Problem Description

Note that, we have scaled down the level of the project from the actual proposal. And, I apologized for the unexpected results on the code and document due to my poor time management.

Our project aims to convert a simple image of a full black Eiffel tower with size `(180,279)` into a specific art style using only circles. The art style is defined as follows: 10-25 shapes are used in total, with each circle having a size of 20 pixels. The shape may overlap, but no more than 50 % of one shape can be covered by another. The backgrounds are full white with RGB (255,255,255), and circles are full black with RGB (0,0,0).The circles should be evenly distributed throughout the image, avoiding over clustering (the area intersection of 3 circles are more than 50% of the area of single circle) in any particular area. Each circle

To accomplish the above, we use genetic models.First, we generate population images through placing 15 - 20 circles. Each image is evaluated with the fitness function MSE to determine similarity of images with given input. We do a tournament selection by selecting a subset of candidate images from the generation phase and allow them to compete based on our fitness score. Then, the best performing images undergo mutations and crossovers. Mutation involves moving the location, changing the circles. For crossover, we select a subset of images from a mutated population. Then, we extract the circle's x-y coordinates from those subset images. The extracted list of circles will become an offspring. Later, we send this population that went through crossover and populate back to our initialization phase. Lastly, we repeat the process till we satisfy our criteria.

After conducting extensive research, we have concluded that a genetic algorithm is the best approach for our project. Genetic algorithms are frequently used to solve optimization problems and are well-suited to explore large search spaces to find an optimal solution. In our case, our objective is to transform a real image of the Eiffel tower into a newly generated one while ensuring that it meets our specific defined art style constraints. Although other techniques such as neural style transfer can produce high-quality images, they may not meet our specific art style properties. In contrast, genetic algorithms are more suitable for generating images that satisfy our art style properties

Method

We create the code from scratch to better meet our own requirements. First, we create a circle object that initializes its x-y coordinates on canvas space. We prepared the size of the circle and step size to control the magnitude of shifting the circle. Next, we use numpy arrays to grayscale our target and generated images,and convert them to a 2D array. Each 1D array corresponds to each row pixel. Then we subtract the difference and take the average of those. Now, we normalize them to spit values between 0 and 1. A value closer to 0, indicates the generated is similar to target.

For the paint object, we generate a list of circle objects. Then, we use the list to create the canvas. Draw.ellipse() from PIL library is used for placing a single circle onto the canvas. We

can mutate the portion of circles in a single paint. Also, once a paint is initialized, mutation, crossover, we create a new sheet of canvas and update the fitness score.

Once we define these 2 main objects, we can use our generate_art() function. This function purpose is to iterate the entire process of the genetic model. In this function, we pass in all the parameters needed from generation to crossovers. The main parameters are explained as follows: radius: controls the size of circles, step_size: how large a circle shifts, population_size: number of candidate images at initial, mutateRate: controls the degree of the circle shifting positions. After initialization, we have tournament_size: select the number of subset from initialization population, NumWinner: number of tournaments to do. At crossover, we have subset_size : number of images to select for crossover, num_offspring: number of images we will have for a single iteration.

Materials

List of Required Resources Physical Resources: The genetic algorithm for image generation requires a Core i7 processor, 16 GB RAM, and a 4GB graphics card. Informational Resources: We require knowledge to guide the project, such as implementing genetic algorithms for AI art and image processing details. Resources can be obtained from IEEE Xplore Digital Library and arXiv. An architecture image is also necessary and can be sourced from Google or Kaggle. Personnel Resources: Dr. Brown and online forums can provide additional assistance.

Evaluation

For a subjective measurement, we sampled 6 people from TRU's dormitory, and asked each of them to evaluate the figure 1.0 (generated image). We asked the following question. Rate the similarity between generated and target images from the scale 0 to 5? 2) Does this look like a tower ? We summarized the data in the plot where the left corresponds to the first question. And the right will be the 2nd question.

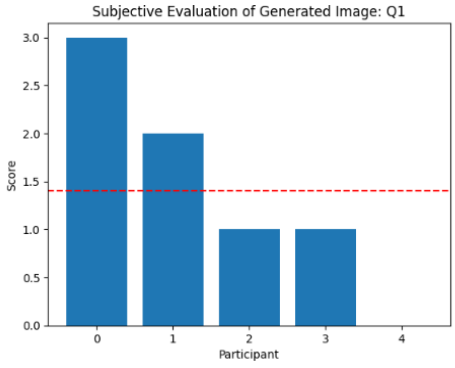


Figure a

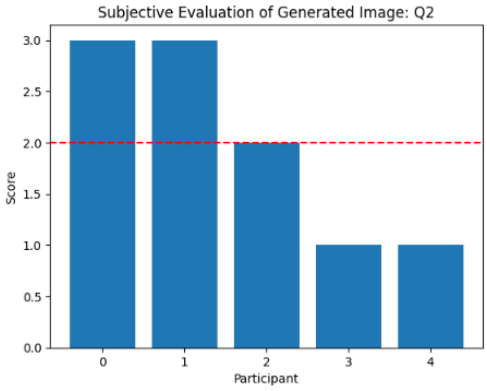


Figure b

Figure 1.0

As we observed, we see that the sample average is below the theoretical average. Thus, we can conclude that the results for our subjective side are a failure.

Now, we consider our objective side. This is based on taking the average values of fitness score

after a population of images went through crossover for 1000 iterations. The average value we obtained was `0.010137200240510381`. Our desired value should be at least 4 to 5 digit decimals from the left. Thus, we concluded that our measure is a failure for the objective side. Thus, objective and subjective both sides are a failure.

Limitations

We have 4 major setbacks we encounter during development. First, our generated image is not getting close to the target. This is due to the lack of experience on doing large scale projects and lack of sense of how the genetic model works. To mitigate this, we analyze and understand how other genetic modelers encounter the building genetic model specifically on painting. Next, we fail to find the correct parameters to get our desired image. To address this, we try to build a smaller version of the genetic model on painting. Then, at each genetic operation, we should visualize, analyze,and record each candidate's images more thoroughly. Third, large scale code creates complexity, which produces a lot of errors. We experienced a lot of logical errors during development. We have somehow rewrote the whole code 3 times since the code was not structured correctly. To mitigate this, we suggest spending time drawing a UML and try to build a genetic model on a very small scale such as creating a genetic model to recreate a simple geometric shape at earliest. Once we build this, we will have a better sense of what kind issues and challenges we will face.

Gallery

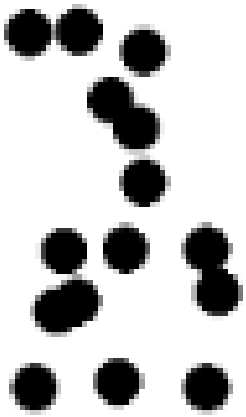
The circle painting genetic art is a stunning example of the intersection between technology and art. Utilizing a genetic algorithm, the goal was to harness the computational power to create abstract compositions that embody both organic and evolving elements.

The core of the algorithm is a process of evolution, where shapes, colors, and other elements are selected and recombined to form new variations. This iterative process aimed to capture the essence of growth and transformation in a visual form.
In this piece, circles serve as the building blocks for a dynamic composition, with each circle symbolizing a unique aspect of the evolution process. The overlapping and blending of circles creates a fluid interplay of color and form, inviting the viewer to explore the world of computational art.

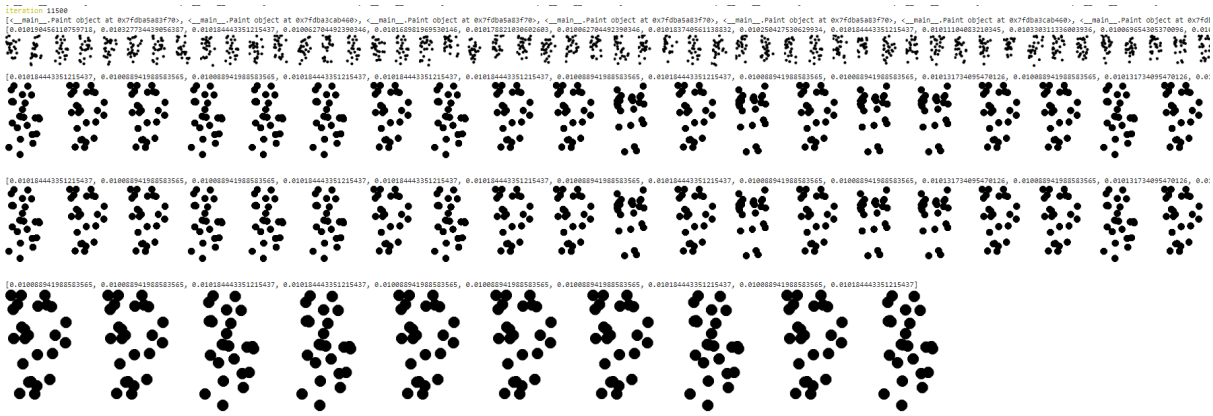
The use of bold and vibrant colors was intentional, meant to evoke a sense of energy and movement. These hues serve as a reminder that, despite its algorithmic origins, this work is a product of human creativity and imagination.

The creation of the circle painting genetic art was not without its challenges. The lack of experience in large-scale projects and an insufficient understanding of the workings of a genetic model resulted in generated images that were far from the target. To overcome these challenges, a thorough analysis of previous genetic models in the painting domain was conducted, and a smaller version of the genetic model was built to better understand the correct parameters. Additionally, the complexity of large-scale code resulted in numerous logical errors, but through rewriting the code and structuring it more effectively, these issues were overcome.

Overall, the circle painting genetic art is a testament to the beauty and diversity that can be achieved through the use of algorithms in the art world. It serves as a powerful reminder of the potential of technology to inspire and challenge our perceptions of the world around us.



Koki's artwork



A single iteration of genetic painting

Next Steps

I will do the following. First, I will test each function more thoroughly to reduce error. Second, I will experiment with each parameter at each phase (crossovers, mutation) more thoroughly. Specifically, we should visualize, record, analyze the effect of each parameter onto the canvas. The overall step till now, should be focused on getting the genetic model to run correctly. Next, we try different functions. This can include SSIM. And, another will be a circle point distance. We can define fitness function by minimizing the distance between circle positions and the black points that represent the outline of the tower in the target image. Note that, doing this, there is a potential that all the circles will be distributed in the same area. To mitigate this, we add a penalty in the circle ensuring that circles are not distributed too much in 1 area.