COS 702

Assignment 4

Daniel Lucas Thompson

March 25, 2016

Introduction:

In this assignment, we will be reconstructing a surface using a 3D point cloud. For the point cloud, we will be using two common test model known as the Stanford bunny and the Stanford dragon. These two point clouds were created by 3D scanning two ceramic figurines. From this scan, a 3D point cloud of x, y, and z coordinates were created corresponding to points on the surface of the figurines. For the problem in this assignment, we will be using two 3D point cloud sets for the bunny and one set for the dragon. The bunny sets contain 8,171 and 1,889 data points. The dragon set contains 22,998 data points. The original ceramic figurine and point clouds can be seen below.

Surface Construction:

Along with the provided 3D point cloud data, the normal vectors for these data points are also provided. These normal vectors will be used to create extra points to be used in the surface reconstruction. The first step in this process is to remove any normal vectors with a zero value and their corresponding points. Next, we use the remaining non zero normal vectors to create two extra points for each original data point. We do this by moving a short step in the normal vector direction and creating a new point. We do the same in the negative normal vector direction. We do this for all N original points that didn't have zero normal vectors. This is accomplished with the following equations where N is the number of original points, δ is the step size, and n is the normal vector.

$$(X_{N+i}, Y_{N+i}, Z_{N+i}) = (X_i + \delta n_i^x, Y_i + \delta n_i^y, Z_i + \delta n_i^z)$$

$$(X_{2N+i}, Y_{2N+i}, Z_{2N+i}) = (X_i - \delta n_i^x, Y_i - \delta n_i^y, Z_i - \delta n_i^z)$$

After creating these data points, there are a total of 3N data points. We can now use these points along with our radial basis function to compute a set of weights that can be used to reconstruct a surface. First we must set the known value for each of these points. We will do that by setting the original surface to zero, the inner points to -1, and the outer points to 1 by the following.

$$(X_i, Y_i, Z_i) = 0,\ i = 1, \ldots, N$$

$$(X_i, Y_i, Z_i) = 1,\ i = N + 1, \ldots, 2N$$

$$(X_i, Y_i, Z_i) = -1,\ i = 2N + 1, \ldots, 3N$$

Using these lists of points and point values, the weights can be calculated using the form Ax=b where A is the distance matrix with the radial basis function applied for all the points, b are the value of the points computed above, and x are the weights that we are solving for. For this assignment, we will be using compact support radial basis functions (CS-RBF). The specific radial basis function (RBF) that will be used are one of Wendland's CS-RBFs.

Compact support radial basis functions are a way of speeding up the computation time of data

reconstruction. When computing the distance matrix of points, only points within a certain radius are computed. This radius is called the support. Many of the points that are far away are not computed, thus they have a zero value. Considering only these neighboring points that fall within the support radius and using them to create a sparse matrix containing many zeros, we are able to make use of sparse matrix computations. The original compact support radial basis function is as follows.

$$CS-RBF=(1-r)_+^6*(35r^2+18r+3)$$

For a Matlab implementation, it is better to write the function in a shifted form. The shifted form is as follows.

$$CS-RBF(Shifted)=r_+^6*(56*spones(r)-88*r+35*r^2)$$

The purpose of the function spones(r) is to make use of the Matlab sparse matrix optimizations.

To reconstruct the surface, we must construct a new set of points to be used in making the surface. These points will be created using a uniform mesh grid. Once these points are created, we can combine them with the previously calculated weights to get the values for these points. Since these data sets can be very large, we will need to partition the calculations in order to conserve memory. Once these new points and values have been calculated, we can draw the surface by only drawing the points of the value zero.

Results:

In trying to reconstruct the surface of this 3D point cloud, the quality of the reconstructed image can be affected by many of the variables. One of the main variables that affect it is the number of mesh grid points used to reconstruct the data. However, the compact support size, radial basis function, and normal vector step size also have an effect. For this assignment, the quality of the surface reconstruction will be determined visually.

To analyze the results, the images will be constructed using several different variations of the parameters. For the value of *ep* the values 100, 150, 200, 250, 300, 350, and 400 will be used. The value of *ep* effects the support size for the CS-RBFs. For the value of *neval* the values 50, 60, 70, 80, 90, 100, 110, and 120 will be used. The *neval* parameter effects the mesh grid size used for reconstruction. The code provided will generate all the images for these sets of parameters, but due to

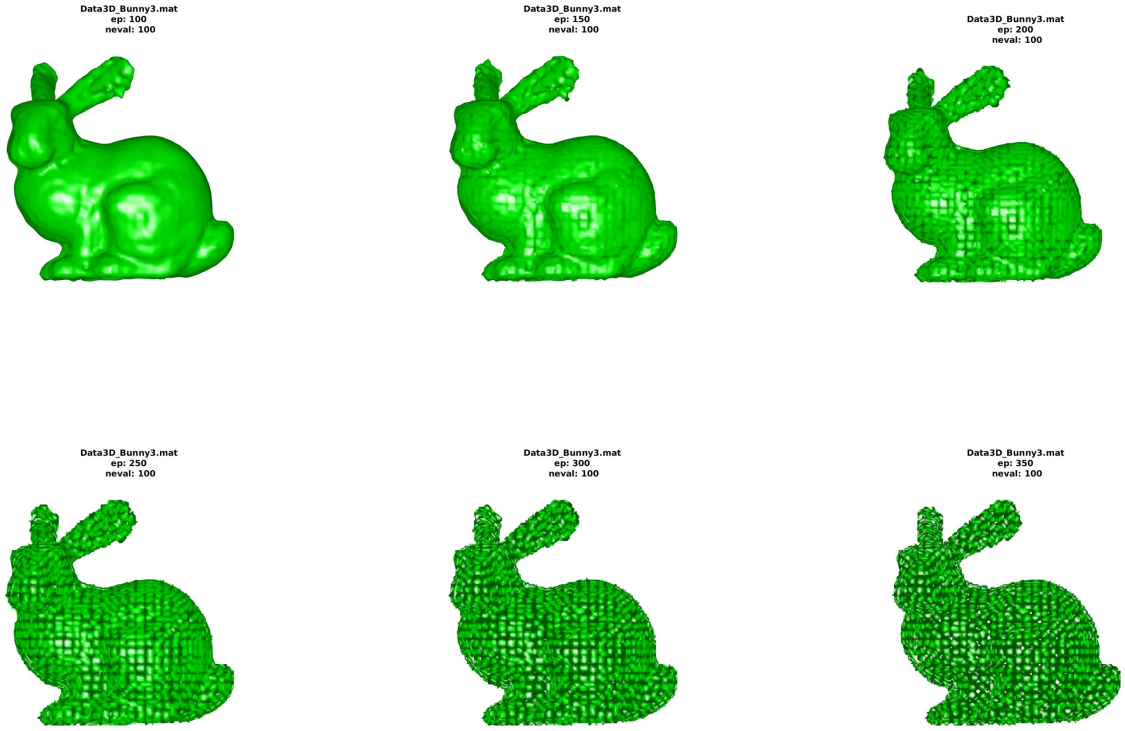
length, only a select few will be displayed in this results section.

In addition to the image surfaces, run times will also be displayed. For each surface reconstruction, the calculations will be timed and displayed in seconds. These will be generated for the same *ep* and *neval* values discussed previously.

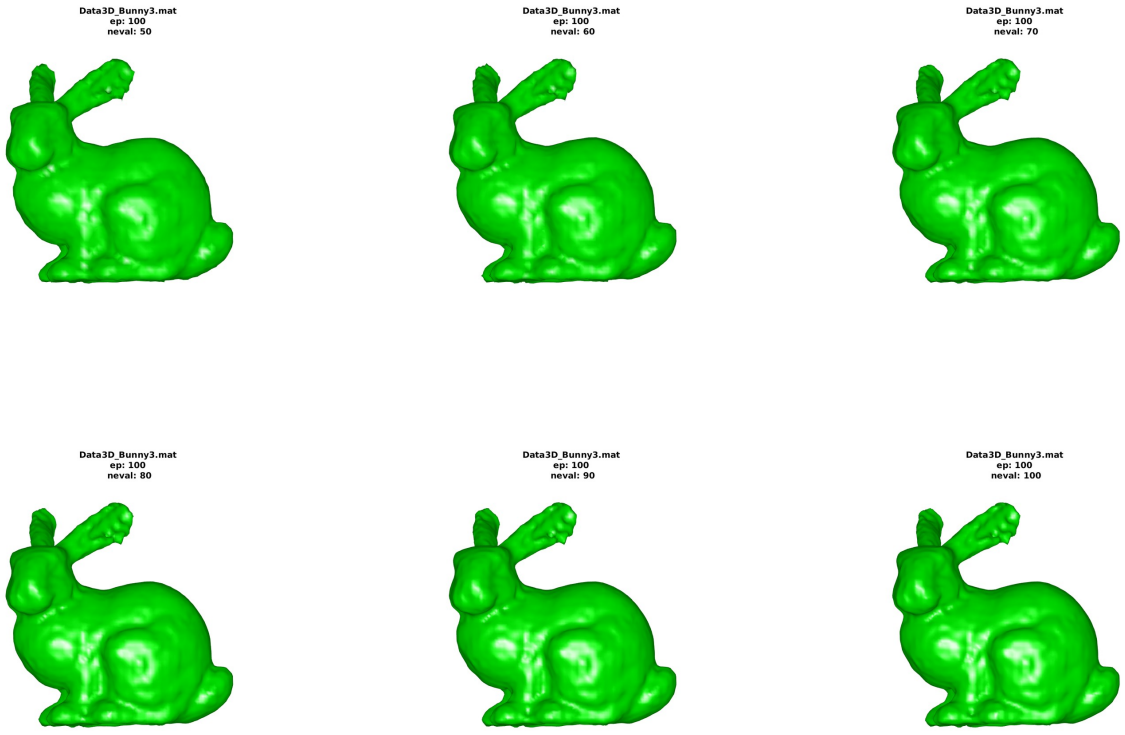
The specifications of the computer used to run these results are as follows. The computer contains a 4 core Intel Xeon processor running at 3.07 GHz. It has 12 GB of RAM. It is running the operating system Kubuntu 14.04.

Bunny with 1,889 Points

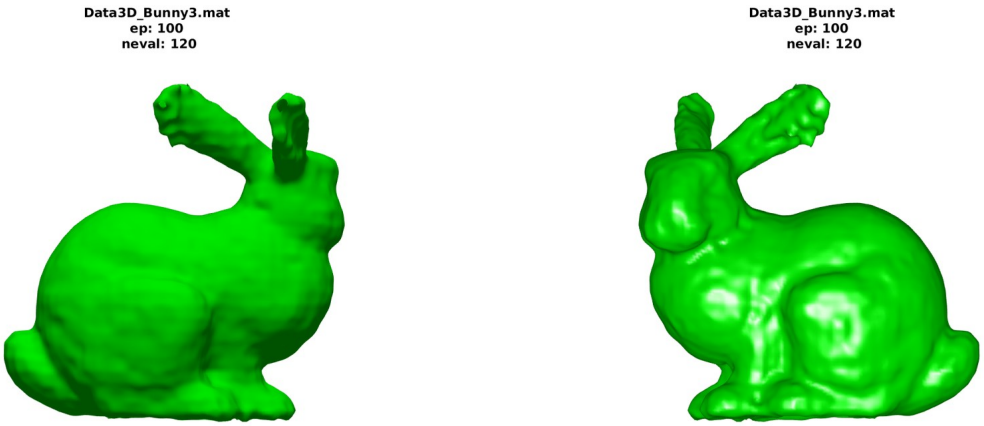
The following results will display images and run times for the Stanford bunny 3D point cloud that contains 1,889 points. In the following image, we show the result of varying the *ep* parameter while keeping the *neval* parameter set at 100. The value of 100 was chosen because it generally gives a good image, and using it will demonstrate the effects of the *ep* value. As can be seen in the following image, as the support size grows smaller, the bunny surface becomes poor. As *ep* increases, the support size decreases which means less points are used in the calculations. Since this figure only has 1,889 points, as the support size decreases, the amount of information available for calculations is reduced resulting in a poorer surface.



In the next image, the effect of the value of *neval* is demonstrated. In the previous image, we can see that the best ep value was 100, so we will demonstrate the effect of *neval* using that parameter. As can be seen in the following image, the surface is recognizable at *neval* of 50, but as it is increased, the surface slowly improves.



In the next image, we have the best bunny generated. It used an *ep* value of 100 and a *neval* value of 120. The image has a small defect on the ear, but recreated the surface in the other areas.

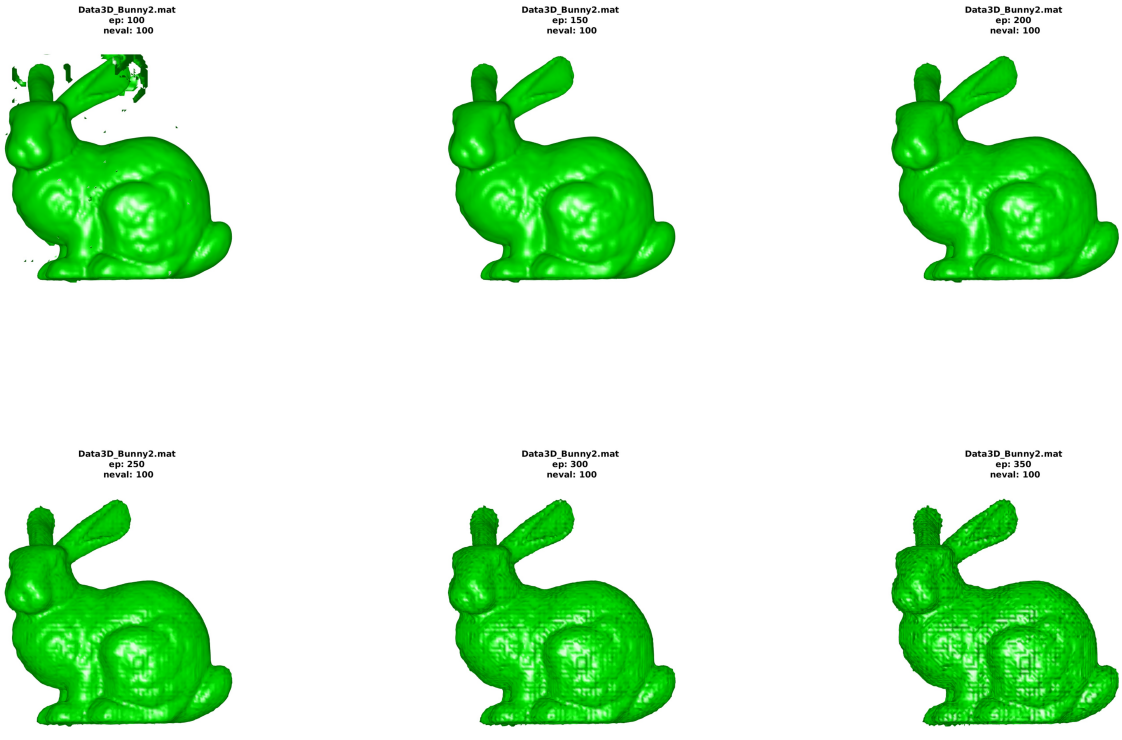


The following table shows the run times for all *ep* and *neval parameters in seconds. The best image run time is underlined.*

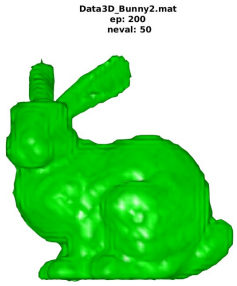
	neval							
ep	**50**	**60**	**70**	**80**	**90**	**100**	**110**	**120**
100	5.7	9.0	15.2	20.3	28.5	38.7	51.3	**73.7**
150	4.8	7.9	12.1	17.8	25.3	34.8	46.0	59.5
200	4.6	7.5	11.5	18.9	24.0	36.4	43.5	56.1
250	4.8	7.3	11.3	16.6	23.5	35.5	42.2	60.8
300	4.4	7.3	11.1	16.3	23.0	31.4	46.5	53.9
350	4.3	7.1	11.0	16.3	25.3	32.9	44.6	53.8
400	4.3	7.8	10.9	17.8	25.0	32.5	42.8	57.6

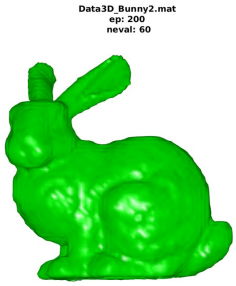
Bunny with 8,171 points

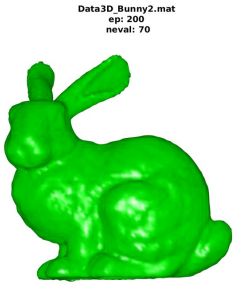
The following results will display images and run times for the Stanford bunny 3D point cloud that contains 8,171 points. In the following image, we show the result of varying the *ep* parameter while keeping the *neval* parameter set at 100 as in the previous results. As can be seen in the following image, the support size starts with *ep* of 100 which has some defects around the ear area. As *ep* increases, the image slowly improves. However, once the *ep* parameter reaches the value of 250, the image quality begins to become poor.

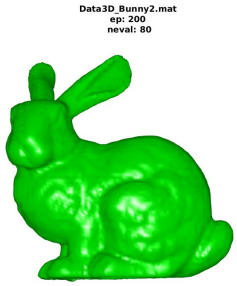


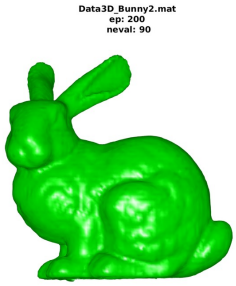
As in the previous results, we now demonstrate the values of *neval*. The best *ep* value seems to be 200, so that value will be used to demonstrate. As can be seen from the following image, at *neval* of 50, the image has the rabbit shape, but it is poorly defined. As *neval* is increased, the image slowly improves until there isn't much difference between the images.

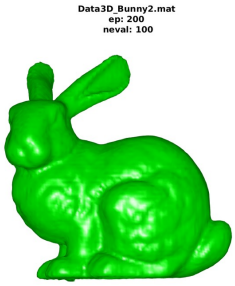




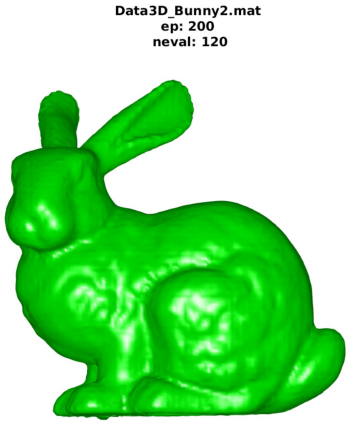


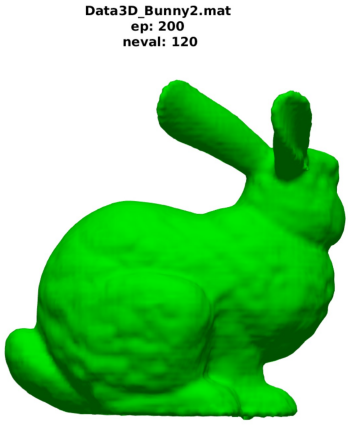






In the next image, we have the best bunny generated. It used an *ep* value of 200 and a *neval* value of 120. The image is a good result, with no defects.



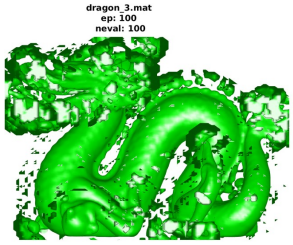


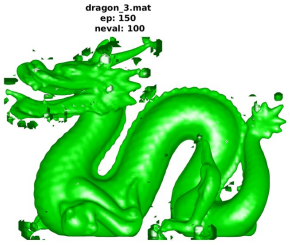
The following table shows the run times for all *ep* and *neval parameters in seconds. The best image run time is underlined.*

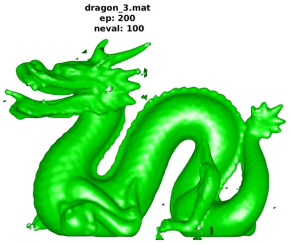
	neval							
ep	**50**	**60**	**70**	**80**	**90**	**100**	**110**	**120**
100	17.3	20.9	29.3	41.8	55.3	74.4	85.7	112.9
150	12.3	16.2	21.4	29.2	38.5	51.3	64.7	82.1
200	10.9	15.0	19.1	26.0	35.2	44.9	56.9	**71.4**
250	11.4	14.0	19.9	24.8	33.4	47.4	54.6	68.9
300	10.3	13.8	17.9	24.5	36.4	42.2	53.5	67.0
350	10.0	13.5	17.5	24.1	32.4	41.8	52.6	65.6
400	9.9	13.3	17.3	23.8	31.8	40.8	57.2	65.0

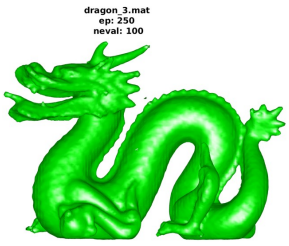
Dragon with 22,998 points

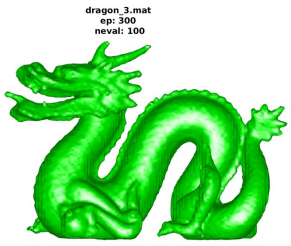
The following results will display images and run times for the Stanford dragon 3D point cloud that contains 22,998 points. In the following image, we show the result of varying the *ep* parameter while keeping the *neval* parameter set at 100 as in the previous results. As can be seen in the following image, the support size starts with *ep* of 100 which has defects around the entire surface. As *ep* grows, the image improves.

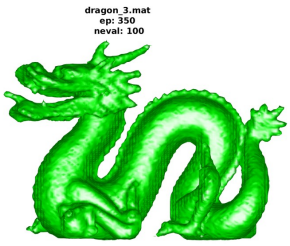




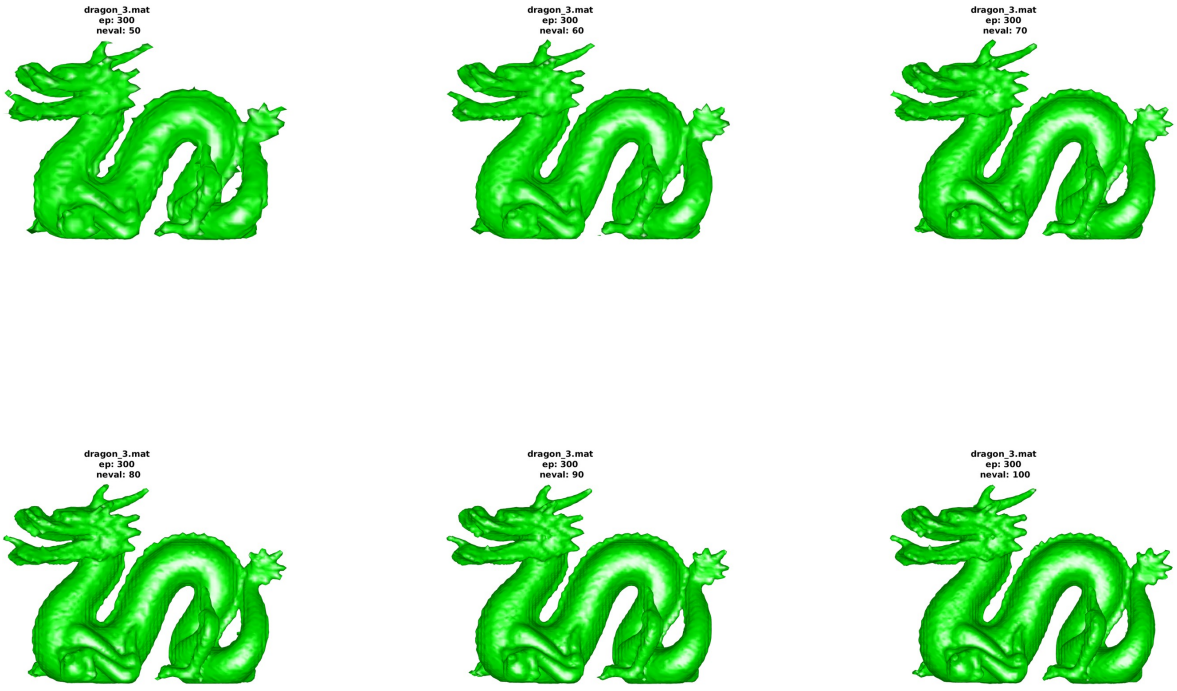




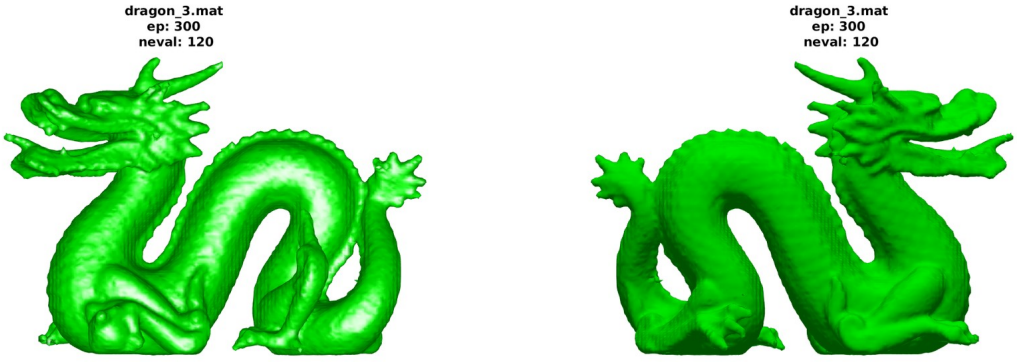




As in the previous results, we now demonstrate the values of *neval*. The best *ep* value seems to be 300, so that value will be used to demonstrate. As can be seen from the following image, at *neval* of 50, the image shows the details poorly. As *neval* is increased, the image slowly improves until there isn't much difference between the images.



In the next image, we have the best dragon generated. It used an *ep* value of 300 and a *neval* value of 120.



The following table shows the run times for all *ep* and *neval parameters in seconds. The best image run time is underlined.*

	neval							
ep	**50**	**60**	**70**	**80**	**90**	**100**	**110**	**120**
100	78.3	91.7	103.7	138.1	160.8	195.4	229.8	267.9
150	61.5	65.1	71.9	89.6	109.8	130.3	152.1	178.7
200	51.0	64.5	63.5	80.2	99.0	113.2	146.0	152.6
250	52.1	55.0	60.9	75.3	92.4	107.0	123.9	158.0
300	50.8	53.7	58.1	73.2	90.3	105.2	120.0	**139.0**
350	45.3	53.0	57.2	75.1	88.9	103.2	119.3	151.0
400	45.0	52.6	58.5	71.7	91.5	120.8	117.4	137.7

Conclusion

As can be seen from the data in the previous sections, it takes much testing to narrow down the correct parameters that will produce the best surface from a 3D point cloud. There are several parameters that can effect it. As we can see from the measured run times, it can also be time consuming to test these parameters. For this assignment, the program was run for several hours to produce images for many combinations of parameters. However, there was a limit to the number of parameters that could be tested. The best images produced in this report are good images, but with more time to adjust and run the parameters, slight improvements could possibly be made. However, it is possible for some of the images, the improvements would be so minor that they would be hard to notice.