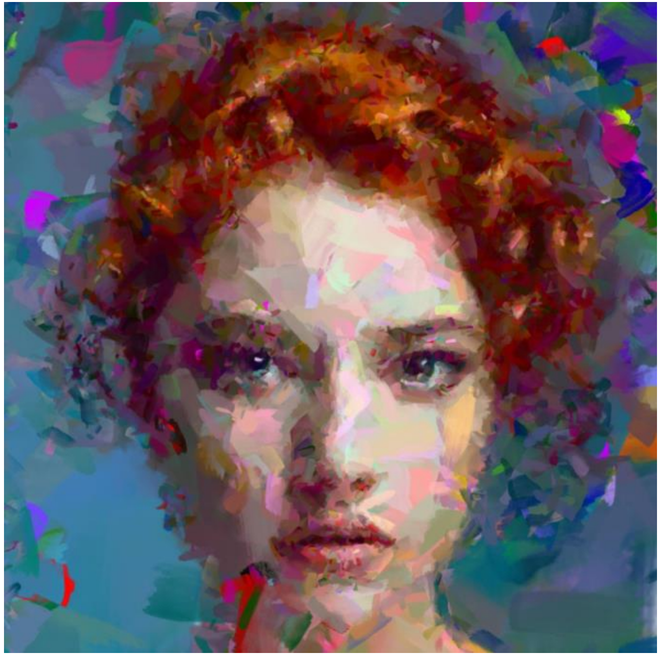


Figure 3, Figure 4: examples of images created with a GA using brush strokes.[4]

3) Generation of Vector-Based Graphics from Existing Bitmap Images by Means of the Genetic Algorithm [4]

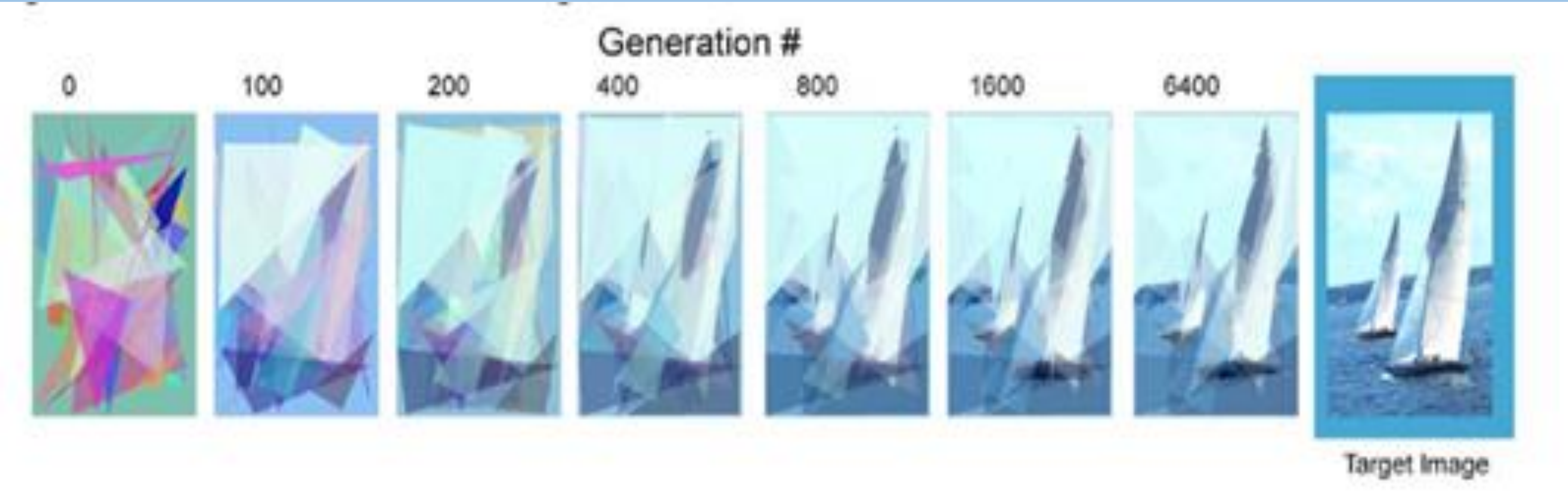
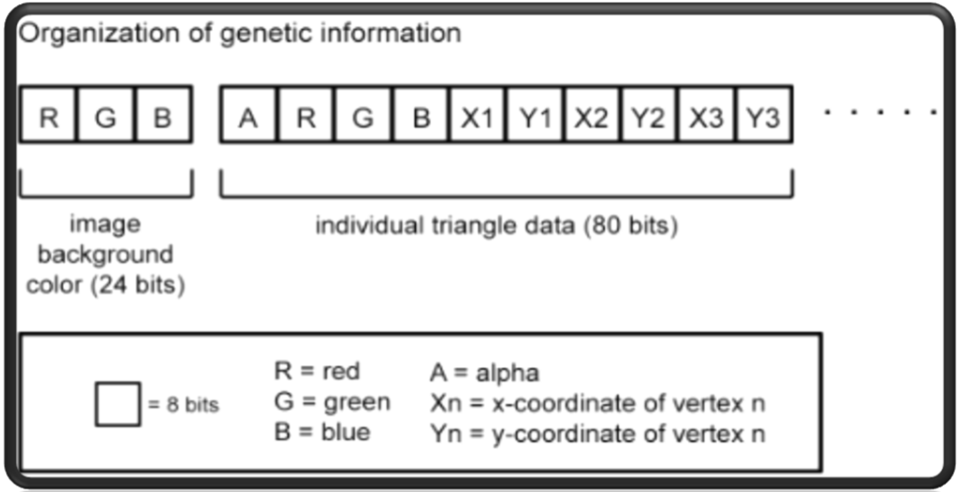


Figure 5. Example effects by generations.[5]

3) Generation of Vector-Based Graphics from Existing Bitmap Images by Means of the Genetic Algorithm [4]

- Initial population - each byte randomized (0-255) from genetic information
- Each triangle contains bytes describing shading (r,g,b and alpha) and position (x1,y1...)
- Fitness calculation and Fitness scaling



Fitness:	(255 * 255 * 3 * target_width * target_height) - sum over all x, y of : (target[x][y].red - solution[x][y].red) ^ 2 + (target[x][y].green - solution[x][y].green) ^ 2 + (target[x][y].blue - solution[x][y].blue) ^ 2
Standardized Fitness:	raw_fitness - worst_raw_fitness + ((best_raw_fitness - worst_raw_fitness) / 9)

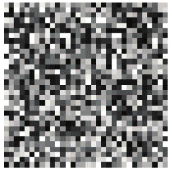
Figure 5 and 6. Organisation of genetic information and fitness function.[6]

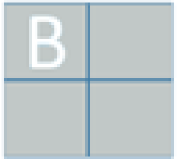
3) Generation of Vector-Based Graphics from Existing Bitmap Images by Means of the Genetic Algorithm [4]

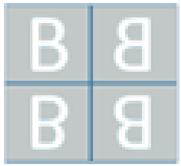
Figure 6, Figure 7: examples of generated images.[5]

4) Generating art tile patterns using genetic algorithm [5]

- ★ Symetric art tile patterns
- ★ Low resolution and grayscale patte
- ★ Initial population
- ★ Basic element
- ★ Cost function
- ★ MATLAB







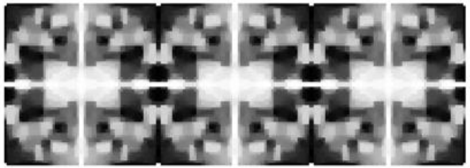
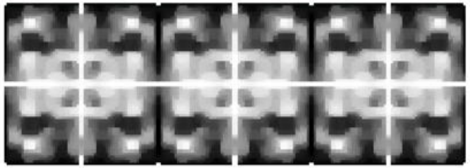
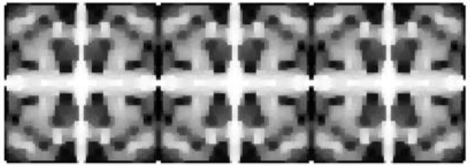
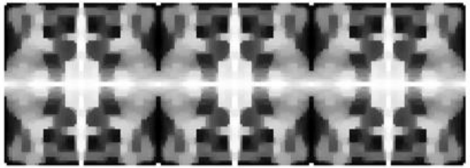
Row No.	Basic Element	In Each Row Three Tiles Is Repeated for Better Showing The Results.
1		
2		
3		
4		

Figure 6. Effects of generation art tiles.[5]

OUR APPROACH AND CHOSEN METHOD AND ALGORITHM AND TOOL

Approach

- Data representation: Chromosome: 20 figures
- Properties (Genes):

  -triangles: R,G,B, Alpha channel (values: 0-255), position: (x1,y2,x2,y2,x3,y3)

  -circles: R,G,B, Alpha channel (values: 0-255), radius r, position: (x, y)

- Initial population: Randomized (20, 50, 100 chromosome)
- End of evolution, Generation:

0	100	200	400	800	1000	5000	8000	10000	15000

Approach

- Fitness (per pixel):

1-abs(target img-current img)	(255 * 255 * 3 * target_width * target_height) - sum over all x, y of : (target[x][y].red - solution[x][y].red)^2 + (target[x][y].green - solution[x][y].green)^2 + (target[x][y].blue - solution[x][y].blue)^2	[1-abs(target img-current img)]^2

- Parent selection: half of the solutions with the highest fitness score

Approach

Genetic operators:

- ★ reproduction (20% of unchanged chromosomes)
- ★ crossover (60% x 40% exchange of data properties)
- ★ mutation:
  a. swap of two figures
  b. decreasing the surface of the figure by 90%
  c. probability: 0,5%, 1%, 1,5%

Environment - PYTHON

Python – extensive library package - good for image processing, readability and clarity

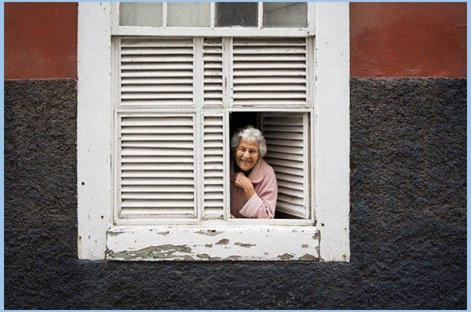
Possible libraries:

- PYGAD
- NUMPY
- TENSORFLOW
- KERAS

Chosen Object[6]

- Different background
- Different details
- Different colours
- Different set up of an image







References

[1] https://en.wikipedia.org/wiki/Evolutionary_algorithm [date access: 16.11.2021]

[2] https://skill-lync.com/student-projects/week-4-genetic-algorithm-316 [date access: 15.11.2021]

[3] Gad, A. (2019) “Reproducing Images using a Genetic Algorithm with Python” [data access: 17.11.2021]

https://heartbeat.comet.ml/reproducing-images-using-a-genetic-algorithm-with-python-91fc701ff84

[4] Shahrabi, S. (2020) “Procedural Paintings with Genetic Evolution Algorithm” [data access: 17.11.2021]

https://shahriyarshahrabi.medium.com/procedural-paintings-with-genetic-evolution-algorithm-6838a6e64703

[5] Weller, C. (2009) “Generation of Vector-Based Graphics from Existing Bitmap Images by Means of the Genetic Algorithm”

http://www-dept.cs.ucl.ac.uk/staff/w.langdon/ftp/papers/koza/sp2002/weller.pdf

[6] M. Heidarpour and S. M. Hoseini (2015) "Generating art tile patterns using genetic algorithm," *4th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS), pp. 1-4, doi: 10.1109/CFIS.2015.7391652*

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7391652

[7] Free images:

https://pixabay.com/pl/photos/ [data access: 16.11.2021]

Thank you