Applications of Genetic Algorithms to Procedural Image Generation

Akhila Ananthram & Griffin Brodman
asa225 & gb282

December 16th

1 Introduction

Genetic Algorithms are a very interesting and unexplored area of Computer Science. We decided it would be interesting to apply genetic algorithms to different solved problems, and see if we could find improvements in performance.

The problem we decided to analyze was that of generating an image out of translucent polygons. We found a project (SOURCE) that given a source image, will modify a canvas by adding polygons, removing polygons, adding or subtracting vertexes to polygons, moving polygons, changing their color or their transparency. He claimed that he was using genetic algorithms, but on inspection, he had mutation but no crossbreeding elements. He would generate a child, see if this was closer to the source image, if it was keep it, if else, throw it away. This isn't really an example of genetic algorithms without crossbreeding, this is just basic hillstepping. We wondering if genetic algorithms could be applied to this problem, and even show some improvements in performance.

We theorized that looking at how genetic algorithms change the performance of this algorithm would give us a much better understanding of real world applications of genetic algorithms, and what their best use case is. From here, we could more accurately identify problems that would benefit from this approach to solving problems. We will vary our approach to genetic algorithms, and try a bunch of modifications to it, modifiers like elitism and multiple parents. We'll then compare the amount of iterations it takes, on average, for hill climbing vs genetic algorithms to get to a certain fitness threshold when compared to the source image.

WE FOUND

2 Problem Definition and Methods

2.1 Task Definition

We wanted to analyze whether hill climbing or genetic algorithms would have better performance in solving the problem of generating an image with procedurally generated polygons. We would see, in numbers of iterations, how long it would take our multiple methods to get to a certain fitness. We'd first run it under hill climbing, then under a variety of genetic algorithm modifications.

2.2 Algorithms and Methods

Our hill climbing algorithm is simple. We mutate an array of polygons, either adding or removing a polygon, adding or removing a vertex to a polygon, or changing the color or transparency of a polygon. If this new array has a better fitness than the old away we keep it, else we throw it away.

Our basic genetic algorithm is simple FILL IN HERE

We developed two different fitness functions, and sampled using both. Our basic fitness function was one that went pixel by pixel, and calculated the euclidean distance between the two images. Our second one relied on opencv feature matching. ADD DETAIL HERE

3 Experimental Evaluation

3.1 Methodology

3.2 Results

4 Related Work

5 Future Work

There were a few things that developed as we worked on the project. For one, working in python severly limited our speed. We did get the results we needed, but we were constrained to smaller images. It would have been nice to support much larger images, which would have opened up different fitness functions to us, as it would have improved feature matching. The project we had used as inspiration had been written in c#, and though we didn't think the difference in languages would have such a significant effect, it seems to be huge.

6 Conclusion

I'd l