Alex Lalejini (aml386)
Justin Brown (jwb126)

CSE 4413 Final Project Report

Proposal

This final project's objective is to create a basic scientific data visualization tool. The primary focus of the tool will be the implementation of data interpolation from a sparse set of points. For the purposes of this project that data will be limited to functions of x, y (e.g. surfaces). Tool will be able to render the data as either a point cloud or a triangle mesh surface.

What was learned

After researching the objectives of this project, it was easily seen how incredibly complicated and impressive a general purpose 3D plotting tool can be. In researching ways to generate a surface given a sparse set of data, two primary options became available to us – generate a triangle mesh directly from a given data cloud or interpolate the data into a grid as a function of x, y and use the structure of the grid to generate a triangle mesh. For our purposes, a triangle mesh simply implies a listing of vertex indices that when rendered using indexed geometry techniques creates a continuous surface. When looking into constructing a surface from a sparse data set, directly generating the trinagle mesh from a technique like marching cubes or one similarly described by Tishchenko in [1], seemed to be out of the scope of this project. As a result of this, we decided to interpolate the data onto a grid and use the grid structure to generate a surface. This decision led us to research various interpolation methods. Of the methods researched three were chosen to be implemented: Shepard's method of interpolation, radial basis function interpolation, and neural network interpolation. Details of these methods will be discussed in later sections. In addition to learning about the previously mentioned techniques, we also learned about doing camera manipulations about scene in OpenGL.

What was attempted

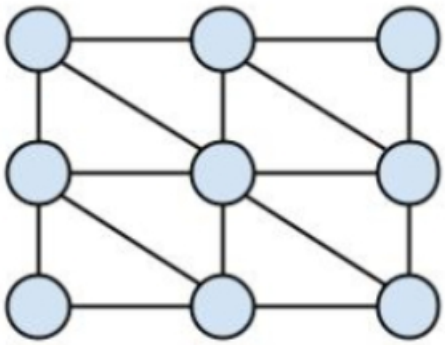
Two methods of rendering were attempted and successfully completed: point cloud and triangle mesh surface rendering. Camera manipulations for viewing the scene were attempted and successfully completed. Three methods of interpolation were successfully implemented: Shepard's method of interpolation, radial basis function interpolation, and neural network interpolation.

How it was attempted

Three tools were implemented for this final project. A data extraction tool was written to extract data from image pixel data and correctly format that data for use with the implemented interpolation and rendering functions. Image pixel data was arbitrarily used to generate scientific data points. This allowed us to collect sample points that could be used for interpolation. The interpolation results using sample points can then be compared against the original image as a performance measure. An interpolation tool was written in order to interpolate sparse data sets, and a plotting tool was written to plot surfaces and point clouds of a specified format.

- Rendering Methods

- Point Cloud
  - Each point in the given data file is expanded into a 1x1x1 cube centered on the data point. 1x1x1 was chosen because the image pixels are 1 unit each apart. If this tool were to be made more generic, the cube size would have to be variable based on the data; or we would just render the points instead of expanding them into cubes. Each cube is made up of 12 triangles; geometry indexing is used to specify the triangles.
- Surface Rendering using Triangle Mesh
  - The data given to this function must be in the form of a 2D numpy array of the format [x][y][z, r, g, b]. This format is relatively rigid considering it requires the data points to each be 1 unit apart in the x and y direction. This restriction is acceptable considering the data we are using to run our tools on – images. This format also requires that the surface to be generated be a well behaved function of x and y. The mesh is generated by taking advantage of the grid structure shown below.



- Interpolation Methods
  - Radial Basis Functions Interpolation (RBF)
    - RBF interpolation interpolates scattered data in n dimensions by approximating a radial basis function to the data. Our implementation of RBF interpolation uses the built in scipy functions available for RBF interpolation. This was chosen to provide an easy to implement baseline performance measure for any other interpolation methods we decided to implement. Scipy implementations of various scientific algorithms are generally accepted as standard implementations.
  - Neural Network Interpolation.
    - Our neural network interpolation function uses a simple feed forward network consisting of an input layer, an output layer, and one hidden layer. The input layer requires two inputs – x and y. The output layer produces 4 outputs – z, r, g, and b. The hidden layer is made up of 5 hidden neurons. The number of hidden layers and number of hidden neurons in those layers is easily variable. The justification for using the network description we used is described by Hu, Hofman, and Hann in [2]. The network is trained using a simple backpropagation algorithm, and is trained for a specified number of epochs. One epoch is one iteration of training on every example in the training set. Our neural network is trained on the sparse data set where z, r, g, and b are given as a function of x and y. The network was entirely implemented by our

group; so, given the time frame to complete the project, the painfully slow runtime of the training, and the fact that we weren't focusing entirely on neural network interpolation, this is the least tested interpolation method used.

- Shepard's Method of Interpolation
  - Shepard's method is also known as inverse distance weighting. Each point being interpolated is the result of a weighted sum of all of the known data point around it. The weight is determined by the inverse distance away from the point being interpolated. We implemented Shepard's interpolation fairly easily relying heavily on the flexibility and functionality available for numpy arrays, which made calculating entire arrays of weights possible in a single line. Our implementation of Shepard's method produces a grid surface as a function of x and y values.

In general for the purposes of this project and ensuring that it's scope was limited enough to complete in the allotted time, we restricted interpolation grid sizes to be 200 by 200. This ensured reasonable testing times and allowed the results of each interpolation function to be comparable.

Results

In this section we will present a comparison of each rendering technique on multiple images. Each original image is 200 pixels by 200 pixels, or 40,000 total data points. Each sample size for the sparse data sets is 7000 randomly taken pixel data from the original image. 7000 was arbitrarily chosen because of potential memory constraints while interpolating using a known set of 7000.

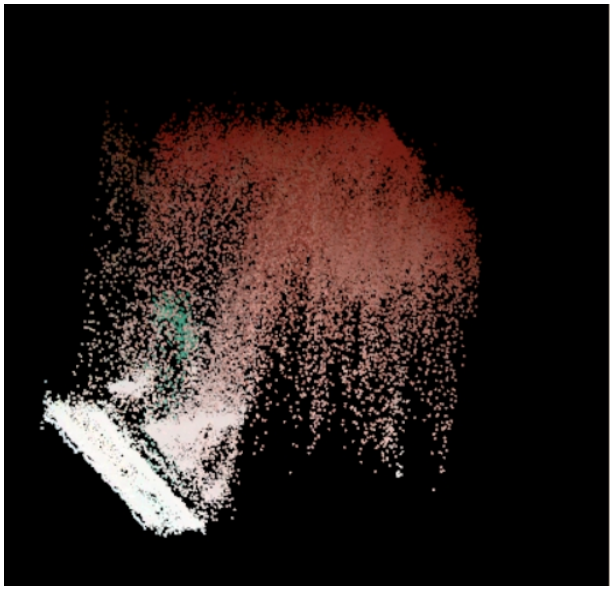
Original Images

Will be refered to as G500 for the rest of this document.

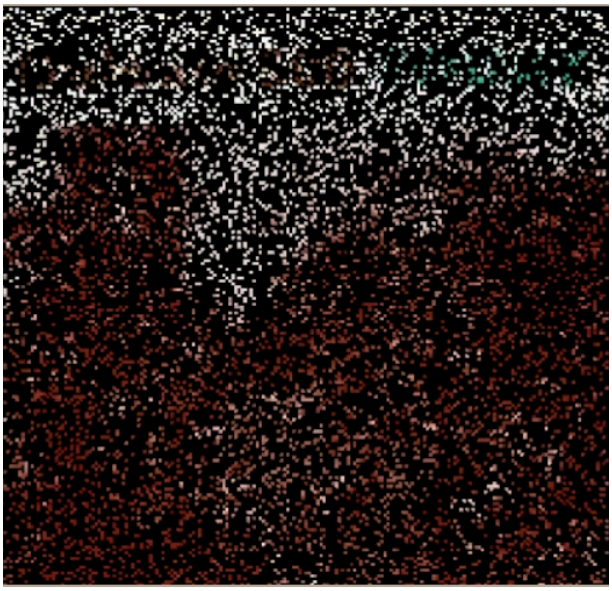
Will be refered to as Terrain for the rest of this document.

Point Cloud Rendering (G500)

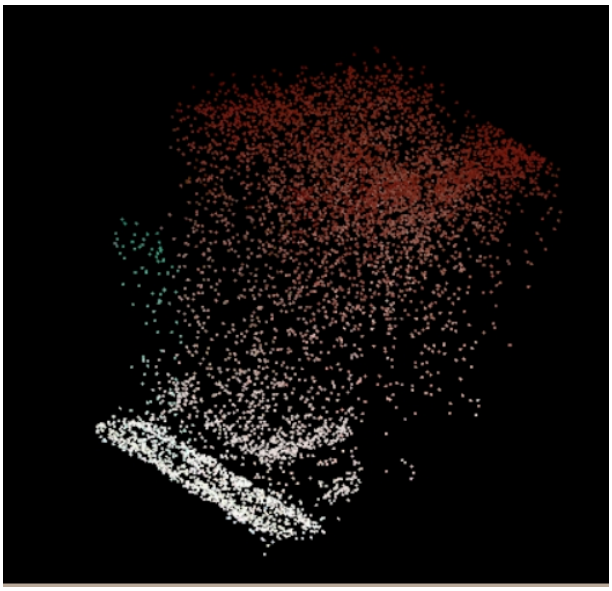
G500.jpg full 40,000 point cloud with camera looking directly down on the cloud.



G500.jpg full 40,000 point cloud with camera repositioned.

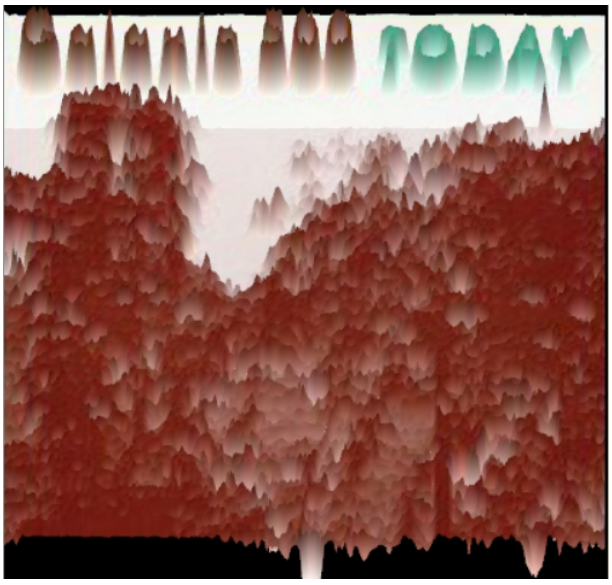


G500.jpg sparse 7,000 point cloud sample with camera looking directly down on the cloud.

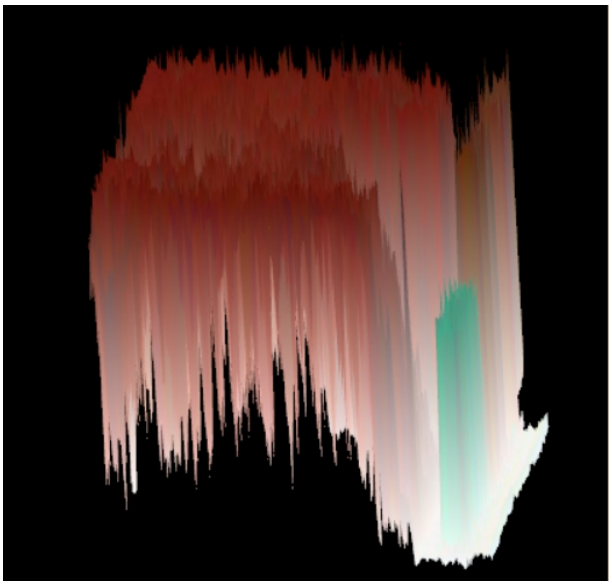


G500.jpg sparse 7,000 point cloud sample with camera repositioned.

Surface Rendering (G500)

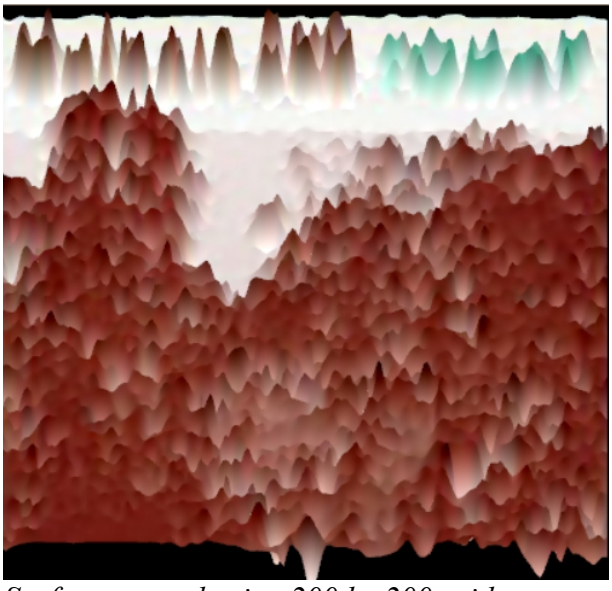


Surface created using 200 by 200 grid generated from original image with camera rotated.

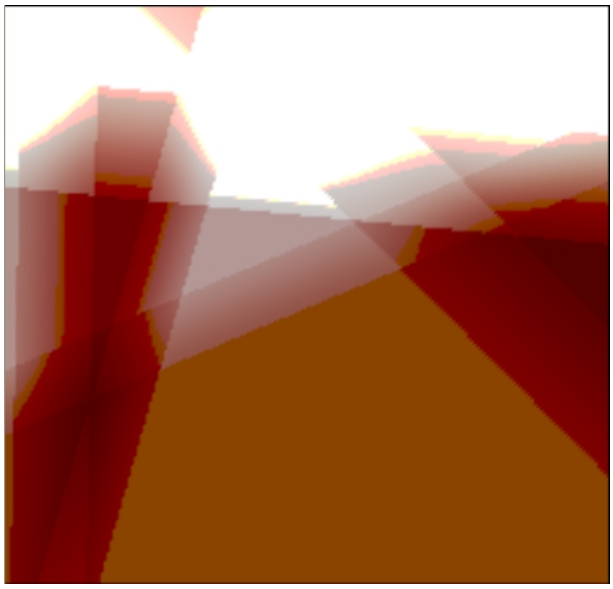


Surface created using 200 by 200 grid generated from original image with camera rotated.

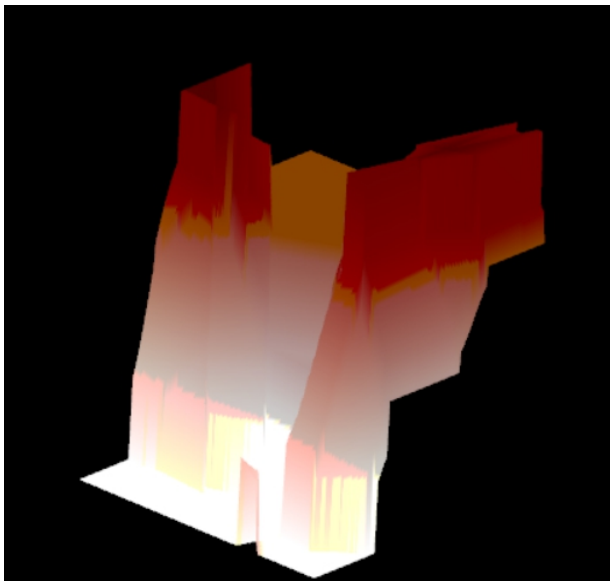
*Surface created using 200 by 200 grid generated by interpolating a sparse set of 7,000 points using **RBF interpolation.** Camera positioned directly above surface.*



*Surface created using 200 by 200 grid generated by interpolating a sparse set of 7,000 points using **RBF interpolation.** Camera rotated.*

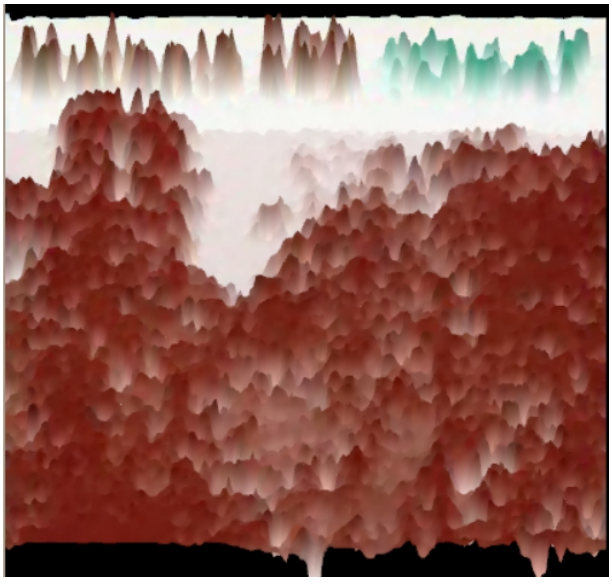


*Surface created using 200 by 200 grid generated by interpolating a sparse set of 7,000 points using **neural network interpolation.** Camera positioned directly above surface.*



*Surface created using 200 by 200 grid generated by interpolating a sparse set of 7,000 points using **neural network interpolation.** Camera rotated.*

*Surface created using 200 by 200 grid generated by interpolating a sparse set of 7,000 points using **Shepard's method.** Camera positioned directly above surface.*



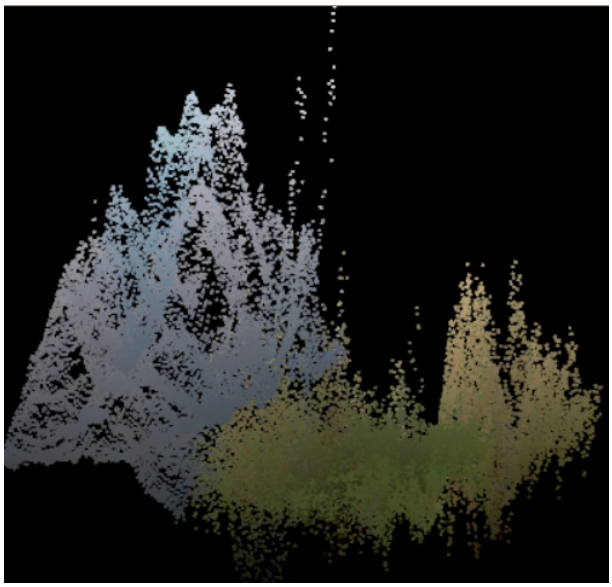
*Surface created using 200 by 200 grid generated by interpolating a sparse set of 7,000 points using **Shepard's method.** Camera rotated.*

From the above results, one can easily visualize the performance of each interpolation algorithm. Shepard's method and RBF interpolation both performed respectably while the neural network interpolation performed poorly relative to the other methods. It is important to note that the Shepard's method result shown above was interpolated with a power of 5. Performance will vary depending on what power

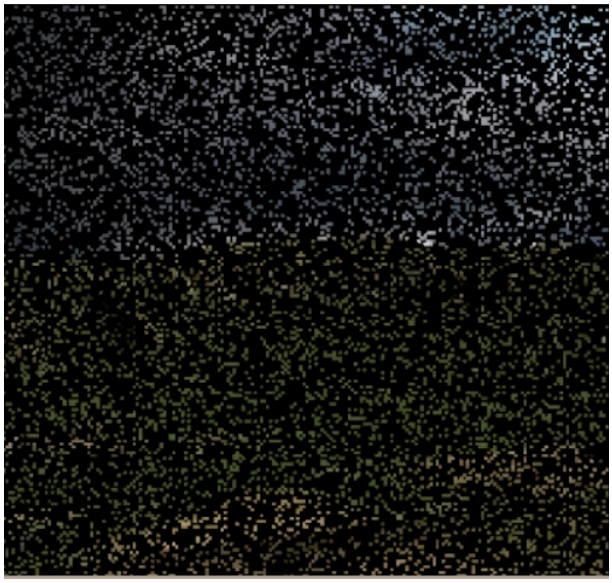
is chosen. Increasing the power makes each interpolated pixel more heavily weighted by the known points closest to it and less weighted by known points further away. Five was arbitrarily chosen – each sparse data set should be interpolated multiple times with different powers to find the best power for that particular set. Sparser sets should select relatively low powers, and less sparse sets should run with higher powers. A higher power could probably have been selected for the above interpolation to improve its performance. The neural network interpolation shown above was trained for 1,000 epochs with a learning rate of 0.1. The above example may not be interpolated using the optimal variable assignments. More training and a higher learning rate could improve the interpolation. It is also possible that there exists a better network description to approximate the function described by the G500 image. A 5 neuron hidden layer was chosen as a result of research; other network sizes were not tested. We did very little testing with the neural network interpolation functionality because of it's incredibly slow run-time. Several optimizations could be implemented to improve run-time; however, such optimizations are out of the scope of this project. We would like to see how the neural network performs at varying learning rates and with more epochs of training. Despite it's poor results relative to other methods, the neural network did learn very high level features as seen in the images above.

Point Cloud Rendering (Terrain)

terrain.jpg full 40,000 point cloud with camera looking directly down on the cloud.



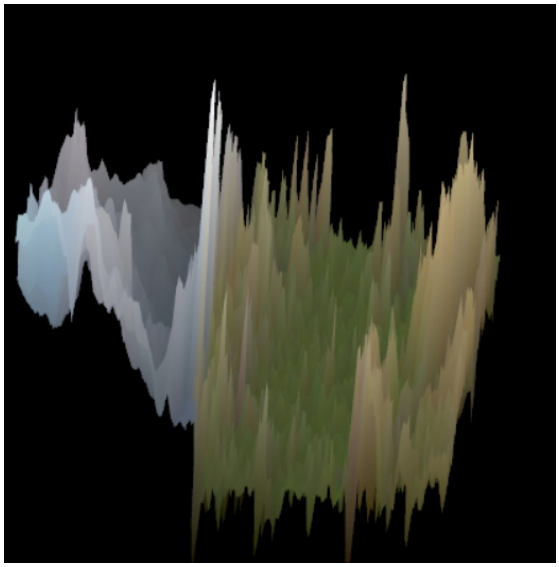
terrain.jpg full 40,000 point cloud with camera repositioned.



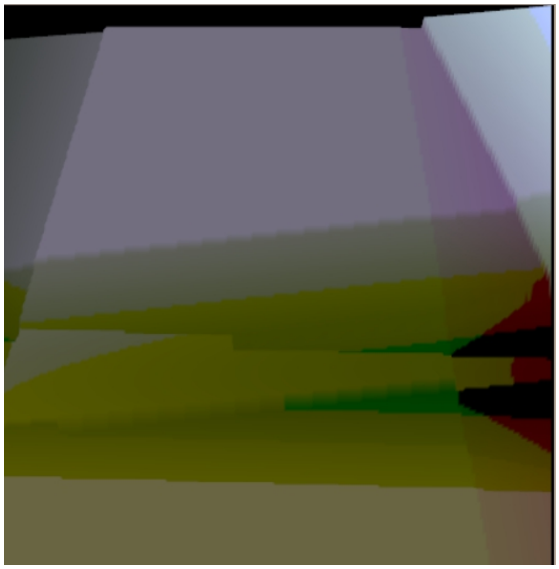
terrain.jpg sparse 7,000 point cloud sample with camera looking directly down on the cloud.

Surface Rendering (Terrain)

Surface created using 200 by 200 grid generated from original image with camera rotated.

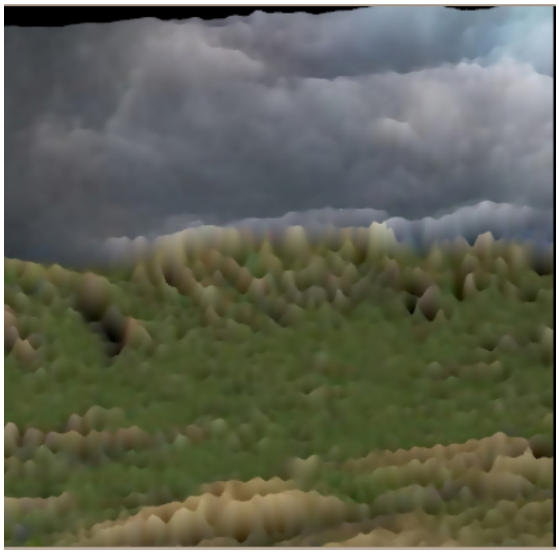


Surface created using 200 by 200 grid generated from original image with camera rotated.

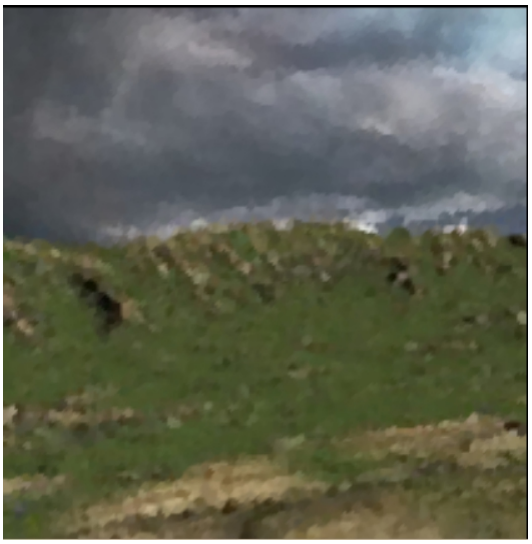


Surface created using 200 by 200 grid generated by interpolating a sparse set of 7,000 points using ***neural network interpolation.*** *Camera positioned almost directly above surface. 2,000 training epochs with a learning rate of 0.25.*

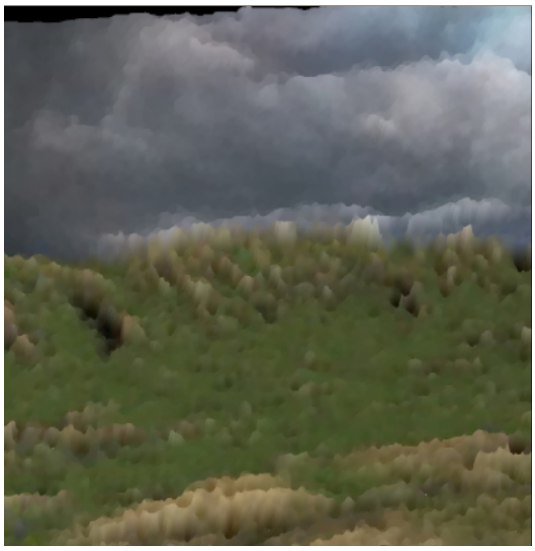
Surface created using 200 by 200 grid generated by interpolating a sparse set of 7,000 points using ***RBF interpolation.*** *Camera positioned directly above surface.*



Surface created using 200 by 200 grid generated by interpolating a sparse set of 7,000 points using ***RBF interpolation.*** *Camera rotated.*



Surface created using 200 by 200 grid generated by interpolating a sparse set of 7,000 points using ***Shepard's method (power 7).*** *Camera positioned directly above surface.*



Surface created using 200 by 200 grid generated by interpolating a sparse set of 7,000 points using ***Shepard's method (power 7).*** *Camera rotated.*

Surface created using 200 by 200 grid generated by interpolating a sparse set of 7,000 points using ***Shepard's method (power 3).*** *Camera positioned directly above surface.*

References

[1] Tishchenko, Inna. Surface Reconstruction from Point Clouds (Bachelor-Thesis). Spring 2010. http://students.asl.ethz.ch/upl_pdf/191-report.pdf

[2] Hao Hu, Paul M. Hofman, and Gerard de Haan. Image Interpolation Using Classification-based Neural Networks. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.106.5785&rep=rep1&type=pdf