Image Approximation via Semi-Transparent Shape Evolution

Matteo Liotta
SM3800072
Optimization for Artificial Intelligence, 2025, UniTS - *Final Project*

Problem Introduction

Problem Introduction

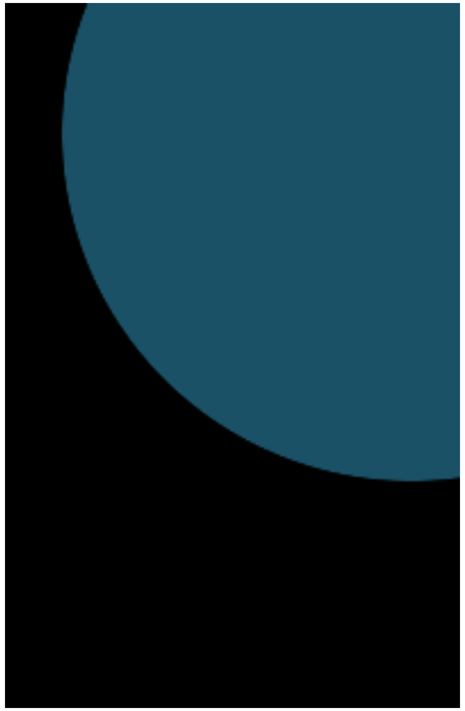
This project is a revisit of the "Genetic-lisa" Peter Braden's project, which aims to recreate images with genetic algorithms.,

It had different implementations over time, from Python to C

The base concept is to **evolve an image**…

- defined as a series of overlapping shapes
- /w (shape and color) mutation
- /w selection on image fitness
- /w crossover

… to reach a certain level of *similitude* with the original picture.



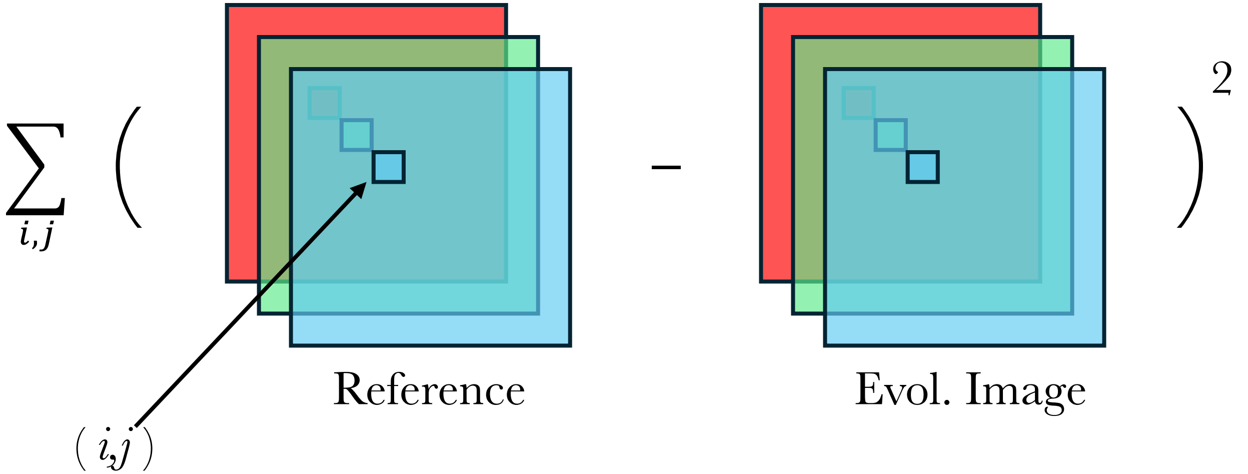
Problem Introduction

Similitude which is measured with defined objective function

$$\sum_{\substack{i=1,\ldots,W \\ j=1,\ldots,H}} \left(r_{i,j_{true}} - \widehat{r_{i,j}}\right)^2 + \left(g_i, j_{true} - \widehat{g_{i,j}}\right)^2 + \left(b_{i,j_{true}} - \widehat{b_{i,j}}\right)^2$$

Problem Introduction

Similitude which is measured with defined objective function



$\forall i = 1, \ldots, W$
$\forall j = 1, \ldots, H$

How are individuals defined?

How are individuals defined?

Shapes

Color

How are individuals defined?
Shape Definition

In the original code different shapes are proposed, but here we can focus on

- Ellipses
- Rectangle
- Triangles

defined as a dictionary of **name** and **coordinates**

```
{'type': 'triangle', 'coords': [44, 213, 93, 148, 163, 220]}
```

How are individuals defined?
Shape Definition

In the original code different shapes are proposed, but here we can focus on

- Ellipses
- Rectangle
- Triangles

defined as a dictionary of **name** and **coordinates**

$$\texttt{\{'type': 'triangle', 'coords': [}\underbrace{\texttt{44, 213}}_{\mathbf{(x,y)}}\texttt{, }\underbrace{\texttt{93, 148}}_{\mathbf{(x,y)}}\texttt{, }\underbrace{\texttt{163, 220}}_{\mathbf{(x,y)}}\texttt{]\}}$$

How are individuals defined?
Shape Definition

In the original code different shapes are proposed, but here we can focus on

- Ellipses
- Rectangle
- Triangles

defined as a dictionary of **name** and **coordinates**

```
{'type': 'rectangle', 'coords': [31, 136, 48, 160]}
{'type': ‘ellispes', 'coords': [31, 136, 48, 160]}
```

How are individuals defined?

Shape Definition

In the original code different shapes are proposed, but here we can focus on

- Ellipses
- Rectangle
- Triangles

defined as a dictionary of **name** and **coordinates**

y **y + shape height**

```
{'type': 'rectangle', 'coords': [31, 136, 48, 160]}

{'type': ‘ellipse', 'coords': [31, 136, 48, 160]}
```

x **x + shape width**

Shape Color Definition

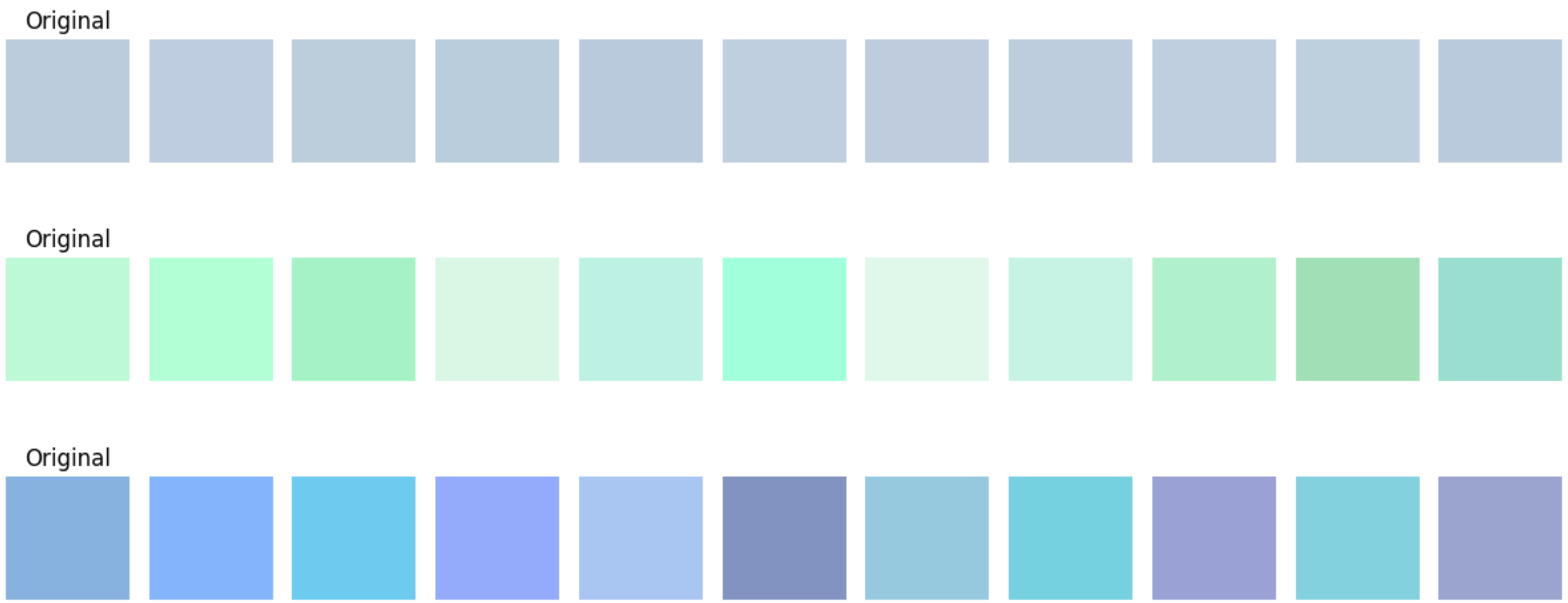
Also color could be generated, but the original code introduced (and didn’t used) the **prev** input

```
def generate_color(prev = None, a_mutation = (-0x5,0x3), rgb_mutation = (-0x5,0x3)):
        if prev:
            return (prev[0] + random.randint(*a_mutation),  # R
                    prev[1] + random.randint(*rgb_mutation),  # G
                    prev[2] + random.randint(*rgb_mutation),  # B
                    prev[3] + random.randint(*rgb_mutation))  # A

       return (random.randint(0x33, 0x99), # R
               random.randint(0, 0xff), # G: between 0 and 255
               random.randint(0, 0xff),     # B: between 0 and 255
               random.randint(0, 0xff))     # A: between 0 and 255
```

Shape Color Definition

This will be used inside the individual shape **mutation**: **using the *prev*** parameter we could *eventually* mutate the color avoiding random re-generation



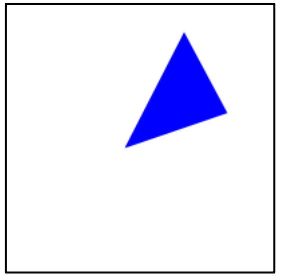
→ *Original Code*

→ $a \pm 0, \text{rgb} \pm 50$

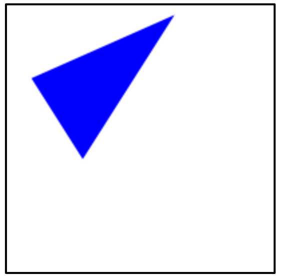
→ $a \pm 30, \text{rgb} \pm 40$

...Back to shape definition

This idea could be introduced in **shape mutation** too: it could avoid, eventually, random shape re-generation



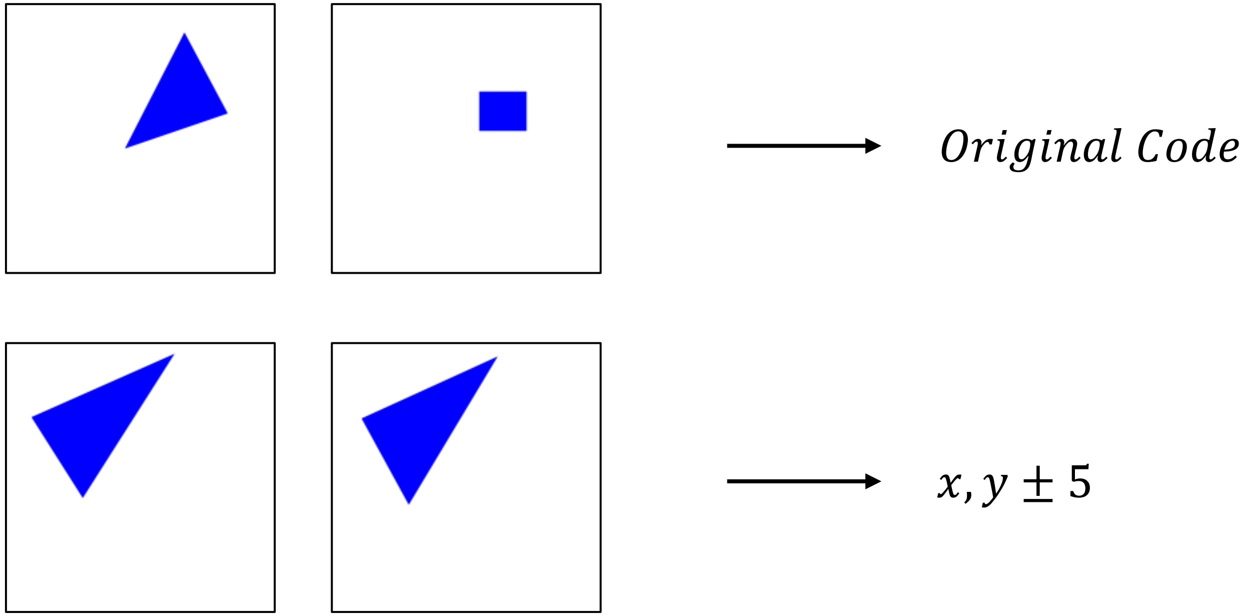
$Original\ Code$



$x, y \pm 5$

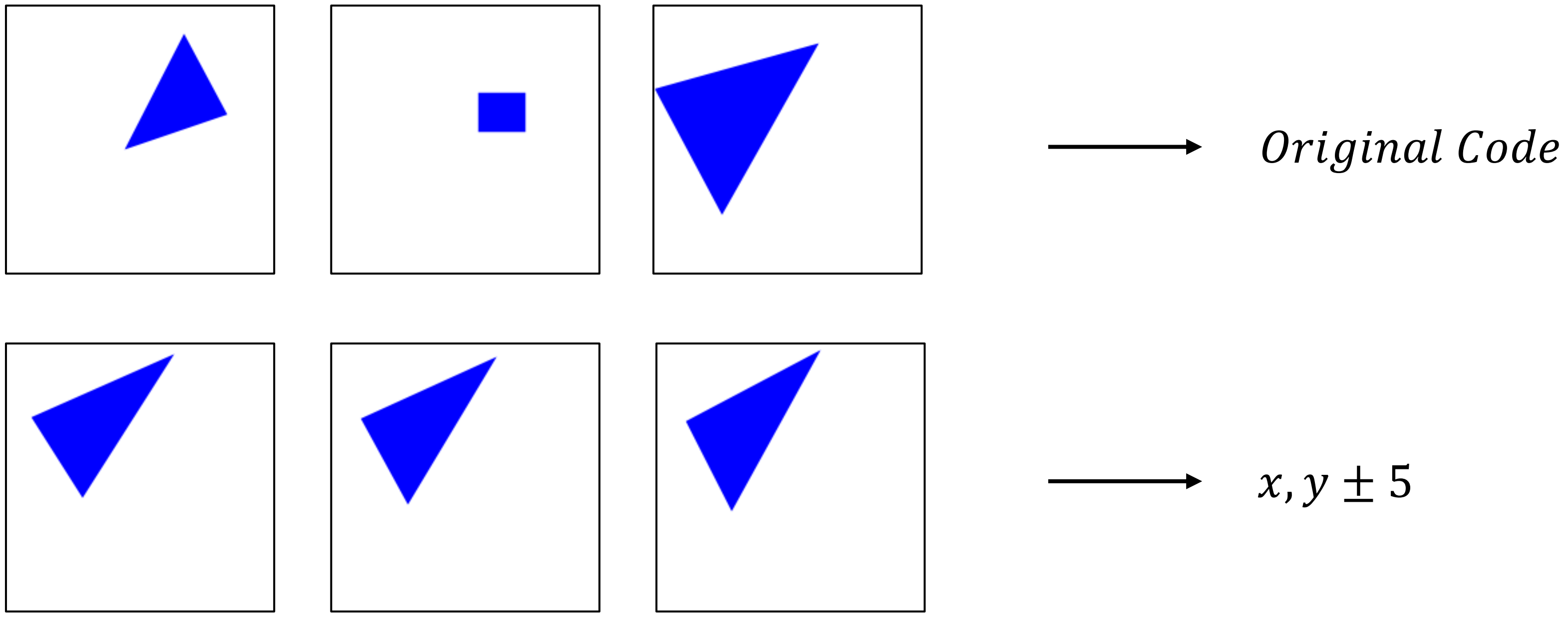
...Back to shape definition

This idea could be introduced in **shape mutation** too: it could avoid, eventually, random shape re-generation



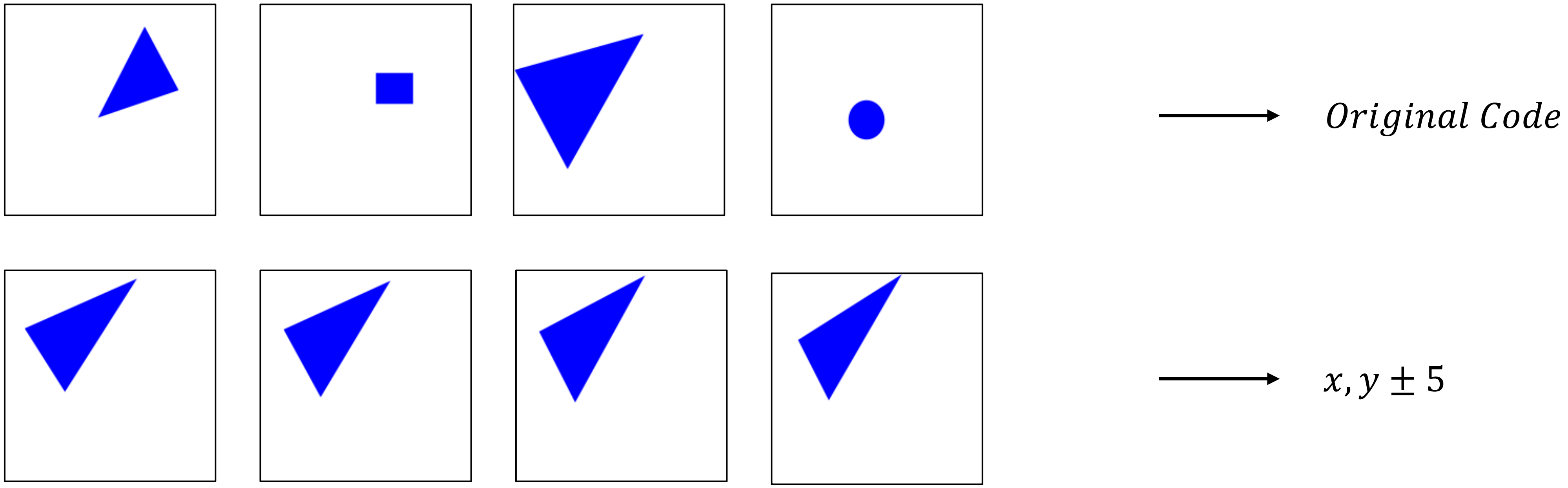
...Back to shape definition

This idea could be introduced in **shape mutation** too: it could avoid, eventually, random shape re-generation



...Back to shape definition

This idea could be introduced in **shape mutation** too: it could avoid, eventually, random shape re-generation



Individuals

The original code proposed a Strain **Class** to handle

- **Genotype** of individuals creation
- **Crossover** for individual creation from parents
- **Mutation**

Leading to a genotype…

Name	***Dna***	***Fitness value***
String	*{shapes: {type, coordinates}, colors : color tuple}*	*Scalar*

Individuals

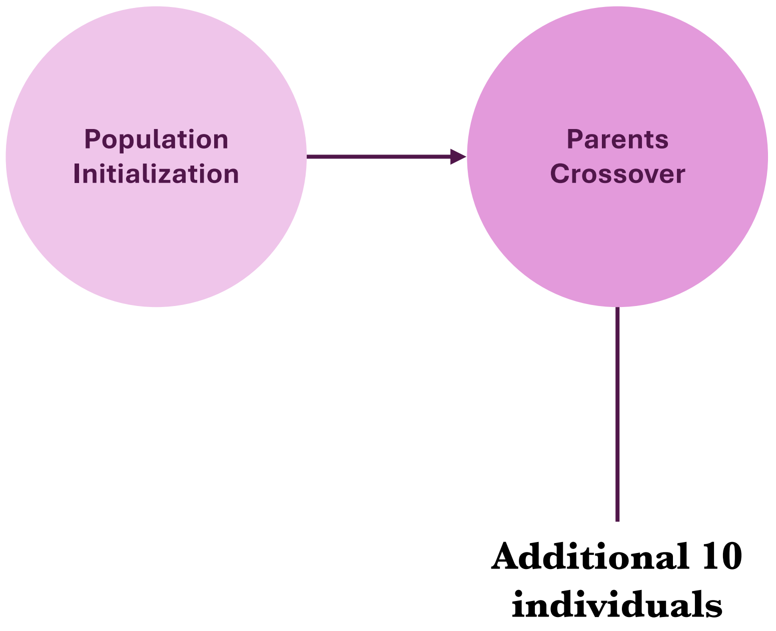
The original code proposed a Strain **Class** to handle

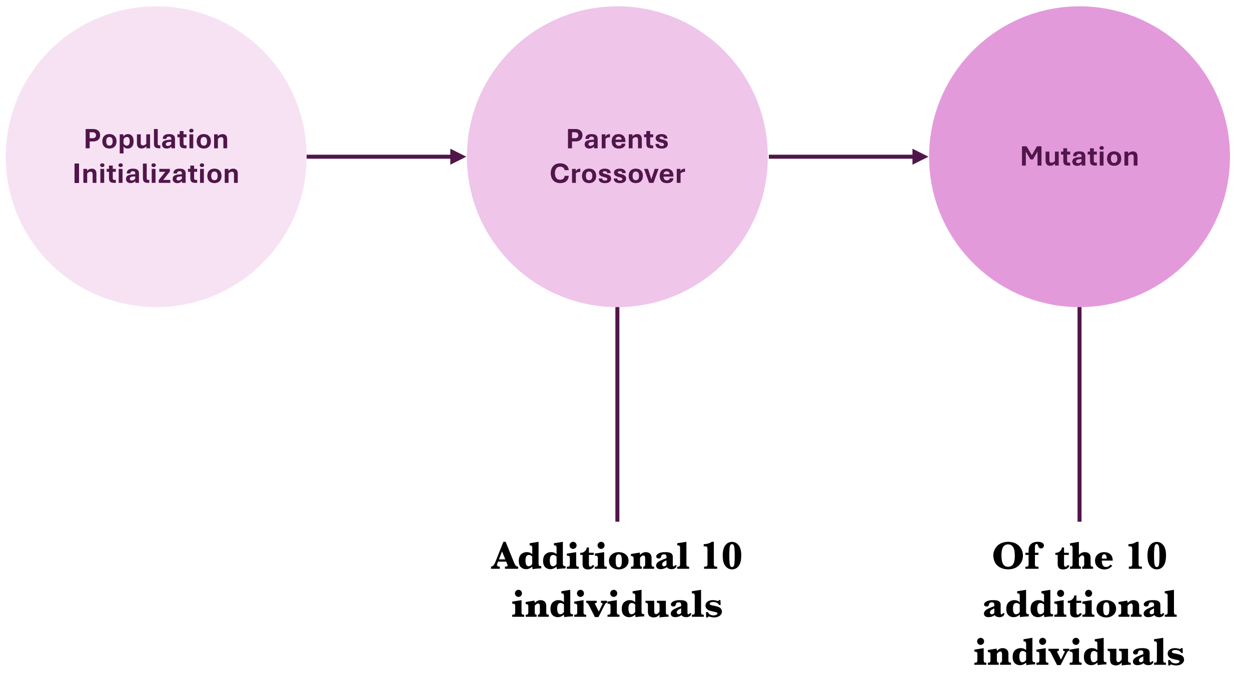
- **Genotype** of individuals creation
- **Crossover** for individual creation from parents
- **Mutation**

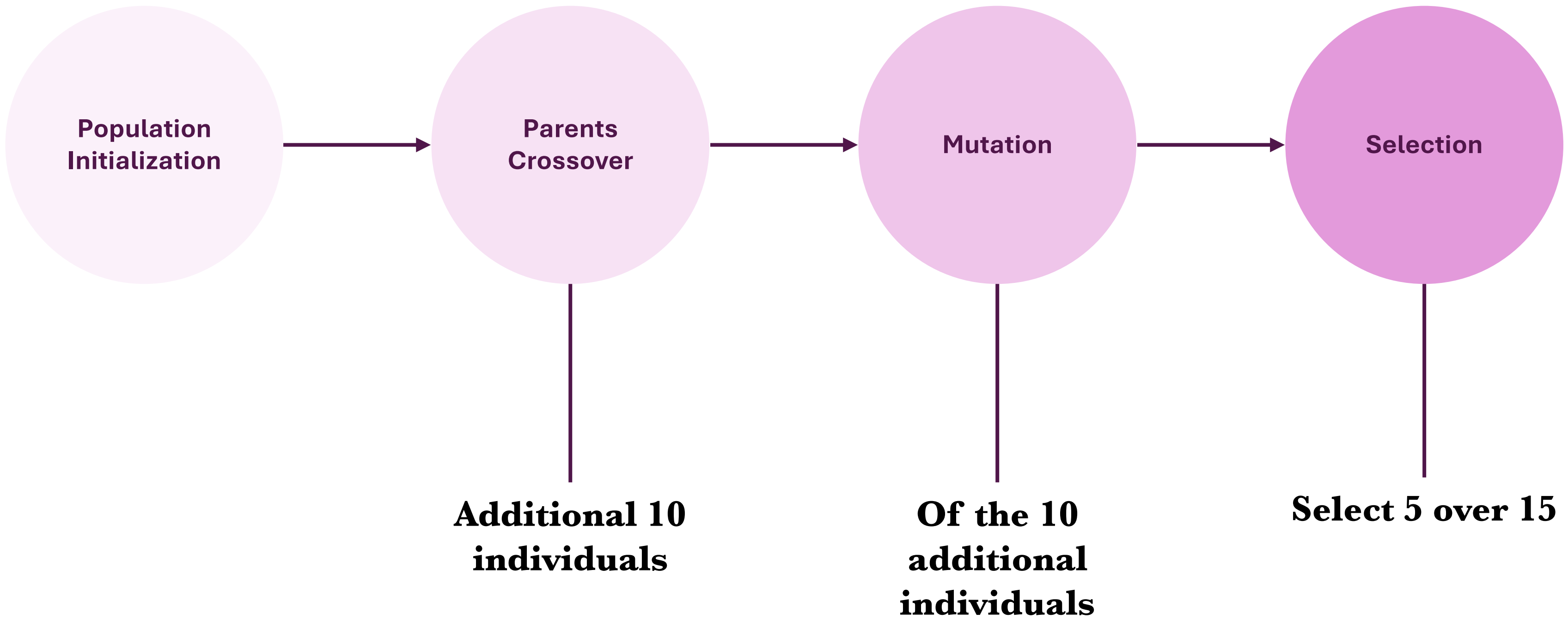
Name	***Dna***	***Fitness value***
String	*{shapes: {type, coordinates}, colors : color tuple}*	*Scalar*

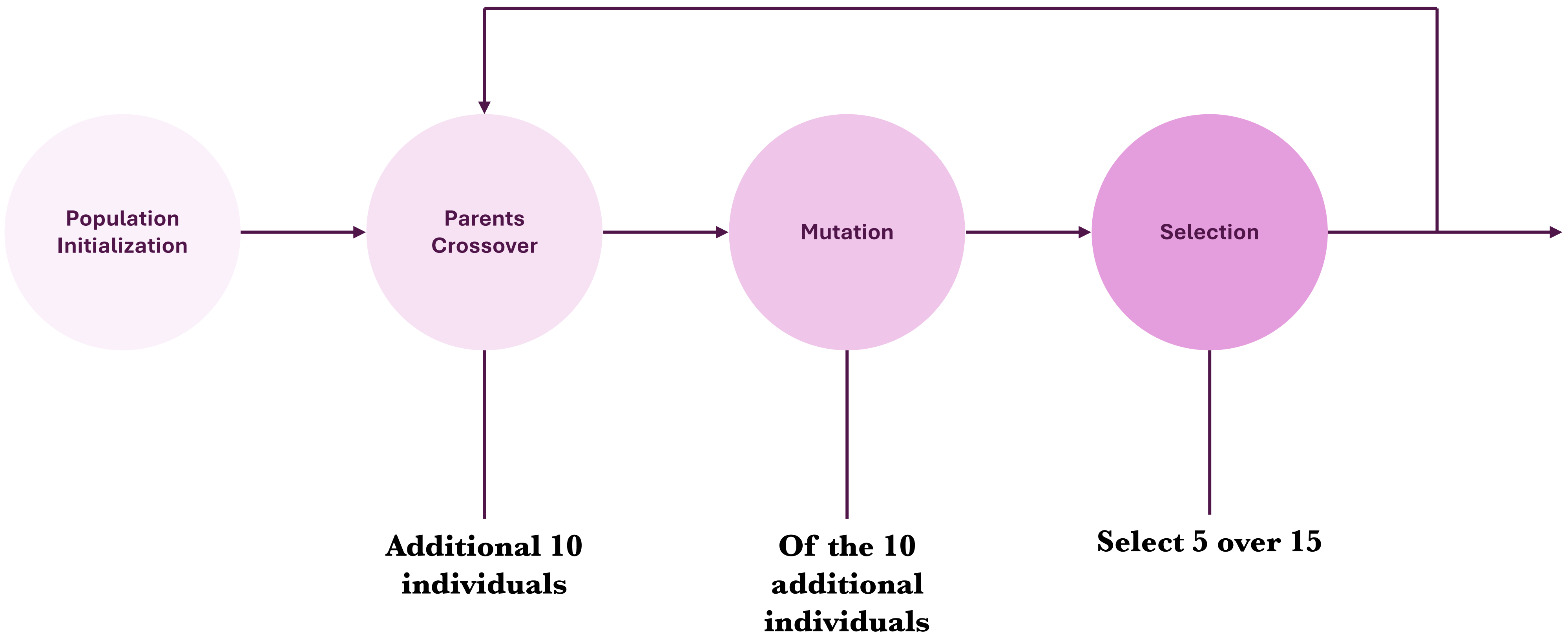
Evolutionary cycle

Population
Initialization


Population
Initialization
Parents
Crossover
Additional 10
individuals


Population Initialization
Parents Crossover
Mutation
Additional 10 individuals
Of the 10 additional individuals


Population
Initialization
Parents
Crossover
Mutation
Selection
Additional 10
individuals
Of the 10
additional
individuals
Select 5 over 15

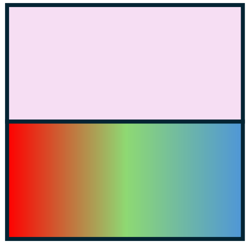

Population Initialization
Parents Crossover
Mutation
Selection
Additional 10 individuals
Of the 10 additional individuals
Select 5 over 15

How is crossover defined?

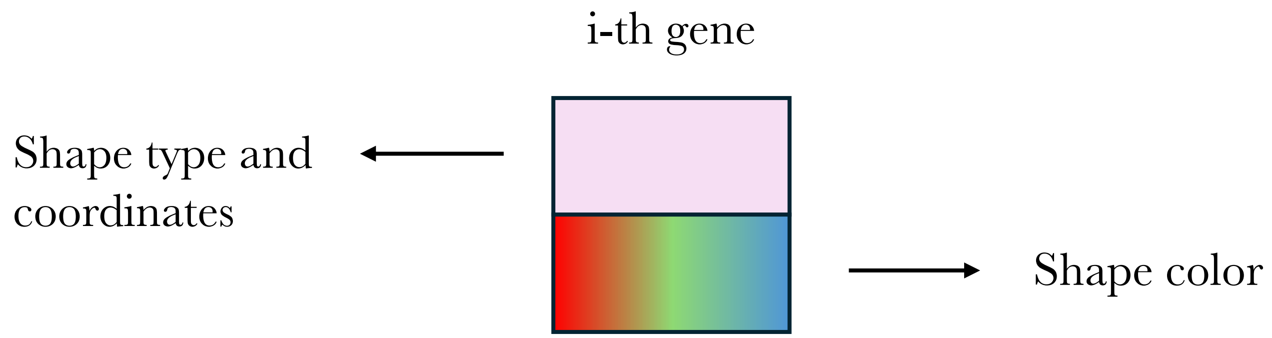
How is crossover defined?

Each individual has its own Dna, defined of atomic units: the genes

i-th gene

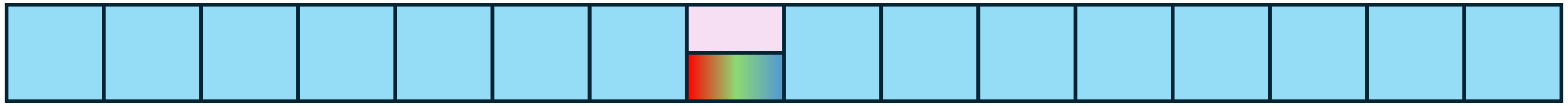


How is crossover defined?



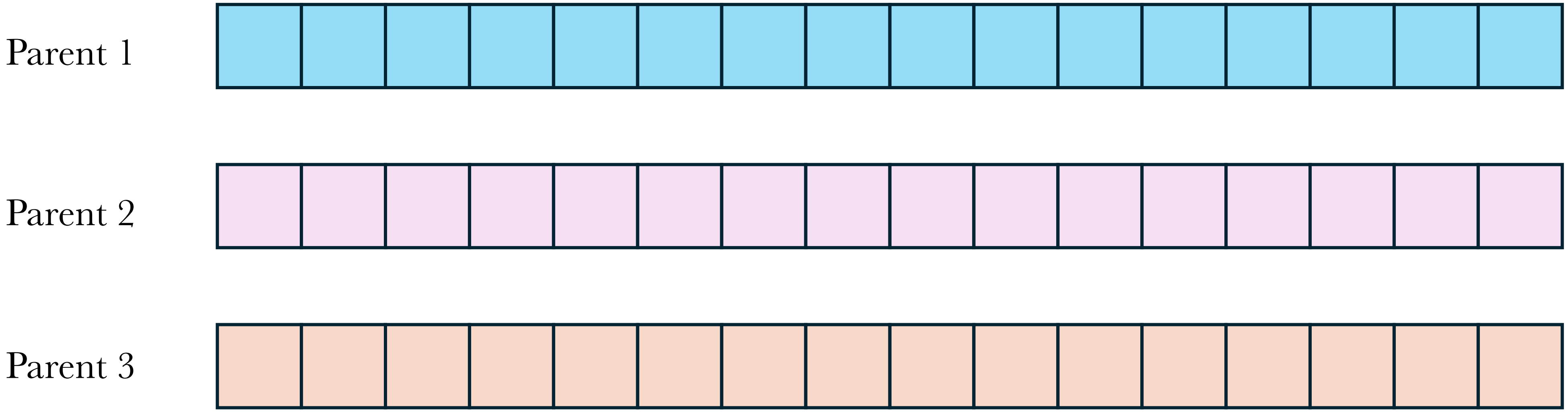
How is crossover defined?

Individual dna



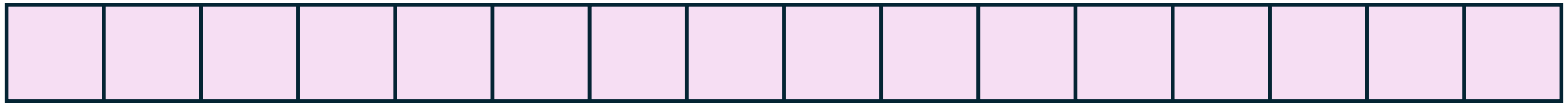
How is crossover defined?

We start with a parent population: each individual has its dna.



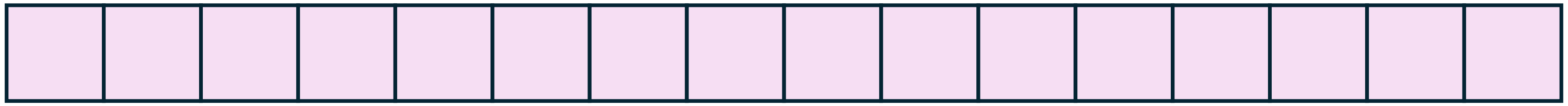
How is crossover defined?

To generate a new individual we select randomly a parent dna and deepcopy it.



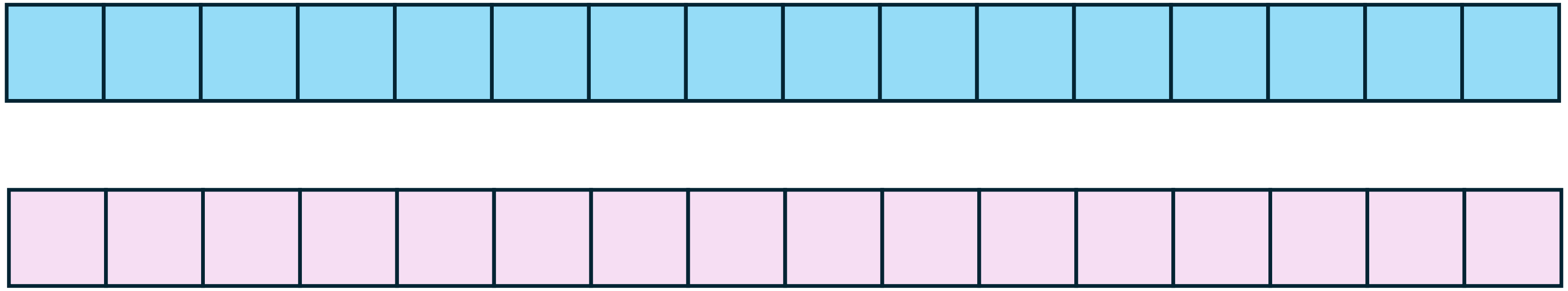
How is crossover defined?

Then iterate on parents: change one random gene of in place of another of this offspring



How is crossover defined?

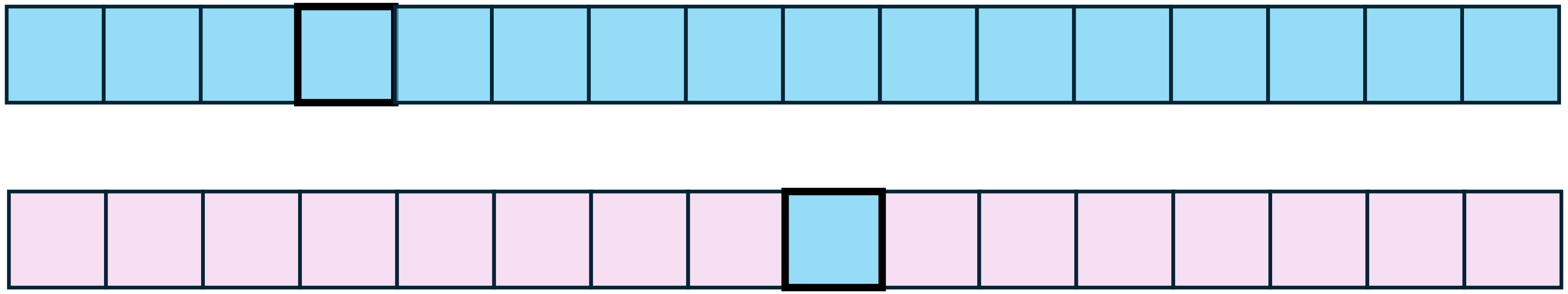
Parent 1



Parent 2

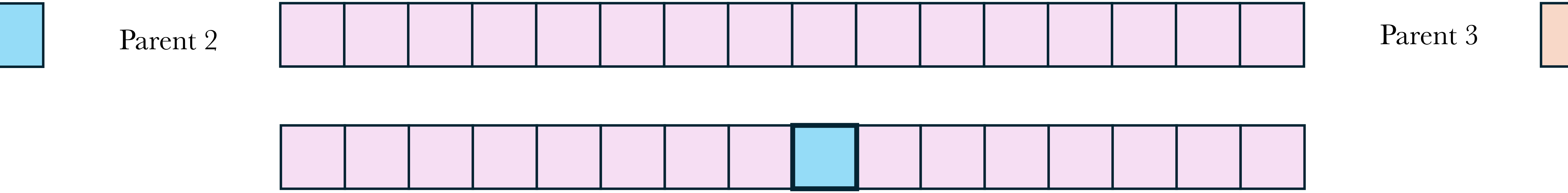
How is crossover defined?

Parent 1

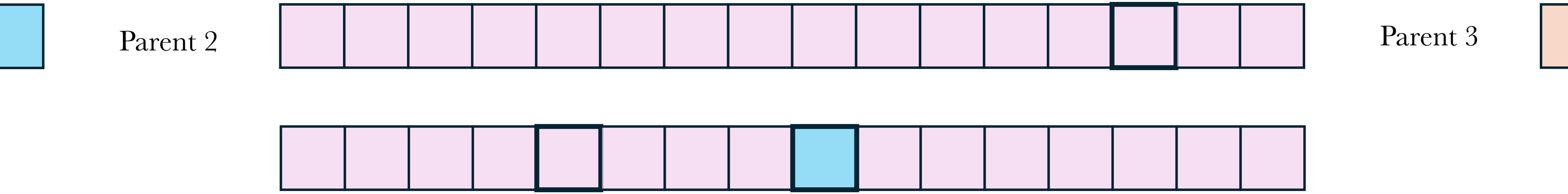


Parent 2

How is crossover defined?

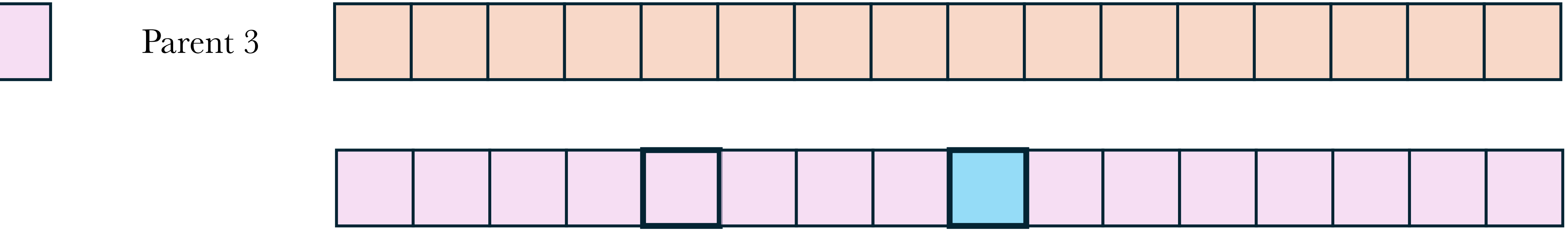


How is crossover defined?



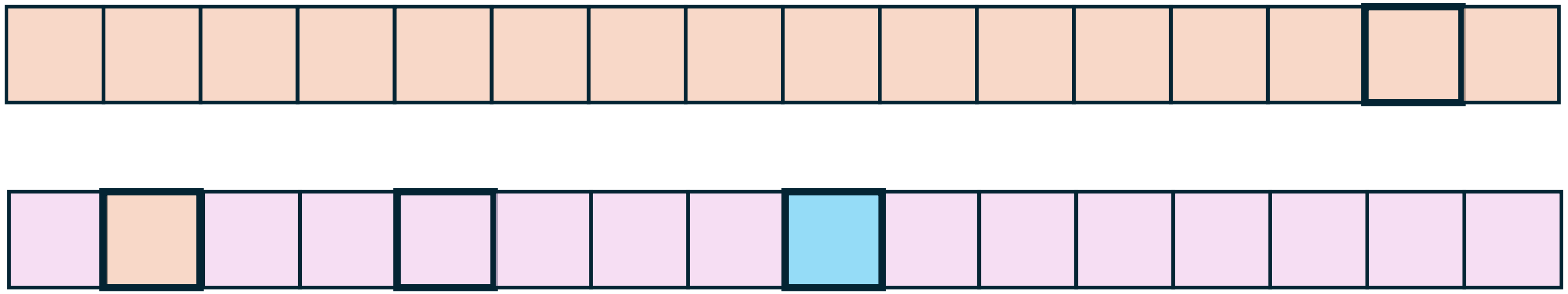
How is crossover defined?

Parent 3



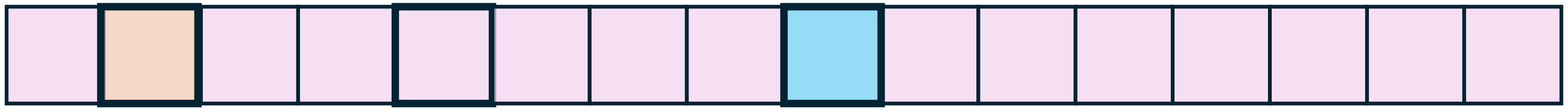
How is crossover defined?

Parent 3



How is crossover defined?

After that, mutation will occur to each new offspring.



How is mutation defined?

Individuals

Three different actions:

- New random shape **appending** to offsprings DNA → ***len<500 and 25% probability***
- Shape change and color **change** of a gene → ***elif 75% probability***
- Random shape **removal** to offspring DNA → ***else***

As introduced before, the function has been changed: now we can change a shape **not by deletion** and **random reinsertion**, but deletion and **conditionate reinsertion**.

(modified*) Original way: performances and results

* Changed mutation, reduced shape size, numpy fitness alternative

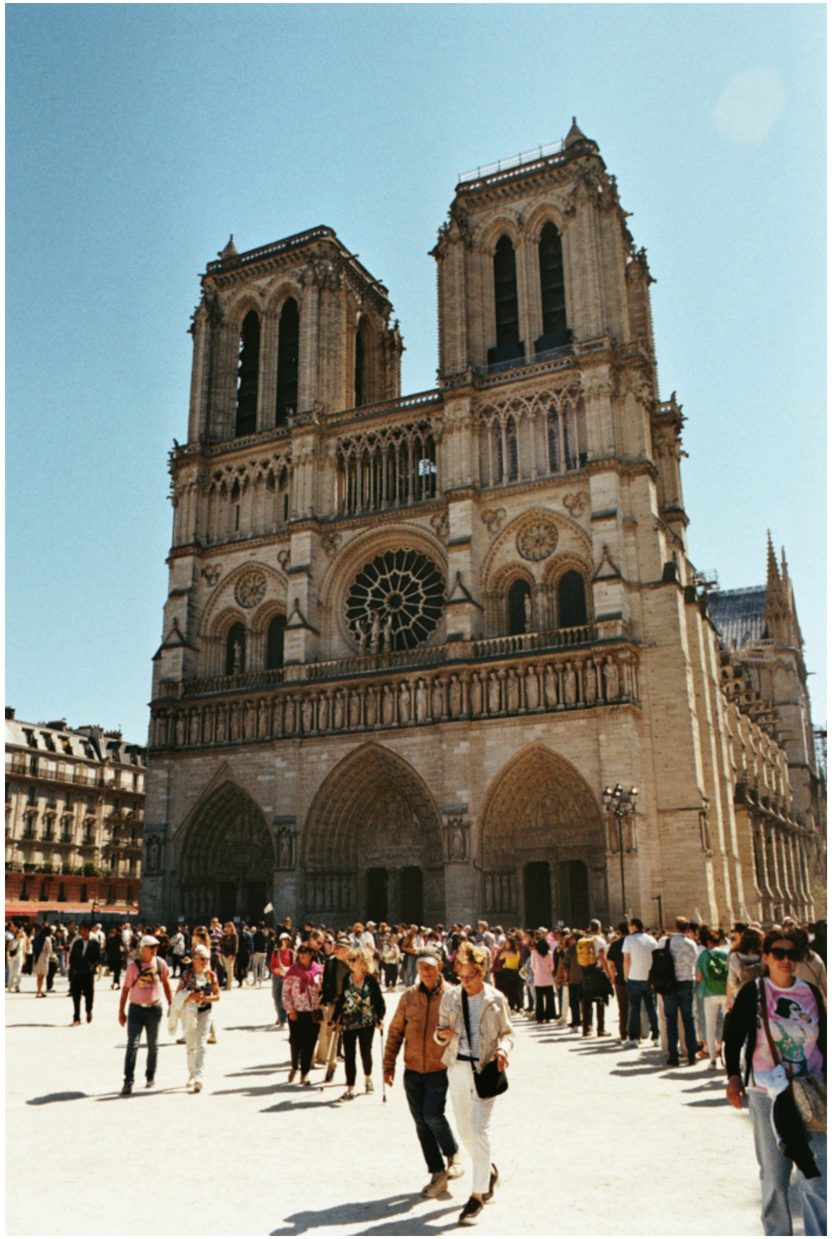
Example on Notre Dame

The example picture is a Notre Dame colored image I took this year in Paris.

The population initial parameters were

- `Image_size: 170 x 256`
- `Generation_population: 10`
- `Crossover_population: 5`

Over 30000 epochs.



The result isn’t so good…

In addition, evaluation time is a problem

Why don’t we change the approach?

How can we speed up evolution?

How can we speed up evolution?

Multiprocessing patches evolution

Fitness simplification

Patches idea introduction

The main idea is to take advantage of multi-core processing.

Notice this is related to the number of cores available

Considering that…

- The image must be big enough
- We cannot evolve on different nodes (evaluation problem)

So the idea is to break and cut the image… into smaller pieces…

Patches idea introduction

Patches idea introduction

… and evolve independently each piece.

We would have many advantages:

- Each core could handle evolution independently on a sub part of the image (patch)
- **Since shape sizes are canvas related, more precision should be present**
- We should notice a **speed increase** since fitness evaluations would be on smaller images, proportional to the number of cores
- Over same number of epochs we end up with core-number times more shapes
- **No assumptions** or reduced set of shapes and colors

+ Now we can measure shape sizes with a parameter

Fitness function revisited

+ we can define an alternative fitness which is the L1 norm:

```
def fitness(self):
        return self._fitness or self._fitness_func()
def _fitness_func(self):
    ref_img = self.reference_image.convert("RGBA")
    draw_img = np.array(self.draw().convert("RGBA"), dtype=np.int16)
    ref_img = np.array(ref_img, dtype=np.int16)
    diff = draw_img[:, :, :3] - ref_img[:, :, :3]
    fitness = np.sum(np.abs(diff))
    self._fitness = fitness
    return fitness
```

Modified way: (compared) performances and results

Time comparison

Original* Project

~**30000** Training Epoch

Evolution Time: ~ **1h 30 min**

Revisited Project

~ **40000** Training Epoch

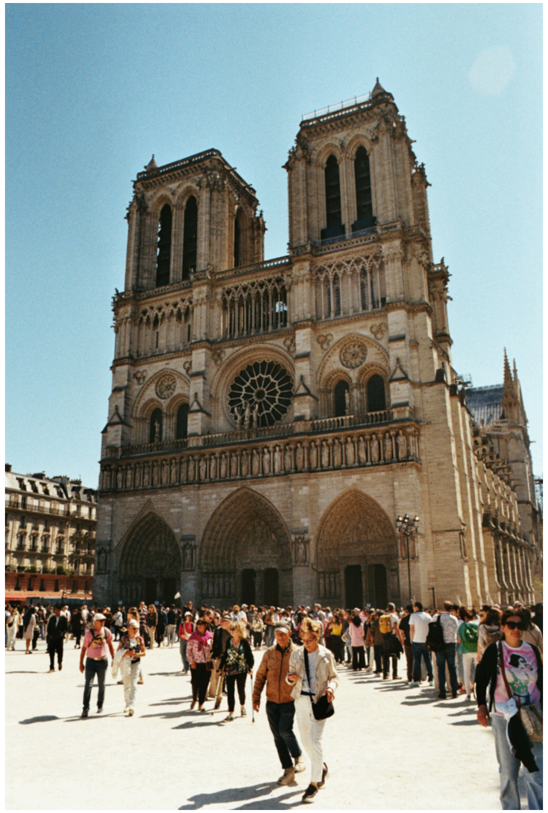
Evolution Time: ~ **30 min**

Same parameters

8 processes on 8 core laptop

L1 approximated fitness

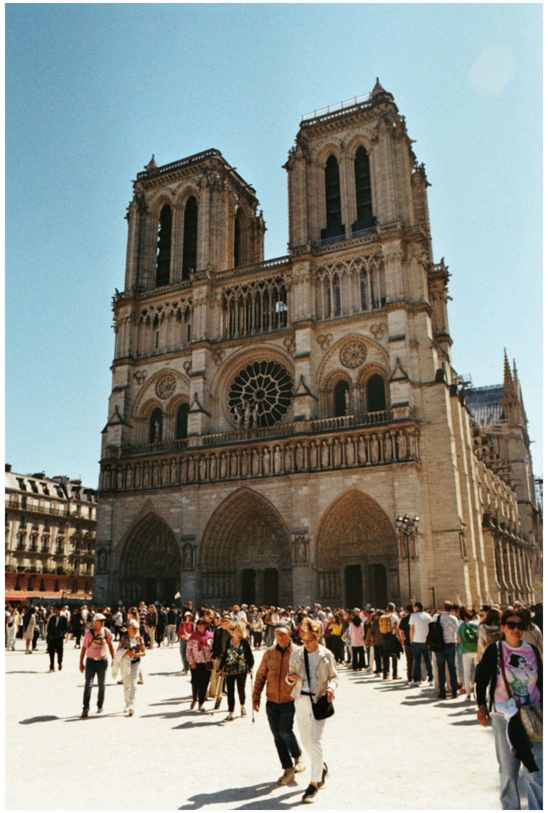
Original*



Reference

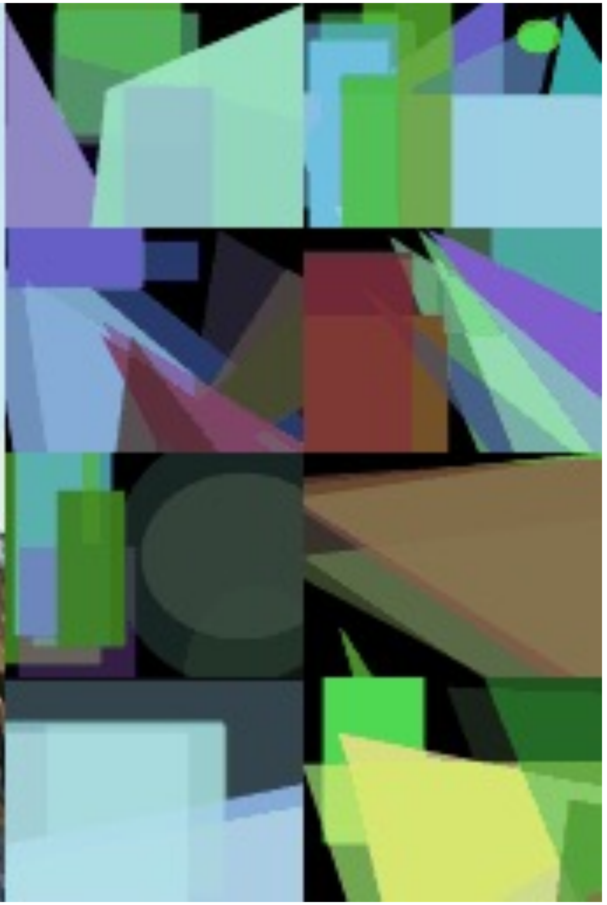
Revisited

Original*



Reference

Revisited

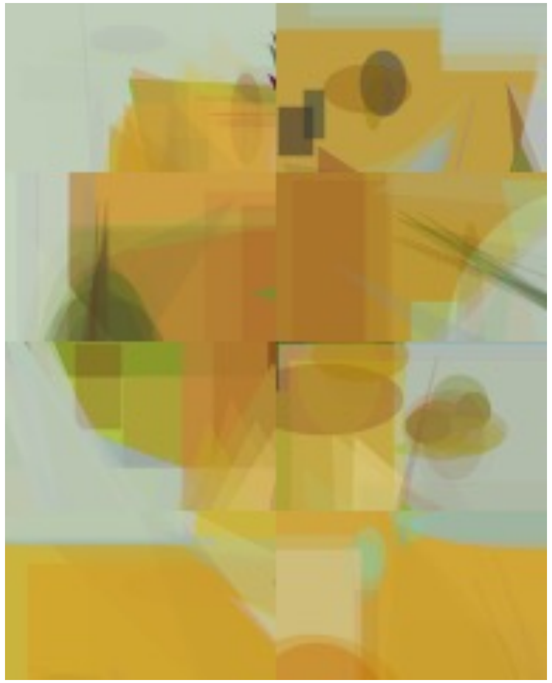
Can we do something more?

Can we do something more? → ***Shape size reduction ratio****

* w.r.t. image size

Reference

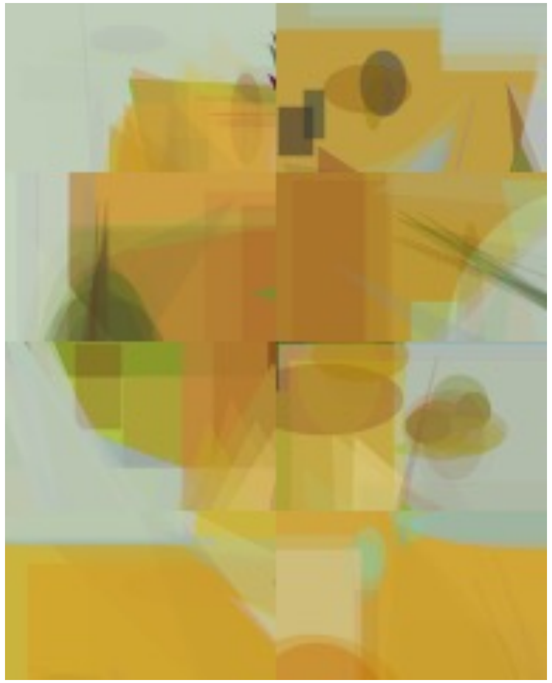
Small Shapes Evolution
1000 epochs



Big Shapes Evolution
1000 epochs

We see a great improvement with a **higher shape size reduction rate** on fixed number of patches (8)

Small Shapes



Big Shapes

* The size is proportional to patch reference size

We can finally see results with:

- Patches and multiprocessing introduction
- More efficient loss
- Random re-generation avoidance for shapes and colors
- Better shape random generation
- Shape size ratio introduction

Reference

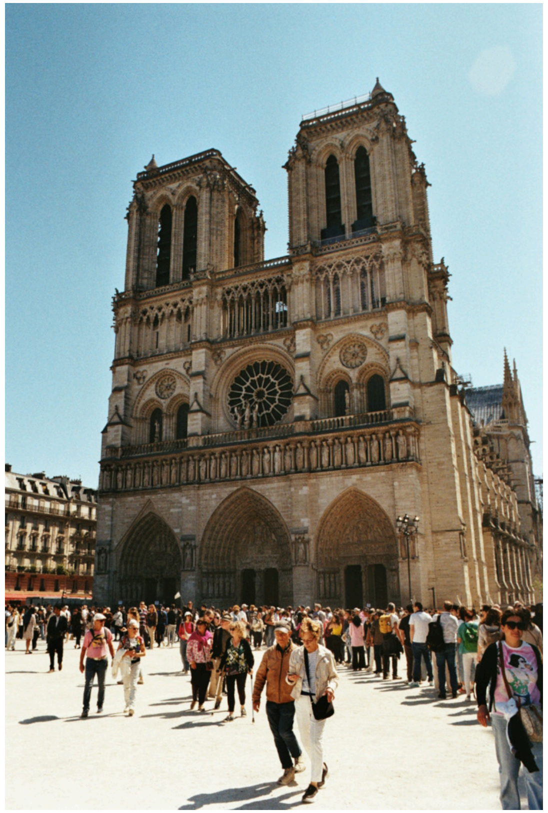
Small Shapes Evolution
50000 epochs

Reference

Small Shapes Evolution
50000 epochs

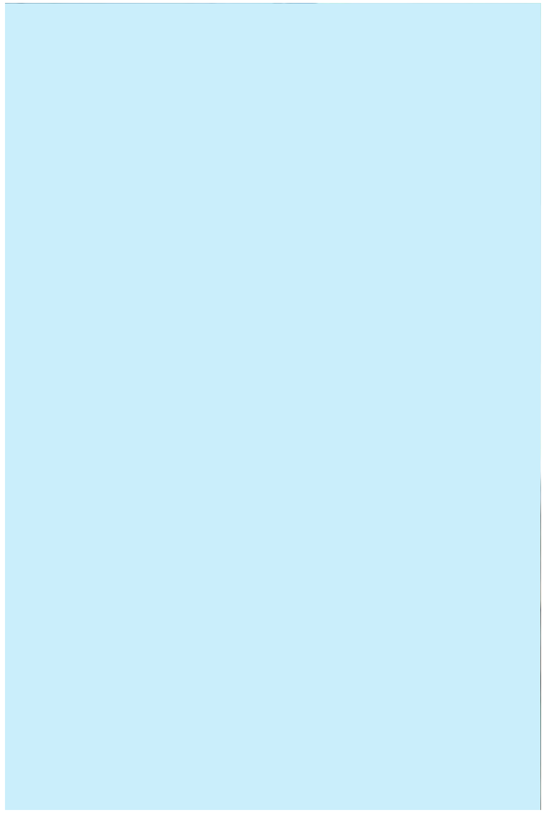
Final Notre Dame visual comparison

Original*



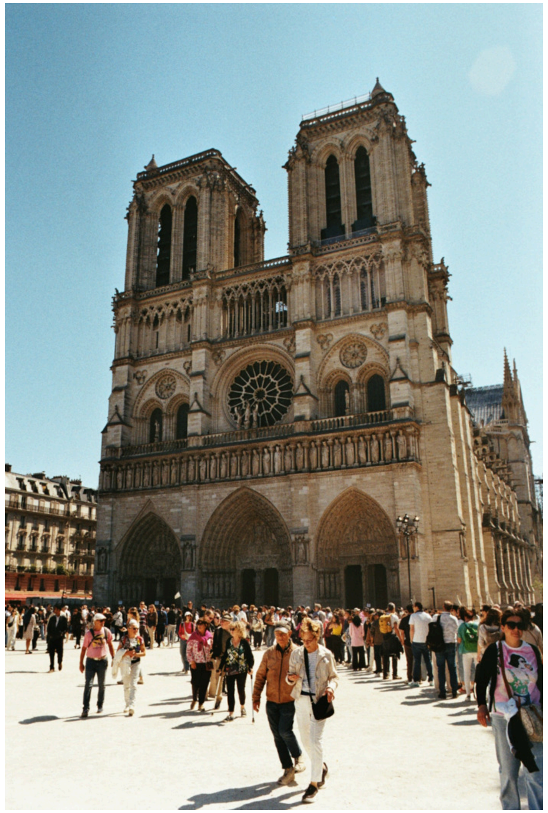
Reference

Revisited



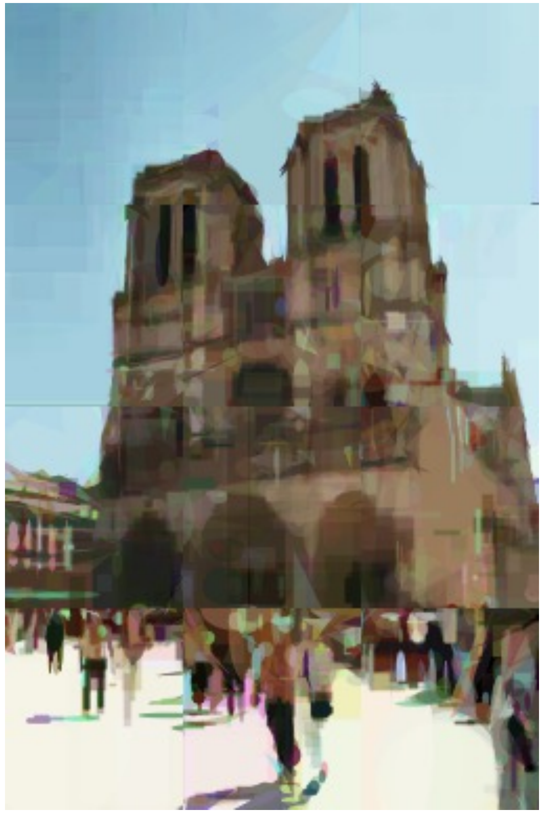
Small shape revisited

Original*



Reference

Revisited



Small shape revisited

Project References

[1] The evolution of a Smile, Peter Braden: https://github.com/peterbraden/genetic-lisa/

[2] Mona Lisa Gif Evolution: https://github.com/peterbraden/genetic-lisa/blob/master/images/lisa-anim.gif

[3] Vase with Twelve Sunflowers (Arles, August 1888), Van Gogh. Neue Pinakothek, Munich: https://commons.wikimedia.org/wiki/File:Vincent_Willem_van_Gogh_128.jpg

[4] La persistenza della memoria, Dalì: https://www.analisidellopera.it/wp-content/uploads/2018/10/Dali_La_persistenza_della_memoria-1.jpg