Evolutionary Pointillist Modules: Evolving Assemblages of 3D Objects

Penousal Machado and Fernando Graça

CISUC, Department of Informatics Engineering, University of Coimbra
Polo II of the University of Coimbra, 3030 Coimbra, Portugal
machado@dei.uc.pt, fejg@student.dei.uc.pt
`http://eden.dei.uc.pt/~machado`

Abstract. A novel evolutionary system for the creation of assemblages of three dimensional (3D) objects is presented. The proposed approach allows the evolution of the type, size, rotation and position of 3D objects that are placed on a virtual canvas, constructing a non-photorealistic transformation of a source image. The approach is thoroughly described, giving particular emphasis to genotype–phenotype mapping, and to the alternative object placement strategies. The experimental results presented highlight the differences between placement strategies, and show the potential of the approach for the production of large-scale artworks.

Key words: Evolutionary Art, Evolutionary Image Filters, Non-Photorealistic Rendering, Assemblage

1 Introduction

The main goal of the research presented in this paper is the creation of large-scale artworks through the assemblage of 3D virtual objects. The main artistic sources of inspiration for this work are: pointillism, mixed media assemblage of objects, and ornamentation techniques (e.g. similar to the ones found in Gustav Klimt works). From a scientific point of view, areas such as evolutionary non-photorealistic rendering and artistic filter evolution, are of particular relevance.

Our approach can be seen as the evolution of a filter that transforms an input source image. More exactly, the evolved individuals take as input (through a terminal node) an image, and produce as output the type, scale, rotation and placement of the objects, which will be placed on the virtual canvas. The color of each object is determined by the color of the corresponding pixel of the source image. In this way, an assemblage of 3D objects, which constitutes a non-photorealistic portrayal of the source image is obtained. Finally, the 3D assemblage is rendered using a raytracer.

We begin with a short survey of related work. Next, in the third section, we make an overview of the different modules of the system. In the fourth section, we describe the evolutionary module, giving particular emphasis to genotype–phenotype mapping, and to the description of the object placement strategies. The experimental results are presented and analyzed in the fifth section. Finally, we draw some conclusions and discuss aspects to be addressed in future work.

2 Related Work

The use of evolutionary algorithms to create image filters and non-photorealistic renderings of source images has been explored by several researchers. Focusing on the works where there was an artistic goal, we can mention the research of: Neufeld and Ross [1, 2], where Genetic Programming (GP) [3], multi-objective optimization techniques, and an empirical model of aesthetics are used to automatically evolve image filters; Lewis [4], which evolved live-video processing filters through interactive evolution; Machado et al. [5], where GP is used to evolve image coloring filters from a set of examples; Yip [6], which employs Genetic Algorithms (GAs) to evolve filters that produce images that match certain features of a target image; Collomosse [7, 8], which uses image salience metrics to determine the level detail for portions of the image, and GAs to search for painterly renderings that match the desired salience maps. Several other examples exist, however a thorough survey is beyond the scope of this article.

3 Overview of the System

Figure 1 presents the architecture of the system, which is composed of two main components: an *Evolutionary* module and a *Previewing and Rendering* module.

The evolutionary module is an expression–based GP [9] interactive breeding tool. It comprises a *Function Visualizer* that depicts a grayscale visualization of the individuals' expression trees. As is usually the case in expression based GP, the grayscale value of a pixel at the (x,y) coordinates (in our case, $x, y \in [-1, 1]$) is determined by the output value of the individuals' expression trees for (x, y). Each individual is an assemblage of 3D objects. Therefore, usually, this visualization mode does not provide enough information to allow educated choices by the user. As such the system also provides a 2D and 3D previewer.

The 2D previewer runs on the master computer. It evaluates the genotypes and places objects accordingly. However, as the name indicates, it doesn't take into consideration the 3D nature of the objects, lighting effects, shadows, etc. The 3D previewer resorts to a Condor–based [10] render farm. The master creates and submits several Condor jobs for each individual of the population. Each job is responsible for: converting the genotype in a Persistence of Vision (POV) 3D scene file[1]; rendering a slice of the resulting 3D scene using POV-Ray; transferring the rendered image slice to the master. The master gathers and merges the rendered image slices, displaying the images as they become available. By changing the settings of the POV-Ray initialization file, the user can adjust the quality and size of the renderings, thus also adjusting the speed.

As mentioned above, we use an interactive breeding approach, i.e., instead of assigning fitness, the user selects two parents, which generate offsprings through crossover and mutation. We also provide a *chromosome replication* operator, which allows the user to select a specific chromosome and transfer mutated versions of it to all the individuals of the population.

[1] `http://www.povray.org/`.

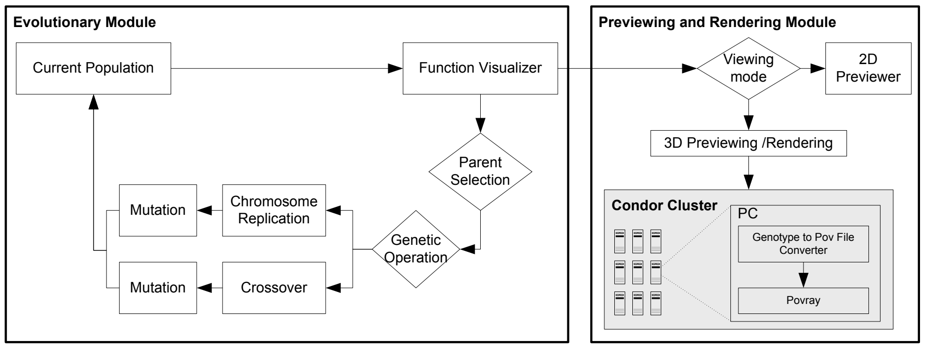


Fig. 1. The main modules of the system.

4 Evolutionary Module

In this section, we describe the evolutionary module, focusing on aspects such as: representation, genetic operators and genotype–phenotype mapping.

4.1 Representation

The genotype of each individual has five chromosomes: <type, rotation, size, x-position, y-position>. Each chromosome is an expression tree, encoding a particular aspect of the 3D assemblage of objects, as follows:

<**type**> − The output value of the *type* expression tree determines what object, from a pool of available ones, will be placed;
<**rotation**> − The rotation that will be applied to the object;
<**size**> − The scaling applied to the object.
<**x-position**> **and** <**y-position**> − The x and y coordinates where the object will be placed;

The output of the trees is calculated for each of the pixels of the source image. In the experiments described in this paper, the function set is:

$$\{sin, cos, max, min, abs, +, -, \times, \%, diff\},$$

where *sin* and *cos* are the usual trigonometric operations; *max* and *min* take two arguments returning, respectively, the maximum and minimum value; *abs* returns the absolute value; $\{+, -, \times\}$ are the standard arithmetic operations; % the protected division operator [3]; $diff$, a function that takes two arguments $(\Delta x, \Delta y)$ and returns the difference between the value of the (x, y) pixel of the source image and the pixel at $(x + \Delta x, y + \Delta y)$. The terminal set used is:

$$\{x, y, image, random_{constants}\},$$

where x and y are variables; *image* is a zero–arity operator that returns the value of the x, y pixel of the source image; $random_{constants}$ are floating point random values between -1 and 1.

4.2 Genetic Operators

Three genetic operators are used: crossover, mutation and chromosome replication.

The crossover operator is the standard GP sub-tree exchange crossover [3]. If we consider two individuals, A and B, this operator will be individually applied to all chromosome pairs (e.g. (A_{type}, B_{type})), with a given probability for each pair. The mutation operator randomly replaces a subtree by a randomly created one. Like for crossover, different mutation probabilities can be specified for each of the five chromosomes.

The chromosome replication operator was introduced to propagate a specific chromosome throughout the entire population. E.g., the user may feel particularly pleased with the rotations applied to the objects in one individual, and wish to use the same rotation expression in all individuals. Alternatively, the user may wish to test small variations of a specific chromosome without changing the remaining ones. To address these needs, the chromosome replication operation copies mutated versions of the chromosome, selected by the user, to all individuals of the population.

4.3 Genotype–Phenotype Mapping

In this section, we describe the genotype–phenotype mapping procedure. To illustrate our explanation we resort to: the genotype presented in Fig. 2, and the source image presented in Fig. 3(a). For the time being, we will assume that the objects are placed following a regular 32×32 grid, and that three types of objects are available: squares, circles and triangles.

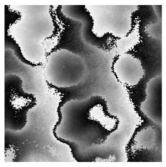
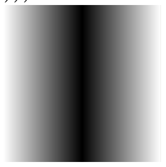
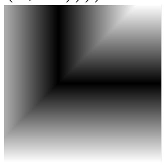
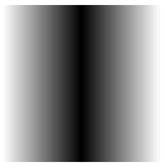
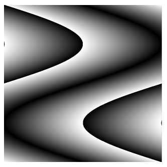
Type	**Rotation**	**Size**	**X-position**	**Y-position**
max(1.79,+(image,x))	min(x,-(1.8, sin(max(y,1.9))))	min(y,-(min (x,x),sin(max (x,1.9))))	abs(x)	-(sin(y),x)
				

Fig. 2. Chromosomes of a sample genotype and the visualization of the corresponding functions over $[-1, 1]$, considering the source image presented in Fig. 3(a)
.

The first chromosome, type, determines which type of object is placed. In this case, values in $]0, 0.33]$ correspond to cubes, in $[0.33, 0.66]$ to spheres, and in $[0.66, 1[$ to triangular shapes. The application of this chromosome, alone, would produce the 3D scene depicted in Fig. 4(a). Likewise, the rotation determines the

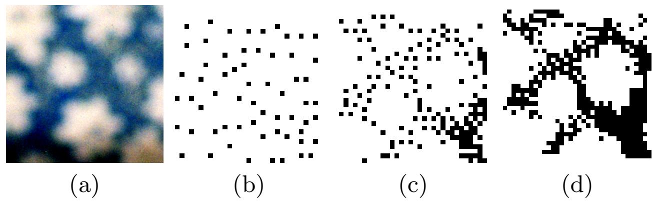


Fig. 3. (a) Source Image; (b) to (d) Dither masks for stages 1 to 3.

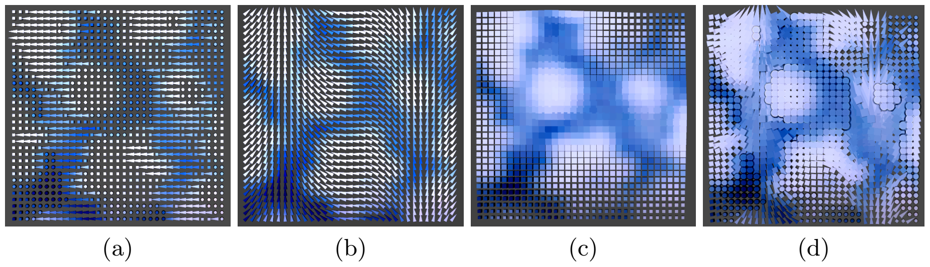


Fig. 4. Images resulting from the application of type (a), rotation (b), size (c), and <type, rotation, size> (d) to the source image of Figure 3(a), assuming a regular grid placement of the objects.

rotation that will be applied to each object, and size determines the scaling that will be applied to the object. Figures 4(b) and (c) depict the results of independently applying these chromosomes, using, respectively, triangular shapes and cubes for easier viewing. The joint application of type, rotation and size would produce the 3D scene presented in Fig. 4(d).

Object Placement So far we considered that the objects are placed on a regular grid. This type of placement has characteristics that we wish to avoid, namely: i) the regularity of the grid can become a visual distraction; ii) it only allows a homogeneous distribution of the objects, making it impossible to ignore regions of the image, and, to clutter objects on certain regions. To overcome this limitation, we introduced the x- and y-position chromosomes, which determine the coordinates where the objects are placed (see Fig. 5(a)).

The number of objects placed is also relevant. To address this issue, we resort to masks. A modified version of a space-filling curve dither algorithm [11, 12] is applied to the source image. By establishing different parameter settings, three dither masks are created (see Fig. 3). The phenotype is rendered in three stages, each using a different dither mask. In each stage, the positions of the objects are calculated using the x- and y-position chromosomes, but an objects is only placed if the mask allows it. In Fig. 5(b), we present the 3D scene corresponding to each rendering stage, and in Fig. 5(c), the one resulting from the combination of the three stages. The color of each object is determined by the color of the area of the source image where the object is placed, which tends to avoid excessive distortions of the original image.

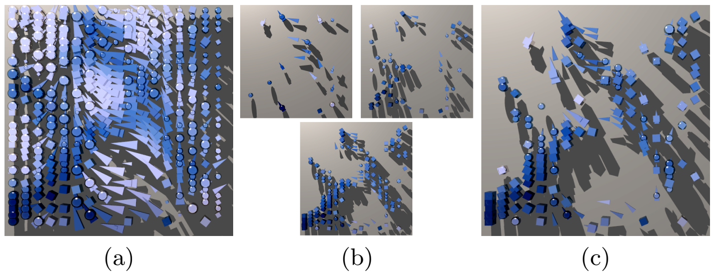


Fig. 5. (a) 3D scene resulting from the application of <type, scale, rotation, x, y>; (b) 3D scene resulting from the stage 1, 2 and 3 dither masks; (c) 3D scene resulting from the combination of the three rendering stages.

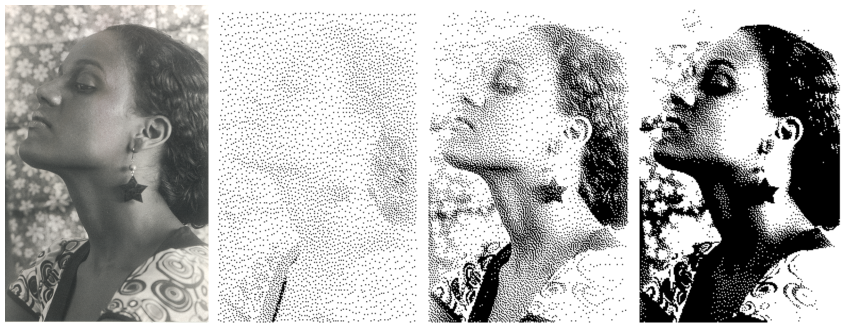


Fig. 6. Source image and corresponding dither masks.

5 Experimental Results

The analysis of the experimental results attained by evolutionary art systems, specially user driven ones, entails a high degree of subjectivity. In our case, there is an additional difficulty — the approach is thought for large-scale formats. Therefore, is close to impossible to adequately convey the real look of the evolved images in the space and format available for their presentation.

Considering these difficulties, we focus on the comparison between different object placement strategies, since an analysis of the effects of type, rotation and size would require a higher level of detail. Due to space restrictions and to the visual nature of the results, we chose to focus on a single evolutionary run.

5.1 Experimental Setup

We used the following experimental settings: Function–set = $\{sin, cos, max, min, abs, +, -, \times, \%, diff\}$; Terminal–set = $\{x, y, image, random_{constants}\}$ (see Sect. 4.1); Population size = 20; Number of Generations = 40; Crossover probability = 0.6; random–subtree mutation probability = 0.2; node–change mutation probability = 0.02 per node; Population initialization method = Ramped half–and–half.

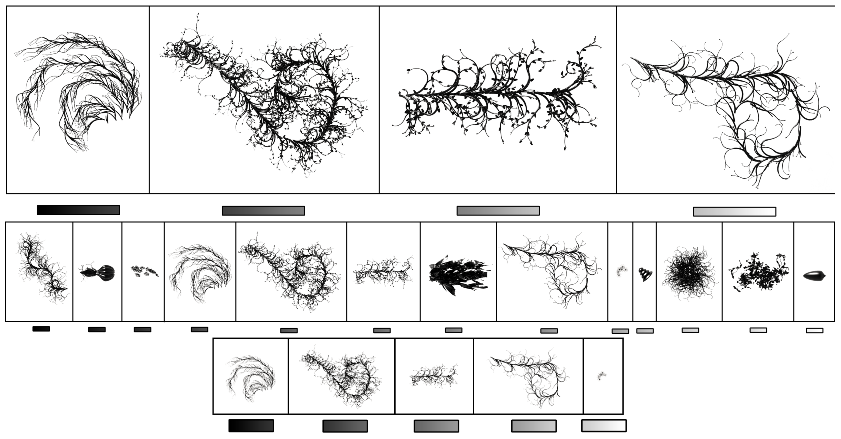


Fig. 7. Pool of objects used in stages 1–3. When no dither masks are used the object pool is equal to the one of stage 2.

The experiments were performed on an Intel Core2Duo, 2.8GHz, Windows master computer. During the course of evolution, an heterogeneous Condor cluster – with 40-70 available machines – was used for the 3D previewing of the populations. To produce large-scale renderings, we used a dedicated Condor cluster with networked file system, composed of 24 Intel Core2Duo, 2.8GHz, running Ubuntu. For previewing, we used a resolution of 800×600 pixels, and the images were rendered without anti-aliasing; the resolution of the large-scale renderings ranged from 3200×2400 to 16000×12000. Typically, previewing took 15 to 30 minutes, and large-scale rendering took 4 to 80 hours (depending on the resolution and number of objects).

In Fig. 6, we present the source image used in the course of these experiments, and the corresponding dither masks for three stage rendering. Figure 7 depicts the pool of objects used in each of the rendering stages, and the associated grayscale gradients. When we follow a object placement strategy that doesn't resort to masks (regular grid or non–masked x- and y-position chromosomes), the object pool is equal to the one of stage 2.

5.2 Results and Analysis

Figure 8, displays the 3D preview of individuals 1–6 of the 1*st* and the 40*th* population rendered using: regular grid placement; evolved x- and y-position chromosomes; three stage rendering with evolved x- and y-position chromosomes and dither masks.

In general, all object placement strategies produced interesting images. During the evolutionary run, the selections of the user were based on the dithered previews. As such, a comparison of object placement approaches, based on the individuals of 40*th* population, would be biased. Therefore, the differences be-

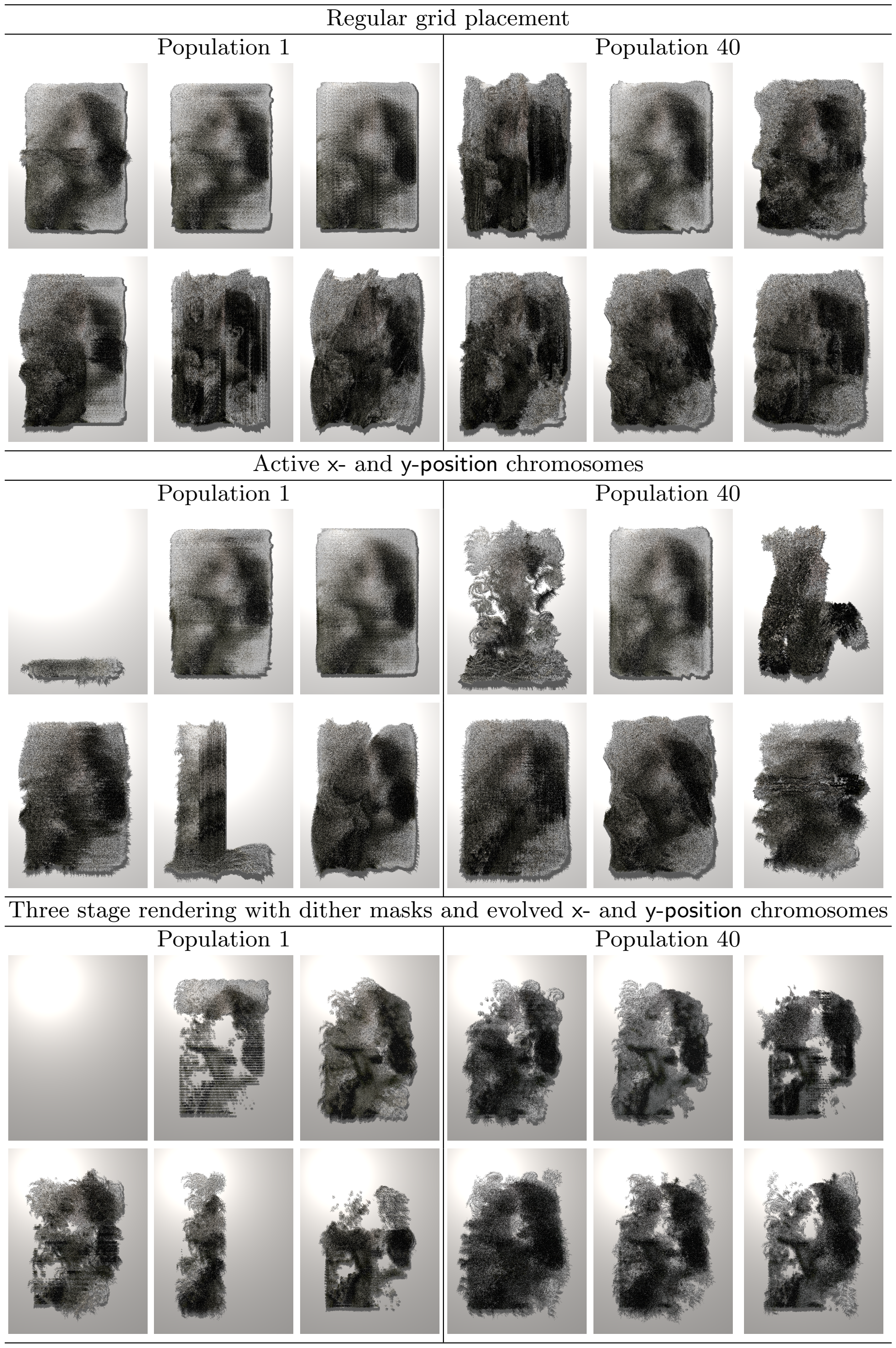

Regular grid placement
Population 1
Population 40
Active x- and y-position chromosomes
Population 1
Population 40
Three stage rendering with dither masks and evolved x- and y-position chromosomes
Population 1
Population 40

Fig. 8. The first 6 individuals of populations 1 and 40 rendered using different object placement strategies.

tween placement strategies are better perceived by observing the images of the 1*st* population.

The limitations of regular grid placement are not visible in this small-scale renderings. The artificial vertical and horizontal artifacts induced by regular placement only become a distraction at larger resolutions. The variations in size and rotation of the objects also help overcoming this limitation, producing, for instance, irregular image boundaries.

As it can be observed, evolving the coordinates of the objects can lead to peculiar images, where significant portions of the image are missing. This expected outcome, is particularly frequent in the beginning of the evolutionary run.

The images produced by three stage rendering with dither masks were the ones that better matched our expectations. The introduction of these masks allowed the "abstraction" of regions of the image, providing the heterogenous level of detail that we wanted.

The comparison of the dithered renderings of the 1*st* and 40*th* population shows that there is less variability in the 40*th* population, indicating some degree of convergence. Unfortunately, the size of the images presented doesn't allow the reader to observe the subtler differences among images, in what concerns object type, rotation, size and placement. Nevertheless, it is safe to say that, according to the opinion of the users guiding the experiment, the images of the 40*th* population are significantly more expressive than the ones from the 1*st*. In other words, the evolutionary algorithm, guided by the user, was able to find regions of the search space that were considered more promising.

In Appendix A, we present larger versions of some of the individuals, and, also, the results of applying a given individual to different source images. These and other images are also available online at: `http://evolving-assemblages.dei.uc.pt`.

6 Conclusions and Future Work

We presented a novel evolutionary approach for the generation of assemblages of 3D objects, and compared the results attained with three different object placement strategies. The experimental results show the potential of the approach, indicating that the proposed three stage rendering, with evolved x- and y-position chromosomes and dither masks, achieves better results.

One of the main limitations of our approach is the computational effort required to preview and render the individuals. To overcome it, we used a Condor-based cluster to distribute the rendering tasks. Nevertheless, this process is still time consuming, which becomes a limitation (specially for previewing since final renderings can be made offline). In the experiments presented, we used fairly complex objects. Using simplified versions of the same objects would greatly reduce previewing time. This is one one of the aspects that will be addressed in the future.

Although the object placement strategies we explored were able to produce interesting results, evolving the masks, would allow greater flexibility. Additionally, image salience analysis can play an important role. The identification of salient detail could be used to guide object placement, in order to promote the placement of objects in areas with salient detail.

Finally, a larger set of experiments – with different types of source images and object pools – is also necessary to better assess the strengths and weakness of the approach.

Acknowledgments. We would like to express our gratitude towards Jennifer Santos, which posed for the photographs employed in this paper. We also acknowledge the contribution of Paulo Marques provided valuable support in the setup of the clusters.

References

1. Ross, B.J., Ralph, W., Hai, Z.: Evolutionary image synthesis using a model of aesthetics. In Yen, G.G., Lucas, S.M., Fogel, G., Kendall, G., Salomon, R., Zhang, B.T., Coello, C.A.C., Runarsson, T.P., eds.: Proceedings of the 2006 IEEE Congress on Evolutionary Computation, Vancouver, BC, Canada, IEEE Press (2006) 1087–1094
2. Neufeld, C., Ross, B., Ralph, W.: The evolution of artistic filters. In Romero, J., Machado, P., eds.: The Art of Artificial Evolution. Springer (2007 (In Press))
3. Koza, J.R.: Genetic Programming: On the Programming of Computers by Natural Selection. MIT Press, Cambridge, MA (1992)
4. Lewis, M.: Aesthetic video filter evolution in an interactive real-time framework. In: Applications of Evolutionary Computing, EvoWorkshops2004: EvoBIO, EvoCOMNET, EvoHOT, EvoIASP, EvoMUSART, EvoSTOC. Volume 3005 of LNCS., Coimbra, Portugal, Springer Verlag (2004) 409–418
5. Machado, P., Dias, A., Cardoso, A.: Learning to colour greyscale images. The Interdisciplinary Journal of Artificial Intelligence and the Simulation of Behaviour – AISB Journal **1** (2002) 209–219
6. Yip, C.: Evolving Image Filters. Master's thesis, Imperial College of Science, Technology, and Medicine (2004)
7. Collomosse, J.P., Hall, P.M.: Genetic paint: A search for salient paintings. In: Applications of Evolutionary Computing, EvoWorkshops 2005, Lausanne, Switzerland (2005) 437–447
8. Collomosse, J.P.: Supervised genetic search for parameter selection in painterly rendering. In: Applications of Evolutionary Computing, EvoWorkshops 2006, Budapest, Hungary (2006) 599–610
9. Sims, K.: Artificial evolution for computer graphics. ACM Computer Graphics **25** (1991) 319–328
10. Tannenbaum, T., Wright, D., Miller, K., Livny, M.: Condor – a distributed job scheduler. In Sterling, T., ed.: Beowulf Cluster Computing with Linux. MIT Press (2001)
11. Velho, L., de Miranda Gomes, J.: Digital halftoning with space filling curves. SIGGRAPH Comput. Graph. **25** (1991) 81–90
12. Shiraishi, M., Yamaguchi, Y.: An algorithm for automatic painterly rendering based on local source image approximation. In: NPAR '00: Proceedings of the 1st international symposium on Non-photorealistic animation and rendering, New York, NY, USA, ACM (2000) 53–58

Appendix: Large-Scale Renderings

As previously mentioned, the techniques explored in this paper are thought for large-scale renderings and printouts. As such, it is difficult to convey the real aspect and visual impact of the final artworks. Additionally, space restrictions also make it impossible to include larger images in the article. To allow a better of the results we include several larger-resolution renderings in this Appendix.

In Figs. 9 to 11 we present a 3200×1600 rendering of the $4th$ individual of the $40th$ population, applied to different source images. In Figs. 12 to 14 we present a 3200×1600 rendering of an individual from an intermediate population.

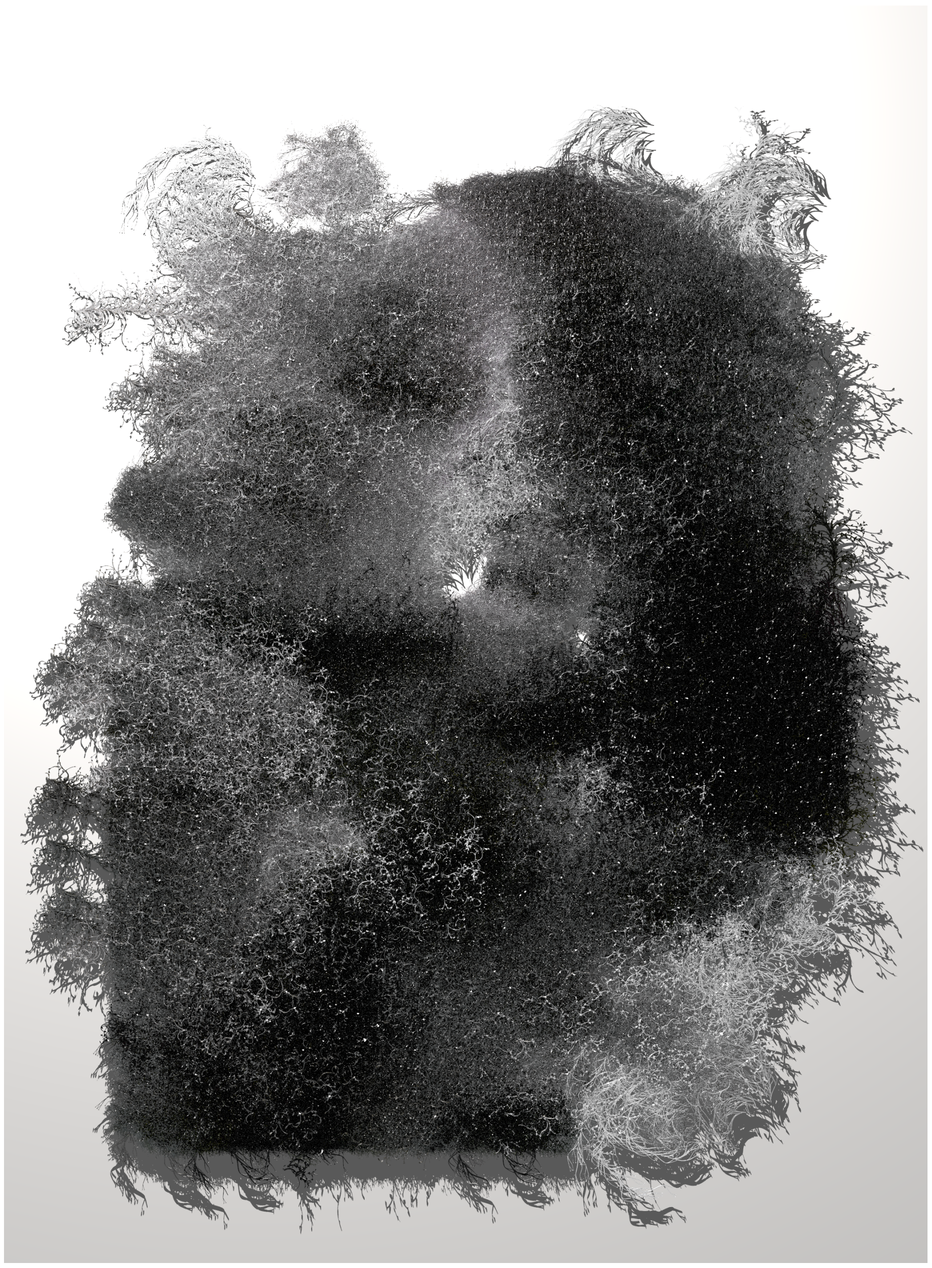


Fig. 9. 3200×1600 rendering of the *4th* individual of the *40th* population.

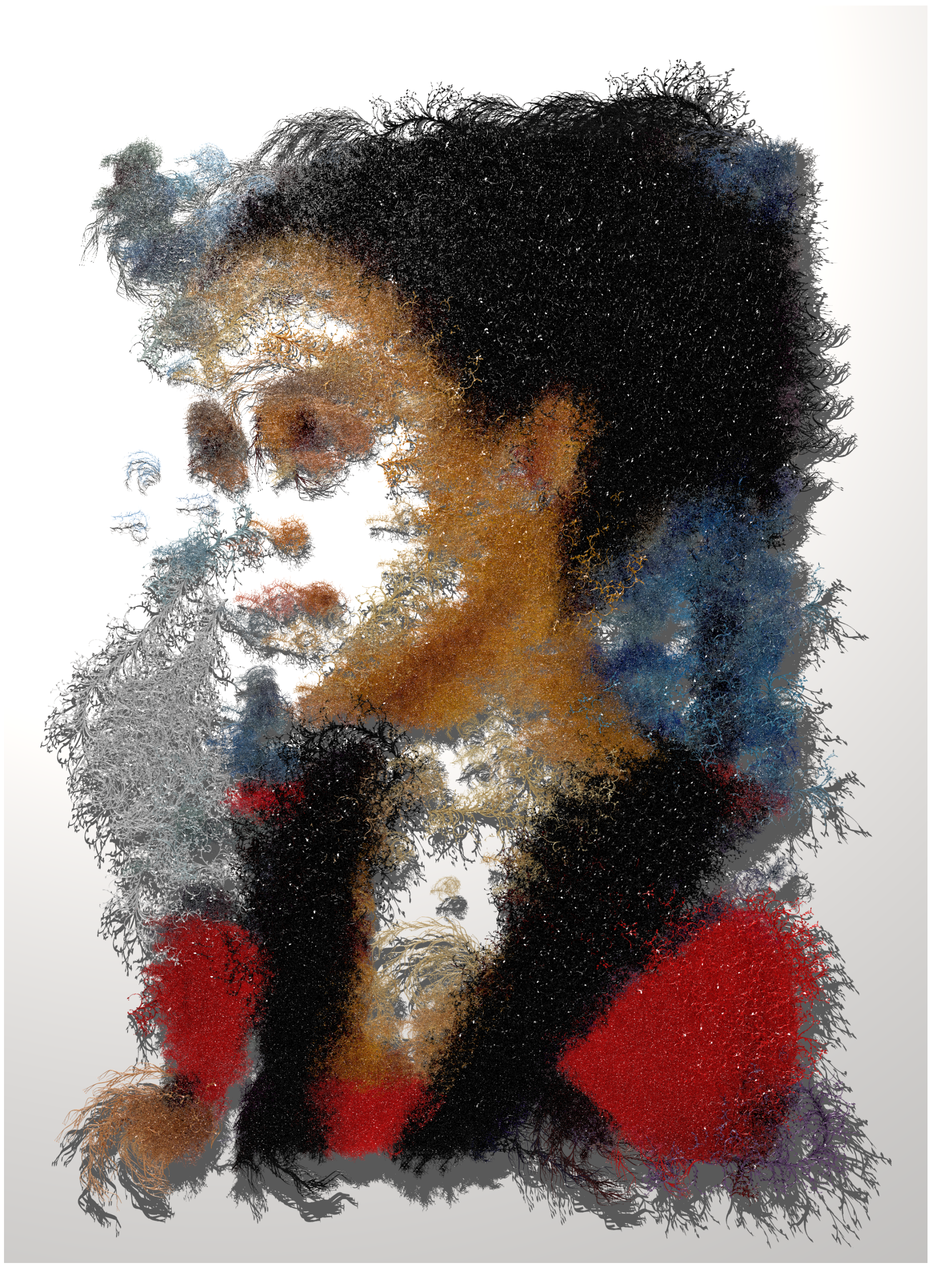


Fig. 10. 3200 × 1600 rendering of the 4*th* individual of the 40*th* population applied to a different source image.

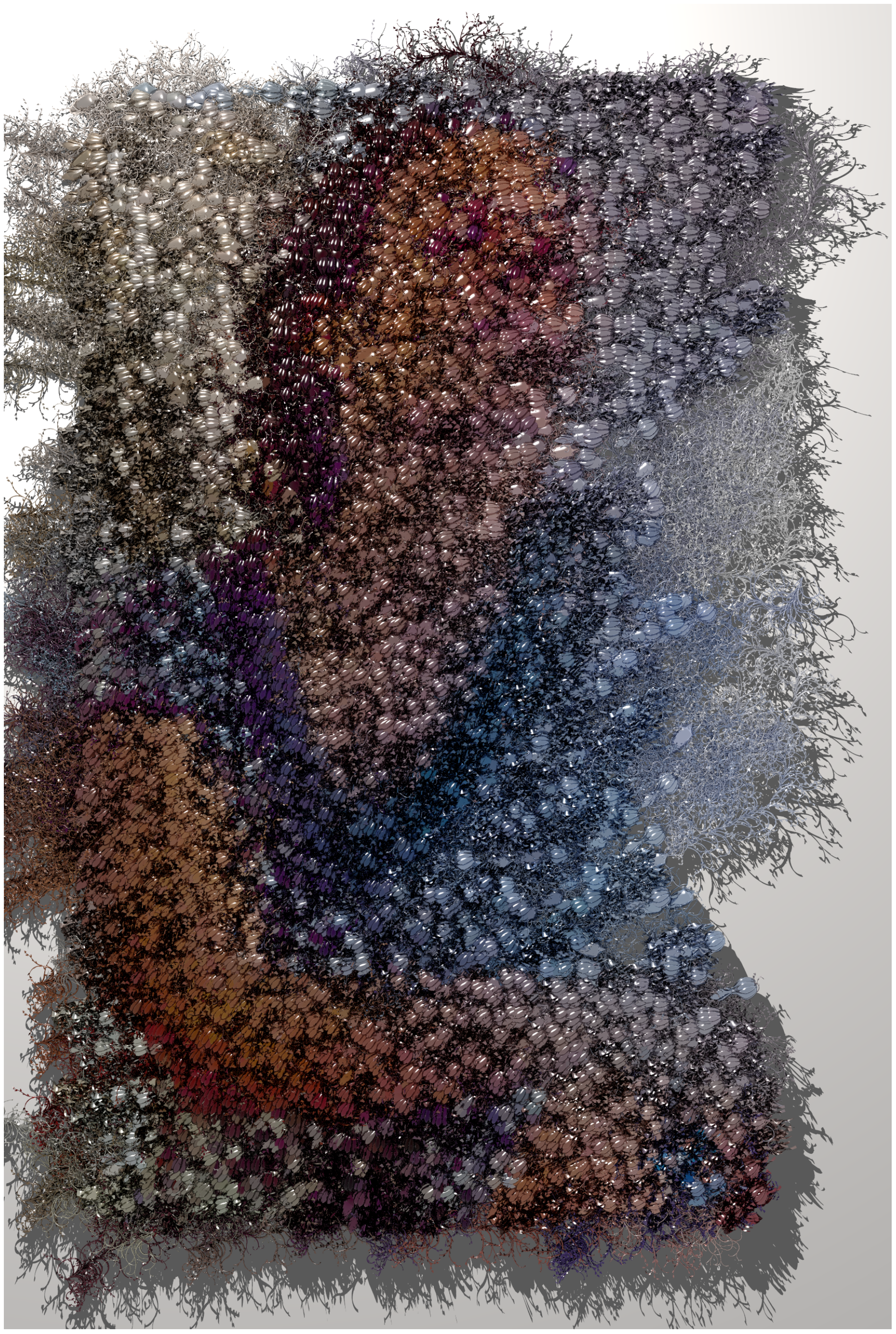


Fig. 11. 3200 × 1600 rendering of the *4th* individual of the *40th* population applied to a different source image.

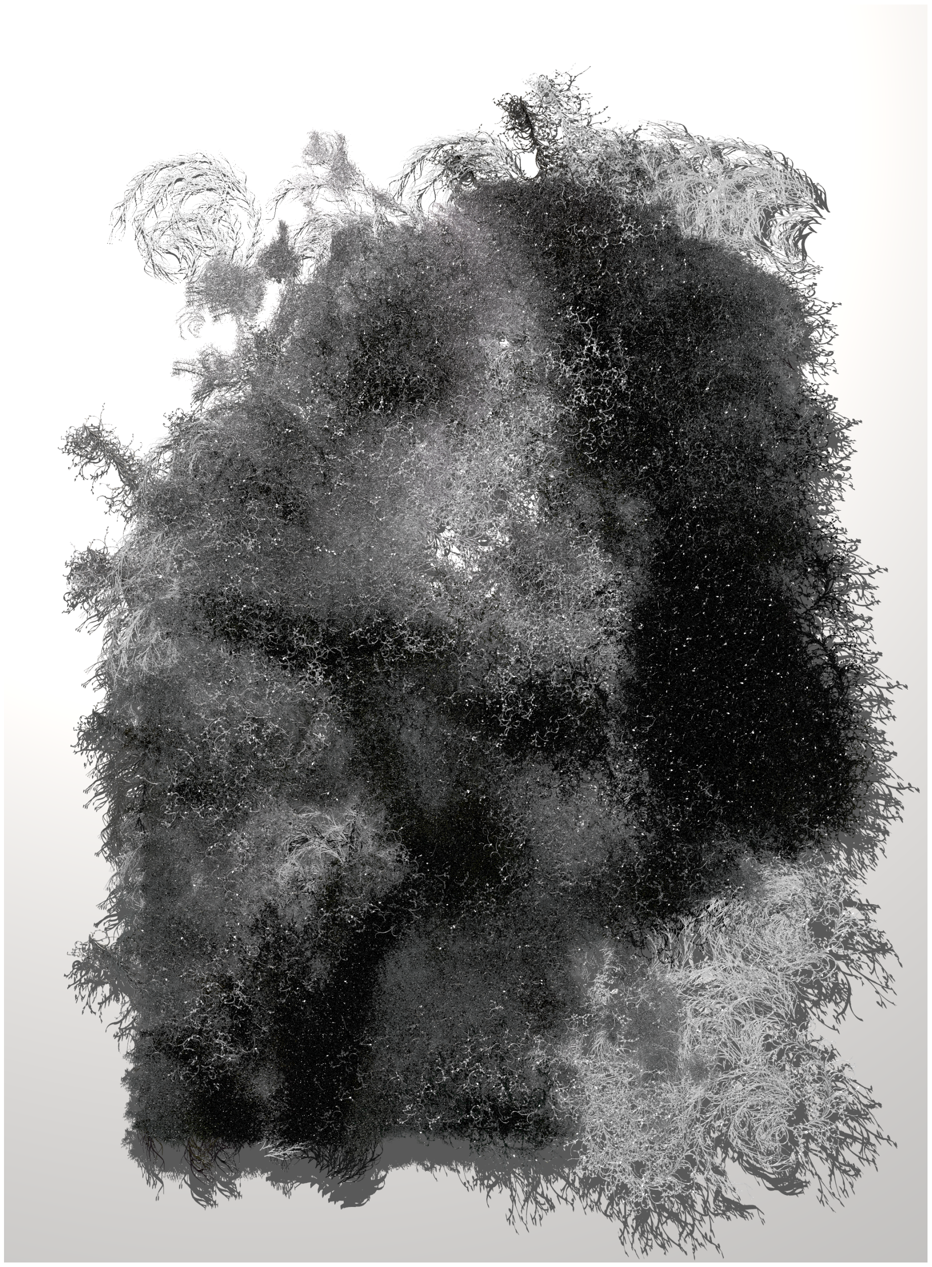


Fig. 12. 3200 × 1600 rendering of an individual of an intermediate population.

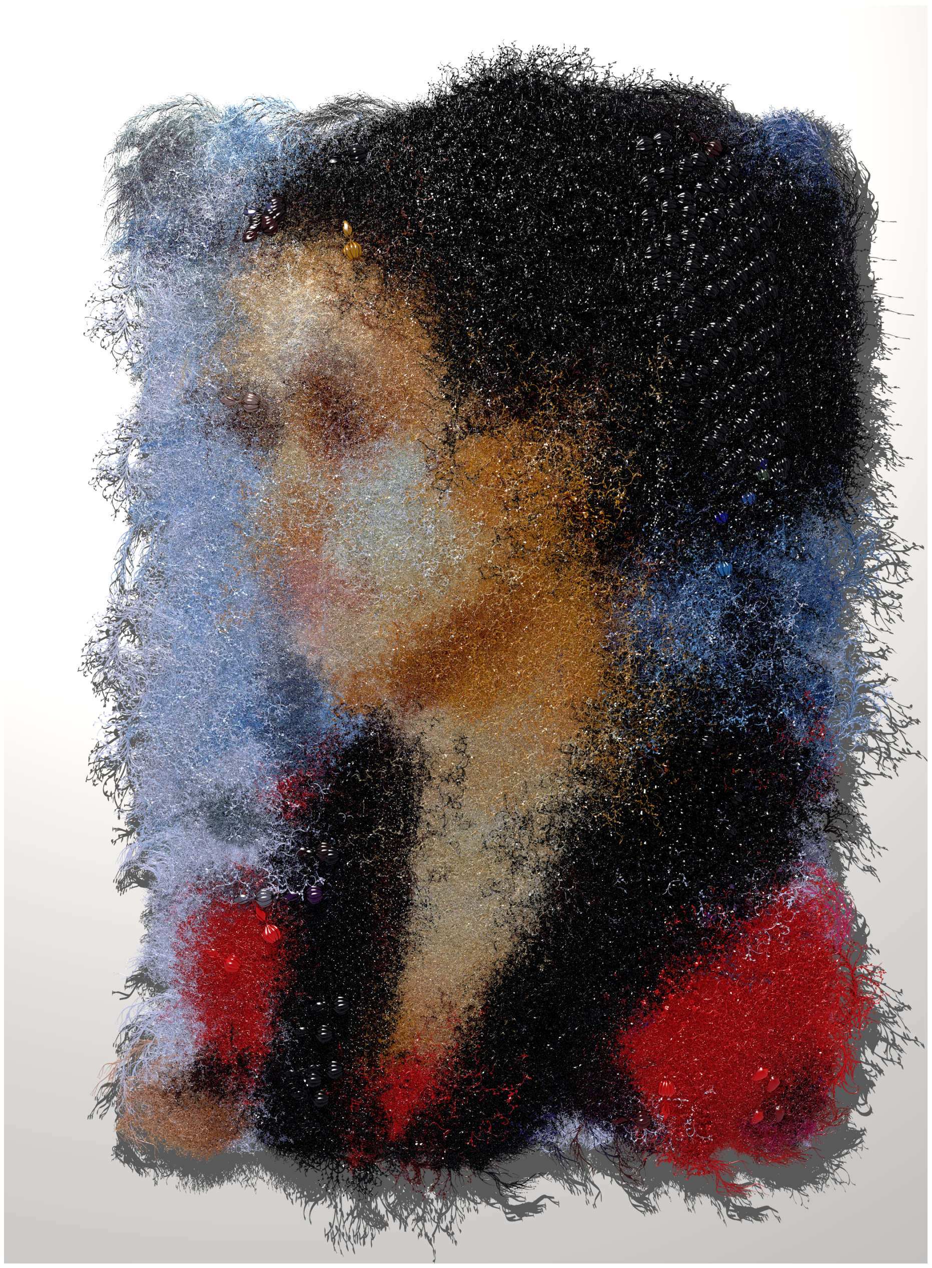


Fig. 13. 3200 × 1600 rendering of an individual of an intermediate population applied to a different source image.

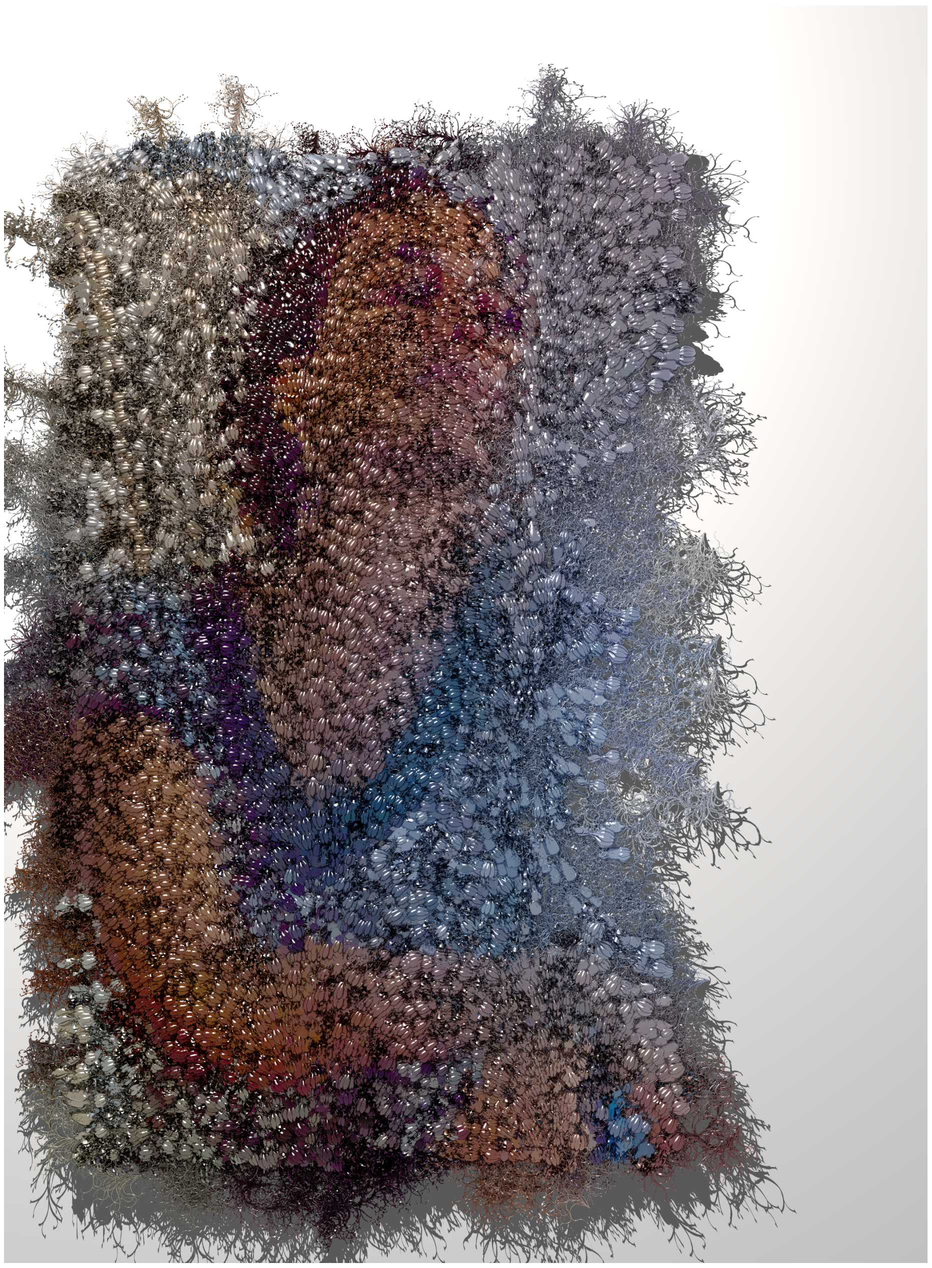


Fig. 14. 3200 × 1600 rendering of an individual of an intermediate population applied to a different source image.