$3-D$ Placenta Image Construction

Mutaz Alsayegh, Daniel Macias, Ramy Heng, Rattana Hor
Supervised By: Dr. Jen-Mei Chang
CSULB

May 19, 2011

Abstract

In this project, we present a methodology to construct a three dimensional image of a placenta from a two dimensional image. This construction is accomplished by first obtaining data extracted from the image data provided in our Math 579 course by Dr. Carolyn Salafia. We will then use a least squares trigonometric interpolation to create real-valued functions to describe our image data, hopefully re-creating the structure from the image. We will then triangulate our data to create a three dimensional interpretation of the placenta image. This three dimensional image will hopefully be useful in future work to find a correlation or relationship between the health of the developing fetus and the geometry of the placenta.

1 Introduction

The placenta is an organ that is developed during pregnancy in the uterus. Its functionality is crucial for the health of the fetus. The placenta connects the fetus to the uterine wall to transfer oxygen and nutrients to the developing fetus, and passes waste product back into the mother's blood stream via a vessel network. We can see this process in Figure 1.

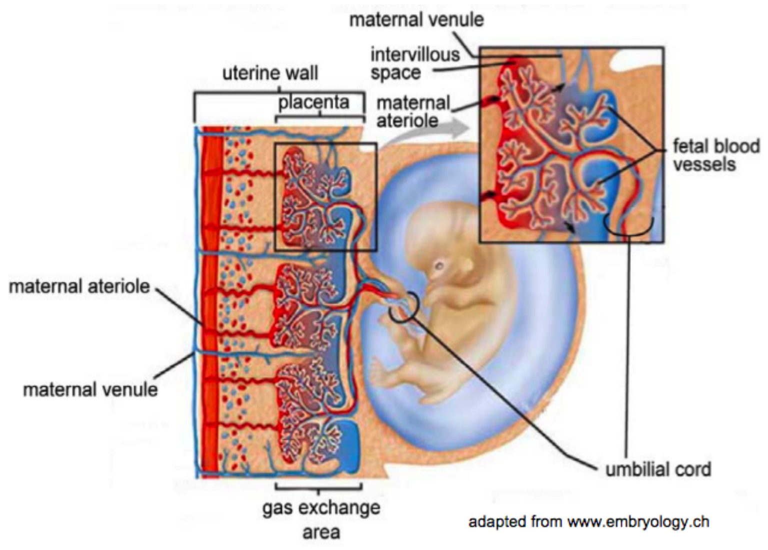

Figure 1: Image to visualize placenta's role in the development of the fetus

Since nutrients and oxygen are crucial for the survival and health of the fetus, and these products are delivered by maternal blood via a vessel network, we thought it would be interesting to study and establish a relationship between the density of the vessel network present in the placenta, and the health of the baby. However, since we lack the ability to directly dissect and measure the amount of vessels present in placentas, we decided to diverge from our goal, and first establish a methodology for constructing a three dimensional image of placenta which may enable us to find a correlation between the density of the vessel network and the three dimensional shape of the placenta. Therefore, the three dimensional image will replace the real placenta for our purposes. Once this method is well established, we planed to create a metric (a way to measure) that yields a relationship between the shape of the placenta, and the density of the vessel network that is present in the placenta. This relationship between the placental geometry and the density of the vessel network may allow us to find a relationship between the health of the baby, and the density of the vessel network. In this report, we present our attempt to construct a three dimensional image of placenta using very limited data that was presented to us. We will now outline the process of constructing the three dimensional image of the placenta, and we will present details of each step in the sections that follow. We will use the following image to derive our initial data.

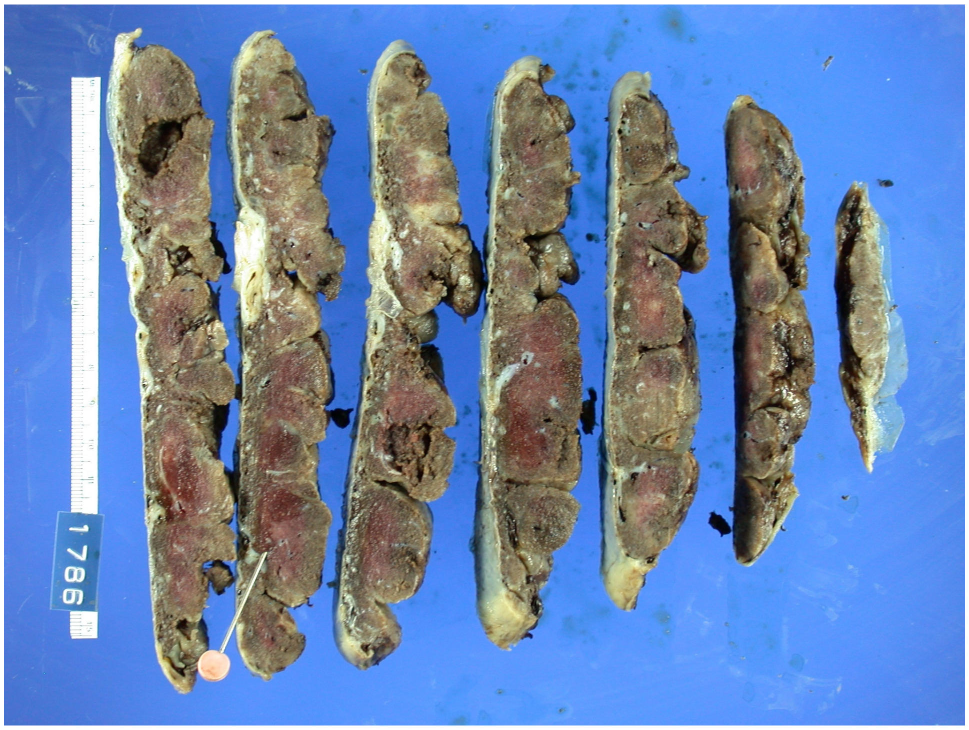

Figure 2: The given data we were presented.

We will outline the boundaries of each slice from the image, we then store the result of each slice in a matlab file storing and reorganizing them in their correct orientation. The tracing process is not done by hand but through a code that we have built using Java. The result of this process applied to a few slices is presented in Figure 3.

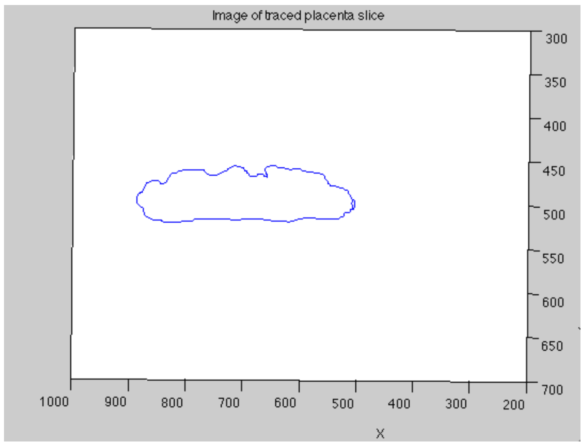

Figure 3: Example of traced placenta slice using the Java code.

The second step of this process is to interpolate each slice. This interpolation will give us a Mathematical function that will allow us to control the graphing process. A function can be thought of as a black box that takes in a number, and outputs another number. Once this process is done for each slice, the functions that will be obtained from the interpolation process will allow us to plot the boundaries of each slice in the order corresponding to the Figure 2. The interpolation process will also allow us to insert points, if necessary, to fill in any gaps that might be present. Now that we have each image in the correct position, we need to fill in the empty spaces between each slice. This process is well know in Mathematics as a triangulation.

2 Research Methods

2.1 Data Points Collection

Our first step is to collect data points on the boundaries of each slice of placenta. The code we built in Java enables us to accomplish this task. This code also allows us to collect data points as many as we want. In our project, for instance, we collect 201 points. We load the Java program and open the image of placenta slices. (They are in two separated windows.) How the code works is after we click on the "run" button to make the code run, it will not run right away. It pauses about 5 seconds, which gives us enough time to place the cursor at the location where we want. We then put the cursor at one end of the slice and move it along the boundary of the slice until the other end. The program stops when it reaches 201 points. See Figure 4 for demonstration. The y-coordinates of those points are actually discrete functions – functions that are represented by points and there are no connections from point to point. Those functions represent the placenta's boundaries.

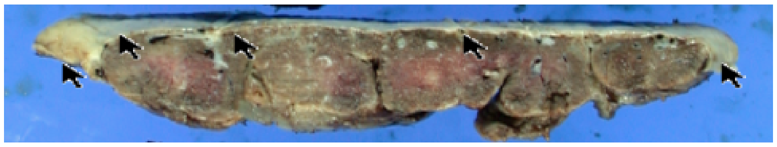

Figure 4: Getting data points by moving a cursor along the top side of placenta slice

2.2 Interpolation

Once we obtain our points, we need to interpolate them. That is, we find a function which fits the data with a smooth continuous real-values polynomial. Hence, interpolation is a method of constructing new data points using the known data points. Since we are dealing with discrete functions, we use a Discrete Least Squares approximation method for the purpose of interpolating these functions. The method is as follow:
Suppose a collection of $2m$ paired data points $\{(x_j, f(x_j))\}_{j=0}^{2m-1}$ is given. In the interval $[-\pi, \pi]$, we have equally spaced points, as shown in Figure 5.

$$x_j = -\pi + \tfrac{j}{m}\pi, \text{ for each } j = 0, 1, ..., 2m-1.$$

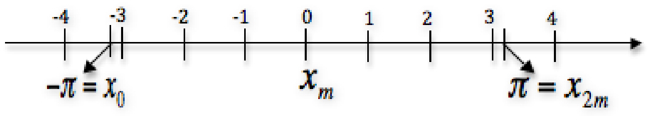

Figure 5: Equally spaced points x_j

Note: For the case of the arbitrary interval $[a, b]$, the linear transformation is used to transform $[a, b]$ to $[-\pi, \pi]$. The transformation is

$$z_j = \pi + \tfrac{2\pi}{b-a}(x_j - b).$$

and the transformed data is of the form

$$\{(z_j, f(\tfrac{b-a}{2\pi}(z_j - \pi) + b))\}_{j=0}^{2m-1}$$

For each $n \in \mathbb{Z}^+$, $\{\phi_0, \phi_1, ..., \phi_{2n-1}\}$ is an orthogonal set with respect to summation over $\{(x_j\}_{j=0}^{2m-1}$ in $[-\pi, \pi]$, where

$$\phi_0(x) = \tfrac{1}{2},$$
$$\phi_k(x) = \cos kx, \text{ for each } k = 1, 2, ..., n,$$
$$\phi_{n+k}(x) = \sin kx, \text{ for each } k = 1, 2, ..., n-1.$$

By this we mean for each $k \neq l$,

$$\textstyle\sum_{j=0}^{2m-1} \phi_k(x_j)\phi_l(x_j) = 0.$$

We have a set $\tau_n = \{S_n(x)\}_{n\in\mathbb{Z}^+}$, where $S_n(x)$, a trigonometric polynomial of degree less than or equal to n, is the linear combination of all the elements in τ_n. That is,

$$S_n(x) = \frac{a_0}{2} + a_n cosnx + \sum_{k=1}^{n-1}(a_k coskx + b_k sinkx). \tag{1}$$

Our goal is to determine S_n that will minimize

$$E(S_n) = \textstyle\sum_{j=0}^{2m-1}[f(x_j) - S_n(x_j)]^2.$$

That is, we want to find constants $a_0, a_1, ..., a_n, b_1, ..., b_{n-1}$ so that

$$E(a_0, a_1, ..., a_n, b_1, ..., b_{n-1}) = \textstyle\sum_{j=0}^{2m-1}\{f(x_j) - [\frac{a_0}{2} + a_n\text{cos}nx_j + \sum_{k=1}^{n-1}(a_k\text{cos}kx_j + b_k\text{sin}kx_j)]\}^2$$

is a minimum.
Those constants can be proved to be

$$a_k = \frac{1}{m}\sum_{j=0}^{2m-1} f(x_j)coskx_j, k = 0, 1, ..., n, \tag{2}$$

and

$$b_k = \frac{1}{m}\sum_{j=0}^{2m-1} f(x_j)sinkx_j, k = 1, 2, ..., n-1. \tag{3}$$

As mentioned in step one, we are able to obtain data points $\{(x_j, f(x_j))\}_{j=0}^{2m-1}$ along each placenta slice's boundary, where $f(x_j)$ are discrete functions that need to be approximated. Once we get those points, we plug them in (2) and (3) to find the constants a_k and b_k. Then we use those constants to determine trigonometric polynomial S_n in (1), and S_n is used to approximate $f(x_j)$. Hence the function S_n is a model for the placenta's boundaries.

2.3 Construction

The construction of the skeleton of our model relied entirely on extracting useful data from each slice to give us the proper shape. The quality and consistency of our structure is paramount in the construction process. Once we had these data points we interpolated them using the trigonometric interpolation process explained in the previous section. The purpose of this was to obtain a real-valued function that would describe and articulate the structure of the edges for each slice. These functions would act as support beams for the maternal side of our 3D placenta model. Once we obtained real-valued functions for each slice we gave them names and stored each of their values in a corresponding vector. These vectors are just sets of points that describe the shape of each slice. Once we obtained all of our data and once the interpolation process was complete, we were ready to proceed to the next step.

2.4 Triangulation

In trigonometry and geometry, triangulation is the process of determining the location of a point by measuring angles to it from known points at either end of a fixed baseline, rather than measuring distances to the point directly; another way of imagining this is as the division of a surface or plane polygon into a set of triangles, usually with the restriction that each triangle side is entirely shared by two adjacent triangles. The point can then be fixed as the third point of a triangle with one known side and two known angles. Figure 6 shows a a simple example of a triangulation done on a set of data points.

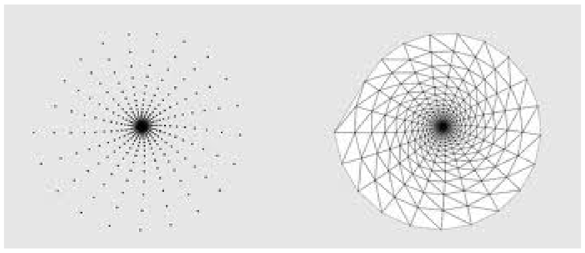

Figure 6: Triangulation on a Set of Data Points

If you imagine our data as a cloud of points scattered over the three dimensional plane (x, y, z); where x and y represent our real-valued function values and z is the thickness imposed on each placenta, then this point cloud created by our data set is the foundation for our triangulation, using these points we create a web connecting each slice and thus creating a surface over our interpolated functions and between all of our points. Figure 7 exhibits the beauty and efficiency of triangulation of a point cloud.

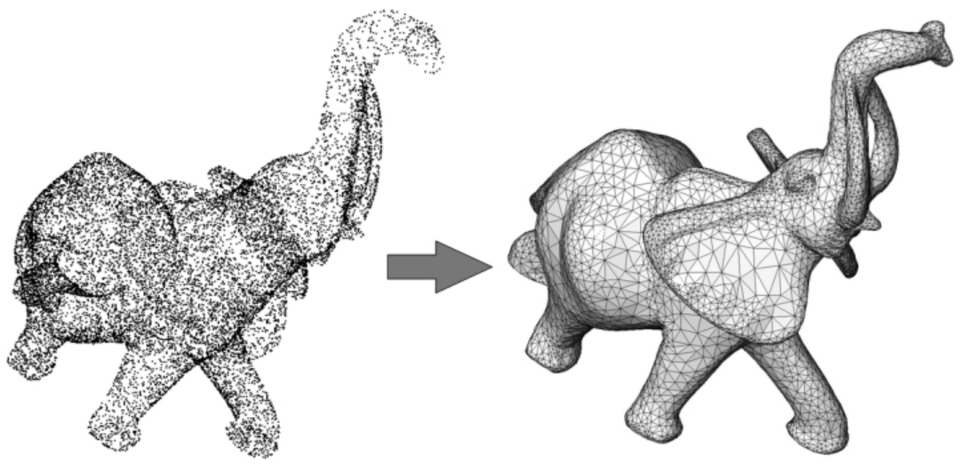

Figure 7: Left: Point Cloud of an Elephant. Right: Triangulation of the Point Cloud

There are various ways of creating surface via method of triangulation; fortunately Matlab is incredibly efficient at triangulation and interpolation so we decided to explore its various built-in functions and toolboxes to see if there was something that could possibly help us reach our goal of creating a three dimensional model of the placenta. After exploring some methods we found a triangulation function built into Matlab called the Delaunay function.

2.5 Delaunay

First developed by Boris Delaunay in 1934, the Delaunay Triangulation of a point set is a collection of edges satisfying an ”empty circle” property, which states that for each edge we can find a circle containing the edge’s endpoints but not containing any other points. Figure 8 displays this correlation.

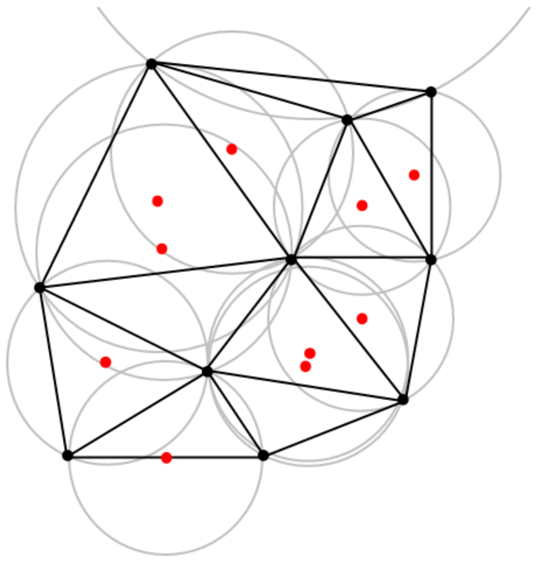

Figure 8: Delaunay Triangulation with Circumcircles

In other words, given a set of data points the Delaunay triangulation is basically a set of lines connecting each point to its natural neighbors, this guarantees that no vertex is in the interior of the circumcircle of any triangle in the triangulation. The Delaunay triangulation is related to the Voronoi diagram in that, the circle circumscribed about a Delaunay triangle has its center at the vertex of a Voronoi polygon as shown in Figure 9.

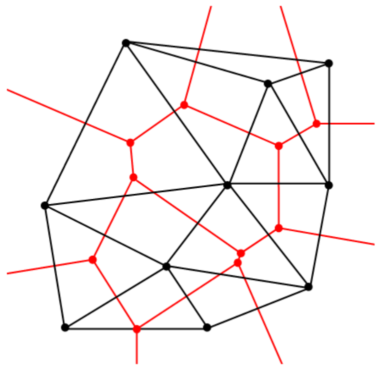

Figure 9: Connecting the centers of the circumcircles produces the Voronoi diagram (in red)

3 Results

The following is a brief description of how we used the Delaunay function in Matlab; I will not go into too much detail since the code is provided in our report. Using the function in Matlab is a relatively simple process; once we have isolated our values representing each slice into specific vectors (a vector is just an ordered collection of numbers) we can begin the triangulation process. We set the variable TRI = Delaunay(x, y); where each set of data points x and y represent the nx2 vectors assigned to each slice value. This allocation of the variable TRI will produce an nx3 matrix which is a set of triangles such that no data points are contained in any triangle's circumscribed circle. Each row of the matrix represents one Delaunay triangle and contains indices into the vectors x and y. Once we have obtained TRI we can plot our triangulated surface. Figure 10 shows two of our interpolated functions plotted together in a three dimensional space.

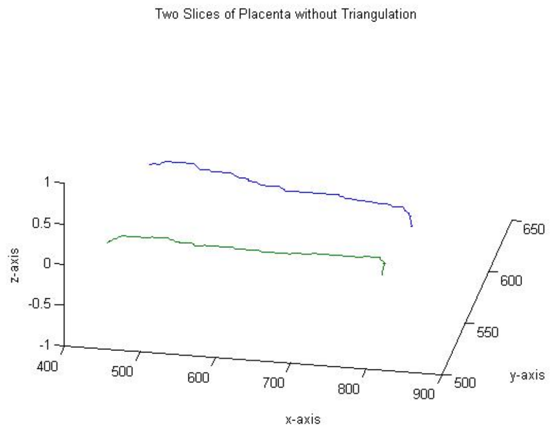

Figure 10: Two Interpolated Functions of Maternal Side of Two Slices

As you can see, problems arise in our initial triangulation. Notice the boundaries of the placenta slices; instead of triangulating only the surface, the function also triangulated the edges onto themselves. If you observe Figure 12 we can see that if we zoom in on only the boundary of the first edge we get this erratic triangulation. Notice the mesh created along the sides, this irregular triangulation is not what we had hoped for and will be problematic once we apply this algorithm to the entire picture. After exploring Matlab a bit more, we were able to obtain a solution for this problem. Using another method of plotting and being a bit creative with the construction of our vectors we were able to refine and isolate our mesh. Figure 13 is of the same two slices only, with a refined mesh and then plotted using the function Meshc. Observing this image in detail, we can see that the erratic triangulation observed in Figure 12 has been taken care of. The next step is to apply this same technique to more points. Because of the time constraints we were only able to apply this method to half of our data. Figure 14 shows the results we obtained when applying this technique to half of the placenta slices. As you can see the top and bottom of a half of our placenta were triangulated, the only task left was to connect these halves to form a complete three dimensional interpretation of the placenta. Obtaining this image was not an easy task and we are very proud of the work we accomplished, hopefully with more time and money this project can be approached with a little more finesse and possibly provide a new angle from which researchers or doctors can better understand a possible correlation between the geometry of the placenta and the health of the developing fetus.

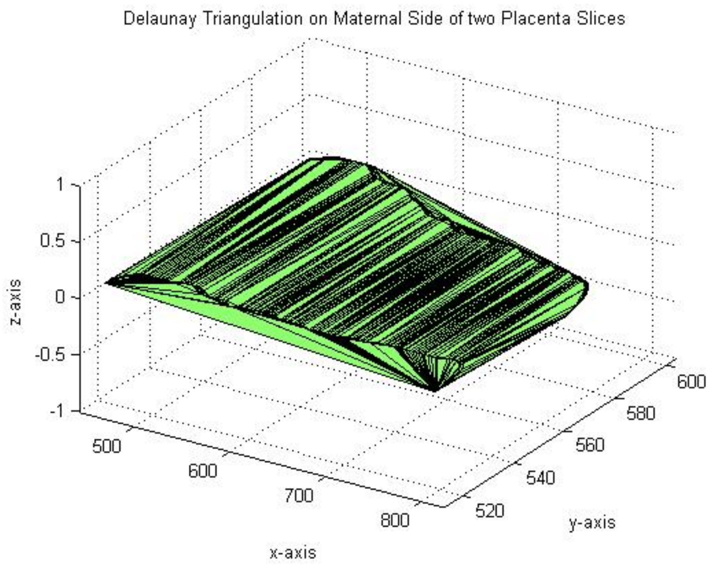
Delaunay Triangulation on Maternal Side of two Placenta Slices
z-axis
1
0.5
0
-0.5
-1
500
600
700
800
x-axis
520
540
560
580
600
y-axis

Figure 11: Triangulation on Two Slices from Previous Figure

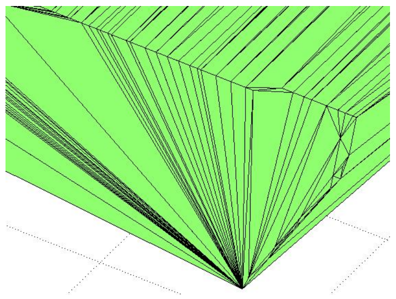

Figure 12: Close-Up on Boundary of Previous Triangulation

4 Future Work

In future work, we hope to complete the construction of the $3 - D$ image, which should not take much longer since we are familiar with the process. We then hope to automate the entire process of the construction. This should be manageable since we have automated every step in the construction process, except for the transitions from step to step. For example, we have the tracing process automated and so is the interpolation process. However, the transition step, from tracing to interpolation is accomplished manually. We also hope to establish some type of measure, that will relate the geometry (shape) of the placenta to the amount of vessel network that is present. A final step for this project will be to study if there exists any correlation between the health of the baby and the amount of vessel network that is present in the placenta.

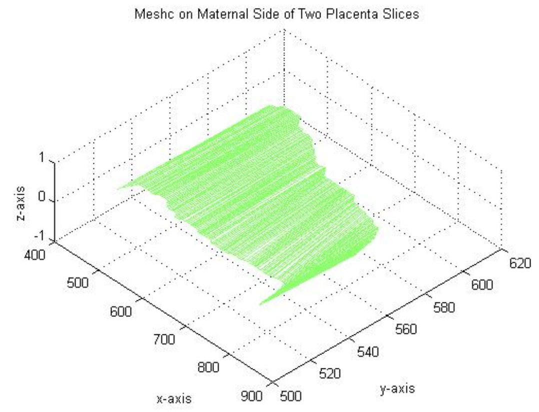

Figure 13: Refined Mesh Plotted with Meshc

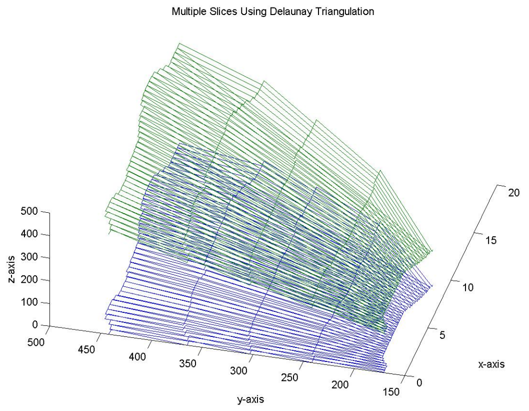

Figure 14: Multiple Slices Triangulated Using Delaunay Function

5 Conclusion

While our group has made significant progress towards our goal, this is a big project, and due to the time constraints, we weren't able to complete the entire project, but a significant part of it; has been realized. Although what we have thus far is half of the placenta constructed, the other half is constructed in the same way. We see the work of the $3-D$ image as a very significant tool in predicting the density of the vessel network that is present in a particular placenta, which we believe has a direct correlation to the health of the baby. Over all, we are excited with our results, and eager to complete the construction of the placenta.

6 Code

```
%The following code is written in Java.
%It's function is to extract data from our image
%This function enable the user to collect data points from any image
%by singly dragging the mouse wherever points are needed.


Import java.awt.Mouse Info;
public class Math579Java  {
        public static void main(String[] args) throws InterruptedException{
                  boolean input = false;
                  System.out.println("A = [");
                  while(linput) {
                            int n = 0;
                           Thread.sleep(5000);]
                           % Here we used loop to collect 201 of data points
                           %for each placenta slice.
                           while (n < 201){
                           Thread.sleep(20);

                           System.out.println(MouseInfo.getPointerInfo().getLocation().x
                           +", "+MouseInfo.getPointerInfo().getLocation().y);
                           n++;
                           }
                           input = true;

                  }

         % here print out the result.
         System.out.println("];");
         }

}
```

```
%Matlab Code
% The following code is written in MatLab.
%This is a Matlab function that preforms interpolation on a set of points
% using least squares trigonometric approximation.
%It takes in a discrete set of points and outputs a real valued function.


function [value_M] = getdata(A,h)


z_11 = A(2,:);
%Here we normalize data points
mag_z_11 = norm(z_11);
z_11 = z_11./mag_z_11;
% Here we divided domain into six subdomain
n_1 = 6;
y= [];
m = n_1/2;
%Here we are just setting the mesh:
z{1} = z_11(1,1:6);

for j = 0:(2*m-1)
     y(j+1) = -pi + (j/m)*pi;
end

% fy is the fuction that is being approximated. or one can put point cloud
% in instead of the function:
 k = 1;
 a = [];
 M = [];
 value_M = {};
 % Here we store the coefficient
 z_1 = z_11(1,1:6);
 for k = 0:n_1
     for i = 1:n_1
         a(i) =(z_1(i)*cos(k*y(i)));
     end
     a = (1/m)*sum(a);
     M(k+1) = a;
 end
 % Here returning value back to the function
 value_M{1} = M;
 % This is a function handle
 f = @(x) M(1,1)/2+ M(1,3)*cos(2*x) + M(1,2)*cos(x);
 x = linspace(0,h, 10);


for i=1:40
    z{i} = z_11(1,1+(i-1)*5: 6+(i-1)*5);
    z_1 = z{i};
    y= [];
    m = n_1/2;
    for j = 0:(2*m-1)
        y(j+1) = -pi + (j/m)*pi;
    end

% fy is the fuction that is being approximated. or one can put point cloud
% in instead of the function:
    a = [];

    for k = 0:n_1
        for v = 1:n_1
            a(v) =(z_1(v)*cos(k*y(v)));
        end
        a = (1/m)*sum(a);
        M(k+1) = a;
    end

    value_M{i} = M;

    f = @(x) value_M{i}(1,1)/2+ value_M{i}(1,3)*cos(2*x) + value_M{i}(1,2)*cos(x);
    x = linspace((i-1)*h,h+(i-1)*h,10);
    plot(x,f(x))
end

end
```

```
%This is the script file
% Here we collect coefficient for each placenta slice in order to use in function handle
for i=1:40
    for k = 1:length(N1)
        for j = 1:7
            x = N1{k};
            NN1(k,j) = x(1,j) ;
        end
    % This is a trigonometric functions
    f1 = @(z) NN1(i,1)/2+ NN1(i,3)*cos(2*z) + NN1(i,2)*cos(z);
    z = linspace((i-1)*h(1),h(1)+(i-1)*h(1),10);

    % here we collect value from function handle
    F1{i}(:,1)=z;
    F1{i}(:,2)=f1(z);

    end
   %This code for  second slice
    for k = 1:length(N2)
        for j = 1:7
            x = N2{k};
            NN2(k,j) = x(1,j) ;
        end

    f2 = @(z) NN2(i,1)/2+ NN2(i,3)*cos(2*z) + NN2(i,2)*cos(z);
    z = linspace((i-1)*h(2),h(2)+(i-1)*h(2),10);

    F2{i}(:,1)=z;
    F2{i}(:,2)=f2(z);

    end
    % This code for third slice
    for k = 1:length(N3)
        for j = 1:7
            x = N3{k};
            NN3(k,j) = x(1,j) ;
        end

    f3 = @(z) NN3(i,1)/2+ NN3(i,3)*cos(2*z) + NN3(i,2)*cos(z);
   z = linspace((i-1)*h(3),h(3)+(i-1)*h(3),10);
    F3{i}(:,1)=z;
    F3{i}(:,2)=f3(z);

    end
    % This code for  fourth slice
```

```
 for k = 1:length(N4)
     for j = 1:7
         x = N4{k};
         NN4(k,j) = x(1,j) ;
     end

 f4 = @(z) NN4(i,1)/2+ NN4(i,3)*cos(2*z) + NN4(i,2)*cos(z);
z = linspace((i-1)*h(4),h(4)+(i-1)*h(4),10);
 F4{i}(:,1)=z;
 F4{i}(:,2)=f4(z);

 end
 % This code for fifth  slice
  for k = 1:length(N5)
     for j = 1:7
         x = N5{k};
         NN5(k,j) = x(1,j) ;
     end

 f5 = @(z) NN5(i,1)/2+ NN5(i,3)*cos(2*z) + NN5(i,2)*cos(z);
 z = linspace((i-1)*h(5),h(5)+(i-1)*h(5),10);
 F5{i}(:,1)=z;
 F5{i}(:,2)=f5(z);

 end
 %This code for sixth slice
 for k = 1:length(N6)
     for j = 1:7
         x = N6{k};
         NN6(k,j) = x(1,j) ;
     end

 f6 = @(z) NN6(i,1)/2+ NN6(i,3)*cos(2*z) + NN6(i,2)*cos(z);
z = linspace((i-1)*h(6),h(6)+(i-1)*h(6),10);
 F6{i}(:,1)=z;
 F6{i}(:,2)=f6(z);

 end
 %This code for seven slice
 for k = 1:length(N7)
     for j = 1:7
         x = N7{k};
         NN7(k,j) = x(1,j) ;
     end
```

```
         f7 = @(z) NN7(i,1)/2+ NN7(i,3)*cos(2*z) + NN7(i,2)*cos(z);
        z = linspace((i-1)*h(7),h(7)+(i-1)*h(7),10);
        F7{i}(:,1)=z;
        F7{i}(:,2)=f7(z);


        end
      axis([0,20,0,0.1,-5,5])

    end
```

References

[1] R. C. Gonzalez, R. E. Woods, S. L. Eddins, *Digital Image Processing*, 2002 Prentice-Hall, Inc. Upper Saddle River, New Jersey 07458

[2] R. L. Burden, J. D. Faires, *Discrete Least Squares Approximation*, 2005 Thomson Brooks/Cole, Bob Pirtle, Belmont, CA 94002

[3] Dr. Carolyn M. Salafia, *Placenta Images and Data* http://www.placentalanalytics.com/index.php?option=com$_c o$ $articleid = 47 Itemid = 63$

[4] *Digital Image Processing*
http://www.aquaphoenix.com/lecture/matlab10/page3.htm

Acknowledgements

We would like to thank Dr. Salafia for providing us with an excellent topic and loads of resources. We also would like to thank Dr. Chang for her patience and helpfulness.

Contact

- Mutaz Alsayegh, email address: malsa001@ucr.edu
- Ramy Heng, email address: hengramy@gmail.com
- Daniel Macias, email address: danmacias13@gmail.com
- Ratana Hor, email address: rattana.hor@gmail.com