Vectorization Using Stochastic Local Search

Byron Knoll
CPSC303, University of British Columbia
March 29, 2009

Abstract:

Stochastic local search can be used for the process of vectorization. In this project, local search is used to optimize the parameters of a set of smooth shapes to represent an image of the Mona Lisa. The shapes are made using quadratic piecewise parametric interpolation. The shapes are partially transparent and are defined by three vertices and a color. The method is successfully applied to a different number of shapes to achieve varying levels of accuracy. More shapes resulted in a greater level of accuracy, but required more time to compute and more memory to store. One of the main advantages of performing vectorization is that it reduces the amount of information required to store an image.

Introduction:

Vectorization is the process of converting raster graphics into vector graphics. A raster graphics image stores a rectangular grid of pixels. Each pixel represents a point of color. A vector graphics image stores geometric primitives such as points, lines, and curves. The two formats each have their advantages and disadvantages depending on the type of image that they are representing. Vector graphics tend to work well on images which do not contain a fine level of detail. Since the image is stored in the form of geometric primitives, it can be magnified to an arbitrary level without any loss of clarity. The file size of vector images also tends to be much smaller than compressed raster images because information only needs to be stored for each geometric primitive instead of each pixel. However, if the image requires a fine level of detail, raster graphics can capture the exact representation of the image for a given resolution. The process of converting vector graphics into raster graphics (so that an image can be output on a display) is called rasterization.

Local search is a technique used to solve hard optimization problems. It involves traversing a large search space in order to find a good solution. The solution may not be optimal since for a given problem the search space may be too large to find the global best solution. Stochastic local search involves an element of randomness when taking steps in the search space in an attempt to find better solutions. There are a variety of different local search strategies to determine the size and direction of steps to make.

Stochastic local search can be directly applied to the process of vectorization. For a given raster image, there are an infinite number of ways that vector graphics can approximately match the original image. The original image can be exactly matched using vector graphics by simply representing each pixel as a colored square. However, using this exact algorithm does not result in any reduction in file size compared to the original image, negating one of the main advantages of vector graphics. If we constrain the type and number of geometric primitives, stochastic local search can be used to find an approximate solution by traversing the search space of possible vector representations.

In this project the search space is constrained to a single type of geometric primitive. The primitive is a smooth shape which uses quadratic piecewise parametric interpolation. The shape is defined by a smooth curve by imposing the first derivative continuity condition. Each shape is constrained to have three knots. Each shape also has a color parameter and a transparency parameter. Since the shapes are uniquely defined by three points, a color, and a transparency level, stochastic local search can be used to modify these parameters to approximate a raster graphics image. This paper will describe in detail how the shapes are defined and analyze the effectiveness of the stochastic local search algorithm used.

Background and Related Work:

This project was inspired by work done by Roger Alsing*. He uses a genetic algorithm to represent an image of the Mona Lisa using semi-transparent polygons. In figure 1 we can compare the original raster graphics image to the vector graphics representation after a certain number of generations in the evolutionary algorithm:

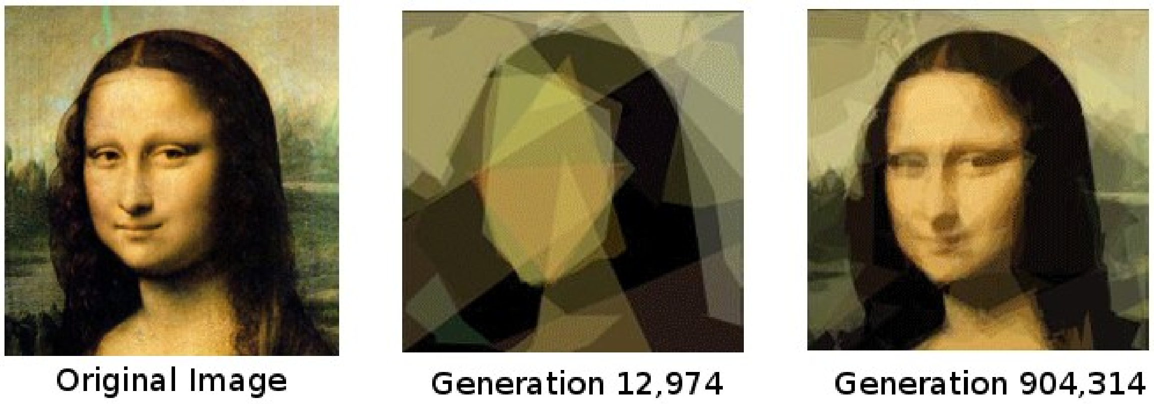

Figure 1

There are a variety of software packages which perform automated vectorization. These software packages achieve a wide range of results when comparing accuracy to computational efficiency. The purpose of this project is not to compare its performance with other approaches, but to analyze the effectiveness of using stochastic local search on the constrained search space. The search will be applied to the raster image of the Mona Lisa seen in figure 1.

Problem Formulation:

This section will describe the constrained search space that is used for the local search and define how the smooth shape is represented. Each shape is defined by the following information:

- *Three points:* Each point consists of two integers. The x-coordinate of a point is between zero and the width of the original raster image. The y-coordinate is between zero and the height of the original raster image.
- *Color:* A color consists of three floating point numbers in the range between zero and one. They represent the red, blue, and green components of the shape's color.
- *Transparency:* A single floating point number in the range between zero and one. It represents how transparent the shape is.

Quadratic piecewise parametric interpolation is used to define the boundary of the shape. Quadratic interpolation was done using the `java.awt.geom.Path2D.Float.quadTo` method. To enforce the first derivative continuity conditions, the midpoints between the three points were used as knots, and the

* see http://rogeralsing.com/2008/12/07/genetic-programming-evolution-of-mona-lisa/

original three points were used as quadratic control points. In figure 2 we see how using the three original points as quadratic control points imposes the first derivative continuity condition:

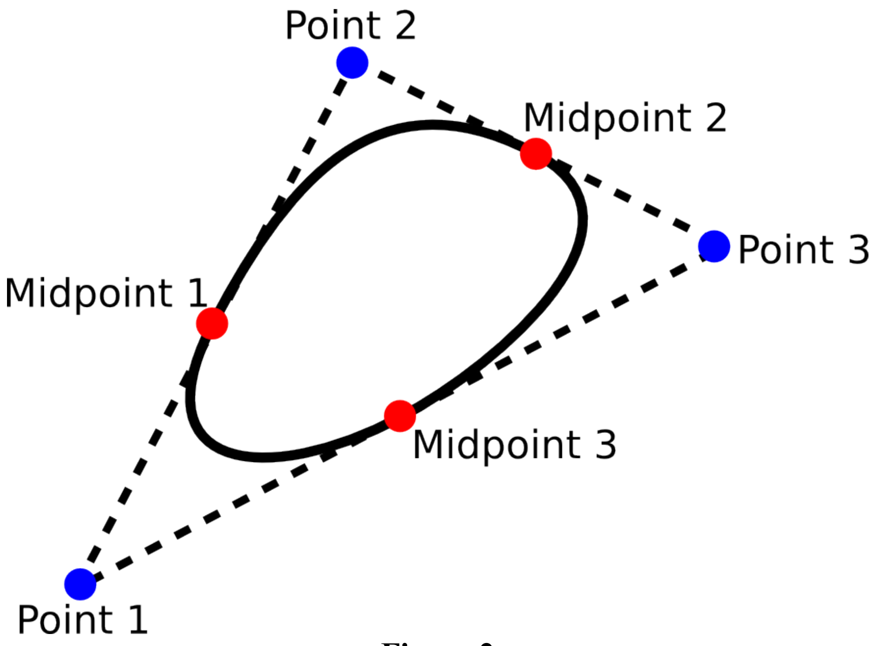

Figure 2

The curve takes the form of a degree two Bézier spline:

$$S(x) := \left\{ \begin{array}{c} S_0(x) := \sum_{v=0}^{2} B_{v,0} b_{v,2}(x) \, where \, x \in [x_{0,} x_1) \\ S_1(x) := \sum_{v=0}^{2} B_{v,1} b_{v,2}(x - x_1) \, where \, x \in [x_{1,} x_2) \\ S_2(x) := \sum_{v=0}^{2} B_{v,2} b_{v,2}(x - x_2) \, where \, x \in [x_{2,} x_0) \end{array} \right\}$$

Given these constraints on the shapes, the local search navigates the search space to optimize the solution. In addition to modifying the shape parameters, the local search also modifies the background color of the canvas. Details on the stochastic local search algorithm used will be given in the next section. The fitness function used to evaluate the quality of a particular set of shapes is done by a pixel by pixel color comparison to the original raster image. The following Java code is the fitness function the local search tries to minimize:

```
    public double fitness (BufferedImage i) {
      double total = 0;
      for (int x = 0; x < img.getWidth(); x++) {
            for (int y = 0; y < img.getHeight(); y++) {
                  Color col1 = new Color(img.getRGB(x,y)); // original image
                  Color col2 = new Color(i.getRGB(x,y)); // vector image
                  total += Math.abs(col1.getRed()-col2.getRed())+
Math.abs(col1.getGreen()-col2.getGreen())+Math.abs(col1.getBlue()-col2.getBlue());
            }
      }
      return total;
    }
```

Solution Methodology:

A single iteration of the designed stochastic local search algorithm performs many steps. A single step is defined to be evaluating the fitness function for a modified vector representation. To ensure that the quality of the solution does not decline after a step, a step is only taken if it results in a decrease in the fitness function. However, this runs the risk of getting stuck in a local minima. To help avoid getting stuck in a local minima, different step sizes can be used (more details below). A larger step size results in solution which is “further away” from the current vector representation. “Further away” means that the step can result in a large change to the parameters of a shape.

A single iteration of the algorithm iterates through all of the shapes in order to modify them individually. By only modifying a single shape instead of all the shapes at once, the neighbor space for a step is reduced. This offers the advantage that in a reduced neighbor space it can be easier to find better solutions. It also has the disadvantage that it becomes easier to get stuck in local minima because all of the neighbors for a given state may result in a higher fitness function. By considering a larger neighbor state (such as modifying the parameters of all the shapes at once), we are less likely to find an improving step, but also less likely to get stuck in local minima. A further reduction of the neighbor space was done by only modifying either the color (and transparency) or the position of the vertices of a shape. These neighboring states are implemented in the “mutateShape” and “mutateColor” functions. The functions also take a “strength” parameter which determines the step size.

The “strength” parameter passed to the “mutateShape” and “mutateColor” functions are essential to taking steps which improve the fitness function. We need to balance making progress on the fitness function and avoiding local minima. If no progress is being made with repeated step evaluations, it indicates we are either considering a neighbor space which is too large, or too small. For each shape and each type of mutation (color or shape) the algorithm stores a value which determines which strength parameter to use. It starts with a large step size, and every time a step is made which does not improve the solution, the parameter is divided by the value 1.3 (causing a reduction in the step size). This means that gradually smaller and smaller steps will be taken to help determine a direction of improvement. As soon as an improving step has been found, this strength parameter gets reset to its initial value. However, doing this when the steps converge to a local minima will result in getting stuck and making no progress at all. To help prevent getting stuck in a local minima, with a 50% chance a random “strength” parameter is used. In summary, we gradually reduce the step size when non-improving steps are made in order to reduce the neighbor space, and use a random step size 50% of the time to help prevent local minima.

Results:

The number of geometric primitives has a large effect on the quality of the solution. With more shapes a more accurate vector representation of the raster image is possible. However, one of the goals of vectorization is to reduce file size by using as few primitives as possible. The resolution of the Mona Lisa picture that the program is tested on is 231 by 242 pixels. That is a total of 55,902 pixels. The stochastic local search for this project will be used on a much smaller scale of information (50 to 250 shapes). Using a smaller number of shapes reduces both the time needed to optimize the picture and the resulting file size. By analyzing the following three executions of the program we can observe that there is a trade off between the quality of the solution and the number of primitives needed to represent it:

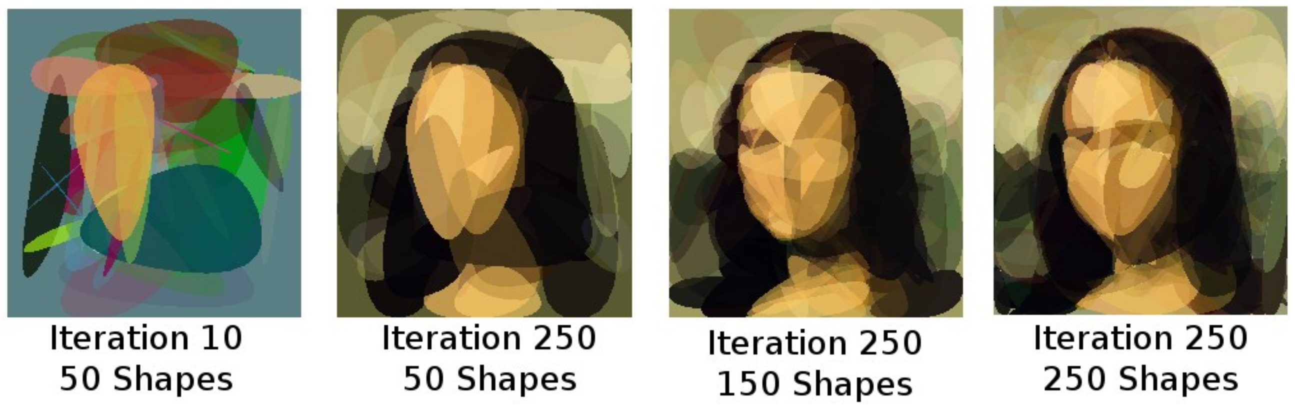

Figure 3

After 250 iterations, the value of the fitness function for 50 shapes was 2,923,728. For 150 shapes it was 2,318,431 and for 250 shapes it was 2,022,309. These results indicate that a larger number of shapes allows the local search to find more accurate vector representations.

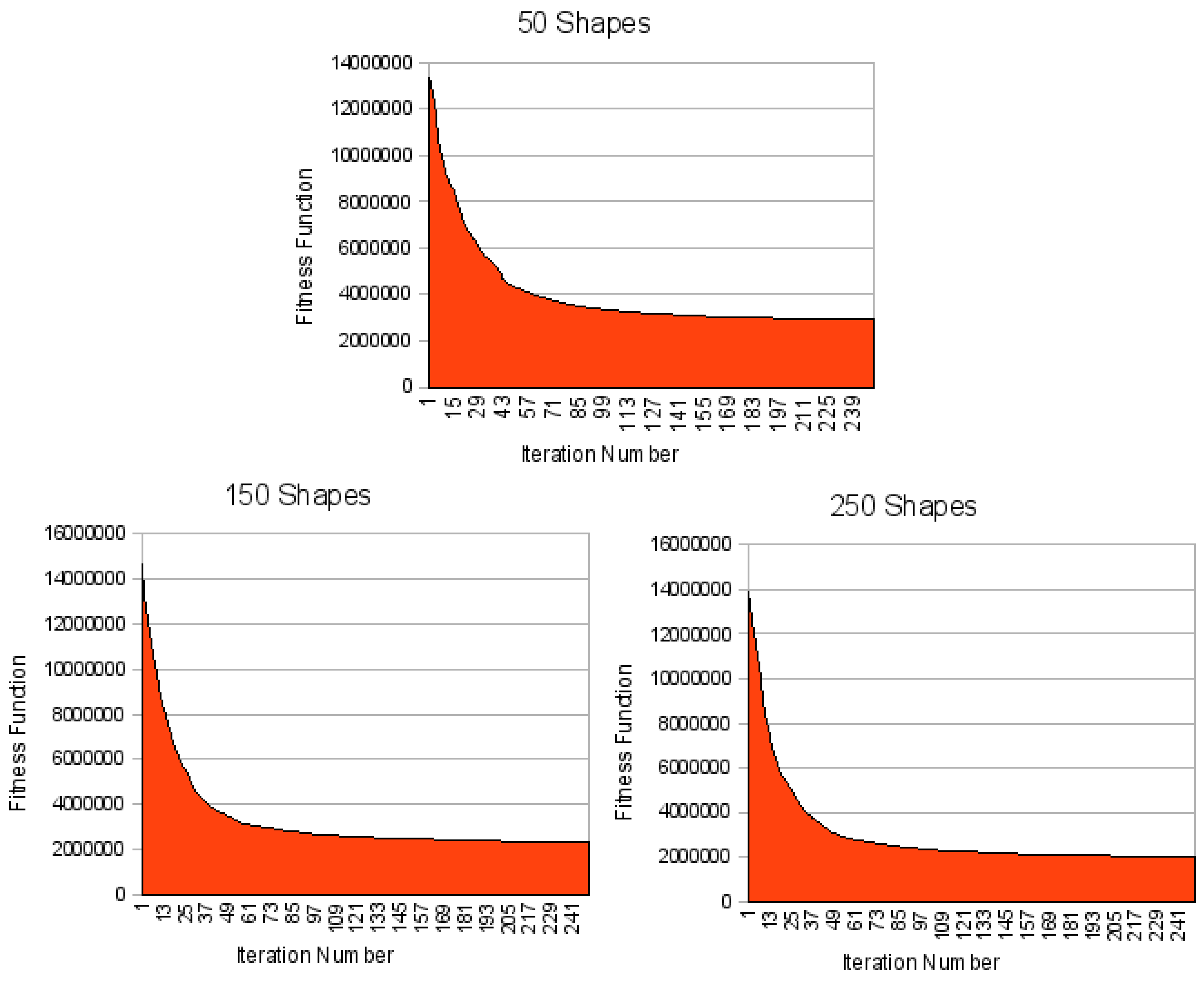

Figure 4

We can see in figure 4 that the shape of the progress curve is similar for different numbers of shapes. The shape of the curve is to be expected since it becomes increasingly difficult to find better solutions at lower values of the fitness function. Given the constraints on the search space (the type/number of geometric primitives) there is always a lower bound to the value of the fitness function. The actual lower bound can not be determined without finding the global optimal solution. For the three executions of the program above, the lower bound is not equal to zero since it would require far more than 250 shapes to match the original image perfectly.

Conclusion and Future Work:

This project successfully applies stochastic local search to perform vectorization using a simple type of geometric primitive. The source code for this project is available at http://byronknoll.googlepages.com/Vectorization.jar. There are a variety of ways to modify this work for future study. To improve the lower bound on the achievable accuracy in the constrained search space, more complex shapes can be considered. For example, the quadratic piecewise parametric interpolation used in this project can easily be extended to more than three knots. The choice of whether to add a knot can be a type of neighbor for a step. It should be noted that the addition of more knots can lead to the problem of self intersection. When the boundary of a shape intersects itself, it can become unclear what defines the interior and exterior of a shape. This problem can be overcome by simply rearranging the order of the knots. For a given set of vertices it is always possible to create a non-self intersecting polygon. One simple way to do this is to choose a single vertex and rearrange the order of the remaining vertices in increasing order of their relative angle to that point. This defines a path between vertices which does not contain self-intersection. Another way that more complex shapes can be created is to change the type of parametric interpolation. For example, polygons can be defined by using straight line segments between the knots.

In addition to modifying the constraints on the search space by introducing more complex geometric primitives, the actual stochastic local search algorithm can be improved as well. Since there are a large variety of different techniques for local search, any of these methods could be applied to vectorization and compared to see their relative efficiency. This analysis would be easy to perform because the progress curve for a given local search technique provides a clear indication of that technique's effectiveness.

Although local search can take much longer to find good solutions than other software packages currently available, it still has the potential to be useful. For example, a more efficient algorithm could be used to find an initial “good guess” at a solution, and then stochastic local search could be applied to improve the result of that algorithm. Local search can combine well with other algorithms because it can always be used to improve a given solution. This is a good potential area for future work because combining existing vectorization algorithms with local search has the potential to improve upon currently available software packages.