Non-Photorealistic Rendering with an Ant Algorithm

Massimo De Marchi
Dept. of Computer Science at UIC
900 W Taylor St, Chicago, IL 60607
mdemar5@uic.edu

Gianluca Venturini
Dept. of Computer Science at UIC
900 W Taylor St, Chicago, IL 60607
gventu4@uic.edu

ABSTRACT

Generative art is a contemporary trend that in whole or in part uses autonomous systems to produce creative artworks. In this field an interesting emerging technique inspired by nature is Swarm Intelligence where the interaction of a group of agents that follow simple rules lead to the emergence of an intelligent global behavior. In this paper we are showing a swarm art technique that produces non-photorealistic rendering drawings. To design our algorithm we were inspired by a technique called "Photogrowth" [5] that uses a painting algorithm inspired by ant colony approaches to create large scale non-photorealistic renderings.

INTRODUCTION

Generative art can be obtained using three different approaches, namely ordered, disordered, or complex. Ordered systems include the use of symmetry, tiling, numeric sequences and series or proportions such as the golden ratio. Disordered systems usually exploit randomization or chaos theories. Complex systems combine both order and disorder and typically this lead to emergence, namely the process in which larger entities arise through interaction of smaller entities. Example of complex generative art systems employ evolutionary algorithms, neural networks and ultimately biologically inspired methods such as Swarm intelligence.

Two of the most interesting subfields of generative art are evolutionary art and swarm art. The techniques that these subfields employ are inspired by nature but they differ in the way they work. In evolutionary art evolutionary algorithms are used to pursue an aesthetic ideal by optimizing a given fitness function. Conversely swarm art employ more unpredictable techniques such as ant systems.

In this paper we are going to show an ant algorithm that we designed, able to produce several different kind of non-photorealistic rendering drawings. The algorithm is inspired by a technique named *Photogrowth* proposed by *Machado et all* [5]. Photogrowth exploits evolutionary algorithms to evolve filters that transform an input source image. The filter is based on ants colony behaviors, namely it exploits the trails of artificial ants to produce artistic rendering. The genetic algorithm is used to evolve the ants behaviors by tuning parameters such as ant speed and possible movement directions.

Figure 1. Example of image generated using Photogrowth evolutionary algorithm.

RELATED WORKS

One closely related work is the one described in [3], in which the algorithm is based on KANTS as a swarm art tool. KANTS (Kohonen Ants) is an ant-base algorithm proposed by [2] for data clustering and classification. The method tunes simple parameters of an ants colony to move the system toward a self organized map where global behaviors emerge. This approach uses input data vectors as artificial ants; when the input of the system is a photograph, the vectors represent its RGB values. Ants communicate via the environment in a process known as *stigmergy* and they are attracted by similar ants. This process changes also the environment and is implemented in the KANTS approach by adjusting the vectors that the ants visit toward their vectors.

An interesting related work has been done by *Love* [4] in which a swarm-based multi-agent system produces expressive imagery using multiple digital images; In the described system every group of agents has an aesthetic ideal represented by the input photograph. As agents moves through the digital canvas they try to reproduce their aesthetic ideal, and when groups with different aesthetic ideals collide a new image is created by the convergence of the different ideals.

Figure 2. Example of image generated with KANTS algorithm.

Figure 3. Example of rendering generated by using multiple group of artificial agents.

Baniasadi et all [1] presented a genetic programming algorithm that evolves an input image through non-photorealistic effects. Interestingly the fitness evaluation function that they use is based on the Ralph's model that describes how the human visual response is attracted by logarithmic changes in colors; In this way the evolutionary algorithm evolves the image toward human aesthetically pleasant changes of colors according to the Ralph's theory.

ALGORITHM OVERVIEW

As mentioned before our algorithm is inspired by a technique called *Photogrowth* that models simple ants behaviors to produce non-photorealistic rendering drawings. The ant behaviors are modeled with several parameters and changing these parameters allows to produce tons of different effects as we will see in the section about experimental results.
The basic idea behind the algorithm is that a set of ants governed by certain behaviors run across the input photograph absorbing energy and leaving their pheromone on a painting canvas. The energy is represented by the luminance of the pixel of the input photograph, that is the brighter the pixels is the higher is the energy that an ant can absorb from it. The actual amount that an ant absorb from the pixel is given by the parameters of the species and this allows the algorithm to draw with different level of transparency. The consumption of energy of the living canvas is due to the fact that when an ant absorbs energy from the pixel, that pixel is updated by subtracting the absorbed amount from each RGB channel. In this way the pixel is progressively updated toward the black color, which represents the absence of energy. The pheromone is represented by the ink deposited in the painting canvas, that is the final effect is a composition of all the trails of the ants that have moved around the canvas.
The movement of the ants is governed by their sensors and by the distribution of energy in the input photograph. The ants use their sensors to seek for areas with high energy, namely the area of the photograph with the brightest pixel colors. Sensors tells the ant the direction and the distance to which look at, and each sensor has also a weight. Changing direction, distance and the weight of the sensor affect the way the ant move through the living image an thus the final result of the rendering.
The life of the ant is ruled by the amount of energy that it has, and by its capability to acquire new energy and resist to low energy levels. When an ant has not enough energy it dies, and when it has too much energy it generates offspring, namely children. Configuring these parameters is also an important task because they dramatically affect the final result. Depending on the energy of the input image it is important to find the correct parameters so that the ants do not die as soon as they appear and can survive enough time to produce something relevant.

Figure 4. Example of an abstract painting generated using evolutionary programming.

In the next section we are going to describe how we implemented these behaviors on GPU.

IMPLEMENTATION ON GPU

The algorithm is based on multiple elaboration of textures. Every frame is a step of the algorithm and produce an intermediate result that is immediately displayed. It can be stopped by the user at any time producing paintings at different level of completion.
The process uses three different textures:

- *Living*: initial image that the user can choose.
- *Painting*: final result of the computation that is displayed, incrementally painted.
- *Ant*: contains the position, the direction and the energy of the ants. Every ant is represented using one pixel, the direction is encoded using the red and green channels and the energy is encoded by the blue channel.

To obtain the maximum efficiency all the computations are executed on the GPU, apart for the living texture which is copied from the CPU memory at the beginning of the execution.
Every frame four scenes are rendered and the result of every rendering is stored in one of the three textures or displayed directly on the screen. We used this approach in order to maintain a modular structure of the program and to overcome the limitation imposed by the GPU rendering pipeline, that is the fragment shader can access and change one fragment at a time.
The scenes are:

- *Ant scene*: renders the new position, direction and energy of the ants based on the luminance of the surrounding pixels. The inputs are the old ant texture and living texture and the output is the new ant texture.
- *Living scene*: renders the new living texture. The input are the ant texture and old living texture. The colors of the old living texture are attenuated in this computation based on the position of the ants and their energy. The purpose of removing color on the living scene is to dissuade the ants to repaint the same pixel infinite times and move to areas not yet painted.
- *Painting scene*: renders the new painting texture. The input are ant texture and painting texture. The ink is added to the painting based on the ant position and energy: ants with high energy deposit larger circles than ants with low energy. The transparency of the deposited ink is proportional to the parameter *depositTransp*.

There are 12 parameters that describe the ant species. The web application we developed allow the user to change them in order to create different paintings. The movement of the ants depends on those parameters and also on the precomputed sensors of the ants. Sensors are vectors that describe a distance and a direction for picking a pixel and measuring the luminance of it; every ant has 10 of them, but this setting could be change to obtain different behavior. The final direction of the ant is the average of all this vectors weighted on the luminance of the corresponding pixels. The user cannot change vector number or position because we observed that with little variations in this vectors the algorithm become unstable.

Parameters

In this sub section the main parameters of the algorithm are presented.

- *gain, decay*: used in order to calculate the amount of energy that an ant receive and lose at every step.
- *depositRate*: the radius of the circle deposited on the canvas for every ant. Bigger values result in less detailed painting and bring to the rendering of *pointillism*-like paintings.
- *depositTransp*: the amount of ink that is deposited on the canvas. Small values below 0.1 give semitransparent painting.
- *initialEnergy*: initial energy of the ant, if it is near zero ants tends to die right after their creation, usually values around 1 are pretty good.
- *deathTreshold*: energy below which the ant will die.
- *offspringTreshold*: energy above which the ant will generate a second ant (offspring).
- *vel*: base speed of the ants. High speed gives less detailed painting.
- *noiseMin, noiseMax*: modify the amount of randomness in the ants movement.

Advantages

This algorithm can successfully produce a large variety of artistic effects from an input photograph due to the fact that the user can tune many parameters. Among the possible effects there are pointillism and oil on canvas. It is also possible to produce a high range of different abstract effects. The user is completely free to create her own personal art technique and apply it to every image she desires. Since the algorithm can be stopped at every time the user can leave a painting partially incomplete.

Disadvantages

The main limitation of this implementation is the constraint of the GPU. In every scene for every rendered pixel it is necessary to calculate the final color that depends also on neighbor pixels. This is a limitation because it requires to parse an N×N grid to obtain the data required. This operation is computationally expensive and for that reason we limited the grid dimension to 13x13 pixels. A single ant can't move more than 5 pixels per frame and can't paint a circle with radius that exceeds 5 pixels.

(a) Input photograph

(b) Painting produced using small depositRate=1 and decay=0.5.

(c) Another pointillism drawing with more details due to smaller ants and higher transparency.

(d) Painting produced using low deposit rate (deposit

Figure 5. This set of pictures shows how different effects can be created with the technique described.

EXPERIMENTAL RESULTS

The results presented in this section have been obtained by tweaking the ant species parameters. Every presented painting is different from the others and this shows the endless possibility that this algorithm allows. The application we designed is interactive, the user can place one or more ants on the canvas, change the species parameters, save them in order to save effects, stop at any time the rendering and save the final result.

The final result depends also from the starting position of the placed ants and for this reason it is almost impossible to produce exactly the same result more than once, even tough similar results can be obtained. This makes every rendering unique in a sense, and uniqueness is a property typical of art produced by humans.

The result in figure 6 is obtained using no transparency (depositTransp = 1), setting an high level of initial energy for the ants (initialEnergy = 1) and decay lower than 1. In this way they rapidly create the colony and fill every space, but they die when they encounter areas with low luminance due to the decay factor smaller than one. The final effect produced is beautiful and very abstract in that it dramatically simplifies the face of the woman and only eyes, mouth and nose can be recognized. The second drawing shown in figure 5(b) is produced using smaller ants (depositRate=1 that means every ant can paint at most one pixel) and a low decay (decay=0.5 that means ants die rapidly). As we can see lowering the size of the trails of the ants result in a more detailed drawing. This painting is particular because of its asymmetry. It was created positioning only one ant in the center of the face, changing the position will change the part of the woman that will be painted. Another particular result was obtained increasing the dimension of the ants (depositRate=1.5) and it is shown in figure 7. In this painting transparency is added (depositTransp=0.1), and ants with low energy paint semi-transparent smaller lines. This effect is very similar to the *pointillism* art technique. The reason why all the points are concentrated in areas with high luminance is that ants are fed with luminance (that is absorbed from the original image). The points are proportional to the energy of the ants and to the consRate

parameter. In figure 5(c) the effect is similar to figure 7, but with more details. This variation was obtained starting from the previous result and tweaking some parameters (changing them randomly). The drawing in figure 5(d) is the most interesting one. It can emulate very well the oil on canvas art technique. This painting is produced using a very low deposit rate (depositRate = 0.03) in order to let the ants run more times on the same pixels. Moreover the *offspringThreshold* is very low (*offspringThreshold* = 0.5) and this results in a lot of ants generated and destroyed by the low decay (*decay* = 0.5).

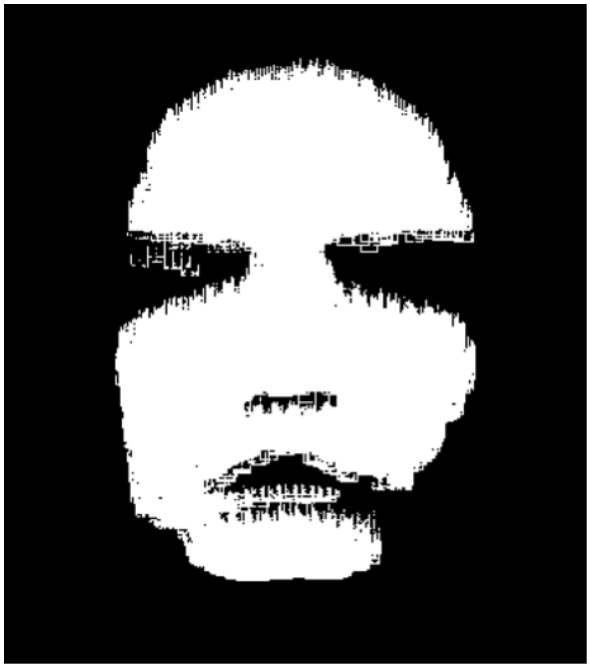

Figure 6. First drawing obtained without using transparency, the result is an abstract painting

Figure 7. Painting produced introducing transparency (depositTransp=0.1) and depositRate=1.5, it imitates pointillism art technique.

CONCLUSIONS

The algorithm that we implemented has proved to be a very effective tool for non-photorealistic rendering. The approach inspired by nature incorporate an intrinsic chaotic behavior that leads to very artistic effects; nonetheless the artist can guide the algorithm to produce an endless number of different effects by tuning the parameters that rule the ants behavior. The unpredictability of the ants movements leads also to produce unique paintings and uniqueness is a property typical of human art.

This technique is open to several improvements, for example a genetic algorithm can be used to pursue a certain aesthetic ideal indicated by a human artist, or for capturing configurations that are esthetically pleasant for a human being. Another interesting improvement could be to empower the ants with coloured ink.

Overall the obtained result is satisfactory because the algorithm is already capable of producing aesthetically pleasant drawings and its flexibility allow countless different effects from the same input photograph.

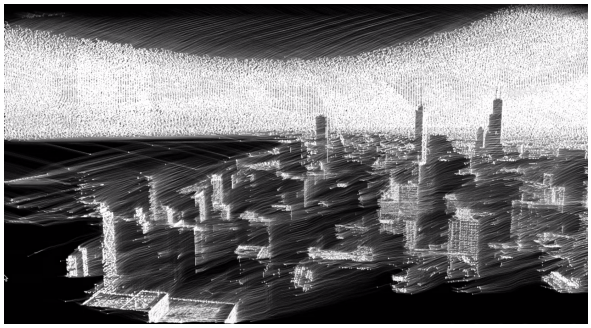

Figure 8. A non-photorealistic rendering of Chicago obtained with our algorithm.

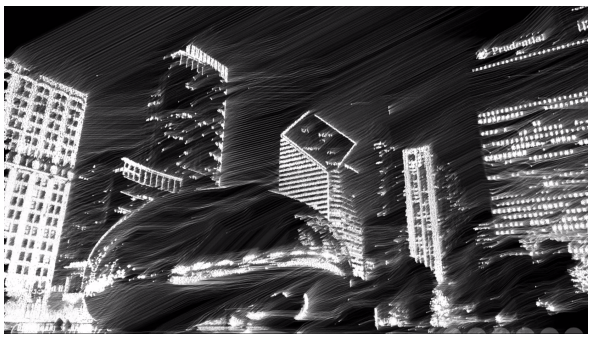

Figure 9. A non-photorealistic rendering of the Cloud Gate in Chicago.

Figure 10. A non-photorealistic rendering of the portrait of a girl.

REFERENCES

1. Baniasadi, M., and Ross, B. J. Exploring non-photorealistic rendering with genetic programming. *Genetic Programming and Evolvable Machines* (2013), 1–29.
2. Fernandes, C., Mora, A. M., Merelo, J. J., Ramos, V., and Laredo, J. L. J. Kohonants: a self-organizing ant algorithm for clustering and pattern classification. *arXiv preprint arXiv:0803.2695* (2008).
3. Fernandes, C. M., Mora, A. M., Merelo, J. J., and Rosa, A. C. Photorealistic rendering with an ant algorithm. In *Computational Intelligence*. Springer, 2015, 63–77.
4. Love, J. *Aesthetic agents: experiments in swarm painting*. PhD thesis, 2012.
5. Machado, P., and Pereira, L. Photogrowth: Non-photorealistic renderings through ant paintings. In *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference*, ACM (2012), 233–240.