2006

Heuristic 3d Reconstruction Of Irregular Spaced Lidar

Nicholas Shorter
University of Central Florida

HEURISTIC 3D RECONSTRUCTION OF IRREGULAR SPACED LIDAR

by

NICHOLAS SVEN SHORTER
B.S.E.E. University of Central Florida, 2005

A thesis submitted in partial fulfillment of the requirements
for the degree of Master of Science
School of Electrical Engineering and Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Summer Term
2006

ABSTRACT

As more data sources have become abundantly available, an increased interest in 3D reconstruction has emerged in the image processing academic community. Applications for 3D reconstruction of urban and residential buildings consist of urban planning, network planning for mobile communication, tourism information systems, spatial analysis of air pollution and noise nuisance, microclimate investigations, and Geographical Information Systems (GISs). Previous, classical, 3D reconstruction algorithms solely utilized aerial photography. With the advent of LIDAR systems, current algorithms explore using captured LIDAR data as an additional feasible source of information for 3D reconstruction.

Preprocessing techniques are proposed for the development of an autonomous 3D Reconstruction algorithm. The algorithm is designed for autonomously deriving three dimensional models of urban and residential buildings from raw LIDAR data. First, a greedy insertion triangulation algorithm, modified with a proposed noise filtering technique, triangulates the raw LIDAR data. The normal vectors of those triangles are then passed to an unsupervised clustering algorithm – Fuzzy Simplified Adaptive Resonance Theory (Fuzzy SART). Fuzzy SART returns a rough grouping of coplanar triangles. A proposed multiple regression algorithm then further refines the coplanar grouping by further removing outliers and deriving an improved planar segmentation of the raw LIDAR data. Finally, further refinement is achieved by calculating the intersection of the best fit roof planes and moving nearby points close to that intersection to exist at the intersection, resulting in straight roof ridges. The end result of the aforementioned techniques culminates in a well defined model approximating the considered building depicted by the LIDAR data.

This work is dedicated to my parents

Kathy and Sven Shorter

And my sister

Nichole Shorter

ACKNOWLEDGMENTS

The following people, Dr. Takis Kasparis, Dr. Wasfy Mikhael, Dr. Michael Georgiopoulos, and Dr. Andy Lee, have greatly enhanced my understanding of the fundamental concepts I have used in my research. I thank Dr. Kasparis for the invaluable knowledge he has passed onto me throughout the course of my college career as an advisor, an instructor and a friend. I thank Dr. Mikhael for not only his instruction on a plethora of Digital Signal Processing related concepts, but for also sharing with me various fundamental tactics for approaching engineering modeling applications. I thank Dr. Georgiopoulos for his instruction on numerous applications related to both supervised and unsupervised learning strategies. I am also thankful for both Dr. Mikhael and Dr. Georgiopoulos for serving on my defense committee. Their valued advice towards my research and feedback in regards to my thesis defense will go a long way. I thank Dr. Lee for providing me with insight on LIDAR applications and educating me about some industrial perspectives on LIDAR and 3D reconstruction.

I would like to thank my immediate family, Nichole, Kathy and Sven Shorter for all of their support. Their motivational coaching has been an abundant source of encouragement for me to strive to push myself to my limits. I would also like to thank my girlfriend Lindsey. Her compassion and care are a source of inspiration and determination for me.

This research would have not been possible without the generous donations of LIDAR test set data provided from several companies, data valued at thousands of dollars. I thank Dr. Simone Clode and Dr. Franz Rottensteiner for giving me access to the LIDAR data set known as the ‘Fairfield’, which was collected and donated by AAMHatch. While only the Fairfield test set is considered in this paper, in future related research works and publications, the other test sets

donated by other companies will be incorporated. I thank Mr. John Ellis with AeroMap, Mr. Paul Mrstik with Terra Point, and Mr. Steffen Firchau with TopoSys for their companies' contributions as well.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

3D	Three Dimensional
ALS	Airborne Laser Scanning
ART	Adaptive Resonance Theory
DSM	Digital Surface Model
DTM	Digital Terrain Model
FSART	Fuzzy Simplified Adaptive Resonance Theory
GPS	Global Positioning System
INS	Inertial Navigation System
LIDAR	Light Detection and Ranging
TIN	Triangulated Irregular Network

CHAPTER ONE: INTRODUCTION TO LIDAR AND 3D RECONSTRUCTION

The concept of deriving three-dimensional models from various sources of data has existed for several decades now. Known as the 3D Reconstruction problem, methodologies for solving this problem and even extending its application have evolved with the advent of new technologies which deliver new and/or improved sources of data. The research presented focuses on 3D Reconstruction via primarily using Light Detection and Ranging (LIDAR) data.

First, various applications of 3D reconstruction are introduced. Following that, a basic overview of how LIDAR works is presented. Then a discussion of 3D reconstruction methodologies in general is presented. A discussion about the drawbacks of interpolating LIDR points to fixed intervals as opposed to working with the raw LIDAR data is then presented. Finally, the specific problem in which this area of research addresses is laid out.

1.1 Applications of 3D Reconstruction

Applications of 3D Reconstruction have valued use for both militaristic and commercial purposes. An example of an application of 3D reconstruction for military applications is as follows. Imagine troops, rather than simply reviewing aerial photos prior to an invasion on a given territory, are instead able to experience a virtual three dimensional walk through of the given terrain from models constructed by a 3D reconstruction algorithm. For commercial uses, the demand for 3D models of buildings has applications such as urban planning, network planning for mobile communication, spatial analysis of air pollution and noise nuisances, microclimate investigations, geographical information systems, and security services. For

entertainment purposes, 3D reconstruction can be used for tourism information systems. Tourists, instead of using 2D maps to find their way around theme parks, could use a kiosk to view a virtual 3D walk through of the park.

1.2 LIDAR Overview

Mounted on the aircraft, helicopter or plane, collecting the LIDAR data, is a Global Positioning System (GPS), an Inertial Navigation System (INS) and a LIDAR sensor system. The GPS returns the longitude and latitude coordinates of the aircraft's current position. The INS tracks the altitude of the LIDAR sensor. The LIDAR sensor itself emits a laser beam from the sensor. This beam then travels till it interacts with a given target.

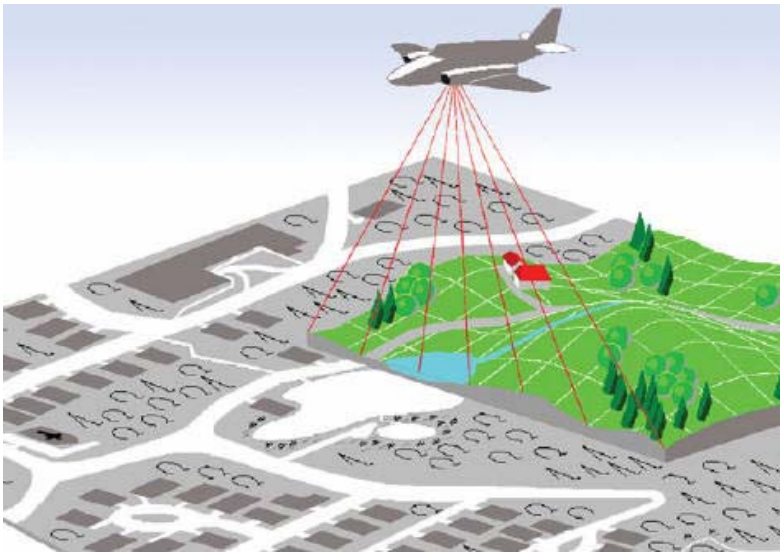


Figure 1: Capturing LIDAR Data

In the case of a building surface, the laser will reflect off of the building's surface and return to the sensor. In the case of tree foliage or vegetation, two possible scenarios arise: the laser beam could pass through the foliage or vegetation and hit the ground, or the laser could

interact with the foliage or vegetation. The first pulses to return to the sensor are labeled First Return Pulses. These pulses consist of laser beams which interacted with the top of foliage and vegetation and building structures. The last pulses to return to the sensors are labeled Last Return Pulses. These pulses consist of laser beams which passed through foliage and/or vegetation and interacted with the ground. These pulses also consist of laser beams which interacted with building surfaces. Based on the time it takes from the emission of the laser from the sensor to the return of the laser beam after it has interacted with a given target, the range from the sensor to the target can be calculated. Differences between first and last return pulses for vegetation and building edges are relatively high. Differences between first and last return pulses for building surfaces are relatively low. When the laser beam, emitted, from the LIDAR sensor hits a building surface, the majority of the beam is immediately reflected leaving minimal difference between first and last returns.

While some primitive LIDAR systems only return the longitude, latitude, and elevation of a given returned point, newer systems can capture sampling time, longitude, latitude, elevation, the intensity of the returned signal, and the first and last return pulses. The data set considered for this research set does contain sampling time, longitude, latitude, returned signal intensity, first and last return pulse attributes and is commonly referred to as 'Fairfield' test set. The 'Fairfield' test set, provided by Dr. Simone Clode and Dr. Franz Rottensteiner and procured by AAMHatch, covers a two square kilometers of both an urban and residential area of Fairfield, Australia. The data set comes with both the LIDAR data, in ASCII format, and a corresponding aerial photograph of the terrain. The aerial photograph has 15 centimeter pixel resolution. This means that each pixel in the aerial photograph corresponds to 15 centimeters in the actual terrain depicted. The LIDAR data has approximately a 1 point per $1.3m^2$ point spacing density. Simone

et. al. in [17] have reported that first and last returns differing in less than 4.6 meters in elevation are not valid. The reason for this was found to be a limitation in the LIDAR sensor itself. The sensor had to reset itself before a second return could be recorded. If a second return comes back to the sensor before the reset time has passed, a dual return was being recorded. Hence if the first and last return were less than 4.6 meters apart, the two returns arrived back at the laser before it could reset itself in time to record the second return, resulting in the LIDAR system simply recording the same return for both first and last return pulses. The algorithm implemented makes use of both returns by only triangulating both returns if their elevation difference is greater than 4.6 meters.

1.3 3D Reconstruction Methodologies

Several defining traits characteristic of a given 3D reconstruction algorithm distinguishes it from other algorithms, traits such as the following: the sources of data the algorithm operates on and the technique the algorithm utilizes to approximate the building surface.

Some algorithms only consider the use of stereo pairs of aerial images procured from satellites, planes and helicopters. A stereo pair of images is two images having a significant amount of overlap depicting the same scene. Huguet et. al. develop a building segmentation method called Color-Based Watershed Segmentation in [22] and employ it to realize 3D Reconstruction of urban scenes from low altitude images in [23]. However, several portions of their algorithm are still undergoing extensive experimental validation. In [41], Suveg and Vosselman present an autonomous 3D reconstruction algorithm which uses a set of basic building models to approximate buildings depicted in a sequence of images and 2D GIS maps

(which contains building outlines or footprints). The algorithm presented in that paper assumes all buildings can be reconstructed from simple building models with flat, gable or hip roof. Furthermore, the algorithm also assumes that the 2D GIS building footprint maps are available for a given area and that those buildings can be partitioned into a collection of rectangles.

The advent of LIDAR sensor systems created a whole new genre of 3D reconstruction algorithms which made use LIDAR data as an additional data source. Another model based approximation algorithm, processing LIDAR data instead of images and GIS maps, is presented in [30] by Mass and Vosselman. As with [41], model based reconstruction assumes the given depicted building can be accurately approximated with a pre-existing building model.

In contrast to model based LIDAR 3D Reconstruction approaches, several data driven approaches have been developed. These data driven 3D Reconstruction algorithms typically begin by separating building points from non building points. They then 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.

Rottensteiner and Briese in [35] construct buildings from LIDAR data by detecting characteristics in the data that delineate buildings from their surroundings and then detecting characteristics specific to those buildings. The LIDAR data is separated into building and non-building regions via the algorithm described in [34]. This process is implemented by using morphological filters for computing a digital terrain model (DTM) and then applying a thresholding technique to the height differences between the DTM and the digital surface model (DSM). Note that in a DTM and DSM the LIDAR data is interpolated to fixed point spacings. The DSM is a model depicting both terrain and non terrain (building) points. By using

morphological filters and other filtering techniques, everything but the terrain is removed from the DSM, thus creating a digital terrain model (DTM). Then roof planes are detected via a curvature based segmentation technique, and then grouped into polyhedral building models.

As in [35], Chen et. al. in [16], follow a similar procedure. First, ground from non ground interpolated points are separated. Then 'building regions' are segmented and co-planarity is analyzed to shape the roof of the building regions. Finally a patented Split-Merge-Shape (SMS) method is employed to create building models from the aforementioned gathered information. While the roofs are generated from the raw LIDAR data, building and non-building classification is still done with DTM and DSM thresholding techniques.

Fujii and Arikawa use both LIDAR data and aerial images in [18]. First LIDAR data, interpolated at fixed intervals, is analyzed for line segments forming object contours. Identifying these contours as buildings makes way for building extraction. Contours in the LIDAR data are then registered with those of aerial imagery of the same scene and then texture mapping from the imagery onto the LIDAR data occurs. A voting technique using the Hough transform is implemented to minimize mismatching.

In [29], Overby et. al make use of a three-dimensional extension of the Hough transform for extracting planes from point cloud data. Geometric and other constraints further refine those planes and vote whether or not to reject them. These planes are then merged to form three-dimensional models.

All of the above mentioned algorithms vary from one another based on the sources of input utilized and the methodology implemented to realize 3D reconstruction. The majority of the above discussed methods use an interpolation of the LIDAR data in some form or another for some part of the algorithm, in most cases for distinguishing building from non-building points.

For the most part, algorithms will use some combination of LIDAR data, aerial imagery and/or GIS ground plan data for 3D reconstruction. Most of the algorithms apply some version of a thresholding technique from the DSM and DTM differences for distinguishing ground points from non ground points. For the 3D reconstruction process, methods either strategically grouped the coplanar data by extracting features or attempted to model the data by fitting pre-existing models. The methods grouping coplanar data did so by extracting key features such as break points and ridges. Some of the algorithms mentioned made use of various versions of the Hough transform to realize this. Discussed in the following chapter is a plethora of algorithms approximating the LIDAR data with a TIN and then performing 3D reconstruction by merging coplanar triangles together to form planes.

1.4 Interpolating Irregular Raw LIDAR to Fixed Intervals

Several data driven methods make use of applying a thresholding technique to the height differences between a Digital Terrain Model (DTM) and a Digital Surface Model (DSM) to distinguish building from non building regions. By applying morphological filtering techniques such as [47] and [4] to a DSM, it is possible to create a DTM, a model representing only the terrain, without buildings, trees, cars, etc. The morphological filters exist as kernel functions, relatively small matrices, which operate on larger matrices. The original LIDAR data, existing at irregular point intervals, is interpolated to fixed point intervals. Therefore the row $(X_1, X_2, X_3, ...)$ and column $(Y_1, Y_2, Y_3, ...)$ position of a given LIDAR point in a

DSM matrix corresponds to its longitude and latitude, respectively. The value of the cell in the matrix corresponds to the elevation (Z_{XY}) at the interpolated location.

	X_1	X_2	X_3	...
Y_1	Z_{11}	Z_{12}	Z_{13}	...
Y_2	Z_{21}	Z_{22}	Z_{23}	...
Y_3	Z_{31}	Z_{32}	Z_{33}	...
⋮	...	...	...	...

Figure 2: LIDAR at Interpolated Spaces

Interpolating the data to fixed point intervals does simplify the problem and enable the use of kernel filtering functions. While morphological filtering may not be used in all cases, still many algorithms ([16], [17], [18], [21], [29],[32], [36], [37], [39], [40], [45]) typically continue to interpolate to fixed intervals to make use of conventional methods.

However, in [17], Clode et. al., report the limits of their building detection technique and how interpolating the LIDAR points only adds to the inaccuracy and limitations of their method and all methods in general using DSMs. The accuracy in which a given algorithm can delineate building from non building regions is dependent on the laser divergence and the flying height of the aircraft procuring the LIDAR data, or ultimately the laser footprint uncertainty. However, if the data is interpolated to fixed intervals, then the limitation of the accuracy is worsened. With interpolation to fixed intervals, now a given algorithm's uncertainty is not simply a function of

the laser foot print uncertainty, but a function of the laser foot print uncertainty and the point spacing combined. In [42], Vosselman elaborates that when the irregular points are interpolated, in instances where heights are interpolated between ground points and points on vegetation or buildings, the height differences in the interpolated data will be reduced. These instances increase the difficulty of making correct classifications distinguishing ground points from non-ground points.

Problem Statement

The problem in which the 3D reconstruction academia attempts to tackle is the 3D reconstruction of models based on multiple sources of data. The 3D reconstruction aims to tactically fuse independent, correlated forms of data to derive the most accurate 3-dimensinal model from the depicted sources' data. Ideally, the algorithm will fuse as many data sources that are available and perform the 3D reconstruction autonomously with no user intervention nor the adjustment of parameters from building to building. One can think of 3D reconstruction as a black box with the input as a collection of one or more of the following sources of data: LIDAR data, aerial photography and GIS plans. The output of this box therefore are virtual, 3D models of the terrain depicted by the original sources of input data.

CHAPTER TWO: LIDAR TRIANGULATION

Several approaches for constructing three dimensional models from LIDAR data were presented in the previous chapter. These approaches, after using varying means of separating building points from non building points, used techniques such as extensions of the Hough transform and other means of feature extraction methods for grouping coplanar points and then forming planes. These planes were in turn merged and geometrically constrained to form three dimensional models.

In this chapter a different methodology is specifically explored. Rather than analyzing the points, there is a class of 3D reconstruction algorithms that instead analyze triangles which the LIDAR points form. Several data driven LIDAR model 3-D reconstruction algorithms currently exist in the literature ([32],[20],[29],[28]) which 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 represented 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: clustering approach; least squares approximation approach; 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 [32] 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 [29], 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. In region growing approaches, the normal vector of a considered triangle is analyzed. If the normal vectors of triangles adjacent to the originally considered triangle fall within a certain threshold from the normal vector of the originally considered triangle, the adjacent triangles are then clustered to the same label as that of the originally considered triangle. Then other adjacent triangles are checked and the process repeats. If the normal vectors of the adjacent triangles ever exceed the threshold in comparison to the normal vectors of the originally considered triangles, then a new cluster label is created.

Hoffman however uses the clustering approach in [20] to group together triangles in the TIN that contain similar properties. In [20], 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 [28] detail several advantages for describing a geo-model with a three dimensional triangulation. These details have been enumerated in [Table 1].

Table 1: Advantages of Representing LIDAR as a TIN

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

Several different methodologies exist for triangulating a dataset. One of the most popular techniques, the Delaunay triangulation, attempts to maximize the lesser two internal angles in a given triangle for all triangles. A detailed derivation of the equations associated with the Delaunay triangulation circle test is presented in section C of the appendix.

For only 2 dimensions, the Delaunay triangulation method, given four points, chooses the diagonal that 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 [8], 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 [43] compare the Delaunay triangulation process against several other triangulation processes when approximating two different terrains from Digital Surface Models (DSMs). 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). 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 [43], a more detailed version of the algorithm's description is presented in [19]. Although in [43] the greedy sequential algorithm is tested on a DSM, the algorithm can be easily modified to work with irregular point spacings instead. Later described in the algorithm implementation section, this facet will be important as the points to be triangulated exist as irregular point spacings.

In [43], 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: triangle edges do not intersect one another; and triangles cannot overlap each other.

A plethora of triangulation methods for digital surface models exist. Geological information system (GIS) users typically prefer to interpolate the irregularly spaced raw LIDAR points, producing a digital surface model, 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 DSMs argue they are an aberration which over-simplifies terrain modeling [27]. 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 [25], Kidner et. al. argue that ultimately for each particular type of application their 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.

CHAPTER THREE: ALGORITHM IMPLEMENTATION

In chapter one, a brief literature review for 3D reconstruction methods in general is presented. In chapter two, a focused literature review for 3D reconstruction algorithms utilizing triangulation algorithms is presented. Furthermore, a literature review of existing triangulation algorithms is also presented. In this chapter, the actual algorithm implemented to realize 3D reconstruction from the raw LIDAR data is presented in several sections: (1) triangulation; (2) filtering modification (3) clustering; (4) regression refinement.

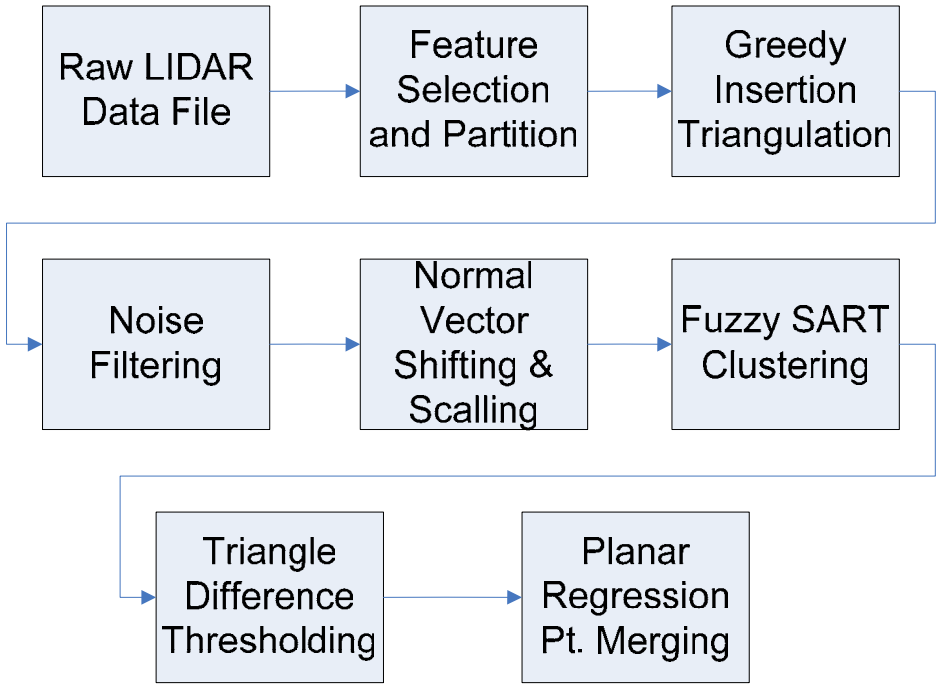


Figure 3 – System Block Diagram

Consider the above system block diagram depicting the implemented algorithm. The original data set (two square kilometers in size) is partitioned into smaller segments and key features, such as first and last return pulses, laser intensity, longitude, latitude and elevation are

extracted. The irregularly distributed (raw) LIDAR data is then triangulated using the sequential greedy insertion algorithm. The traditional greedy insertion algorithm has however been modified to filter systematic errors within the LIDAR data. Several proposed processing techniques are then implemented. The entire input space is translated to center at the origin. The normal vectors of the triangles generated from the greedy insertion algorithm are then clustered by a Fuzzy SART clustering algorithm in order to construct a rough grouping of coplanar triangles. The groups of coplanar triangles then undergo a multimodal (or planar) regression analysis to calculate planes to approximate those groups of coplanar triangles. Then, for each group of coplanar triangles, outliers and erroneous triangles are removed, and a refined selection of coplanar triangles is retained. Of this selection, an improved planar approximation representing these coplanar triangles is calculated and finally a plane approximating the triangles is formed.

3.1 Triangulation – Greedy Insertion

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 high 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.

3.2 Triangulation Rules

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

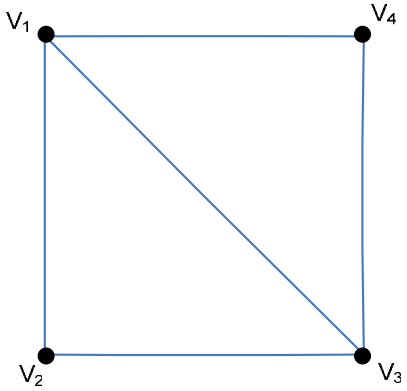
1. No intersecting triangle edges are to exist within the TIN.
2. Furthermore, no overlapping triangles are to exist within the TIN.
3. No gaps are permissible within the TIN.
4. When considering a point for the formation of a triangle, the neighboring points closest to the point in consideration must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

3.3 Selected Triangulation Algorithm

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 [19], Garland and Heckbert present both the sequential and parallel 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.

The sequential greedy insertion algorithm simultaneously optimizes 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).



$$V_4 = <\max(x_i), \max(y_i)>$$
$$V_1 = <\min(x_i), \max(y_i)>$$
$$V_2 = <\min(x_i), \min(y_i)>$$
$$V_3 = <\max(x_i), \min(y_i)>$$

Figure 4: Initial Triangulation

That formation is then checked to see if flipping the diagonal will optimize the arbitrarily formed configuration to conform to Delaunay triangulation.

For all triangles, the distances between the triangles (planes) [Figure 5] and the points that they encompass (in x and y or longitude and latitude spacing) are calculated.

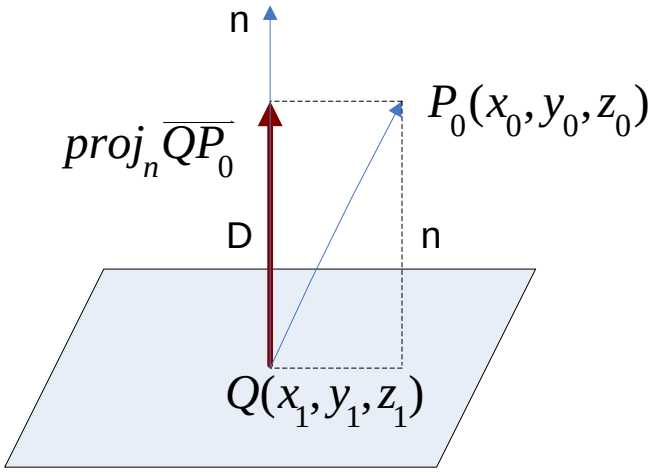


Figure 5: Distance between a given point and a plane

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, which encompasses a given point not yet inserted, is presented in section B 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 triangles data structure.

All of the LIDAR points that 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 at the edge of the outermost initial quadrilateral; and (3) the candidate point is inserted on a triangle edge.

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 6].

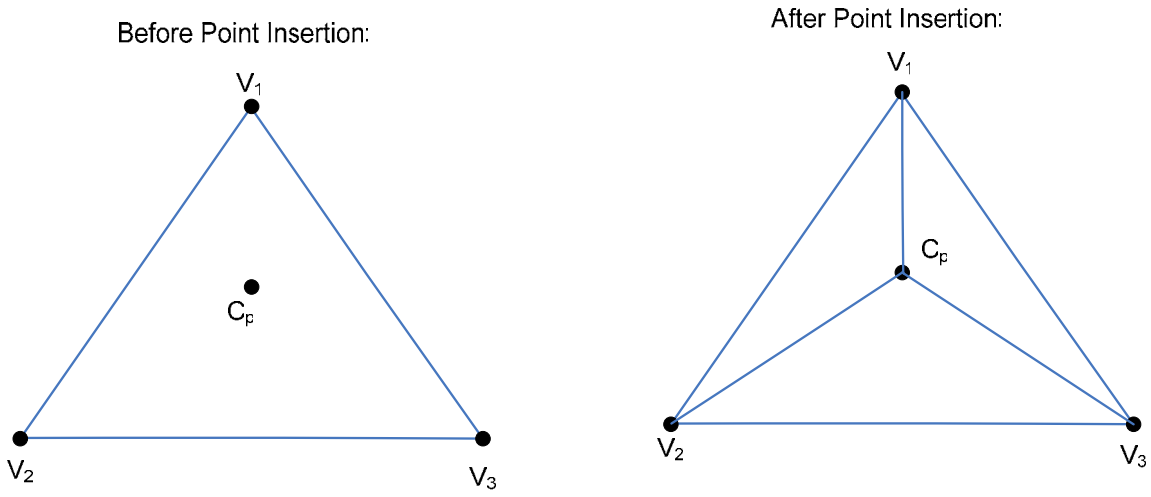


Figure 6: Point Insertion (Case 1)

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 7].

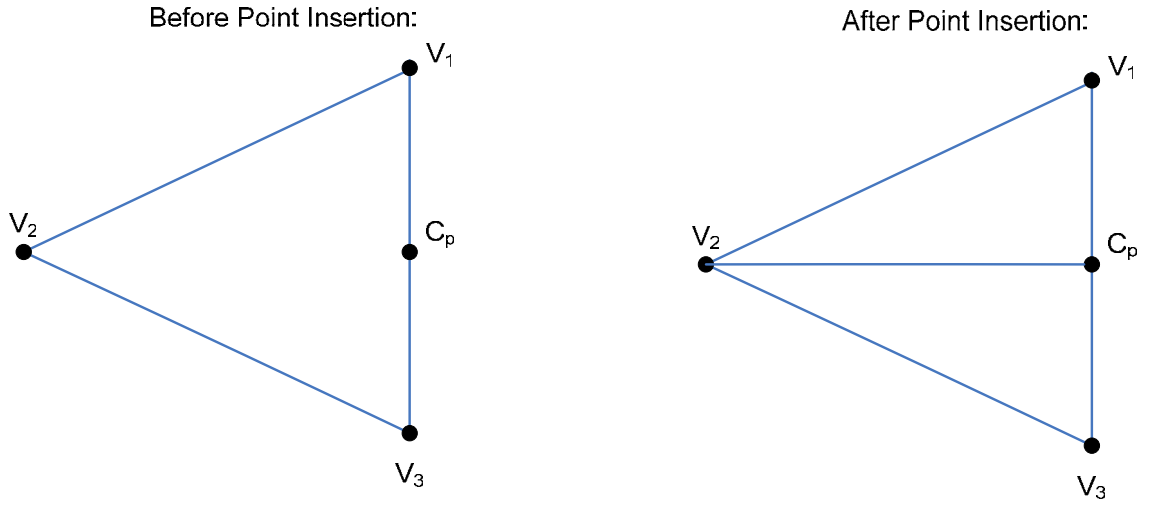


Figure 7: Point Insertion (Case 2)

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 8].

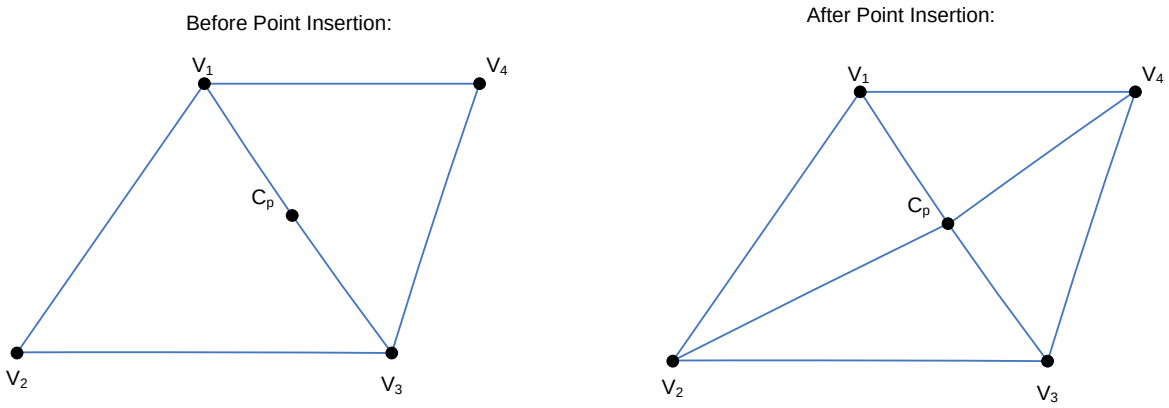


Figure 8: Point Insertion (Case 3)

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 'relatively close' to 180 degrees. Therefore, triangle 'B' depicted in [Figure 9] is desired over triangle 'A'.

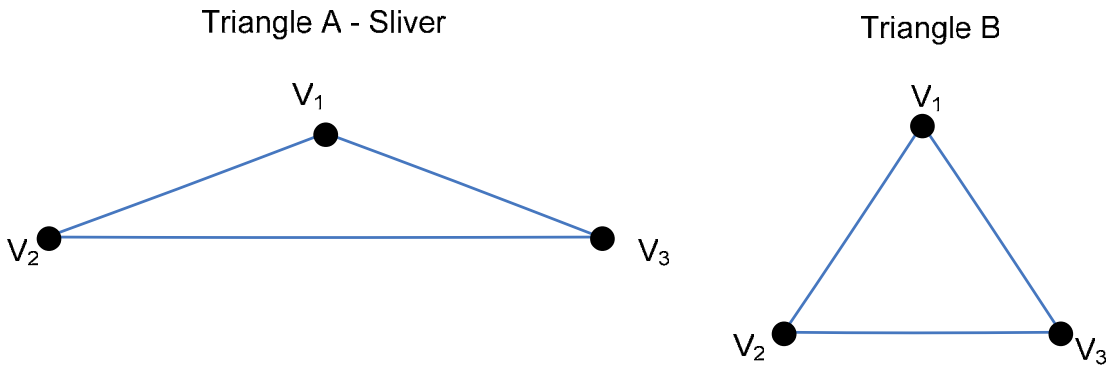


Figure 9: Sliver Example

All of the above procedures are depicted in the block diagram contained in [Figure 10].

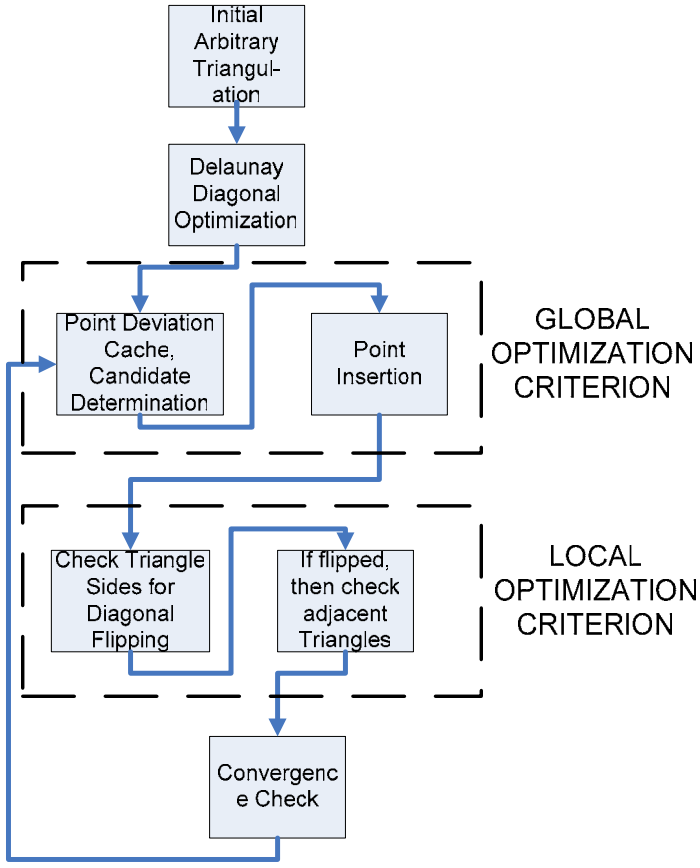


Figure 10: Greedy Insertion Block Diagram

3.2 Filtering Modification

In order to group the coplanar triangles together, the normal vectors of the triangles generated from the greedy insertion algorithm are passed as inputs to the Fuzzy SART clustering algorithm. In order gain information from the magnitude of the normal vectors, the vector spanning from the origin to the triangle center is projected onto the normal vector. Therefore the orientation of the vector aligns with the orientation of the normal vector and the magnitude represents the distance from the origin to the plane encompassing the considered triangle.

The elevation coordinates of the LIDAR data are actually only accurate to a certain order of magnitude (in the order of centimeters). Making matters worse, the LIDAR data suffers from systematic errors and noise. Therefore, noise is existent in the data and presents difficulties for coplanar clustering based on the normal vectors of the triangles existent in the TIN generated from the raw LIDAR. An ideal set of coplanar triangles [Figure 11], actually exist as points jittering about that plane, as shown in [Figure 12]. The noise causes the LIDAR points to deviate from the ideal plane, thereby causing the normal vectors of the triangles to deviate from their ideal directions.

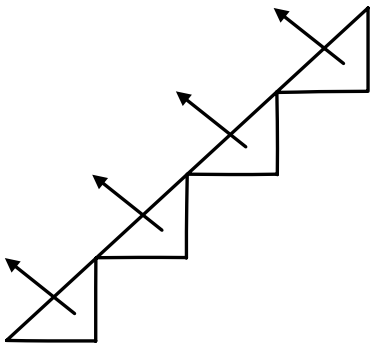


Figure 11: Ideal LIDAR Points

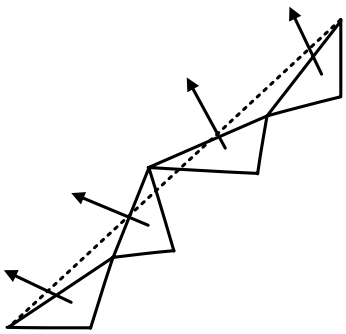


Figure 12: Actual LIDAR Points

One way to filter these errors would be to exploit the very nature of the triangulation algorithm selected. Sequential greedy insertion inserts the points farthest away from the initial plane established. Therefore points along roof ridges, roof corners, and building edges are the points inserted first. The points inserted last are the points closest to an established plane, the points with the smallest errors. It is possible to simply program the sequential greedy insertion triangulation algorithm to only triangulate points above a certain error threshold. However, the insertion of fewer points leads to a less accurate TIN and furthermore, leads to fewer triangles sharing the same plane. Rather than not inserting the triangles, leading to fewer members of a given coplanar cluster, it would be advantageous to correct the inaccuracies of the points along the z or height dimension. Since the points, which jitter about the already established roof plane, are contained in a well defined plane, it is possible to remove the jitter or systematic error or noise by placing points below a certain threshold distance on the plane in which they are contained. While the longitude and latitude dimensions were preserved, the elevation dimension of a candidate point was modified if the candidate point met the following conditions: the perpendicular distance, defined in equation (1), of the candidate point was less than .2 meters ($D_c \leq .2m$) from the containing triangulated plane; and the pitch of the roof , defined in equation (2) was less than 60 degrees ($\theta \leq 60^o$).

$$D_c = \frac{\left|a\cdot(x_0 - x_1) + b\cdot(y_0 - y_1) + c\cdot(z_0 - z_1)\right|}{\sqrt{a^2 + b^2 + c^2}} \tag{1}$$

$$\theta = 90 - \sin^{-1}\left(\frac{D_c}{Z_c}\right) \tag{2}$$

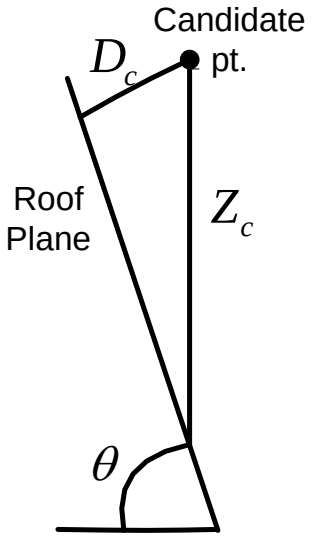


Figure 13 - Pitch of Roof Plane (Theta)

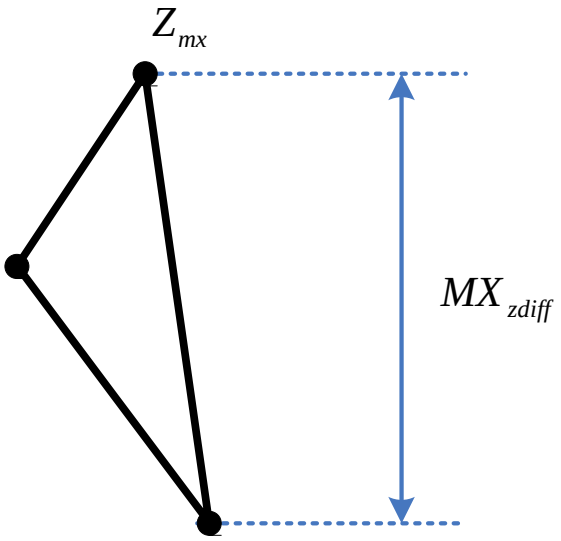


Figure 14 - Triangle Elevation Difference

Without the constraint imposed on the roof plane pitch ($\theta \leq 60^o$), building edge points, which were not yet inserted/triangulated and less than .2 meters perpendicular distance from the building were being merged into the building's edge, thus distorting the building outline. The second constraint therefore confines points which only exist on a plane with a pitch ($\theta \leq 60^o$) to become merged with that existent roof plane. Most of the building structures existent in the data set considered had roof planes with pitches less than 60 degrees.

This filtering technique was found to remove the noise depicted in [Figure 12] and therefore significantly improve normal vector triangulation clustering results. Merging these noisy points' elevation (z) dimension reinforced the presence of existing roof plane clusters resulting in an improved clustering performance by Fuzzy SART.

Only the spherical coordinates of the normal vectors of roof triangles were passed to the Fuzzy SART clustering algorithm. Triangles belonging to building walls and terrain were disregarded. In order to distinguish roof triangles from all other triangles the following measures

were implemented. For all triangles, the difference in elevation between the two vertices farthest from one another in a given triangle is calculated (MX_{zdiff} in [Figure 14]). All triangles having MX_{zdiff} greater than 2 meters were isolated. Then the average, Z_{avg}, of the z-dimension (elevation) of the highest vertex in a given triangle, Z_{mx} in [Figure 14], for all triangles with $MX_{zdiff} \geq 2m$ was taken. All triangle centers must have an elevation greater than Z_{avg} in order to be considered as a candidate for a roof plane. The restriction of having the triangle differences being greater than 2 meters implements the assumption that all the buildings are greater than 2 meters or 6 and ½ feet.

3.3 Fuzzy SART Clustering

One important point referenced by Rui Xu and Donald Wunsch in the introduction of their "Survey of Clustering Algorithms" paper [46] is there is no widespread agreement up on the definition of clustering algorithms. Several somewhat vague definitions are attempted, but ultimately the nature of a clustering algorithm, and therefore its definition classifying it, seems to vary from particular application to application. Furthermore, with a given problem there's a given number of clustering algorithms particularly suited for that problem and with those clustering algorithms there's an optimal range of adjustable parameters that can be customized for that given problem. Ultimately, there is no universal clustering algorithm that will be optimal for all problem sets. Rather, algorithms are adapted to tailor to given problem specifications.

One distinctive classifier of clustering algorithms is whether or not that algorithm is dealing with supervised or unsupervised classification. In supervised classification, a set of input data, with a given dimensionality, is mapped to a discrete set of class labels via a mathematical

function dependent on the input data and a set of adjustable parameters. The values of these parameters are in turn adjusted to minimize a given risk function for mismatching input data to the wrong class. The difference then between unsupervised and supervised, is in unsupervised classification, the input data has no associated labeling scheme. In the case of the LIDAR data, there is no label attached to the data points establishing which plane or building those points belong to. Therefore, an unsupervised learning strategy, Fuzzy SART, will be used.

Adaptive Resonance Theory (ART) [11] has been popular for neural networks-based clustering. Developed by Carpenter and Grossberg as a solution to the stable convergence dilemma, ART can learn inputs stable and fast enough to have the capability to perform online training. ART2 [15] extends the binary limited applications of ART1 [10] to analog input patterns. ART3 [12] further builds on these architectures by implementing an optimized search strategy for hierarchical structures. The ARTMAP [13] system, equipped with an ARTa and ARTb-ART modules realizes a system utilized for supervised classifications. Via the tweaking of the vigilance parameter, the match tracking algorithm guarantees consistency for category prediction for both models. Larger values of the vigilance parameter yield more clusters. As the value of the vigilance parameter approaches zero, the algorithm becomes a nearest neighbor approach. Fuzzy ART (FA) [14], which is ART incorporating fuzzy set theory, has the ability for online training, stable fast learning, and atypical pattern detection. Unfortunately FA suffers from minimal robustness to noise and has the weakness of representing clusters as hyper-rectangles. Several algorithms have been proposed to circumvent these inherent weaknesses: Gaussian ART (GA) [44]; Hypersphere ART (HA) [3]; Ellipsoidal ART (EA) [2], SART [5]; Fuzzy SART [7]; and FOSART.

Coplanar triangles are defined as triangles sharing the same normal vector. The spherical coordinates of these normal vectors are what is passed to an unsupervised clustering algorithm. Fuzzy Simplified Adaptive Resonance Theory (Fuzzy SART) possesses several characteristics which make it a desirable clustering algorithm for this area of research. First, Fuzzy SART has no "mandatory" preprocessing techniques. Mandatory is put in quotes as several preprocessing techniques are presented in which considerably improve the performance of Fuzzy SART for clustering the coplanar triangles (normal vectors in spherical coordinates) in a LIDAR based TIN. Second, Fuzzy SART is not as sensitive to the input order as other versions of ART. Third, the activation function in Fuzzy SART forms hyper-spherical arcs in which Fuzzy SART uses to encode coplanar triangle normal vector patterns. Fourth, the Fuzzy SART activation function is a measure, not an estimate of the correlation of a given long term weight to an input. Fifth, the long term weights have intuitive meanings. Sixth and finally, Fuzzy SART only contains two user parameters, both with clear, intuitive meanings. An optimal choice for a clustering algorithm which will exploit the vectors being represented as spherical coordinates, is Fuzzy SART.

Baraldi and Parmiggiani's Fuzzy Simplified ART (SART) clustering algorithm [7] is presented as a combination of their SART architecture with a Kohonen-based soft learning strategy which employs a fuzzy membership function. One of the key features of Fuzzy SART which makes it choice for this clustering problem is its activation function. The activation function, also called the Vector Degree of Match function, employed in the Fuzzy SART algorithm was derived with the following objectives in minds: the activation function must be a measure (as opposed to an estimate) of the matching degree between the input and weight vectors; and the activation function output must range from 0 to 1.

The Vector Degree of Match function consists of the product of two functions: the Module Degree of Match (MDM) equation (3) and the Angle Degree of Match (ADM) equation (4).

$$MDM(T,X)=\min\{|T|/|X|,|X|/|T|\} \tag{3}$$

$$ADM(T,X)=(\pi-\alpha)/\pi \tag{4}$$

$$\alpha=\cos^{-1}(\gamma) \tag{5}$$

$$\gamma=(X\circ T)/(|X|\cdot|T|) \tag{6}$$

Both of these functions, (3) and (4), have values that range from 0 to 1 corresponding to their input component similarity. In other words, MDM approaches unity as the two vectors inputted to the function approach equal magnitude. As the inputs, the template vector T or normal vector representing the ideal direction of a given roof plane and the input vector X or vector representing a given triangle's normal vector, approach the same orientation and direction, the ADM approaches unity. The VDM is a nonlinear combination of both the MDM and ADM, such that the VDM is smaller than the smallest term between the MDM and ADM. The VDM is a dimensionless value that ranges from 0 to 1 corresponding to component similarity of its inputs. It is capable of adjusting the width of its domain of acceptance to the pair of vectors being compared.

The two user defined parameters for Fuzzy SART are τ and VDMT (the Vector Degree of Match Threshold, also called the vigilance parameter). The vigilance parameter, VDMT, restricts the accepted range of similarity in which the input vector and long term memory template vector must satisfy in order for an input pattern to be mapped to an associated neuron

(cluster). Generally, as the VDMT parameter approaches zero, Fuzzy SART becomes biased against creating new clusters. On the other hand, as the VDMT parameter approaches 1, Fuzzy SART will be biased in favor of creating more clusters. The user defined parameter τ is proportional to the time available for the cognitive system to realize the pattern recognition task.

Assuming the VDMT is held constant, the VDM function, applied to a vector pair T and X, defines a hyper-volume in bi-dimensional feature space [Figure 15].

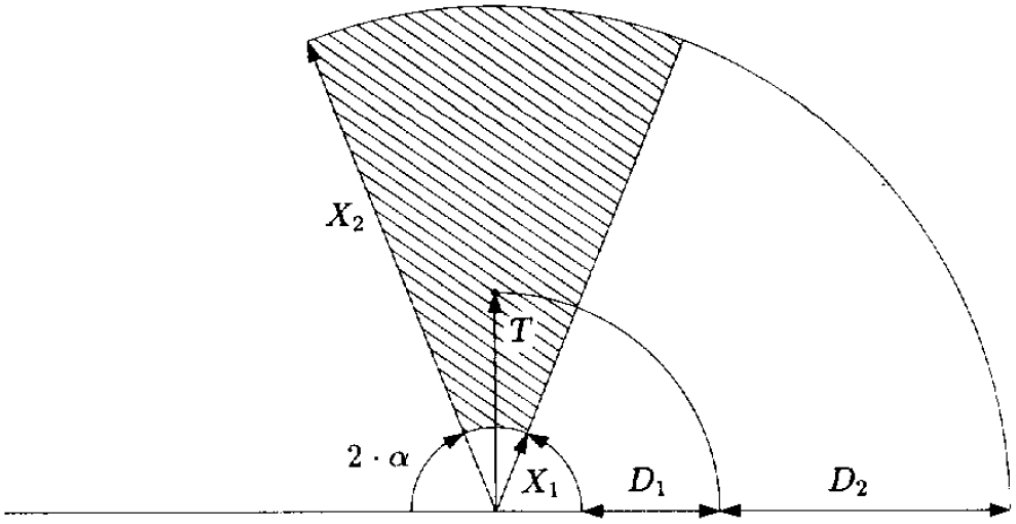


Figure 15: Hyper-Volume Acceptance for a VDM Assessment of a Vector Pair T and X

It is important to observe that D1 is less than D2. The template T represents a cluster center or in this case an established roof plane vector. The angle α is derived from equations equation (6) and equation (7).

Note that the cluster encoding regions formed are all circular about the origin. All of the original building coordinates were all positive values. It was found that the performance of Fuzzy SART for clustering the normal vectors improved if the building coordinates were shifted

such that the building was centered on the origin. Thus each dimension (longitude, latitude and elevation) was modified as follows:

$$X^d(i) = X^d(i) - \left(X^d_{\min} + \frac{X^d_{\max} - X^d_{\min}}{2} \right) \tag{7}$$

Where i represents the i-th point in the data set, d represents the dimension, and $X^d_{\max}$ and $X^d_{\min}$ represents the maximum and minimum points in those respective dimensions, respectively.

Consider the following simple case depicted in [Figure 16]. Two planes in the input dimension space have different normal vectors (N_1 and N_2). In the limit, as the two planes depicted in [Figure 16] are shifted outwards towards infinity, the two normal vectors describing those two planes converge to the same value. Obviously this is incorrect as ideally those two vectors, representing two different planes, should differ from one another.

Consider the same case depicted in [Figure 17]. While the magnitudes of the normal vectors are now equal to one another, the orientations of those vectors are 180 degrees out of phase. By translating the input space, or the planes to center around the origin, the normal vectors of the planes have effectively been further removed from one another from Fuzzy SART's coplanar clustering perspective. By further separating the normal vectors of different planes in the LIDAR TIN input space, the coplanar clustering performance of Fuzzy SART is increased.

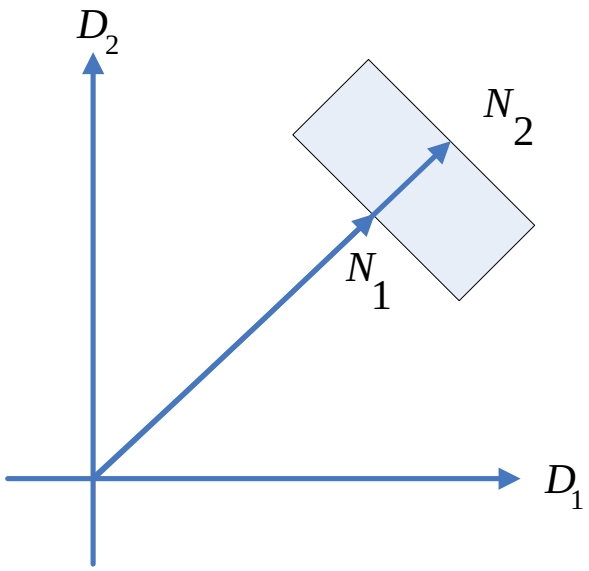


Figure 16 – Not Shifted

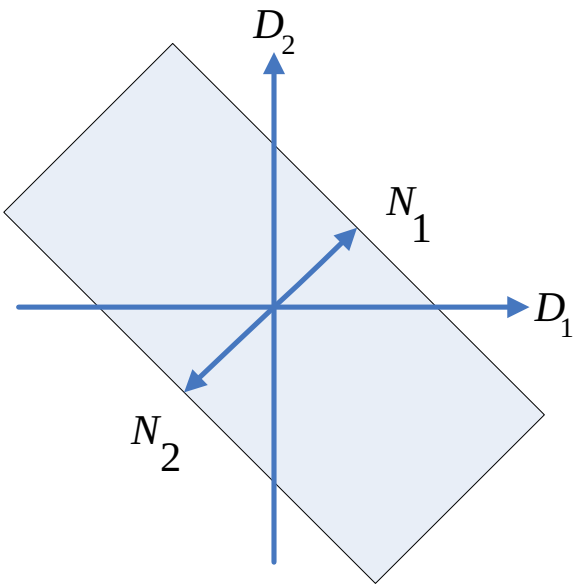


Figure 17 - Shifted

It was also experimentally determined that the coplanar clustering performance of Fuzzy SART increased if all the dimensions of the input data presented to it existed in the same range. If one dimension had larger maximum values than another given dimension, Fuzzy SART wound up assigning a higher weight of importance to that dimension when clustering data. Therefore, all dimensions of the data were scaled such that they existed within the same range (had the same maximum value).

3.4 Heuristic Procedures

The vigilance parameter for Fuzzy SART, ranging from 0 to 1, is set to 0.7. This biases Fuzzy SART to create new categories for presented Inputs rather than merge them with existing categories. This is done to minimize false positives or the incorrect grouping of a triangle which does not belong to a given long term weight or roof plane cluster. The trade off of biasing Fuzzy SART to being more conservative in accepting additional inputs to an existing cluster is Fuzzy

SART ends up creating multiple clusters to represent a single plane. Because Fuzzy SART's VDMT or vigilance parameter is set so high, Fuzzy SART is creating multiple clusters to represent a single roof plane. Therefore, a plane merging algorithm merging clusters which belong to the same best fit plane is proposed.

Only clusters containing $M_{\min}$ members and having triangle centers existing above Z_{avg} are considered for best fit plane formation, where $M_{\min}$ is defined as follows:

$$M_{\min} = \frac{TT}{C} \tag{8}$$

Where TT is the total number of triangles in the entire TIN for a given building and C is the total number of clusters formed by Fuzzy SART. 'Planar' regression, formulated by minimizing the sum of the squared error, is then done on all of the clusters passing the aforementioned restrictions. The 'planar' regression algorithm solves for the planar coefficients that will place the LIDAR points or triangle vertices belonging to a given roof plane on an approximation of that roof plane. Consider the equation of a plane:

$$a \cdot x + b \cdot y + c \cdot z + d = 0 \tag{9}$$

Solving for z:

$$z = -\frac{d}{c} - \left(\frac{a}{c}\right) \cdot x - \left(\frac{b}{c}\right) \cdot y \tag{10}$$

Then, making substitutions for the coefficients yields the following:

$$z = \beta_0 + \beta_1 \cdot x + \beta_2 \cdot y \tag{11}$$

The sum of the squared error between an estimate of z and the actual value of z based on n points (or in this case n triangle centers belonging to the considered roof plane) can be represented as follows:

$$SSE = \varepsilon_1^2 + \varepsilon_2^2 + \varepsilon_3^2 + \varepsilon_4^2 + \ldots + \varepsilon_n^2 \tag{12}$$

A picture depicting the best fit plane representing the above error terms is as follows:

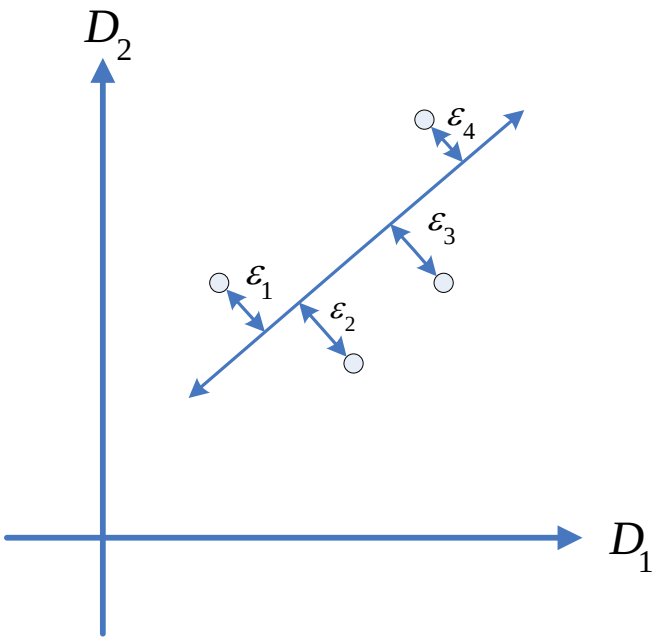


Figure 18 – Sum of Squared Error

Another way to represent the sum of the square error is look at the difference between the actual value of z and an estimate of that value.

$$SSE = \sum_{i=1}^{n} \left[z_i - \left(\beta_0 + \beta_1 \cdot x_{i_1} + \beta_2 \cdot x_{i_2} \right) \right]^2 \tag{13}$$

The derivative of the sum of the squared error with respect to each coefficient is taken:

$$\frac{\partial SSE}{\partial \beta_0} = 2 \cdot \sum_{i=1}^{n} \left[z_i - \left(\beta_0 + \beta_1 \cdot x_{i_1} + \beta_2 \cdot x_{i_2} \right) \right] (-1) \tag{14}$$

$$\frac{\partial SSE}{\partial \beta_1} = 2 \cdot \sum_{i=1}^{n} \left[z_i - \left(\beta_0 + \beta_1 \cdot x_{i_1} + \beta_2 \cdot x_{i_2} \right) \right] \left(-x_{i_1} \right) \tag{15}$$

$$\frac{\partial SSE}{\partial \beta_2} = 2 \cdot \sum_{i=1}^{n} \left[z_i - \left(\beta_0 + \beta_1 \cdot x_{i_1} + \beta_2 \cdot x_{i_2} \right) \right] \left(-y_{i_1} \right) \tag{16}$$

Then setting those derivatives equal to 0 yields the following system of equations:

$$n \cdot \beta_0 + \beta_1 \cdot \left(\sum_n x_i \right) + \beta_2 \cdot \left(\sum_n y_i \right) = \sum_{i=1}^{n} z_i \tag{17}$$

$$\beta_0 \cdot \left(\sum_n x_i \right) + \beta_1 \cdot \left(\sum_n x_i^2 \right) + \beta_2 \cdot \left(\sum_n x_i y_i \right) = \sum_{i=1}^{n} z_i \cdot x_i \tag{18}$$

$$\beta_0 \cdot \left(\sum_n y_i \right) + \beta_1 \cdot \left(\sum_n x_i \cdot y_i \right) + \beta_2 \cdot \left(\sum_n y_i^2 \right) = \sum_{i=1}^{n} z_i \cdot y_i \tag{19}$$

The above system of 3 equations with 3 unknowns or the planar 3 coefficients can therefore be solved.

All points with a perpendicular distance less than a certain threshold are then merged with the plane defined by the coefficients derived from the regression. Planar regression is then done again on the newly formed group points with a more relaxed threshold further absorbing additional points still belonging to that plane. Because the 2nd regression is based on the majority of the existent points representing the given plane, the coefficients describing the constructed plane are fairly accurate and contain few outliers. The relaxed threshold enables the merging of additional points reasonably close the defined plane. The thresholds used in this implementation were .1 and then .2 meters respectively.

CHAPTER FOUR: RESULTS

The sequential greedy insertion triangulation algorithm, the aforementioned filtering technique, the Fuzzy SART clustering algorithm and the planar regression algorithm have all been implemented and tested. The results of triangulating, filtering, clustering and then planar regression refinement of four different buildings have been presented.

Several measures presented in the previous section were proposed to improve the clustering performance on normal vectors in a TIN for identifying coplanar LIDAR points. The visual improvements these measures yield are presented in this section.

The aerial photos, corresponding to the reconstructed buildings presented later in this section, are shown in the following figures: [Figure 19], [Figure 20], [Figure 21], and [Figure 22]. These digital aerial images, of the reconstructed buildings, were captured from a digital camera mounted on a plane which flew over the terrain where these buildings exist.

Figure 19 - Building #1

Figure 20 –Building #2

Figure 21 –Building #3

Figure 22 –Building #4

The following figures contain 3-dimensional scatter plots of the LIDAR data: [Figure 23], [Figure 24], [Figure 25], and [Figure 26]. The data depicted in these scatter plots is the raw LIDAR data with no preprocessing techniques. This data represents the original, unaltered input to the 3D reconstruction algorithm. The colors in these plots denote elevation. In these scatter plots, the hotter the color, the higher in elevation the LIDAR point exists.

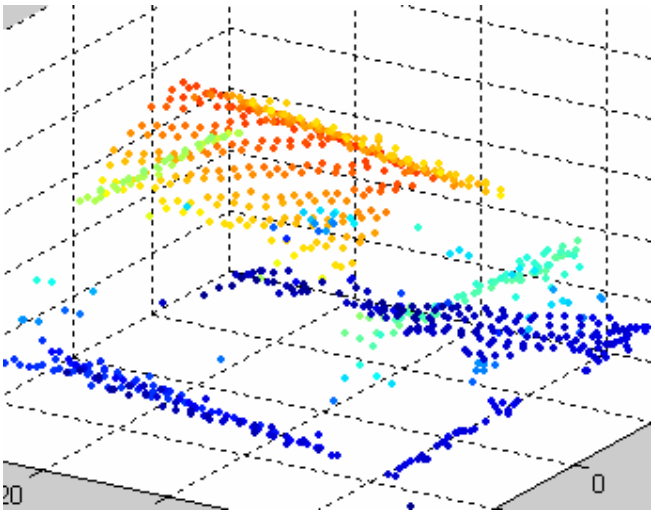


Figure 23 - Building #1

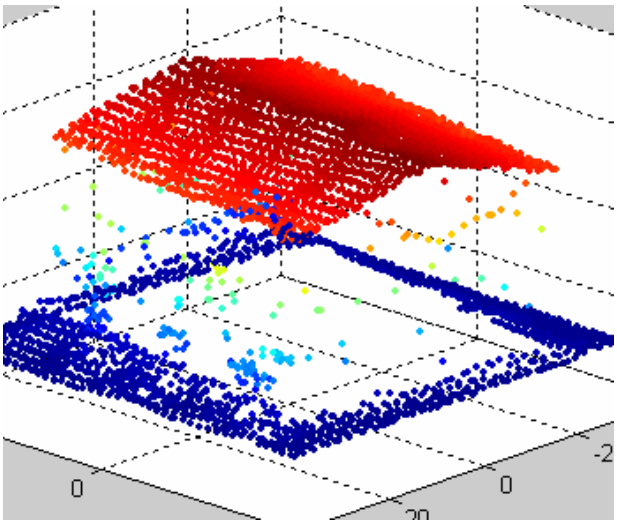


Figure 24 - Building #2

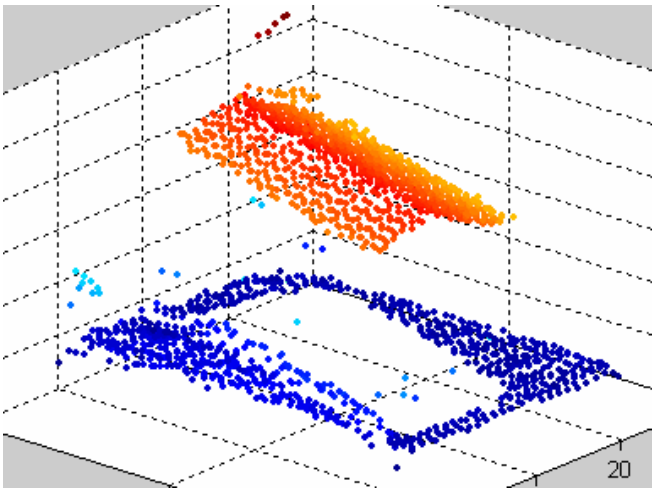


Figure 25 - Building #3

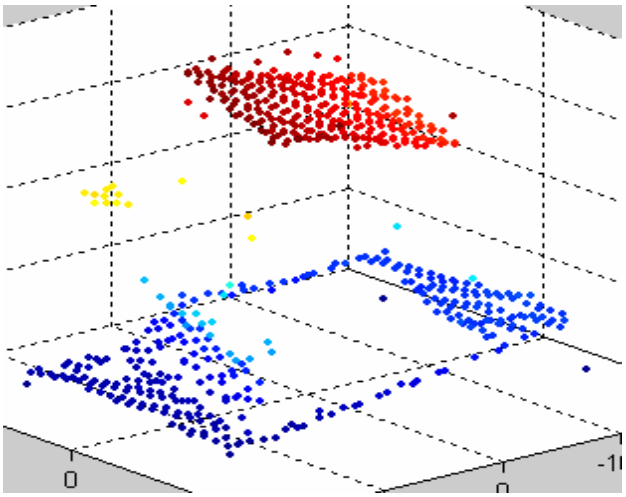


Figure 26 - Building #4

All of the subsequent figures presented in this section document the performance increase of implementing the aforementioned, proposed steps to enhance coplanar clustering on irregular TINS generated from raw LIDAR data. These figures depict triangulations of the LIDAR in which, ideally, triangles of the same color should belong to the same roof plane.

First, the four test buildings were simply triangulated and passed to Fuzzy SART as shown in the following figures: [Figure 27], [Figure 28], [Figure 29], and [Figure 30]. No logical, clear, discernable roof planes can be seen in this plot. Specifically, in the aforementioned figures, the first three buildings contain two roof planes and the last building, Building #4, contains only a single roof plane. Ideally, all of the triangles contained within these roof planes should have the same color. However, in the depicted figures, triangles of all different colors belong to all of the roof planes depicted.

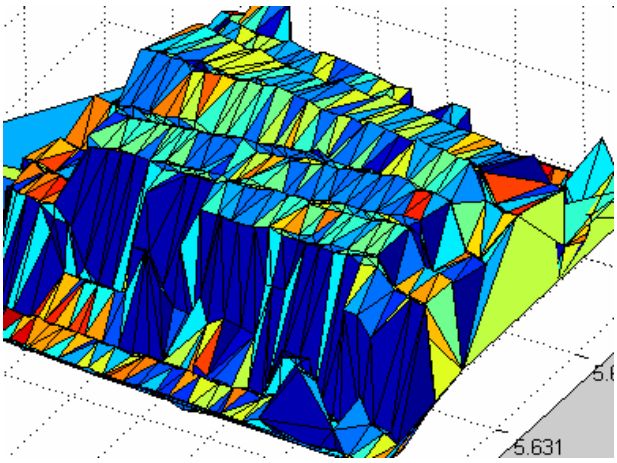


Figure 27 - Building #1

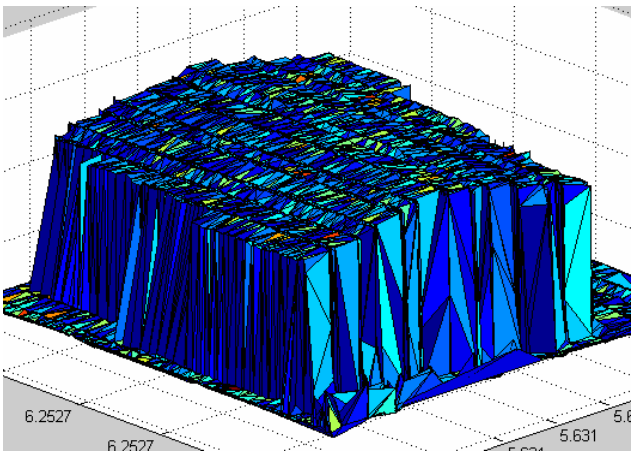


Figure 28 - Building #2

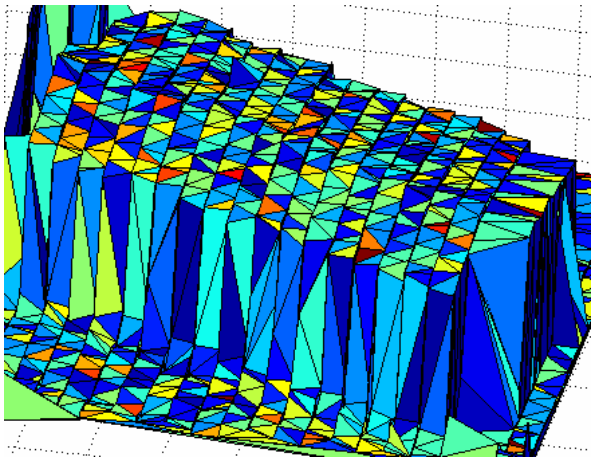


Figure 29 - Building #3

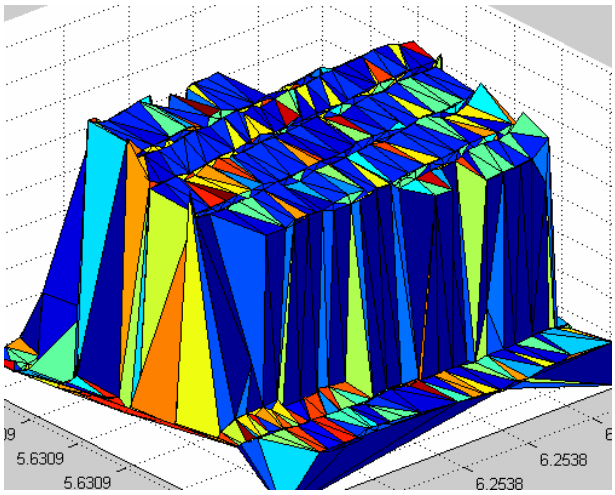


Figure 30 - Building #4

Second, the four test buildings had each of their dimensions scaled. Then, the entire data set shifted such that it was centered on the origin. Finally, the data was triangulated. The following figures present the improved results generated from the scaled, shifted and triangulated points: [Figure 31], [Figure 32], [Figure 33], and [Figure 34]. Notice in several of the plots, predominant colors representing established clusters or planes emerge. However, the results still contain a lot of stray, outlying, clusters. In [Figure 32], Building #2, a dark blue triangle represents one of the roof planes. In [Figure 33], Building #3, a roof plane is depicted by teal triangles. The planes depicted by the aforementioned triangles however contain several other triangles belonging to different clusters.

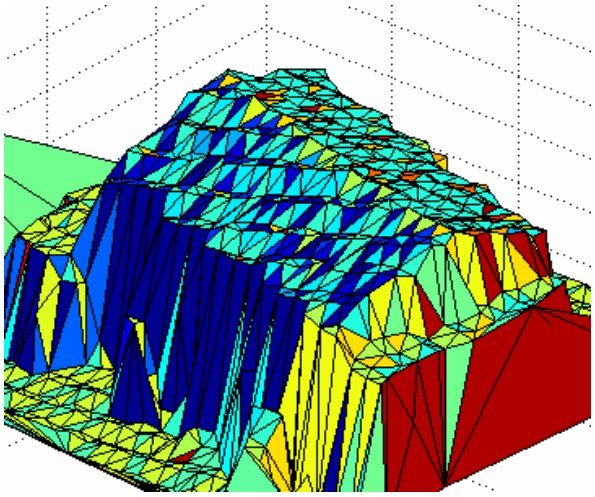


Figure 31 – Building #1

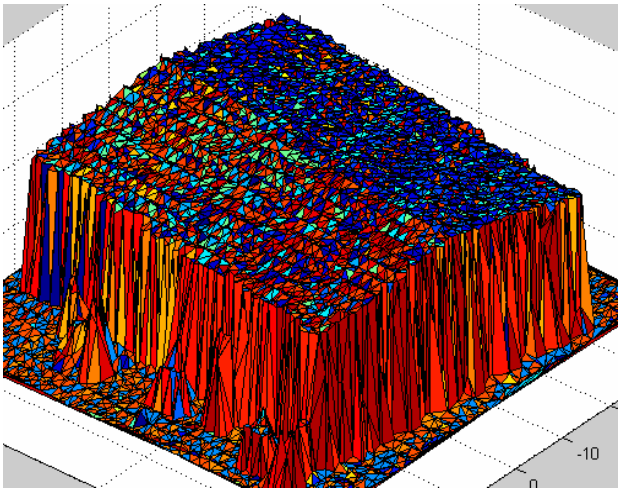


Figure 32 – Building #2

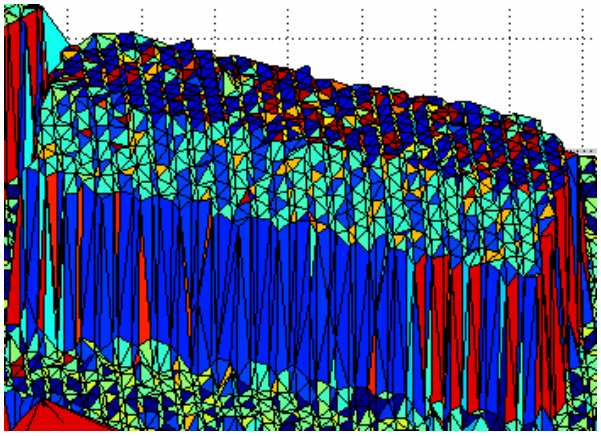


Figure 33 – Building #3

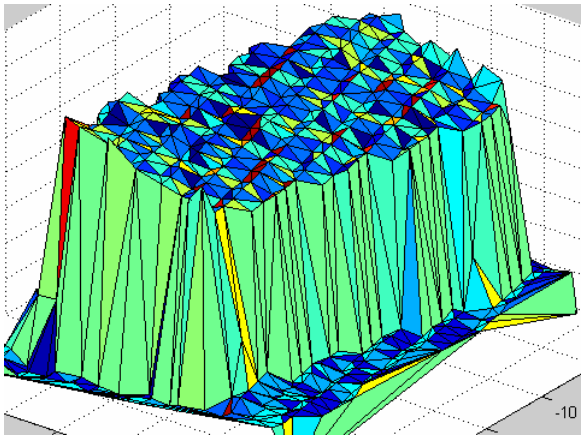


Figure 34 – Building #4

Third, the data again was scaled and shifted and this time during triangulation the aforementioned filtering technique was applied, see following figures for the improved results generated: [Figure 35], [Figure 36], [Figure 37], and [Figure 38]. Additional improvements in clustering accuracy were observed. In these plots significant clusters emerge in the data. Consider Building #2, in [Figure 36], a teal cluster represents the majority of the plane depicted on the left. A collection of blue clusters represents the plane depicted on the right. In Building #1, in [Figure 35], the left plane mostly contains blue clusters, while the right plane contains dark

blue clusters. Notice in all of the buildings depicted, although the building roof planes are mostly made up of dominant cluster colors, some other outlying cluster colors still exist in the planes. For example, in Building #2, in the plane dominated by teal colored triangles, some orange triangles still exist. In building #4, in [Figure 38], what should be a single plane is occupied by several triangles of varying colors (dark blue, light blue, orange and light green) or varying clusters. As mentioned before, the vigilance parameter in Fuzzy SART was set to 0.7. This somewhat biases Fuzzy SART against allowing triangles to join existing clusters. This results in Fuzzy SART creating multiple clusters to represent what would be ideally a singular cluster.

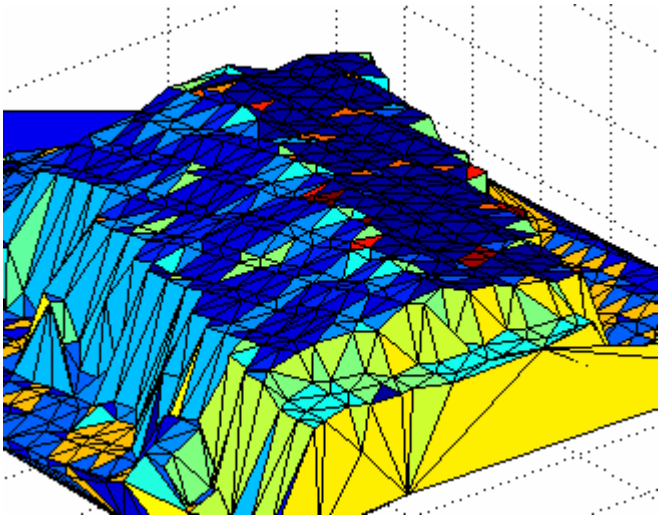


Figure 35 – Building #1

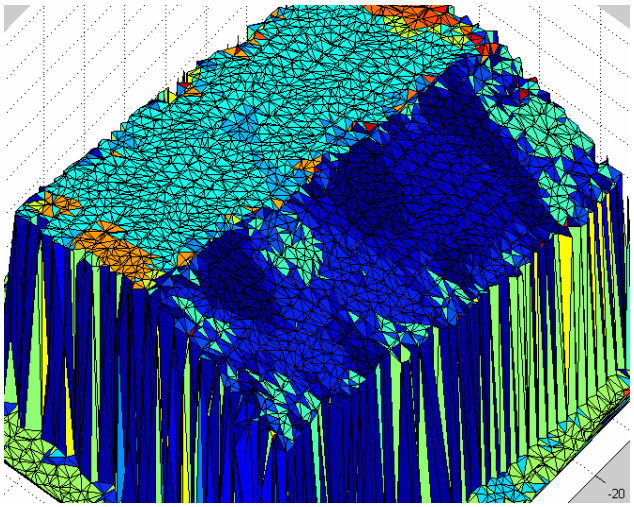


Figure 36 – Building #2

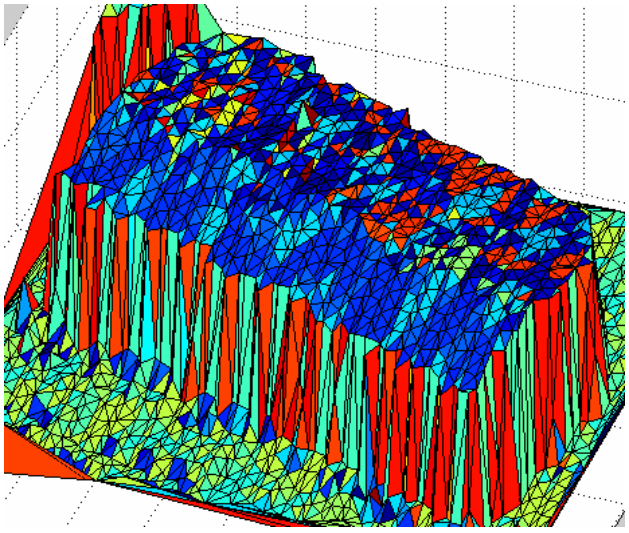


Figure 37 – Building #3

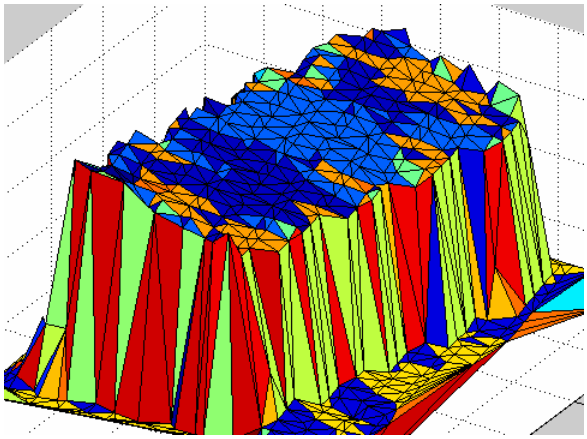


Figure 38 – Building # 4

Fourth, the planar regression algorithm operates on the filtered, scaled, shifted data to yield optimal clustering/plane segmentation results. These results are depicted in the following figures: [Figure 39], [Figure 40], [Figure 41], and [Figure 42]. Multiple clusters representing the same planes have now been merged to form single clusters representing single planes.

Furthermore, in these figures, the roof triangles have been distinguished from the triangles not belonging to a given building's roof. In the aforementioned figures, triangles of the same color which are not red belong to the same roof plane. Triangles which are red are triangles not part of a building roof plane.

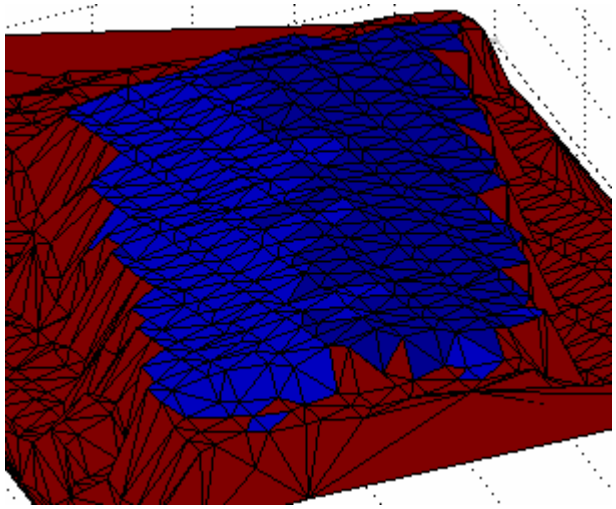


Figure 39 – Building #1

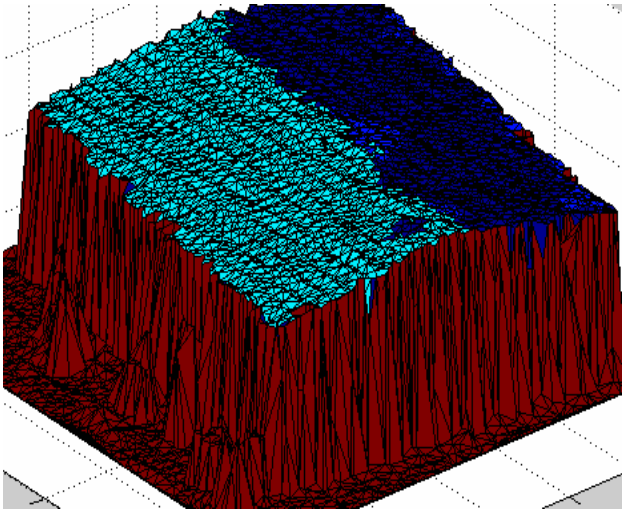


Figure 40 – Building #2

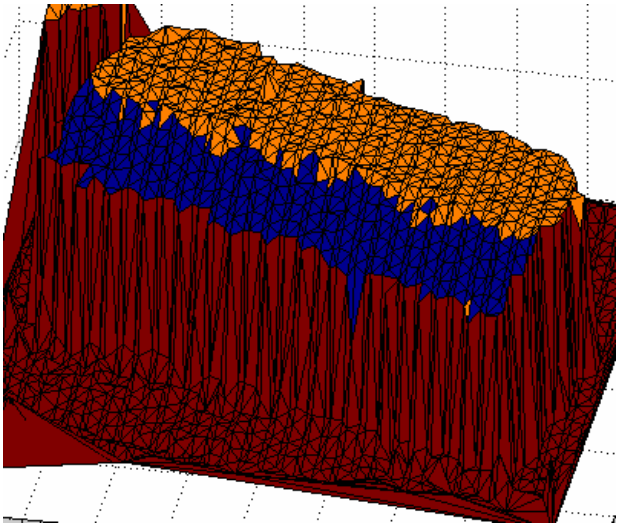


Figure 41 – Building #3

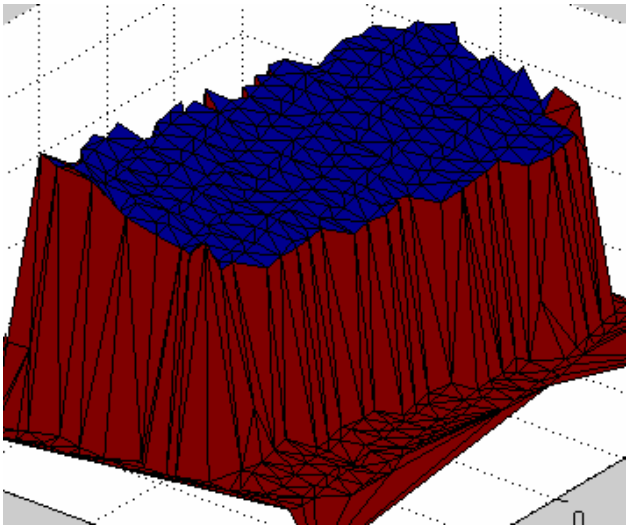


Figure 42 – Building #4

Finally, in the following figures, [Figure 43], [Figure 44], [Figure 45], and [Figure 46], the final results are presented. In these plots, all of the triangles have had their z-dimension (elevation) component merged with the plane in which they belong, resulting in a smoother depiction of the building. Furthermore, the roof lines, in which the intersecting roof planes form, have been calculated, and near by points existing within a certain threshold from those roof lines are moved onto the roof lines (resulting in straight roof edges). Furthermore, in the collection of red triangles, wall triangles were distinguished from non building triangles. All of the non building triangles were filtered to exist on a ground plane, thus removing nearby trees and cars existing close to the depicted building.

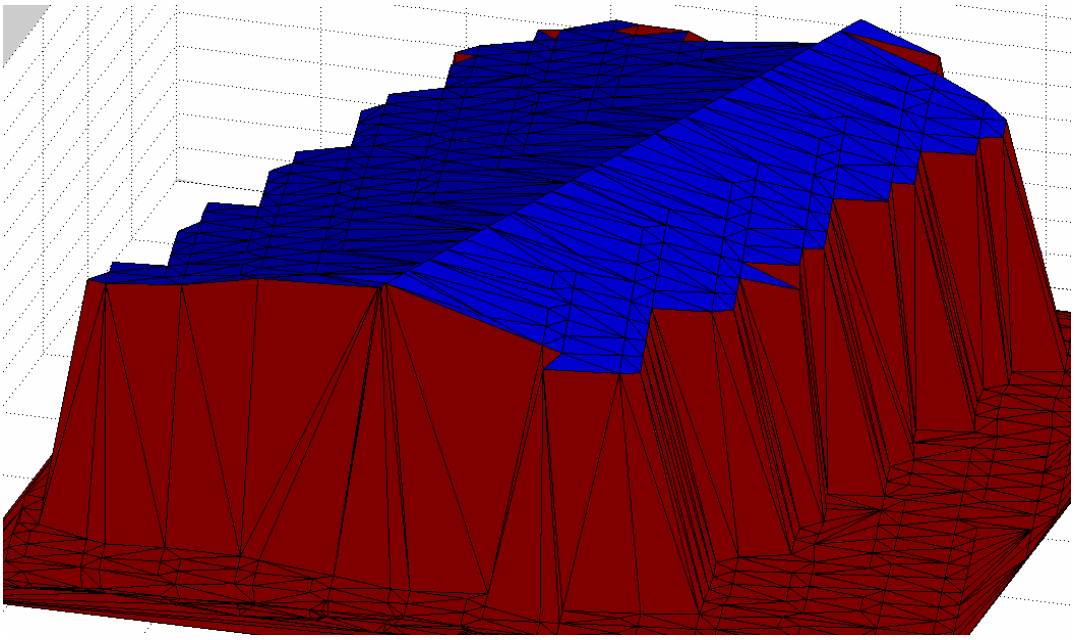


Figure 43 – Building #1

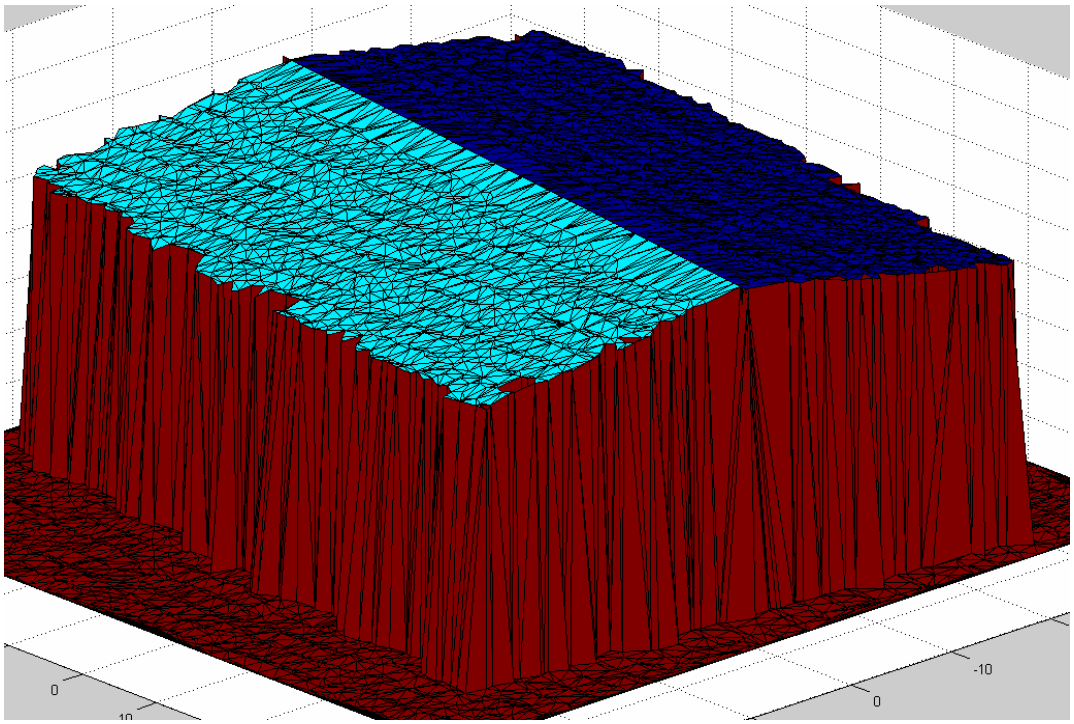


Figure 44 – Building #2

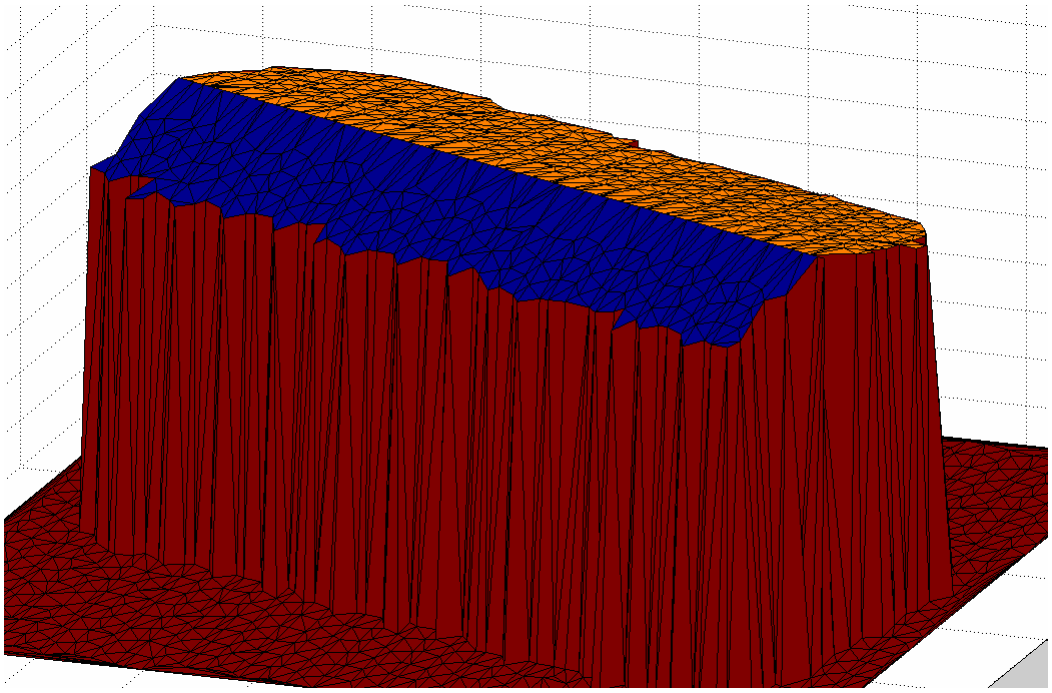


Figure 45 – Building #3

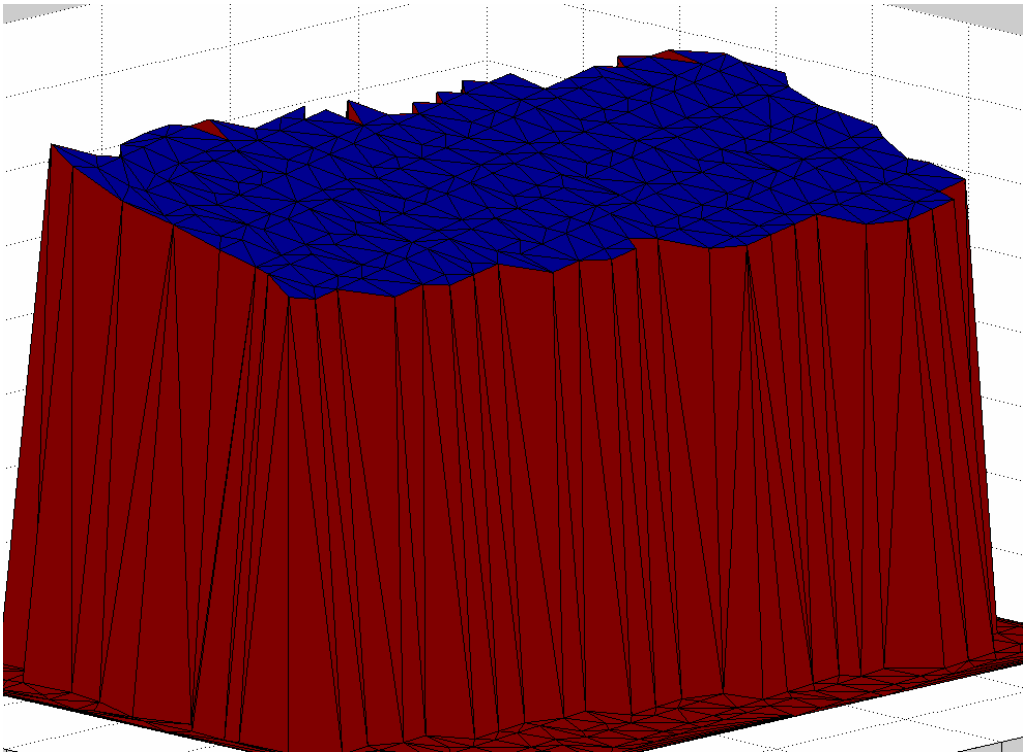


Figure 46 – Building #4

CHAPTER FIVE: CONCLUSION

5.1 Conclusions

The Sequential Greedy Insertion filtering method, the Fuzzy SART preprocessing techniques, and the various heuristic procedures presented all culminate in the development of an autonomous 3D reconstruction algorithm which works with irregularly distributed LIDAR data.

The proposed Sequential Greedy Insertion filtering method significantly improved the performance of Fuzzy SART for clustering coplanar triangles in the presented TINs derived from the LIDAR data. Equations for the pitch of an established roof plane and equations establishing criterion in which to merge points to those existing, established roof planes were formulated. The Greedy Insertion algorithm can generate variable resolution TINs by only inserting a fraction of the total points to be triangulated. The resolution can also be adjusted by stopping once the next candidate point is within a certain distance from the existing plane. However, experimental results have shown it is better to insert all the points and merge them to existing planes. By merging the points to existing planes, triangles possessing normal vectors similar to that of the normal vectors of existing planes are created. The addition of these similar normal vectors helps reinforce the presence of that roof plane as a predominant cluster during Fuzzy SART clustering. Experimentally, it was found that Fuzzy SART's performance increased when clusters had more members with similar attributes rather than fewer members with accurate attributes. Meaning, Fuzzy SART's performance increased with adding the filtered points as opposed to not adding them and generating a TIN with a decreased resolution (fewer points).

Several preprocessing techniques were proposed and developed in an effort to improve the performance of Fuzzy SART. It was found that scaling each dimension to the same maximum value resulted in Fuzzy SART assigning an equal weight to each dimension in terms of how to cluster the Input. Furthermore, it was shown that translating the input space to center on the origin separates planes farther apart from one another in the input space from Fuzzy SART's coplanar clustering perspective. Implementing these proposed preprocessing techniques resulted in noticeable improvement in the performance of Fuzzy SART when clustering coplanar triangles via the normal vectors of the TIN derived from the LIDAR data.

Several heuristic approaches were implemented to further refine the presented results. Fuzzy SART's vigilance parameter was set relatively high to avoid incorrectly merging triangles to an established roof plane in which those triangles did not belong to that plane. This resulted in creating several clusters to represent a singular plane. The largest cluster was looked at and planar regression, forming the best fit plane to approximate the cluster, was carried out. Then all other clusters which fell within a certain distance of that best fit plane were then merged to that best fit plane or cluster. Then the intersection of roof planes was calculated and all points close to that intersection or line were moved onto the line. Results show a significant visual improvement of the presented data with these aforementioned heuristic procedures implemented.

All of the aforementioned presented proposed methods result in the development of a new 3D reconstruction algorithm. To the author's knowledge, this is the first algorithm in existence which investigates the use of adaptive resonance theory to cluster the normal vectors of triangles from a LIDAR based TIN to determine co-planarity

Like other methods existent in the literature, the proposed method did have trouble accurately delineating the roof planes of small houses. This is due to the fact that the houses are

a fraction of the size of commercial buildings and contain on the order of as many as five times as many roof planes as commercial buildings. The increased number of roof planes and overall smaller size of the house make it increasingly difficult for Fuzzy SART to distinguish planar regions. Had the point spacing density been higher, 3D reconstruction for the houses may have been possible.

5.2 Future Work

This paper describes a work in progress. Exploration of further optimization of several portions of the implemented algorithm is currently underway. Future research will be conducted in the following areas. Other unsupervised learning algorithms, variations of ART and other algorithms such as single linkage clustering, ISODATA, etc. will be tested for their accuracy in delineating roof planes. The presented research work concentrates on the 3D reconstruction aspects of recreating a given building. Additional research will be done on autonomously distinguishing buildings from one another and isolating them.

APPENDIX A:
SEQUENTIAL GREEDY INSERTION STEPS

Step 1 - Initial Triangulation
Step 1a – Select the 4 outermost corner points of the LIDAR data (some points may be artificially created)
Step 1b – Perform Delaunay triangulation of selected 4 points (2 triangles formed) swapping the edges to obtain the optimal mesh
Step 1c – Mark the 4 points as used
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
Step 1d (i) – Cache the candidate point (point farthest away from triangle in z-direction) for each triangle formed
Step 2 - Largest Deviation Point Insertion
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
Step 2b – Insert the Point into the Triangulated Mesh (mark it as used)
Step 3 – Locate and Flip if Necessary
Step 3a – Locate the triangle within the triangulated mesh containing the recent inserted point
Step 3b – Split the located triangle into the necessary triangles containing the inserted point (based on the condition of insertion – [Figure 6],[Figure 7], or [Figure 8])
Step 3c - Remove the original triangle (triangles are not allowed to overlap one another)
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.
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).
Step 4– In the regions affected by insertion and flipping, recalculate the following parameters
Step 4a - The plane equations associated with the modified triangulations
Step 4b – Locate the triangles containing the unused points
Step 4c – Calculate the error between the unused point and the triangulated surface
Step 4d – For each triangle record the candidate value for the unused points (point with largest error deviation)
Step 5 - Return to step 2 and repeat if point budget or error approximation has not been met
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 B:
PLANAR COEFFICIENTS

What follows is a 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 47].

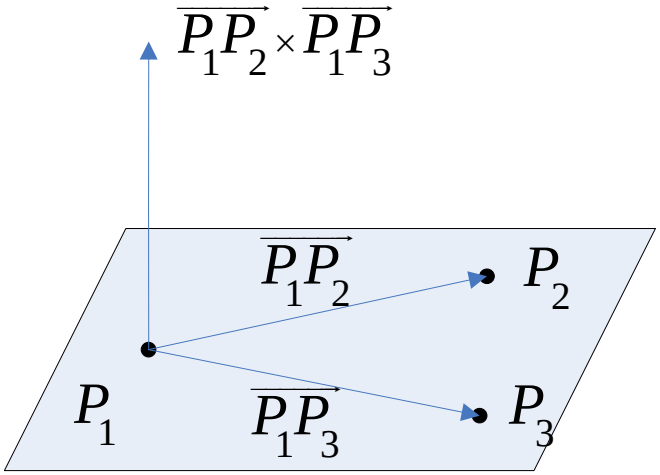


Figure 47: Normal Planar Vector Derived from Three Points

First, the two vectors $\overrightarrow{P_1P_2}$ and $\overrightarrow{P_1P_3}$, are formed from points P_1 and P_2 and P_2 and P_3 respectively via the following equations:

$$\overrightarrow{P_1P_2} = (x_2 - x_1, y_2 - y_1, z_2 - z_1) = (x_{12}, y_{12}, z_{12}) \tag{20}$$

$$\overrightarrow{P_1P_3} = (x_3 - x_1, y_3 - y_1, z_3 - z_1) = (x_{13}, y_{13}, z_{13}) \tag{21}$$

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

$$\overrightarrow{P_1P_2} \times \overrightarrow{P_1P_3} = (y_{12}z_{13} - y_{13}z_{12}, -x_{12}z_{13} + x_{13}z_{12}, x_{12}y_{13} - x_{13}y_{12}) \tag{22}$$

$$\overrightarrow{P_1P_2} \times \overrightarrow{P_1P_3} = (a, b, c) \tag{23}$$

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:

$$a \cdot (x - x_1) + b \cdot (y - y_1) + c \cdot (z - z_1) = 0 \tag{24}$$

$$a \cdot x + by + cz + (-ax_1 - by_1 - cz_1) = 0 \tag{25}$$

The ‘d’ coefficient is equal to the 4th term in equation (25). From equation (23) and equation (24), the values of ‘a’, ‘b’, ‘c’ are found to the following:

$$a = y_{12}z_{13} - y_{13}z_{12} = (y_2 - y_1)(z_3 - z_1) - (y_3 - y_1)(z_2 - z_1) \tag{26}$$

$$b = -x_{12}z_{13} + x_{13}z_{12} = -(x_2 - x_1)(z_3 - z_1) + (x_3 - x_1)(z_2 - z_1) \tag{27}$$

$$c = x_{12}y_{13} - x_{13}y_{12} = (x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1) \tag{28}$$

$$d = -(ax_1 + by_1 + cz_1) \tag{29}$$

APPENDIX C:
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.

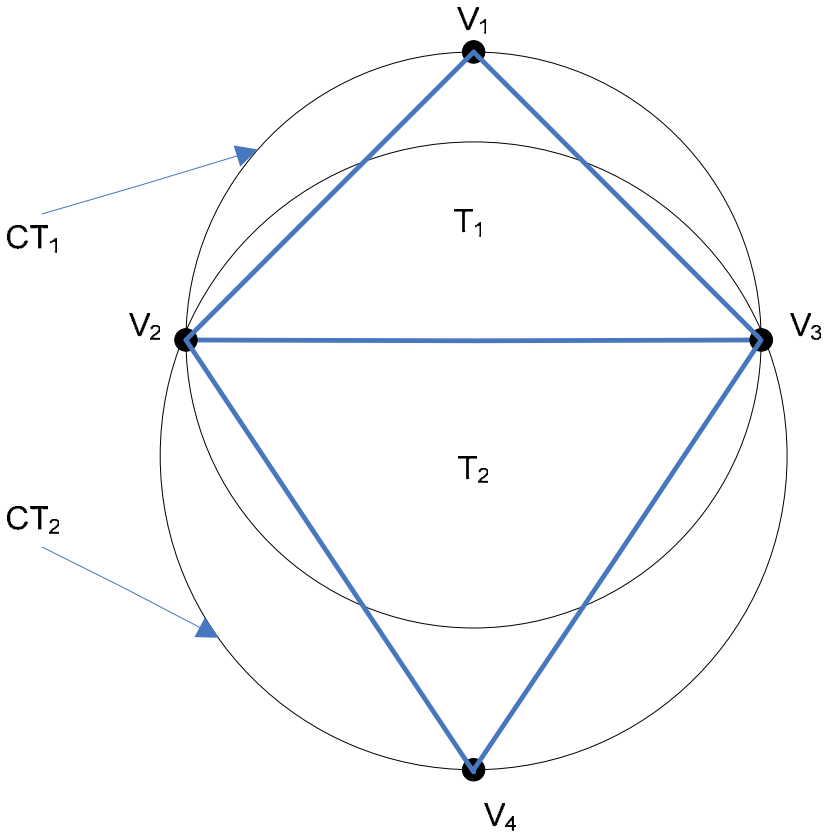


Figure 48 – Delaunay Circle Test

For example, in [Figure 48] consider the two triangles, T1 (composed of vertices V1, V2, and V3) and T2 (composed of vertices V2, V3, and V4). Because vertex V4 lies outside of circle CT1 formed by the three points V1, V2, and V3 (conversely because vertex V1 lies outside of circle CT2 formed by the three points V2, V3, and V4), the configuration of the triangles exist such that their lesser internal angles are maximized.

In order to perform this test, the center of the circle, formed by the three vertices must be calculated. In [Figure 48], consider the lines formed by vertices V1, V2, and V1, V3, 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 V1, V2, and V1, V3 are defined as follows:

$$V_1 = (x_1, y_1); V_2 = (x_2, y_2); V_3 = (x_3, y_3) \tag{30}$$

$$m_{12} = \frac{y_1 - y_2}{x_1 - x_2} \tag{31}$$

$$m_{13} = \frac{y_1 - y_3}{x_1 - x_3} \tag{32}$$

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

$$m_{12_p} = \left(\frac{-1}{m_{12}} \right) \tag{33}$$

$$m_{13_p} = \left(\frac{-1}{m_{13}} \right) \tag{34}$$

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

$$y - y_p = m \cdot (x - x_p) \tag{35}$$

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

$$y - y_1 = m_{12_p} \cdot (x - x_1) \tag{36}$$

$$y - y_3 = m_{13_p} \cdot (x - x_3) \tag{37}$$

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

$$y = m_{12_p} \cdot x - m_{12_p} \cdot x_1 + y_1 \tag{38}$$

$$y = m_{13_p} \cdot x - m_{13_p} \cdot x_3 + y_3 \tag{39}$$

Setting the equations equation (38) and equation (39) equal to one another and solving for x (the intersection of the two perpendicular bisectors or the center of the circle in [Figure 48]):

$$x_C = \frac{m_{12_p} \cdot x_1 - m_{13_p} \cdot x_3 + y_3 - y_1}{m_{12_p} - m_{13_p}} \tag{40}$$

Plug (40 back into (38 to solve for the y-coordinate center of the circle:

$$y_C = m_{12_p} \cdot x_C - m_{12_p} \cdot x_1 + y_1 \tag{41}$$

If $\left\| R_c \right\| - \left\| V \right\|_4 > 0$, 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.

APPENDIX D:
PERPENDICULAR DISTANCE FROM AN ARBITRARY POINT TO A GIVEN PLANE

Calculating the distance between a given point and the plane formed by a given triangle is derived as follows.

The projection of one vector onto another is illustrated in [Figure 49] and described mathematically via the following equations:

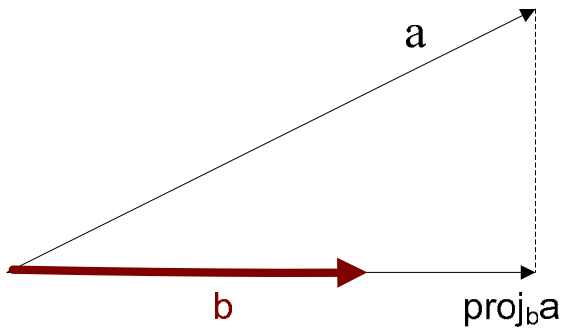


Figure 49 – Projection of a onto b

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

$$proj_a b = (a \cdot b)b \tag{42}$$

After normalizing the b vectors:

$$proj_a b = \left(a \cdot \frac{b}{\|b\|} \right) \left(\frac{b}{\|b\|} \right) \tag{43}$$

And then simplifying, the following equation results

$$proj_a b = \frac{a \cdot b}{\|b\|^2} b \tag{44}$$

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 50].

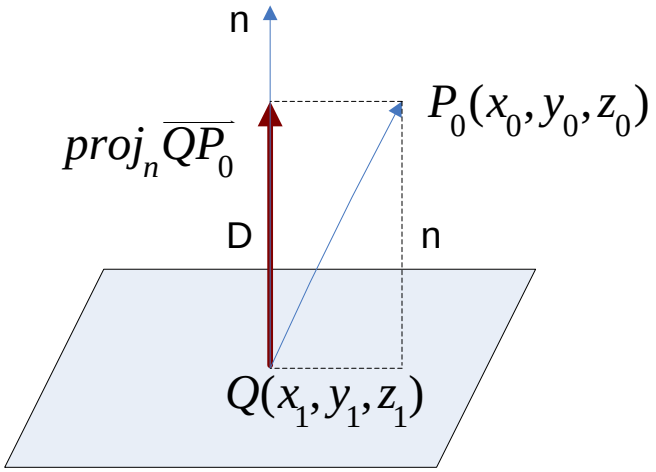


Figure 50 – Perpendicular Distance from Point to Plane

The distance between the two aforementioned points is defined as follows:

$$D = \left\| proj_n \overrightarrow{QP_0} \right\| = \frac{\left| \overrightarrow{QP_0} \cdot n \right|}{\|n\|} \tag{45}$$

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

$$\overrightarrow{QP_0} = \left\langle x_0 - x_1, y_0 - y_1, z_0 - z_1 \right\rangle \tag{46}$$

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

$$\overrightarrow{QP_0} \cdot n = a \cdot (x_0 - x_1) + b \cdot (y_0 - y_1) + c \cdot (z_0 - z_1) \tag{47}$$

The magnitude of the normal vector is as follows:

$$\|n\| = \sqrt{a^2 + b^2 + c^2} \tag{48}$$

Note equation (46), equation (47), equation (48), which after plugging into equation (45), yields equation (49):

$$D = \frac{\left| a \cdot (x_0 - x_1) + b \cdot (y_0 - y_1) + c \cdot (z_0 - z_1) \right|}{\sqrt{a^2 + b^2 + c^2}} \tag{49}$$

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

$$d = -ax - by - cz \tag{50}$$

Remember $Q(x_1, y_1, z_1)$ lies within plane F and satisfies equation (45), therefore equation (50). Plugging equation (50) into equation (49) results in the following equation defining the distance D between unused points in the original LiDAR data and the triangulated surface approximation:

$$D = \frac{\left| a \cdot x_0 + b \cdot y_0 + c \cdot z_0 + d \right|}{\sqrt{a^2 + b^2 + c^2}} \tag{51}$$

The derivation of equation (51) is described in detail because this distance is equal to the perpendicular distance of a given candidate point, about to be inserted into the LIDAR based TIN, to the TIN itself. This distance is used in the Greedy Insertion Filtering technique.

APPENDIX E:
FUZZY SART TRAINING PHASE

The steps for the Fuzzy SART training phase are enumerated below:

Step 1: Initialize User Defined Parameters
Where $VDMT \in [0,1]$ and $\tau \in [0,+\infty]$
Step 2: Initialize Algorithm Parameters
M = 0 (Initialize Number of Neurons)
EPC = 0 (Initialize Number of Epochs)
Step 3: Present Input Pattern - X_k
Step 3a: if M = 0, then do the following (M = Total Number of Neurons)
Initialize: M = 1;
Initialize: $T_M = X_k$; (T_M = Template vector)
Initialize: $t_M = 1$; (t_M = Current Age of the Neuron)
Go to Step 4
Step 3b: if $M \geq 1$ then do the following for each Neuron:
Compute the activation function (Refer to [Table 2])
Compute the membership function:

$$VDM_j^* = VDM(T_j^*, X_k) = \max\{VDM_j : j = 1,...,M\} \quad (52)$$

Step 4: Detect the Winner Template ()
Step 5: Perform Vigilance Test:

$$VDM_j^* > VDMT \quad (53)$$

Step 5a: If Vigilance Test fails, do the following:
M = M+1 (Create new Neuron)
Initialize: $T_M = X_k$; (T_M = Template vector)
Initialize: $t_M = 1$; (t_M = Current Age of the Neuron)
Go to step 7
Step 5b: If Vigilance Test passes, do the following:
Compute learning rate of the winner neuron:

$$\alpha^* = \alpha^*(u_j^*, t_j^*) = u_j^{t_j^*/\tau} \quad (54)$$

Compute variable Vector Degree of Match neighborhood Size (VDMS)
(Refer to [Table 2])

Step 6: For every Neuron compute the following:
Compute the inter-template similarity value:

$$ITS = VDM(T_j^*, T_h) \quad (55)$$

Step 6a: Check for resonance neurons:

$$ITS \geq VDMS \tag{56}$$

Step 6a(i): Check passes

Compute learning rate of resonance neuron:

$$\alpha = \alpha(u_h, t_h, t_j^*) = u_j^{[(t_h + t_j^*)/\tau)]} \tag{57}$$

Update:

$$T_h = T_h + \alpha(X_k - T_h) \tag{58}$$

Update:

$$t_h = t_h + \alpha \tag{59}$$

Step6a(ii): Check Fails

Break (skip cycle to save execution time)

Step 7:

Update:

$$T_j^* = T_j^* + \alpha^* \tag{60}$$

Update:

$$t_j^* = t_j^* + \alpha^* \tag{61}$$

Step 8: Increase Epoch => EPC = EPC+1

Step 9: Epoch Check (Does Epoch == 1?)

Step 9a: Epoch does = 1

For j = 1,…,M

$$Oldt_j = t_j \tag{62}$$

$$OldT_j = T_j \tag{63}$$

Go to Step 1

Step 9b: Epoch does not equal 1

For j = 1,…,M perform the following checks:

Step 9b(i): if $Oldt_j == t_j$ then Remove E_j from list of neurons (it was never selected)

Step 9b(iI): if $Oldt_j != t_j$ (!= means does not equal) do the following:

$$Oldt_j = t_j \tag{64}$$

$$OldT_j = T_j \tag{65}$$

If $||OldT_j - T_j|| > \delta$ then go to step 10, else go to step 1

Step 10: Algorithm Execution Finished

Table 2 – Fuzzy SART Activation Function Equations

Equation	**Equation Caption**
$VDM_j = VDM(T_j, X_k)$	**Activation Function**
$VDM(T,X) = MDM(T,X) \cdot ADM(T,X)$	**Activation Function**
$MDM(T,X) = \min\{\|T\|/\|X\|, \|X\|/\|T\|\}$	**Modulus Degree of Match**
$MDM \in [0,1]$	**Range in which Modulus Degree of Match exists**
$ADM(T,X) = (\pi - \alpha)/\pi$	**Angle Degree of Match**
$\alpha = \cos^{-1}(\gamma)$	**Alpha term defined for ADM equation**
$\gamma = (X \circ T)/(\|X\| \cdot \|T\|)$	**Gamma term defined in alpha equation**
$\gamma \in [-1,+1]$	**Range in which gamma (for ADM equation) exists**
$\alpha \in [0,\pi]$	**Range in which alpha (for ADM equation) exists**
$VDM \in [0,1]$	**Range in which VDM exists**

APPENDIX F:
FUZZY SART PERFORMANCE PHASE

For the performance phase of the algorithm, since the user defined parameters have already been optimized for the training set for the data, there is no reason to re-initialize them as they have already been selected. Step 1 of the training phase is not repeated in the performance phase.

The algorithm parameters, such as the number of Neurons and Epochs, do not need to be re-initialized. There will be only 1 epoch in the training phase of the algorithm and the number of neurons is already pre-defined based on the training phase results. Step 2 of the training phase is not repeated in the performance phase.

During the training phase, at least 1 neuron has to be established for the data to be clustered even into one large group. Therefore a conditional branch statement checking to see if at least one neuron exists is trivial. It is safe to assume that one neuron does indeed exist and therefore steps 3a and 3b of the training phase is not repeated in the performance phase.

For the performance phase, only one pattern presentation, and therefore, only one epoch occurs. Therefore, Step 8 in the training phase where we increment the number of epochs is no longer needed.

All of step 9 for the training phase is also unnecessary for the performance phase. Step 9 checks the previous template vector values to the current values to see if they fall within a certain convergence limit (DELTA variable). Since only one epoch is presented in the performance, no previous values will be available to compare with, therefore step 9 of the training phase is also not included in the performance phase.

With the above amendments in mind, the performance phase of Fuzzy SART reduces to the following:

Step 1: Present Input Pattern - X_k
Step 2: Do the following for each Neuron:
    Compute the activation function (Refer to [Table 2])
    Compute the membership function equation (52)
Step 3: Detect the Winner Template equation (52)
Step 4: Perform Vigilance Test equation (53)
Step 4a: If Vigilance Test fails, do the following:
    M = M+1 (Create new Neuron)
    Initialize: $T_M = X_k$; (T_M = Template vector)
    Initialize: $t_M = 1$; (t_M = Current Age of the Neuron)
    Go to step 7
Step 4b: If Vigilance Test passes, do the following:
    Compute learning rate of the winner neuron equation (54)
    Compute variable Vector Degree of Match neighborhood Size (VDMS)
    Compute the activation function (Refer to [Table 2])
Step 5: For every Neuron compute the following:
    Compute the inter-template similarity value equation (55)

Step 5a: Check for resonance neurons equation (56)
Step 5a(i): Check passes

Compute learning rate of resonance neuron equation (57)

Update:

$$T_h = T_h + \alpha(X_k - T_h) \tag{58}$$

Update:

$$t_h = t_h + \alpha \tag{59}$$

Step5a(ii): Check Fails

Break (skip cycle to save execution time)

Step 6:

Update:

$$T_j^* = T_j^* + \alpha^* \tag{60}$$

Update:

$$t_j^* = t_j^* + \alpha^* \tag{61}$$

Step 7: Algorithm Execution Finished

LIST OF REFERENCES

1. Anagnostopoulos, G. C.; "Novel Approaches in Adaptive Resonance Theory for Machine Learning." *Doctoral Thesis, University of Central Florida*, Orlando, Florida, 2001.

2. Anagnostopoulos, G.C.; and Georgiopoulos, M.; "Ellipsoid ART and ARTMAP for Incremental Clustering Classification." *Proceedings of the IEEE-INNS-ENNS International Join Conference on Neural Networks (IJCNN '01)*, vol. 2, pp. 1221-1226, 2001

3. Anagnostopoulos , G.; and Georgiopoulos, M.; "Hypersphere ART and ARTMAP for unsupervised and supervised incremental learning," *in Proc. IEEE-INNS-ENNS Int. Joint Conf. Neural Networks (IJCNN'00)*, vol. 6, Como, Italy, pp. 59–64., 2000

4. Arefi, H.; Hahn, M.; "A Morphological Reconstruction Algorithm For Separating Off-Terrain Points from Terrain Points in Laser Scanning Data." *ISPRS Workshop Laser Scanning*, 2005

5. Baraldi, A.; and Parmiggiani, F.; "A neural network model for unsupervised categorization of multivalued input patterns: an application to satellite image clustering," *IEEE Trans. Geosci. Remote Sensing*, vol. 33, no. 2, pp. 305-316, March 1995.

6. Baraldi, A.; Parmiggani, F.; "A self-organizing neural network merging Kohonen's and ART models", *Proceedings., IEEE International Conference on*, vol. 5, 27, pp. 2444 - 2449 vol.5, 1995

7. Baraldi, A.; Parmiggiani, F.; "Fuzzy combination of Kohonen's and ART neural network models to detect statistical regularities in a random sequence of multi-valued input patterns", *Neural Networks,1997., International Conference on*, vol. 1, 9-12, pp.281 – 286, June 1997

8. Barry, Joe; "Delaunay versus max-min solid alngle triangulations for three dimensional mesh generation" *Internal Journal for Numerical Methods in Engineering*, vol. 31, pp. 987-997,1991

9. Benediktsson, J. A.; Swain, P.H.; and Ersoy, O. K. ; "Neural network approaches versus statistical methods in classification of multisource remote sensing data," *IEEE Trans. Geosci. Remote Sensing*, vol. 28, pp. 540 – 551, July 1990

10. Carpenter, G.A. & Grossberg, S. "A massively parallel architecture for a self-organizing neural pattern recognition machine." *Computer Vision, Graphics, and Image Processing*, vol. 37, 54-115. 1987

11. Carpenter, G.A. & Grossberg, S. "Adaptive Resonance Theory." *In M.A. Arbib (Ed.), The Handbook of Brain Theory and Neural Networks, Second Edition,* Cambridge, 2003

12. Carpenter, G.A. & Grossberg, S. "ART 3: Hierarchical search using chemical transmitters in self- organizing pattern recognition architectures." *Neural Networks*, 3, 129-152, 1990

13. Carpenter, G.A.; Grossberg, S.; & Reynolds, J.H.; "ARTMAP: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network." *Neural Networks*, vol. 4, pp. 565-588, 1991

14. Carpenter, G.A.; Grossberg, S.; Rosen, D.B.; "Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system", *Neural Networks*, vol. 4, pp.759-771, 1991

15. Carpenter, G.A.; Grossberg, S.; & Rosen, D.B.; "ART 2-A: An adaptive resonance algorithm for rapid category learning and recognition." *Neural Networks*, vol. 4, pp. 493-504, 1991

16. Chen, Liang Chien; Teo, Tee-Ann; Shao, Yi-Chen; Lai, Yen-Chung; Rau, Jiann-Yeo; "Fusion of LIDAR Data and Optical Imagery for Building Modeling." *International Archives of Photogrammetry and Remote Sensing*, vol. 35, no. B4, pp. 732-737, 2004

17. Clode, S.P.; Kootsookos, P.J; and Rottensteiner, F. "Accurate Building Outlines from ALS Data." *12th Australasian Remote Sensing and Photogrammetry Conference*, 2004

18. Fujii, Kensaku; and Arikawa, Tomohiko; "Urban Object Reconstruction Using Airborne Laser Elevation Image and Aerial Image." *IEEE Transactions on Geoscience and Remote Sensing*, vol. 40, no. 10, pp. 2234-2240, 2002

19. Garland, M.; Heckbert, P. S.; "Fast polygonal approximation of terrains and height fields." *Technical Report, Department of Computer Science*, Carnegie Mellon University, 1995.

20. 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*, 2004

21. Huber, Martin; Schickler, Wolfgang; Hinz, Stefan; Baumgartner, Albert; "Fusion of LIDAR Data and Aerial Imagery for Automatic Reconstruction of Building Surfaces." *2nd Joint Workshop on Remote Sensing and Data Fusion over Urban Areas*, 2003

22. Huguet, B. Adriano; de Andrade, Marcos C.; Carceroni, Rodrigo L.; and de Araujo, Arnaldo. "*Color-Based Watershed Segmentation of Low-Altitude Aerial Images.*"

Proceedings of the Computer Graphics and Image Processing, XVII Brazilian Symposium on, pp. 138-145, 2004

23. Huguet, B. Adriano; Carceroni, Rodrigo L.; and de Araujo, Arnaldo. "Towards Automatic 3D Reconstruction of Urban Scenes From Low Altitude Aerial Images." *Proceedings of the 12th International Conference on Image Analysis and Processing*, pp. 254-259, 2003

24. Kangas, J.A.; Kohonen, T.; and Laaksonen, J. T.; "Variants of self-organizing maps," *IEE Trans. Neural Networks*, vol. 1 pp. 93-99, 1990.

25. 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

26. Kohonen, T.; "The self-organizing map", *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464-1480, Sept. 1990

27. Lee, J; "Analyses of visibility sites on topographic surfaces." *International Journal of Geographic Information Systems*, vol. 5, pp. 413-29, 1991

28. 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

29. 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

30. Mass, H; and Vosselman, G. "Two Algorithms for Extracting Building Models from Raw Lser Altimetry Data." *ISPRS Journal of Photogrammetry & Remote Sensing*, vol. 54, pp. 153-155, 1999

31. 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"*, 2001

32. Morgan, Michel; Habib, Ayman; "Interpolation of LIDAR Data and Automatic Building Extraction." *ACSM-ASPRS2002 Annual Conference Proceedings*, 2002

33. Overby, Jens; Bodum, Lars; Kjems, Erik; and Ilsoe, Pee M; "Automatic 3D Building Reconstruction from Airborne Laser Scanning and Cadastral Data Using Hough Transform." *Geo-Imagery Bridging Continents 20th ISPRS Congress*, 2004

34. Rottensteiner, F.; and Briese, Ch.; "A New Method for Building Extraction in Urban Areas from High-Resolution LIDAR Data." *IAPRSIS*, vol. 34/3A, pp. 295-301, 2002

35. Rottensteiner, F.; and Briese, Ch.; "Automatic Generation of Building Models from LIDAR Data and the Integration of Aerial Images." *ISPRS*, vol. 34, 2003

36. Rottensteiner, Franz; Trinder, John; Clode, Simone; Kubik, Kurt; "Detecting Buildings and Roof Segments by Combining LIDAR Data and Multispectral Images." *Image and Vision Computing Proceedings*, 2003

37. Rottensteiner, Franz; Trinder, John; Clode, Simone; Kubik, Kurt; Lovell, Brian; "Building Detection by Dempster-Shafer Fusion of LIDAR Data and Multispectral Aerial Imagery." *Proceedings of the 17th International Conference on Pattern Recognition*, 2004

38. Schwalbe, Ellen, Mass, Hans-Gerd, Seidel, Frank; "3D Building Model Generation from Airborne Laser Scanner Data Using 2D GIS Data and Orthogonal Point Cloud Projections." *Proceedings of the ISPRS Workshop Laser Scanning*, 2005

39. Sohn, G.; Dowman, I.; "Building Extraction Using LIDAR DEMS and IKONOS Images." *Proceedings of the ISPRS Working Group 3 workshop*, 2003

40. Straub, Bernd M.; Gerke, Markus; Koch, Andreas; "Automatic Extraction of Trees and Buildings from Image and Height Data in an Urban Environment." *International Workshop on Geo-Spatial Knowledge Processing for Natural Resource Management*, pp. 59-64, 2001

41. Suveg, Ildiko; and Vosselman, George. "Automatic 3D Building Reconstruction." *Proceedings of SPIE*, vol. 4661, pp. 59 – 69, 2002

42. Vosselman, G.; "Slope Based Filtering of Laser Altimetry Data." *International Archives of Photogrammetry and Remote Sensing*, vol 33, part B3?2, pp. 935-942, 2000

43. 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

44. Williamson, J; "Gaussian ARTMAP: A neural network for fast incremental learning of noisy multidimensional maps," *Neural Networks*, vol. 9, no. 5, pp. 881–897, 1996

45. Weed, Christopher A.; Crawford, Melba M.; Neuenschwander, Amy L.; Gutierrez, Roberto; "Classification of LIDAR Data Using a Lower Evelope Follower and Gradient-based Operator." *Geoscience and Remote Sensing Symposium*, vol. 3, pp. 1384-1386, 2002

46. Xu., R.; and Wunch, D.; II, “Survey of Clustering Algorithms”, *IEEE Transactions on Neural Networks*, Vol. 16, No. 3, pp. 645-678., May 2005

47. Zhang, Keqi; Cheng, Shu-Ching; Whitman; Dean, Shyu, Mei-Ling ;Yan, Jianhua; and Zhang, Chengcui. “A Progressive Morphological Filter For Removing Non-Ground Measurements from Airborne LIDAR Data.” *IEEE Transactions on Geoscience and Remote Sensing*, vol 41, no. 4, pp. 872-882, 2003