Image Reconstruction Using Genetic Algorithm

Introduction

The purpose of this project was to explore image reconstruction techniques using a genetic algorithm. The project aimed to generate images that closely resemble a target image by evolving a population of chromosomes through successive generations. Each chromosome represented an image composed of geometric shapes, and the genetic algorithm was utilized to improve the resemblance of the generated images to the target image.

Modules

Helper Module

The "helper" module provides the necessary classes and functions to support the genetic algorithm operations. It includes the following classes:

- **Point :** Represents a 2D point
- **Color :** Represents a color using “RGB” values
- **Gene :** Represents a gene that defines a geometric shape in the image. It includes properties such as size, diameter, position, and color. The gene is capable of mutating to introduce variations.
- **Chromosome :** Represents a chromosome that contains a collection of genes. It facilitates mutation, crossover, and drawing of the image based on its genes.

Functions Module

The "functions" module includes various utility functions that support the genetic algorithm operations. These functions include:

- **GetImage(path) :** Loads the target image from the specified file path.
- **foo(img1, img2) :** Calculates the difference between two images based on the sum of squared differences of their pixel values.
- **Fitness(img1, img2) :** Evaluates the fitness score of an image chromosome based on its resemblance to the target image.
- **Crossover(parents, number) :** Performs uniform crossover between selected parent chromosomes to generate a specified number of child chromosomes.
- **Mutate(parent, number, target) :** Applies mutation to the parent chromosome to create a specified number of mutated child chromosomes.
- **GetSave(parent, generation, score) :** Saves the generated image and displays the generation and fitness score information.

The fitness function takes two images, img1 and img2, as input. It utilizes the foo() function to calculate the difference between these two images. The foo() function performs a pixel-wise comparison and computes the sum of squared differences between the corresponding pixels of the two images.

Once the difference and size information are obtained from foo(), the fitness function normalizes the difference value by dividing it by 255.0 (maximum pixel value) and then multiplies it by 100. This normalization ensures that the fitness score is within a reasonable range for comparison purposes. Finally, the normalized difference is divided by the size (total number of pixels) of the images to obtain the fitness score.

A lower fitness score indicates a higher resemblance between the generated image and the target image. Therefore, the genetic algorithm aims to minimize this fitness score over successive generations by selectively breeding and mutating the fittest individuals.

It's important to note that the fitness function can be customized based on the specific requirements and characteristics of the image reconstruction problem. Depending on the complexity of the images and desired objectives, alternative metrics or algorithms may be employed to evaluate the fitness of the generated images.

Feel free to modify the fitness function according to our specific needs and experiment with different metrics or algorithms to enhance the performance and quality of the image reconstruction process.

Main Module

The "main" module serves as the entry point of the program. It initializes the genetic algorithm parameters, loads the target image, and performs the evolutionary process. The main operations include:

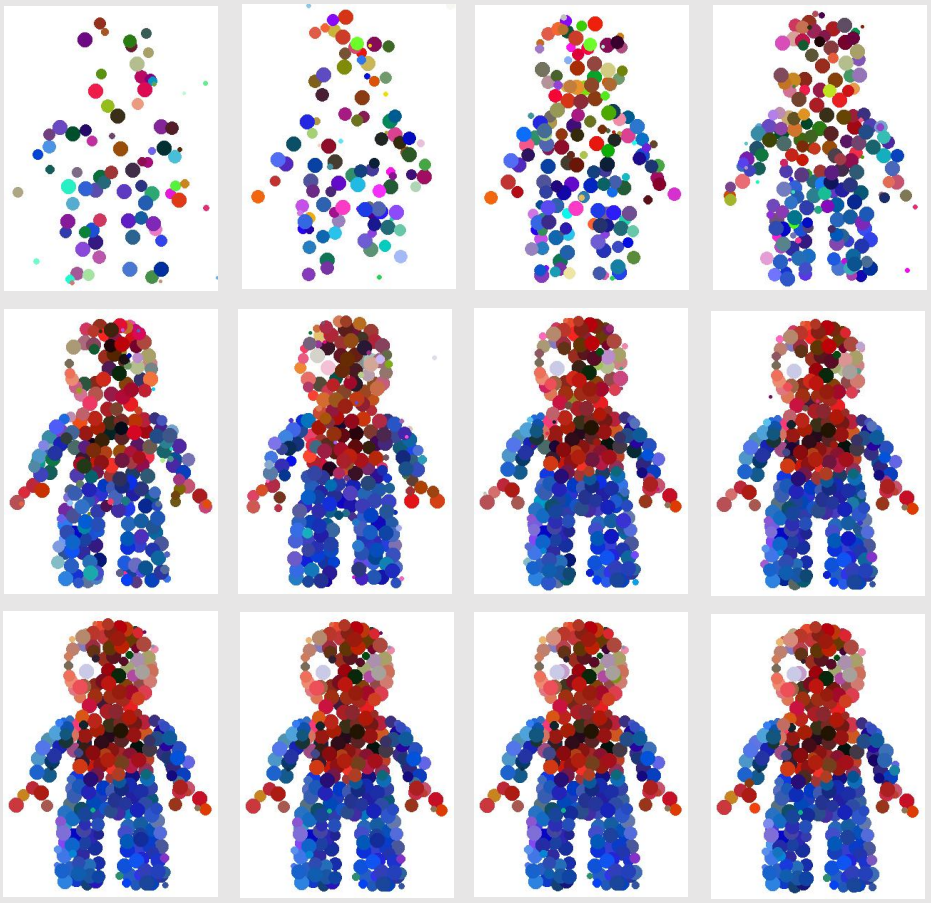
- Initializing the target image and setting up the genetic algorithm parameters.
- Generating an initial parent chromosome.
- Evaluating the fitness score of the parent chromosome.
- Iterating through generations and performing genetic operations, such as mutation and crossover.
- Saving the generated image and displaying the generation and fitness score information at regular intervals.

Results

The project successfully applied the genetic algorithm to reconstruct images using geometric shapes. The generated images gradually evolved and improved their resemblance to the target image over successive generations. By adjusting the genetic algorithm parameters and the number of generations, various results were obtained, demonstrating the effectiveness of the approach.

Original Image :

Generations:



Conclusion

In conclusion, this project demonstrated the application of a genetic algorithm for image reconstruction using geometric shapes. By employing the principles of mutation, crossover, and fitness evaluation, the algorithm iteratively generated images that approached the appearance of a target image. The modular design and implementation allowed for flexibility and extensibility, enabling further experimentation and optimization. The project opens avenues for future research in the field of computational intelligence and image reconstruction.

References

- Eiben, A. E., & Smith, J. E. (2015). Introduction to Evolutionary Computing (2nd Edition). Springer.

Author : Sayed Mahdi Rezaei
Student No. : 40100243
Teacher : Dr.M Safayani
Semester : Spring-402