EVOLVING IMAGES USING TRANSPARENT OVERLAPPING POLYGONS

William Tarimo, Chen Xing, Linyu Dong, Chao Li

Brandeis University

ABSTRACT

Our project attempted to use machine learning to recreate or redefine features of an image using an arrangement of transparent overlapping polygons. From genetic algorithm[1] and hill climbing[2], we came up an implementation that starts by generating a random sequence of polygons then iteratively mutating the sequence (slightly modifying a random attribute of a random polygon), incrementally building on mutations that yield results that are closer to a target image[3]. In the context of evolving images using polygons, we learned and explored the balance between visual appeal of the generated images and the efficiency of the implementation.

***Index Terms*—** Image Evolution, Hill-climbing, Genetic Algorithms, Image Processing

1. INTRODUCTION

1.1. Motivation

In this project we attempted to provide an implementation to the question whether it is possible to recreate images using transparent overlapping polygons. Our inspiration is based on the image processing coursework we have covered, the use of machine learning to create art, and a handful of similar projects we came across on the internet. With this idea, we aimed at exploiting the power of machine learning to recreate image features in a non-conventional way, as opposed to the pixel-based image representation.

1.2. Problem Definition and Theoretical Approach

By transparent overlapping polygons we mean multiple polygons with varying degrees of transparency and colors. And by arrangement we imply an optimal positioning of the polygons on a canvas in such a way that defines the features of an actual target image.

The structure is an array of polygons each with different color, transparency, and a set of (x,y) coordinates defining the polygons location and shape on the canvas. We refer to this array of polygons as an image genome.

This setup calls for a search and optimization approach where the search space is the multitude of all possible positioning, colors, transparencies and other attributes that can be defined for the sequence of the polygons. The optimization aspect is based on the fact that we have a target image that we are attempting to recreate, and thus the goal is to learn a set of attributes for the sequence of polygons such that the image they form is as close as possible to the target image.

To this end, we came up with an implementation that is based on search and optimization aspects of genetic algorithms and the hill climbing algorithms.

So far its obvious that re-creating images with satisfactory visual appeal would require many polygons and thus many attributes and computational time. To this end, we decided to design an efficient implementation model. The setup requires the following global parameters; a target image, the dimension of the image, the number of polygons, and vertices, and the number of generations that controls the image evolution loop. The program starts by generating a random genome sequence, this is the initial parent. Then, loops for the given number of generations, at each iteration do: 1. Slightly mutate the parent genome to make a new daughter genome, 2. Render both parent and daughter genomes into images, 3. Calculate their fitness scores relative to the target image, 4. If the daughter genome is closer to the target image than the parent then the daughter becomes the new parent, discarding the old parent genome, otherwise keep the parent - discarding the less fit daughter 5. Go back to step 1. The program flow is visually depicted in Figure 1.

2. METHODOLOGY

2.1. Image Genome Structure

An image genome is a cell-array of polygons. Each polygon is itself a cell-array of attributes defining that polygon. The attributes are color value (a single gray value or an array of red, green, blue values for color images) - value range [0-255], a transparency/alpha value - range [0-1], and a cell-array of vectors X and Y with values representing (x,y) coordinates for each of the vertices of the polygon - X range [0 - width of the target image] and Y range [0 - height of the target image]. An example genome would look like 250, 0.4, [12, 42, 35], [43, 65, 87], 34, 1.0, [32, 65, 43], [90, 53, 10], ...

Thanks to Antonella Dilillo for teachering.

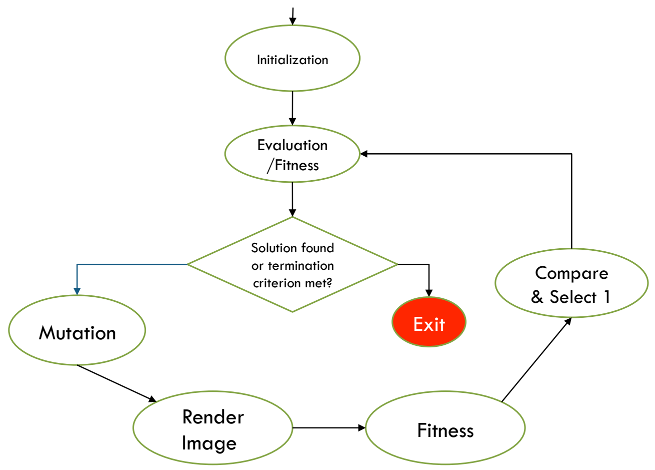

Fig. 1. Implementation of image evolution with adapted GA and Hill-climbing.

2.2. Mutation Operations

Mutation is simply slightly modifying a random part of the genome with the hope to get a new better genome. Based on the mutation function used, the function picks a random polygon from the genome, and then picks a random attribute to modify according to a mutation function. In our implementation we started with two mutation functions, medium and soft mutations. Medium mutation replaces the picked attribute to a new random value within the attribute range. Soft mutation changes the picked attribute by adding +/-10% of its initial value. We then combined soft and medium mutations at ratio 2:1 to make a new function, hybrid-mutation, which performs better than the other two when used individually.

2.3. Image Rendering: From a genome to an image

To create an image from a genome sequence of polygons, we create a black image canvas, then draw each polygon on the canvas one by one. The final result is an image whose features are defined by the genome.

2.4. Genome Fitness Function

The fitness of a genome is the measure of how close its rendered image is to the target image. To calculate this value, sum up all the absolute pixel-by-pixel differences between the genome image and the target image. The lower this value is, the more alike the two images are, and the better the genome.

3. EXPERIMENT AND RESULTS

3.1. Image Types

In this project we implemented image evolution programs for truecolor and grayscale image types. Shown in figures 2 and 3 are sample evolved grayscale and truecolor images respectively. We wrote two separate code modules for evolution the two color types because they have different variable dimensions and color channels. Evolution with color images is slower than gray images due to the additional color attributes (r,g,b) which which requires more mutations.

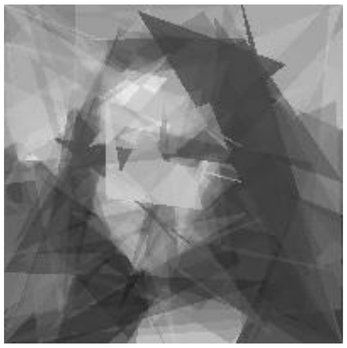

Fig. 2. Sample evolved image in grayscale implementation

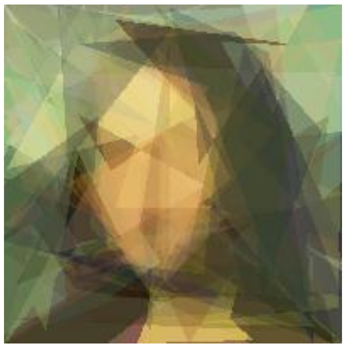

Fig. 3. Sample evolved image in truecolor implementation

3.2. The Selection of Mutation Function

There are different types of mutation function, medium mutation, soft mutation, and hybrid mutation. In the experiment, we find the performance of hybrid mutation function is better than both median mutation and soft mutation (shown in figure 4).This demonstrates the efficiency of diverse mutation function, efficient evolution requires both large and small random genome modifications as long as there is a fitness function controlling the desired optimization.

3.3. The Number of Polygons

The number of polygons is a parameter of genome structure. The results shown in figures 5 and 6, we used one genome with 15 octagons and another with 45 octagons. We could find that the output image using 45 octagons is much closer to target image, and it also takes longer time to evolve. In another

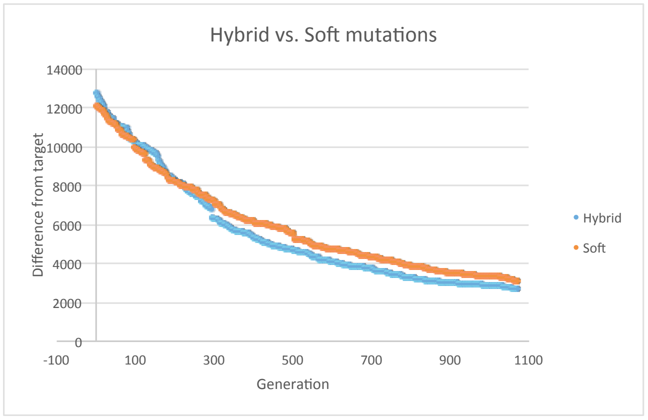

Fig. 4. Hybrid vs. Soft mutations

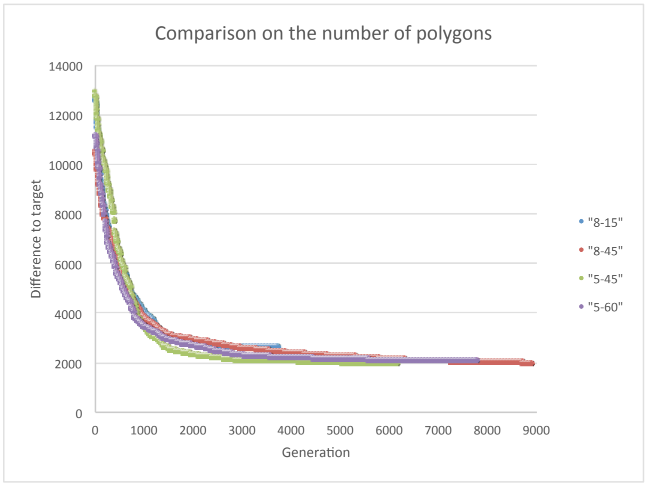

Fig. 7. Comparison on the number of polygons

experiment (shown in figure 7), we used one genome with 45 pentagons and another genome with 60 pentagons to generate. Then we use the number of generation as X-coordinate, the fitness score as Y-coordinate, to represent the relation between them. In the graph, it is obviously that using more polygons to evolve would result in lower fitness score which means much closer to source image. However, it will take much longer time and higher computations. So the best way is to find the balance between fitness score and total costs, which depends on the resolution and size of target image.

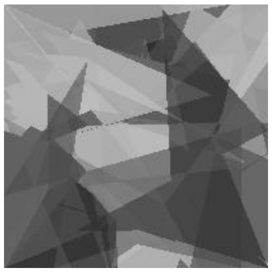

Fig. 5. Image evolved with 15 octagons

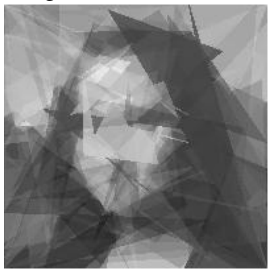

Fig. 6. Image evolved with 45 octagons

3.4. The Number of Vertices

The number of vertices is another parameter of genome structure. In the experimental results shown in figure 8, we used the number of generation as X-coordinate, the fitness score as Y-coordinate to represent the fitness score of different types of polygons. We could get a conclusion that, there is an optimal value for the number of polygon vertices for good visually appealing results. Too many vertices would result in extra computations and longer evolution time, where as too few vertices would require more polygons to effectively define the target image features.

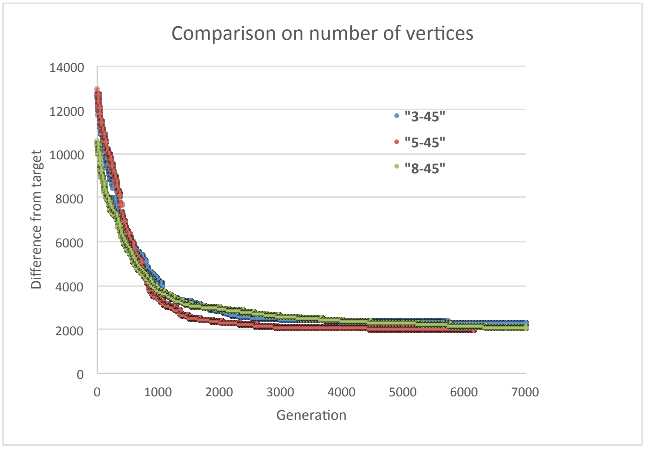

Fig. 8. Comparison on number of vertices

4. CONCLUSION

In the most ideal situation, if we could afford time cost and computation power, we could always use more polygons and an optimal number of vertices to evolve high-resolution images which will be very close to target replicas. If the target image has high resolution and size, then we need to find the optimal genome structure to evolve images with best quality and efficiency. Meanwhile, the mutation function also plays important role in affecting the efficiency of algorithms. The hybrid mutation, which combines medium mutation and soft mutation with ratio 1:2, shows the effectiveness of mixed or adaptive mutation functions.

5. REFERENCES

[1] Fernando Buarque de Lima Neto Renata L. M. E. do Rego, "Evolutionary algorithm for 3d object reconstruction from images," *Neural Networks*, pp. 54–59, 2006.

[2] Steve R. Bergen, "Evolving stylized images using a user-interactive genetic algorithm," *ACM*, pp. 2745–2752, 2009.

[3] Karl Sims, "Artificial evolution of computer graphics," *Computer Graphics*, vol. 25(4), pp. 319–328, 1991.