Int. J. Math. And Appl., 8(3)(2020), 71–76
ISSN: 2347-1557
Available Online: http://ijmaa.in/

International Journal of *Mathematics And its Applications*

Delaunay Triangulations

Soham Konar[1,*]

1 Thomas Worthington High School, Columbus, Ohio, United States.

Abstract: We examine a methodology to construct 3D maps. We assume the height above sea level is known only at a finite number of points. Mathematically, this problem is equivalent to reconstructing the graph of a function of two variables when the values of the function are only known at a finite number of points. This leads to a discussion of Delaunay triangulations and the algorithm to compute them. We illustrate the concept discussed with examples.

Keywords: Computational Geometry, Delaunay Triangulations, Algorithms.

1. Introduction

Suppose you want to create a digital model of a 3D terrain. It is infeasible to measure the height of the terrain above sea level at every point on the terrain because there would be an infinite number of data points to measure and process. That is why the model can only approximate the terrain based on a finite set of data points. This set of data points will consist of a number of 2D points and the height of the terrain above sea level at each of those points. Plotting this data set would give us an image of multiple points floating in 3D space. To create a model of the terrain in question, we need to draw line segments between these 3D points in such a way that the points contained in the polygons bounded by these segments accurately approximate the actual terrain at those same coordinates. In this article, we will explore the process through which this model is made.

The problem reduces to approximating a function's value at every point in its domain given only a finite set of points from that domain along with their values on the function. To begin, let us look at how this approximation is done in two dimensions. We want to approximate a function in the xy-plane but we are only given a finite set S of points on the function. Let S consist of the points $p_0, p_1, \ldots, p_n$ with their x-coordinates being $x_0, x_1 \ldots, x_n$, respectively, and $x_0 < x_1 < \ldots < x_n$. No two distinct points in S can have the same x-coordinates because otherwise the function would have two different y-coordinates for the same x-coordinate and we would not have a function.

We approximate the function by first splitting the x-axis into intervals such that the intervals' bounds are the x-coordinates of the points in S. The leftmost interval would be $[x_0, x_1]$, the next interval from the left would be $[x_1, x_2]$, and so on until the last interval which is $[x_{n-1}, x_n]$. We cannot approximate the function outside of the interval $[x_0, x_n]$ because we cannot extrapolate the function given only the set of points S. Once we have split the x-axis into intervals, we can then draw line segments between the endpoints of each interval which will approximate the section of the curve within that interval. Now

* *E-mail: sohamkonar28@gmail.com*

that we have seen how to approximate a 2D function given a finite set of points taken from the function, we will now move on to our main goal: approximating a 3D function given a finite set of points taken from the function. Just as how we split the 2D function into sections or regions that were easy to approximate, we will do the same for the 3D function. In this case, we will have triangles in 3D space play the role that the line segments did for the 2D function. We choose triangles because every polygon can be split into triangles and that allows for more flexibility in the model and so more accurate approximations on the model. To find these triangles, we must triangulate the set of points in two dimensions and then raise each vertex of the triangles to their corresponding heights. Delaunay triangulations belongs to the field computational geometry. We refer the reader interested to learn more about the topic to [1, 2].

2. Comparing Triangulations

Now, we will look at different triangulations of the same set of points and see how one triangulation may be more optimal or accurate than the other. The two triangulations we will be examining will be the same except one edge in the first triangulation will be flipped in the second triangulation as on the next page.

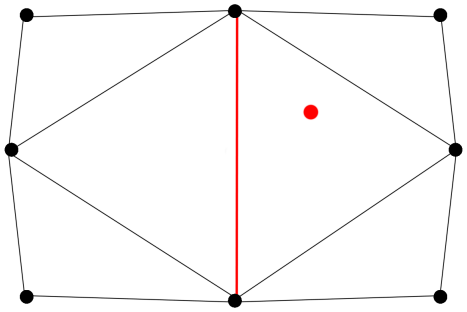

Figure 1: Triangulation 1

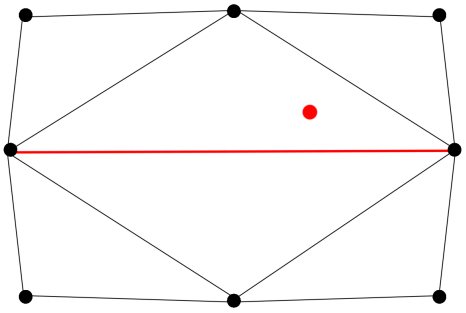

Figure 2: Triangulation 2

In the two figures above, each triangulation has a pair of unique triangles that differ by the red edge that has been flipped in Figure 2. We will determine which triangulation is more optimal by analyzing how a point in each triangulation is approximated by the triangle that contains the point.

Looking at the red point that is at the exact same position in both triangulations, we can see that the triangles that contain the red dot in each triangulation (those triangles in Figures 1 and 2 will be called Triangles 1 and 2, respectively) share two vertices. However, the third vertex of Triangle 2 is much farther from the red dot than the third vertex of Triangle 1. This means that the value of the red dot would be better approximated by the vertices of Triangle 1 and so Figure 1 would be more optimal. This means that we don't want long and skinny triangles whose vertices would be far away from points in the triangle and so we want to make the angles of the triangles as large as possible. Comparing the six angles in each pair of triangles, we see that the minimum angle in Figure 1's pair is greater than the minimum angle in Figure 2's pair and this aligns with our claim that Figure 1 is more optimal.

Now that we have seen how flipping an edge can make a triangulation more optimal, we can say that an edge in a triangulation is illegal if flipping it makes the triangulation less optimal. Because there are a finite number of triangulations of a set of points, you will always be able to obtain a most optimal triangulation because flipping an illegal edge in a triangulation can only make the triangulation more optimal. This most optimal triangulation has the largest minimum angle out of all of the other triangulations and it is called the Delaunay triangulation.

3. Algorithm

Now that we know how a Delaunay triangulation is defined, we can look at the algorithm that is used to construct them. Let the set of points that we are triangulating be $S = \{p_0, p_1, p_2, \ldots, p_n\}$ where the order of the points is completely randomized. Essentially, the algorithm consists of sequentially taking each point of S, finding which triangle in the current triangulation contains this new point (we will call the new point p_r and the triangle containing p_r $p_ip_jp_k$ which has vertices p_i, p_j, and p_k), and then drawing edges from p_r to p_i, p_j, and p_k. Whenever p_r is added, we need to check that the edges of $p_ip_jp_k$ are still legal. Notice that if one of those edges are flipped, one of the new edge's vertices will be p_r. Whenever an edge is flipped, we need to make sure that any other edges don't become illegal and so we need to recursively check the other 4 edges of the quadrilateral in which the edge is being flipped until all potentially illegal edges are determined to be legal.
We stated earlier that we need when we take a point p_r, we need to find out which triangle p_r is contained in. However, at the beginning there are no triangles and there is also no guarantee that p_r is in the current triangulation at all. That is why we need two special points p_{-1} and p_{-2} such that p_{-1}, p_{-2}, and the point in S that has the largest y-coordinate form a triangle that contains all of the points in S. If we start the triangulation with that large triangle, p_r will always be in a triangle of the triangulation and so we can can proceed with the procedure that was outlined earlier. At the end, we can erase p_{-1}, p_{-2}, and all of the edges to which either p_{-1} or p_{-2} belong to to get our final triangulation.

4. Application and Analysis

Now, we will put everything we have discussed to use by examining a 3D function in five different ways:

- an accurate 3D plotting of the function
- a model formed from a Delaunay triangulation of a set of deliberately chosen points on the function
- a model formed from a non-Delaunay triangulation of the same set of deliberately chosen points on the function
- a model formed from a Delaunay triangulation of a set of randomly chosen points on the function
- a model formed from a non-Delaunay triangulation of the same set of deliberately chosen points on the function

For simplicity's sake, we will be examining the function $z = e^{-x^2}$. Figure 3 below is the function plotted in 3D.

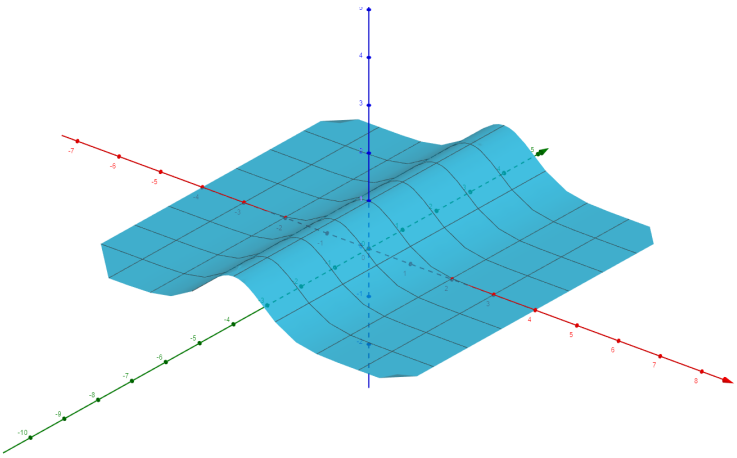

Figure 3: Accurate 3D plotting

Now, we will look how the function will be modeled when formed from different triangulations of a set of deliberately chosen points on the function.

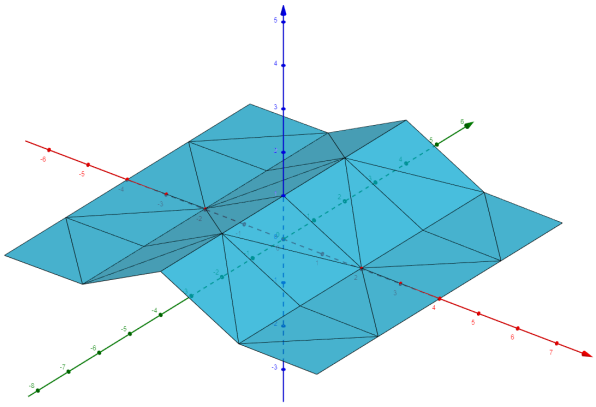

Figure 4: Deliberately chosen points, Delaunay triangulation

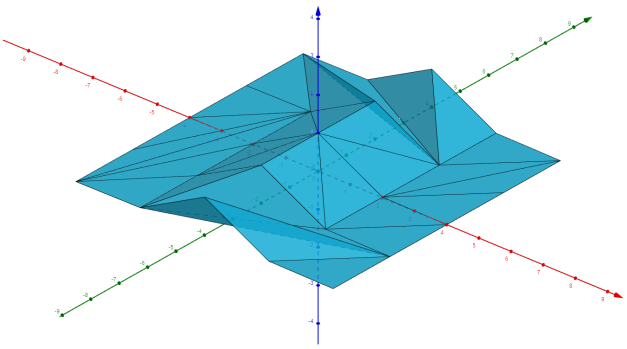

Figure 5: Deliberately chosen points, non-Delaunay triangulation

Finally, we will look how the function will be modeled when formed from different triangulations of a set of randomly chosen points on the function.

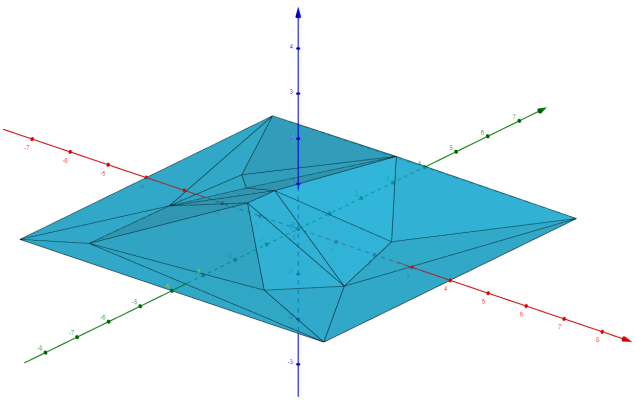

Figure 6: Randomly chosen points, Delaunay triangulation

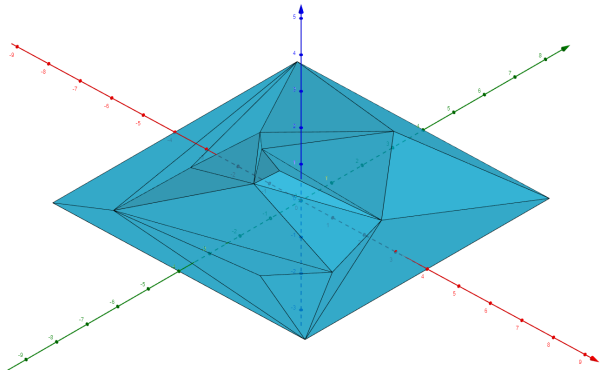

Figure 7: Randomly chosen points, non-Delaunay triangulation

Looking at Figures 3 to 7, we can see that the four models in Figures 4 to 7 are all relatively accurate models of the original function $z = e^{-x^2}$. However, the problem with the models formed from the non-Delaunay triangulations is that at some points with x close to 0, the model would have a z-value of 0 even though the z-value of the corresponding point on the actual function was close to 1. While the problem with $x = 0$ we previously described is specific to the task of modeling $z = e^{-x^2}$, it can be generalized to all models formed in this way, whether it is a 3D function of a real-world terrain being modeled. Any time an edge does not follow the natural lines of the function or terrain is being modeled, the chance that points on and near the edge will not accurately approximate the corresponding points on the function or terrain increases. However, this does not mean that a Delaunay triangulation will always yield the best model. For example, there may be terrain that is smooth and continuous except for a wedge that was taken out due to a natural disaster. Then, we would want the edge in that area to be perpendicular to what would be the natural line of the terrain. This shows that while Delaunay triangulations are effective in modeling objects with continuity, irregularities in the object may need to be accounted for by humans. This can be done in three ways:

- human alteration of the model's edges, which we want to avoid because it may not be feasible for large data sets
- collection of more data points, which we might also want to avoid because we do not want to use more data points than necessary to model the object
- a better selection of data points, which we also want to avoid for the same reasons as for human alteration of the model's edges. However, this leads to our last discussion point.

Looking at the figures above, we can see that in general the models with deliberately chosen points more accurately modeled the function than the models with randomly chosen points. That is because humans can process a near infinite number of data points at once while computers can only process one at time and so it is much easier for humans to select data points that follow the lines and curves of the object being modeled. However, selecting a large number of data points is much more efficient when done randomly by a computer instead of by hand by a human. This highlights that the trade off for having an optimal a data set and an accurate model is efficiency in terms of time and storage and this makes sense because the more details one wants a model to have, the more time and effort is needed to create the model. Thus, a model formed from a Delaunay triangulation of a set of deliberately picked data points will be the most accurate but will require the most time and effort to construct.

5. Conclusion

Now that we have reached the end of our discussion, let us recap everything we have discussed thus far. We started with the goal of finding a way to efficiently and accurately model a 3D terrain. To do this, we first discussed how this is done with in 2D with line segments estimating sections of the function. We took the method we used for 2D functions and translated it to a method for 3D functions. Because our set of data points can be triangulated in so many ways, we needed to find an algorithm that helps us determine the triangulation that is most optimal for our purpose, which is called the Delaunay triangulation. Using this algorithm, we made multiple models of the function $z = e^{-x^2}$ and varied whether a Delaunay triangulation is used or not and if the data set is deliberately chosen or not. Finally, we examined the differences between the models and their accuracy in modeling the original function and so we came to the conclusion that the Delaunay triangulation of a set of deliberately picked data points yields the best results for the problem we were trying to solve.

Acknowledgements

I would like to thank my advisor Professor Guillermo Goldsztein for all the help and guidance he has given me in learning this concept as well as writing this paper. I could not have done it without him, so thank you Professor Goldsztein.
The textbook from which I learned this concept was Computational Geometry: Algorithms and Applications, Third Edition by Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. Figures 1 and 2 were made with the mobile application Autodesk Sketchbook and Figures 3 to 7 were made with the browser application Geogebra: 3D Calculator.

References

[1] Mark De Berg, Marc Van Kreveld, Mark Overmars and Otfried Schwarzkopf, *Computational Geometry: Algorithms and Applications*, Springer, (1997).

[2] Franco P Preparata and Michael I Shamos, *Computational geometry: an introduction*, Springer-Verlag, New York, (2012).