
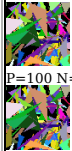

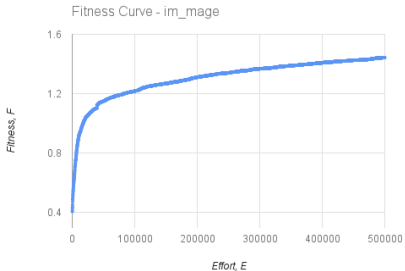
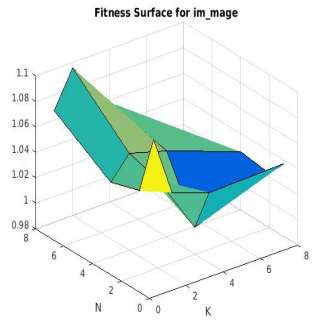

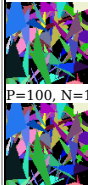
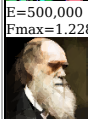
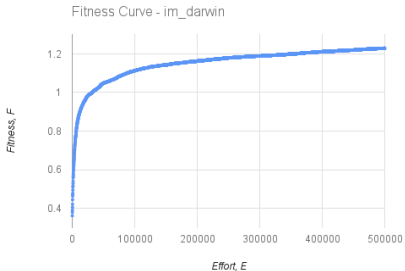
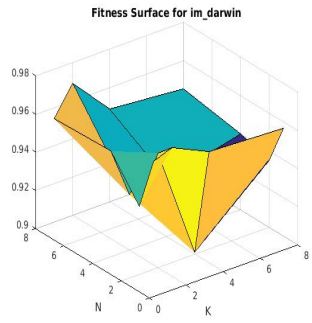

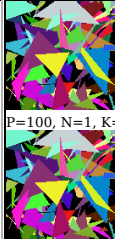
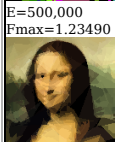
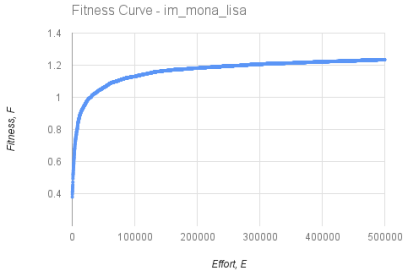
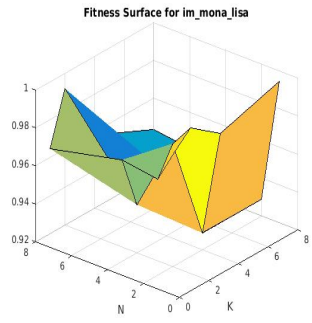
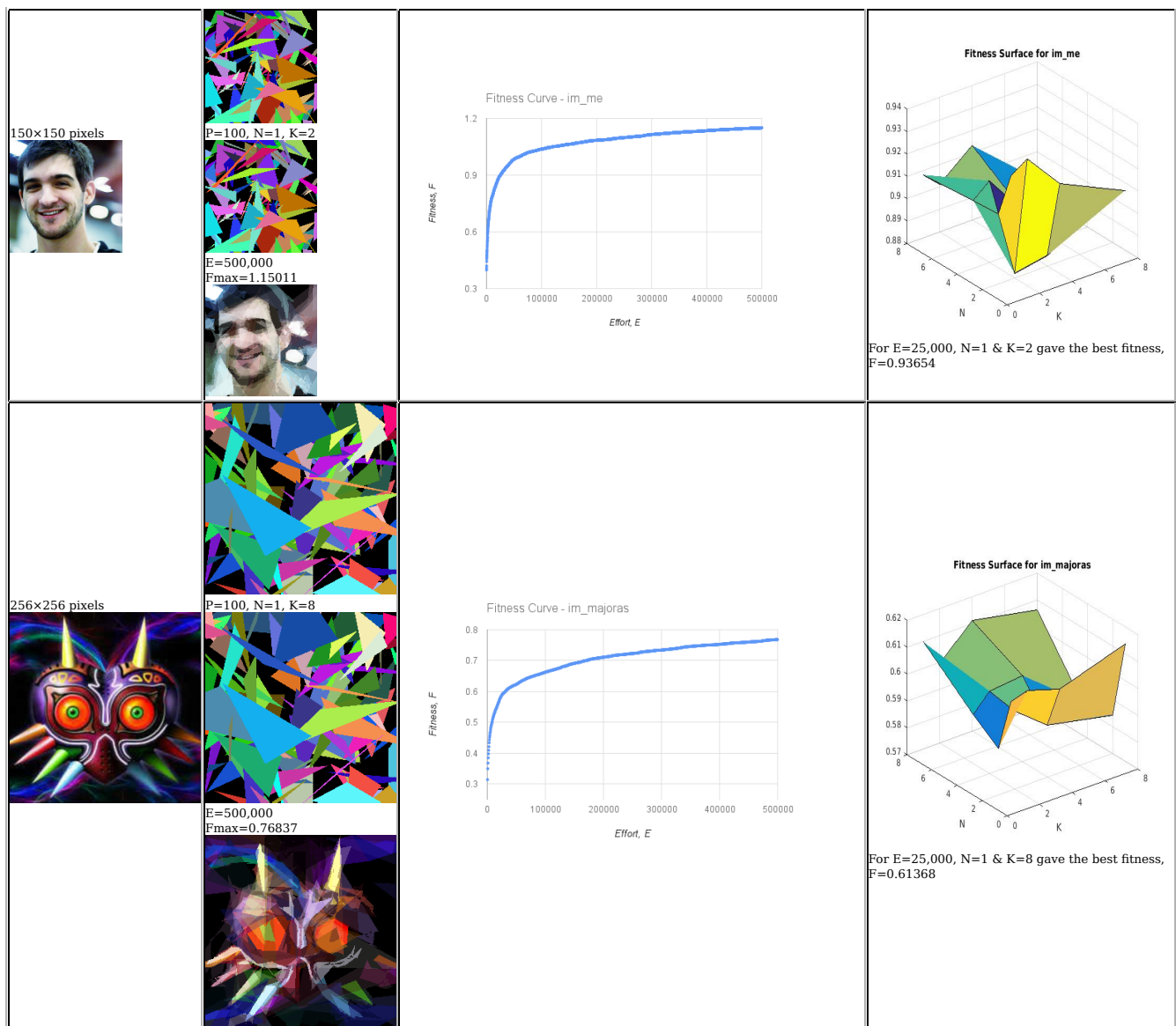


EECS 492 Artificial Intelligence
Assignment 1 Results
Bill Hass
September 29, 2015

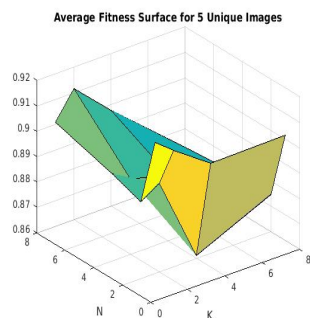
Assignment 1 Results

The purpose of this project was to recreate [Roger Alsing's work on genetic programming to create art](#). Given an image, can we take an initial population of random scatterings of polygons and allow them to reproduce and mutate to approximate the given image? Lets take N as the population size and K as the number of new children created per generation, T. The fitness of the image is a measure of how closely it matches the original image. Each image uses P polygons, where P=100 for the results on this page. Initially, the polygons are generated randomly - then K pairs of parents are allowed to reproduce to create K children. The entire N+K population is then arranged by fitness, and the lowest K fitness in the population are killed off. This process is repeated until a certain effort, $E = N + KT$, is reached or time ends (the program is terminated). The learning curves show how the fitness of the best approximation improves over time. The 3D graphs depict how the fitness is parameterized by N and K after an effort of 25000.

Original Image	Approximation	Learning Curve	3D Parameter Graph
<div>75×75 pixels</div> 	<div>P=100 N=8 K=2</div>  <div>E=500,000 Fmax=1.44365</div> 	<div>Fitness Curve - im_image</div> 	<div>Fitness Surface for im_image</div>  <div>For E=25,000, N=8 & K=2 gave the best fitness, F=1.09601</div>
<div>96×96 pixels</div> 	<div>P=100, N=1, K=2</div>  <div>E=500,000 Fmax=1.22866</div> 	<div>Fitness Curve - im_darwin</div> 	<div>Fitness Surface for im_darwin</div>  <div>For E=25,000, N=1 & K=2 gave the best fitness, F=0.96739</div>
<div>128×128 pixels</div> 	<div>P=100, N=1, K=2</div>  <div>E=500,000 Fmax=1.23490</div> 	<div>Fitness Curve - im_mona_lisa</div> 	<div>Fitness Surface for im_mona_lisa</div>  <div>For E=25,000, N=1 & K=2 gave the best fitness, F=0.99675</div>



Generally, if we take the mean average fitness of all 5 images parameterized by N and K up to 25,000 effort, we can see that the best overall value for (N,K) is (1, 1) with avg fitness 0.9171.



Mutations used:

- 10% chance to mutate *color* and *shape* of up to 10% of all polygons in genome.
- 10% chance to replace a polygon in the genome with a random triangle.
- 20% chance to swap the order of two polygons which are adjacent in the genome.
- 20% chance to mutate an existing polygon's *color* and *shape*.
- 20% chance to mutate an existing polygon's *color*.
- 20% chance to mutate an existing polygon's *shape*.

Shape Mutation:

- With 1/3 probability, remove a vertex.
- With 1/3 probability, add a vertex close to an edge.
 - If the slope of the edge was undefined, the y component was chosen at the midpoint, and the x component was chosen with a Gaussian distribution w/ mean midpoint_x and stdev of 10% of total length of the edge.
 - If the slope of the edge was 0, the x component was chosen at the midpoint, and the y component was chosen with a Gaussian distribution w/ mean midpoint_y and stdev of 10% total length of the edge.
 - Otherwise, the x component was chosen uniformly between x_min and x_max of the edge while the y component was chosen uniformly between y_min and y_max of the edge.
- With 1/3 probability, move a vertex slightly.
 - A Gaussian distribution w/ mean 0 and stdev 5 pixels was added to the x component and y component of the vertex.

Color Mutation:

- Mutate each of the polygon's R,G,B,A values:
 - With 1/2 probability add 'val' chosen with uniform distribution in set {0, 5, 10, 20, 30}
 - With 1/2 probability subtract 'val' chosen with uniform distribution in set {0, 5, 10, 20, 30}

Random triangles were created by picking a point on the image with uniform distribution, then adding a Gaussian distribution of mean 0 and stdev 20% of the image width to the initial point's x coordinate and adding a Gaussian distribution of mean 0 and stdev 20% image height to the initial point's y coordinate to obtain the second and third vertices. Random colors were picked using uniform distribution over the RGB color space.

Parents were selected from an array of individuals sorted in decreasing order of fitness. The first parent was chosen using the index `abs(random.gauss(0, 0.1*N))`. Then, the first parent was removed from possible parent choices, and the second parent was chosen using `abs(random.gauss(0, 0.1*N))`. This made the more fit individuals more likely to be chosen for crossover. Once the individuals were chosen, the number of polygons to take from the first parent was chosen uniformly using `random.randint(0, P)`.