Generating the stomach fibres for Gastric Motility

Amogh Mishra, Dr. Shameer Sathar. Auckland Bioengineering Institute, The University of Auckland, New Zealand.

AIM: The focus of this report is to discuss the methods for the generation the three stomach fibers to study the effect of the electrical activity on the smooth muscle cells(SMCs).

BACKGROUND: Gastrointestinal motility is coordinated by the electrical activity known as the slow waves. These waves are responsible for the phasic contraction of SMC and Interstitial cells of Cajal (ICC) are responsible for generation and the propagation of these waves through the network. Understanding the effects of the different muscle layer orientations on the underlying slow wave patterns is important to identify diagnostic and predictive tools for therapeutic purposes. In this report, we discuss the method of generating the three stomach fibers and also the methods which are used to classify the points to their corresponding surface.

SOFTWARES USED:

- For the purpose of the generating the conductivity tensors, we have used the CHASTE framework. CHASTE stands for **C**ancer, **H**eart **a**nd **S**oft **T**issue **E**nvironment, which is a framework used for simulating various problems associated to biology and physiology. This framework is maintained by the University of Oxford.

- For the purpose of visualizing the generated meshes from the CHASTE, we have used ParaView which is an open-source data visualisation tool.

- For the purpose of creating a geometry, we have used Blender which is an open-source 3D graphics and animation software.

- For the purpose of generating tetrahedral elements, we have used TetGen which was used to generate tetrahedral elements.

- For the purpose of converting stl files to .poly, .smesh, .ele, .face, .node, we have used the command line switches of the Triangle- a 2D mesh generator.

- Python language was used for other computational and file operation purposes.

CODES: All the codes related to this report are available in the Github repository.
URL:github.com/AmoghM/UoA_Internship

METHODS:

1) Generation of the Oblique Layer

- The stomach is composed of three types of layers: 1) Circular muscle layer 2) Longitudinal muscle layer 3) Oblique muscle layer. To generate these muscle fibers, we used CHASTE framework to obtain the conductivity tensors in the three directions. These are: 1) Fibre, 2) Sheet 3) Cross. These three tensors are orthogonal to each other and they orient themselves according to the fiber layout in the stomach. Our aim was to generate the oblique layer for which we took a dummy geometry of a cuboid of the dimensions, 3cm x 2cm x 0.5cm. This geometry was built using the blender software. We used tetgen command line syntax to generate a several tetrahedral elements for better visualisation purpose.
- Using CHASTE, we iterated through all the tetrahedral element of the geometry to calculate the coordinates of the centroid of each element. The entire width of 0.5cm was trisected and each section was responsible for the creation of Circular, Oblique, Longitudinal muscle layer respectively.

- To generate the oblique layer, we interpolated the surface for the angle 60 degree and 30 degree on the principal diagonal corners and 45 degree each on the other diagonal corners to retrieve the rotation angle with which the tensors should be rotated. The interpolation was done using the scipy package of the python programming language. We rotated the tensors across the z-axis by the angles obtained using the transformation formula:
    - x' = x*cos q - y*sin q
    - y' = x*sin q + y*cos q
    - z' = z
- The resulting visualization of the rotated tensor exhibited curving along the surface verifying the generation of the layer

2) Classification of nodes to surfaces

- This was a crucial step since the necessary information about the realistic models of the stomach fibers can be obtained by solving the Laplace Dirichlet equation over the stomach geometry for which it is necessary to assign correct nodes with the gradient value.

- We ran three tests to classify the points. Those three tests were:
    - Test-1: Testing if the foot of the projection of a node lies inside the triangle
    - Test-2: Testing if the node to be classified lies itself in the triangle
    - Test-3: K-means clustering, Agglomerative clustering

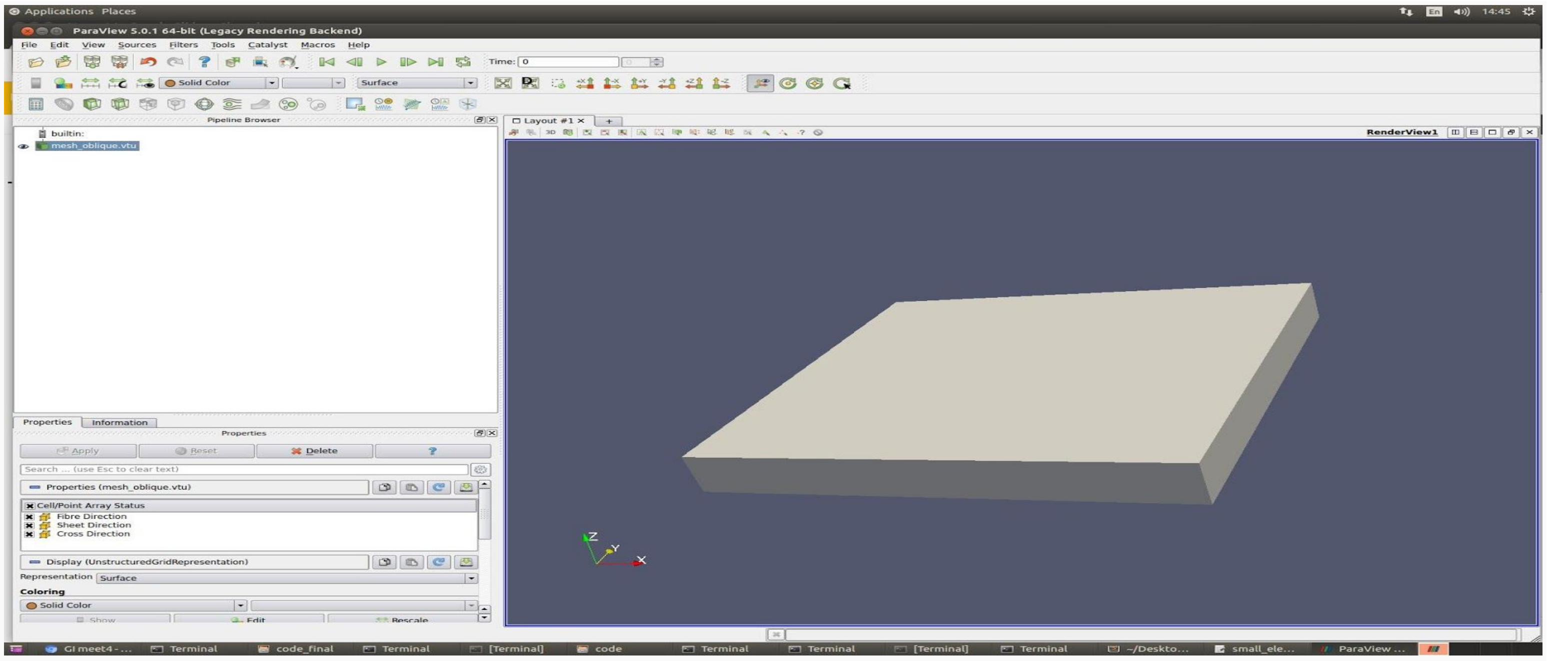
- Test 1: The algorithm for this test is explained in the stackexchange for mathematics, link: Determine if projection of 3D point onto plane is within a triangle. Once we calculated the coordinate point of the foot of projection, we calculate the euclidean distance between the point whose projection is calculated and the foot of projection to determine the margin of error by which a point is away from the surface. Ideally, the distance should be 0 for classification but we have witnessed the point to be lying slightly away from the surface hence a threshold value needs to be decided. If the distance calculated comes in the range of this threshold value which signifies the margin of error, then we classify the point to lie on that surface.

- Test 2: The algorithm for this test is explained in the stackexchange for mathematics, link: Determine if projection of 3D point onto plane is within a triangle. The idea of this test was based on simple reasoning, for a point that needs to predicted to belong to a surface, the point should lie on any one of the triangulated element. This leads us to the ideal condition that we mentioned in the test-1, the distance between this point and the point of projection should be 0 always. Since it is a idealistic test, the result we obtained were not satisfactory which motivated us to try the Test-1 only.

- Test 3: This test uses the power of machine learning unsupervised learning algorithm, K-means clustering and agglomerative clustering. The reasoning behind choosing this technique is fairly simple, points belonging to a surface would lie near to each other making clusters and for classifying a given point, we just need to calculate the cluster to which this point is nearer to. Although the logic is simple and convincing, this theory couldn’t justify the results obtained due to the problem of unequal blobs. Not all the surfaces had large number of points to help the algorithm create clusters and hence resulted in smaller clusters getting engulfed by the other bigger clusters.

RESULTS:

1) Oblique Layer Generation:



Properties (mesh_oblique.vtu)

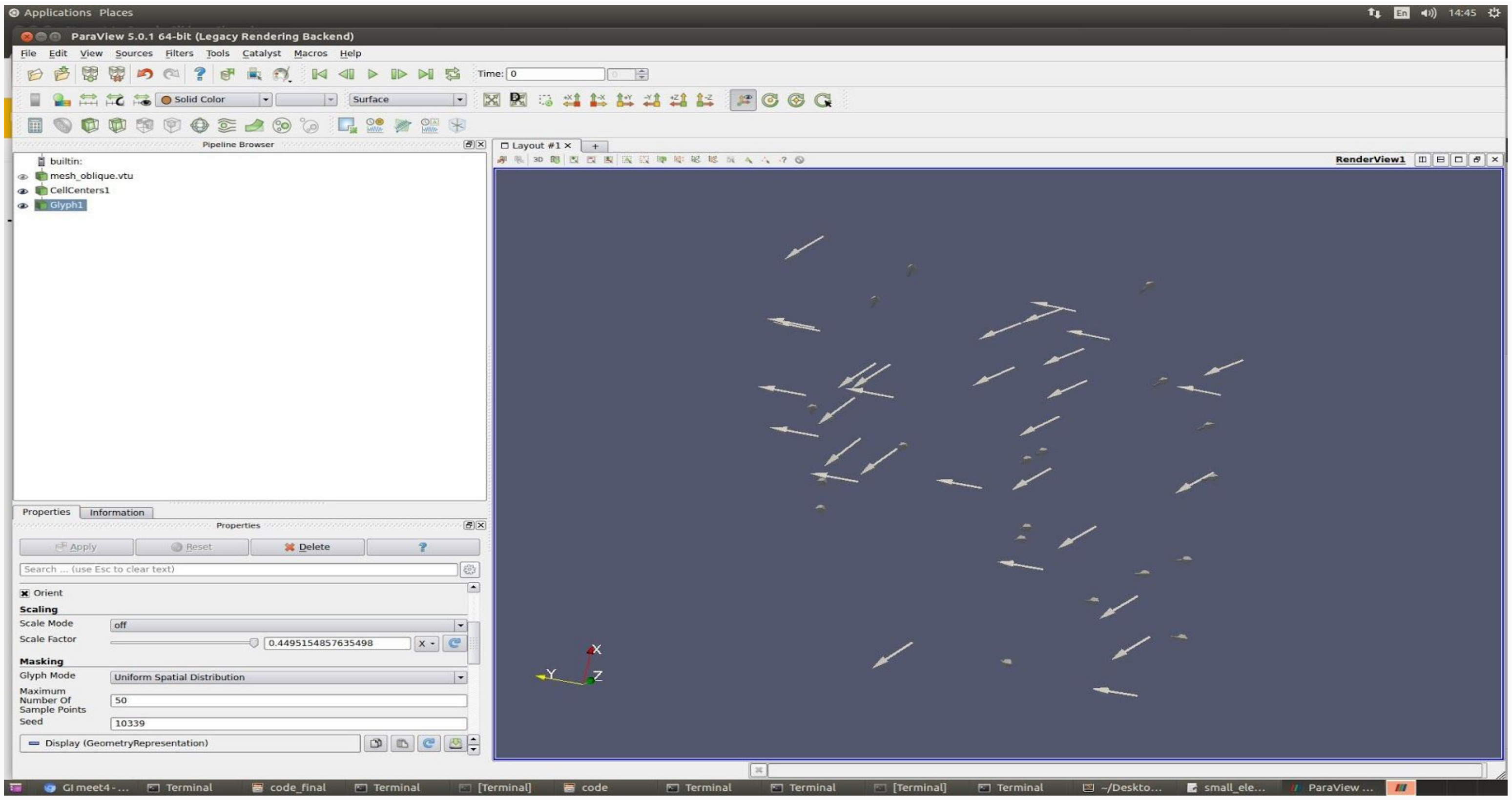
Cell/Point Array Status
- [x] Fibre Direction
- [x] Sheet Direction
- [x] Cross Direction

Display (UnstructuredGridRepresentation)

Representation: Surface

Coloring: Solid Color

Cuboidal Geometry



- [x] Orient

Scaling

Scale Mode: off

Scale Factor: 0.4495154857635498

Masking

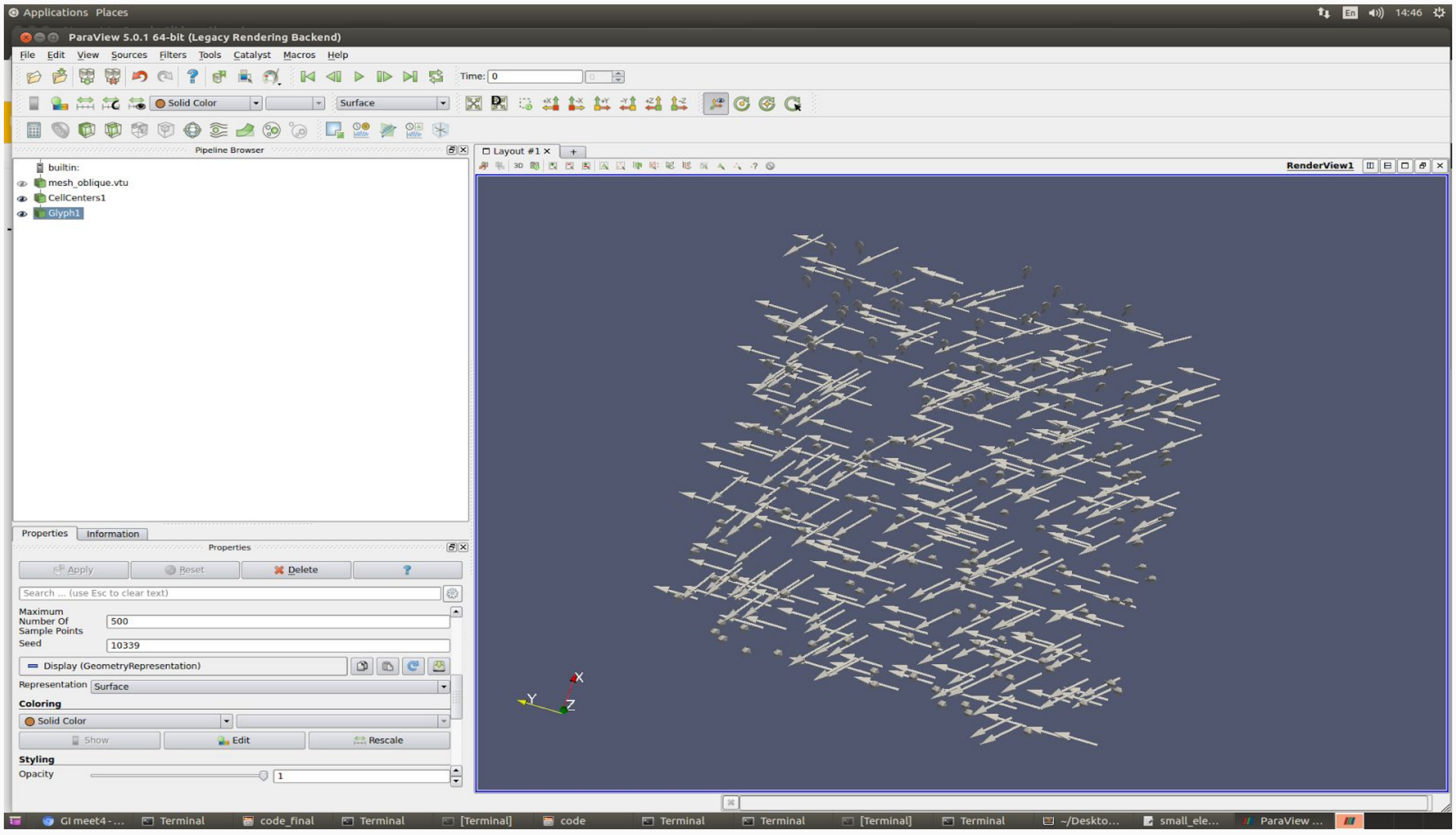
Glyph Mode: Uniform Spatial Distribution

Maximum Number Of Sample Points: 50

Seed: 10339

Display (GeometryRepresentation)

Oblique layer generation (visualised with less points)



Time: 0

Maximum Number Of Sample Points: 500

Seed: 10339

Display (GeometryRepresentation)

Representation: Surface

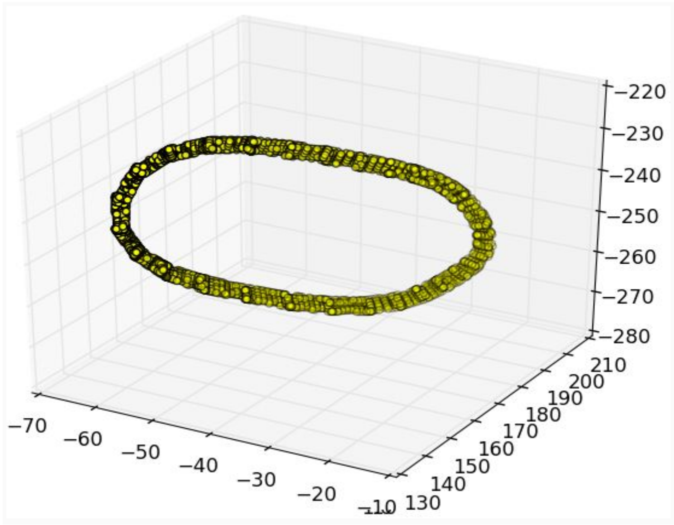
Coloring: Solid Color

Opacity: 1

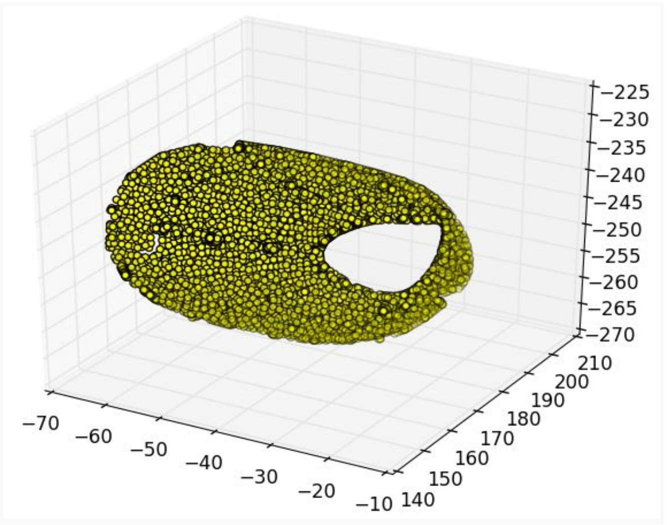
Oblique Layer Generation (with more points)

2) Classification of nodes to surfaces:

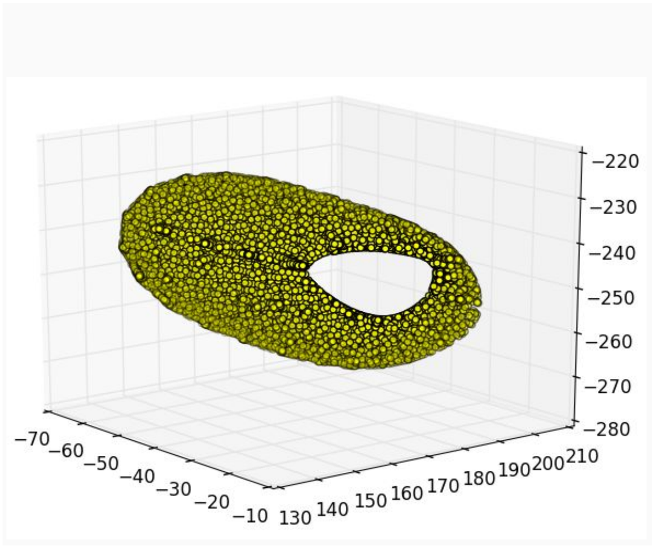
TEST 1: Following are the results of the scatter plot representing the stomach element with the threshold of 0.1 units



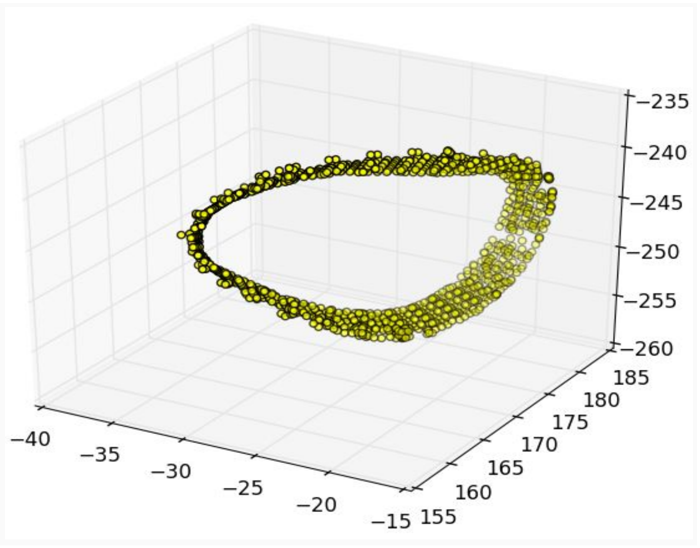
Surface 1



Surface 2

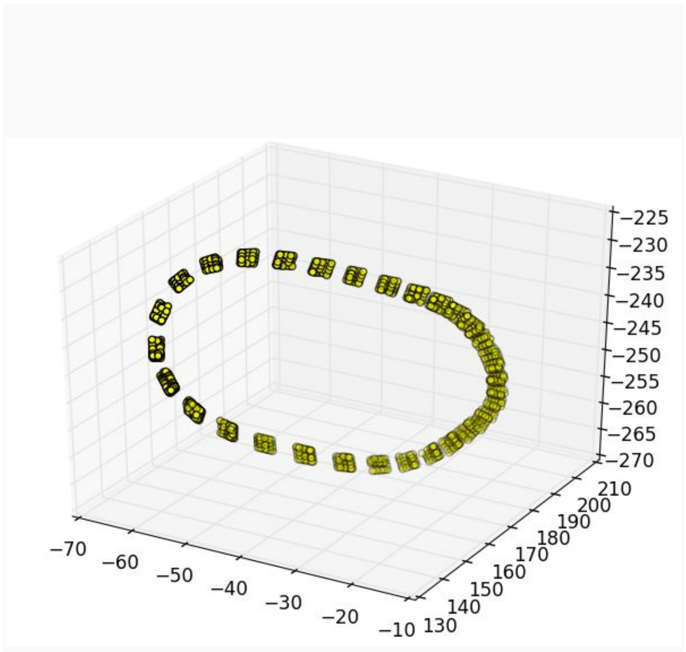


Surface 3

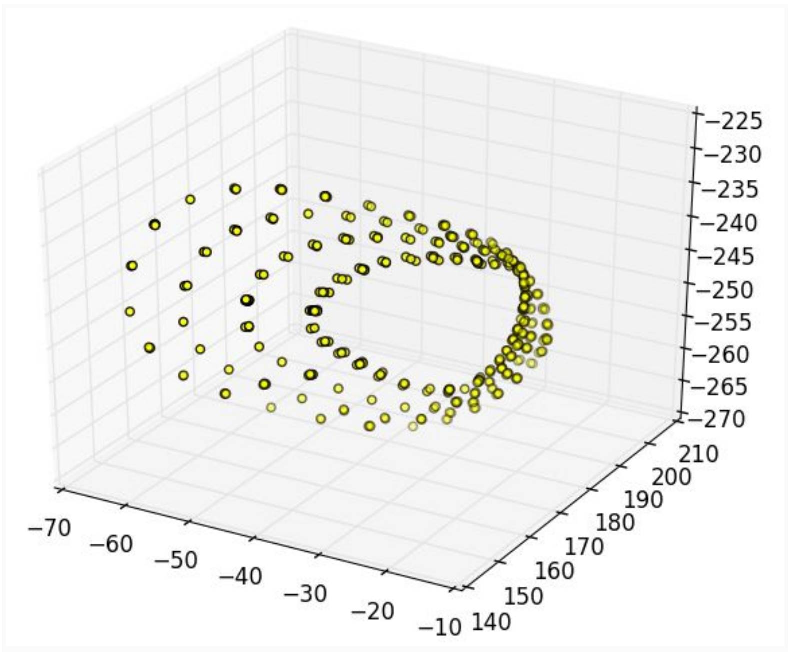


Surface 4

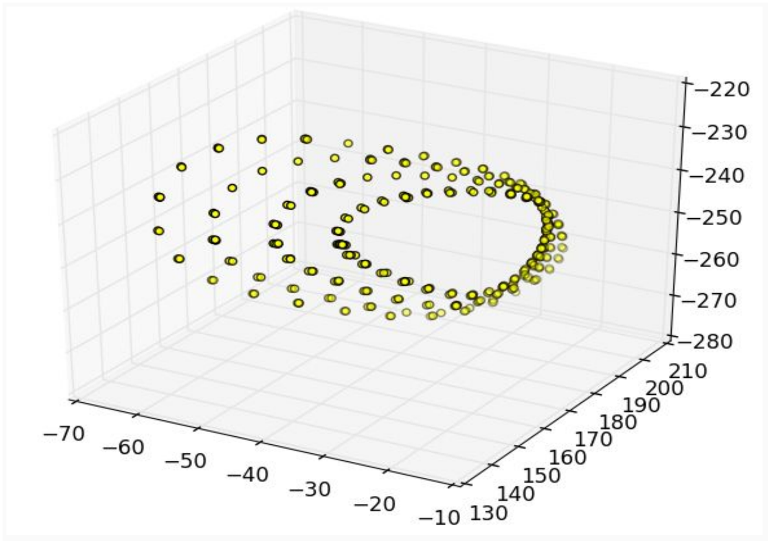
TEST 1 and Test 2: Following are the scatter plot representing the stomach element with both the tests combined.



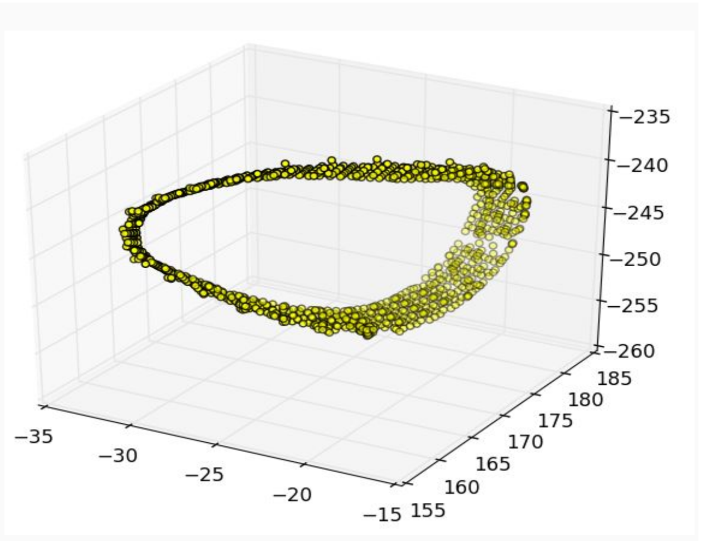
Surface 1



Surface 2

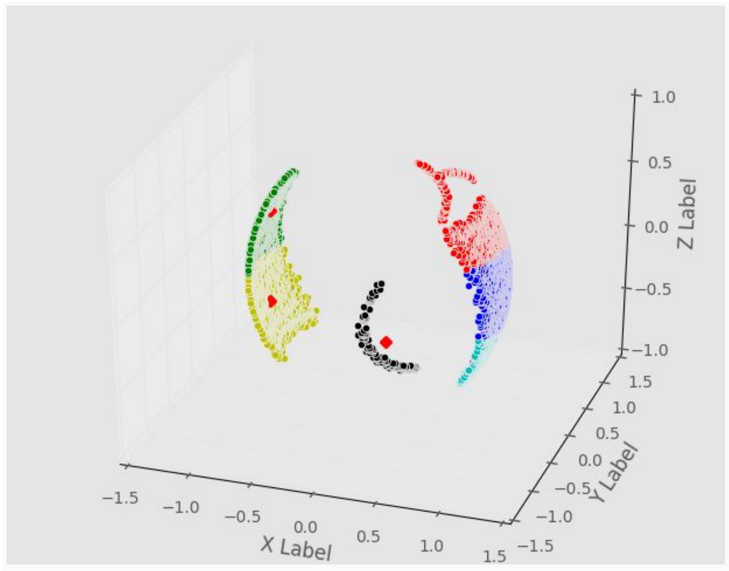


Surface 3

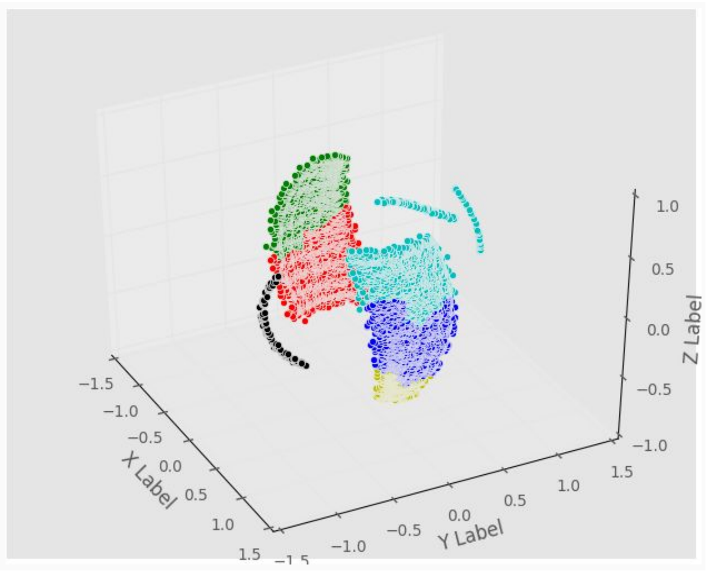


Surface 4

TEST 3:



K-Means Clustering where k=6



Agglomerative Clustering

CONCLUSIONS:

- Test (1) alone was capable to classify considerable amount of points.
- Test (1) and (2) together gave a harder restriction to classify the points. Hence, too less points were classified.
- Clustering was incapable of classifying the nodes due to unequal blobs.