UNIVERSITÀ DI TRENTO

Department of Information Engineering and Computer Science

Master’s Degree in Artificial Intelligence Systems

Evolutionary Image Vectorization

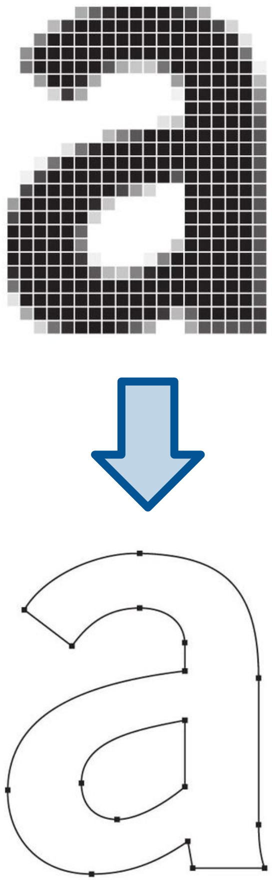
Academic year 2020/2021

Matteo Destro

What is Vectorization?

Vectorization: conversion of **raster graphics** into **vector graphics**:

- **Raster graphics**: matrix of pixels, each with a color value
- **Vector graphics**: set of points connected by lines or curves



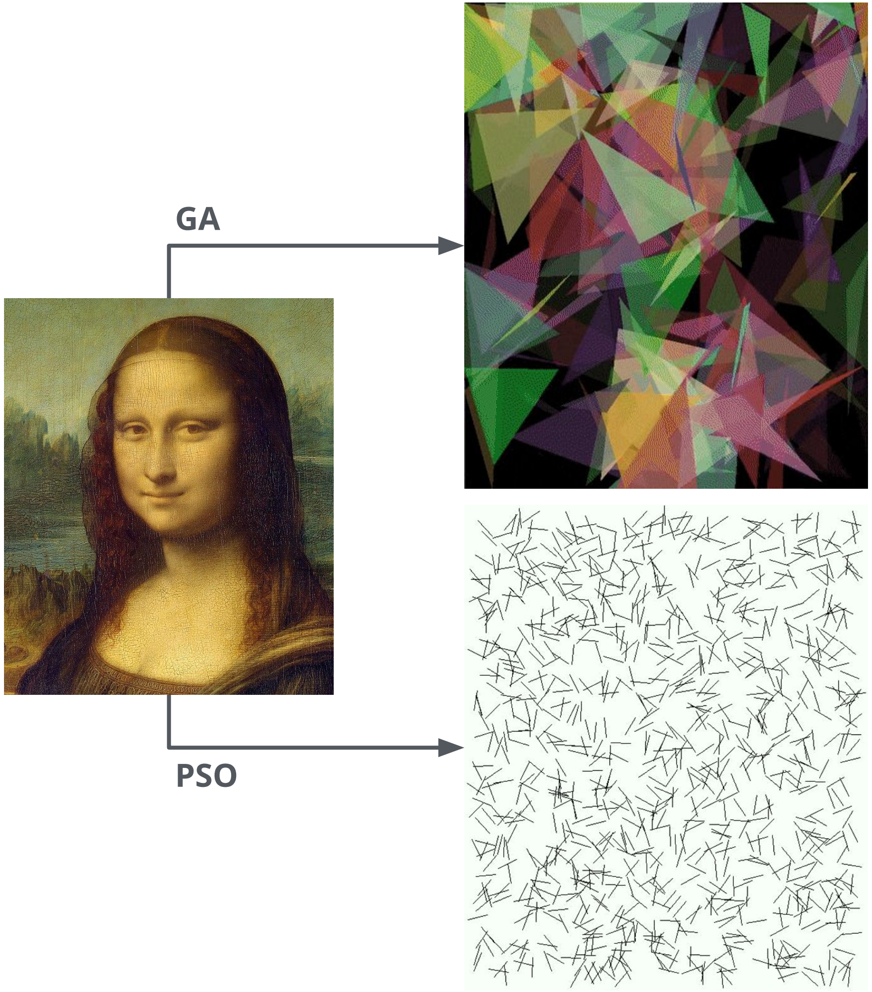
Advantages of vector graphics:

- Reduced file size
- Rescaling without any quality loss
- Can be easily edited and converted to raster graphics

Solutions

Two approaches explored:

- **Genetic Algorithm**: evolve a set of colored polygons trying to recreate the target image
- **Particle Swarm Optimization**: move a set of particles trying to reconstruct the most relevant contours



Genetic Algorithm

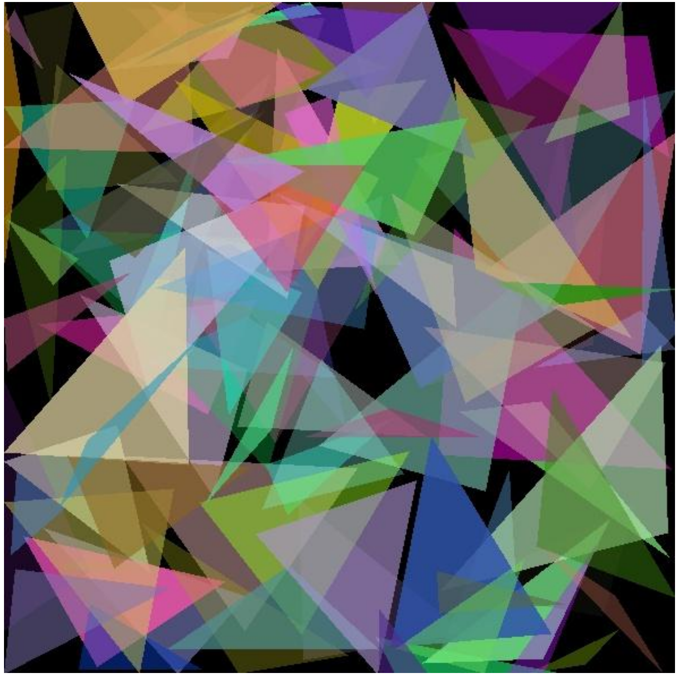
Evolve a set of polygons to reconstruct the target image:

- **Individuals**: composed by *n* polygons, each with:
    - *v* vertices
    - RGB color
    - alpha value (transparency)

- **Fitness**: *minimize* sum of squared residuals w.r.t. target image

$$f(x) = \frac{1}{WHC} \sum_{i=0}^{W} \sum_{j=0}^{H} \sum_{c=0}^{C} (x_{ijc} - I_{ijc})^2$$

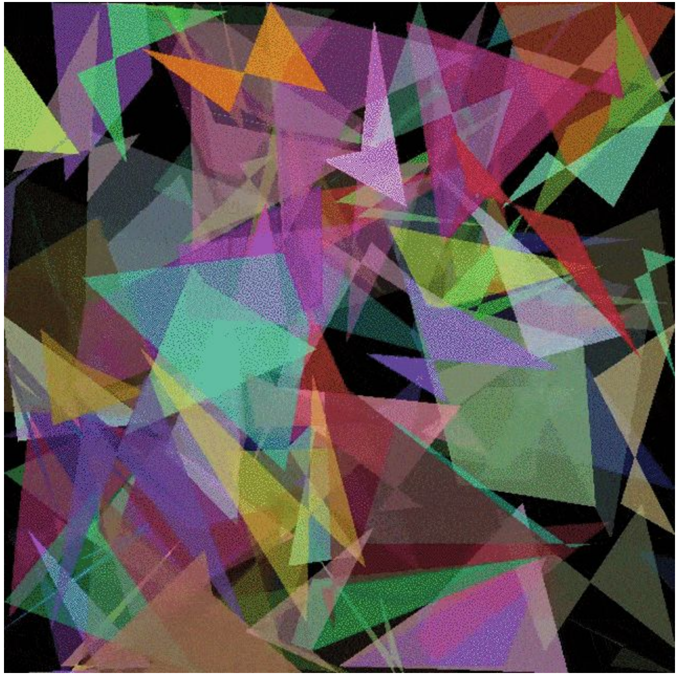
- **Initial population**:
    - random vertices (avoid large polygons)
    - random RGB color in [0, 255]
    - random alpha value in [20, 220]



Polygons of an individual drawn over a black canvas

Genetic Algorithm

- **Parent selection**: *roulette-wheel*, *rank-based*, *truncated rank-based* or *tournament selection*
- **Crossover**: *one point*, *uniform* or *arithmetic*
- **Mutation**: Gaussian mutation, using either:
  - 3 step-sizes for vertices coordinates, color channels and alpha value
  - *Evolution Strategies* with `(2*#vertex + #channels + 1)` self-adaptive mutation step-sizes
- **Replacement strategy**: *(μ, λ)*, *(μ + λ)* or *crowding replacement*

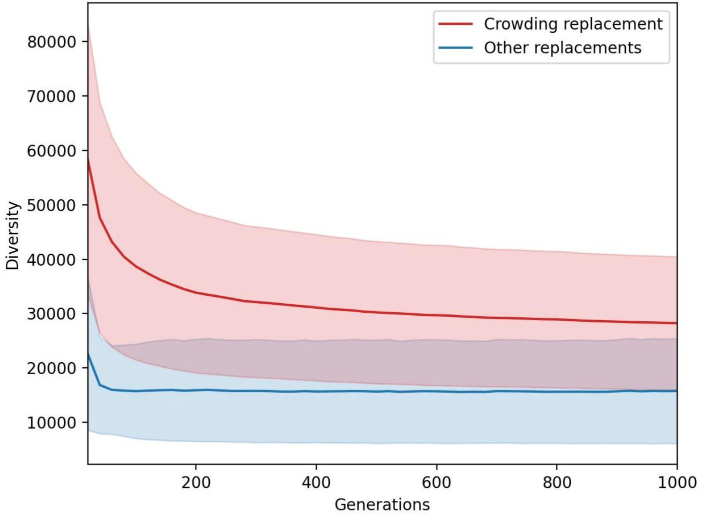


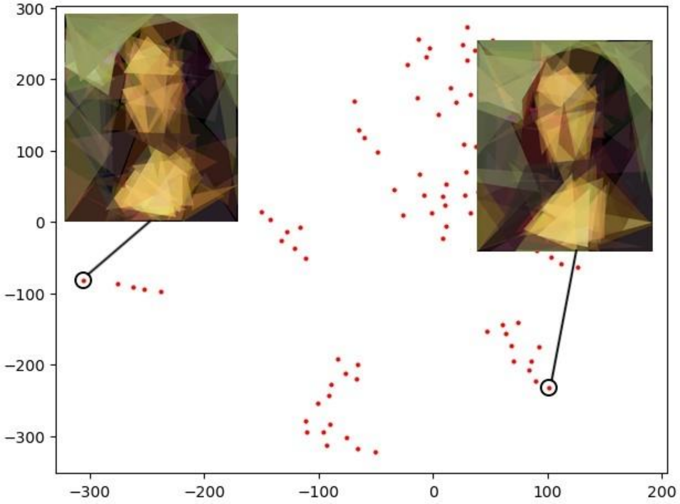
Polygons of an individual drawn over a black canvas

Crowding Replacement

Diversity preservation mechanism. For each offspring:

1. Sample **random pool** of individuals from current population
2. Take the individual in the pool **closer** to the offspring
3. **Replace** that individual with the offspring if the offspring fitness is better

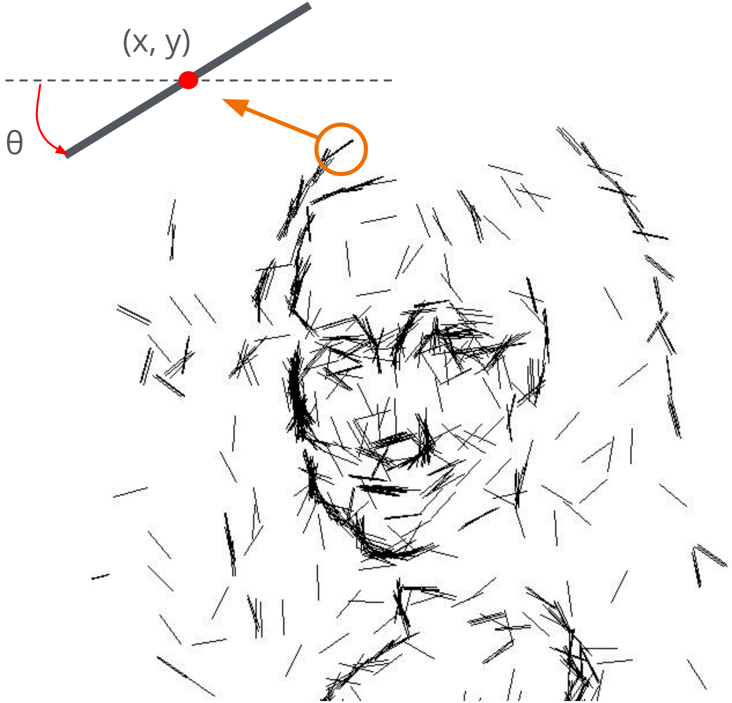




Particle Swarm Optimization

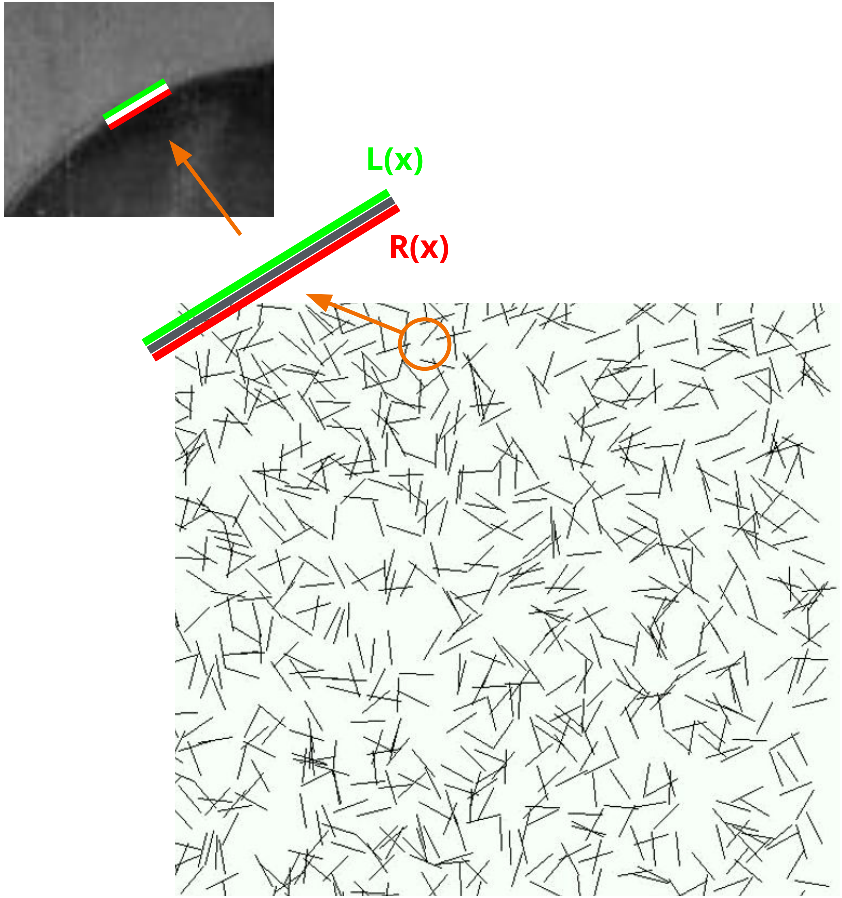
Move a set of particles trying to reconstruct the most relevant **contours** of the target image (*edge detection*):

- **Particles**: represented by three values:
  - x, y coordinates
  - θ rotation angle
- **Neighborhood**: *distance-based* or *list-based* (*ring* or *star*)
- **Velocity update**: *standard*, *Fully-informed* or *Comprehensive Learning*



Particle Swarm Optimization

- **Fitness**: *maximize* particle's gradient w.r.t. position and rotation: $$f(x) = \left| \sum_{p_R \in R(x)} I(p_R) - \sum_{p_L \in L(x)} I(p_L) \right|$$
- **Velocity clamping**: avoid large velocity values
- **Separation rule**: maintain a minimum distance between particles: $$v_i = v_i - \sum_{p \in S_{d_{\min}};\, p \neq i} (x_p.\text{center} - x_i.\text{center})$$

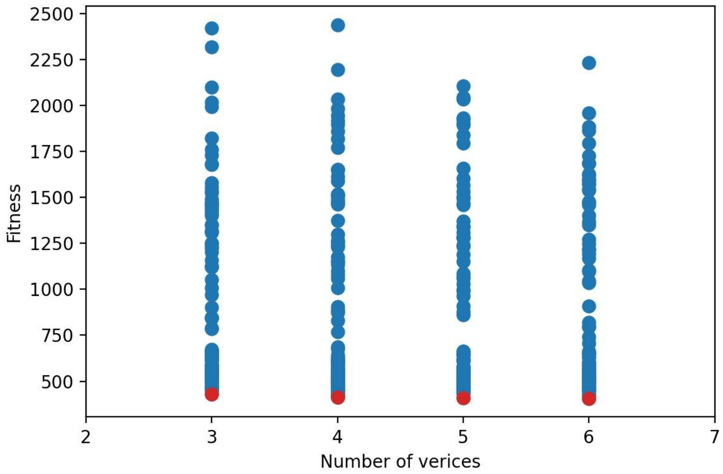


Experiments

Different combinations of hyper-parameters tested for both approaches:

- **GA**: 360 runs, 1000 generations
- **PSO**: 230 runs, 100 iterations

Note: for GA, only polygons with 3 vertices were used in the experiments, since an higher number of vertices did not bring any significant improvement.

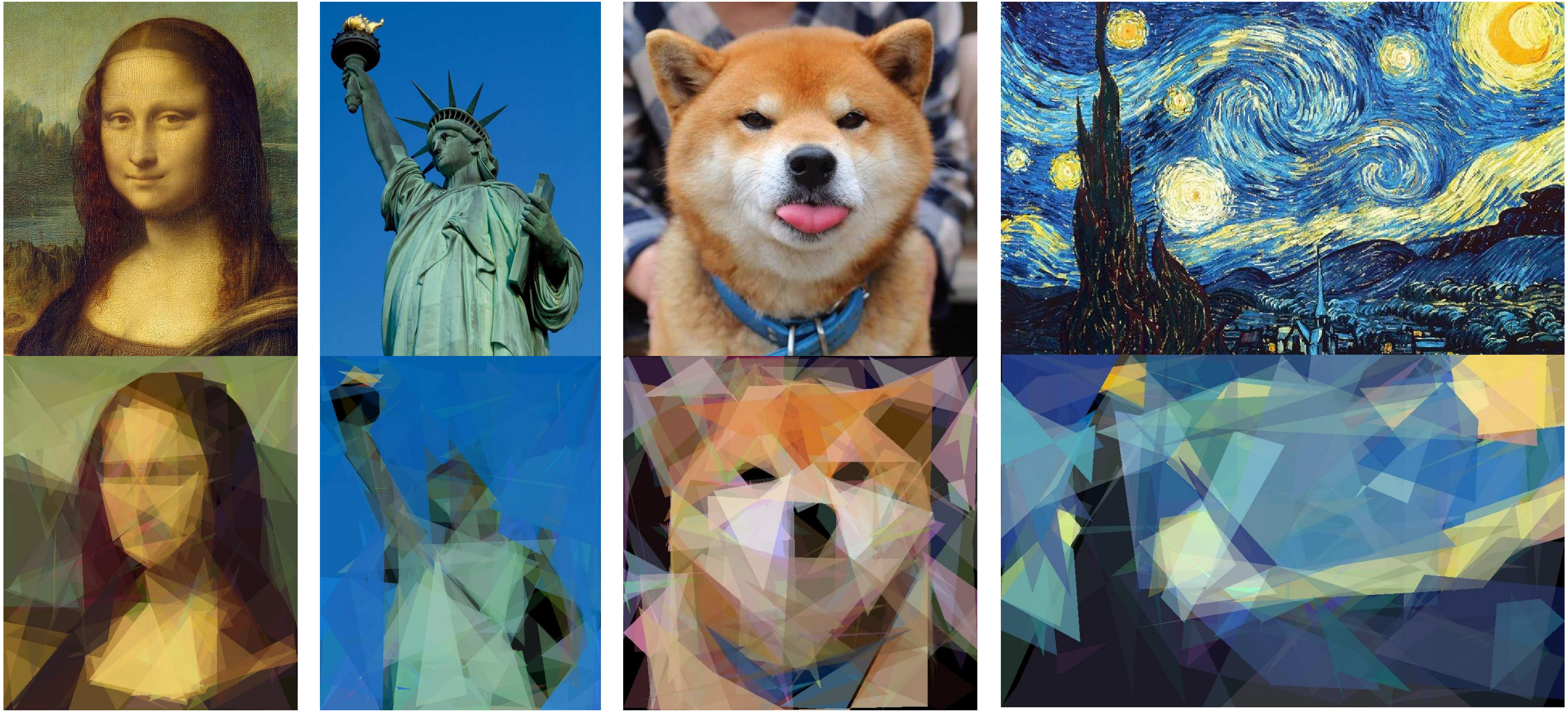


Results: Genetic Algorithm

Genetic Algorithm approach:

pop. size	num. poly	selection	replacement	crossover	self adaptive	best fitness (%)	avg fitness	std fitness
100	50	truncated(0.1)	(μ, λ)	uniform	False	422 (93%)	414.70	83.20
100	100	truncated(0.2)	crowding(2)	uniform	False	447 (93%)	425.61	49.51
100	200	tournament(10)	$(\mu + \lambda)$	uniform	False	455 (93%)	444.95	37.28
100	200	roulette-wheel	crowding(5)	uniform	False	458 (93%)	431.60	93.18
100	200	tournament(10)	(μ, λ)	uniform	True	513 (92%)	471.44	85.33
100	200	truncated(0.1)	crowding(5)	one-point	False	516 (92%)	509.82	18.51
50	100	truncated(0.1)	$(\mu + \lambda)$	uniform	True	517 (92%)	496.20	94.65
50	200	roulette-wheel	crowding(2)	uniform	False	524 (92%)	520.34	16.34
100	100	truncated(0.2)	(μ, λ)	one-point	False	524 (92%)	517.76	17.41
50	100	roulette-wheel	$(\mu + \lambda)$	uniform	False	526 (92%)	519.46	16.36

Results: Genetic Algorithm

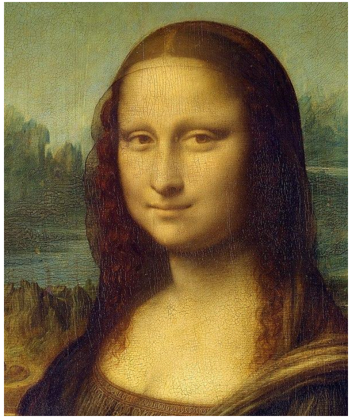


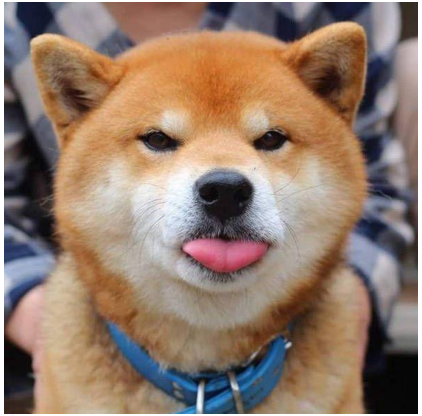
Results: Particle Swarm Optimization

Particle Swarm Optimization approach:

swarm size	segment size	velocity update	neigh. topology	neigh. size	(w, ϕ_1, ϕ_2)	d_{min}	v_{max}	best fitness	avg fitness	std fitness
1000	10	standard	distance	5	(0.1, 1.7, 1.2)	2	50	67.94	66.57	3.75
500	20	standard	star	5	(0.1, 1.7, 1.2)	2	20	65.83	62.87	8.66
500	10	standard	star	5	(0.1, 1.7, 1.2)	2	50	62.21	62.14	0.19
1000	10	standard	star	3	(0.1, 1.7, 1.2)	2	50	57.29	54.86	9.04
1000	20	FIPS	star	3	(0.7, 1.5, 1.5)	2	50	44.65	42.12	8.73
1000	20	standard	distance	3	(0.1, 1.7, 1.2)	2	50	42.70	40.92	7.28
500	20	FIPS	distance	5	(0.1, 1.7, 1.2)	2	50	41.87	37.13	9.79
500	10	CLPSO	ring	5	(0.1, 1.7, 1.2)	2	20	32.49	31.11	3.19
1000	20	standard	ring	5	(0.1, 1.7, 1.2)	10	20	22.35	19.83	8.98
500	10	CLPSO	star	5	(0.1, 1.7, 1.2)	2	50	21.92	20.70	3.74

Results: Particle Swarm Optimization



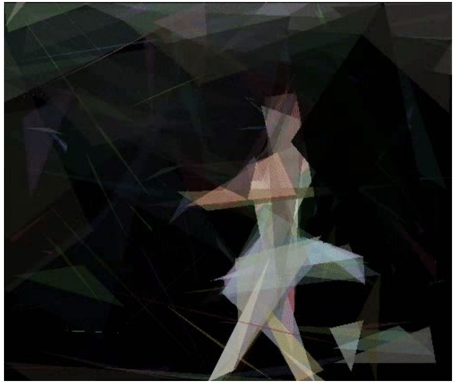




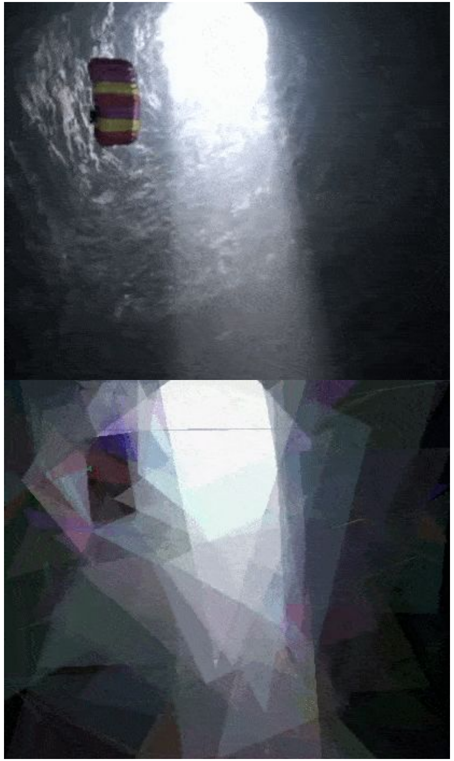

Video Vectorization

The same algorithms can be easily applied to **videos**:

- Optimize over the first frame for 1000 generations
- For the following frames:
    - Start from the previous frame solution
    - Optimize for 100 generations
    - Move to the next frame
    - Repeat



Video Vectorization Results



UNIVERSITÀ DI TRENTO

Department of Information Engineering and Computer Science

Master’s Degree in Artificial Intelligence Systems

Thank you

Matteo Destro

Evolutionary Image Vectorization