Andra Maria Pascale – a.m.pascale@student.vu.nl

Algorithms in computer generated art

Literature Study

Abstract

The purpose of this document is to explore the existing algorithms involved in computer generated art with a focus on visual art such as drawings, paintings and compositions. Section 1 provides an introduction into the field of computer generated art and section 2 continues by explaining the basic principles that stand behind it. More detailed and specific algorithms are presented in sections 3 and 4 such as evolving artistic filters, introducing disease as a creative element, using data flow networks to improve the computing speed and using a fluid dynamics model to simulate watercolors. In section 5 one can read about the major open problems that await further research and finally section 6 draws a conclusion on what has been achieved and what can be expected to be achieved in the future.

1. Introduction

Since the earliest times, visual art has been a powerful tool of expression and a fascinating subject. People have been using a variety of tools when creating visual art but the true essence of their creation was not influenced by these tools. No one gives credit to the paint brush or the easel although they do make a major difference in the quality of the final product. The artistic process happened solely in the head of the artist, his movements guided by his talent and creativity. Some people feel a deep spiritual connection with their paintings, truly believing that each painting is a distinct impression of their soul. Many of the greatest artists went as far as imagining that their work resembled the creation of God and that they had a special purpose of earth, being on a level far above the rest of the people. While such claims of self-importance do not improve the artistic process in any way, the true dedication of the artists can be felt in their paintings as they become more than paint on a canvas and spark deep emotions inside the viewers.

Naturally, when moving on to the newest tool at our disposal, the computer, many were skeptic and quick to dismiss its power. Even when used only as a tool, the computer seems to downgrade the beauty or importance of the creation in the view of many people. However, the natural trend of evolution pushed people to allocate time and resources to explore the extent to which a computer can

be used in the art making process. There are several directions one can approach when creating computer generated art but most of the solutions seem to converge to the concept of evolutionary art. When reaching this point, the computer is not considered a tool anymore, its newly found importance being the topic of much debate. It is not clear anymore what the concept of art stands for and if the computer can be considered to be creative in the same sense that humans are.

In the following sections a detailed background is provided to acquaint the reader to the basic principles behind evolutionary computation and how it is applied in art. The paper then continues to discuss several algorithms and the theoretical principles behind them while also considering the meaning of the process. Can this be called art and can we really expect a computer to understand what beauty means?

2. Background

Principles of evolutionary art

Terminology

Evolutionary computation is based on computational models inspired from the natural processes of evolution and selection. Its terminology was borrowed from biology and genetics since it simulates the natural processes in a computing environment. Thus, a candidate solution is called an *individual* while the entire set of possible solutions is called a *population*. In real life, individuals have their attributes stored in their genes and so in order to simulate this, each individual is represented by its *genome or genotype* which consists of a sequence of *genes*. The *phenotype* is the actual individual constructed from the genotype. The value of a gene is called an *allele*. In order to evolve, individuals have to *breed* and produce another individual called a *child*. In nature, the strongest individuals survive and breed thus passing on their genes but in this computational environment we have to specify how we measure the strength. Depending on the problem to be solved, we have a *fitness function* that we use in order to give each candidate solution a *grade* which will be used to decide if the candidate should breed or not. When the entire population is replaced by offspring, it is said that we have a new *generation*.

Evolutionary algorithms

The basic idea behind evolutionary algorithms is to initialize a population and establish a goal. Then one evaluates all the candidates by applying the fitness function and selects a group to be the parents. The next step is to apply genetic operators on the parents in order to produce offspring. Then the new individuals are evaluated and it is decided how they will be integrated within the population, either by replacing some of the parents or by replacing all the parents with the children. This process is repeated until the goals have been achieved and the process is stopped because it reached its maximum number or repetitions.

The initial population can be created randomly or it can be a set of rough solutions that need to be improved or adjusted. In order for the candidates to be evolved in the right direction, it is important to choose the parents that have a high fitness grade. There are several selection techniques:

- **Fitness proportional selection**: all the grades are normalized and based on this grade each individual is assigned a probability of being chosen, which means that in general the fittest will be chosen most frequently

- **Ranked selection**: assign each individual a fitness grade and then rank them according to this grade. Assign each a probability of being chosen based on this rank, which means that the individuals which are better than their peers will be chose more frequently

- **Tournament selection**: randomly select *n* individuals and choose the one with the highest fitness among them, then repeat the process until you have selected enough parents. Just like in real life, here it only matters if you are the fittest among a group of randomly selected peers.

- **Truncation selection**: choose only a proportion of the best individuals in the group

After you have selected your parents, it's time to apply the genetic operators. There are two types of operators:

- **Mutation**: use only one parent and apply a random variation to one or more genes to create a child
- **Recombination**: use two parents and combine their genes to create a child

Once all the children have been created and evaluated it's time to find a place for them in the population. There are two possible paths to take here:

- **Generational** evolutionary algorithm : replace the entire population with the offspring
- **Steady-state** evolutionary algorithm: replace only a subset of the parents with the children

Evolutionary Design

When applying evolutionary computation to solve engineering design problems there are 3 major concerns that need to be carefully considered in order to achieve the desirable result:

- **Representation** : selecting the appropriate representation can be difficult because it not only has to correctly describe the individuals, but also to take into account matters such as search efficiency and mapping between the phenotype and genotype
- **Genetic operators**: highly dependent on representation. For a binary string representation one could use the bit flip operator together with 1-point crossover which means that you copy the bits from the mother until you reach the crossover point and then you copy the remaining bits from the father. For a real valued representation, swapping/averaging the values of the genes taken from the parents is commonly used.
- **Evaluation function**: the algorithms use the feedback from the evaluation function to bias the search. Therefore, choosing a good fitness function is crucial when designing the evolution process.

Creative Design

In evolutionary design problems, one of the major concerns is the ability of the system to be creative. There has been some debate about what creativity actually means in this context, with some people considering that the introduction of novelty is not sufficient. In order for a system to be creative it has to produce results which are unexpected. There were also opinions that suggested the systems need to be able to evolve themselves, including the number of attributes of an individual, the fitness function and the mapping function. Altshuller introduced a number of paradigms that govern the measurement of creativity in a system:

- **Selection:** this is the first level where no novelty is introduced, design concepts are simply selected from a larger set

- **Modification**: searching for an optimal solution in a parameterized representation space of a class of engineering designs. Again, nothing new is introduced here.

  Innovation: this is the first step that described the introduction of novel information. Two or more populations can evolve simultaneously but separately and from time to time they exchange small amount of information. This type of evolution is called *the island model*

- **Invention**: in order to achieve this level the evolutionary algorithm must evolve the attributes themselves not only their value. This means that it must have the ability to add or delete some attributes.

- **Discovery**: this is considered to be the highest attainable level. In this case the mapping function has to be evolved. For example, if the mapping function reads some values and constructs a picture, it must be able to evolve the way it constructs the picture.

Another important concept closely related to creativity in design is emergence. An emergent design concept is a constructed attribute whose introduction simplifies the search space or makes the search process more efficient. It can be compared to the discovery of new important properties of the design space.

Representation

A representation of an engineering design is a computation description that can be used to reconstruct that design. There is a 1 to 1 mapping between the representation and the actual design. Usually each gene corresponds to an attribute and represents a dimension of the search space. The genes can be binary representing the presence or absence of a feature or real valued in which case they can be numerical or symbolic.

Depending on the goal of the algorithm the focus can be on:

- **Optimality**: the system should be direct (encode possible solutions), parameterized (allow only for small variations) and usually it should incorporate domain knowledge to make the search more efficient. Usually numerical attributes are used.
- **Creativity**: more complex representations are needed that range from direct to highly indirect (encode rules on how to build solutions) and usually allow for a wide range of variations to introduce novelty

There are five requirements for designing a good representation:

- **Non-redundancy**: mapping between solutions and encodings should be 1 to 1
- **Legality**: any combination or permutation of an encoding corresponds to a solution which implied that applying genetic operators will not result in illegal attribute sets
- **Completeness**: any possible solution has an encoding, ensuring that all the possible solutions have the ability of being found
- **Lamarckian property**: the meaning of the value of a gene is not context dependent which means that good traits can be inherited by children
- **Causality**: small variations in the genes result in small variations in the phenotype. Note that the binary representation violates this requirement.

Representations can be linear in case of a string representation for example and non-linear in case of multi-dimensional structures such as trees and arrays.

Another classification divides them into fixed length when the number of attributed remains fixed and variable length when the number of attributed changes.

Constraint Handling Methods

It is necessary to include constraints in any optimization or search mechanism exploring design spaces and thus we have special methods for evolutionary algorithms.

Penalty functions

The penalty functions measure the amount of constraint violation present in a solution. There are several types of such functions:

- **Static:** remain constant during the entire process
- **Dynamic**: change throughout the process
- **Annealing**: use a technique similar to simulated annealing
- **Adaptive**: change based on feedback received
- **Co-evolutionary**: penalty factors are evolved in another parallel population
- **Death**: rejects infeasible solutions

The biggest concern with penalty functions is achieving a balanced penalty value because using this value the search algorithm's space is constrained to a feasible space. If the penalty is too low the search will be carried out in an unfeasible space.

Other methods

Other solutions for this problem include simplifying the shape of the search space and introducing special genetic operators that preserve the feasibility of the solutions. Another possibility is to search near the boundary of the feasible region by going back and forth.

Repair algorithms have also been successful in combinatorial optimization problems where the cost of turning an infeasible solution into a feasible one is low.

Another class of methods is focused on separation of constraints and objectives and one example includes the competitive coevolution where the candidate solutions are evolved in one population but the constraints are contained in another population. A candidate solution has high fitness it is satisfies many constraints and a constraint has high fitness if there are many solutions violating it.

Finally we have the hybrid methods, the most interesting of which seem to be the fuzzy logic method and the ant colony one. The fuzzy logic method relaxes the constraints such as to allow more freedom in the search space. A constraint is violated from a scale from 0 to 1 which means that it can be only half violated having a violation factor of 0.5, etc. The ant colony method is inspired by real ants which have a unique method of finding the shortest path in a graph. They keep walking randomly until they find a source of food and then they return home but they leave a trail of pheromones which is used by other ants to retrace its steps. The pheromones evaporate in time which means that the longer trails will run out of pheromones faster. After a while it will be only the shortest path remaining since it was one the most frequently walked on and therefore the pheromone trails will never completely vanish being constantly supplied.

Multi-objective design

Multi-objective optimization evolutionary algorithms are designed to handle multiple conflicting goals. The two major goals of such algorithms are to find a high number or Pareto optimal solutions and to have widely differentiated solutions. A Pareto optimal solution is a permutation that ensures at least one individual is better-off and none of its peers are worse-off. The most popular multi-objective algorithms include:

- **Aggregated functions**: multiple objectives are combined into a single one using multiplication, addition or other operators. A frequent operator is the weighted sum where each objective is assigned a weight, although it has proven to be difficult to find appropriate values

- **Vector evaluated**: the survival selection mechanism of the genetic algorithm is modified in order to process multiple objectives

- **Multi objective genetic algorithm**: defines a rank of an individual depending on how many individuals in the population it is dominated by

- **Niche Pareto genetic algorithm**: it uses Pareto dominance to build a tournament selection process.

Co-evolutionary design

Biological coevolution is encountered frequently in nature and it has inspired a new class of evolutionary algorithms. The main idea is that populations evolve separately and simultaneously and there is no objective fitness function. Rather, individuals are evaluated based on how they interact with individuals from their peer populations. This method can be used for complex engineering problems that can be decomposed into smaller problems and solved separately. If the problem cannot be linearly decomposed then some success was achieved by using a cooperative co-evolutionary model.

Evolutionary visual art

This section is focused on presenting an example on how evolutionary computation can be applied to create visual art. The idea behind creating a 2D painting for example, is to have a mathematical function that takes the coordinates of a pixel and return a color. By applying this function for all the pixels on digital canvas, a new painting is created.

In order to evolve such an equation you can start with a number or functions such as sin, addition, log, exponential and then build a graph where these functions are the internal nodes and the leaves are variables or constants. This is an example of a multi- dimensional representation. In order to perform recombination one would take the tree from the mother and then replace a randomly selected subtree with a subtree from the father thus creating a new mathematical formula. In order to perform the mutation you could swap two nodes. When applying these operators it is important to remember that you must check that no illegal operations are born.

In order to evaluate these individuals, you usually need human input. There are voting servers where people can vote between several pictures and the most frequently chosen are the ones that will be chosen to participate in further recombination or mutation.

Another technique used real paintings as an input in order to evolve formulas that were then used to create other images. As mentioned before, a system with a high degree of creativity could not only evolve its attributes but also its mapping function which in this case would modify the way the colors are assigned to pixels.

3. Evolutionary algorithms

Autonomous art and artificial life

Image Synthesis

In the paper *Evolutionary Image Synthesis Using a Model of Aesthetics* by Ross, Ralph and Zong, the automatic synthesis of aesthetically pleasing images is investigated using genetic programming with multi objective fitness evaluation where the main feature test uses Ralph's model of aesthetics.

A mathematical model of aesthetics

Although it seems unlikely that such a model should exist, it has been discovered that many paintings that are considered to be works of art conform to a normal or bell curve distribution of color gradients. Thus the paintings that appeal most to people are the ones that do not have large areas with the same color or large areas where many different colors are adjacent. The majority of neighboring pixels have a balanced degree of different shades. Having discovered this fact, the authors are now using it to compute the image's curve bell gradient in order to deduct if the picture is appealing in a painterly fashion or not.

To compute this value they first compute the stimulus for each pixel using the formula:

$$S_{i,j} = \sqrt{\left|\nabla r_{i,j}\right|^2 + \left|\nabla g_{i,j}\right|^2 + \left|\nabla b_{i,j}\right|^2}$$

meaning that $S_{i,j}$ computes the aggregated stimuli produced by all three channels, red, blue and green

where $$\left|\nabla r_{i,j}\right|^2 = \frac{(r_{i,j} - r_{i+1,j+1})^2 + (r_{i+1,j} - r_{i,j+1})^2}{d^2}$$

and the same goes for green and blue. This basically means that for each pixel they compute the amount of difference in shade compared to its neighboring pixels for each color channel.

Since the human being's response to visual stimuli is logarithmic, the authors use a logarithmic function to compute the response based on the stimuli.

$$R_{i,j} = \log {S_{i,j}}/{S_0}$$

The next step is to compute the distribution of the responses for an image based on the principle that the probability that a person will pay attention to a region of the painting is proportional to the magnitude of the stimulus residing there. The normal distribution of R is then calculated using a normal weighted distribution defined by mean μ and a standard deviation σ^2 .

$$\mu = \frac{\sum_{i,j}(R_{i,j})^2}{\sum_{i,j} R_{i,j}} \qquad \sigma^2 = \frac{\sum_{i,j} R_{i,j}(R_{i,j} - \mu)^2}{\sum_{i,j} R_{i,j}}$$

The authors then proceed to create a histogram with bin width $\sigma/100$ where each $R_{i,j}$ fills its corresponding bin using a weight of $R_{i,j}$.

The final step involved computing the deviation from normality score or DFN. This score measures how far away a picture is from fulfilling the desired bell distribution.

$$DFN = 1000 \sum p_i \log(\frac{p_i}{q_i}) ,$$

Where p_i is the observed probability in the ith bin and q_i is the expected probability for a perfect normal distribution. If the DFN is 0 then the image has a perfect normal distribution.

Since it provides a measure of how fit a picture is, the DFN is used as a fitness grade in the evolutionary algorithm to determine the successful individuals.

Feature Evaluation

Each produced image will be evaluated by using two feature tests. The first test is the DFN described above that will determine how close the image is to having a gradient distribution similar to those found in famous works of art since this is a good indicator of its perceived beauty.

The test called CHISTQ compares the color distribution in the image with the color distribution in a target image by creating histograms for both images. Then a distance can be calculated between the two images and thus obtain a second score of how fit our produced image is.

Since we have two feature tests, this is a case of multi-objective algorithm which considers each test to be a separate dimension of the search space. All the individuals are ranked based on these scores using the concept of Pareto dominance. An individual dominates another one if both its scores at least as good and at least one is higher. All individuals with rank 1 are not dominated and therefore are considered to be valid solutions.

To compare individuals that have the same rank, they are again evaluated based on their diversity. Textures that are more diverse compared to their location in the search space are considered superior since in the end we aim at breeding diverse solutions which have the best ranks in order to obtain even better offspring. The diversity score is computed by computing the nearest neighbor distance in both feature tests' spaces and then averaging the two results. The individuals with the highest diversity are assigned the lowest ranks.

Results

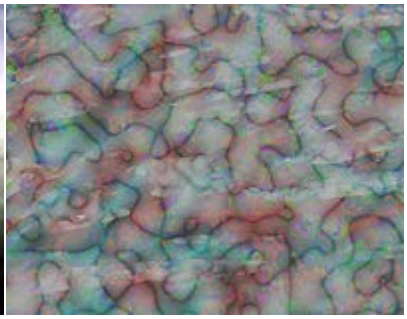




Figure 1 **Figure 2** **Figure 3 – Color target**

Figures 1 and 2 show two results obtained by using figure 3 as a color target. The authors also mention that some images were considered to be fascinating although their DFN score was very high but when removing the DFN requirement, the images were bland, chaotic and rarely appealing. Therefore the DFN provides a good heuristic to direct the search toward the feasible space but some freedom must be allowed since it is not a strict indicator and beauty in paintings can sometimes be achieved even when violating this condition.

Artificial Life, Death and Epidemics

This paper explores different strategies for slowing the onset of convergence in an evolving population of agents, such as maintaining separate agent sub-populations and migrating between them, and introducing virtual diseases that co-evolve parasitically within their hosts.

Motivation

One of the main problems identified in the exploration of evolutionary art was the early convergence of the population implying an impoverished genetic pool. While this situation might be desirable in some cases where you are looking for an optimal solution, it should be avoided when creating art since you do not know in advance what you are looking for. When your goal is to explore the search space, you aim at having a diversified genetic pool so that unexpected results can be produced. Otherwise your results will eventually all look similar and become less interesting from an artistic perspective.

The truth is that this convergence also occurs in real life in some geographic regions but the best individuals are usually born from the combination of two very different parents. The authors were inspired from the Artificial Life literature such as *PolyWorld* by Yaeger where the individuals fight and kill each other and the computer simulation project *Tierra* by Thomas Ray where the individuals can the infected by parasites which evolve as simple organism lacking a portion of their genetic code. In order to reproduce there parasites are look for organisms that have the sequence of code that they miss an then infect them, just like in real life. When the individuals who are naturally immune multiply and attempt to take over, the parasites also evolve to become a threat for the previously immune organisms.

Epidemiology theory

Although it might seem unfit to include an unpleasant concept such as disease in art, there have been many instances in the past when artists found inspiration in such places since it inspires strong emotions in people, being they even of fear and disgust. However, it is advised that we exclude the emotional connotations of the concept of disease at first, since the focus here is how we can use it in an artificial life model in order to create something beautiful and inspirational, not to traumatize people. Considering that no beings suffer in our process and we are simply manipulating pixels and performing computations, there should be no negative connotations attached to this process.

In the standard epidemiological model, the population is divided into three groups:

- **S** – susceptible
- **I** - infected and capable of transmitting it to others
- **R** – removed, meaning that they are immune or infected but isolated and therefore unable to transmit it to other

There are specific stages of infection:

- **Latent** period : victim is infected but cannot infect others and shows no symptoms
- **Incubation** period: victim can infect others but shows no symptoms
- **Post-incubation** period : victim shows symptoms and can be avoided by susceptible individuals

Probabilistic models operate in discrete time steps such that in every new step:

$$P \propto S * I ,$$

Where P is the probability of a new case of disease appearing, S is the number of susceptible individuals and I is the number of infectives. This is a rather simple model that presumes the homogenous mixing of the population. Other complex models have been proposed that account for the geographical isolation of people by dispersing them along a line, across a lattice or over a network. Some simulations also include the concept of carriers of viruses, such as mosquitos.

There are two theorems from epidemiology that are particularly relevant for the presented project:

The threshold theorem: the density of the population has to be above a certain threshold in order for an epidemic to take place. If the disease spreads, then the disease will reduce the population to a number P which fits in the following equation, considering that T is the threshold and I is the initial number of people:

$$P = T - (I - T) ,$$

meaning that the disease will basically correct the error in the population number. If the population had 50% more people than the number N of individuals it should have had, the disease will not only reduce the number of remaining people to N but further reduce it to 50% less than N.

Herd Immunity: this theorem follows from the theorem above and says that in order to prevent an epidemic, you have to immunize a certain number of people and not the entire population. This number can be calculated but it was shown that it is not always accurate since it assumes that the population is homogenously mixed.

Description of the world

The simulation contains a number of agents which are represented by boxes of different sizes and colors. They move in discrete time steps over a continuous space and are able to detect the presence of other agents that are in their visual range, as well as their attributes such as color and size.

There are two functions called like and dislike which evaluate the other boxes an agent meets and depending on its preferences decide how agreeable or disagreeable it finds them. The usual speed at which a box travels is inversely proportional to their volume but if they find a box which is extremely agreeable or very disagreeable they might speed up and change their direction.

When moving through space two agents that meet might like each other. In this case they mate and have one child which inherits genes from both parents. The genes of the child encode the size and color as floating point values, as well as a color they will like and look for and one they will dislike and run away from. The mating process uses a single crossover point meaning that the child copied genes from the mother up to a point when it switches to the father and copies the rest of the genes from him. It also uses mutation for each child by a small random amount in the region of +/-5%.

However, the space is limited and a new birth is approved only if there is enough space in the population. If there is no space left, the birth is refused and a random individual may be removed from the population to make room for new ones. This method of removing an individual randomly just to make space for a new born seems cruel and not found in real life.

Another interesting aspect of the world is that each box has an amount of energy it can use to move and reproduce. Every new time step a box received an amount of energy proportional to its volume but if it runs out of energy it is removed from the people, i.e. it dies. When two boxes reproduce, an amount of energy is subtracted from each of them and given to the newborn.

Each box has a lifespan and when it reaches the end of this lifespan it is removed from the population. Every newborn cannot reproduce until it reaches a certain age which is the same for all the boxes.

Diseases

Diseases are introduced in the environment and can exist as long as there is at least one infected host. Each disease has a color signature meaning that a box will be infected with a probability proportional to the amount of similarity between its color and the color of the disease. An agent can be infected only if it is susceptible and meets an infected agent. If a box is infected with a virus it cannot be infected with a second virus.

Each disease causes a certain amount of devastation every time step. The amount of devastation in terms of subtracted energy increases proportionally with the amount of similarity between the color of the host and the color of the disease. The more devastation an agent incurs, the more likely it is that will spread the disease to a susceptible. If the agent manages to keep itself alive until the disease reaches the end of its lifespan, it acquires immunity to that disease and adds it on its immunity list. If the host dies, the disease dies with it irrespective of its lifespan. Each virus also has a period of latency and incubation to resemble the real life model as closely as possible.

Since in real life, viruses mutate faster than the host's organism can, a similar aspect has been introduced into this simulation. Every time step the disease reproduces asexually and can randomly mutate any attribute such as the amount of devastation, lifespan or incubation period as well as the parameter indicating how frequently it mutates. This addition allows for the disease to co-evolve with the population.

The inception of a disease is introduced once in every 100 000 agent updates and its colors is chosen depending on the color of the unlucky box, all its other attributes being random. The newly infected is then set free in the world and it may or it may not cause an epidemic.

Results

The purpose of this simulation was to prevent the convergence to the gene pool to an impoverished, boring set. The introduction of the disease element successfully achieved this goal and very effective against individuals clustering into tight groups of similar size and color. It also effective against the complete elimination of certain colors or size ranges.

After the concept of disease was introduced the population was more uniformly dispersed across the entire search space instead of forming a few groups in a much restricted search space.

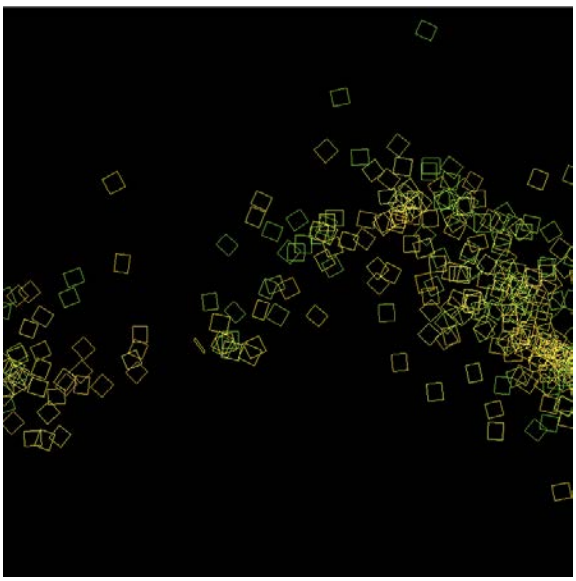


Figure 4

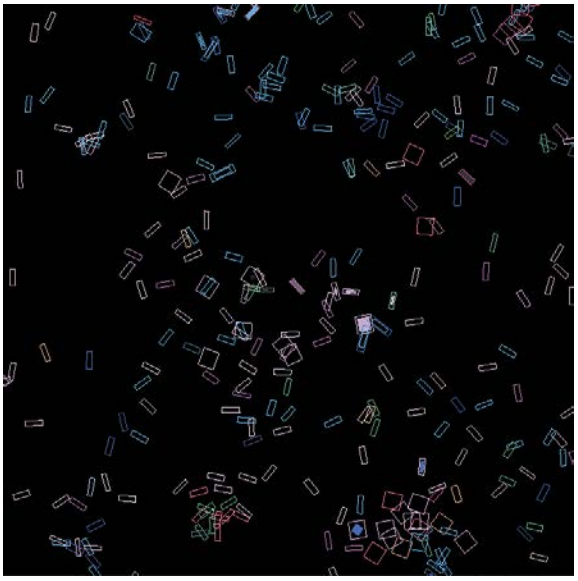


Figure 5

In figure 4 one can see the population without the disease factor. It is to be noted how it formed groups of preferred colors and sizes and left out a large portion of the search space, in the bottom and top. Figure 5 shows the population with the introduced element of disease. It is clear that in this case the gene pool is much more diverse and the population is more uniformly distributed across the search space. Both simulations were stopped after 14 000 steps.

Several different scenarios were observed when studying the spread of the newly introduced diseases:

Immediate disease elimination: if the disease is unsuccessful by itself because it's too short lived it will reach the end of its lifespan before it can infect others. The same immediate elimination of the disease happens if the population is not sufficiently dense or if there are no neighboring boxes with the same color.

Immediate disease spread: if the infected host co-habits with many individuals of the same color then the disease can easily spread quickly. However, the most common case was that the disease quickly mutated to be able to infect host of different colors and then spread quickly. These agents of different colors were only slightly affected by the disease but were able to spread it around fast.

Eventual disease elimination: sometimes a disease dies out after a period of time because it runs out of susceptible to infect. This may happen if the disease was not very devastating and produced many immunes or if it was extremely devastating, wiping out the entire susceptible population before it has the change to produce offspring.

Continual disease spread: when a disease was very successful in that it was devastating enough to be spread quickly but not devastating enough to kill off all its susceptible individuals, it persisted in the

population for a long period of time. Another trait of such a self-sustaining disease was that it was stable enough to not produce mutations that would make it unsuccessful but still mutated enough to ensure it can still infect different hosts. One the most successful disease was one that sought prolific breeders and was passed on when two agents reproduced. The disease was also passed on to the newly born and remained dormant enough time to ensure it was spread before the agent was weakened. This class of successful self-sustaining diseases is surprisingly similar to real life sexually transmitted diseases but this observation is equally disturbing and beyond the purpose of this paper.

The algorithm was applied to a simulation of a 3D world where humans could interact with the boxes and produce children if the boxed found them attractive. The purpose was to produce a visually appealing population of boxes but of course this can be easily extended to produce all kinds of interesting shapes. Being still in an exploratory faze, the art produced might not be impressive at first but the diversity of the produced individuals is inspiring enough to inspire future development since the way has been successfully paved.

Artistic Filters

This is an exploration into automatically evolving non-photorealistic filters that tend to produce images with painterly, balanced and harmonious characteristics.

Motivation & Idea

The main motivation for this algorithm was to produce harmonious, painterly images that can be recognized as works of art without having a human input at every step of the way.

In order to automatically evaluate the images produced, Ralph's model of aesthetics is used in a similar manner to the method described in the previous paper. The main difference here is that the authors do not produce images, but filters that they apply to an already existing painting or photograph to turn it into something new and unique. They have used this method because producing images from scratch created very artificial and mathematical looking images that cannot be considered artistic.

The basic difference between an image filter and an algorithm that generates images is that the image filter takes as an input the color of a pixel and returns another color whereas the image generator uses a function to take in the coordinates of the pixel and produce a color.

When a filters is added to a population, multiple feature tests are used to derive a score which will be used to impose a Pareto ranking using the principle of domination in the same way as described in the Image Synthesis section of this document. These multiple feature tests imply multiple scores which could be merged together into one objective by introducing ad hoc weights, but the authors prefer to use a multi-objective approach.

The fitness function and experiment details

Similar to the paper described in the Image synthesis section the authors use the DFN – deviation from normality score as a fitness function since it evaluates the image's bell curve against a desired bell curve that belongs to aesthetically pleasing, painterly images.

A known painting is used as the source image for a run, such as Mona Lisa. The same painting can be used as a color palette target or another one can be used for this purpose. The canvas is initialized with the source image and then is dynamically altered by the filters. A filter is an expression tree comprising of a series of language components selected from a library of functions and terminals. Some pre-filtered images are also created by pre-processing the source image.

For the purpose of this project the genetic programming based system Gentropy is used, which is designed especially to evolve 2D textures by combining common mathematical and image manipulation functions into formulas. This process is usually very difficult for a human being to master considering the complexity involved.

The filter is applied to the entire canvas from left to right, top to bottom and for each pixel a new RGB component is calculated using its initial RGB, luminosity and Sobel filter which basically calculates how likely it is for that particular point in the image to be an edge or moment of abrupt change.

Image preprocessing

A number of filtered images are produced in the pre-processing stage to be used by the GP filter. A luminosity filter is created by applying the transformation:

$$L = 0.299r + 0.587g + 0.114b$$

and also a sobel filter is also generated which recognizes the edges in the source image. Finally a non-photorealistic paint primitive is computed. First we need to find the points where most likely a brush was applied by applying a threshold of 0.5 to the Sobel image. The result is a binary image where the 1's represent the points. For each point, a color difference image is computed by taking a N by N area around the point and computing the relative color distance between the center point and all the other pixels in the area. The results obtained are in the range of 0.0 – 1.0 where 0 means that the difference is large and 1.0 means it the same color.

Next, moment calculations are performed which determine the length, width, angle and center coordinate offsets of a brush stroke to be placed at that point. If these computations don't correspond to a valid pixel location, they will be turned into 0.

The filter language uses two data types, float and vector. The float operators are simple operations such as returning the red, green or blue channels of the source of canvas as a specified pixel, average two values or compute the moment and return the x or y coordinate of the center of brush stroke. The if operator takes in three arguments and returns the second one if the first one is larger than or equal to 0.5 or the third one if the first argument in smaller than 0.5.

The vector operators are more interesting since they are not all built into the system and predefined. Apart from the simple operators such as copy and blend which averages the colors of two pixels, there are two operators, paint and paint_general which apply expressions to the current pixel in order to produce a color. The arguments supplied to the paint operator are not predefined but evolved internally as expression subtrees. Whatever values these expression output are fed into the paint operator to make the filters evolve and not always produce the same results. The general_paint is similar to the paint operator but has one added argument that indicates what color should be used , whereas the paint operator always used the source image's color.

Fitness evaluation

Two feature tests are performed once the filters were applied to the image. The first test is the CHISTQ test which just as described before computes the similarity between the image's color histogram and the color histogram of a chosen target image.

The second feature test is the popular bell curve score, the DFN. Since the most desirable result would have a DFN of 0, they attempt to minimize this score as far as possible. Having two objectives, there may be images who obtain one very good score and one very bad but using the principles of multi-objective ranks, in the end they obtain images that perform well in both tests.

Results and discussion

Figure 6 - DFN score = 4.3

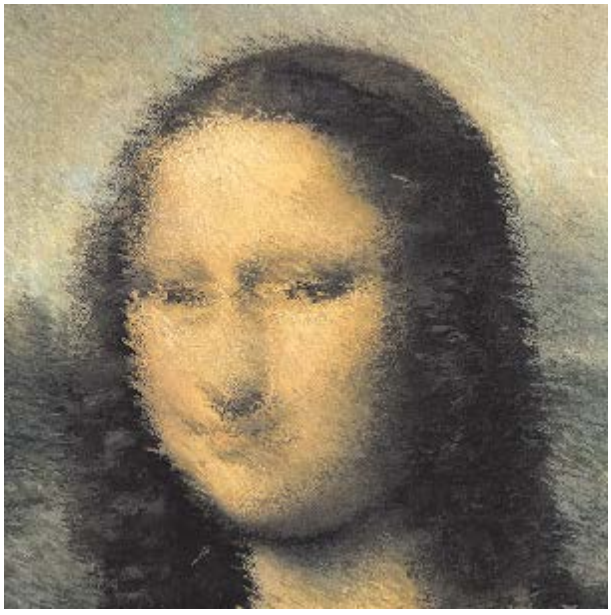


Figure 7 – DFN score = 28.8

Here you can see two filters applied to a section of the Mona Lisa painting. FIGURE 6 has obtained a DFN score of 4.3 which is very low and indicates a high degree of beauty whereas FIGURE 7 has obtained a much higher DFN score of 28.8 which indicates it is not so attractive.

Another attempt was made by allowing the filters to completely change the colors by using a predefined color palette.

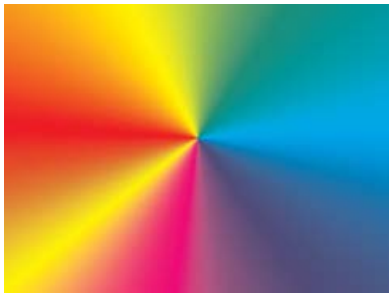


Figure 8 – Color palette

Figure 9 – DFN = 119.3

Figure 10 - DFN = 77.8

In FIGURE 9 we can see a noisy and unpleasant result which is also indicated by its high DFN score. The bright and varied color palette also had a detrimental effect on the DFN score which prefers a more conservative range of colors. In FIGURE 10 we can see an interesting result that has true potential. It was achieved by combining the color palette with the initial colors of the source image. This might be a good compromise to introduce novelty in the creation without producing artificial looking images that resemble something you see on TV when the transmission is really bad.

Figure 11

Figure 12

The authors mention that most of their results were not impressive or were interesting but the image portrayed awakened intense negative feelings in the viewer and therefore were considered unsuccessful. It is to be expected that you have to weed out a lot of results before you find something worth mentioning. FIGURE 11 and FIGURE 12 show two results that in my opinion are quite fascinating and original. Although you can distinguish the shape of a face, it does not immediately lead you to think of the Mona Lisa painting and they can be considered beautiful in themselves.

A relationship has been discovered by the authors between using the painting primitive and obtaining good DFN scores. When the painting primitive was used, 79% of the images had a good DFN score, whereas only 53% qualified when no painting primitive was used. Although the relationship is not very strong it comes to reinforce the idea that the painting primitive does indeed produce images that are perceived as having been made by a real painter.

Data Flow Networks

The real-time nature of the data flow networks yields an additional avenue of design space exploration. Assuming the individuals can be generated quickly enough from a corresponding set of genes, not only is it possible to explore the parameter space vertically by examining multiple generations, it also becomes possible to explore the space of possibilities horizontally by adjusting the evolution control sliders.

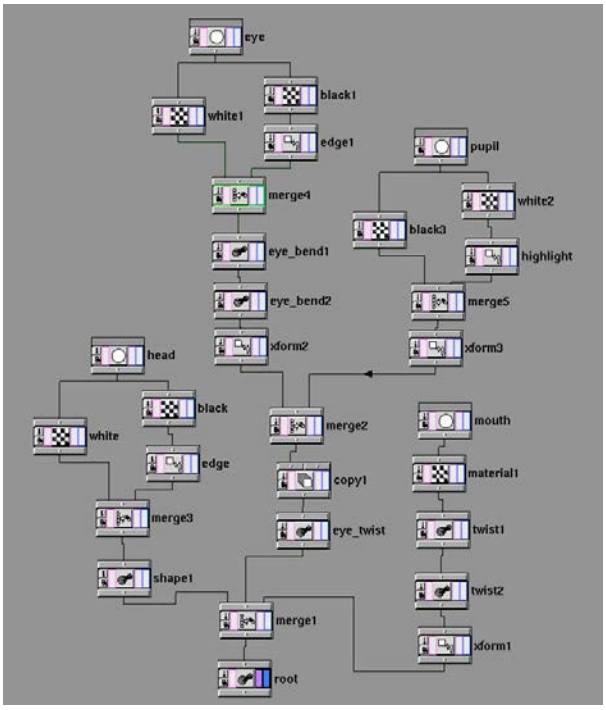
Motivation & Idea

This aesthetic evolutionary design system was designed to allow non-expert people to discover and explore interesting design solutions in a simple and interactive manner. The user has to be familiar with Houdini which is a 3D modeling and animation package. Houdini allows you to create a simple model using a data flow network and this feature is exploited by the evolutionary algorithm. Once the user has created a model he can drop it into the evolution interface in order to improve it. To illustrate how the system works, the author create a simple face prototype made out of circles.

The idea behind this system was to use the dataflow network created by Houdini to evolve the model faster and thus allow for a more extensive search and exploration.

Prototype representation

The geometric model utilized here was composed of a stack of black and white disks to construct something similar to a face. The representation is in the form of a directed acyclic graph where the leaf nodes are geometric primitives and in this case, circles. The purpose of the other nodes is to modify the geometric primitives in some way as the information travels from the root to the leaf nodes. Each individual has a genome composed of 17 genes each one being responsible for one property in the individual's network.



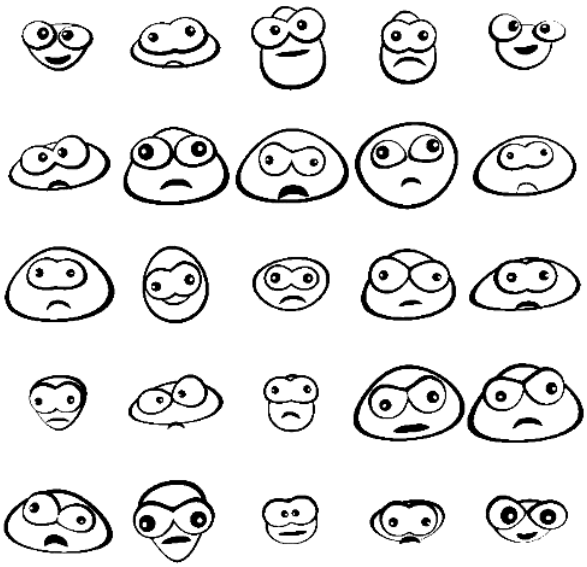


Figure 13 – Data flow network for simple faces

Figure 14 – A population of faces

In FIGURE 13 one can see how the graph looks like for the simple example of faces and in FIGURE 14 one can see an example population of faces that can be constructed from the prototype. There are different types of operators in the graph that can be seen above, such as primitive operators that produce the circles, material operators that color them black or white, a copy operator used to duplicate the eye, transform operators that scale, rotate or translate and twist operators that bend the geometry.

A rather complex prototype has been created representing a 3D human shape. It is constructed from a hierarchy of 35 ellipsoids and consists of 96 genes. This prototype can be seen in FIGURE 15.

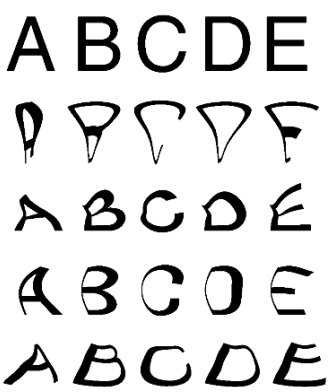


Figure 15 – Human prototype

Figure 16 – Square letter prototype

Figure 17 – Letter prototype

FIGURE 16 represents another simple prototype with only 9 genes with a letter inside a circle inside a square to prove how diversified the results can be even for such a simple model. In FIGURE 17 a different kind of model is represented where the letters are deformed by magnet operators placed around the letter. This is however a work in progress.

Evolution

The interface of the system produces a random initial population and then asks the user to select its favorite individuals. Once he is done deciding the user can press the *Make new generation* button to see the offspring of his selected parents and he can then choose again. Each individual is represented by a channel that stores arrays of values consisting of its genes. The evolution network consists of channel operators called CHOPS. To initialize the population a noise CHOP is used to create individuals.

When the user hits the *Make new generation* all the genes are copied in a file which is loaded by the load genes CHOP. First, the genes belonging to the individuals that were not selected are deleted so that only the prospective parents are left. In order for offspring to be produced a mate chop operator is used which randomly selects pairs of parents and performs several crossovers when copying their genes to their child. After the entire offspring population is produced, the mutation rate indicated by the user is enforced by randomly mutating the genes of the children.

The data flow network representation allows the system to generate the individuals quickly enough to allow the user to explore the space horizontally. You can modify the various parameters such as mutation rate and crossover amount and see the modified population immediately. Therefore you can make the necessary adjustments before deciding on the final values of the parameters and moving on to the next run. Being able to adjust these parameters every step of the way, broadens the searched space and offers more opportunities for finding interesting phenotypes.

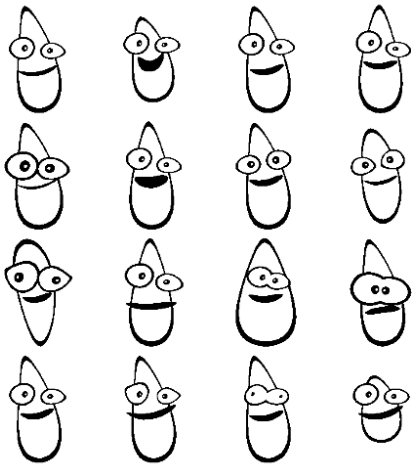


Figure 18 - Low mutation

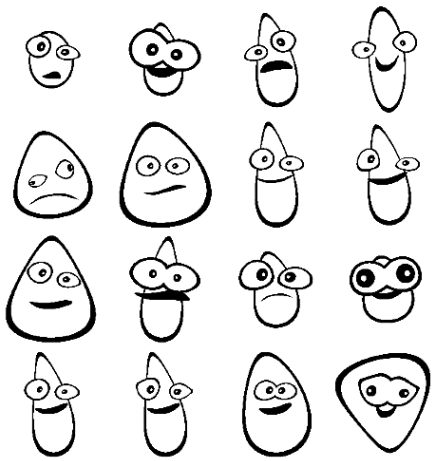


Figure 19 – High mutation

In FIGURE 18 one can see a population produced with a low mutation rate and in FIGURE 19 one can see a population produced from the same parents with a high mutation rate. It is obvious that the differences between the two populations are rather major and that the choice of the mutation rate parameter can make all the difference between a good result and no result. This comes to reinforce the idea that being able to control the mutation rate dynamically after having a preview of the results is very important factor for the success of the system.

Future work

The next step for this evolutionary system would be to create a prototype for evolving painterly images. The authors are already working on this but the real challenge here is to port it into Houdini in order to be able to take advantage of its modeling and design algorithms. Nevertheless, the system offers an original solution that allows the user to explore the search space faster and guide the system's evolutionary mechanism by adjusting the parameters, even though in doing so it actually takes a step backward from being autonomous.

4. Non-photo realistic rendering

Using image technology

This section is meant to provide an overview of two methods used to simulate artistic tools in computer generated images.

Graphite Pen

Here the algorithms behind simulating the graphite pen will be presented as well as some applications.

Computer Generated Watercolor

The unique features of the watercolor inspired the authors to recreate the same effect using a computer for several different purposes such as including it in a painting program, automatic *watercolorization* of images and as a mechanism for non-photo realistic rendering of 3D environments.

Water color properties

In order to create a decent simulation of the watercolor one has to understand not its physical properties but also to be aware of all the effects that can be achieved with this technique in order to build a useful tool for artists and amateurs alike.

Watercolor images are created by applying a watercolor paint to a piece of paper. Normally one can used any kind of paper for this purpose, but in order to obtain the best results, you need a special kind of paper. The watercolor paint is made out of pigment particles suspended in a solution of water, binder and surfactant which ensures that the paint will adhere to the paper and be absorbed by it.

Figure 20 – Watercolor paper

Figure 21 – Pigment

Watercolor paper

The special paper used for watercolor painting has a rough texture as it can be seen in FIGURE 20. It is not made out of wood but most usually from linen or cotton rags pounded into small fibers. These small fibers form a web with many pores that need to be filled with sizing made out of cellulose in order to bring its absorbancy rate to a desired value.

Pigment

The pigment as it can be seen in FIGURE 21 is a solid material in the form of many small particles that are usually ground into a fine powder. The pigment penetrates the paper and then settles soon afterwards, but lighter pigments can stay suspended in water for a longer time and thus travel further away on the paper before coming to a halt. The staining power of a pigment is a measure of its tendency to adhere to paper fibers. An important characteristic of pigments is granulation which happens when particles settle in the hollows of the rough paper.

Binder and surfactant

The binder ensures that the paint adheres to the paper and the surfactant allows the water to soak into the paper. Although they are both very important for the success of the painting process, they are not included in the model since it's the responsibility of the paint manufacturer to find the optimum balance between them.

Watercolor effects

There are two main techniques of applying the watercolor, namely the wet-on-wet method in which the paper is first soaked with water before applying the paint and wet-on-dry where the paper is kept dry before applying the paint. Depending on these two methods and on the movements of the artist there are several effects that need to be taken into account when simulating the watercolor.

a – drybrush b-edge darkening c- backruns d - granulation e – flow effects f – glazing

Figure 22 – Watercolor effects

Drybrush: a dry brush applied at the proper angle will leave a color trail on the elevated areas of the paper thus revealing an irregular pattern with gaps

Edge darkening: this is an effect that many of us have noticed when taking painting classes in primary school. When applying a wet brush on dry paper, the surface tension of the water tends to keep it together and allowing the pigment to flow towards the edges of the water bubble thus leaving a more intense impression on the edge of the trail. When this was not intended it can ruin your painting but when you are counting on it to achieve a special effect, it is very important for it to happen.

Backruns: This is another annoying feature of watercolor painting. When the paper tends to dry unevenly and some water puddles spread over dried regions, washing away the color and forming irregular patterns looking like branches. However, it is important to include even this feature because some people actually try to achieve it on purpose.

Granulation: as it can be seen in the FIGURE 22 d, the pigment tends to settle in such a way as to reveal and emphasize the peaks and valleys of the paper. This happens when the paper is very wet.

Flow effects: when using the wet-on-wet method, the surplus of water allows the paint to flow freely and form beautiful shape with irregular feather like edges.

Glazing: this is a very important feature of watercolors that can be easily misunderstood. It consists of applying several thin layers on top of each other usually using different colors in order to optically mix them. This means that they are not physically mixed but simply super imposed on the paper. The final color is created as the light is sequentially absorbed as it goes through all the layers. It is said to produce an especially beautiful effect, creating the illusion of illuminating the painting from within.

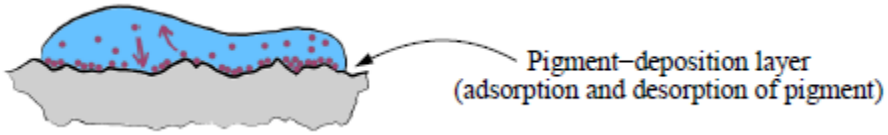
Computer model

A complete painting is represented as an ordered set of washes over a rough paper surface. Each wash may contain a different set of pigments spread over a region of the paper and it is stored in a data type called glaze. Each glaze is constructed by running a fluid simulation to create the impression that the color would leave when flowing across the paper.

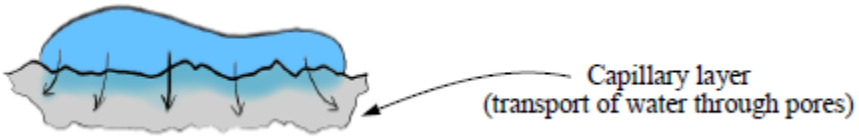
Each wash is composed from three layers:

Shallow water layer: water and pigment flow above the surface of the paper. The area of flow is bounded by the wet area mask M which is 1 if the paper is wet and 0 otherwise. The water flows with velocity u,v in the x, y direction and exerts a pressure p. The concentration of pigment in the water is denoted by g^k. Other important quantities here are the slope Δh of the rough paper and viscosity μ and viscous drag κ of the watercolor medium.

Pigment deposition layer: the pigment travels between this layer and the shallow water layer by absorption and desorption. These phenomena are affected by the physical properties of the pigments such as density, granularity and staining power. The deposited pigments are denoted by d^k.



Capillary layer: water that is absorbed by the paper is diffused into a capillary shape. This layer allows for the expansion of the wet area of the paper. The relevant quantities here are water saturation and fluid capacity of the paper.



The paper is modeled simply as a height field and the fluid capacity of the paper is directly proportional to the height. If a point on the paper has 0 height then it has the minimum fluid capacity and if it has the maximum height of 1 then it has the maximum fluid capacity.

The fluid simulation

There is a main procedure that loops over a predefined number of time steps and it contains four functions. As in input it takes the initial wet area mask, initial velocity u,v of the water and its pressure p, the water saturation s of the paper and the pigment coefficients.

MoveWater (M, u,v, p) : this function takes as an input the wet area mask M to have a boundary for moving the water, u and v which represent the velocity of the water and p which is the water pressure. There are some basic rules that are modeled here to ensure that the flow adhere to the laws of nature.

First the flow must be constrained so that the water remains within the wet area mask and a surplus of water in one area should case flow outward from that area into nearby regions. These conditions are imposed by implementing the shallow water equations from physics. For further details please refer to the appendix. Then the flow must be damped to minimize oscillating waves and it must be affected by the texture. These two features are taken care of by including the viscous drag and the paper slope into the simulation.

Finally there should be an outward flow of fluid towards the edges in order to produce the edge darkening effect and any local change should have a global effect. In order to achieve the edge darkening effect, there is a subroutine which at each time step removes an amount of water from each cell, considering that the paper is divided in a grid of cells, depending on the distance from the edge of the water boundary. Therefore if you have a water color puddle, at each time step a larger quantity of water is removed from the middle of the puddle and a very small quantity is removed from the near edge surface to simulate the outward flow of the water. In order to account for the second effect, there is another subroutine which relaxes the divergence of the velocity field until it reaches some threshold by redistributing the fluid into the neighboring cells.

MovePigments (M, u,v, g^1,…g^n): this functions takes care of moving the pigments. It distributes pigment from each cell to its neighboring cells depending on the rate of fluid movement out of the cell.

TransferPigment(g^1,…g^n, d^1,…d^n) : at each step of the simulation, pigment is absorbed by the pigment deposition layer at a rate and washed away back into the fluid at another rate. This is controlled by the density and staining power of each pigment. The granulation also affects the rate with which a pigment is absorbed or desorbed.

SimulateCapillaryFlow(s,M): this method was included to account for the backrun effect as described above. This subroutine simulates the water diffusion into the capillary layer. At each time step each cell transfers water to its neighbors until they are fully saturated. If a cell reaches a certain level of saturation, the wet area mask is enlarge to include it since it is now considered to be wet.

Finally, the drybrush effect is achieved by excluding from the wet area mask any points that are under a height threshold.

Rendering the pigmented layers

The authors use the Kubelka-Munk model to simulate the optical compositing of the glazing layers. Each pigment is assigned an absorption and a scattering coefficient. These coefficients are a function of wavelength and control how much energy is absorbed and scattered back per unit distance in the pigment layer. Each layer has a thickness which is a sum of the thickness of all the pigments that can be found in that layer. All these coefficients are applied in the KM equations in order to computer the overall color of a pixel which is then rendered. Please refer to the appendix for more details about the KM equations.

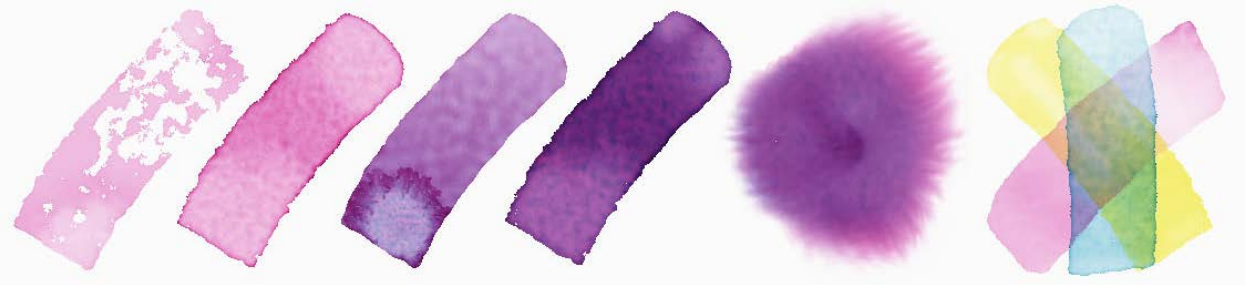


Figure 23 – simulated watercolor effects

In FIGURE 23 one can see how the various watercolors effects look like when they are simulated using the methods described above. Except for the granulation effect which looks a bit artificial, the rest are impressively accurate and to an untrained eye are indistinguishable from the real effects.

Applications

The authors developed a painting simulator application which can simulate a watercolor painting being given a set of glazes with their colors and a wet mask by the user.

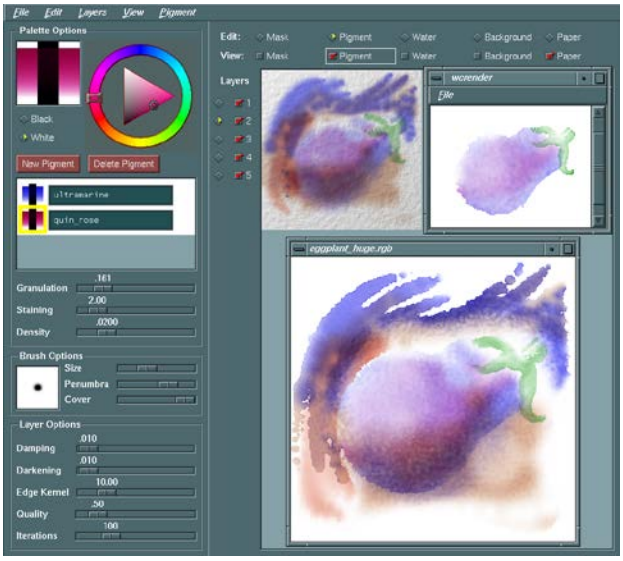


Figure 24 – Watercolor paint simulator

As one can see in FIGURE 24 there is a very large number of parameters that are left to the discretion of the user and in the right side one can see a painting after a glaze layer was applied to it. The glaze layer can be seen in the upper side of the image.

In my opinion, the most successful application of this technique was in automatic image watercolorization. They first employ a method called color separation which tries to find the perfect combination of pigments that would produce a color similar to the color the method reads from the actual image. Then they apply the second method called brushstroke planning.

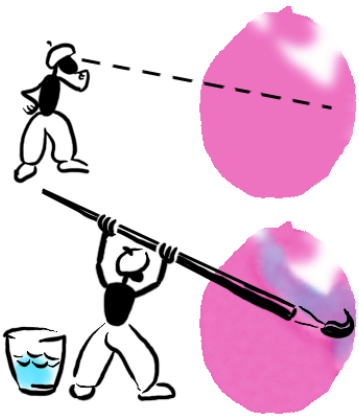


Figure 25 – Brushstroke planning (a)

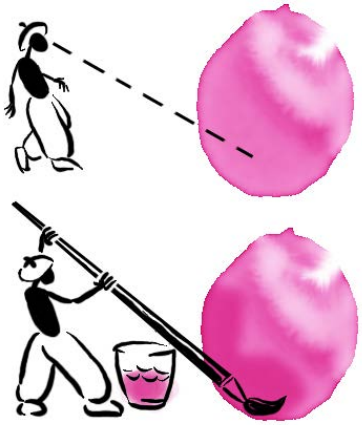


Figure 25 - Brushstroke planning (b)

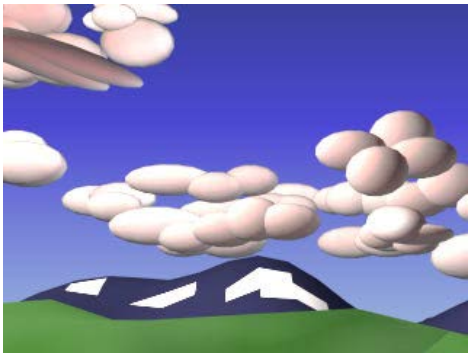
In FIGURE 25 (a) the painter notices a region with too much pigment and decides to dilute it by adding water. In (b) the painter notices a region where there is not enough pigment and add some more. These two situations are also simulated for the image in order to perform the image watercolorization.

Figure 26 – Initial image

Figure 27 – Watercolor image

In FIGURE 26 you can see the initial photograph of two apples and in FIGURE 27 you can see the result of the watercolorization process, now it looks like a watercolor painting of the two apples.

Finally there was an application of this process to the simulation of a 3D environment.



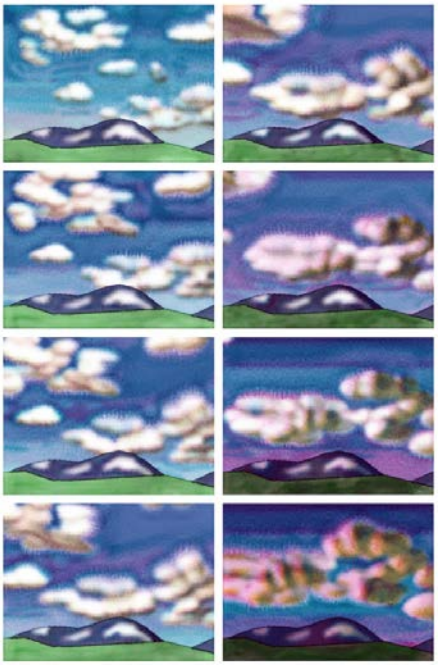


Figure 28 (a) – the 3D environment (b) – the watercolor frames of a moving clouds animation

In FIGURE 28 one can see the result of this application. In (a) we have the initial 3D environment and in (b) we can see some frames from an animation of moving clouds where each frame was watercolorized.

Conclusion

The watercolorization of images presents a new method of creating painterly images. This brings forth the obvious idea of using an evolutionary algorithm to generate an image and as an added bonus, apply the watercolorization algorithm to the resulted image to enforce a painterly look in all images. However, this method would not account for the other types of paintings, the most popular one being oil painting.

5. Future developments

With a focus on research

This section identifies and discusses five open problems that remain to be solved or addressed regarding generative art.

Searching for Interesting Phenotypes

The problem of producing unexpected results has been explored but there is still plenty of space for exploration. People have observed that when you start with a basic set of mathematical functions you end up with images looking similar. In order to improve the appearance you must introduce new mathematical functions and change the class of the output. However, a truly creative system should be able to introduce novelty within itself.

In order for this to happen, more research is needed to produce a system that is capable of evolving its attributes, introduce new attributes and evolve its mapping function thus achieving a level or creativity similar to the natural process of evolution.

The Problem of Aesthetic Selection

There is only a limited space of phenotypes that are produced and thus no system can expect to find the most beautiful picture that could be created since the limitations of the search are extremely constraining. First of all, all of the aesthetic selection is done by humans and we are such limited creatures when it comes to raw computation. People cannot be expected to compare a very large number of paintings in order to choose for the best one. It is approximated that you achieve the best results when you have to compare 6 or less paintings at once. Another limitation comes in the form of size which has to be preserved at an appropriate level in order for humans to have the ability to view it all at once on a computer. There is also no absolute beauty formula that people use to guide themselves, sometimes their mood and life experience can significantly influence their decisions.

Therefore more research is needed in devising a system capable of assessing how aesthetic a picture would be to a human without being necessary to actually ask a human being. Of course, this problem is extremely difficult as there are not many theories describing beauty in computational terms but there are some people who have devised some sort of mechanism to measure beauty with complex measures such as orderliness and harmony. Scientists need to try and translate these terms into computable attributes. One popular method includes Ralph's model of aesthetic which was described in detail in the Image synthesis section of this document. The pioneering work in this domain was done by Baluja et al. who used an artificial neural network to estimate the beauty of an image. The neural network was trained on example images that were considered to be aesthetically appealing but it was unclear it any specific principles were being applied when a new image was being assessed.

Machado et al. proposed a different model of aesthetics where an image would be considered beautiful if it were both complex and easy to interpret by the brain. These two features were translated into two mathematical functions that were used to compute how appealing a new image was. The first function computed the image complexity measured by the JPEG compression rate and the second function computed the visual self-similarity measured by the fractal compression ratio. Finally, another method

was proposed by Svangard et al. which would compute a distance metric between the new image to be evaluated and a collection of images that were stored in memory. The distance metric was computed based on how much information the two images shared. If there was more than one image in storage then it would concatenate all the genomes into one string and use it to compute the distance since in this manner it would use all the available information at once. The stored images were obtained by asking the user to pick them from a larger collection of images produced. After a while, the algorithm would start to produce picture that were more and more appealing to the user. The major drawback with this method was that it was not completely automated and it produced pictures considering only one user's preferences and therefore made no assumption of being able to evaluate a global level of beauty.

All of the methods proposed used some sort of similarity to known works of art to evaluate the level of beauty in newly generated paintings but there is still room for more research since using these methods rarely succeeds in producing beautiful paintings and cannot be expected to produce original beauty.

Produce art recognized by humans as creative

This problem is extremely non-trivial and has been compared to the Turing test in which a computer is challenged to talk to a human and fool him that he is talking to another human being not a machine. The goal here is to make people recognize the output produce by a machine as art. There is no clear direction yet on how to achieve that since humans tend to associate art with something that comes out of the hands of an artist, irrespective of how ugly and not interesting that art piece would be. There have been many theories suggesting that the artist pours his soul into his paintings, thus awakening feelings inside the hearts of the viewers. It is unreasonable to expect a computer to have a soul but it is very likely that a computer generated painting would stir an emotion inside the soul of a viewer. The trick here might be to not tell people that the paintings were generated by a computer. However, the method of creating paintings that would trick people into believing they were made by a real artist, remain open.

Create artificial ecosystems where agents create and recognize their own creativity

The challenge here is to create artificial ecosystems that produce agents who can create and recognize their own creativity. This is supposed to help us understand more about creativity and art following the concept of emergence. Basically, the goal would be to build a system that can explain to us humans what art is and what creativity really means by discovering new attributes and relationships that would illuminate us.

As exciting as this may seem, I believe that a lot more research is needed to solve the other open problems before finally attempting to solve this one since the goal is too vague and requires a lot of knowledge that we do not have yet as well techniques that we do not master sufficiently.

Develop art theories of evolutionary and generative art

The title here is self-explanatory. However, one important thing to mention is that we need to make a distinction between art theory and art criticism. There is extensive literature on how to analyze art in the context of a critical framework but now enough that describes art within a theoretical framework. If there was a basic set of rules defining what makes art beautiful or important then it could make a difference in getting computer generated art accepted for its artistic value.

6. Conclusions

Having explored the various algorithms used to produce art, it becomes clear that there is a way even if it still needs more paving. The evolutionary algorithms borrow ideas from nature to improve themselves until they produce acceptable results, even if the ultimate decision of success is taken by a human being. The concept of art is very complex and difficult to understand completely but no one ever doubts that it is meant to be appreciated by humans and humans alone.

However, the achievements of the evolutionary algorithms presented here are not to be taken lightly. They have indeed made progress and came closer to creating real works of art but the concept behind it must be questioned. The striking resemblance to the laws of nature might mean that we are indeed on the right path leading to success or it might mean that we are not capable of devising a method that is above the ways of mother nature. Taking a closer look at the nature of humanity, the wonderful result of millions of years of evolution, have we decided that we are pleased with the results? Maybe we have come to terms to the idea that although not perfect, it is as good as it gets and decided we need to copy it.

Although computers are able to produce art even with little human interaction, ultimately it is of no use for them because they cannot derive any pleasure or emotion from it. Moreover, they cannot truly understand their work and most important of all, they cannot understand their goals in a clear manner. It is extremely difficult for us to teach a computer what artistic value is because we do not know that on a purely theoretic level. The true meaning of art cannot even be put into words, let alone into equations. The closest we can get is to attempt to reproduce the color palette or distribution that we have observed in other works of art created by humans and that is simply not accurate enough.

Creativity can be programmed since it ultimately is an inspired choice among various random elements. The power of the human brain lies in the ability to produce true randomness and quickly evaluate it

before moving forward to the next option. However, when evaluating it, the artist does not follow a complicated equation or color distribution, he just follows his instincts most of the time. The true artist has a story to tell and he feels the need to share it with the world through his creation. His final work is not meant to be beautiful or pleasing, it is meant to express his feelings and ideals through colors and shapes. We recognize the works as beautiful and valuable because they strike a chord within us, they touch our souls allowing us to experience the passion and vision of the artist.

Perhaps the approach to computer generated art needs to be changed completely. Instead of trying to come up with an equation for beauty, we should try to identify equations that describe powerful human emotions such as love, serenity, anguish, desolation, hope, anger, fury and euphoria. Then use the evolutionary algorithms to combine and mutate them until the final work of art manages to spark an original and unexpected feeling inside its viewers, thus truly creating art that can be accepted by humans not because it resembles something visually pleasant, but because its meaning is deep and beautiful.

7. References

1. Baluja, S., Pomerleau, D., Jochem, T. (1994). **Towards Automated Artificial Evolution for Computer-generated Images**. Connection Science, 6(2/3): 325–354

2. Boden, Margaret A. and Edmo.nds, E111eStA.(2009) "**What is generative art**?", Digital Creativity, 20: 1, 21-46

3. **"Computer-Generated Graphite Pencil Rendering of 3D Polygonal Models"**, Mario Costa Sousa and John W. Buchanan, Volume 18 (1999), Number 3

4. "**Computer-Generated Watercolor**", Cassidy J. Curtis, Sean E. Anderson,Joshua E. Seims, Kurt W. Fleischer, David H. Salesin

5. Computers & Structures, Volume 83, Issues 23-24, September 2005, Pages 1943-1978, **Evolutionary computation and structural design: A survey of the state-of-the-art**, Rafal Kicinger[a] , Tomasz Arciszewski[a] and Kenneth De Jong

6. Dorin, A., "**Artificial Life, Death and Epidemics in Evolutionary, Generative Electronic Art**", Proceedings of 3rd European Workshop on Evolutionary Music and Art, Applications of Evolutionary Computing: EvoWorkshops 2005, Lausanne, Switzerland, March 30-April 1, 2005, Rothlauf et al (eds), Springer-Verlag Berlin, Heidelberg, pp448-457

7. Lewis, Matthew. "**Evolutionary Visual Art and Design**", chapter in The Art of Artificial Evolution, Penousal Machado and Juan Romero (Eds), Springer, 2008

8. "**The Evolution of Artistic Filters**", Craig Neufeld, Brian J. Ross and William Ralph, The Evolution of Artistic Filters, Craig Neufeld, Brian J. Ross and William Ralph, THE ART OF ARTIFICIAL EVOLUTION, Natural Computing Series, 2008, Part V, 335-356, DOI: 10.1007/978-3-540-72877-1_16

9. "**Evolutionary Art Using Summed Multi-Objective Ranks**," Steven Bergen and Brian J. Ross, GENETIC PROGRAMMING THEORY AND PRACTICE VIII, Genetic and Evolutionary Computation, 2011, Volume 8, 227-244, DOI: 10.1007/978-1-4419-7747-2_14

10. Lewis, M., **Aesthetic Evolutionary Design with Data Flow Networks**, in Proc. Generative Art 2000, 2000.

11. Machado, P., Cardoso, A. (2002). **All the Truth About NEvAr**. Applied Intelligence,16(2): 101–118

12. "**Open problems in evolutionary music and art",** J McCormack - Applications on Evolutionary Computing, 2005 – Springer

13. RoboRealm**, Sobel Edge Filter**, http://www.roborealm.com/help/Sobel.php

14. Ross, B., Zhu, H. (2004). **Procedural Texture Evolution Using Multiobjective Optimization**. New Generation Computing, 22(3): 271–293

15. Ross, B.J.; Ralph, W.; Hai Zong; , "**Evolutionary Image Synthesis Using a Model of Aesthetics,**" Evolutionary Computation, 2006. CEC 2006. IEEE Congress on , vol., no., pp.1087-1094,0-00
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1688430&isnumber=35623

16. Side Effects Software. **Houdini**. 2000, http://www.sidefx.com

17. Svangard, N., Nordin, P. (2004). Automated Aesthetic Selection of Evolutionary Art by Distance Based Classification of Genomes and Phenomes using the **Universal Similarity Metric**. In: Applications of Evolutionary Computing: EvoWorkshops 2004. Springer, 447–456 LNCS 3005.

18. Thomas Ray, **TIERRA**, http://www.talkorigins.org/faqs/tierra.html

19. Wiens, A., Ross, B. (2002). **Gentropy**: Evolutionary 2D Texture Generation. Computers and Graphics Journal, 26(1): 75–88

20. Larry Yaeger's, **PolyWorld**, http://www.beanblossom.in.us/larryy/polyworld.html#ALifeXII

Appendix I

Shallow water equations

The Shallow Water Equations are a system of hyperbolic/parabolic Partial Differential Equations governing fluid flow in the oceans, coastal regions, estuaries, rivers and channels. The general characteristic of shallow water flows is that the vertical dimension is much smaller than the typical horizontal scale. In this case we can average over the depth to get rid of the vertical dimension. The SWE are derived from the Navier-Stokes equations, which describe the motion of fluids. The Navier-Stokes equations are derived from the equations for conservation of mass and linear momentum.

The underlying assumption is that the depth of the fluid is small compared to the wave length of the disturbance. For example, we do not ordinary think of the Indian Ocean as being shallow. The depth is two or three kilometers. But the devastating tsunami in the Indian Ocean on December 26, 2004 involved waves that were dozens or hundreds of kilometers long. So the shallow water approximation provides a reasonable model in this situation.

The equations are derived from the principles of conservation of mass and conservation of momentum. The independent variables are time, t, and two space coordinates, x and y. The dependent variables are the fluid height or depth, h, and the two-dimensional fluid velocity field, u and v. With the proper choice of units, the conserved quantities are mass, which is proportional to h, and momentum, which is proportional to uh and vh. The force acting on the fluid is gravity, represented by the gravitational constant, g. The partial differential equations are:

$$\frac{\partial h}{\partial t} + \frac{\partial (uh)}{\partial x} + \frac{\partial (vh)}{\partial y} = 0$$

$$\frac{\partial (uh)}{\partial t} + \frac{\partial (u^2 h + \frac{1}{2} g h^2)}{\partial x} + \frac{\partial (uvh)}{\partial y} = 0$$

$$\frac{\partial (vh)}{\partial t} + \frac{\partial (uvh)}{\partial x} + \frac{\partial (u^2 h + \frac{1}{2} g h^2)}{\partial x} = 0$$

Figure I – A water drop starts a wave that reflects off the boundary

Kubelka-Munk equations

In typical applications of KM theory, the absorption coefficient *K* and scattering coefficient *S* for a given colorant layer are determined experimentally, using spectral measurements from layers of known thicknesses. However, in the application presented here in section 4, it was more convenient to allow a user to specify the *K* and *S* coefficients interactively, by choosing the desired appearance of a "unit thickness" of the pigment over both a white and a black background. Given these two user-selected *RGB* colors *Rw* and *Rb*, respectively, the *K* and *S* values could be computed by a simple inversion of the KM equations:

$$S = \frac{1}{b} \cdot \coth^{-1} \frac{b^2 - (a - R_w)(a - 1)}{b(1 - R_w)}$$

$$K = S(a - 1)$$

where

$$a = \frac{1}{2}\left(R_w + \frac{R_b - R_w + 1}{R_b}\right)$$

$$b = \sqrt{a^2 - 1}$$

The above computations are applied to each color channel of S, K, R_w, and R_b independently. In order to avoid any divisions by zero, it is required that $0 < R_b < R_w < 1$ for each color channel.

Given scattering and absorption coefficients S and K for a pigmented layer of given thickness x, the KM model allows to compute reflectance R and transmittance T through the layer:

$$R = \sinh bSx / c$$

$$T = b / c$$

$$c = a \sinh bSx + b \cosh bSx$$

Then they use Kubelka's optical compositing equations to determine the overall reflectance R and transmittance T of two abutting layers with reflectances R_1, R_2 and T_1, T_2, respectively:

$$R = R_1 + \frac{T_1^2 R_2}{1 - R_1 R_2}$$

$$T = \frac{T_1 T_2}{1 - R_1 R_2}$$

The computation is repeated for each additional glaze. The overall reflectance R is then used to render the pixel. For individual layers containing more than one pigment of thicknesses x^1,…, x^n ,the S and K coefficients of each pigment k are weighted in proportion to that pigment's relative thickness x^k. The overall thickness of the layer x is taken to be the sum of the thicknesses of the individual pigments.

In their fluid simulation they use g^k to denote the concentration of pigment in the shallow-water layer, and d^k for the concentration of pigment deposited on the paper. These values are summed to compute the thickness parameter x^k used by the Kubelka-Munk equations.