

Nicholas Shorter

PID: N0425664

The development of 3-D Triangulated Irregular
Networks of LiDAR data as a preprocessing technique
for clustering of LiDAR point cloud data

Table of Contents

1. Introduction – How and Why TIN fits into LiDAR 3-D modeling	3
2. Triangulated Irregular Networks.....	3
3. Problem Definition.....	4
3.1. Algorithm Features	4
3.2. Triangulation Rules.....	5
4. Selected Triangulation Algorithm Process – Sequential Greedy Insertion	5
5. Data Structures for Implemented Algorithm	6
6. Sequential Greedy Insertion Steps	7
Appendix.....	8
1. Appendix 1 - Figures	9
2. Appendix 2 – Tables	17
3. Appendix 3 – Equations.....	18
4. Appendix 4 – Supplementary Discussion	21
4.1 Derivation of calculations used to obtain planar coefficients and triangle normal vector.....	21
4.2 Derivation of error function	21
4.3 Delaunay Triangulation Circle Test.....	23
4.4 Derivation of <code>edg_RL</code> function.....	24
5. Appendix 5 – Reference List	26

1. Introduction – How and Why TIN fits into LiDAR 3-D modeling

Currently there exist two methodologies for scene reconstruction from LiDAR data: model based approaches and data driven approaches. Model based approaches attempt to shape pre-existing models, by modifying parameters of those models, to fit the point cloud data [10]. Problems with this method arise when a given building depicted by the point cloud data is structured such that a pre-existing model does not exist for it. Another issue with model based methods is when the combination of several models is necessary to approximate shapes in which no single model can fathom, arising in additional computation and complexity. Data driven approaches typically group like points together and then derive a model to approximate those points that yields minimum error from the original points. This approach approximates segmented areas with planes and then merges those planes to form three-dimensional shapes depicting the captured LiDAR scene.

2. Triangulated Irregular Networks

Several data driven LiDAR model 3-D reconstruction algorithms currently exist in the literature ([1],[2],[3],[4]) that utilize triangulated irregular networks (TINs) to construct model approximations of depicted urban and residential scenes. Triangulated irregular networks are a 3-dimensional depiction of LiDAR point cloud data with a series of connected, non-overlapping triangles which have no intersecting edges. Three methods of utilizing the TIN structure to extract information from LiDAR point cloud data exist as follows: (1) clustering approach; (2) least squares approximation approach; (3) TIN region growing algorithm approach.

The following methods use the least squares approximation in conjunction with the TIN region growing to extract 3-D features from the point cloud data. Morgan and Habib in [1] use a region growing TIN algorithm, based on the least-squares adjustment, to extract building facades from the transformed point cloud data (transformed to the triangulated feature space). Chen et. al., in [3], also use a region growing TIN algorithm, considering both the height difference between triangles and the angle difference between normal vectors of neighboring triangles for merging criterion for planar approximation.

Hoffman however uses the clustering approach in [2] to group together triangles in the TIN that contain similar properties. In [2], the position of each triangle is mapped out in spherical coordinates which are the dimensions of the triangles that are clustered.

Lattuada et. al. in [4] detail several advantages for describing a geo-model with a three dimensional triangulation. These details have been enumerated in [Table 1]. The Delaunay triangulation forms triangles such that all vertices belong to the triangle form a circle that does not enclose any points. For only 2 dimensions, the Delaunay triangulation method, given four points, chooses the diagonal that splits the quadrilateral formed by the four points into two triangles. These triangles are such that the lesser of their internal angles are maximized. However, as already mentioned, this property of Delaunay triangulation, as proved by [6], only holds in 2 dimensions. It is possible to produce a Delaunay triangulation for a 3-dimensional data set, the z-coordinate is simply ignored. Methods that consider the z-coordinate for triangulation are referred to as data dependent triangulation methods.

Wang et. al. in [5] compare the Delaunay triangulation process against several other triangulation processes when approximating two different terrains from Digital Elevation Models (DEMs). Several conclusions derived from the paper are presented. The quality of the generated TIN is dependent on both the vertex placement and connection. Processes that iteratively select points during triangulation grossly outperform processes that separate the point selection and triangulation procedures. While TIN generation from separate procedures is comprehended and implemented with ease, the separation of the procedures suffers from the following drawbacks. Point selection via filters is very sensitive to data errors and surface variations. Point selection is a static process (as opposed to the dynamic adaptive process implemented in greedy insertion). When a point is chosen and inserted, the configuration of the TIN is modified and therefore the importance of the remaining

unused points changes. All computational efforts executed to find the surface specific points are not utilized in the final construction of the TIN and are thus wasted. Therefore the implementation of an algorithm integrating point selection and triangulation as a unified procedure, while being more complex than algorithms that separate the procedures, is preferred due to the increase in performance and the ability of this preferred methodology to overcome the aforementioned drawbacks.

Among all of the triangulation methods tested, Wang et. al. found the sequential greedy insertion algorithm performed the best in terms of accuracy. While the sequential greedy insertion algorithm is briefly described in [5], a more detailed version of the algorithm's description is presented in [9]. Although in [5] the greedy sequential algorithm is tested on a DEM, the algorithm can be easily modified to work with irregular point spacings instead. Later described in the problem definition statement, this will be important as the points to be triangulated exist as irregular point spacings.

In [5], Wang et. al. elaborate on the usefulness of TINs, explaining that the variable resolution and high capability of capturing significant terrain features makes TINs attractive modeling techniques for surface reconstruction and representation. Two fundamental rules for triangles constructed from a TIN exist as follows: (1) triangle edges do not intersect one another; (2) triangles cannot overlap each other.

A plethora of triangulation methods for range images exist. An image is defined as a two dimensional array containing intensity values and often is represented as a matrix. Geological information system (GIS) users typically prefer to interpolate the irregularly spaced raw LiDAR points, producing a digital elevation model (DEM or sometimes referred to as a range image), and operate on the DEM with conventional image processing algorithms. With the point spacings existent as a 2-dimensional array of values, it is possible to operate on the array or matrix with image processing kernel functions. However, critics of these interpolated range images or DEMs argue they are an aberration which over-simplifies terrain modeling [8]. Obviously working with the raw LiDAR data and generating a TIN from it instead of working with the interpolated data will yield more accuracy. It is important to ensure that this increase in accuracy is worth the complexity associated with the irregular spacing of the raw data and the inability to use the conventional image processing algorithms existent for regular point spacings.

One topic, typically raised during TIN discussions, is the architecture of the data structure used to encode the TIN. In [7], Kidner et. al. argue that ultimately for each particular type of application there exists an optimal data structure. Therefore, no singular data structure can be optimal for all applications. It is therefore necessary to define and model a chosen data structure architecture based on a formulated problem definition. One feasible data structure, containing a matrix of values, approximating triangles in a given configuration is presented in [Figure 1].

3. Problem Definition

3.1. Algorithm Features

Many of the application specific needs will ultimately determine the nature of the triangulation algorithm chosen. The definition of these needs will therefore reduce the number of algorithms that will be choice for the problem at hand. In order to retain the most amount of information and accuracy as possible, it is imperative that the TIN is derived from the raw LiDAR point cloud data. The selected triangulation algorithm therefore must have significant accuracy in approximating the raw LiDAR point cloud data with the implemented TIN. The triangles in the TIN are to be clustered by a clustering algorithm. The dimensions of the triangles that are of importance to the clustering algorithm are as follows: the triangles' vertices, their centers, and the normal angle to their defined surface. These dimensions therefore must be incorporated in the data structure encoding the resulting TIN. In later experimentation, it would be nice to have the TIN generation algorithm produce variable resolution TINs based on a user defined parameter. If the TIN generation algorithm can reduce the number of

triangles existent within the TIN with minimal loss in representation accuracy of the original raw LiDAR point spacings, this will ultimately yield lower computation times for the subsequent clustering algorithm.

3.2. Triangulation Rules

In order to design or select an algorithm, the necessary rules for the desired triangulation must be specified:

1. No intersecting triangle edges are to exist within the TIN.
2. Furthermore, no overlapping triangles are to exist within the TIN.
3. No gaps are permissible within the TIN.
4. When considering a point for the formation of a triangle, the neighboring points closest to the point in consideration must have the highest favored potential for triangle formation
5. As a result from rules 1 and 2, from a top down, normal view, all triangles must be visible. Therefore, the formation of triangles in 3-dimensional space, surfacing over triangles underneath, is prohibited.

4. Selected Triangulation Algorithm Process – Sequential Greedy Insertion

The algorithm selected to realize the triangulation of the irregular point spacings in the provided LiDAR data is Garland and Heckbert's sequential greedy insertion algorithm. In [9], Garland and Heckbert present both the sequential and greedy insertion algorithms. The version of the greedy insertion algorithm which only inserts a single point in each pass is called sequential greedy insertion, while the version of the algorithm in which inserts multiple points in each pass is called parallel greedy insertion. While the parallel version does cut down execution time, the savings realized come at the cost of the algorithm's performance in terms of accuracy; which is why the sequential version is selected.

4.1 Algorithm Initial Triangulation

The sequential greedy insertion algorithm realizes two adaptive optimization cost functions: (1) local Delaunay triangulation; (2) global point insertion. The algorithm starts by considering the quadrilateral formed by the outermost four points in terms of x and y or longitude and latitude spacing. Then an arbitrary triangulation is formed (two triangles are randomly formed from the 4 points). That formation is then checked to see if flipping the diagonal will optimize the arbitrarily formed configuration to conform to Delaunay triangulation.

4.2 Greedy Insertion

Error Derivation

For all triangles, the distances between the triangles (planes) [Figure 3] and the points that they encompass (in x and y or longitude and latitude spacing) are calculated. A detailed analysis showing the derivation of the calculations necessary to derive the planar coefficients describing the plane formed by the three vertices of a given triangle is presented in section 4.1 of the appendix. After all of the distances between the unused points and the existing triangulated surface are calculated, for each triangle, the unused point furthest from that triangle is cached into that triangle's data structure.

Point Insertion

All of the triangle data structures are considered, the point having the greatest distance from the TIN (labeled the candidate point) is the point inserted next (hence the name greedy insertion). Three cases can occur when inserting a given point: (1) the candidate point is inserted inside a triangle; (2) the candidate point is inserted on a triangle edge; and (3) the candidate point is inserted at the edge of the outermost initial quadrilateral.

The first point insertion case results in the formation of three triangles. The point is inserted and three lines are drawn from the point to the vertices of the encompassing triangle. This scenario is depicted in [Figure 8]. For the second point insertion case, the candidate point is inserted at the edge of the TIN resulting in the formation of 2 new triangles, as depicted in [Figure 9]. In the third point insertion case, the candidate point is inserted

along the edge of a triangle. The algorithm is designed to delete the edge and then connect lines from the candidate point to the vertices of the two triangles which share the common edge in which the candidate point was inserted along. The third point insertion scenario is depicted in [\[Figure 10\]](#).

Delaunay Triangulation

After the insertion of the points, the edges of the triangles are checked for flipping. The edges are flipped to form a new diagonal if the flipping maximizes the lesser of the interior angles of the triangles (Delaunay triangulation). If for two given triangles, their edges are flipped, then all of the adjacent triangles to those triangles are then checked to see if edge flipping should be done with triangles adjacent to them. This process continues until it is determined that no adjacent triangle will further optimize the TIN via diagonal flipping in accordance to Delaunay triangulation. This local optimization procedure is implemented to combat the formation of slivers. A sliver is qualitatively defined as a triangle whose largest angle is ‘close’ to 180 degrees. Typically the triangle ‘A’ depicted in [\[Figure 11\]](#) is desired over triangle ‘B’. A detailed analysis of this procedure is further described in section 4.3 of the appendix.

All of the above procedures are depicted in the block diagram contained in [\[Figure 13\]](#).

Several steps are necessary to implement a triangulation algorithm capable of realizing the requirements enumerated in the problem definition section. The data structure needed to realize the aforementioned requirements in the problem definition and the necessary steps of the algorithm are detailed in the sections that follow.

5. Data Structures for Implemented Algorithm

For each triangle, the following information is recorded in a data structure:

1. The three vertices composing the triangle - $P_1(x_1, y_1, z_1); P_2(x_2, y_2, z_2); P_3(x_3, y_3, z_3)$
2. The planar equation describing the triangle – [\[Equation 4\]](#)
3. The normal vector to the plane formed by the triangle – [\[Equation 16\]](#)
4. The point contained within the triangle with the greatest height deviation from the triangulated surface and the point
5. A triangle index number (a counter, incremented for all newly generated triangles, assigns a number to uniquely index all triangles)
6. The indices of the vertices in relation to their original position in the point input array.

For each point, the following information is recorded:

1. x, y, and z coordinates
2. a triangle index (not if this number = 0, the bit is unassigned) indicating which triangle the points are contained within

6. Sequential Greedy Insertion Steps

6.1. Step 1 - Initial Triangulation

- 6.1.1. Step 1a – Select the 4 outermost corner points of the LiDAR data (some points may be artificially created)
- 6.1.2. Step 1b – Perform Delaunay triangulation of selected 4 points (2 triangles formed) swapping the edges to obtain the optimal mesh
- 6.1.3. Step 1c – Mark the 4 points as used
- 6.1.4. Step 1d – For each of the two triangles formed, calculate the distance between the unused points and the plane formed by the triangle encompassing those unused points in x and y dimensions
 - 6.1.4.1. Step 1d (i) – Cache the candidate point (point farthest away from triangle in z-direction) for each triangle formed

6.2. Step 2 - Largest Deviation Point Insertion

- 6.2.1. Step 2a - Select the candidate point (the point with largest deviation from triangulated mesh). Note: if this is the first iteration of the algorithm, all errors must be calculated as none are cached

- 6.2.2. Step 2b – Insert the Point into the Triangulated Mesh (mark it as used)

6.3. Step 3 – Locate and Flip if Necessary

- 6.3.1. Step 3a – Locate the triangle within the triangulated mesh containing the recent inserted point
- 6.3.2. Step 3b – Split the located triangle into the necessary triangles containing the inserted point (based on the condition of insertion – [\[Figure 8\]](#), [\[Figure 9\]](#), or [\[Figure 10\]](#))
- 6.3.3. Step 3c - Remove the original triangle (triangles are not allowed to overlap one another)
- 6.3.4. Step 3d – Recursively check each of the outer edges of the triangle containing the inserted point to see if flipping the edges will further optimize the existing triangulated mesh
- 6.3.5. If a triangle edge is flipped, check the edges of both of those triangles and see if their adjacent triangle diagonals should be flipped (repeat until flipping will no longer further optimize the TIN according to local cost function)

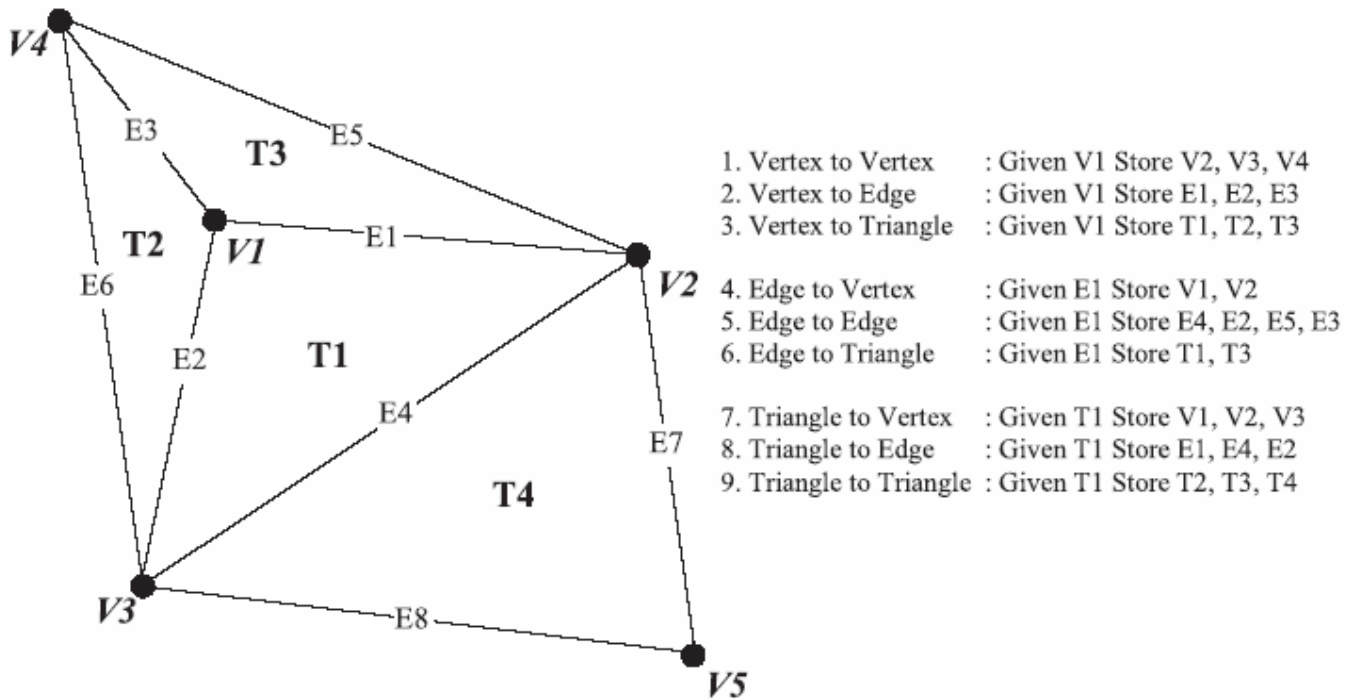
6.4. Step 4– In the regions affected by insertion and flipping, recalculate the following parameters

- 6.4.1. Step 4a - The plane equations associated with the modified triangulations
 - 6.4.2. Step 4b – Locate the triangles containing the unused points
 - 6.4.3. Step 4c – Calculate the error between the unused point and the triangulated surface
 - 6.4.4. Step 4d – For each triangle record the candidate value for the unused points (point with largest error deviation)
- 6.5. Step 5 - Return to step 2 and repeat if point budget or error approximation has not been met
 - 6.6. If convergence in Step 5 was realized, finish inserting points and remove all triangles associated with artificial points (effectively removing those points from the triangulation)

Appendix

1. Appendix 1 - Figures

Figure 1 – Potential Data Structure for Encoding TIN



Triangles	Vertices (i.e. TV Relation)	Adjacent Triangles (i.e. TT Relation)
T1	V1, V2, V3	T3, T4, T2
T2	V4, V1, V3	T3, T1, Null
T3	V1, V4, V2	T2, Null, T1
T4	V3, V2, V5	T1, Null, Null

Figure 2 – Projection of a onto b

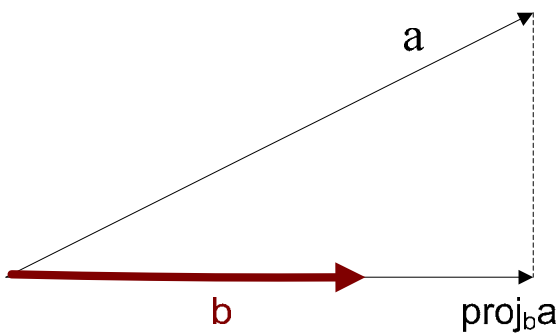


Figure 3 – Distance between a given Point and Plane

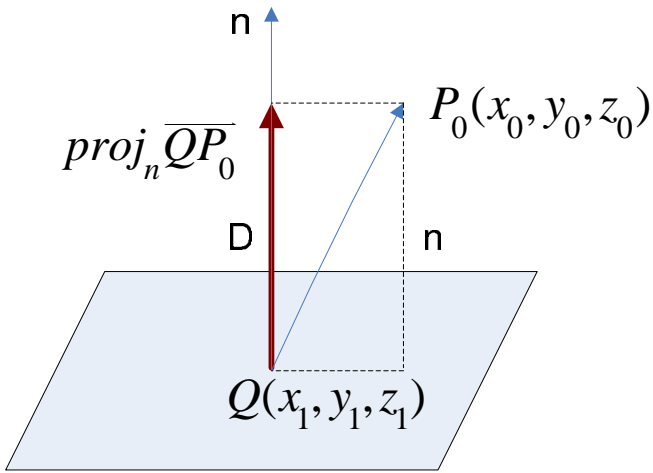


Figure 4 – Test of Point to the left or right of line/edge

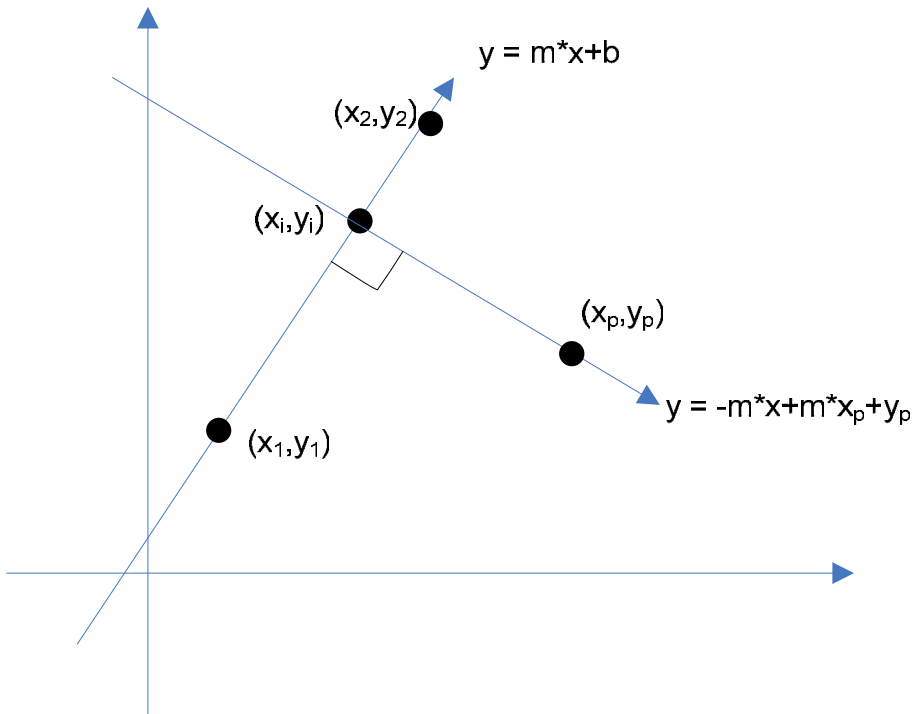


Figure 5 – Function Hierarchy

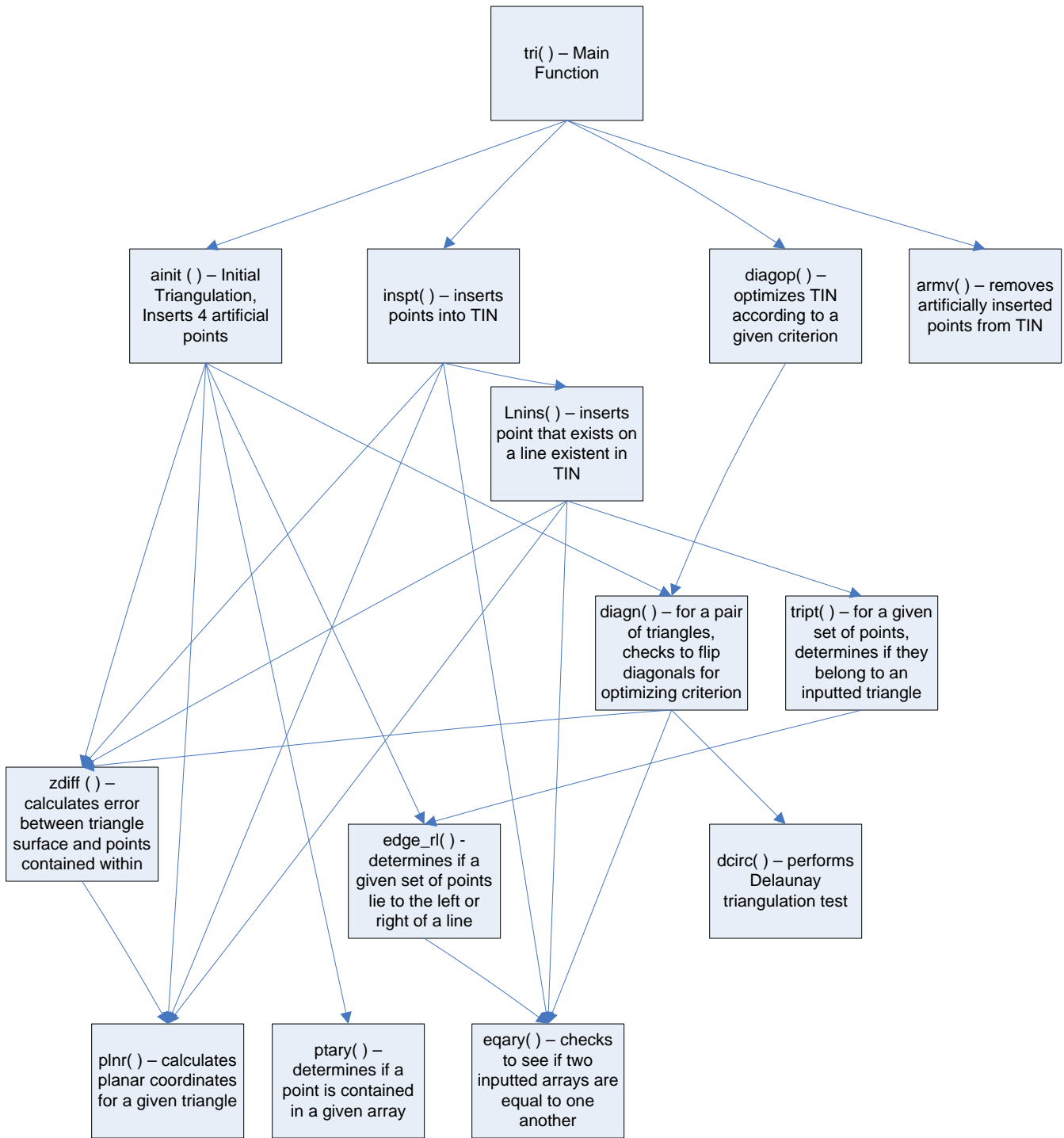


Figure 6 – Delaunay Triangulation Circle Test

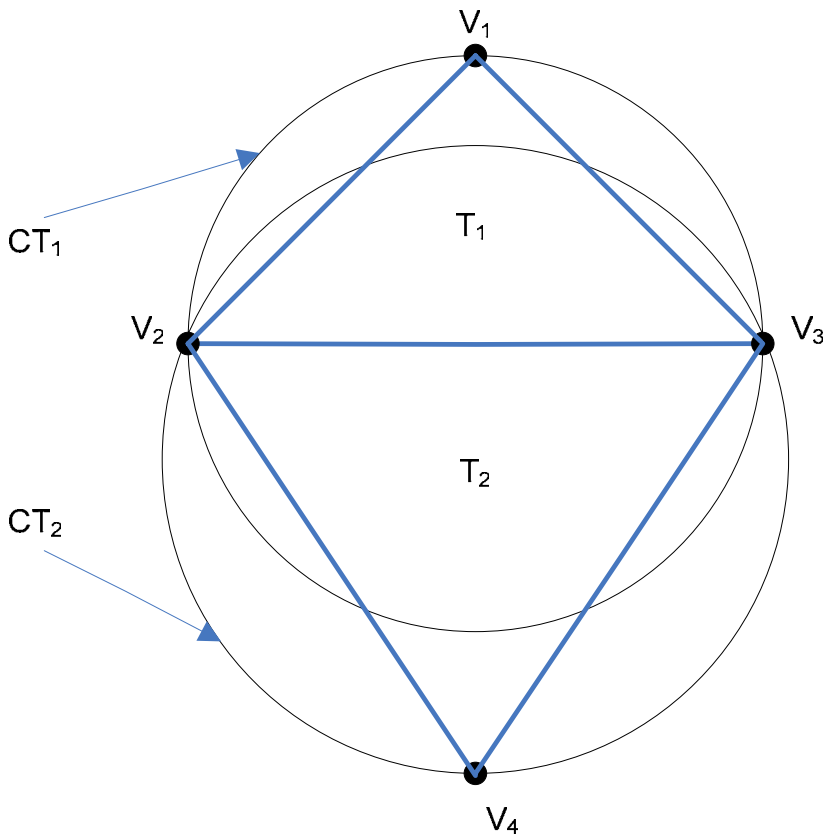


Figure 7 – Normal Planar Vector Derived from Three Points

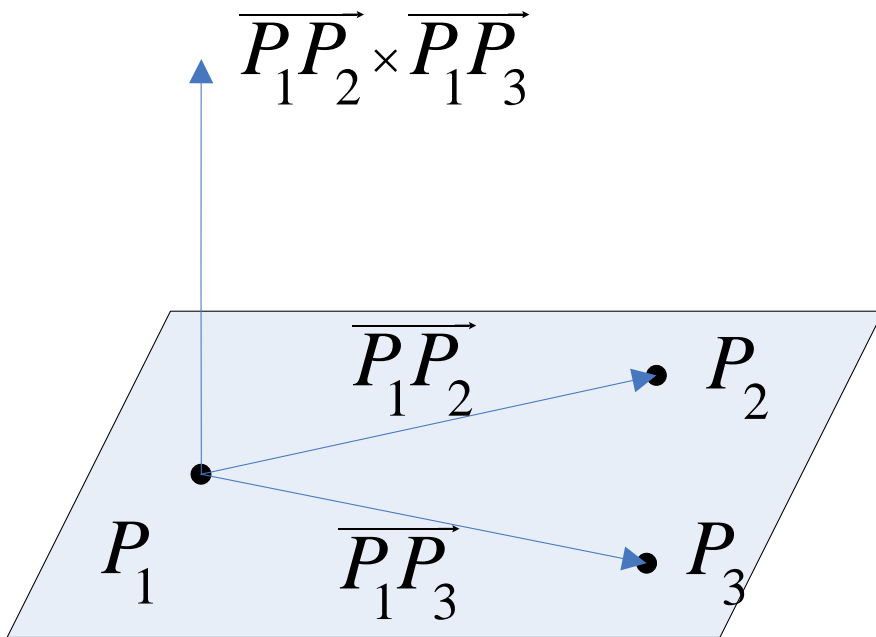


Figure 8 – Point Insertion (Case 1)

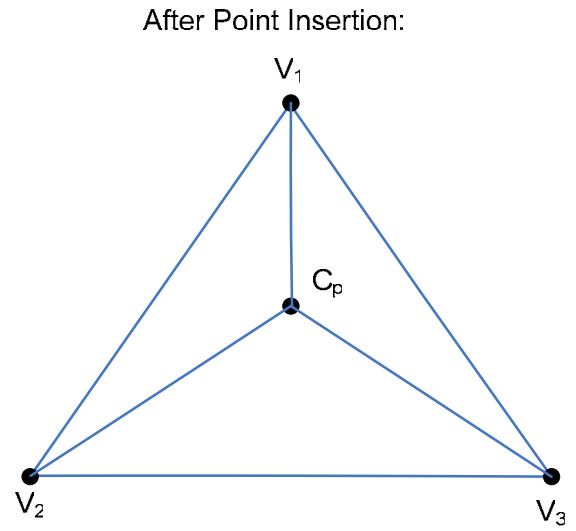
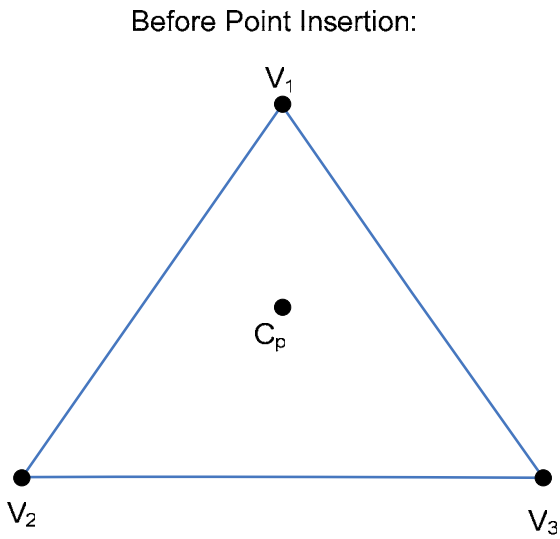


Figure 9 – Point Insertion (Case 2)

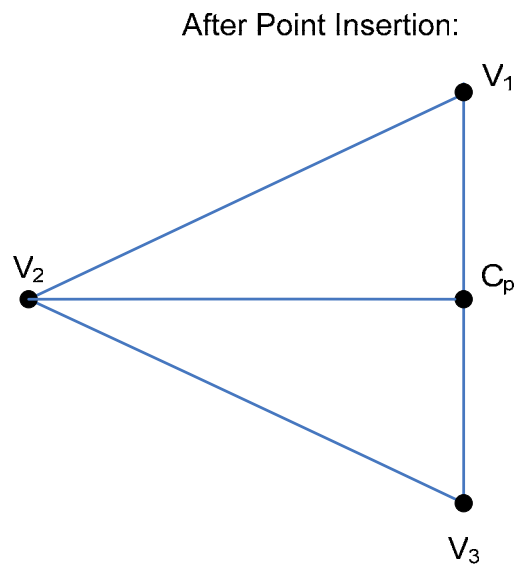
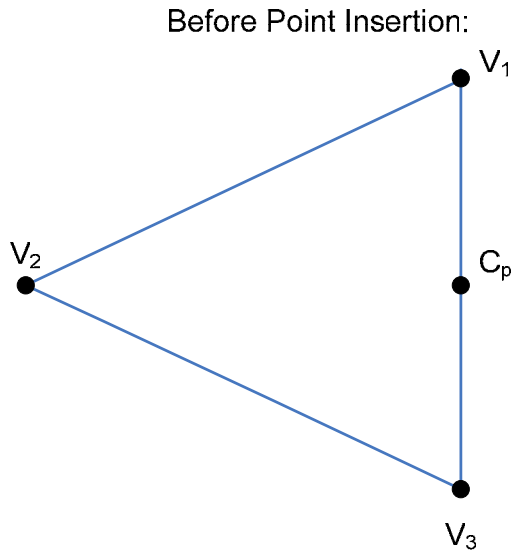


Figure 10 – Point Insertion (Case 3)

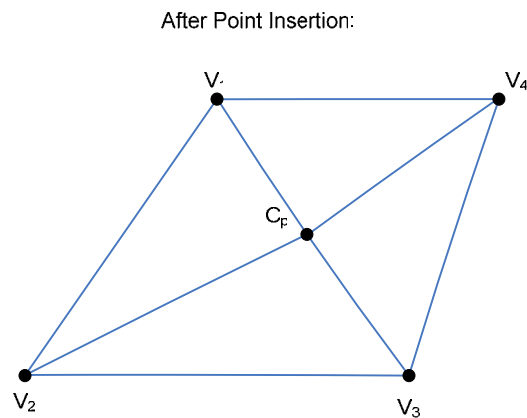
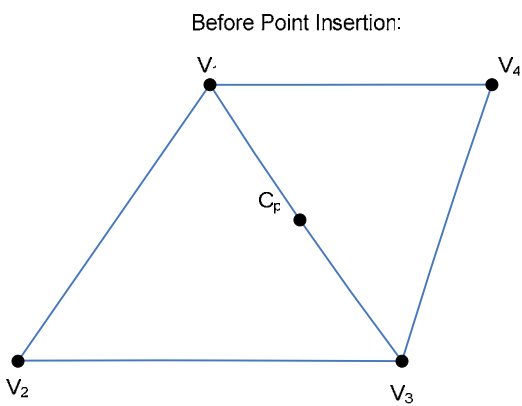


Figure 11 – Sliver Example

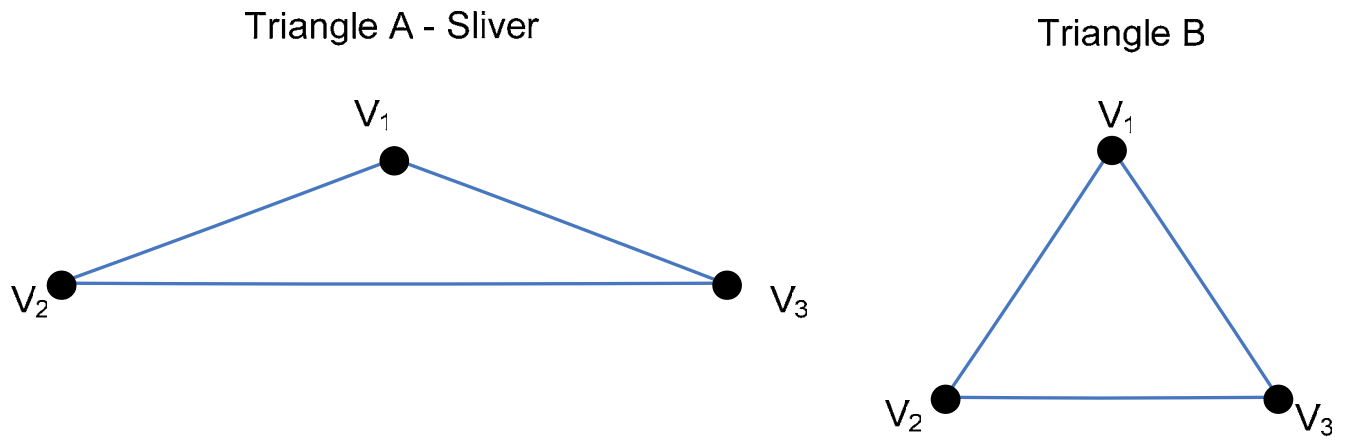


Figure 12 – Circle Center Calculation

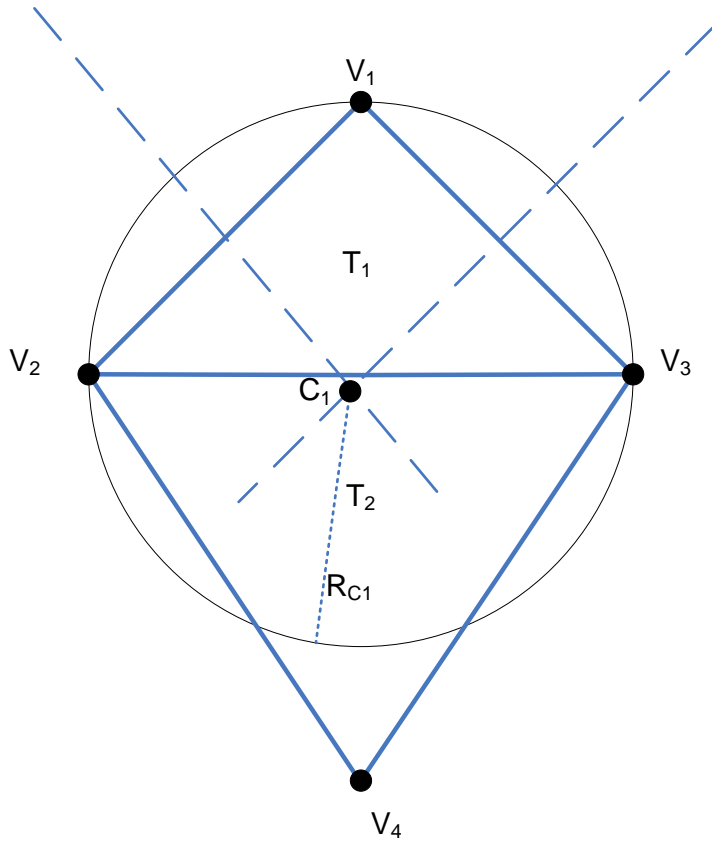


Figure 13 – Greedy Insertion Block Diagram

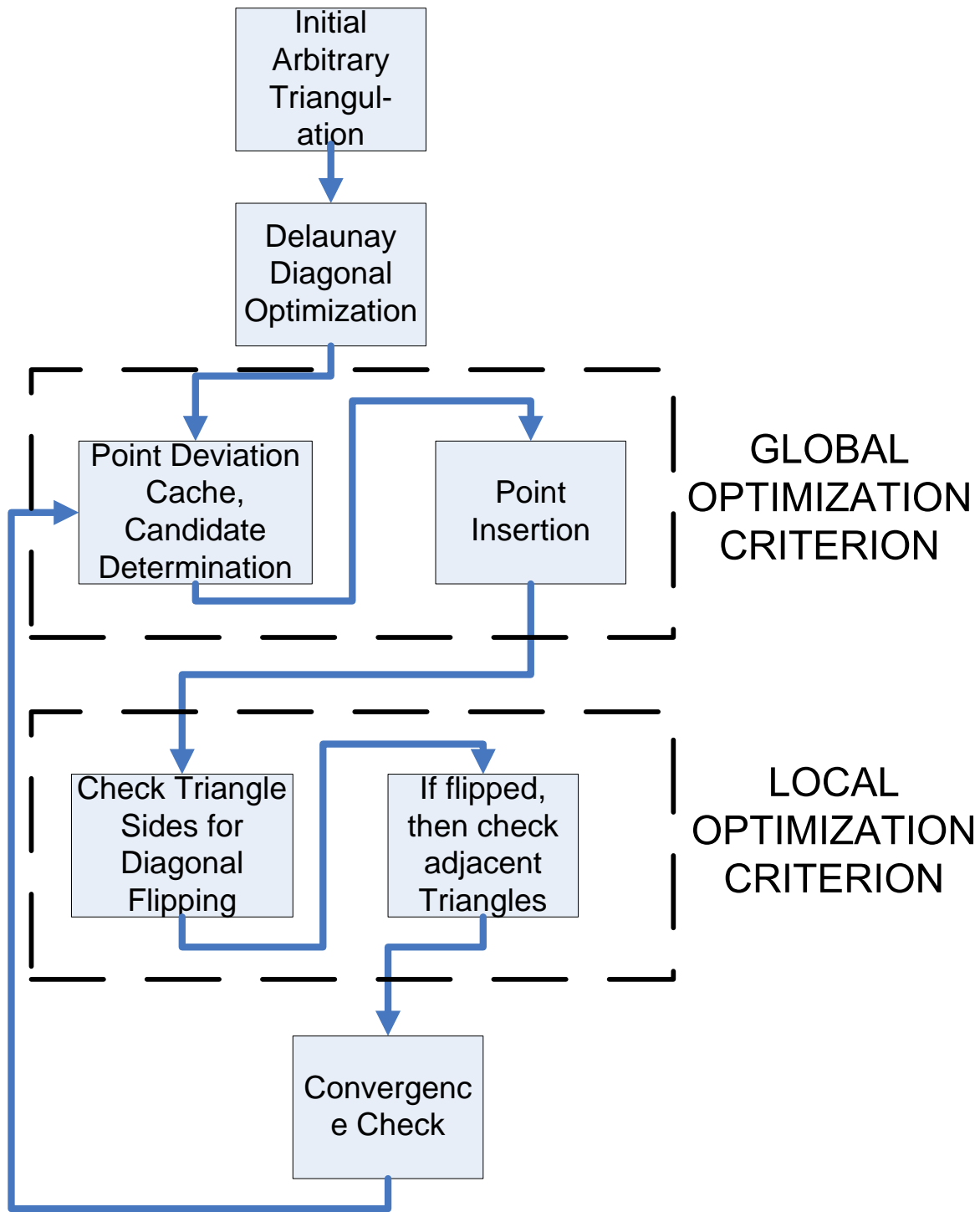
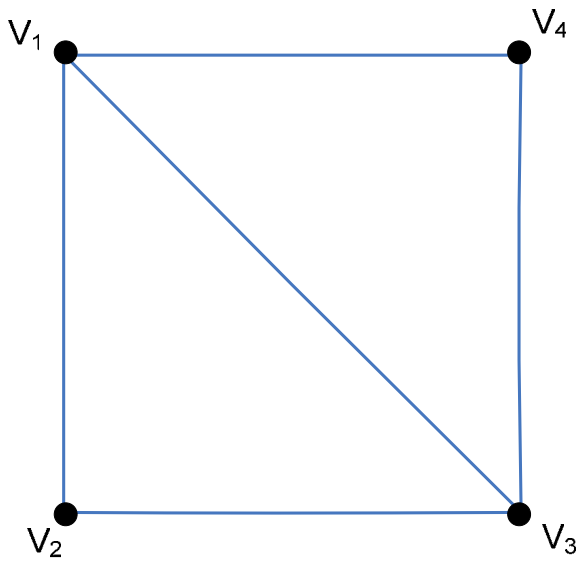


Figure 14 – Initial Triangulation



$$V_4 = \langle \max(x_i), \max(y_i) \rangle$$

$$V_1 = \langle \min(x_i), \max(y_i) \rangle$$

$$V_2 = \langle \min(x_i), \min(y_i) \rangle$$

$$V_3 = \langle \max(x_i), \min(y_i) \rangle$$

2. Appendix 2 – Tables

Table 1

3-D Triangulated Irregular Networks Advantages	
1	the generation algorithm is fully automatic and therefore objective
2	space is uniquely defined and cells are spatially indexed
3	size of elements can be adjusted locally as a function of the complexity of the model
4	the model can easily be edited manually
5	topology is derived from neighborhood relationships
6	constrained triangulation means we can use vectors or surface constrains (i.e. to represent trends)
7	use of triangular elements is the perfect choice for visualization since this is the basis for rendering techniques
8	good accuracy and approximation compared to block models
9	integral properties are efficient and easy to calculate
10	we can easily extract from the 3D solid representation of an object the 3D triangulated surface which is its boundary
11	spatial searches and relational queries are easy to implement
12	good performance of Boolean operations

3. Appendix 3 – Equations

Proving distance between a given point and plane

Equation	Description
$proj_a b = (a \cdot b)b$	Equation 1 – Projection of vector b onto a
$proj_a b = \left(a \cdot \frac{b}{\ b\ } \right) \left(\frac{b}{\ b\ } \right)$	Equation 2 – Normalization of b vectors
$proj_a b = \frac{a \cdot b}{\ b\ ^2} b$	Equation 3 – Projection of vector b onto a
$a \cdot x + b \cdot y + c \cdot z + d = 0$	Equation 4 – Equation of a Plane
$D = \ proj_n \overline{QP_0}\ = \frac{ \overline{QP_0} \cdot n }{\ n\ }$	Equation 5 – Projection of point onto plane
$\overline{QP_0} = \langle x_0 - x_1, y_0 - y_1, z_0 - z_1 \rangle$	Equation 6 – vector from plane to point
$\overline{QP_0} \cdot n = a \cdot (x_0 - x_1) + b \cdot (y_0 - y_1) + c \cdot (z_0 - z_1)$	Equation 7 – dot product of planar normal vector and vector from plane to point
$\ n\ = \sqrt{a^2 + b^2 + c^2}$	Equation 8 – magnitude of planar normal vector
$D = \frac{ a \cdot (x_0 - x_1) + b \cdot (y_0 - y_1) + c \cdot (z_0 - z_1) }{\sqrt{a^2 + b^2 + c^2}}$	Equation 9 – Distance from point to plane (un-simplified)
$d = -ax - by - cz$	Equation 10 – value of d-coefficient
$D = \frac{ a \cdot x_0 + b \cdot y_0 + c \cdot z_0 + d }{\sqrt{a^2 + b^2 + c^2}}$	Equation 11 – Distance from point to plane

Derivation of Planar Coefficients from 3 arbitrary defining points

Equation	Description
$P_1(x_1, y_1, z_1); P_2(x_2, y_2, z_2); P_3(x_3, y_3, z_3)$	Equation 12 – Three Coplanar Points
$\overline{P_1P_2} = (x_2 - x_1, y_2 - y_1, z_2 - z_1) = (x_{12}, y_{12}, z_{12})$	Equation 13 – Planar Vector
$\overline{P_1P_3} = (x_3 - x_1, y_3 - y_1, z_3 - z_1) = (x_{13}, y_{13}, z_{13})$	Equation 14 – Planar Vector
$\overline{P_1P_2} \times \overline{P_1P_3} = (y_{12}z_{13} - y_{13}z_{12}, -x_{12}z_{13} + x_{13}z_{12}, x_{12}y_{13} - x_{13}y_{12})$	Equation 15 – Normal Vector
$\overline{P_1P_2} \times \overline{P_1P_3} = (a, b, c)$	Equation 16 – Normal Vector
$a \cdot (x - x_1) + b \cdot (y - y_1) + c \cdot (z - z_1) = 0$	Equation 17 – Point/Normal form of Planar Equation
$a \cdot x + by + cz + (-ax_1 - by_1 - cz_1) = 0$	Equation 18 – Plane Equation
$a = y_{12}z_{13} - y_{13}z_{12} = (y_1 - y_2)(z_1 - z_3) - (y_1 - y_3)(z_1 - z_2)$	Equation 19 – ‘a’ coefficient
$b = -x_{12}z_{13} + x_{13}z_{12} = -(x_1 - x_2)(z_1 - z_3) + (x_1 - x_3)(z_1 - z_2)$	Equation 20 – ‘b’ coefficient
$c = x_{12}y_{13} - x_{13}y_{12} = (x_1 - x_2)(y_1 - y_3) - (x_1 - x_3)(y_1 - y_2)$	Equation 21 – ‘c’ coefficient
$d = -(ax_1 + by_1 + cz_1)$	Equation 22 – ‘d’ coefficient

Derivation of Point Left or Right of Line/Edge test

Equation	Description
$y - y_1 = m \cdot (x - x_1)$	Equation 23 – Equation of Line in terms of points
$m = \frac{y_1 - y_2}{x_1 - x_2}$	Equation 24 – Slope of Line in terms of two points
$y = m \cdot x + y_1 - m \cdot x_1$	Equation 25 – Line in terms of points
$b = y_1 - m \cdot x_1$	Equation 26 – b constant
$y - y_p = -m \cdot (x - x_p)$	Equation 27 – perpendicular containing point p
$y = -m \cdot x + y_p + m \cdot x_p$	Equation 28 – perpendicular containing point p
$x_i = \frac{1}{2 \cdot m} (m \cdot x_p + y_p + b) - b/m$	Equation 29 – intersecting point i
$\Delta x = x_i - x_p$	Equation 30 – Delta x
$\Delta x = \frac{1}{2 \cdot m} (m \cdot x_p + y_p + b) - b/m - x_p$	Equation 31 – Delta x expanded

Derivation of Circle Test

Equation	Description
$V_1 = (x_1, y_1); V_2 = (x_2, y_2); V_3 = (x_3, y_3)$	Equation 32 – Points
$m_{12} = \frac{y_1 - y_2}{x_1 - x_2}$	Equation 33 – Slope of Line Segment 12
$m_{13} = \frac{y_1 - y_3}{x_1 - x_3}$	Equation 34 – Slope of Line Segment 13
$m_{12p} = \left(\frac{-1}{m_{12}} \right)$	Equation 35 – Perpendicular Bisector of 12
$m_{13p} = \left(\frac{-1}{m_{13}} \right)$	Equation 36 – Perpendicular Bisector of 13
$y - y_p = m \cdot (x - x_p)$	Equation 37 – Point Slope Equation of a line
$y - y_1 = m_{12p} \cdot (x - x_1)$	Equation 38 – Point Slope of 12
$y - y_3 = m_{13p} \cdot (x - x_3)$	Equation 39 – Point Slope of 13
$y = m_{12p} \cdot x - m_{12p} \cdot x_1 + y_1$	Equation 40 – Point Slope of 12
$y = m_{13p} \cdot x - m_{13p} \cdot x_3 + y_3$	Equation 41 – Point Slope of 13
$x_C = \frac{m_{12p} \cdot x_1 - m_{13p} \cdot x_3 + y_3 - y_1}{m_{12p} - m_{13p}}$	Equation 42 – Center of Circle
$y_C = m_{12p} \cdot x_C - m_{12p} \cdot x_1 + y_1$	Equation 43 – Center of Circle

4. Appendix 4 – Supplementary Discussion

4.1 Derivation of calculations used to obtain planar coefficients and triangle normal vector

The planar equation defining the plane a given triangle forms is derived from the triangles' three vertices [Figure 7]. First, the two vectors $\overline{P_1P_2}$ and $\overline{P_1P_3}$, are formed from points P_1 and P_2 and P_2 and P_3 respectively via the following equations:

$$\overline{P_1P_2} = (x_2 - x_1, y_2 - y_1, z_2 - z_1) = (x_{12}, y_{12}, z_{12}) \quad \text{[Equation 13 – Planar Vector]}$$

$$\overline{P_1P_3} = (x_3 - x_1, y_3 - y_1, z_3 - z_1) = (x_{13}, y_{13}, z_{13}) \quad \text{[Equation 14 – Planar Vector]}$$

The cross product of those equations defines the vector normal to the plane composed of the three vertices:

$$\overline{P_1P_2} \times \overline{P_1P_3} = (y_{12}z_{13} - y_{13}z_{12}, -x_{12}z_{13} + x_{13}z_{12}, x_{12}y_{13} - x_{13}y_{12}) \quad \text{[Equation 15 – Normal Vector]}$$

$$\overline{P_1P_2} \times \overline{P_1P_3} = (a, b, c) \quad \text{[Equation 16 – Normal Vector]}$$

Now that the normal vector is acquired, the coefficients ('a', 'b', 'c' and 'd') defining the plane in which the considered triangle's three vertices form, can be derived:

$$a \cdot (x - x_1) + b \cdot (y - y_1) + c \cdot (z - z_1) = 0 \quad \text{Equation 17 – Point/Normal form of Planar Equation}$$

$$a \cdot x + by + cz + (-ax_1 - by_1 - cz_1) = 0 \quad \text{Equation 18 – Plane Equation}$$

The 'd' coefficient is equal to the 4th term in []. From [Equation 15] and [Equation 16], the values of 'a', 'b', 'c' are found to the following:

$$a = y_{12}z_{13} - y_{13}z_{12} = (y_2 - y_1)(z_3 - z_1) - (y_3 - y_1)(z_2 - z_1) \quad \text{[Equation 19 – 'a' coefficient]}$$

$$b = -x_{12}z_{13} + x_{13}z_{12} = -(x_2 - x_1)(z_3 - z_1) + (x_3 - x_1)(z_2 - z_1) \quad \text{[Equation 20 – 'b' coefficient]}$$

$$c = x_{12}y_{13} - x_{13}y_{12} = (x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1) \quad \text{[Equation 21 – 'c' coefficient]}$$

$$d = -(ax_1 + by_1 + cz_1) \quad \text{[Equation 22 – 'd' coefficient]}$$

4.2 Derivation of error function

Several metrics exist for evaluating the quality of a data dependent triangulated mesh. One of the methods involves calculating the distance between a given point and the plane formed by a given triangle. The derivation for this metric is as follows:

The projection of one vector onto another is illustrated in [\[Figure 2\]](#) and described mathematically via the following equations:

The orthogonal projection of the vector a onto b is defined as follows:

$$proj_a b = (a \cdot b)b \quad \text{[Equation 1 – Projection of vector b onto a]}$$

After normalizing the b vectors

$$proj_a b = \left(a \cdot \frac{b}{\|b\|} \right) \left(\frac{b}{\|b\|} \right) \quad \text{[Equation 2 – Normalization of b vectors]}$$

And then simplifying, the following equation results

$$proj_a b = \frac{a \cdot b}{\|b\|^2} b \quad \text{[Equation 3 – Projection of vector b onto a]}$$

Now consider point $Q(x_1, y_1, z_1)$ located on plane F and point $P_0(x_0, y_0, z_0)$ illustrated accordingly in [\[Figure 3\]](#). The distance between the two aforementioned points is defined as follows:

$$D = \|proj_n \overline{QP_0}\| = \frac{|\overline{QP_0} \cdot n|}{\|n\|} \quad \text{[Equation 5 – Projection of point onto plane]}$$

The equation defining the vector from the plane to the point is defined as follows:

$$\overline{QP_0} = \langle x_0 - x_1, y_0 - y_1, z_0 - z_1 \rangle \quad \text{[Equation 6 – vector from plane to point]}$$

The dot product of the above vector and the vector normal to the plane formed by the considered triangle's vertices is as follows:

$$\overline{QP_0} \cdot n = a \cdot (x_0 - x_1) + b \cdot (y_0 - y_1) + c \cdot (z_0 - z_1) \quad \text{[Equation 7 – dot product of planar normal vector and vector from plane to point]}$$

The magnitude of the normal vector is as follows:

$$\|n\| = \sqrt{a^2 + b^2 + c^2} \quad \text{[Equation 8 – magnitude of planar normal vector]}$$

Note [\[Equation 6\]](#), [\[Equation 7\]](#), [\[Equation 8\]](#), which after plugging into [\[Equation 5\]](#), yields [\[Equation 9\]](#):

$$D = \frac{|a \cdot (x_0 - x_1) + b \cdot (y_0 - y_1) + c \cdot (z_0 - z_1)|}{\sqrt{a^2 + b^2 + c^2}} \quad \text{[Equation 9 – Distance from point to plane (un-simplified)]}$$

[\[Equation 9 – Distance from point to plane \(un-simplified\)\]](#)

Further simplifying the above equation, we can define the d coefficient as follows:

$$d = -ax - by - cz \quad \text{[Equation 10 – value of d-coefficient Plugging]}$$

Remember $Q(x_1, y_1, z_1)$ lies within plane F and satisfies [\[Equation 4\]](#), therefore [\[Equation 10\]](#). Plugging [\[Equation 10\]](#) into [\[Equation 9\]](#) results in the following equation defining the distance D between unused points in the original LiDAR data and the triangulated surface approximation:

$$D = \frac{|a \cdot x_0 + b \cdot y_0 + c \cdot z_0 + d|}{\sqrt{a^2 + b^2 + c^2}} \quad [\text{Equation 11 – Distance from point to plane}]$$

The derivation of [Equation 11](#) is described in detail because this distance is equal to the local error between unused points in the original LiDAR data and the triangulated surface approximation.

4.3 Delaunay Triangulation Circle Test

After the insertion of a point, the edges of a triangle, with one of its vertex's being the candidate point, are checked to see if the adjacent triangle, sharing that edge, is compatible to have its diagonal flipped in accordance with Delaunay triangulation. This check is done via the circle test. The circle test aims to change the diagonal of a given two considered triangles to maximize the smaller interior angles in a given pair of triangles.

For example, in [Figure 6](#) if consider the two triangles, T_1 (composed of vertices V_1 , V_2 , and V_3) and T_2 (composed of vertices V_2 , V_3 , and V_4). Because vertex V_4 lies outside of circle CT_1 formed by the three points V_1 , V_2 , and V_3 (conversely because vertex V_1 lies outside of circle CT_2 formed by the three points V_2 , V_3 , and V_4), the configuration of the triangles exist such that their lesser internal angles are maximized.

In order to do this, the center of the circle, formed by the three vertices must be calculated. In [Figure 12](#), consider the lines formed by vertices V_1 , V_2 , and V_1 , V_3 , and then their perpendicular bisectors. Note that the three perpendicular bisectors of the sides of a triangle are concurrent at a point called the circumcenter (center of the circle). The slopes of the line segments formed by vertices V_1 , V_2 , and V_1 , V_3 are defined as follows:

$$V_1 = (x_1, y_1); V_2 = (x_2, y_2); V_3 = (x_3, y_3) \quad \text{Equation 32 – Points}$$

$$m_{12} = \frac{y_1 - y_2}{x_1 - x_2} \quad [\text{Equation 33 – Slope of Line Segment 12}]$$

$$m_{13} = \frac{y_1 - y_3}{x_1 - x_3} \quad [\text{Equation 34 – Slope of Line Segment 13}]$$

Therefore, the slopes of the perpendicular bisectors are defined as follows:

$$m_{12p} = \left(\frac{-1}{m_{12}} \right) \quad [\text{Equation 35 – Perpendicular Bisector of 12}]$$

$$m_{13p} = \left(\frac{-1}{m_{13}} \right) \quad [\text{Equation 36 – Perpendicular Bisector of 13}]$$

Using the point slope form of the equation of a line:

$$y - y_p = m \cdot (x - x_p) \quad [\text{Equation 37 – Point Slope Equation of a line}]$$

The two perpendicular bisector line equations can be calculated as follows:

$$y - y_1 = m_{12} \cdot (x - x_1) \quad [\text{Equation 38 – Point Slope of 12}]$$

$$y - y_3 = m_{13} \cdot (x - x_3) \quad [\text{Equation 39 – Point Slope of 13}]$$

Distributing the slope and solving in terms of 'y', the above equations reduce to:

$$y = m_{12} \cdot x - m_{12} \cdot x_1 + y_1 \quad [\text{Equation 40 – Point Slope of 12}]$$

$$y = m_{13} \cdot x - m_{13} \cdot x_3 + y_3 \quad [\text{Equation 41 – Point Slope of 13}]$$

Setting the equations equal to one another and solving for x (the intersection of the two perpendicular bisectors or the center of the circle in [Figure 12](#)):

$$x_c = \frac{m_{12} \cdot x_1 - m_{13} \cdot x_3 + y_3 - y_1}{m_{12} - m_{13}} \quad [\text{Equation 42 – Center of Circle}]$$

Plug [] back into [] to solve for the y-coordinate center of the circle:

$$y_c = m_{12} \cdot x_c - m_{12} \cdot x_1 + y_1 \quad [\text{Equation 43 – Center of Circle}]$$

If $\|R_c\| - \|V\|_4 > 0$, then the point is contained within the circle and flipping occurs. Else, if the

aforementioned condition is not satisfied, then the point is outside the circle, the lesser of the two internal angles are already maximized for the considered pair of triangles and no flipping occurs.

4.4 Derivation of edg_RL function

Consider the points $(x_1, y_1); (x_2, y_2); (x_p, y_p); (x_i, y_i)$; in [Figure 1](#). The 1 and 2 are points defining an edge of a triangle. The point p is an arbitrary point in which we wish to determine if that point exists to the left or right of the edge defined by points 1 and 2. Point i is the intersection of a line bisecting the edge defining by points 1 and 2 and containing point p. The slope in terms of points 1 and 2 (the vertices of a given triangle) is defined by [Equation 24](#). For the equation of the line [Equation 25](#), the b term is therefore equal to [Equation 26](#) in terms of points 1 and 2. The slope of a line perpendicular to points 1 and 2 is equal to $-m$. Therefore, the equation of the line that is perpendicular to the line defined by points 1 and 2 and contains points p is equal to [Equation 27](#) or after further simplification [Equation 28](#). After setting the two equations ([Equation 25](#), [Equation 28](#)) equal to one another, and solving for x, one can derive the following equation [Equation 29](#), which is in terms of the point p, the slope m, and the b term of the line defined by points 1 and 2. Note that in [Equation 29](#) the 'x' term becomes xi because setting the two equations ([Equation 25](#), [Equation 28](#)) equal and solving solves for the point of intersection between the two lines. Now that both point i and p are solved for, the difference between the x values determines if the point p is to the right or left of the line defined by points 1 and 2. If the term delta x is defined as follows [Equation 30](#), or in terms of m, b, and point p – [Equation 31](#), then for all positive values of delta x, the point p is to the right of the line. Also

for all negative values the point p is to the left of the line defined by points 1 and 2. If the delta x value is equal to 0, the point exists on the line defined by points 1 and 2.

Some rare cases of concern arise however. What if the points defining the line, 1 and 2, have the same x or y values? If they have the same x values, then simply compare the x value of 1 and 2 with the point p . If points 1 and 2 have the same y value, then base the test on a comparison of the y value in points 1 and 2 and the point in question. However, in this case, you are now answering the question is the point p above or below the line defined by points 1 and 2. In any case, the purpose of the `edg_RL` function is to group all points inputted to the function based on what side of the line they exist on. The mathematical equations necessary to realize this grouping have been derived and described above.

5. Appendix 5 – Reference List

References:

1. Morgan, Michel; and Habib, Ayman; “3D TIN For Automatic Building Extraction from Airborne Laser Scanning Data.” Proceedings of the ASPRS “Gateway to the New Millennium”, St. Louis, Missouri (23-27 April, 2001).
2. Hofmann, A.D.; “Analysis of TIN-Structure Parameter Spaces In Airborne Laser Scanning Data for 3-D Building Model Generation.” Geo-Imagery Bridging Continents XXth ISPRS Congress, 12-23 July 2004, Commission 3
3. Liang-chien C. , Tee-ann T., Yi-chen S., Yen-chung L., Jiann-yeou R.; “Fusion of Lidar Data and Optical Imagery for Building Modeling.” Geo-Imagery Bridging Continents XXth ISPRS Congress, 12-23 July 2004, Commission 4, p. 732 ff.
4. Lattuada, Roberto; Raper, Jonathan; “Applications of 3D Delaunay triangulation algorithms in geoscientific modeling.” The Third International Conference/Workshop on Integrating GIS and Environmental Modeling CD-ROM, January 1996
5. Wang, Kai; Lo, Chor Pang; Brook, George; Arabnia, Hamid; “Comparison of existing triangulation methods for regularly and irregularly spaced height fields.” INT. J. Geographical Information Science, vol. 15, no. 8, pp 743-762, 2001
6. Barry, Joe; “Delaunay versus max-min solid angle triangulations for three dimensional mesh generation” Internal Journal for Numerical Methods in Engineering, Vol 31, 987-997,1991
7. Kidner, David B.; Ware, Mark J.; Sparkes, Andrew J.; Jones, Christopher B.; “Multiscale Terrain and Topographic Modeling with the Implicit TIN.” Transactions in GIS, Vol. 4, no. 4, pp 370-408, 2000
8. Lee, J; “Analyses of visibility sites on topographic surfaces.” International Journal of Geographic Information Systems, vol. 5, p 413-29, 1991
9. Garland, M.; Heckbert, P. S.; “Fast polygonal approximation of terrains and height fields.” Technical Report, Department of Computer Science, Carnegie Mellon University, 1995.
10. Mass, H. G., and Vosselman, G., 1999, Two algorithms for extracting building models from raw laser altimetry data, ISPRS Journal of Photogrammetry & Remote Sensing, 54, 153-163.