Genetic Programming and Evolvable Machines (2023) 24:2
https://doi.org/10.1007/s10710-023-09449-z

Framework for unsupervised incremental evolution of stylized images

Florian Uhde[1]

Received: 30 July 2021 / Revised: 2 January 2023 / Accepted: 4 January 2023 /
Published online: 28 February 2023

Abstract
This paper examines and showcases a framework to generate artworks using evolutionary algorithms. Based on the idea of an incremental abstract artistic process stylized images are generated from different input images without human supervision. After explaining the underlying concept, the solution space of different styles is explored and its properties for style consistency and style variety are discussed. A first step towards better control of the outcome is implemented through masking, followed by a discussion about potential improvements and further research.

Keywords Genetic algorithm · Artistic rendering · Computational art

1 Introduction

Whereas evolutionary systems are often used to support optimization-focused, domain specific design tasks, the act of creating artistically pleasing artworks remains a challenge. Based on the idea of art creation as a series of composeable, stacked actions towards an desirable result, this work conceptualizes a framework of artistic creation. It uses a genetic algorithm and the means of evolution to produce artistic artifacts without human supervision. Using simple building blocks and their composition the algorithm exhibits a wide variety of parameters which allow to configure the emerging human-like painting process. A variety of different styles and expressions can be achieved, while each of those can be applied to different inputs, producing consistent results. The following section positions this work in the wider context of creative evolutionary systems and highlights similarities and core differences to existing approaches. Afterwards, Sect. 3 explains the concept and Sect. 4 explores aspects of its implementation.

✉ Florian Uhde
florian.uhde@posteo.de
https://www.inf.ovgu.de/en/

[1] Faculty of Computer Science, Otto-von-Guericke-University, Magdeburg, Germany

Fig. 1 Example results created within the scope of this work

Finally Conclusion and Future Work highlights some improvements and possible next steps (Fig. 1).

1.1 An incremental artistic process

The process mimicked by the algorithm in this paper defines the creation of art as an overlapping series of actions. An artist, striving to express an object o_{truth}, will do so by firstly perceiving it as o_{artist} and secondly expressing this representation of the object in a medium, creating o_{art}. The perception of the artist is shaped by a multitude of factors: Inner convictions, social surroundings, upbringing, education, ideology and so forth. This results in a transformation during the perception, turning o_{truth} into a personalized version of the object o_{artist}. When creating art based on o_{artist} the used medium and the skill of the artist influence the outcome o_{art}. The transformation of o throughout this process is what defines the style and signature of an artist. Both are embodied in the personalized way of viewing o_{truth}, as well as in the ability and limitations of the expression within the chosen medium.

While executing these transformations the artist takes a series of actions, each step being perceived as the current, most valuable one. As an example, when constructing a landscape painting, an artist might start with a rough composition, coloring large-scale features to give a backdrop and then further refine the outlines, adding details and fine grained shades further into the process. This abstract way of art creation is the foundation of the algorithm designed and explored in this work. The framework presented in this work simulates this process by splitting the image generation process into a series of brush strokes, that are optimized against a global fitness function. Different experiments are conducted to investigate the expressiveness and consistency of the developed system, to validate its usefulness as a starting point for further research.

2 Related work

At their core all evolutionary algorithms solve a search problem for a good candidate of a certain fitness function within a vast solution space. To achieve this, the algorithm utilizes two systems, one for creating and modifying such candidates, and

another for rating them in terms of their fitness. After already being used for various tasks in high-knowledge domains like architecture and engineering, supporting the human knowledge workers in different applications, their usage as generative systems for art was pioneered by Dawkins[1] and later on popularized by Sims [2] and Todd [3]. Today numerous systems exist that generate 2d image artifacts via various approaches [4–11]. In the context of this work the concept of creating two dimensional artworks can be divided into *imaginative*- and *interpretative*-systems (for a much finer classification see [12]).

Imaginative systems try to evolve and create the very object that should be expressed as an artwork, while *Interpretative systems* strive to reinterpret an existing object artistically. Examples from the first category are often expression based systems, modeling the generated picture as a set of functions [2, 13–15]. Approaches from the second category seek to replicate a given source image by *reinterpreting* it [4, 9, 16, 17], shifting the focus from the generation of an interesting object, towards an interesting interpretation. The algorithm in this work follows a *interpretative* approach, implementing a simplified artistic process. A common problem, given the vast solution space and the subjectivity of artworks, is the rating of candidates [18]. Evaluating the aesthetics of a generated image is hard using evolutionary algorithms, due to the complexity of fitness function that would incorporate the notion of aesthetics. The solution space also contains many undesirable results, either because of missing aesthetic features, or because they are unimpressive and just 'more-of-the-same'. One possible solution to this problem is to include human interaction in the design process [19]. Those *interactive evolutionary computing* systems are able to produce a variety of artifacts for images [14, 15, 19–21]. At the same time involving a human slows the generative process down and, due to the subjective nature, also comes at a cost for consistency and coverage [18, 22]. While the mentioned problem of efficient solution space exploration is less prevalent in interpretative systems, as the content of the painting is defined by the input image, the generation of an interesting and artistically pleasing result remains a difficult task. Fully automated algorithms struggle to identify visually interesting, so called salient, elements of the source image, something that humans easily do [23]. Failing to identify those elements and creating a painting by some form of uniform optimization [9], "tends to produce a machine-generated signature in the resulting painterly renderings" [23]. More recent approaches therefore "[...] trend away from use of local low-level image processing operators towards the incorporation of mid-level computer vision techniques in stroke placement heuristics" [23]. Those techniques include color segmentation [8], analysis of interest by eye-tracking [24] and image heuristics like salience mapping [25] or complexity [26] to guide the algorithm in the generative process. While the problem of salience is not addressed as directly as in other works [23, 25] it allows for some intrinsic benefits (see Sect. 4.2).

The approach of this work, outlined in the next section, draws inspiration from existing *interpretative* systems, especially the concept of composing the final image from a set of brush-strokes [4–6, 27], while this approach works with primed textures of strokes that are imprinted like stamps, rather than constructing the stroke curvature itself. Other implementations exist, which explore a similar direction [10, 11, 28]. Contrary to a global generation and optimization of a final image, this

works focus on a local, limited generation, combined with a global fitness function. This yields a composition of multiple optimized steps, which can only reach a certain fitness on their own, rather than a globally optimized result. Incorporating more complex measures to evaluate the fitness of the candidates [29, 30] may increase the perceived creativity [7] and overall complexity [26] of the results, as discussed in Sect. 5.

3 Evolutionary artistic rendering

As described in Sect. 1.1 the idea of this work is to transform the art of painting into an incremental optimization problem. Instead of optimizing a number of fully evolved candidates globally, it limits the optimization process to a number of sub routines, each optimizing up to a certain fitness ceiling, before expanding the solution space. By transforming the problem of *"What is the best candidate"* into *"What are the best next n steps to take"* the system mimics an incremental process, constructing the painting piece by piece. By restricting the possible actions for the algorithm to choose from, a consistent and expressive style can be created and applied to input images.

3.1 Overview of the system

This section explains the high level workings of the algorithm and shows the different parts and how they interact with each other. This serves as a foundation for the following sections.

The overall system used for the generative process is shown in Fig. 2: A source image (▪ ▪ ▪ ▪ ▪) and a configuration serve as the input of the system. To mimic the incremental approach described in Sect. 1.1 the system goes through a number of i **iterations**, each resulting in c strokes permanently added to the canvas (– – –). Every iteration starts with a fresh population of *potential* individual (▬▬), which are optimized through multiple **generations**, using the means of evolutionary algorithms explained in Sect. 3.2, to find the current best c strokes and imprints them onto the canvas. This process continues until a finishing criteria (total number of iterations, or overall fitness of the candidate image), is achieved. Given c strokes per iteration and i iterations the total number of strokes on the canvas after finishing the process is $i * c$.

3.2 Evolutionary algorithm

To generate a desirable image the process uses a genetic algorithm to optimize candidates. A decent familiarity with evolutionary, especially genetic algorithms, is assumed. A more complete introduction into genetic algorithms is given in [12] or [31]. This section features key areas of interests of the artistic process. First the

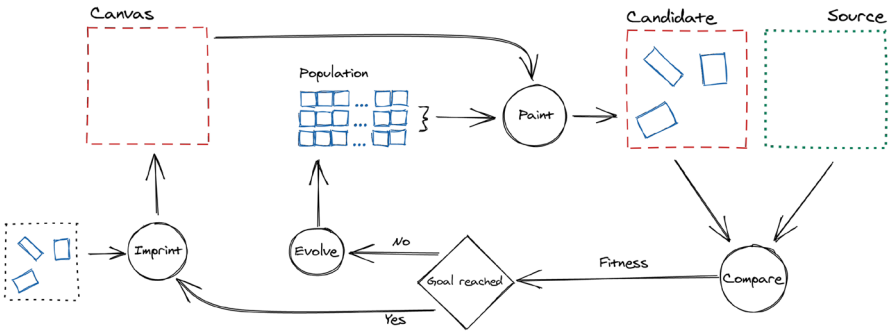

Fig. 2 High-level overview of the procedure

genotype and phenotype representation, defining how candidates are stored and rendered, are explained, then the manipulation and selection strategies and lastly the possible parameter space for the image generation, as well as some aspects of the implementation.

3.2.1 Genotype & phenotype

The genome for a single candidate consists of n sub-sections, each describing a single brush stroke, where n equals the number of strokes per candidate defined globally for the generation process. The maximal configuration of a single brush stroke part is shown in Fig. 3. As mentioned in Sect. 3.3, values that are removed from the evolutionary process, for example by fixing their value, are pruned and will not show up in the genome. Each gene provides the interpreter with a value between 0 and 1, which then in turn is translated to form the phenotype of a brush stroke. The phenotype of a single brush stroke, as shown in context in Fig. 4, translates the [0,1] values from the genotype into the transformation and shape to be imprinted on the canvas. Technically a brush stroke is a transparent quad, with a certain texture, which is scaled, rotated and positioned on the canvas. The position is encoded as a vector within the canvas object space. This means $(P_1, P_2) = (0, 0)$ corresponds to the lower left corner, (1,1) to the upper-right respectively. The quad can be rotated clockwise, e.g. a value $R_1 = 0.25$ yielding a 90 degree rotation. The quads uniform scale is defined by a base size (see Sect. 4.2), which is calculated by the overall algorithm and not evolved with the candidates. The scale gene can be used to deviate from this base size by $[-5\%, 5\%]$. This mechanic and its effect is explained further in Sect. 4.2. The actual texture content is selected from a texture array of size t. The array holds multiple brush strokes, allowing the algorithm to evolve the stroke textures used by each candidate, as shown in Sect. 4.1. To allow this, B_1 gets translated into an index of the array, indicating which texture to pick. If the array only contains a single texture (see Sect. 5.: *Cubic* brush pack) B_1 has no effect and is pruned from the genome. Lastly the texture selected is tinted by a RGB color defined by (C_1, C_2, C_3) multiplied by the transparency value of the texture.

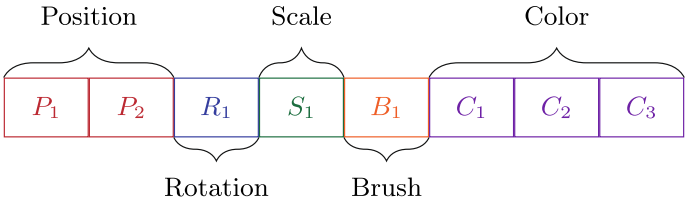

Fig. 3 Maximum genome configuration for a single brush stroke. Each box corresponds to one gene, which yields a value within [0,1]

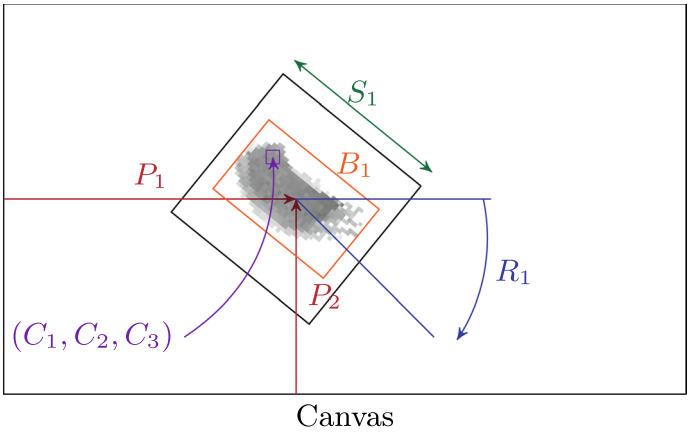

Fig. 4 Phenotype of a single brush stroke, with a certain color, placed and transformed on the canvas

3.2.2 Manipulation

One important part of evolutionary algorithms are the means of manipulating candidates. Two general concepts for this are *mutation* and *recombination*. The proposed algorithm implements rather simple variants of both mechanics to change the genome of candidates. *Mutation* is controlled by an overall mutation rate [0,1] expressing the chance to mutate for each candidate. If a mutation occurs a two step process selects a gene within the candidate and then a single bit within this gene to flip. While this method is quite robust and can execute reasonably fast, it has the drawback of high interdependence between the number of brush strokes, as well as the precision (see Implementation) of the genome representation. A higher number of strokes n gives a longer genome per candidate and as only a single mutation is carried out, the chance of each brush to be mutated is $\frac{1}{n}$. Furthermore, due to the binary representation of numbers, the position of each gene that is mutated has a strong influence on the resulting numerical change. A mutation on position c in our gene will change the value of the gene by $\frac{2^c}{maxValue}$, where *maxValue* depends on the chosen data type and is used to normalize the gene. The current implementation balances these shortcomings by having a generally high mutation rate of 90%. This allows for a fast exploration of the solution

space, which leads to quick convergence to sensible brushes, due to the high selection of the tournament selection. *Recombination* executes a uniform crossover strategy between two candidates A and B [32]. For each gene index of A and B a swap is performed with a likelihood of 50%, exchanging the gene of A with the gene of B and vice versa. As this crossover is performed on each individual gene it is independent of the genome length. As a single gene fully encodes a phenotype property as mentioned in Sect. 3.3, crossover will never change the value of a gene, instead only swap properties between two candidates.

3.2.3 Selection and fitness

The last component of the evolutionary algorithm is the selection of candidates to create the next generation for the population. This approach uses Tournament Selection [33]. This selection model provides a reasonably high selection pressure [34], a higher-than-average fitness in each child generation and beneficial implementation properties, due to the ability to run in parallel. The fitness function is used by the evolutionary algorithm to rate the performance of different individuals. In this work the fitness function compares the artifact generated by painting the candidates strokes onto the current canvas with the original image. A simple way of comparing two images is the negated sum of the difference between all pixels (Eq. 1).

$$1 - \frac{\sum_{x=0}^{width} \sum_{y=0}^{height} \Delta_{pixel}(candidate(x,y), source(x,y))}{width * height * maxDifference} \tag{1}$$

Equation 1: Fitness function, used to evaluate sets of brush strokes.

This work uses a slightly improved variation of Δ_{pixel} (see Eq. 2), scaling the difference of each color channel to approximate the visual sensitivity of human perception [35]. $\bar{r}$ is mean red value, ΔR,G and B are the Euclidean distances within each color channel. The Sect. 5 discussed various further improvements that can be made to this, which might unlock more sophisticated image analogies.

$$\Delta_{\text{pixel}}(C_1, C_2) = \sqrt{\left(2 + \frac{\bar{r}}{256}\right) \times \Delta R^2 + 4 \times \Delta G^2 + \left(2 + \frac{255 - \bar{r}}{256}\right) \times \Delta B^2} \tag{2}$$

Equation 2: Equation to calculate color distance, per pixel, between source and candidate image [35].

3.3 Implementation

The software is implemented in C# and HLSL[1] using the Unity3d Engine[2] for rendering and GeneticSharp [36] for the evolutionary optimization. The source code

[1] https://docs.microsoft.com/en-us/windows/win32/direct3dhlsl/dx-graphics-hlsl, accessed 27.10.2020.
[2] https://unity.com/, accessed 27.10.2020.

is publicly available at https://github.com/floAr/EvolutionaryArtistUnity, including Unity prefabs of the experiments used in the following section. The genome structure is a custom implementation to allow for fast normalization and adaptive pruning of genes, that are set to constant values. One important concept is that a single gene, for the purpose of genetic algorithms, holds multiple bits of data, which map to a single property of a brush stroke. This allows operations like mutation and crossover to either operate on bit level (operate the numeric value of a property) or gene level (operate on the whole property). By default gene values are represented using 16bit unsigned integers which are mapped from [0, 65.535] to [0,1],[3] resulting in a minimum step size of $1.52e^{-5}$ between possible values. If desired, this could be decreased to use 8bit unsigned integers, with a step size of $3.9e^{-2}$ and a smaller memory footprint, or increased up to 64bit, resulting in a step size of only $5.5e^{-20}$. The computation of heavy operations like painting a candidate, imprinting the canvas and comparing a candidate with the source image is implemented using shaders and executed on the GPU, to allow parallelization.

4 Evaluation

In this chapter the properties and artistic capabilities of the system are explored and evaluated. This work focuses on the expressiveness of the system (see Sect. 4.1), as it is fundamental to help its user to express an artistic idea. Figure 5 shows the input images used for the experiments. Each image is scaled to 512 by 512 pixels and used in the experiments without further modification. The first two images are paintings by Johannes Vermeer [37] and Vincent Van Gogh [38], the third and fourth images are photographs, with minor modifications (some houses on the horizon were cropped out) and the last image serves as a benchmark for stroke precision and color fastness.

In total four different brush texture packs were used: *Watercolor*, *Droplets*, *Cubic* and *Stroked*. The appendix lists the brush textures used in each pack.Unless specified otherwise the experiment settings used for the genetic algorithm are 50 iterations, with 7 strokes per candidate. This yields results made up from 350 individual strokes, as each iteration uses a population-size of 100 individuals per generation, each individual encoding 7 strokes. The evolutionary algorithm optimizes until the terminal condition of 20 generations of stagnant fitness is reached, then the winning individual is imprinted onto the canvas and a new iteration is started. Selection is done with *Tournament Selection* [33], with a tournament size of two genomes. Mutation is handled by flipping a random bit, the mutation chance starts at 90% and decreases by 10% every 50 generations, down to a minimum of 20% (see. Sect. 3.3 for the reasoning behind this). Crossover uses the *Uniform Crossover* strategy [32] with a fixed probability of 50%. For the experiment all features of a brush stroke are evolved as described in Sect. 3.2.1. Other default settings are base opacity value

[3] https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/builtin-types/integral-numeric-types, accessed 22.10.2020.

Fig. 5 Input images used. From left to right: Girl with a Pearl Earring [37], Wheat Field With Cypresses [38], Photos of a dog under tree and a church (2020) and benchmark image

for each brush, set as $\alpha = 0.7$ (70% opacity) and a brush base size which interpolates between 0.8 (which corresponds to 80% of the canvas) and 0.025 over all iterations. The functions used to interpolate are shown in Fig. 9 and their effect evaluated in Sect. 4.2. While adding those features to the evolutionary set widens the search space, the decision to fix them for most experiments was made to limit the scope and increase the comparability of the results. The generative process starts from a white, or black canvas, depending on the overall background color of the source image. All experiments with their parameters can be seen in Fig. 17. A recording, showing the setup and a run of the algorithm, is available at https://youtu.be/KEuT2mphq0w.

4.1 Expressiveness

A core requirement of an artistic system is the ability to express the users desired artistic outcome. Therefore this section evaluates the expressiveness of the algorithm. For this, two different aspects are to be considered: *Style Variety*, which is the scope of different styles that can be generated, as well as *Style Consistency*, which means generating consistent results, when applying a selected style to different input images. Both features are important to model the signature and style described in Sect. 1.1. To model a signature and style of an artist the system needs to be able to generate diverse, but also consistent styles.

4.1.1 Style variety

The simplest way of creating different styles is adjusting the brushes used to paint the image. In Fig. 6 three different examples of possible styles, which vary only in the selected brush texture, are shown on four different images. Each row shows how a single image can be represented differently by changing the brush texture, creating variety in plasticity and style, while preserving the content. This is in line with [4] which found that brush characteristics are a mayor factor influencing the outcome of the generative process.

The first two columns show how relatively similar brushes, both are opaque and laminar, can invoke very different detail textures. The difference in shape (stretched vs circular) as well as in border structure (smooth vs rigged) translate nicely into the structure of the painting, without creating noticeable artifacts. The third column shows how far a style can differ, given a more diverse brush set. The thin strokes

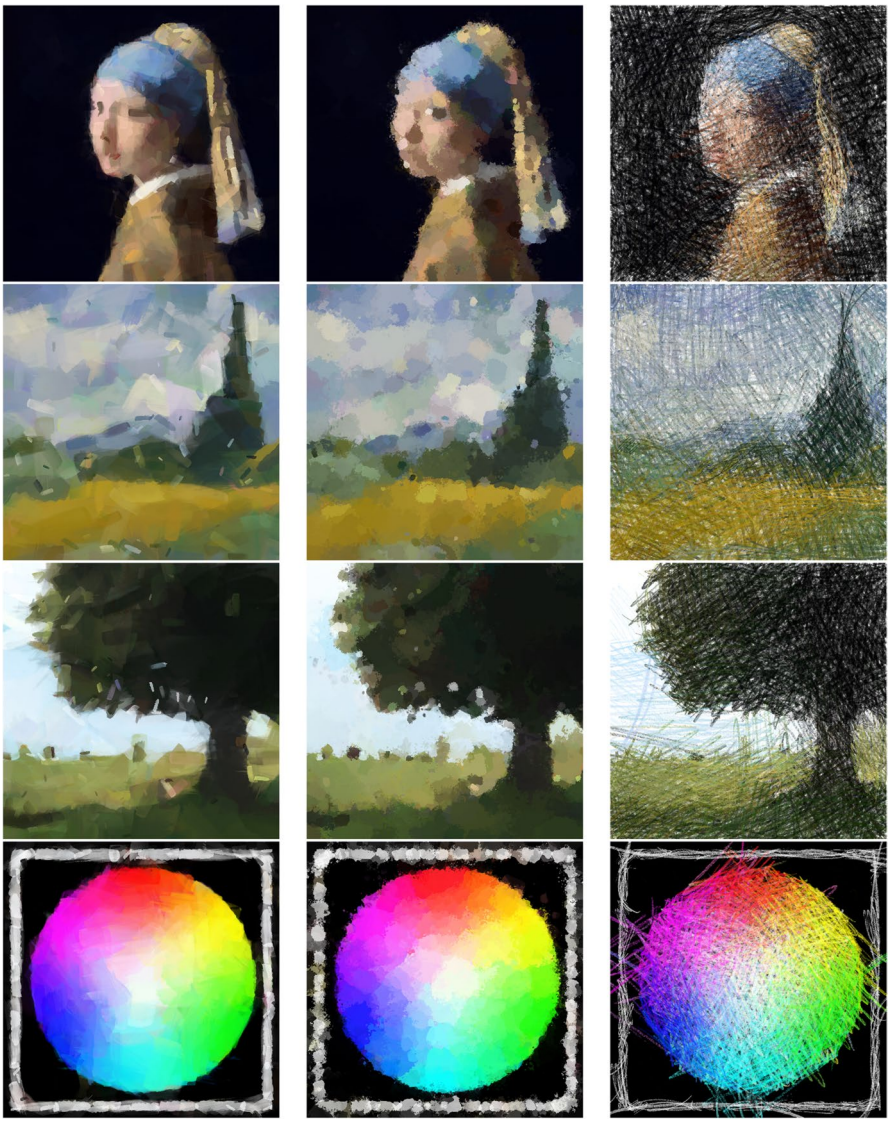

Fig. 6 Different brushes yield varying results. From left to right: The *Watercolor*, *Droplet* and *Strokes* packs were used (see Sect. 5)

create a hatched and sketchy look, overshooting the content target (especially noticeable in the benchmark image on the very bottom), as their inherit error for this is way smaller compared to more laminar brushes.Other means of variation are shown in Fig. 7: Using the same brush texture different images can be generated by setting other constrains. The first column shows the image generated by the *Cubic* brush packs, which uses a single, white square as the only brush texture. The upper image uses the default configuration and is able to replicate the image quite well using

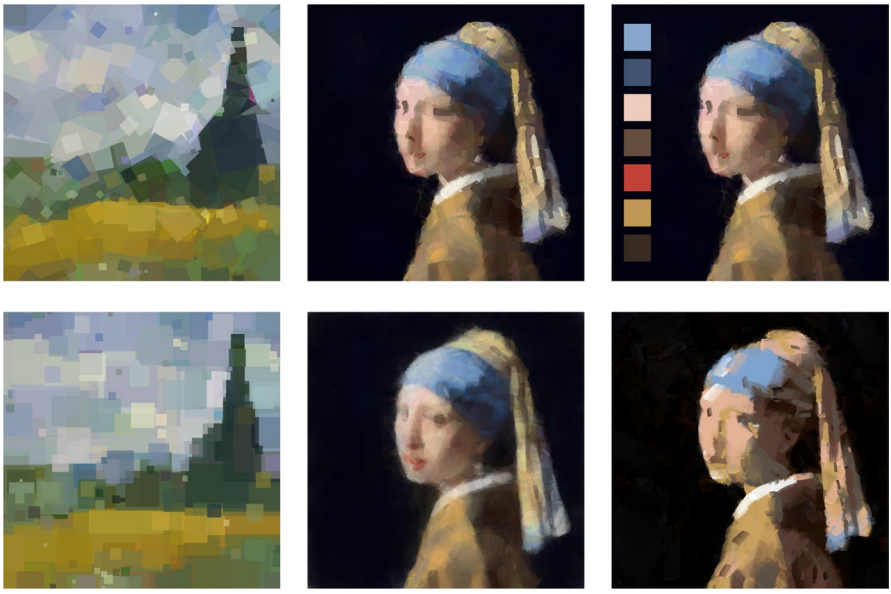

Fig. 7 Example of fine grained style variation. The upper image is the 'default', the lower one a more restricted result. The leftmost column has constrained rotation, the center one uses a lower transparency value and the right one is limited to a set of seven colors

rotation and scale to vary the single texture. The lower image shows a result with fixed rotation. In this case the R_1 gene (see Sect. 3.2.1) is removed for each brush and instead provided as a fixed value. This leads to axis aligned blocks, which can not approximate the geometry of the source as well as before, yielding a mosaic like style. The center column shows the difference between the default opacity value (70%) in the first row and 20% opacity below. The image with lower alpha, albeit using the same brush pack, appears smoother and more continuous. The last column shows the default result for the *Watercolor* brush and a variant where the color genes where removed below. Instead a single gene was used in the genome to evolve a one-of-n color selection. Just like brush selection this allows the algorithm to only evolve colors from a predefined set. The color space consists of seven colors sampled from the image (shown in upper image) as well as black and white. Furthermore the α for the brushes was fixed on 100% opacity for this experiment, to prevent color mixing due to blending. This limitation results in a posterized look of the image, bringing forth sharp contrasts and cutting smaller features due to missing means of approximation. This examples show the capabilities of the system to express multitudes of styles given the possible combinations of different restrictions.

4.1.2 Style consistency

Given the ability to generate a variety of different styles, enabling a wide spectrum of styles to realize a custom reinterpretation, consistency is as important when

considering an artistic tool. As described in Sect. 1.1 the combination of both factors allow to mimic the transformative process of creating an artwork from an internal representation. Looking at Fig. 6 each column shows that a style produces consistent results over a variety of different images, creating an equable look and feel. Between different types of input images, style elements remain noticeable and create a recognizable set of interpretations. This effect increases as the difference between the used style restrictions does. The more regulated and therefore specific a certain style is, the easier it is to recognize those peculiarities in the generated artifacts.

4.2 Image saliency

Within approaches that aim to transform an input image into stylized artefacts, a common problem is detection of salient regions. One way to counteract this problem is to employ more complex operators in the fitness function, like higher order computer vision mechanics [8, 25] to preprocess the image, or to be used in the fitness function. This approach instead makes use of emerging properties of the image generation itself: The way an image is constructed by the algorithm resembles the construction of a painting as done by humans (see Sect. 1.1). This behaviour emerges because of two properties of the algorithm: By only being able to place a limited number of brush strokes the algorithm has a fitness ceiling for each iteration. Given seven strokes in the first iteration the target image can only be approximated to a certain amount. This leads to the construction of the artwork from coarse to fine, as filling the most erroneous large areas will yield the highest fitness gain. Furthermore the adjustment of the brush size over the course of the process strengthens the behaviour to start with larger features and move to more detailed adjustments later on, starting already from a more sophisticated representation. This progression from coarse to fine features forces the algorithm to only add smaller details, after the overall color composition has been already executed, which favors regions with smaller details in later iterations. This behaviour is visible in Fig. 8, which shows the canvas at different stages in the evolution process. The first iteration provides the overall shading, as the seven brush strokes are used to cover the white background. Four iterations later the outline of the image is roughly sketched, and the following iterations keep on refining the outline and adding finer shading. After 50 iterations the overall outline of the image is clearly defined, with the latest and smallest strokes adding highlights and details. While this does not directly map to salience in all cases, it puts focus onto adding high contrast details, which correlates with visually highly interesting regions [23].

A more detailed examination of different brush size progressions (Fig. 9) and its influence on the generated result can be seen in Fig. 10. The target image contains many details, such as small shaded areas, color gradients and tiny features, which makes it hard to replicate truthfully. Between the three results the only difference is how the brush size was interpolated between 0.8 and 0.025. Equation 5 shows an

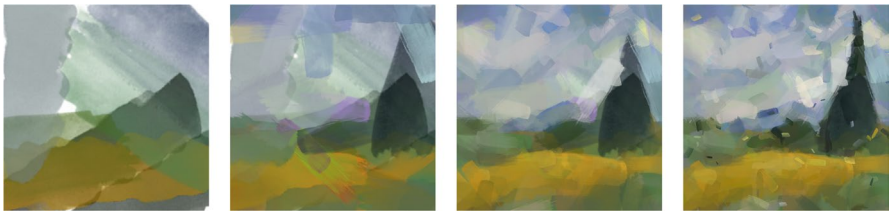

Fig. 8 Different stages of the generation of a single image. Left to right: 1, 5, 25 and 50 iterations

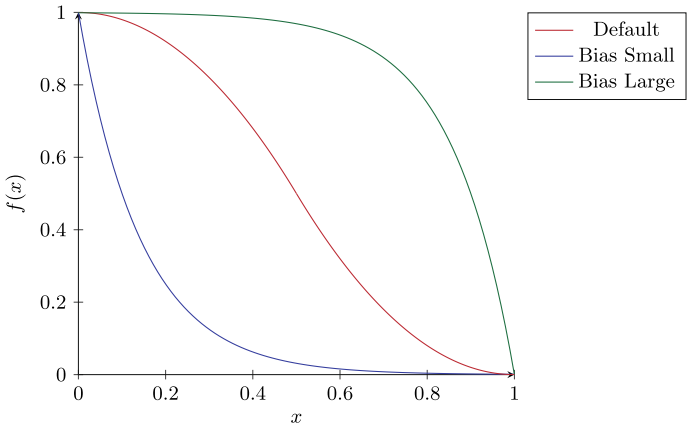

Fig. 9 Interpolation functions to calculate brush size

approximation three different lerp-functions used.[4] the brush size is calculated for each iteration i by interpolating between the maximum and minimum size. In Fig. 10 the results of different lerp-functions are shown: Image (A) uses the *Bias Small* function (blue), which converges fast to small brush sizes, (B) uses the *Default* function (red) and image (C) uses the *Bias Large* function (green), with a focus on larger brush sizes.

$$blue = 1 - \sqrt{2 - x} + \sqrt{x} \tag{3}$$

$$red = \frac{1}{2}(1 + \cos \pi + x) \tag{4}$$

$$\text{green} = \sqrt{1 - x^2} \tag{5}$$

[4] Those functions are Unity specific implementations so the actual implementation might vary slightly

Fig. 10 Influence of different size compositions. From left to right: **a** Bias towards small brush strokes, **b** default and **c** bias towards large brush strokes

Equation 5: Approximations of different lerp functions for scaling the brushes.

The results show the effect of brush size on plasticity, with the smaller brushes creating a very 'rough' surface, whereas the large brushes tend to blend together, exhibiting smoother gradients and softer edges. By adjusting the brush stroke sizes available to the algorithm certain focus on detail can be triggered, as the smaller strokes tend to embed smaller features into the design, as their effect to the fitness function is greater on small, high contrast features, than on large areas.While this helps to alleviate some of the problems in regards to missing salience detection, it does not solve the problem in itself, as all these intrinsic optimizations operate on a uniform level. This can be seen in Fig. 6, when comparing the landscape painting with the portrait: While the landscape painting has a relatively even distribution of salience, the portrait has specific details, like the eyes and lips, which are not captured well with this approach. A hybrid is the photo of the dog under the tree where the algorithm adds detail to the landscape, but also fails to add enough detail to the dog itself, reducing its presence in the final outcome.

4.3 Masking

Further guidance of the process is achievable, by masking important regions of the image. Masking results in an adjustment of the error values, which are multiplied by a value between 0 and 1, therefore shifting the importance of those regions accordingly. A value of 1 (white) means the error produced in this pixel is facilitated fully into the overall error, whereas a value of zero (black) would remove this pixel completely from the error metric, effectively allowing the algorithm to fill this pixel with any color, without any effect for the evolutionary algorithm. This technique is especially valuable on source images with a high noise, as it allows the artist to specify precise points of interest, preventing the process to get stuck on local detail, that are not adding to the overall, desired appearance of the result. If a mask is provided, the fitness function in Sect. 3.2.3 is multiplied by the r channel of the mask texture, as shown in Eq. 6.

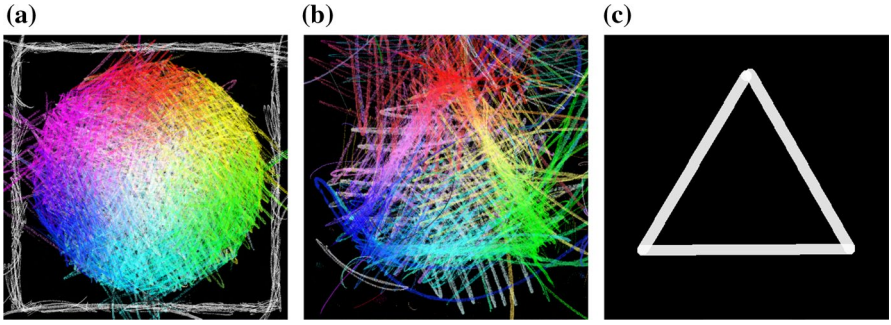

Fig. 11 Adding a mask to the process allows to focus on specific regions. From left to right: **a** Unmasked, **b** result using the same parameters as well as a mask, **c** mask used

$$1 - \left(\frac{\sum_{x=0}^{width} \sum_{y=0}^{height} \Delta_{pixel}(candidate(x,y), source(x,y)) * mask(x,y)}{width * height * maxDifference} \right) \tag{6}$$

Equation 6: Fitness function, with additional masking.

In Fig. 11 the result of a masked generation is shown: Given the binary nature of the mask the process ignores the region that is not masked in white completely and tries to put as many strokes on the triangle outline as possible. While this provides effective means of shaping the focus of the algorithm towards the source image it remains a complex task to establish mask for more complex source images.

Figure 12 shows more complex images produced used the same setting as in Fig.6, but facilitating different masks. Depending on the mask the process exhibits a multitude of properties that might be desirable. An interesting comparison is between the leftmost and center column. The left mask is a rough manual annotation, whereas the center mask was generated by edge detection. It can be observed that with the center mask it is harder to find spots where brushes can fit easily, as the many, spread out high-value pixels have a larger influence on the error compared to the 50% black error in the manual annotation. This leads to many brush strokes parking in the black areas of the image when it the process can not find a good candidate to improve the image and therefore discards the brush stroke by moving it into an area where it does not influence the error metric. At the same time the noisy mask generated by edge detection provides better stability of the brushes around the focal point of the image, which produces a more detailed texture of the main object.

Vastly different results can be achieve by removing error restraints from areas that are defining parts of the source image. The rightmost image in Fig. 12 shows the source image with most of the center ignored for error calculation. The mask only focuses on the background, with a slight noise filter to blend the mask a few pixels into the depiction of the woman. This leads to a chaotic assembly of color, which maintains the shape of the primary object and stills hints at the original by bleeding in the original colors from the outline. As shown in this section masking the generative process allows to apply, as well as relax, constraints of the image generation process, leading to interesting, sometimes unexpected results. While this pays into the expressiveness

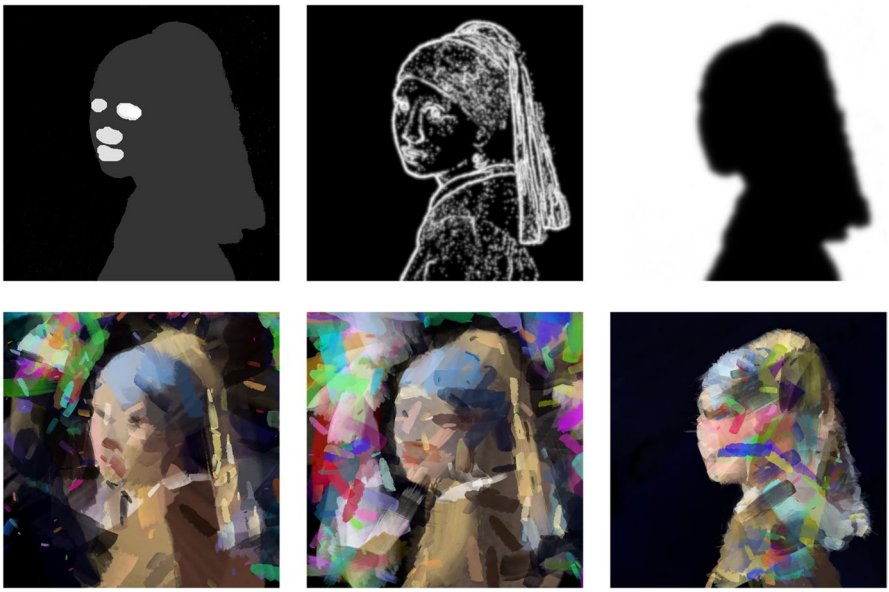

Fig. 12 Different effects of multiple masks (Top row) and the corresponding results (Bottom row). Besides the mask all settings are equal to *Watercolor* from Fig. 6

of the system it is only partly suitable to guiding the artistic process and solving the problem of image saliency [25]. This could be improved further by adjusting the mask during generation by a human actor, to shift focus of the process while detail emerge.

5 Conclusion and future work

This work has shown a generative system based on an abstract art generating process and a genetic algorithm. Different features have been explored and evaluated in regards of their possible solution space and shortcomings. As mentioned by Collomosse [23] detection of salient regions is a core feature of unsupervised generative systems. While this work employs techniques to improve detail placement for uniformly salient images, it remains a problem for source images with small salient regions. Masking of the images strikes a middle-ground by front loading some design by human actors before the creative process, but can only solve the salient problem to a limited extend. Other techniques exist and can be paired with this approach, like manually authored guidance maps for stroke placement [28] or higher order computer vision methods, like edge detection or color segmentation [8, 24, 25]. Recent advances in neural network research allow for high fidelity and automated extraction of salient regions [39]. Other neural network research areas like style transfer [40, 41] might also provide benefits and allow for interesting results by generating more complex brush strokes and enriching the details of the final artwork. Conceptually this work is based on an abstract art generating

process (see Sect. 1.1) but focuses mainly on the second transformation, the expression of the internal representation into a medium. The first transformation, the personalized perception of objects is modelled by the calculation of the fitness function as it, defines how the system (artist) can perceive the ground truth (input image). By basing the fitness only on pixel errors, the algorithm always compares against o_{truth}, our input image itself. A more sophisticated method could also introduce additional artistic traits and use preprocessing like segmentation [5, 30], or extracting latent vectors [42], to achieve a higher order of 'understanding' of the source image . Various error metrics, like Wasserstein distance [43], complexity measures [26] or style transfer loss metrics [40] allow the comparison of images within a higher order space and could yield artifacts that go beyond simple pixel similarity [11]. The current system operates under a one-shot model: After configuring the algorithm it runs without any human interaction. While this proves to be beneficial for runtime, it severely limits the flexibility of the system to adapt to the artists desires [18, 21, 44]. Penitential extension of the systems could transform it more into a interactive operation, giving the user the ability tow pause the generate and augment the current state. A non exhaustive list of ideas include live manipulation of the mask, shifting focal areas during the run, pinning and deleting of individual strokes and even painting directly on the canvas. The aim with this augmentations is to build up a conversation between the user and the algorithm, to slowly adjust the objective o_{truth} (pixel distance) into a personalised o_{truth} personalized to the user themselves. Further improvements can be made in regards to the current implementation of the system. The Sect. 3.2.2 mentions the high interdependence between genome length and mutation chance, especially with the brush texture. Normalizing those values would allow to reuse settings between different styles more robustly and lessen the required user input when exploring different interesting styles. Another aspect is the continuity of the generation process. Iterations are decoupled from each others, the algorithm has no means to determine how many of them happened already, or how many are left. Currently each iteration starts with an imprinted canvas, onto which the strokes of the best candidates are added. In regards of the candidates itself, 'catastrophic forgetting' [45] occurs after each generation. Instead of imprinting only the best candidates, it would also be possible to build the stack of actions in memory, enabling parallel processing of the best n candidates of each generation. While this will exponentially increase memory consumption and time, it will allow the buildup of more complex patterns, that take several layered brush strokes, and therefore more iterations to evolve. Given these improvements, different paths ahead are possible, reinforcing the autonomous capabilities of the system presented in this work, or fusing them with user controlled input and turning it into a semi-supervised content creation tool. Other exotic use cases can be found in emerging properties of the generated artifact. One particular exotic use case would be compression of source images, given that a genome representation of an image is roughly 10 times smaller than its pixel data. Yet to enable those use cases more sophisticated methods of salience detection and non-uniform detail preservation are to be implemented .

Appendix

Brush packs

All brush textures, unless stated otherwise, where created for this work using Gimp 2.10.4[5]. Figures 13, 14, 15, 16, 17.

Fig. 13 *Watercolor* from [28]

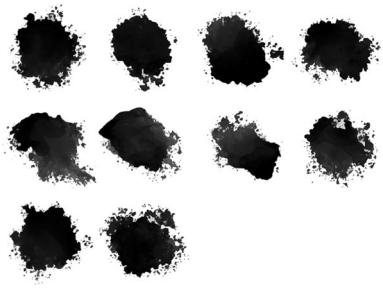

Fig. 14 *Droplets* from [46]

Fig. 15 *Strokes*, ballpoint pen and scanned

Fig. 16 *Cubic*

[5] https://www.gimp.org/, accessed 28.10.2020.

Experiment	Resolution	Image	Brushes	Iterations	Strokes	Mutation (%)	Crossover (%)	Termination Criteria (Fitness stagnation)	Base Size [Max,Min]	Size lerp function	Opacity	Canvas Color	Mask	Special
Brush 1-1	512x513	Girl with a Pearl Earring	Watercolor	50	7	90	50	20 generations	[0.8,0.025]	1-(-(cos(PI * x) - 1) / 2)	0.7	#000000	-	-
Brush 1-2	512x514	Girl with a Pearl Earring	Droplets	50	7	90	50	20 generations	[0.8,0.025]	1-(-(cos(PI * x) - 1) / 2)	0.7	#000000	-	-
Brush 1-3	512x515	Girl with a Pearl Earring	Strokes	50	7	90	50	20 generations	[0.8,0.025]	1-(-(cos(PI * x) - 1) / 2)	0.7	#000000	-	-
Brush 2-1	512x516	Wheat Field With Cypresses	Watercolor	50	7	90	50	20 generations	[0.8,0.025]	1-(-(cos(PI * x) - 1) / 2)	0.7	#FFFFFF	-	-
Brush 2-2	512x517	Wheat Field With Cypresses	Droplets	50	7	90	50	20 generations	[0.8,0.025]	1-(-(cos(PI * x) - 1) / 2)	0.7	#FFFFFF	-	-
Brush 2-3	512x518	Wheat Field With Cypresses	Strokes	50	7	90	50	20 generations	[0.8,0.025]	1-(-(cos(PI * x) - 1) / 2)	0.7	#FFFFFF	-	-
Brush 3-1	512x519	Dog under Tree	Watercolor	50	7	90	50	20 generations	[0.8,0.025]	1-(-(cos(PI * x) - 1) / 2)	0.7	#FFFFFF	-	-
Brush 3-2	512x520	Dog under Tree	Droplets	50	7	90	50	20 generations	[0.8,0.025]	1-(-(cos(PI * x) - 1) / 2)	0.7	#FFFFFF	-	-
Brush 3-3	512x521	Dog under Tree	Strokes	50	7	90	50	20 generations	[0.8,0.025]	1-(-(cos(PI * x) - 1) / 2)	0.7	#FFFFFF	-	-
Brush 4-1	512x522	Benchmark	Watercolor	50	7	90	50	20 generations	[0.8,0.025]	1-(-(cos(PI * x) - 1) / 2)	0.7	#FFFFFF	-	-
Brush 4-2	512x523	Benchmark	Droplets	50	7	90	50	20 generations	[0.8,0.025]	1-(-(cos(PI * x) - 1) / 2)	0.7	#FFFFFF	-	-
Brush 4-3	512x524	Benchmark	Strokes	50	7	90	50	20 generations	[0.8,0.025]	1-(-(cos(PI * x) - 1) / 2)	0.7	#FFFFFF	-	-
Limit 1	512x525	Wheat Field With Cypresses	Cubic	50	7	90	50	20 generations	[0.8,0.025]	1-(-(cos(PI * x) - 1) / 2)	0.7	#FFFFFF	-	Fixed rotation
Limit 2	512x526	Girl with a Pearl Earring	Watercolor	50	7	90	50	20 generations	[0.8,0.025]	1-(-(cos(PI * x) - 1) / 2)	0.2	#000000	-	-
Limit 3	512x527	Girl with a Pearl Earring	Watercolor	50	7	90	50	20 generations	[0.8,0.025]	1-(-(cos(PI * x) - 1) / 2)	0.2	#000000	-	Color limited to 7 main colors from source image
Size 1	512x528	Church	Watercolor	50	7	90	50	20 generations	[0.8,0.025]	1-(sqrt(1 - pow(x - 1, 2)))	0.2	#FFFFFF	-	-
Size 2	512x529	Church	Watercolor	50	7	90	50	20 generations	[0.8,0.025]	1-(-(cos(PI * x) - 1) / 2)	0.2	#FFFFFF	-	-
Size 3	512x530	Church	Watercolor	50	7	90	50	20 generations	[0.8,0.025]	sqrt(1 - pow(x, 2))	0.2	#FFFFFF	-	-
Mask 1-1	512x531	Benchmark	Strokes	50	7	90	50	20 generations	[0.8,0.025]	1-(-(cos(PI * x) - 1) / 2)	0.2	#000000	-	-
Mask 1-2	512x532	Benchmark	Strokes	50	7	90	50	20 generations	[0.8,0.025]	1-(-(cos(PI * x) - 1) / 2)	0.2	#000000	Triangle	-
Mask 2-1	512x533	Girl with a Pearl Earring	Watercolor	50	7	90	50	20 generations	[0.8,0.025]	1-(-(cos(PI * x) - 1) / 2)	0.2	#000000	Face details	-
Mask 2-2	512x534	Girl with a Pearl Earring	Watercolor	50	7	90	50	20 generations	[0.8,0.025]	1-(-(cos(PI * x) - 1) / 2)	0.2	#000000	Outlines	-
Mask 2-3	512x535	Girl with a Pearl Earring	Watercolor	50	7	90	50	20 generations	[0.8,0.025]	1-(-(cos(PI * x) - 1) / 2)	0.2	#000000	Cutout	-

Fig. 17 Detailed configuration of all experiments

Funding Open Access funding enabled and organized by Projekt DEAL. No funding was received for conducting this study.

Declarations

Conflict of interest All authors certify that they have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript.

References

1. R. Dawkins, *The Blind Watchmaker - Evidence of Evolution Reveals a Universe without Design* (WW Norton & Company, 1986)
2. K. Sims, Artificial evolution for computer graphics, in *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '91, Association for Computing Machinery, New York, NY, USA (1991), p. 319-328 https://doi.org/10.1145/122718.122752http://www.karlsims.com/papers/siggraph91.html
3. S. Todd, W. Latham, Evolutionary art and computers. Academic Press, Inc., USA (1992), http://portal.acm.org/citation.cfm?id=561831
4. P. Haeberli, Paint by numbers: abstract image representations. Comput. Gr. (ACM) **24**(4), 207–214 (1990). https://doi.org/10.1145/97880.97902
5. M. Valstar, S. Colton, M. Pantic, Emotionally aware automated portrait painting. Belgian/Netherlands Artificial Intelligence Conference (2008), pp. 407–408
6. A. Hertzmann, A survey of stroke-based rendering. IEEE Comput. Gr. Appl. **23**(4), 70–81 (2003)

7. J. Correia, P. Machado, J. Romero, P. Martins, F. Amílcar Cardoso, Breaking the mould an evolutionary quest for innovation through style change (2019)
8. B. Gooch, G. Coombe, P. Shirley, Artistic vision: painterly rendering using computer vision techniques. NPAR symposium on non-photorealistic animation and rendering (10 2003)
9. A. Hertzmann, Paint by relaxation, in *Proceedings of Computer Graphics International Conference, CGI* (2001), pp. 47–54
10. S. Shahriar, Procedural paintings with genetic evolution algorithm (2020), https://github.com/IRCSS/Procedural-painting
11. Z. Huang, S. Zhou, W. Heng, Learning to paint with model-based deep reinforcement learning, in *Proceedings of the IEEE International Conference on Computer Vision* 2019-Octob, 8708–8717
12. Z. Huang, S. Zhou, W. Heng, Learning to paint with model-based deep reinforcement learning, in *Proceedings of the IEEE International Conference on Computer Vision* 2019-Octob, 8708–8717
13. T. Unemi, SBART 2.4: Breeding 2D CG images and movies and creating a type of collage, in *International Conference on Knowledge-Based Intelligent Electronic Systems, Proceedings, KES*. (1999), pp. 288–291
14. S. Rooke, Chapter 13 - Eons of Genetically Evolved Algorithmic Images. In: Bentley, P.J., Corne, D.W.B.T.C.E.S. (eds.) *The Morgan Kaufmann Series in Artificial Intelligence*. Morgan Kaufmann, San Francisco (2002), pp. 339–365 http://www.sciencedirect.com/science/article/pii/B9781558606739500525
15. D. Hart, Toward greater artistic control for interactive evolution of images and animation, in *ACM SIGGRAPH 2006: Sketches, SIGGRAPH '06*. ed. by M. Giacobini (Springer, Berlin Heidelberg, Berlin, Heidelberg, 2006), pp.527–536
16. A. Hertzmann, C.E. Jacobs, N. Oliver, B. Curless, D.H. Salesin, Image analogies, in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques SIGGRAPH* 01 2001(August), 327–340
17. A. Hertzmann, C.E. Jacobs, N. Oliver, B. Curless, D.H. Salesin, Image analogies, in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques SIGGRAPH* 01 2001(August), 327–340
18. J. McCormack, Open problems in evolutionary music and art. Tech. rep. (2005), http://www.csse.monash.edu.au/~jonmc
19. H. Takagi, Interactive evolutionary computation: fusion of the capabilities of EC optimization and human evaluation. Proc. IEEE **89**(9), 1275–1296 (2001)
20. P. Machado, A. Cardoso, All the truth about NEvAr. Appl. Intel. **16**(2), 101–118 (2002)
21. J. Secretan, N. Beato, D.B. D'Ambrosio, A. Rodriguez, A. Campbell, J.T. Folsom-Kovarik, K.O. Stanley, Picbreeder: a case study in collaborative evolutionary exploration of design space. Evolut. Comput. **19**(3), 373–403 (2010). https://doi.org/10.1162/EVCO_a_00030
22. P.J. Bentley, D.W. Corne, in *An introduction to Creative Evolutionary Systems*. eds by P.J. Bentley, D.W.B.T.C.E.S. Corne, The Morgan Kaufmann Series in Artificial Intelligence, Morgan Kaufmann, San Francisco (2002), pp. 1–75 http://www.sciencedirect.com/science/article/pii/B9781558606739500355
23. J.P. Collomosse, in *Evolutionary Search for the Artistic Rendering of Photographs*. ed. by J. Romero, P. Machado, The Art of Artificial Evolution, Springer Berlin Heidelberg, Berlin, Heidelberg (2007), pp. 39–62https://doi.org/10.1007/978-3-540-72877-1_2
24. A. Santella, D. Decarlo, Abstracted painterly renderings using eye-tracking data. NPAR Symposium on Non-Photorealistic Animation and Rendering (7 2002)
25. J.P. Collomosse, P.M. Hall, Genetic paint: a search for salient paintings. Lect. Notes Comput. Sci. **3449**, 437–447 (2005)
26. P. Machado, J. Romero, M. Nadal, A. Santos, J. Correia, A. Carballal, Computerized measures of visual complexity. Acta Psychologica **160**, 43–57 (2015). https://doi.org/10.1016/j.actpsy.2015.06.005
27. S. Colton, Stroke Matching for Paint Dances. Computational Aesthetics 2010: Eurographics Workshop on Computational Aesthetics in Graphics, Visualization and Imaging (Victoria, British Columbia, Canada, May 28–30, 2009), pp. 67–74
28. A. Opara, Genetic drawing project (2020), https://github.com/anopara/genetic-drawing
29. P. Torres, C. Simon, LNCS 5484 - Evolving approximate image filters. Appl Evolut Comput 1–11 (2009), http://link.springer.com/chapter/10.1007/978-3-642-01129-0_53
30. S. Colton, Automatic invention of fitness functions with application to scene generation. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 4974 LNCS, 381–391 (2008)
31. C.G. Johnson, J.J.R. Cardalda, Genetic algorithms in visual art and music. Leonardo **35**(2), 175–184 (2002). https://doi.org/10.1162/00240940252940559

32. G. Syswerda, Uniform crossover in genetic algorithms, in *Proceedings of the 3rd International Conference on Genetic Algorithms*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1991), pp. 2–9
33. D.E. Goldberg, K. Deb, in *A Comparative Analysis of Selection Schemes Used in Genetic Algorithms*. ed. by RAWLINS, G.J.E.B.T.F.o.G.A. Foundations of Genetic Algorithms, vol. 1, Elsevier (1991), pp. 69–93 http://www.sciencedirect.com/science/article/pii/B9780080506845500082
34. B.L. Miller, D.E. Goldberg, Genetic algorithms, tournament selection, and the effects of noise. Complex Syst. **9**(3), 193–212 (1995)
35. CompuPhase: Colour metric (2019), https://www.compuphase.com/cmetric.htm
36. D. Giacomelli, GeneticSharp (2013), https://github.com/giacomelli/GeneticSharp/
37. J. Vermeer, No Meisje met de parel (1665), https://www.mauritshuis.nl/en/explore/the-collection/artworks/girl-with-a-pearl-earring-670/
38. V. Van Gogh, Wheat Field With Cypresses (1889), https://www.vincentvangogh.org/wheat-field-with-cypresses.jsp
39. J. Wei, S. Wang, Z. Wu, C. Su, Q. Huang, Q. Tian, Label Decoupling Framework for Salient Object Detection, in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (6 2020), pp. 13025-13034
40. F. Uhde, S. Mostaghim, Towards a General Framework for Artistic Style Transfer, in *Computational Intelligence in Music, Sound, Art and Design*. ed. by A. Liapis, J.J. Romero Cardalda, A. Ekárt (Springer International Publishing, Cham, 2018), pp.177–193
41. H.Y. Lee, H.Y. Tseng, J.B. Huang, M. Singh, M.H. Yang, Diverse image-to-Image translation via disentangled representations. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 11205 LNCS, 36–52 (2018)
42. F. Uhde, S. Mostaghim, Dissecting Neural Networks Filter Responses for Artistic Style Transfer, in *International Conference on Computational Intelligence in Music, Sound, Art and Design (Part of EvoStar)* Springer (2021), pp. 297–312.
43. M. Arjovsky, S. Chintala, L. Bottou, in *Wasserstein generative adversarial networks*, ed. by D. Precup, Y.W. Teh. 34th International Conference on Machine Learning, ICML 2017 Proceedings of Machine Learning Research, vol. 1. PMLR, International Convention Centre, Sydney, Australia (2017), pp. 298–321http://proceedings.mlr.press/v70/arjovsky17a.html
44. S. Colton, M. Cook, A. Raad, Ludic considerations of tablet-based evo-art. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 6625 LNCS(PART 2), 223–233 (2011)
45. J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A.A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, R. Hadsell, Overcoming catastrophic forgetting in neural networks, in *Proceedings of the National Academy of Sciences of the United States of America* 114(13), 3521–3526 (3 2017), http://www.pnas.org/content/114/13/3521.abstract
46. Starline: Watercolor Splatters, https://www.freepik.com/free-vector/black-ink-watercolor-splatters-drips_10555025.htm