COS 702

Assignment 6

Comparison of Estimated Normal Vector Functions and Reconstruction of a 3D Scanned Person Using Estimated Normals

Daniel Lucas Thompson

May 12, 2016

Introduction:

In this assignment, we will be comparing the errors of several functions that estimate normal vectors of each individual vertex in a 3D point cloud. There are three functions that will be compared. Two of the functions, 'findPointNormals' and 'points2normals', come from the MathWorks file exchange website. The third function, 'pcnormals', comes from the Matlab computer vision toolbox.

To compare these functions, we will be using the bunny file that includes 8,171 points with normal vectors. These normal vectors supplied with the bunny file will be compared to the estimated normal vectors from each of the above functions. The function with the lowest error will then be used to reconstruct a 3D scan of a person from a set of vertex points.

Comparison of Data:

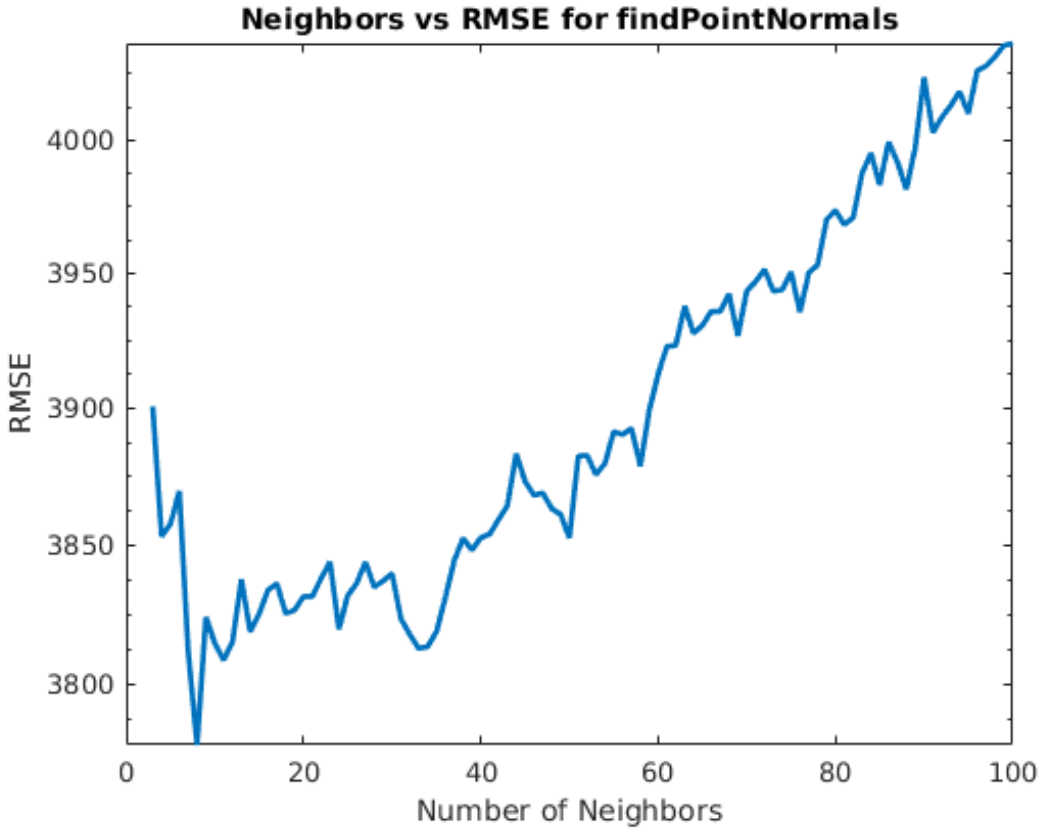
In the following sections, we will be comparing several normal vector estimation functions. These functions will be compared using the correct normal vectors from the original bunny file with 8,171 points and the estimated normal vectors. In order to measure the error between the normal vectors, we will be calculating the number of degrees between the original normal vector and the estimated normal vector. For a perfect estimate, the degrees should be zero. These difference in degrees will then be used to calculate the root mean squared error (RMSE) of the normal vectors.

The findPointNormals Function:

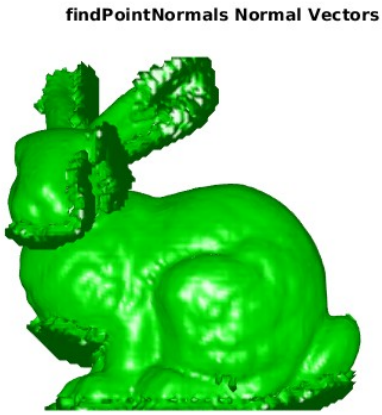
This function comes from the MathWorks file exchange website. It estimates the normal vector for a point by taking a set of neighboring points and using those to approximate a plane through the points. The normal vector for that plane is then used for that vertex. The function takes four values as input. The first value is the points to use to estimate the normal vectors. The second input is the number of neighbors to use in constructing the plane. The third input is described as the direction all normals

will point towards which was an unclear description, so I'm not sure of its purpose. After some testing, the default value seemed to perform well. The fourth input is a flag to use only the largest normal in determining direction with respect to the third parameter. After testing, the default value for this parameter was used also.

To find the best nearest neighbor parameter for this function, I tested and compared the errors between the original normals from the bunny file and the estimated normals from this function using a set of neighbor parameters. The graph of the number of neighbors vs root mean squared error between the estimates and real values follows. As can be seen from the graph, the number of neighbors that produce the smallest error is 8.

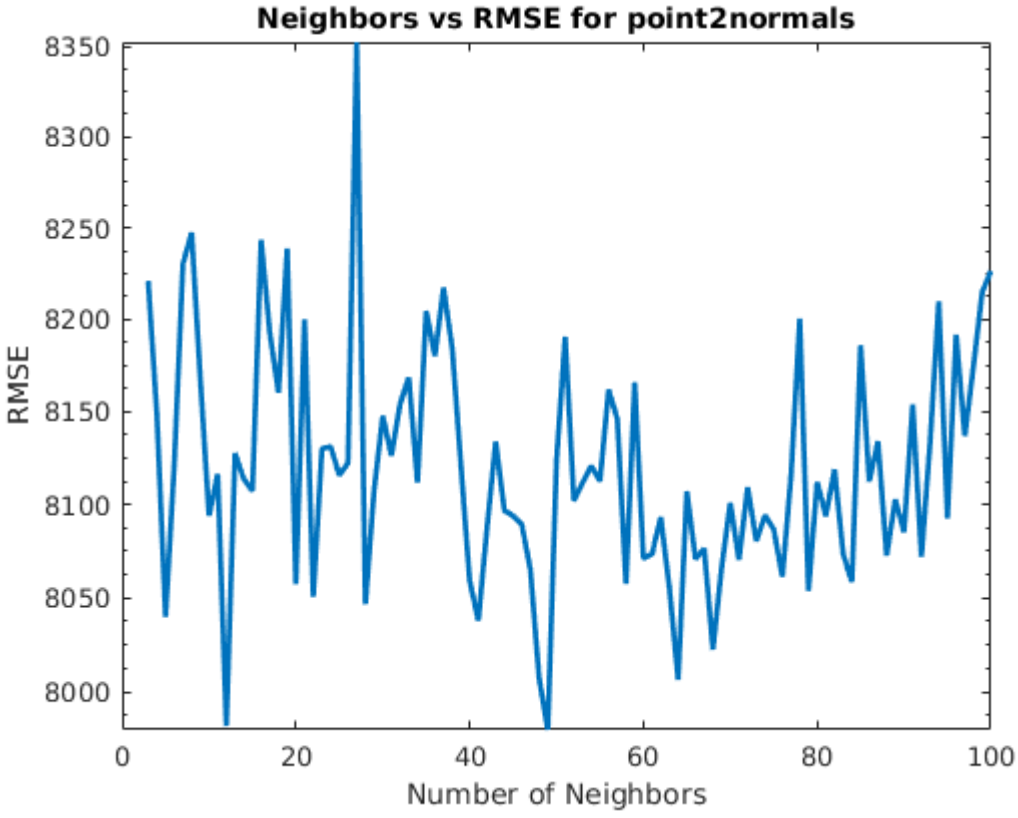


Using the best neighbor value from the above tests, we can attempt to estimate the surface of the bunny. For the surface estimation, we will use the best parameters that were determined in assignment 4 with an ep value of 200 and a neval of 100. As can be seen in the comparison below, the normal vectors are estimated well except for around sharp edges. At these sharper edges, the error is greater.



The point2normals Function:

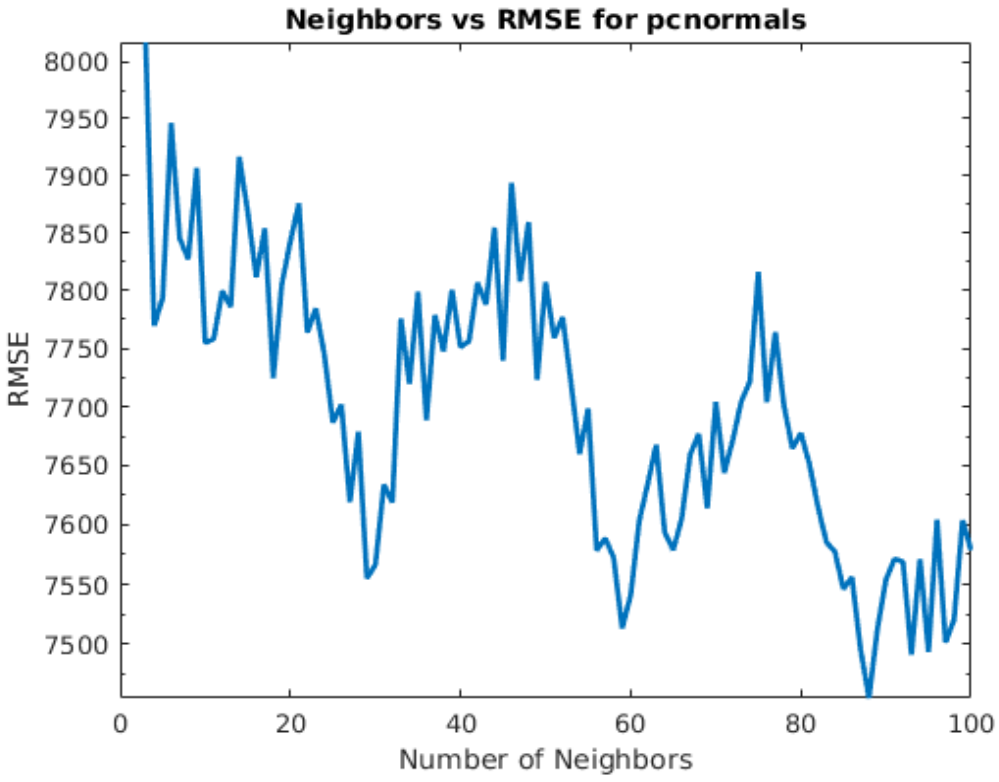
This function comes from the MathWorks file exchange website. It is part of a larger package that requires the statistics and machine learning toolbox. I am unsure if this single function requires that package. This function does a least squares normal estimation using principal component analysis (PCA). It takes as inputs the points to estimates the normals. By default it used 100 neighboring points to make this estimate. The code was changed to take neighboring points as a parameter. The estimates from this function was then compared using the bunny file as in the previous section. As we can see from the graph below, the error for this function is much higher than the previous function. The number of neighbors with the lowest error for this function is 49.



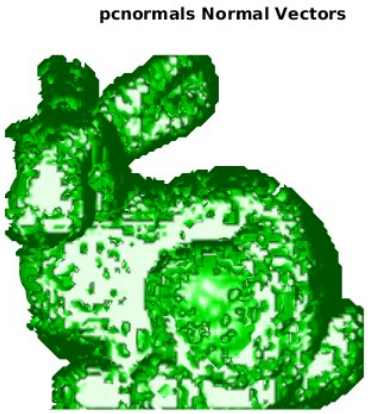
Using the same implementation as the previous example, we can produce the surface of the bunny. As can be seen in the comparison below, the surface forms the recognizable shape of the bunny, but it is a very distorted bunny with many holes. Overall, this function does a poor job of estimation.

The pcnormals Function:

This function is contained in the Matlab computer vision toolbox. This function estimates the normals similar to the first function. This function uses a set of neighboring points to calculate an estimated plane close to the vertex. The normal vector for this plane is then used for that vertex. The function takes two inputs. The first input is the points to estimate normals for. The second input is the number of neighbors. The estimates for this function were then compared to the bunny file as in the previous sections. As can be seen from the error graph below, the error is very high. The lowest error in this graph comes from the neighbor value of 88.



Using the same implementations as previous sections to reconstruct the bunny, we can see the results of this function below. This function also does a poor job of estimating the normal vectors. As we can see from the image, there are many holes in the reconstruction.



Normal Vector Error:

As can be seen from the previous sections and graphs, the error for the normal vector is very high. At first it seems the error is too high, but there is a simple explanation. For some vertices, the normal vector is calculated in the opposite direction. Below is a small subset of the degree difference between some points in one of the estimates. Below we see the difference in degrees between the real normal vectors and the estimated normal vectors for 33 points.As can be seen, several points have extremely large differences between the two normal vectors which show a normal vector pointing in almost the complete opposite direction. In the normals estimated by the 'findPointNormals' function, which produced the best results, a total of 1827 normal vectors had a difference above 90 degrees, 1807 had a difference above 135 degrees, and 1532 had a difference above 170 degrees. So as can be seen, there are some extreme errors produces by the normal vector estimates. As can be seen from the rabbit pictures, these errors seem to happen mostly along the sharp edges of an image.

7055	1.8535
7056	0.0676
7057	3.5454
7058	7.4117
7059	4.4499
7060	4.0258
7061	5.5973
7062	2.8447
7063	8.1498
7064	5.0735
7065	0.7665

7066	1.7672
7067	174.8126
7068	164.8263
7069	150.3959
7070	157.8698
7071	170.8629
7072	162.2037
7073	167.5895
7074	166.8496
7075	175.2641
7076	169.4122

7077	160.5654
7078	165.2103
7079	175.9050
7080	173.0211
7081	102.0210
7082	177.8207
7083	179.3414
7084	176.4977
7085	177.0741
7086	169.8795
7087	174.4960

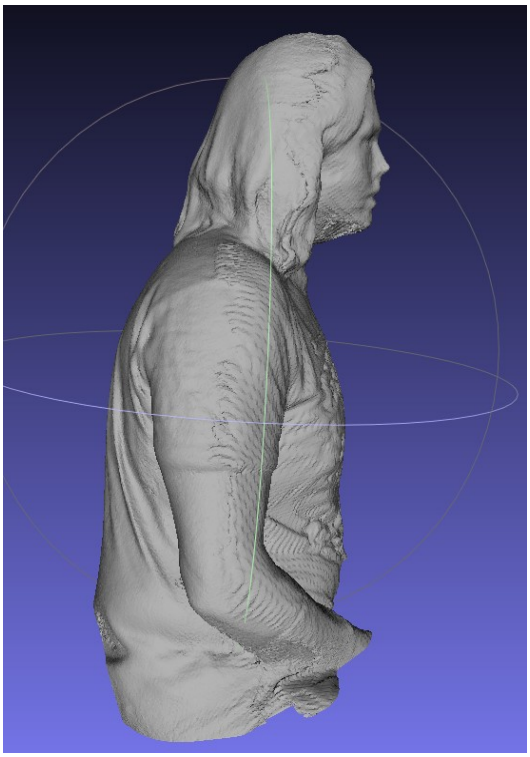
3D Scan Reconstruction:

The initial plan for this assignment was to reconstruct a 3D scan of myself. However, with the poor normal vector estimation, the reconstructions came out poorly. The 3D scanned images have many more edges than the rabbit example, so this data set will show the quality in using these normal vector estimations with a much more complicated image.

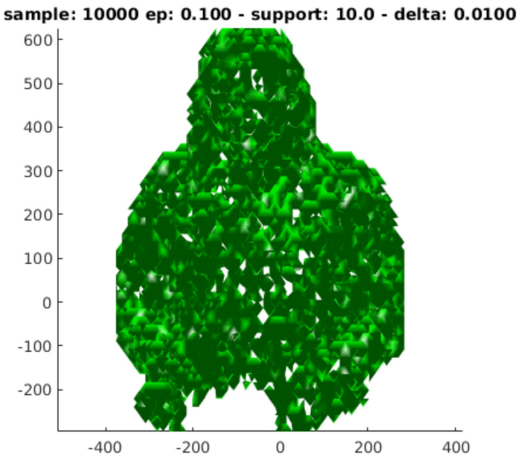
To create the 3D scan, an Xbox One Kinect was used along with Microsoft 3D Builder Studio on Windows 10. The 3D Builder software uses the various cameras on the Kinect to stitch multiple images together. To get a complete image of a person, the person must rotate in place. This was done by sitting on a stool in front of the camera and spinning the stool.

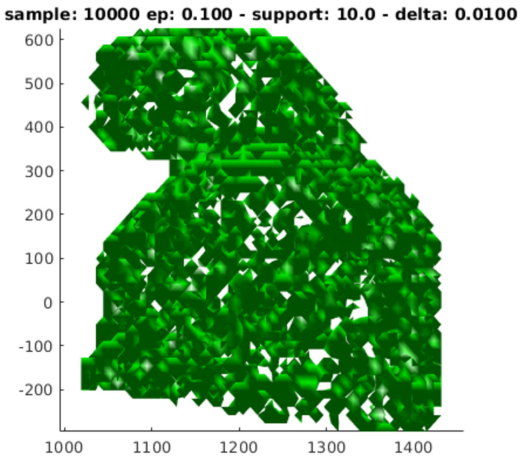
Next, the 3D Builder program converts the stitches images into an Stereolithography (STL) file. These STL files can contain several types of information, but the STL used with this scan provides the vertices of points of the scanned object, the faces of the polygons, and the normal vectors to those polygons. Since the normal vectors to the vertices are not supplied, we must use the estimated normal vector functions covered in the previous section of this paper. Since the 'findPointNormals' function provided the lowest error in the bunny data set, it will be used to estimate the normals.

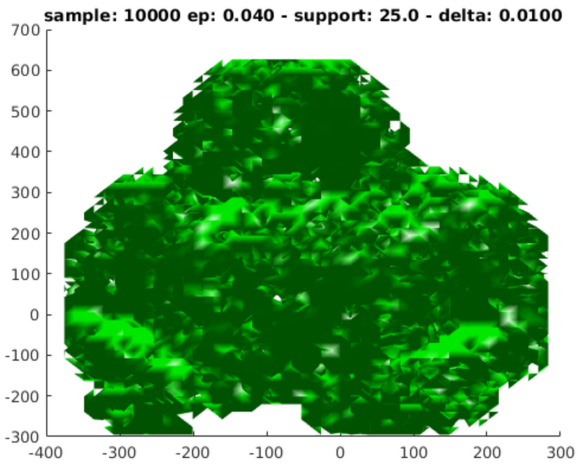
The following are the original images that we are attempting to recreate.

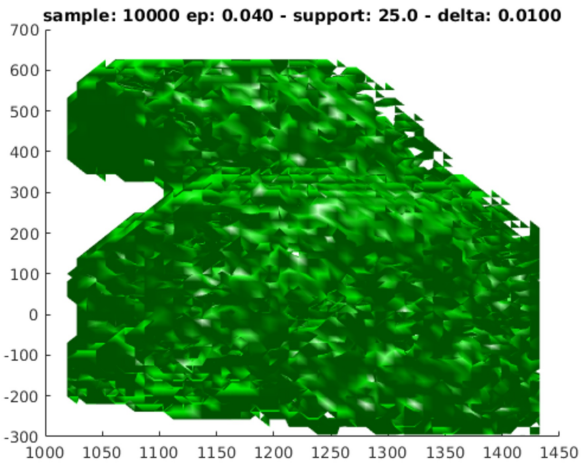


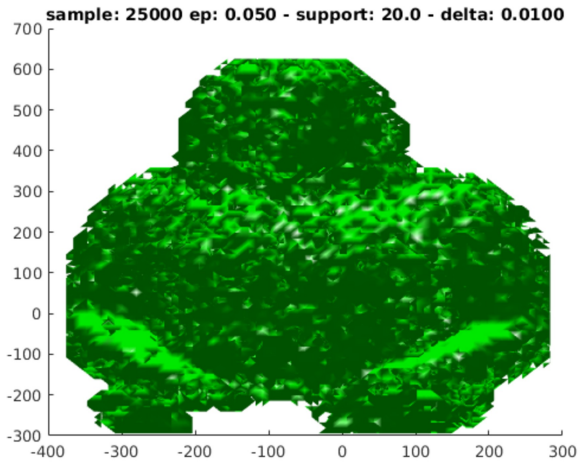
After many hours of trying different parameters, no satisfactory reconstructions were ever produced. Since this method of using compact support radial basis functions depends highly on accurate normal vectors, this result was expected. The following images show a few of the several attempts with different parameters and settings to reconstruct the above images.

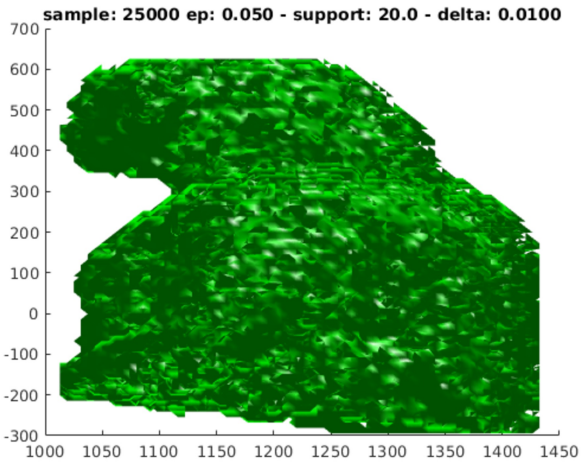


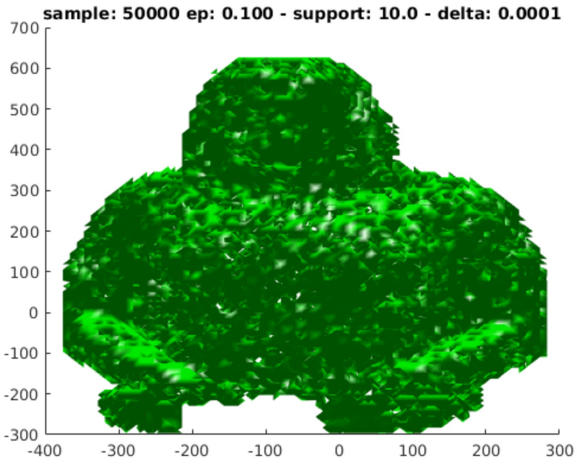


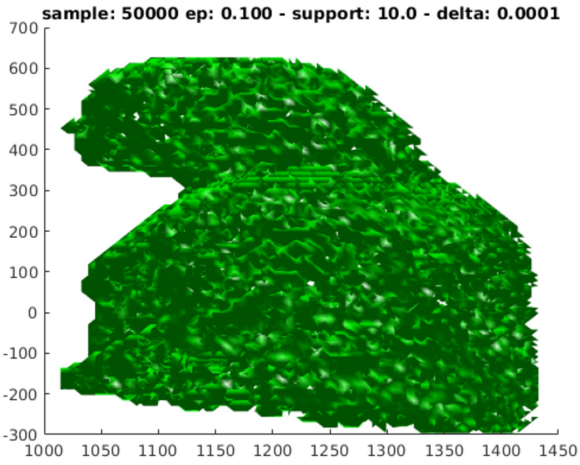


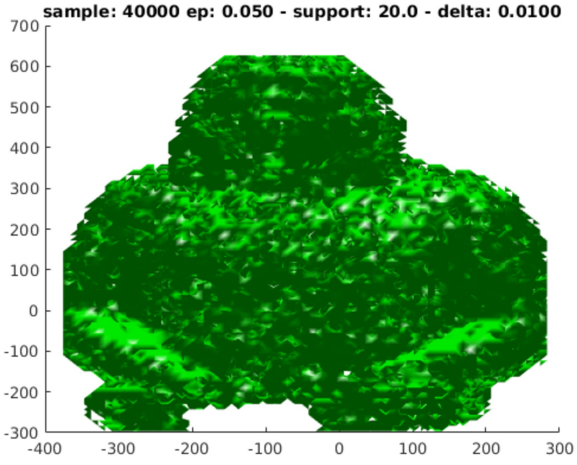


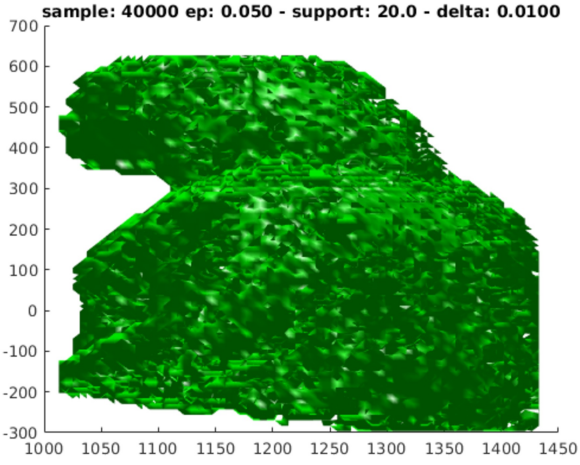

sample: 25000 ep: 0.050 - support: 20.0 - delta: 0.0100

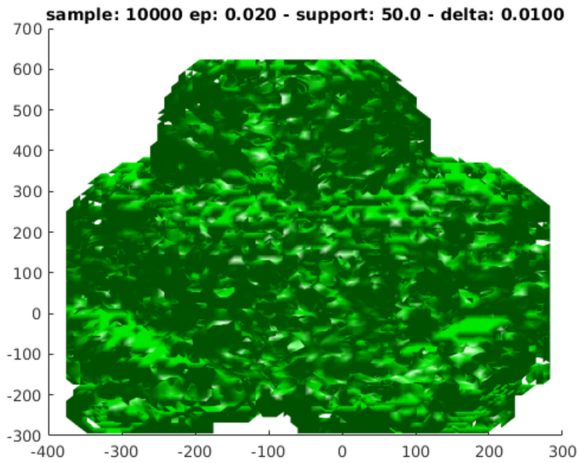

sample: 25000 ep: 0.050 - support: 20.0 - delta: 0.0100

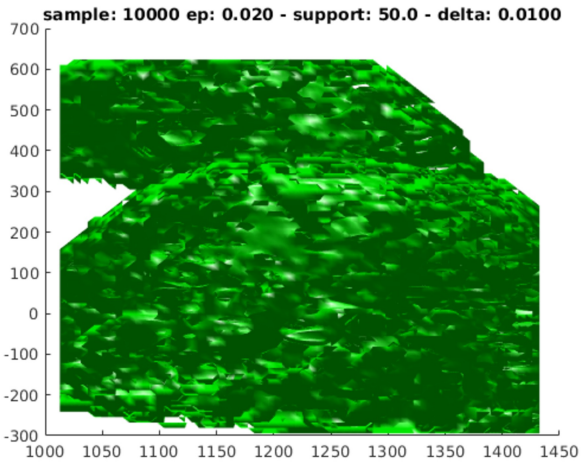

sample: 50000 ep: 0.100 - support: 10.0 - delta: 0.0001


sample: 50000 ep: 0.100 - support: 10.0 - delta: 0.0001


sample: 40000 ep: 0.050 - support: 20.0 - delta: 0.0100


sample: 40000 ep: 0.050 - support: 20.0 - delta: 0.0100


sample: 10000 ep: 0.020 - support: 50.0 - delta: 0.0100
700
600
500
400
300
200
100
0
-100
-200
-300
-400
-300
-200
-100
0
100
200
300


sample: 10000 ep: 0.020 - support: 50.0 - delta: 0.0100
700
600
500
400
300
200
100
0
-100
-200
-300
1000
1050
1100
1150
1200
1250
1300
1350
1400
1450