

**Automated Conversion of 3D Point Clouds to Solid Geometrical  
Computer Models**

A Thesis

Submitted to the Faculty

of

Drexel University

by

Matthew S. Brown

in partial fulfillment of the

requirements for the degree

of

Master of Science in Mechanical Engineering

June 2018

## Dedications

## Acknowledgements

## Table of Contents

<b>Abstract</b>	<b>xi</b>
<b>1 Introduction</b>	<b>0</b>
1.1 Motivation.....	0
1.2 Problem Statement and Approach .....	0
1.3 Related Work .....	2
1.3.1 Geographical Topography Mapping .....	2
1.3.2 Building Informational Modeling .....	2
1.3.3 Aerial Scanning for GPS Overlay .....	3
1.3.4 Semi-Automated Point Cloud to FEA modeling .....	3
1.4 Thesis Outline .....	3
<b>2 Background</b>	<b>4</b>
2.1 Collection of Point Cloud Data .....	4
2.1.1 Passive collection methods: Stereogrammetry and Structure from Motion.....	4
2.1.2 Active Methods: Laser Scanning and LiDAR .....	7
2.2 Point Cloud Pre-processing Methods.....	11
2.2.1 Registration .....	11
2.2.2 Random Sample Consensus (RANSAC) .....	12
2.2.3 Filtering .....	15
2.2.4 Down-Sampling .....	16

2.2.5	Handling Occlusion .....	17
2.3	Machine Learning for Object Recognition and Segmentation .....	20
2.3.1	Unsupervised Methods .....	20
2.3.2	Supervised Method — Convolutional Neural Network.....	23
2.4	Converting a Discrete Point Cloud to a Bounded Area Surface Mesh ...	28
2.4.1	Mesh Optimization .....	39
<b>3</b>	<b>Objective, Hypothesis, and Technical Approach</b>	<b>43</b>
3.1	Research Objective .....	43
3.2	Hypothesis Statement .....	43
3.3	Technical Approach.....	44
3.3.1	General: Zero Noise .....	44
3.3.2	General: Induced Noise .....	45
3.3.3	LiDAR: Velodyne HDL-32E .....	46
3.3.4	Segmentation Method Exploration .....	48
3.3.5	Initial Mesh Exploration .....	49
3.3.6	Optimization Steps .....	49
3.3.7	Final Mesh Verification .....	50
<b>4</b>	<b>Results</b>	<b>52</b>
4.1	Simulated Data: Zero Noise.....	52
4.1.1	Segmentation Method Comparison .....	54
4.1.2	Initial Meshing Methods .....	55
4.1.3	Optimization .....	58

4.1.4	Quality Analysis .....	66
4.2	Simulated Data: Applied Gaussian noise $\pm 2cm$ .....	67
4.2.1	Segmentation Method Comparison .....	70
4.2.2	Initial Mesh Methods.....	71
4.2.3	Optimization .....	74
4.2.4	Quality Analysis .....	79
4.3	HDL-32E LiDAR Scan Data .....	80
4.3.1	Segmentation Method Comparison .....	82
4.3.2	Initial Meshing Methods .....	83
4.3.3	Optimization .....	86
4.3.4	Quality Analysis .....	92
<b>5</b>	<b>Conclusion</b>	<b>95</b>
5.1	Finite Element Models of Compatible Meshes.....	96
5.2	Algorithm Programmability .....	96
<b>6</b>	<b>Future Work</b>	<b>98</b>
6.1	Guaranteed FEA compatibility .....	98
6.2	Informed Shape Estimation .....	98
<b>Bibliography</b>		<b>101</b>

## List of Tables

2.1	Quantification of mesh quality .....	40
4.1	Zero noise initial mesh quality .....	58
4.2	Zero noise mesh exit quality .....	67
4.3	Two centimeter noise induced initial mesh quality .....	73
4.4	Induced noise mesh exit quality .....	80
4.5	HDL-32E LiDAR scan initial mesh quality .....	85
4.6	Experimental scan exit characteristics .....	93
4.7	Experimental scan exit characteristics after manual modification.....	93
5.1	Final results comparison .....	95
5.2	Quality Metric Analysis .....	95
5.3	Algorithm flow with user input parameters.....	97

## List of Figures

2.1	Structure from Motion collection method .....	5
2.2	Structure from Motion algorithm pipeline .....	6
2.3	Structure from Motion results .....	7
2.4	Comparison of laser scanning methods .....	8
2.5	Demonstration of distance calculation via laser scanner triangulation ..	9
2.6	Single frame of low tier LiDAR sensor data collection .....	10
2.7	Example of a high tier LiDAR sensor data output .....	11
2.8	Intrinsic Shape Signature feature vectors .....	15
2.9	Effects of noise filter on simulated point cloud objects .....	16
2.10	Effects of down-sampling on simulated point cloud objects .....	17
2.11	Demonstration of occlusion .....	18
2.12	Example of range based occlusion diffusion .....	20
2.13	Comparison of divisive and agglomerative clustering methods .....	22
2.14	High level flow chart of a convolutional neural network .....	25
2.15	Common CNN non-linear activation functions .....	27
2.16	Block diagram of CNN flow .....	27
2.17	Simple example of surface mesh components .....	29
2.18	Components of a boundary representation .....	30
2.19	Example of a 2-dimensional nurbs spline .....	32
2.20	2D delaunay triangulation .....	33
2.21	Example of an edge vs. a half edge .....	36
2.22	Effects of global Laplacian smoothing with $\lambda = 0.1$ .....	38
2.23	visualization of hole filling, refinement, and fairing .....	39
2.24	Definitions for triangulation quality measures .....	40
2.25	Demonstration of voronoi relaxation over several iterations .....	41
3.1	Simulated high density lab point cloud .....	45
3.2	Simulated high density lab point cloud with 2cm noise induction .....	46
3.3	Velodyne LiDAR Operation .....	47
3.4	Flow chart of point cloud concatenation process .....	48
3.5	Simulated LiDAR scan at 100% sampling rate .....	48
4.1	Simulated laboratory point cloud data with zero noise induced .....	53
4.2	Zero noise simulated data after being filtered with voxel size of $0.1 \text{ in}^3$ , 15 minimum point neighbors at a distance of 1 standard deviation from the mean point to point distance .....	53
4.3	Comparison of unsupervised segmentation techniques on a simulated dataset .....	54

4.4	Euclidean distance segmentation with ideal parameters. ....	55
4.5	Initial meshing using a raw advancing front approach .....	56
4.6	Initial meshing using a scale space reconstruction with $S = 2$ .....	56
4.7	Initial meshing using a scale space reconstruction with $S = 4$ .....	57
4.8	Initial meshing using a scale space reconstruction with $S = 15$ .....	58
4.9	Advancing Front mesh after 200 iterations of voronoi relaxation .....	59
4.10	Scale space reconstruction $S = 2$ after 200 iterations of voronoi relaxation	60
4.11	Scale space reconstruction $S = 4$ after 200 iterations of voronoi relaxation	60
4.12	Advancing Front mesh after ODT with a 30 second clock cap .....	61
4.13	Scale space reconstruction $S = 2$ after ODT with a 30 second clock cap	62
4.14	Scale space reconstruction $S = 4$ after ODT with a 30 second clock cap	62
4.15	Advancing Front mesh after perturbation with a 30 second clock cap ...	63
4.16	Scale space reconstruction $S = 2$ after perturbation with a 30 second clock cap .....	64
4.17	Scale space reconstruction $S = 4$ after perturbation with a 30 second clock cap .....	64
4.18	Advancing Front mesh after exudation with a 30 second clock cap .....	65
4.19	Scale space reconstruction $S = 2$ after exudation with a 30 second clock cap .....	66
4.20	Scale space reconstruction $S = 4$ after exudation with a 30 second clock cap .....	66
4.21	Simulated LiDAR scan at 20% sampling rate .....	68
4.22	Simulated point cloud with 2cm magnitude gaussian noise induced .....	69
4.23	2 cm gaussian noise simulated data after being filtered with voxel size of $0.1 \text{ in}^3$ , 15 minimum point neighbors at a distance of 1 standard deviation from the mean point to point distance .....	70
4.24	Comparison of unsupervised segmentation techniques on simulated dataset with induced noise. ....	71
4.25	Initial meshing using a raw advancing front approach .....	72
4.26	Initial meshing using a scale space reconstruction with $S = 2$ .....	72
4.27	Initial meshing using a scale space reconstruction with $S = 4$ .....	73
4.28	Scale space reconstruction $S = 2$ after 200 iterations of voronoi relaxation	74
4.29	Scale space reconstruction $S = 4$ after 200 iterations of voronoi relaxation	75
4.30	Scale space reconstruction $S = 2$ after ODT with a 30 second clock cap	76
4.31	Scale space reconstruction $S = 4$ after ODT with a 30 second clock cap	76
4.32	Scale space reconstruction $S = 2$ after perturbation with a 30 second clock cap .....	77
4.33	Scale space reconstruction $S = 4$ after perturbation with a 30 second clock cap .....	78
4.34	Scale space reconstruction $S = 2$ after after exudation with a 30 second clock cap .....	79
4.35	Scale space reconstruction $S = 4$ after exudation with a 30 second clock cap .....	79
4.36	Individual LiDAR scan frame.....	80
4.37	Concatenation of 100 individual LiDAR scan frames .....	81

4.38 Concatenated LiDAR scans after noise filtering and down sampling .....	82
4.39 Euclidean clustering results on HDL-32E LiDAR scans .....	82
4.40 Advancing front reconstruction of segmented LiDAR data .....	84
4.41 Scale space reconstruction at scale $S=2$ of segmented LiDAR data .....	85
4.42 Scale space reconstruction at scale $S=4$ of segmented LiDAR data .....	85
4.43 Lloyd + Scale space reconstruction at $S=2$ of segmented LiDAR data..	87
4.44 Lloyd + Scale space reconstruction at $S=4$ of segmented LiDAR data..	87
4.45 Lloyd + ODT +Scale space reconstruction at $S=2$ of segmented Li- DAR data .....	88
4.46 Lloyd + ODT + Scale space reconstruction at $S=4$ of segmented Li- DAR data .....	89
4.47 Lloyd + ODT + perturb +Scale space reconstruction at $S=2$ of seg- mented LiDAR data .....	90
4.48 Lloyd + ODT + perturb + Scale space reconstruction at $S=4$ of seg- mented LiDAR data .....	91
4.49 The full suite of optimization steps on the LiDAR reconstruction at $S=2$ . .....	92
4.50 The full suite of optimization steps on the LiDAR reconstruction at $S=4$ . .....	92
4.51 Modified optimized surface mesh at scale space 2. ....	94
4.52 Modified optimized surface mesh at scale space 4. ....	94
5.1 FEA loading results on the zero noise beam .....	96
6.1 Flow chart of informed shape estimation algorithm .....	100
6.2 Demonstration of the Informed Shape Estimation algorithm on a sim- ulated point cloud.....	100

**Abstract**

Automated Conversion of 3D Point Clouds to Solid Geometrical Computer Models

Matthew S. Brown

Antonios Kontsos, Ph. D.

While current engineering design and construction methods include computer aided design drawings in addition to simulation, analysis, and several visualization tools, there is still the need to create computer models of the vast number of structures found in the built environment. With the slew of point cloud collection tools now readily available, it is not difficult to create point cloud representations of real objects. However, it is a rare occurrence that raw clouds are isolated enough and clean enough to be directly converted to bounded, non-zero volume computer aided design models. In this context, this thesis proposes a method and an associated toolbox to utilize machine learning in conjunction with meshing techniques to autonomously segment raw point cloud data and reconstruct the resulting segments into computer aided design compatible surface meshes by applying a series of meshing and optimization algorithms to the point cloud. The result is a highly adjustable, generalized algorithm to convert raw point clouds of variable resolution, accuracy, and occlusion level to create uniform meshes that are computer aided design compatible.

## 1. Introduction

### 1.1 Motivation

Currently, methods to collect and convert point cloud data into usable models (CAD, finite element, etc.) are scarce. Until the recent past, instruments capable of acquiring dense and accurate point cloud datasets were rare, expensive, or non-versatile. The recent boom in automated vehicles has caused the sensing and localization field to grow exponentially, allowing a host of highly effective instruments and methods to become easily obtainable by consumers. The common usages for this equipment is for automated awareness and unmanned navigation, but this data can also be used to make accurate measurements of structures and objects in real space.

### 1.2 Problem Statement and Approach

Currently, the methods available to take discrete point scans of an object from variable collection devices – all with varying accuracy and resolution – and convert the result into utilizable CAD models are either expensive to obtain, non-versatile, or split throughout a number of different softwares. The entire process involves five main steps: Data collection, data pre-processing, point cloud segmentation, initial meshing, and mesh optimization. The algorithm proposed in this thesis is therefore designed to handle all of these steps in a single pipeline.

The data collection phase encompasses the actual scanning of the object of inter-

est, whether that be via laser scanning or camera imagery, or any method capable of returning a point cloud. The pre-processing step involves converting any datatype that is not already explicitly in cartesian coordinates to cartesian format, concatenating all of the relevant data collected, and filtering the result to remove overtly noisy information.

The segmentation step is a more advanced filtering step, to isolate the target object from its surrounding environment so multiple objects in the point cloud are not meshed together in the next step. The initial meshing step deals with determining how the points in the target cloud are related to one another on the surface of the object. The final step, optimization, handles remedying any non-ideal artifacts created by the initial meshing algorithm.

This thesis provides a quantification of the effectiveness of the methods used, as well as the relationship between descriptors of the datasets and the success of the methods. The proposed steps in the algorithm are applied to three sets of point cloud data with different properties. It is also the creation and definition of a tool to fluidly and easily modify the methods and parameters inside of the proposed pipeline.

The first dataset is a true control set of simulated data designed to meet the resolution of the LiDAR device used for the true experimental portion. The simulated dataset has no induced noise or occlusion, and is the ideal case for the meshing algorithm. The second dataset is identical to the control set in resolution, with the exception of a significant amount of noise induced into the dataset. This dataset provides a benchmark to test the proposed algorithms robustness to noise. The third and final dataset is true experimental data, collected with a LiDAR device. This data acts as

the sub-ideal case for occlusion, and acts as a benchmark for the algorithms ability to fill information gaps due to occlusion.

All three tests have an identical target object. Analysis of the effectiveness of the pipeline is made by calculating the minimum mesh quality values and attempting to convert the mesh into a solid representation format.

### 1.3 Related Work

Surface reconstruction from a 3D point cloud is not a novel problem. In the past, groups have developed meshing algorithms for digital art replication, geographical topology analysis, and — more recently — structure health monitoring. None of these processes, however, provide a general method to automate the entire pipeline between point-cloud collection and CAD geometry.

#### 1.3.1 Geographical Topography Mapping

In 2014, Sebastian Siebert implemented LiDAR devices mounted to UAVs to provide 3D mapping of earthwork projects for surveyors [1].

#### 1.3.2 Building Informational Modeling

[2]

### **1.3.3 Aerial Scanning for GPS Overlay**

### **1.3.4 Semi-Automated Point Cloud to FEA modeling**

[3] [4] [5] [6]

## **1.4 Thesis Outline**

The structure of the thesis is the following:

First, all of the underlying methods used in the proposed algorithm are described in detail in the background section. After this, the hypothesis statement and research objectives are described. A detailed description of the technical approach used to make informed decisions on the validity of the hypothesis is enunciated directly after this. Finally, the results of the proposed algorithm operating within the space defined by the technical approach are shown. In the conclusion, a detailed analysis of these results is made.

## **2. Background**

### **2.1 Collection of Point Cloud Data**

There are many ways of collecting point cloud data, ranging from passive methods where the collection tool retrieves information about the environment without probing it, such as stereogrammetry and structure from motion, to active methods where the instrument is applying energy to probe its environment directly. LiDAR devices, laser scanners and ultrasonic sensors fill this category. Each collection technique has its own set of parameters, accuracy ratings, and speed of collection / calculation.

#### **2.1.1 Passive collection methods: Stereogrammetry and Structure from Motion**

Stereogrammetry and Structure from Motion have become industry standard methods for extracting 3-dimensional point cloud data using images from cameras. These methods are popular due to the quality of results compared with the low cost of the sensing equipment (1-2 cameras). Both methods are similar in operation, with the difference being whether or not they are precalibrated.

## Stereogrammetry

### Structure from Motion

Structure from Motion is a method of extracting 3-dimensional data from a series of images by determining the relative translation between distinctive feature points in images. The underlying foundation of this method is the disparity image. Objects in the foreground of a viewpoint move much more drastically than objects in the background. This can be envisioned by imaging a passenger in a car driving by a mountain. After some timestep, the tree 10 feet away from the car has changed drastically in view location, but the mountain that is much farther away looks almost exactly the same [7].

Using the disparity between feature points in an image, Structure from Motion provides a relative distance relationship, scaled to pixel size. This of course requires many overlapping images of the same object at different views, so distinctive features are repeated in the views. Figure 2.1 shows a demonstration of the amount of overlap expected in an input image set.

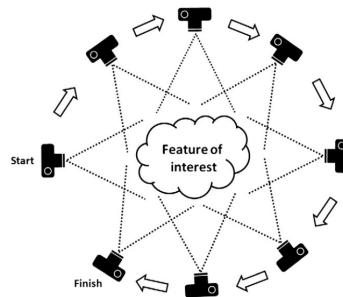


Figure 2.1: Structure from Motion requires collecting a series of overlapping images to generate point clouds [8].

Structure from Motion has a number of steps in the pipeline, which can be seen in Figure 2.2. At a high level, the pipeline consists of first collecting the image data, detecting key features, matching key features between images and determining their relative locations (bundle adjustment), and transforming point clouds of disparity pairs to the global reference frame. The last step requires establishing a global reference frame – typically the results of the first disparity set become the global frame. Subsequent pointsets are scaled to match the global set by forcing the scale of the current pointset to match the scale of the previous pointset using similar feature points between sets [8].

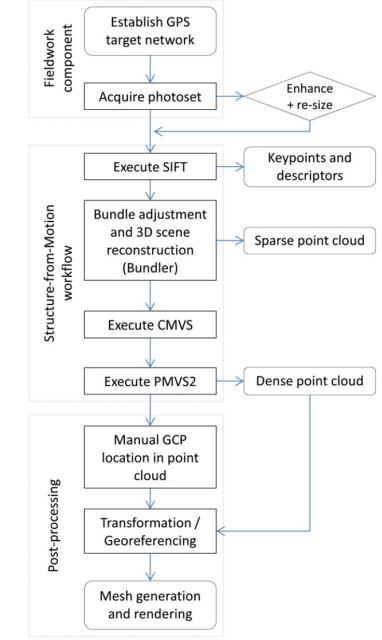


Figure 2.2: A broad overview of the Structure from Motion pipeline [8].

After the calculation of the sparse point cloud, the algorithm in Figure 2.2 fits the original images to the cloud by creating a crude surface mesh with the sparse pointset

and superimposing the images to the surface. This is outside the scope of what the envisioned return of Structure from Motion data collected for this thesis.

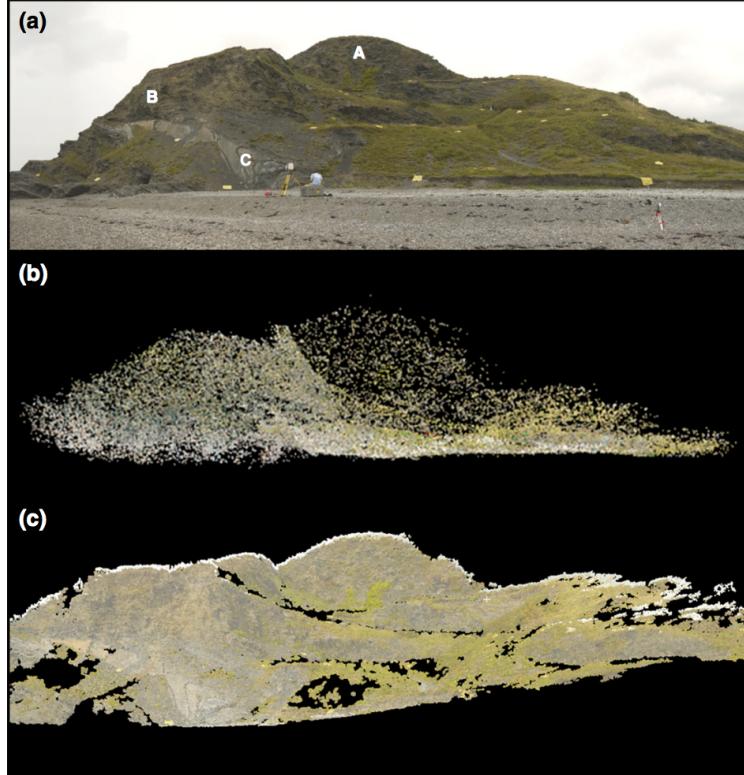


Figure 2.3: Structure from Motion point cloud based on a series of panoramic images taken from the ground. a) panorama of the scene b) sparse point cloud reconstruction c) image overlay [8].

### 2.1.2 Active Methods: Laser Scanning and LiDAR

Active collection methods are those that directly probe the environment via some form of energy output. If the scanner itself is non-static, as is the case for laser-scanners and light detection and ranging (LiDAR) devices, the sensor also returns

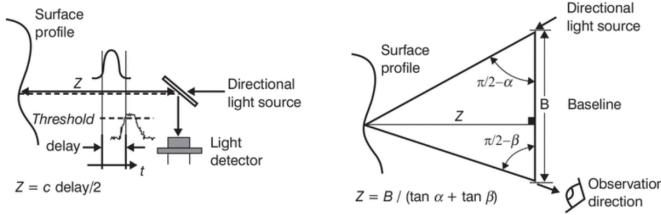


Figure 2.4: comparison of laser scanning methods. The left image represents time of flight measurement, and the right represents triangulation measurement [9].

some timestamped orientation parameters to provide a local reference frame.

## Laser Scanning

Devices classified as laser scanners are capable of capturing and recording the geometry of 3-dimensional surfaces in dense point clouds by taking full-field optical measurements. The types of measurements these devices take can be broken down into two categories: Light transition time measurement, and triangulation [9]. Figure 2.4 demonstrates the difference between how the two measurements are collected.

The first method, time of flight measurement, is the method used in LiDAR devices. LiDAR devices are described in more detail in the next section.

Time of flight measurements make use of a fundamental property of light: its propagation speed. Because light travels at a constant speed in any given medium, laser scanners can calculate proximity to an object by recording the amount of time it takes for an emitted laser beam to return to the sensor after being reflected. The equation to calculate this distance is shown below, where  $\rho$  represents distance,  $c$  represents the speed of light in a vacuum,  $n$  represents the refractive index of the

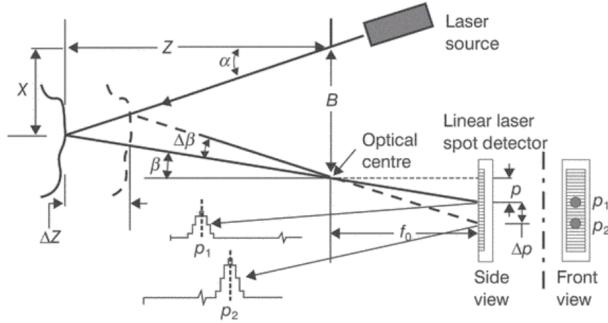


Figure 2.5: Demonstration of distance calculation via laser scanner triangulation [9].

propagation medium, and  $\tau$  represents the time between laser emission and echo return.

$$\rho = \frac{c}{n} \frac{\tau}{2} \quad (2.1)$$

Time of flight measurement techniques are typical for long range applications, so are commonly used by aircraft localization and terrestrial scanning purposes. Triangulation-based measurements, on the other hand, are optimal for systems designed to measure distances below five meters. Triangulation-based methods work by the same principle as human eyes to determine distance. A laser is first projected onto the surface of interest. The reflected light – now scattered – is collected by a receiver which calculates the distance between the scattered rays. Using basic trigonometry, and knowing the relative angles between the laser source and the receiver, the distance from the laser source can be derived. Figure 2.5 shows a visualization of the process with a single beam.

This uncertainty in this method is mainly due to the estimation of the position of the laser reflections on the receiver. Because the method is a direct range value, and the typical usage cases involve small scales, this method often yield highly accurate results. The downside is a lack of mobility, and a low collection resolution [9].

## LiDAR

LiDAR scanners vary greatly in accuracy, complexity, and resolution. At the lower end of the spectrum, devices contain one laser and return anywhere from 10-100 points per second along a horizontal plane. The outputs for these LiDAR sensors is timestamped returns of  $z$ -axis rotation angle  $\theta$  and a distance magnitude.

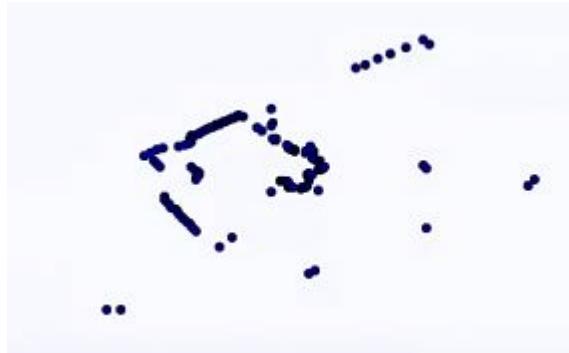


Figure 2.6: Data collected from a single rotation of the Scanser Sweep LiDAR device.

High end LiDAR devices have a series of lasers distributed through a range of angles. This allows the LiDAR to provide a much more comprehensive pointset of its scanned environment. These sensors can provide anywhere between 100,000 and 1.5 million points per second. The typical return for high end sensors is a package of ordered data: the timestamp pertaining to data acquisition, the rotation angle of the

beam reflector, and a vector containing the distance returns of each sensor. Knowing the offset angle of each laser, the dataset can be easily converted into spherical coordinates, and recast in cartesian coordinates. These sensors typically provide a distance uncertainty of  $\pm 2\text{cm}$  or lower.

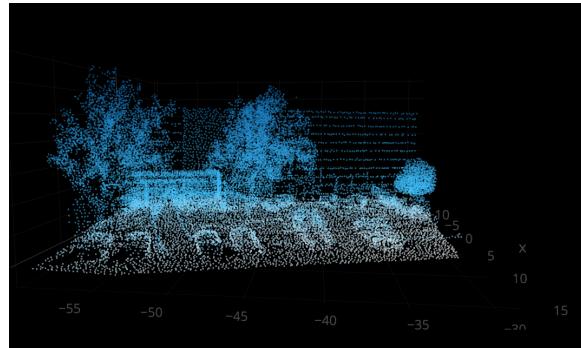


Figure 2.7: Data collected from an HDL-32E LiDAR device [10].

## 2.2 Point Cloud Pre-processing Methods

### 2.2.1 Registration

#### Position Data

Point cloud collection devices are typically mobile, and often contain a number of localization measures. The standard localization methods and instruments used are a GPS, Inertial Measurement Units, and some form of computer vision. When data from these elements are fused via either a Kalman Filter [11], particle filter [12], or any other sensor fusion method, all six parameters of the global position of the collection device can be accurately obtained (x, y, z, pitch, roll, and yaw).

When onboard position data is collected, concatenating – in layman’s terms, adding – individual collection frames becomes a matter determining the transformation matrix between frames using the orientation and position of the UAV, and applying small corrections based on a combination of random sample consensus techniques (RANSAC) and Intrinsic Shape Signatures (defined in the next section).

### 2.2.2 Random Sample Consensus (RANSAC)

Random sample consensus, or RANSAC, is an iterative parameter estimation algorithm designed to fit a model to data containing outliers. It operates on the principle that data is comprised of two groups: inliers, which can be properly explained by a set of model parameters, and outliers, which do not fit the system model. Outliers can be viewed as products of noise or some form of incorrect measurement techniques.

The RANSAC algorithm can be broken down into two steps. First, a randomly selected subset is sampled from the dataset. A model – defined by the user – is fitted to the sample set. After fitting the model to the sample set, the rest of the dataset is checked against the model to determine the total error between the model parameters and the dataset. These two steps are repeated iteratively until convergence criteria for error defined by the user is met [13].

Standard termination criteria for the algorithm is based on the minimum number of samples required to ensure with confidence level,  $\eta_0$ , that one of the selected subsets is free of outliers. Given some ratio of inliers,  $\epsilon$ , the probability of a sample set being free of outliers is  $\epsilon^m$ , where  $m$  represents the size of the sample set. The probability of selecting  $k$  outliers in the dataset is then defined as  $(1 - \epsilon^m)^k$ . The number of

samples then, to assure an outlier free set with probability  $\eta_0$ , can be defined using the following equation:

$$k = \frac{\log(1 - \eta_0)}{\log(1 - \epsilon^m)} \quad (2.2)$$

This equation is not directly solveable, as  $\epsilon$  is not a pre-defined value. The remedy for this is an continuous update of the value of  $\epsilon$  based on the updating model parameters. At each iteration, the inlier ratio is set to the lowest ratio encountered by the model [14].

## Intrinsic Shape Signatures

To stitch individual frames together, distinctive, repeatable features from each frame are found, and the most likely transformation between the frames is calculated via RANSAC estimation. There are numerous ways to classify distinctive features, but the results in this thesis focus on Intrinsic Shape Signatures due to its reliability and computational efficiency. An intrinsic shape signature consists of two things:

1. An intrinsic reference frame.
2. A highly discriminative feature vector encoding the 3D shape characteristics.

**Intrinsic Reference Frame Calculation** To calculate the orientation of the Intrinsic Reference Frame, the relationship of a point,  $p_i$ , with the points inside of its neighborhood is calculated. The neighborhood is described by any points within distance  $r_{density}$  to interest point,  $p_i$ .

1. Compute a weight for each point  $p_i$  inversely related to the number of points within 2-norm distance  $r_{density}$ :

$$w_i = \frac{1}{\sum |p_j| \text{ for } |p_j - p_i| < r_{density}} \quad (2.3)$$

This weight is used to compensate for uneven sampling of the 3D points, so that points at sparsely sampled regions contribute more than points at densely sampled regions.

2. Compute a weighted scatter matrix  $cov(p_i)$  for  $p_i$  using all points  $p_j$  within distance  $r_{frame}$ :

$$cov(p_i) = \sum |p_j - p_i| < r_{frame} \frac{w_j (p_j - p_i)(p_j - p_i)^T}{\sum |p_j - p_i| < r_{frame} w_j} \quad (2.4)$$

3. Compute the covariance matrix eigenvalues in order of decreasing magnitude and their resulting eigenvectors.

4.  $p_i$  is now the origin of the intrinsic frame, with  $e^1$ ,  $e^2$ , and their cross product as the  $x$ ,  $y$ , and  $z$  axes, respectively [15].

**3D Shape Feature Extraction** The goal of the extraction is to create a view invariant “feature” vector providing some unique qualities about the point relationships within the intrinsic reference frame. At each point in the point cloud, or in increments of voxel stride size  $s$ , a sphere of some desired radius  $r$  centered at  $p_i$  is created and divided into 66 distinct partitions in angular space  $(\theta, \psi)$ . A distinctive

feature vector with 66 values – 66 is standard for this procedure – is then computed by summing the radial distances  $\rho_i$  in each bin [15].

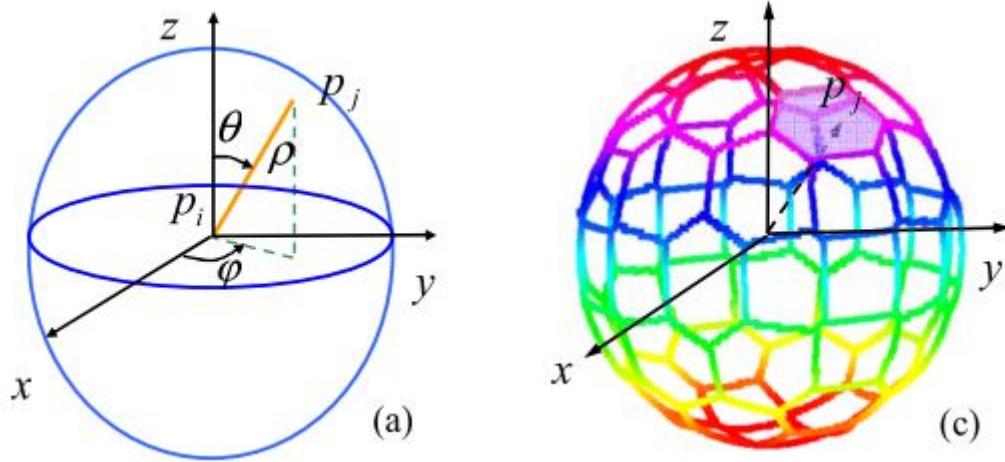


Figure 2.8: Feature vector calculation via spherical bin decomposition [15].

### 2.2.3 Filtering

To minimize the amount of noise in the resulting dataset, a statistical algorithm requiring each point to have  $k$  neighbors within  $d$  standard deviations from the mean density radius of the cloud is applied to the dataset. This allows for controlled outlier removal, and a smoother cloud with fewer sharp edges.

$$P_x = p_i \mid \sum_{j=1}^n |p_j - p_i| \leq (r_{density} + d) \geq k \quad (2.5)$$

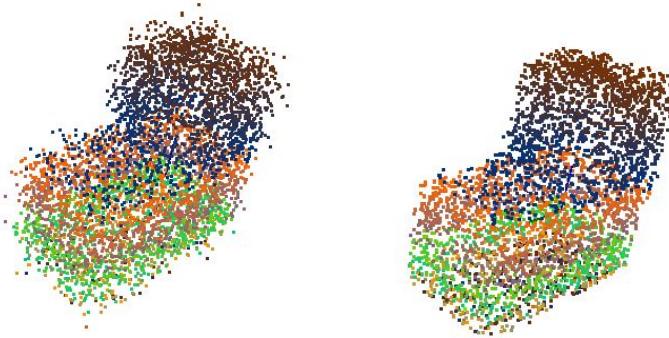


Figure 2.9: The effects of noise filtering on a simulated L block with 10% induced noise.

#### 2.2.4 Down-Sampling

Down-sampling is the process of fixing a point cloud's mean density to a voxel of size  $n$ . This is done by iterating the voxel throughout the cloud's entire volume and replacing all points occupying a voxel with a single point in the mean position of the voxel. For an  $n$ -dimensional feature-set, the downsampling equation can be defined by Equation 2.6

$$p_{new}(i) = \frac{\sum_{p(i) \in V}^N p(i)}{N} \quad (2.6)$$

Where  $p(i)$  represents the  $i^{th}$  point in the dataset,  $V$  represents the bounding box of the voxel in  $n$ -dimensional space, and  $N$  represents the total number of points inside the voxel. The result of this equation is the average position of the points inside the voxel.

Figure 2.10 shows the results of a simulated shape of initial point-to-point average

distance of 0.05 inches down-sampled with a voxel size of  $0.25 \text{ in}^3$ .

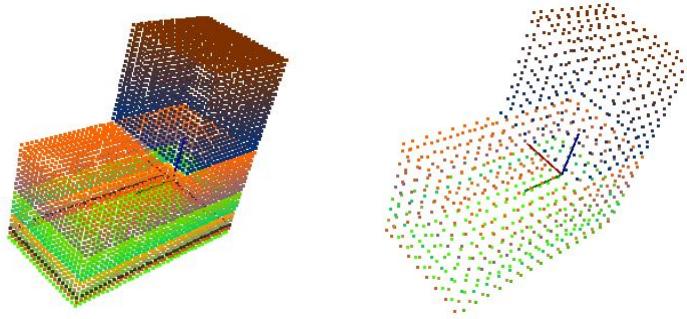


Figure 2.10: A simulated L block point cloud down-sampled with voxel size =  $0.25 \text{ in}^3$ .

Downsampling is crucial for processing point clouds. The obvious benefit is decreased computation time, but it also allows for control of the density of the point cloud. Controlling the cloud density allows for meshing parameters to be non-modular, as the input cloud can be forced to a desired density. Section 2.4 describes the meshing steps in more detail.

### 2.2.5 Handling Occlusion

Occlusion is a common issue in the perception world, defined by the lack of information in an image / 3D scan due to other objects blocking a direct view. A simple example: In 3D scene reconstruction from images, it is impossible to accurately reconstruct the contents inside an opaque box because the images do not provide any information on this space. This phenomenon is called occlusion. In the field, it is nearly impossible to fully avoid occluded datasets when scanning an object via Li-

DAR equipped vehicles. It is crucial to be able to develop methods to alleviate this problem.

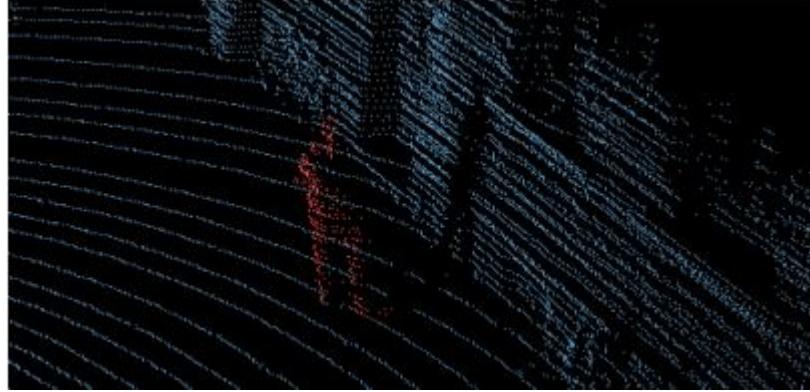


Figure 2.11: Section of a building occluded by an object in the foreground [16].

## Range Segmentation

In urban scanning situations, it is nearly impossible to avoid scan noise caused by obstacles occluding the target surface. Biasuttia et. al. propose a way to cast these interruptions to the surface they desire to map by converting the xyz point cloud to a range image — a three parameter map of distance  $r$  from the device plotted against  $\theta$  and  $\phi$ , the rotation about the  $z$  and  $x$  axes, respectively.

Once the points are cast to a range image, a range histogram is created. The histogram is segmented into  $S$  classes, and the centroid of each class is calculated using the following equation.

$$C_s^i = \frac{\sum_{b \in C_s^i} b \times h_s(b)}{\sum_{b \in C_s^i} h_s(b)} \quad (2.7)$$

Any centroids within some user-defined distance,  $\tau$ , are merged as a single cloud.

$$d(C_s^i, C_r^j) = |C_s^i - C_r^j| \quad (2.8)$$

An algorithm built under the pretext of Gaussian diffusion is then used to project points with a significantly different normals to conform with their range image neighbours. This approach requires structured data that is also time-stamped. In the equation below,  $u$  represents the  $(\phi, \theta)$  coordinates of a point in the merged dataset, and  $\Omega$  represents the full range image of the merged dataset, and  $\eta$  represents the orthogonal projection of each pixel in the range image. The aim is to solve the following disocclusion problem.

$$\begin{cases} \frac{\partial u}{\partial t} - \Delta u = 0 & \in \Omega \times (0, T) \\ u(0, x) = u_0(0) & \in \Omega \end{cases} \quad (2.9)$$

In the equation above,  $\Delta$  represents the Laplacian operator, defined below.  $v_i$  represents the  $i^{th}$  unit vector direction in  $n$  dimensions.

$$\Delta u = \sum_{i=1}^n \frac{\partial^2 u}{\partial v_i^2} \quad (2.10)$$

When scanning large objects, this method proves to be quite effective at removing sources of noise that are significantly smaller than the object being scanned. For buildings, bridges, and large-scale structures, this is a viable first step in removing excess noise / unnecessary information from the cloud [16]. An Example of the effects of range image diffusion is shown below.

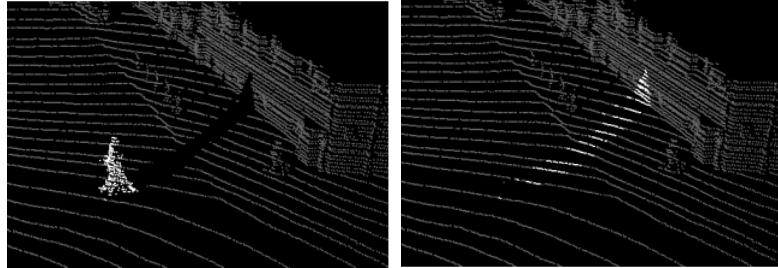


Figure 2.12: The left image shows the unchanged collected point cloud. The right demonstrates how the algorithm diffuses the occluding object into the background [16].

## 2.3 Machine Learning for Object Recognition and Segmentation

### 2.3.1 Unsupervised Methods

#### K-means Clustering

K-means clustering is an iterative method that groups  $n$ -dimensional datasets into  $k$  clusters based on a minimization of the Euclidean distance cost function  $|x - c|^2$ . Initially,  $k$  centroids are placed randomly inside the dataset, and all data points are placed in bins  $S$  depending on which centroid minimizes their cost function. At each iteration, the cluster centroids  $c_i$  are re-calculated. Criteria for convergence is a maximum Euclidean distance change  $\sigma$  between centroid position  $c_n$  and  $c_{n+1}$ .

$$\operatorname{argmin}_S \sum i = 1^k \sum x \in S_i |x - c_i|^2 \quad (2.11)$$

K-means clustering is arguably the most well known clustering method, and is useful for a massive variety of naïve classification problems. In any situation where the bin-size is known and the data is clearly separated in some feature space  $\mathbb{R}^n$ , k-

means proves to be an effective clustering method. It's ease of implementation comes at the cost of computate. As with most unsupervised clustering methods, the major downside to k-means is its non-reusability. Where supervised methods require their computational time upfront during the training phase, k-means requires significant computational time with every dataset. It does not develop a set of multiplier weights, so it must start the clustering algorithm from scratch with each new input set.

### Fuzzy C-means Clustering

Fuzzy C-means (FCM) is very similar to K-means clustering. Once again, points are iteratively grouped to k centroids based on their Euclidean distance to the centroid. The significant difference is that points do not belong exclusively to a single group. Instead, points are weighted by their degree of belonging in each cluster.

$$c_k = \frac{\sum x w_k(x)^m x}{\sum x w_k(x)^m} \quad (2.12)$$

Each point is provided a weight vector  $w$  [0, 1] for its likelihood of belonging in each cluster, where the weight function is as follows:

$$w_{ij} = \frac{1}{\sum_{k=1}^c \left( \frac{|x_i - c_j|}{|x_i - c_k|} \right)^{\frac{2}{m-1}}} \quad (2.13)$$

Fuzzy clustering is differentiated from k-means in that points are not isolated to a singular cluster until the final thresholding step after FCM has reached its exit criteria. This allows for all points in the dataset to continually effect the location of every cluster centroid. FCM finds its worth in  $n$ -dimensional datasets that are not

clearly segregated by their cost functions, but do have a series of denser clouds. FCM is even more computationally intensive than its counterpart, k-means, because it must iteratively calculate the location of every point in relationship to every centroid, and then also calculate the weights of each point in reference to each centroid.

### Agglomerative and Divisive Hierarchical Clustering

In Agglomerative clustering, each point is initially considered its own cluster. Iteratively, the points are grouped together based on a user-defined cost function – in this case, Euclidean distance. The exit conditions for this method are either convergence upon a set of clusters, or a predefined number of clusters. Divisive clustering approaches the clustering problem in an exactly opposite fashion. The algorithm initializes the dataset as a single cluster and iteratively splits the remaining clusters until reaching the same exit conditions as the Agglomerative method.

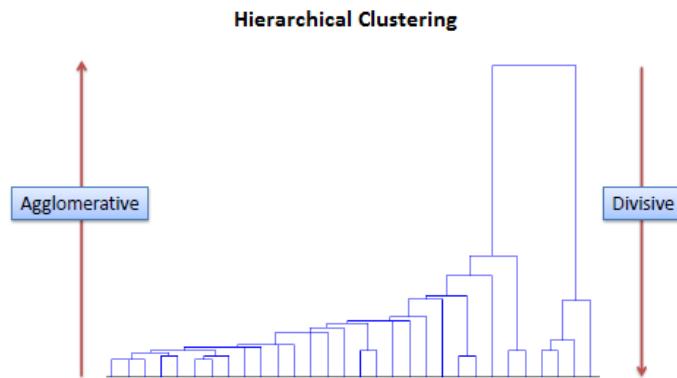


Figure 2.13: Comparison of divisive and agglomerative hierarchical clustering

Hierarchical clustering is not as computationally efficient as k-means or FCM, as

the number of calculations it must make for each point is drastically higher. At each iteration, each point,  $p_i$ , is compared with every point inside of its bin, to determine what bin it will be divided into. This means Hierarchical clustering methods occupy  $\mathcal{O}(n^3)$  time complexity space, and requires  $\mathcal{O}(n^2)$  memory. This makes Hierarchical clustering very unwieldy with medium and large datasets. However, it proves to be one of the more robust supervised segmentation algorithms, as the cost function is easily modifiable by the user.

### **Euclidean Distance Clustering**

Perhaps the simplest of the algorithms listed above, Euclidean distance clustering operates on the pretense that objects are separated significantly enough spatially from another that clustering points based on their proximity to other points in the cloud is sufficient to properly segment the dataset. This algorithm involves no iterative process, and requires three inputs: Maximum point-to-point distance,  $r$ , minimum number of points per cluster,  $k_{min}$ , and maximum number of points per cluster,  $k_{max}$ .

Euclidean clustering occupies the lowest time complexity and memory space of its unsupervised brethren, and does not require an expected bin-size.

#### **2.3.2 Supervised Method — Convolutional Neural Network**

**Definitions** To detect distinctive features in a dataset – from a machine’s perspective – the dataset needs to be modified to enunciate those features. Key features in machine vision include edges, corners, and areas with distinctive geometry. Convolutions are the key to bringing these features to the forefront of the image. A

convolution kernel is a weighted square matrix of dimensions  $m$ , and depth equal to the rank of the feature space of the dataset. The kernel acts as a filter for the image as it strides from supervoxel to supervoxel. At each step, the dot product of the kernel with data values inside the current super-voxel provide a convolved image of the dataset while retaining characteristic features. The equation below illustrates the math behind a convolution kernel.  $C$  represents the convolved image, and  $x_i$  and  $w(i)$  are the  $i^{th}$  point and weight of  $N$  points located inside the voxel.

$$C_{new} = \sum_i^N w(i)x_i \quad (2.14)$$

Another type of convolution is called pooling, or subsampling. This convolution steps through the dataset with a stride value greater than one, resulting in a smaller image of the original set. In pooling, the kernel will pull either the largest intensity from each super-voxel, or the average intensity of the data points inside the super-voxel. This allows for the machine to decrease the size of the dataset while maintaining distinctive features.

$$C_{new} = \frac{\sum_i^N x_i}{N} \quad (2.15)$$

**Convolutional Neural Networks** Convolutional Neural Networks (CNNs) are modeled after the visual cortex in the brain. They consist of layers of convolutional networks. Each network contains “neurons” with simple feature reception fields. Through layers upon layers of these networks, objects can be classified. The biases and weights on these networks can be adjusted based on learning algorithms REF 1 in

proposal. CNNs have become the standard in feature classification for image processing, but the accuracy that they provide comes at the price of processing time. Every CNN can be broken down into the following steps: Convolution, max/mean pooling, activation function, fully connected layer, repeat. The diagram below illustrates the overarching structure of a basic Convolutional Neural Network:

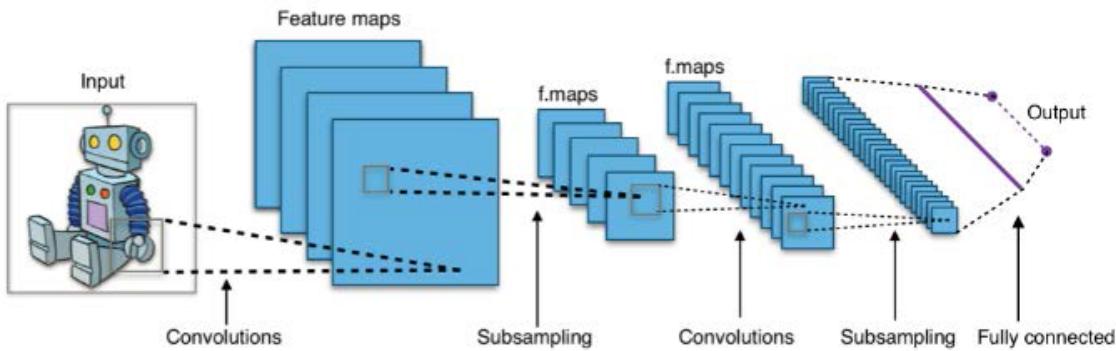


Figure 2.14: Map of a convolutional neural network with two hidden layers [ref]

In the case of point cloud processing, the input is a raw xyz set of some size  $n \times 3$ . The first step in the network is a series of convolutions of the dataset. Each kernel in the layer contains  $m \times m \times 3$  trainable weights, which are iteratively modified using a steepest descent numerical solver during the training phase of the system. The convolved images are stacked in a block, called a feature map, or convolutional layer. From there, the convolved images are pooled (or subsampled) to condense the size of the image stack. At this point, there is a large stack of feature maps drawn from the original input dataset. In a simple linear system, these features are combined into a single weighted summation function, where the input is each individual feature value, and the output is a vector representing the probability of the image belonging to a

certain class.

$$\begin{bmatrix} C_1 \\ \vdots \\ C_n \end{bmatrix} = \begin{bmatrix} w_{1,1} & \dots & w_{1,m+1} \\ \vdots & \ddots & \dots \\ w_{n,1} & \dots & w_{n,m+1} \end{bmatrix} \begin{bmatrix} f_1 \\ \vdots \\ f_n \\ 1 \end{bmatrix} \quad (2.16)$$

The equation above represents the transformation from feature space to the “fully connected layer.”  $C_i$  represents the probability of the input image belonging to class  $i$ , and  $f_i$  represents the value of the  $i$ th feature in the feature map. Linear classification methods limit the versatility of the CNN, as many object distinctions do not follow a linear pattern in  $n$ -dimensional space. To account for this, most CNNs – including the ones utilized in this paper – incorporate a de-linearizing element dubbed the “activation function.” The activation function applies a nonlinear operation to the values in the convolution layer, which allows for the CNN to become a very powerful nonlinear fit function. Typical activation functions include the hyperbolic tangent function, the sigmoid function, and – most popularly – the rectifier function. Each of these functions are show in the figure below:

From input to output, all CNNs have the same skeleton structure, with a varying number of layers between the raw input and the fully connected layer:

For the system to be accurate, the weights for each convolution and connection must be trained. This training is done through a process called backpropagation. An image-set of known classifications is fed to the untrained system, and the error between the system’s classification and the true classification of each image is used to

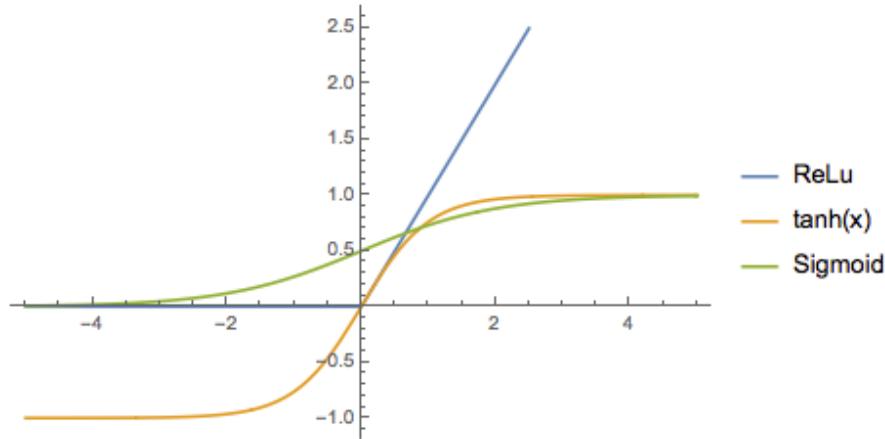


Figure 2.15: Visualization of commonly used non-linear activation functions

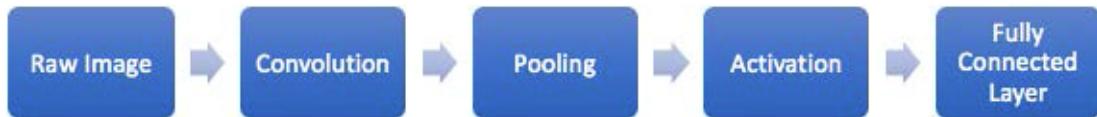


Figure 2.16: Block diagram of a CNN's skeleton structure

update the weights iteratively until the machine’s class prediction closely resembles ground truth. Most algorithms use numerical solving methods to sharply diminish the number of iterations required for convergence. The “gradient descent” method is commonly used in the machine learning world due to its rapid convergence properties and low computational complexity. The method is shown below:

$$x_{k+1} = x_k - \gamma \nabla F(x_k) \quad (2.17)$$

$F(x)$  is described as the residual function. It is the difference between the result of

the cost function  $C(x)$ , and the current  $x$  variable values.  $\gamma$  represents a scalar input parameter determining the scale of the effect of  $F(x_k)$  on  $x_{k+1}$ .

$$F(x) = \begin{bmatrix} C_1 \\ \vdots \\ C_n \end{bmatrix} - \begin{bmatrix} w_{1,1} & \dots & w_{1,m+1} \\ \vdots & \ddots & \dots \\ w_{n,1} & \dots & w_{n,m+1} \end{bmatrix} \begin{bmatrix} f_1 \\ \vdots \\ f_n \\ 1 \end{bmatrix} \quad (2.18)$$

The speed, accuracy, and versatility of a CNN are functions of the number of hidden layers, the size of the convolutional layers, the type of activation functions, and the size and versatility of the training dataset [17]. Because CNNs – and all supervised learning methods – require a large amount of training data, they do not fit within the scope of the proof of concept inside of this paper. However, supervised learning has proven to be one of the most effective forms of classification, and will be addressed in the future work section of this thesis.

## 2.4 Converting a Discrete Point Cloud to a Bounded Area Surface Mesh

### Definition of a Surface Mesh

Surface meshes, in simple terms, are a series of connected areas defined by their vertices in  $n$ -dimensional space. The areas are typically triangles or tetrahedrons, and form together to define a continuous surface area using discrete data points. There are many ways to store surface mesh objects, but they all require two components:

1. A set of vertices,  $v$ , occupying  $n$ -dimensional space  $\mathbb{R}^n$ .

2. A set of connection indices  $I$ , which dictate how the vertices connect to one another.

In a fully bounded surface mesh, the combination of each triangulation object creates a shell, which represents the surface area of the shape.

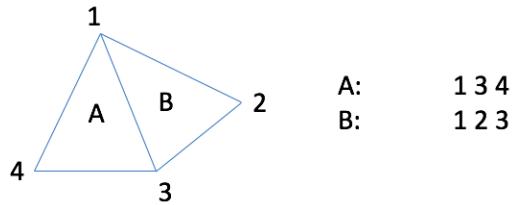


Figure 2.17: Visualization of necessary components in a surface mesh.

Typical file-types for surface meshes contain the location of the vertices in 3-dimensional space, followed by definitions for their connectivity in index form.

### Definition of a Volume Mesh

A volume mesh is defined with the same parameters as a surface mesh, with the key difference being that it defines the entire exterior and interior of the object. One application – a future application for this thesis – is finite element analysis, a numerical method for solving structural analysis problems through discretization achieved by assuming a number of domain partitioning elements in an  $n$ -dimensional geometry and calculating their relative parameters in reference to their connected neighbors.

## Definition of a Boundary Representation (B-REP)

Boundary Representations, often called the "skin" of a solid object, consist of two parts: topology and geometry. The topological portion is made from the faces and vertices created by a surface mesh. In the boundary representation of an object, the edges bounding a triangulation object – or face – is defined as a loop. The geometric portion consists of equations defining the edges and faces. There are a number of different ways in which the geometry of an object's surface area can be modeled, but the most common is the Non-Uniform Rational Basis Spline (NURBS).

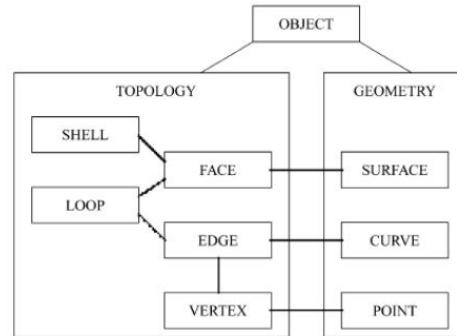


Figure 2.18: An overview of the components involved in a boundary representation of a solid object [18].

Boundary representations can be definitions for both solid volumes and hollow shells. If the surface of the B-REP is not completely bounded, CAD analysis software is forced to interpret the object as a shell. [[ more detail ]]

## Non-Uniform Rational Basis Spline (NURBS)

NURBS curves are a method of interpolating a discrete set of vertex points into a continuous function. Each vertex point is considered a “control point,” and a series of smooth polynomials are concatenated locally over individual sets of control points. An  $n$ -dimensional NURBS curve  $f(u)$  can be defined as the following:

$$f(u) = \frac{\sum_{i=0}^{K-1} w_i B_{i,n}(u) P_i}{\sum_{i=0}^{K-1} w_i B_{i,n}(u)} \quad (2.19)$$

Where  $P_i$  represents the  $i^{th}$  control point,  $n$  represents the polynomial degree of the blending function,  $B_{i,d}(u)$ , and  $w_i$  represents the weight of the  $i^{th}$  control point. The equation is simplified accordingly by inserting the rational basis function relationship described in Equation 2.20 to the piecewise B-Spline equation shown in Equation 2.19.

$$R_i(u) = \frac{w_i B_{i,n}(u)}{\sum_{i=0}^{K-1} w_i B_{i,n}(u)} \quad (2.20)$$

Leaving us with the following relationship:

$$f(u) = \sum_{i=0}^{K-1} R_i(u) P_i \quad (2.21)$$

Using these weighted concatenations, smooth surfaces are generated from a series of sharp geometric edges.

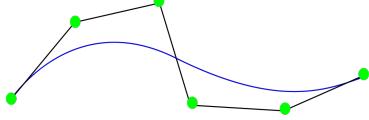


Figure 2.19: A 2-dimensional example of a nurbs spline. Green points represent the control points,  $P$ , black lines represent their connectivity, and the blue line represents one possible nurbs spline.

### Delaunay Triangulation

Surface meshing is the method of inferring a continuous shape topology from a discrete,  $n$ -dimensional point cloud. There are many different approaches to converting from discrete points to surface meshes, but at their core, nearly all of them rely on the Delaunay triangulation method. Delaunay triangulation finds its routes in Voronoi tessellation, a method of constructing non-overlapping geometrical tiles. Voronoi tessellation states the following: For a given set of points in space,  $\{P_k\} — k = 1, \dots, K$ , the regions  $\{V_k\}$  are polygons assigned to each seed point  $P_k$ , such that  $V_k$  represents the space closer to  $P_k$  than any other point in the set.

$$V_k = \{P_i \mid |p - P_i| < |p - P_j|, \forall j \neq i\} \quad (2.22)$$

If every point pair sharing a Voronoi boundary are connected, the result is a triangulation object encasing the pointset. This object is referred to as a Delaunay triangulation [19].

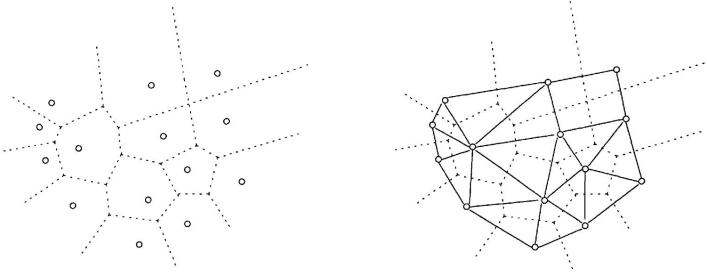


Figure 2.20: Visual representation of a 2-dimensional delaunay triangulation process. The figure on the left shows the voronoi diagram of the vertex set, and the image on the right shows the triangulation result [19].

### Advancing Front / Marching Triangles

The Advancing Front method is common in computer graphics software – especially in procedurally generated games – because of it's speed and computational simplicity.

In the Advancing Front method, a mesh is constructed by progressively adding tetrahedra starting at the boundaries of the previous surface elements. At each iteration, the mesh boundary is propagated further across the set of surface points. New tetrahedra are determined based on the minimal delaunay triangulation between the boundary vertices and unreferenced points in space. Each new triangulation object is created by adding a single unreferenced point at a time [20].

### Scale Space Reconstruction Method

At a grand scale, the Scale Space Reconstruction Method aims to optimize surface meshing in the face of discrete point cloud data. No matter how accurate or dense a point cloud may be, there is no way to verify the topology defined by the cloud is

accurate to the true object topology. To simplify this ill-posed problem, and reduce mesh quality damage due to noisy points, the Scale Space algorithm casts the raw point cloud to a space of scale  $N$  by iteratively calculating the mean curvature of a neighborhood of points and casting each point  $p_k$  to its nearest point on the curve. This results in a far more uniform point cloud, which can mesh via Delaunay triangulation at a high quality level. Once the meshing has occurred, the pointset is then recast to its original scale to maintain complex features.

**Algorithm 1:** Mean Curvature Calculation

**Input:** A point set  $P$ , a query point  $p$ , and a radius  $r$ .

**Output:** A point  $p'$ , the result of one discrete step of the mean curvature calculation applied to  $p$ .

```

for ( p in P )
    get neighbors from p
    if num(neighbors) < 5
        remove p
    set  $p_{bar} = \frac{\sum_{q \in neighbors} w(q)q}{\sum_{q \in neighbors} w(q)}$ 
    set  $C = \sum_{q \in neighbors} w(q)(q - p_{bar})(q - p_{bar})^2$ 
    set  $v_0 = arg_{min}eigenvector(C)$ 
    set  $p' = p - \langle p - p_{bar}, v_0 \rangle v_0$ 
     $p' \cdot n = \frac{p - p'}{|p - p'|} \cdot sign(\langle p - p', p \cdot n \rangle)^*$ 

```

**Algorithm 2:** Scale Space Iterator

**Input:** A point set  $P$ , a number of iterations  $N$ , and a radius  $r$

**Output:** A modified point set  $P_N$

```

for p in P
    set p.origin = p
    set idx = 0
    for ( i = 0, \ldots , N-1 )
        new_idx = mod(idx, 2) + 1
        for p ∈ Pidx
            p' = MCC(p, Pidx, r)
            store p' in Pnew_idx
            p'.origin = p.origin
            if ( idx > 0 )
                remove Pidx
        idx = new_idx
    
```

Using the Ball Pivoting Algorithm [21], a reconstruction is implemented that does not change the location of the smoothed vertex positions or add new vertex positions. In a nutshell, the Ball Pivoting Algorithm iterates a sphere of radius  $r$  throughout the pointcloud. If three points are located on the sphere, a triangulation with those vertices is created. If no points are found, the radius is expanded.

At this point, the vertices of the smoothed mesh are recast to zero-scale ( their initial points in the point cloud). The result is a mesh built based on a smoothed input, with minimal noise reduction in the process [22].

## Hole Filling, Refinement, and Fairing

**Hole Filling** Surface meshes, defined in Section 2.4, contain vertices and faces. Inherently, they do not contain any methods to maintain a bounded volume. For this reason, it is necessary to detect discontinuities in the mesh, and develop a method to resolve them. Two types of mesh edges are defined in this section: A full edge, and a half edge. A full edge occurs when an edge is referenced by more than one polyhedral object. What this means in physical terms is that the edge is not on the boundary of the shape. Half edges, on the other hand, are only referenced by a single polyhedral object. Half edges are indicators of holes in the mesh, as they are unreferenced anywhere else in the surface area.

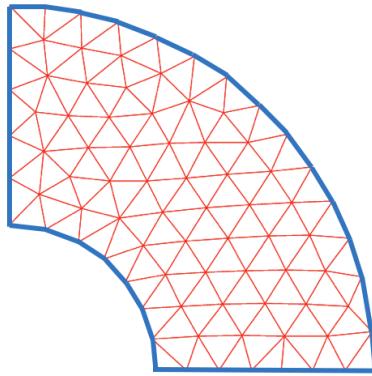


Figure 2.21: A visualization of the difference between an edge and a half edge. Half edges are outlined in blue.

To remedy this, a delaunay triangulation search is run between half edges, and create the least costly triangulations. While this seals the mesh, the result is a non-uniform surface mesh, where the new sections have a far larger surface area than the surrounding mesh. For this reason, refinement is necessary to return the mesh to an

isotropic state.

**Refinement** Mesh refinement can be broadly described as the artificial increase in mesh resolution by the introduction of new vertices and triangulation objects inside of pre-described edges. This allows for precise creation of uniform resolution throughout the features of the mesh.

Specifically, refinement of newly created surface areas inside of the hole filling algorithm occurs by first estimating the density of the triangulations around the new area and subsequently matching that density by inserting vertices into the new area. [23].

**Fairing** The fairing algorithm smooths the newly refined surface into a continuous shape based on a minimization of a linear Laplacian system with boundary constraints at the previous half edge points [24]. Laplacian smoothing is an iterative scheme which shifts vertex locations to satisfy the discrete partial differential equation shown below:

$$p_i^{(t+1)} = p_i^{(t)} + \lambda L(p_i^{(t)}) \quad (2.23)$$

Where  $p_i$  is the  $i^{th}$  point in the cloud,  $t$  is the current iteration, and  $L$  is the Laplacian operator, defined as the following:

$$L(p_i) = \frac{1}{2}((p_{i+1} - p_i) + (p_{i-1} - p_i)) \quad (2.24)$$

Converting to matrix form, the smoothing function operating on the full point set can be defined quite succinctly.

$$P^{(t+1)} = P^{(t)} - \lambda L P^{(t)} \quad (2.25)$$

$$P = \begin{bmatrix} x_1 & y_1 & z_1 \\ \vdots & \vdots & \vdots \\ x_n & y_n & z_n \end{bmatrix} \quad L = \frac{1}{2} \begin{bmatrix} 2 & -1 & 0 & 0 & 0 & -1 \\ -1 & 2 & -1 & 0 & 0 & 0 \\ \dots & & & & & \\ 0 & 0 & 0 & -1 & 2 & -1 \\ -1 & 0 & 0 & 0 & -1 & 2 \end{bmatrix} \quad (2.26)$$

The effects of Laplacian smoothing on a mesh can be drastic, as it relies solely on the shifting of vertex locations based on the connection to their neighbors. Figure 2.22 shows the effects of global smoothing with  $\lambda = 0.1$ .

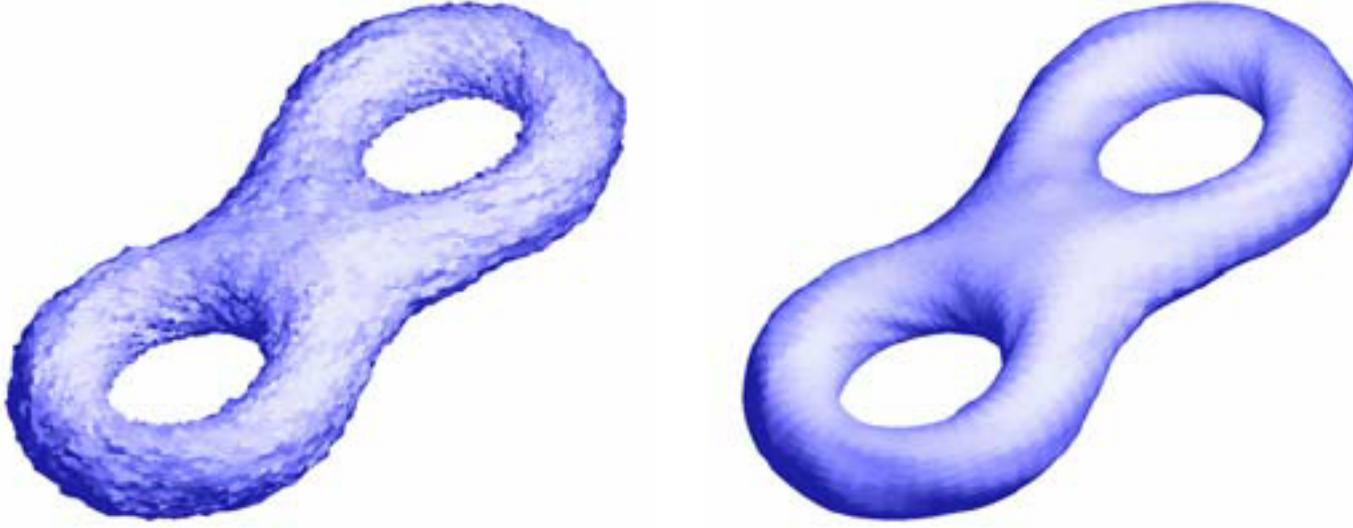


Figure 2.22: Effects of global Laplacian smoothing on a noisy "8"-shaped mesh [25].

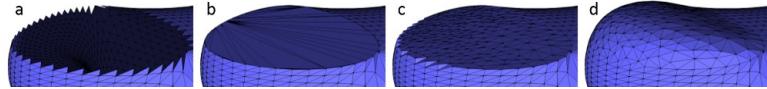


Figure 2.23: Results of the main steps of the algorithm. From left to right: (a) the hole, (b) the hole after its triangulation, (c) after triangulation and refinement, (d) after triangulation, refinement and fairing [26].

#### 2.4.1 Mesh Optimization

Now that the objects in the cloud are properly segmented, they must be meshed in a way that is both accurate to the real-world definition of the object and suitable to be converted from a surface mesh to a solid volume.

#### Criteria for Mesh Quality and Failure Criteria for Solid Volume Conversion

**Failure Criteria** The first criterion for surface mesh compatibility with solid volume conversion is “water-tightness.” Meaning, there are no gaps, holes, or unbounded edges (called half-edges in this thesis) in the triangulation. These gaps can be quantified as any edge in a polyhedron that is referenced by no other polyhedron in the triangulation. The other criteria are more abstract in nature and are more difficult to detect and handle independently. These criteria are defined as mesh “Quality.” Quality is a quantification of the level of simplicity of a triangulation object by evaluating ratios of different elements in the triangulation.

In order to resolve low quality areas autonomously, what it means to be “low quality” must be explicitly defined mathematically. Table 2.1 lists some of the ratios to define a mesh’s quality level. Low quality values in mesh return problematic polyhedrons for solid volume conversion, typically looking like those shown in figure

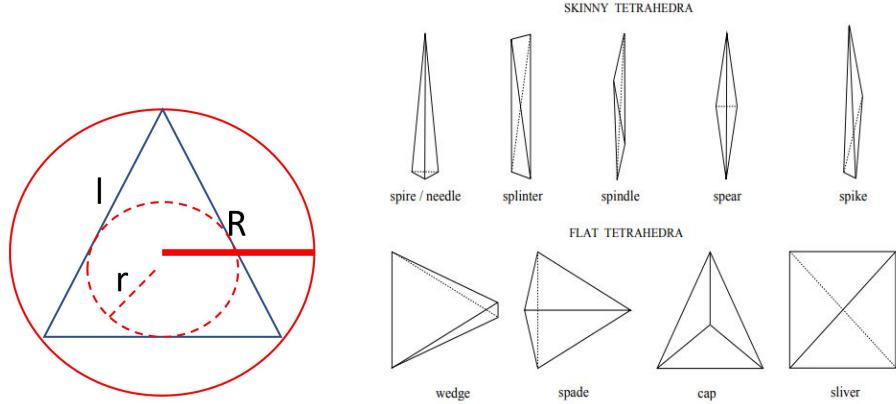


Figure 2.24: Definitions for triangulation quality measures, and typical shapes with poor quality.  $R$  = circumsphere radius,  $r$  = inscribed sphere radius,  $l$  = edge length.

2.24.

Table 2.1: Quantification of mesh quality

Inner/outer radius edge ratios	$Q_1 = \frac{l_{min}}{R}$ , $Q_2 = \frac{r}{l_{min}}$
Aspect ratio	$Q_3 = \frac{r}{R}$
Edge ratio	$Q_4 = \frac{l_{min}}{l_{max}}$

### Voronoi Relaxation / Lloyd's Algorithm

Voronoi relaxation operates on the same principle as k-means clustering. At each iteration, the centroid of each Voronoi region is calculated, and the concurrent vertex is moved to the centroidal location. At convergence, the resulting mesh is uniform in tetrahedron/triangulation size. Voronoi relaxation can modify a shape's topology significant due to its heavy smoothing capabilities [27].

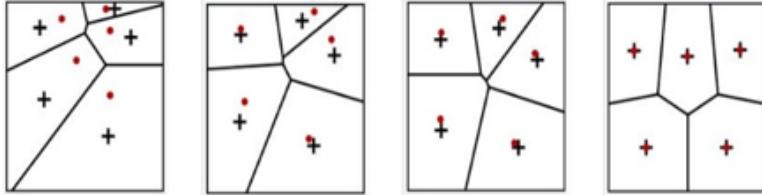


Figure 2.25: From right to left: Voronoi relaxation over 1 iteration, 5 iterations, 10 iterations, and 15 iterations

### Optimal Delaunay Triangulation

Delaunay triangulation, as described in Section 2.4, optimizes the connectivity for a finite point set through voronoi propagation. Optimal triangulations then, are triangulation objects that minimize some cost function  $Q$ , defined by the linear interpolation error of the mesh. To calculate the optimal triangulation of the pointset defined by a discrete set of vertices, the vertices of the current triangulation are unfixed from their respective points in space. As a caveat, because this optimization method involves shift the locations of the vertices, the topology of the mesh is altered in the process.

To define an optimal triangulation, given a continuous convex function  $f$  on  $\Omega$  and  $1 \leq p \leq \infty$ , and triangulation  $\tau^* \in P_N$  the criteria in the equation below must be satisfied.

$$Q(\tau^*, f, p) = \min_{\tau \in P_N} (\tau, f, p), 1 \leq p \leq \infty \quad (2.27)$$

Where  $f$  represents some interpolation function (typically euclidean distance), and  $p$  represents the interpolation error norm space ( $1 \leq p \leq \infty$ ) [? ].

Optimized Delaunay Triangulation is known for maximizing the minimum angle between edges, and is therefore an effective means for increasing the quality of a mesh by the standards set in the previous section.

### **Mesh Perturbation**

While Voronoi relaxation and ODT are large scale smoothing and refinement techniques, they have no constraints on slivers present in the mesh. Perturbation and exudation are necessary to oust any slivers remaining in the triangulation. Slivers are defined as any triangulation with an angle less than  $\alpha$ , a user-defined parameter. The algorithm iteratively increases the angles created in a triangulation by applying a pseudo-random perturbation vector,  $p_v$ , to vertices coincident with triangulations defined as slivers. If the perturbation results in a success, resulting triangulation is kept. Otherwise, a new perturbation vector is calculated to create a higher quality triangulation [28].

### **Mesh Exudation**

Exudation again is a method to remove any slivers remaining in the surface. Each point in a tetrahedron classified as a sliver is assigned a weight based on its distance relationship to its neighbors. The point weighted most heavily, then, is the tip of the sliver, and is modified to place the tetrahedron within the acceptable bounds of mesh quality [29]. [[specifics]]

### 3. Objective, Hypothesis, and Technical Approach

#### 3.1 Research Objective

The objective of this thesis is to develop and validate a computational approach capable of converting raw point cloud data to solid models. The approach accomplishes this by segmenting point cloud data of some expected density  $\rho$  and within some accuracy tolerance  $\sigma$  into clusters of meaningful shapes, and converting the resulting clusters into surface meshes valid for conversion to simply connected solid volume representations.

#### 3.2 Hypothesis Statement

The hypothetical claims made in this thesis are the following:

1. Point cloud scans oriented towards a specific object can be segmented to isolate said object with or without supervised clustering methods.
2. Provided a point cloud meets some required criteria, there is a generalized algorithm to convert raw clouds to solid volume adaptable surface meshes.

The second hypothesis attempts to nail down an explicit definition of what it means for a mesh to be solid volume compatible. Determining what conditions a point cloud must meet to be properly meshed, and what modifications can be made and to what extent are all problems that have yet to be fleshed out completely in

the domain of real world to simulation conversion technology. This thesis aims to quantify criteria for raw point clouds to be meshed properly, criteria for smoothing limits before the resulting volume can no longer be realistically considered to have similar properties to its real world counterpart, and criteria for a surface mesh to be successfully converted to a volume.

Knowing these limits allows for collection of point cloud data for the sake of information extraction to be much more educated. Defining criteria for the amount of a surface that must be visible in the cloud, accuracy limits, and smoothing limits are critical for creating an automated approach to CAD representation of real objects with reliable results. The following chapters quantify the effect of parameter modification at each step of the algorithm.

### **3.3 Technical Approach**

To fully evaluate the effect of each step in the algorithm and test its robustness, it is applied to a series of point clouds, both simulated and real, with varying levels of noise and occlusion. The goal is to accurately mimic the cloud resolution of the test device, and to prove the robustness of the algorithm at resolutions expected from other measurement devices. Simulated data has been crafted to model resolution, accuracy, and occlusion levels with the approaches that follow:

#### **3.3.1 General: Zero Noise**

The pipeline is first explored with a fabricated point cloud of varying resolutions. This point-cloud does not omit any features of the simulated room to mimic occlusion,

and has zero simulated noise. In short, every point in the cloud is in its true position in 3-dimensional space. These clouds shown below are optimal cases and are entirely unacquirable with any instruments in use today, or in the foreseeable future. They do, however, serve as a proof of concept for the meshing pipeline as a form of optimal results.

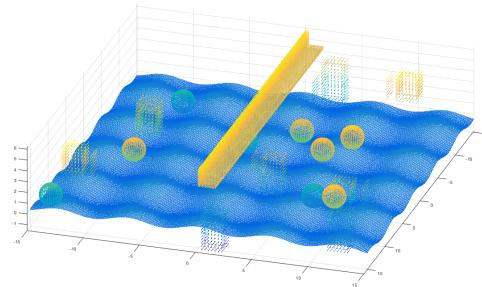


Figure 3.1: Virtual lab point cloud with zero induced noise and occlusion. there are 114,808 points in the cloud.

### 3.3.2 General: Induced Noise

It is crucial to view the effects of noise on the control dataset, as no method or instrument yet created can provide a fully noiseless discrete dataset. To model the extreme noise cases, gaussian noise of some value,  $\sigma$ , is applied to the mesh. This  $\sigma$  value is determined based on the likely characteristics of the instruments, or based on test cases with some desired accuracy parameter. Figure 3.2 shows the results of induced noise on a simulated LiDAR scan with  $\sigma = 2cm$ .

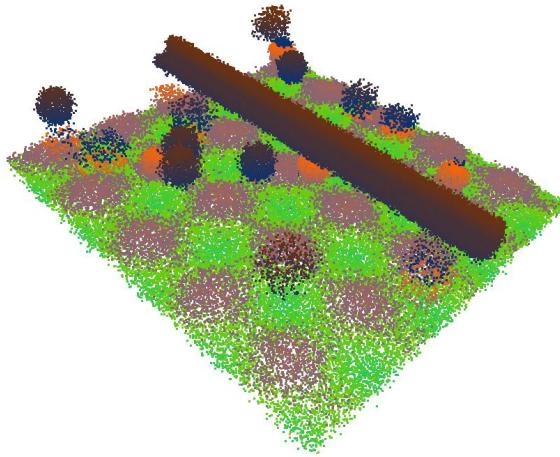


Figure 3.2: Simulated high density lab point cloud with 2 centimeter noise induction.

### 3.3.3 LiDAR: Velodyne HDL-32E

The Velodyne HDL-32E is a high end LiDAR device capable of collecting 700,000 points per second at a frame rate of 20 Hz (rotations per second). It is equipped with 32 lasers aligned from  $+10.67^\circ$  to  $-30.67^\circ$

As described in Section 2.1.2 of the introduction, LiDAR devices provide explicit distance feedback in spherical coordinates, where each point is defined by  $\phi$ , its angle about the  $z$ -axis,  $\theta$ , its angle about the  $y$ -axis, and  $r$ , the absolute distance from the sensor. The return can be classified as ordered data, and easily mimicked in simulation by creating a series of solid objects, simulating the output of the LiDAR, and simulating the LiDAR motion. This allows for fast simulation of point cloud collection with varying levels of noise, occlusion, and sample rate.

The LiDAR is fixed to a mobile table of fixed height during the collection process. This means the amount of acquireable data on the beam's shape is severely limited. In

Section 4.3, the effects of this scanning technique can be seen in the lack of thickness data on the horizontal portion of the beam.



Figure 3.3: Visualization of how the Velodyne HDL-32E operates.

Individual scans are iteratively concatenated in the manner described in Figure 3.4. At each iteration, a new frame is loaded, and the Intrinsic Shape Signature map for the cloud is generated. Using a RANSAC fitting algorithm – described in section 2.2.2, the transformation between the newest frame and the previous frame is calculated, and added to the global transformation matrix. The new frame is transformed and added to the global cloud. This technique allows for comparison of only two frames at a time, instead of having to calculate the Intrinsic Shape Signature map of the concatenated point cloud, which increases in size at each iteration.

In the experimental test, the LiDAR device is mounted to a mobile horizontal surface and moved in a sinusoidal fashion down the length of the beam. Figure 3.5 shows the collection path and estimated resolution of the returned cloud. Note that this simulation does not accurately capture the effects of occlusion or concatenation artifacts.

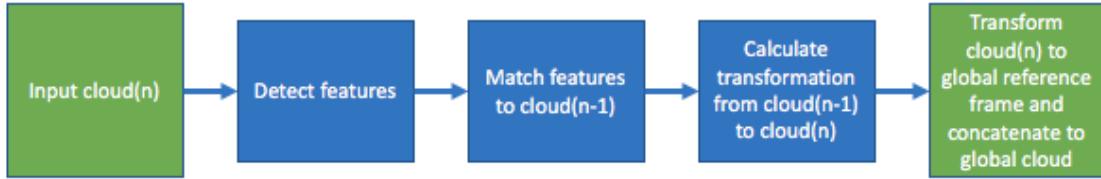


Figure 3.4: Flow chart of point cloud concatenation process.

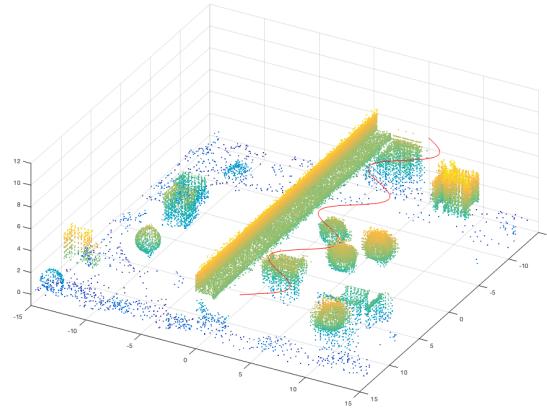


Figure 3.5: Simulated LiDAR scan of the virtual lab point cloud at 100% sampling rate. The red trail shows the collection path of the sensor.

### 3.3.4 Segmentation Method Exploration

To mesh a point cloud object properly, the object must be isolated robustly from the point cloud. This section deals with the discovery of the most effective and robust methods for segmentation of unordered point cloud into meaningful clusters in an entirely automated way.

As a result, the methods evaluated in this thesis are reserved purely to those which are classified as unsupervised. Selected based on potential success, popularity, and ease of implementation, the following segmentation methods are analyzed

in Chapter 4: K-Means Clustering, Fuzzy C-Means Clustering, Euclidean Distance Clustering, and Agglomerative Hierarchical Clustering.

Detailed descriptions of how these algorithms operate, as well as general use cases and a list of pros and cons, can be found in Section 2.3. Each method is applied to the simulated datasets, and their results are visualized. Based on observation, an optimal clustering method is found and the resulting segments are run through the meshing and optimization phase of the algorithm.

### **3.3.5 Initial Mesh Exploration**

Each of these methods are evaluated by their abilities to create comprehensive surface areas and their quality metrics after hole patching and refinement.

### **3.3.6 Optimization Steps**

The goal of the optimization process is to remedy or remove any elements in the mesh that violate the criteria for B-REP and subsequent finite element conversion compatibility. These criteria are defined in Section 2.4.1. Each step in the optimization pipeline specializes in modifying specific characteristics of the mesh which can be defined as having low quality using the metrics in Section 2.4.1.

The first step in the optimization process is voronoi relaxation. voronoi relaxation seeks to resolve triangulations with low [[add quality metrics resolved by voronoi relaxation]]. voronoi relaxation comes with the caveat that it moves the location of the shape vertices, so this step comes at the cost of modifying the topology collected directly by the sensor. Depending on the accuracy of the collection instrument, this

can be either a curse or a blessing.

The next step is the Optimized Delaunay Triangulation method. ODT is another relaxation method capable of modifying the object’s topology in order to increase the overall quality of the mesh. Between voronoi relaxation and ODT, every quality metric should – while not guaranteed to be so – be within the tolerance set by the exit conditions applied, with the exception of slivers. To remedy these, perturbation and exudation are necessary.

Perturbation and exudation are included in the pipeline of algorithm for two reasons. The first being that both voronoi relaxation and ODT are quality improving algorithms for all of the parameters listed in Section 2.4.1, except for slivers. Perturbation and Exudation both specialize in the removal of these tetrahedra. The second reason is to relieve the effects ODT often has on the mesh. ODT can have a “pock-marking” effect upon the mesh, as it optimizes using delaunay radii as a cost function, and this often means vertices are shifted towards the interior of the surface mesh.

### 3.3.7 Final Mesh Verification

The final verification of the meshes output by the algorith occurs in two stages. The first being a measure of each desired quality parameter set in this paper. As there is no explicit definition of what parameters – or what values those parameters must take – define a mesh qualified for B-REP conversion, an analysis of the resulting minimum quality values occuring in the mesh and their relationship to the mesh’s convertibility is made. The second is an exportation of the mesh to AutoDesk Fusion,

which has in-built functionality to convert surface meshes to boundary representations and subsequently finite element volumes. If both of these verifications are successful, the resulting mesh is placed under typical loading conditions to compare its response to the expected response of the object.

[[set finite views for each image]]

## 4. Results

### 4.1 Simulated Data: Zero Noise

The first instance of the algorithm is the control set, and first step in the ground-up development of the pipeline. There is no noise induced in the system, and all objects are significantly separated in 3-dimensional space. Figure 4.1 shows the point cloud. The total number of points in the cloud is 114,808 and beam point-to-point distance is 0.03 in. Beam dimensions are 10ft in length, with a cross-section of the following specifications:

Width: 5in (12.7cm)

Height: 5in (12.7cm)

Thickness: 0.25 in (0.635cm)

Figure 4.1 shows the unmodified pointcloud at high resolution. The dataset in this section has zero noise and zero occlusion, and provides insight into what each step of the mesh optimization process accomplishes. Including the ground plane, there are 17 distinct objects in this point cloud.

While not necessary in the zero noise case, filtering and downsampling prove to be crucial portions of the algorithm. In Figure 4.2 a filter is applied to remove any points with fewer than 15 neighbors, where neighbors are defined as being within one standard deviation plus the average point-to-point distance of the entire cloud. After filtering, the cloud is subsampled to have a maximum density of 0.1 centimeters squared.

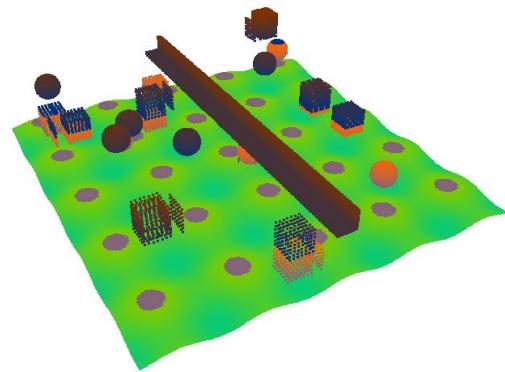


Figure 4.1: Simulated laboratory point cloud data with zero noise induced.

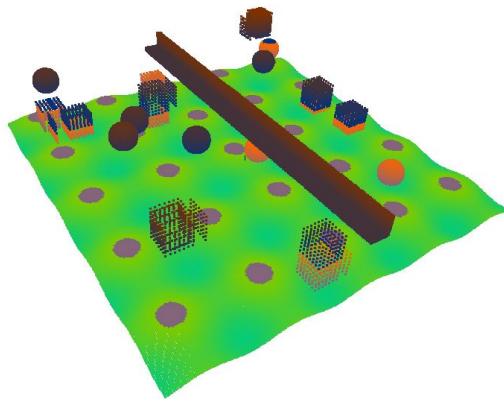


Figure 4.2: Zero noise simulated data after being filtered with voxel size of  $0.1\text{in}^3$ , 15 minimum point neighbors at a distance of 1 standard deviation from the mean point to point distance.

The resulting cloud size is [[insert cloud size]], with an average point-to-point distance of 0.1 inches. Being able to actively control the point cloud density is crucial to have control over proper meshing parameters and fail conditions.

#### 4.1.1 Segmentation Method Comparison

With the exception of euclidean distance clustering, each unsupervised clustering method requires a number of bins. For FCM and k-means, these bins represent the centroids of the clusters, and for Hierarchical, they represent the exit condition for number of divisions. Ideally, the number of bins should match the number of desired segmentations, so the bin size is set to 17. Euclidean clustering requires a minimum radius between points as an input. As the mean point-to-point distance is forced to 0.1 inches, a higher value encapsulates surface gaps which are encountered in the non-ideal cases. Setting the distance to 0.5 inches proves to have satisfactory results.

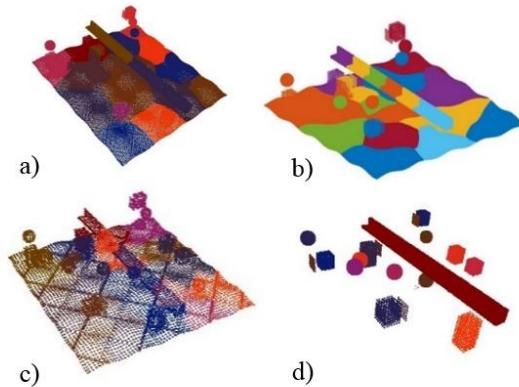


Figure 4.3: Resulting cloud cluster from a) k-means, 17 centroid b) Fuzzy c-means, 17 centroids c) Agglomerative clustering, 17 bins and d) Euclidean clustering with radius of 0.5 in and a maximum cluster size of 50,000 points.

It is clear from the results in Figure 4.3 that Euclidean clustering far outshines its unsupervised brethren in ability to segment the noiseless point cloud in a meaningful way. To clear out undesired segments, criteria on the minimum and maximum clouds size are imposed. In Figure 4.4 a minimum cloud size of 3,000 points, and a

maximum size of 50,000 is imposed on the segmentation results.

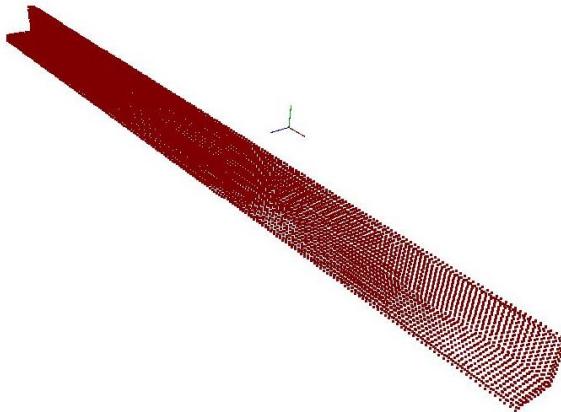


Figure 4.4: Euclidean clustering with radius of 0.5 in, minimum cluster size of 3,000 points and a maximum cluster size of 50,000 points.

Hence it is concluded that the filtering and segmentation parameters derived from this zero noise situation are adequate for isolating the beam from the rest of the point cloud.

#### 4.1.2 Initial Meshing Methods

In a zero noise situation, the advancing front method returns a nearly perfect mesh, with the exception of a poor quality polygon linking the rear segment of the mesh. This is caused by the “marching triangles” attempting to close the final corners of the mesh. This meshing method provides high quality meshes while maintaining accurate features in zero noise situations, but with the introduction of noise the Advancing Front method quickly fails to produce a reliable initial mesh for successful optimization.

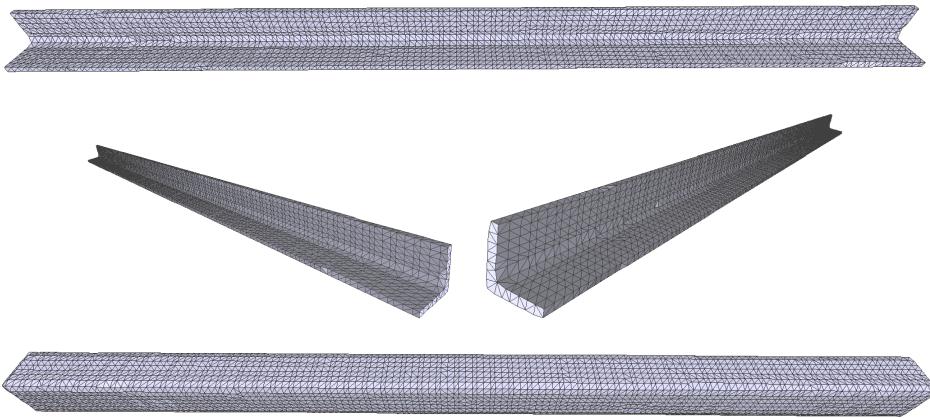


Figure 4.5: Result of initial meshing using the advancing front method.

Figure 4.6 shows the effects of Scale Space reconstruction at scale  $S = 2$  – definition for scale space reconstruction found in Section 2.4. The predominantly visible effect is the change in topology of the beam. because of the localized function estimation that occurs during the scale-space casting phase, the planar surface of the beam is reconstructed with a parabolic shape. The overall thickness is reduced.

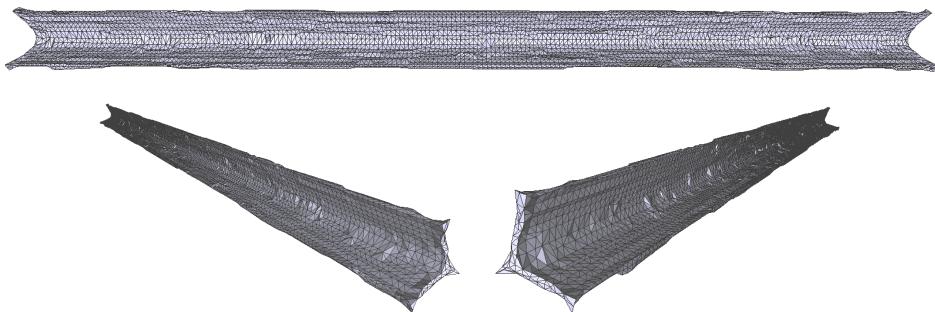


Figure 4.6: Result of initial meshing using the scale space reconstruction method with  $S = 2$ .

As the beam is cast into higher and higher scale-spaces, the thickness of the beam

is continually reduced. At scale-space 4 and higher – shown in Figures 4.7 and 4.8, the beam thickness (originally  $\frac{1}{4}$ in) is reduced to zero. This is due again to the piecewise function estimation that occurs in the specified scale space. Once all of the points are cast down far enough, they are no longer registered as being in two separate neighborhoods, and are instead identified as belonging to the same 2-dimensional curve.

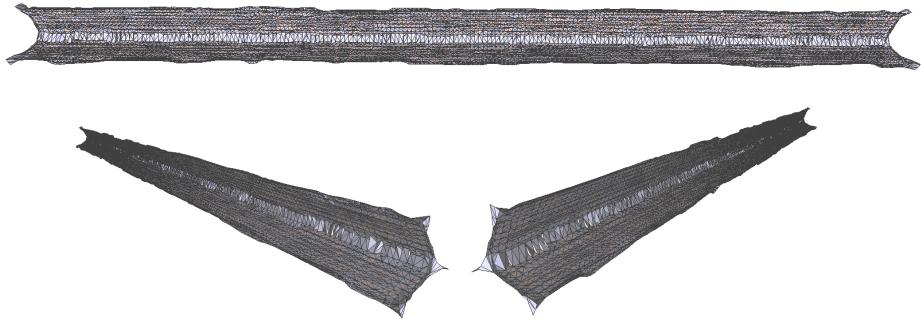


Figure 4.7: Result of initial meshing using the scale space reconstruction method with  $S = 4$ .

Figure 4.8 shows the extremes of what the scale space reconstruction can do to alter the topology of a mesh. Because the points are cast to such a high scale, there is virtually no thickness between points on one side of the beam and points on the other. For this reason, the shape is cast as a 2-dimensional object. It is clear that casting an object like the beam to as high a scale as shown here is not viable for initial meshing.

In the zero noise case, Advancing Front reconstruction proves to be the most accurate to the beam's dimensions. However, as seen in future sections, this method is not robust to noise. Both of these statements are unsurprising, as this reconstruction

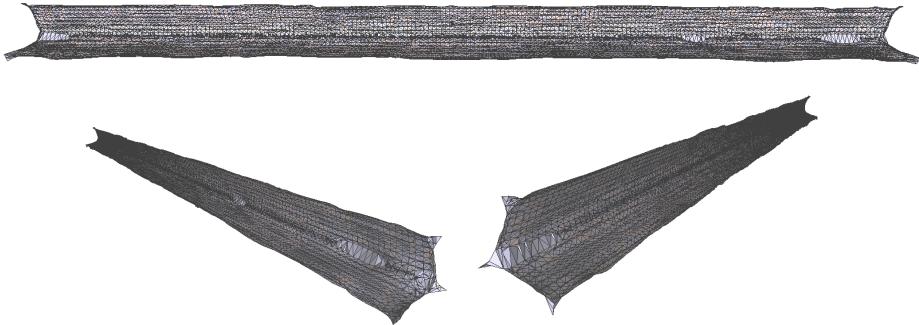


Figure 4.8: Result of initial meshing using the scale space reconstruction method with  $S = 15$ .

method does not modify the topology defined by the pointset provided, and therefore directly represents the shape defined by the input cloud.

Table 4.1: Initial quality analysis of simulated zero noise surface mesh.

Mesh Type	$Q_1$	$Q_2$	$Q_3$	$Q_4$	Bounded Surfaces	B-REP Compatible	FE Compatible
Advacing Front	0.0548	0.2916	0.0270	0.0274	1	Yes	Yes
Scale Space 2	0.0577	0.0665	0.0082	0.0587	1	No	No
Scale Space 4	0.0038	0.0185	0.0005	0.0035	1	No	No
Scale Space 15	0.0022	0.0186	0.0002	0.0015	1	No	No

Quality values shown are defined in Section 2.4.1 and represent the absolute minima in each category.

The advancing front case is the only case to be CAD compatible at first pass. This is due to the input set being completely noiseless. None of the Scale Space reconstructions pass the conversion test, as they all contain regions that are nearly or entirely two dimensional. These methods are revisited after the optimization steps are complete.

#### 4.1.3 Optimization

[

## Effects of Voronoi Relaxation

Voronoi relaxation, defined in Section 2.4.1, aims to regulate the tetrahedral areas in the mesh by iteratively shifting point vertices to their respective voronoi centroids. This method is effective in increasing the mesh quality ratios defined in Table 2.1, with the exception of volume ratio, which can be visualized as slivers in the mesh (Figure 2.24).

The relaxed mesh after 200 iterations of voronoi relaxation is shown in Figure 4.9. Note the uniformity in tetrahedral area, as well as the change topology around the edges of the beam.

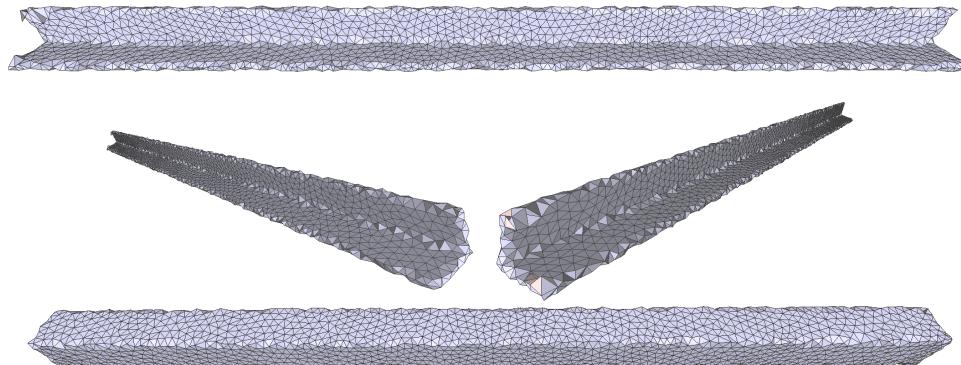


Figure 4.9: Result of Advancing Front mesh after 200 iterations of voronoi relaxation.

Voronoi Relaxation has interesting effects of the scale space meshes. The mesh in Figure 4.10 is a relatively smooth, pock-marked version of the initial scale-space mesh. Notice on the left-most corner of the beam there are a few tall, narrow shapes. These shapes are the slivers remaining in the mesh. The next steps in the optimization process aim to remove these slivers.

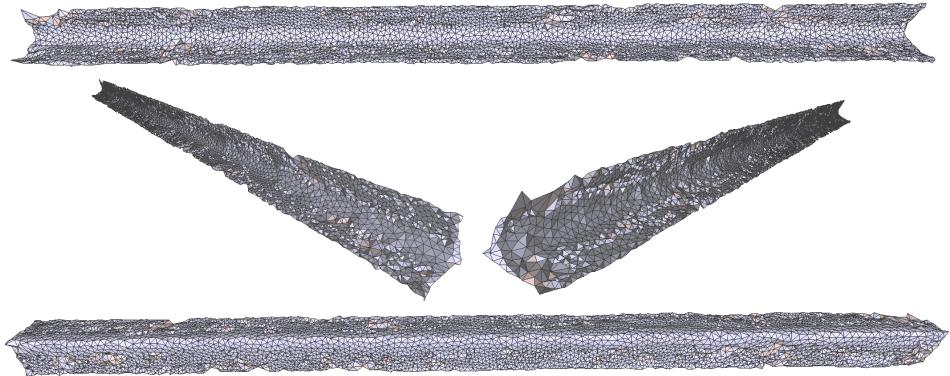


Figure 4.10: Scale space reconstruction  $S = 2$  after 200 iterations of voronoi relaxation.

Similar results are shown in Figure 4.11. This mesh is 2-dimensional due to the scale-space casting, but still shows the effects of voronoi relaxation quite vividly.

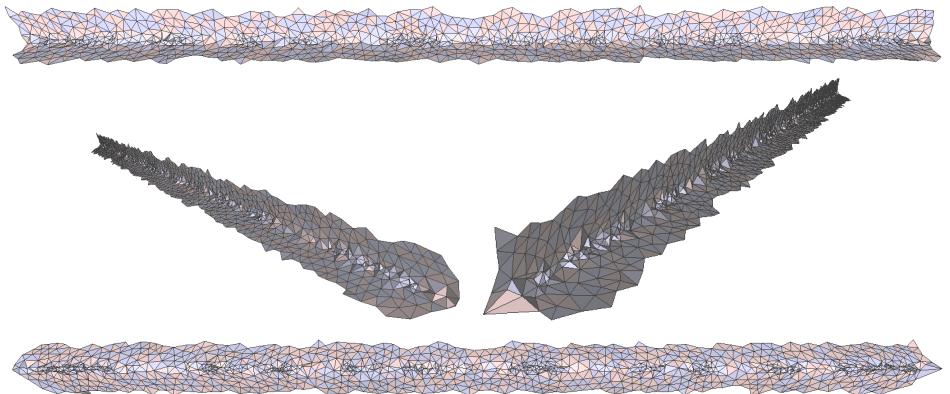


Figure 4.11: Scale space reconstruction  $S = 4$  after 200 iterations of voronoi relaxation.

## Effects of Delaunay Optimization

Delaunay optimization seeks to find the minimal triangulation conditions for the mesh by iteratively shifting the vertices of the mesh to positions that minimize the cost function defined in Section 2.4.1. For this reason, “pock-marks” begin to appear in the mesh, as these positions minimize the Delaunay radii apparent in the mesh. Correcting this phenomenon is a large portion of the reason perturbation and exudation exist in meshing pipeline.

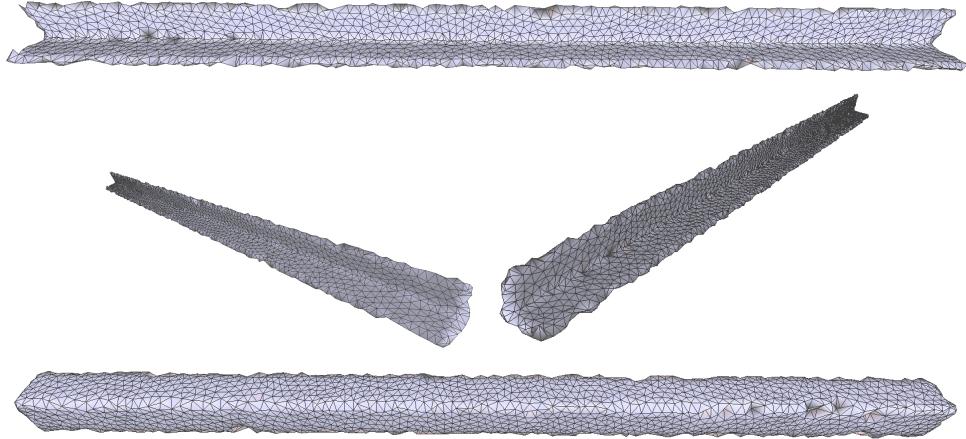


Figure 4.12: Result of Advancing Front mesh after ODT with a 30 second clock cap.

The negative effects of ODT are increasingly apparent at each greater scale of the scale space reconstruction. Any imperfections in the mesh are enunciated by a heavy, vacuum-like extrusion.

It is clear at this point that for low thickness objects in zero noise situations, 4<sup>th</sup> scale space is too high, as it results in a zero volume surface mesh. There are no remedies for a mesh that fails to generate any surface hull at all.

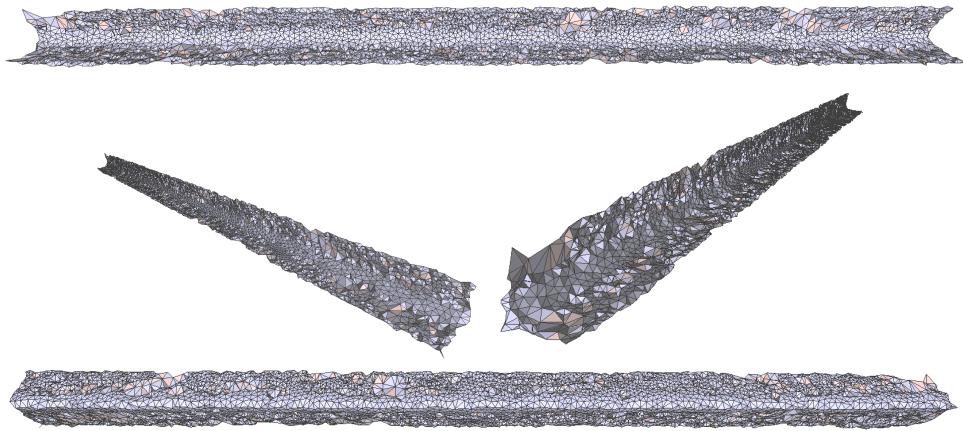


Figure 4.13: Scale space reconstruction  $S = 2$  after ODT with a 30 second clock cap.

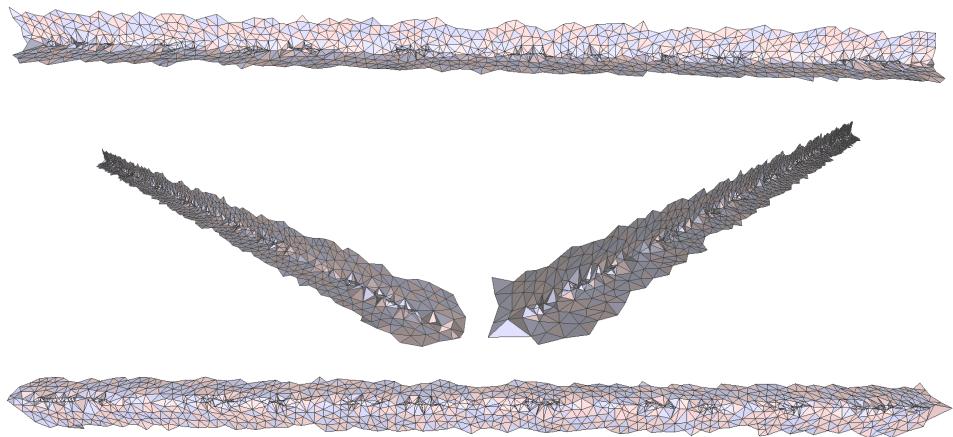


Figure 4.14: Scale space reconstruction  $S = 4$  after ODT with a 30 second clock cap.

### Effects of Mesh Perturbation

Perturbation serves two purposes: To relax any remaining slivers in the mesh (slivers defined in Section 2.4.1) into higher quality tetrahedra, and to relieve inward deviations created by the ODT step.

In Figure 4.15, the result of the Advancing Front mesh after being perturbed for

30 seconds is seen. In this case, there were not many – if any at all – tetrahedra that met the criteria to be slivers. The only sliver correct visible is in the lower lefthand corner of the beam. In earlier steps of the algorithm, there is a slight dog-ear, which is now far less exaggerated.

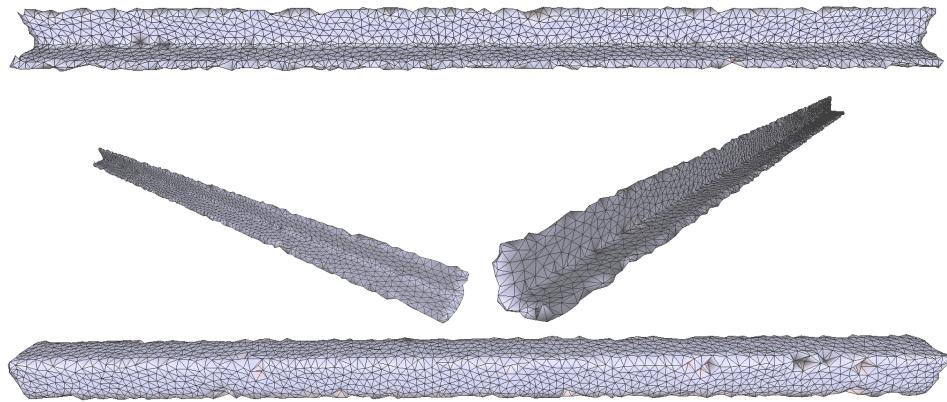


Figure 4.15: Result of Advancing Front mesh after perturbation with a 30 second clock cap.

The next figure is the reconstruction at scale space 2. Again the mesh is severely pock-marked, but far less so than its predecessor. Provided a higher clock limit, these pock-marks are slightly more relieved, but unfortunately perturbation does not converge at the “smoothest” mesh, but at the mesh that satisfies its constraints for quality.

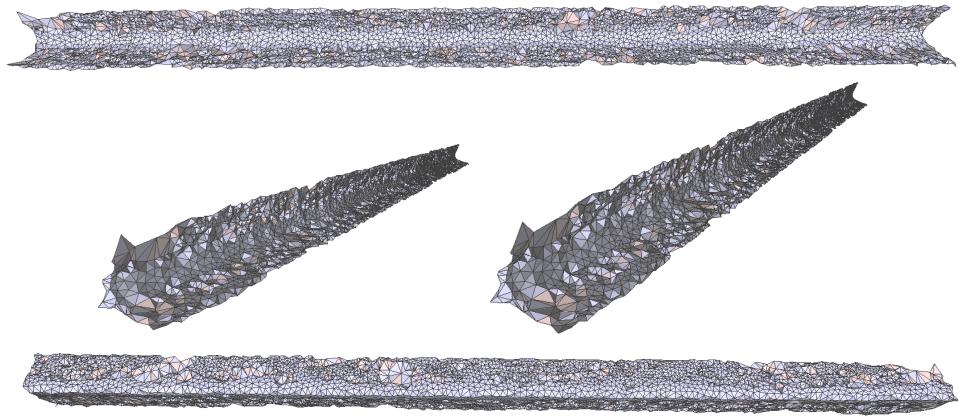


Figure 4.16: Scale space reconstruction  $S = 2$  after perturbation with a 30 second clock cap.

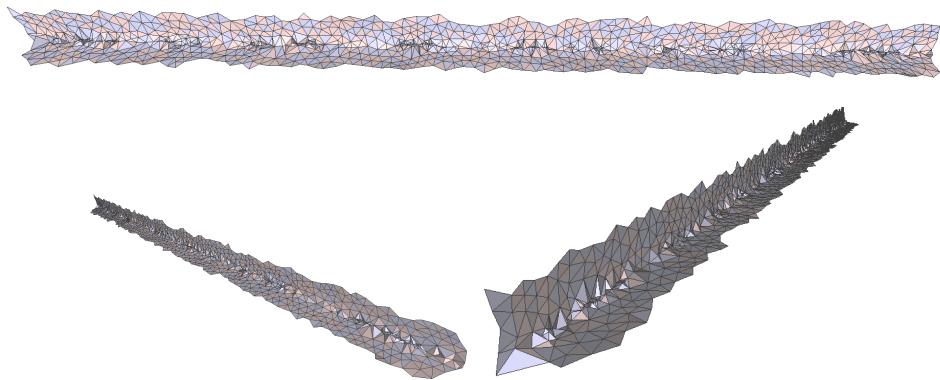


Figure 4.17: Scale space reconstruction  $S = 4$  after perturbation with a 30 second clock cap.

## Exudation

Exudation, similar to perturbation, aims to relieve any slivers remaining in the mesh, as well as to re-smooth the pock-marks generated by the ODT portion of the pipeline. In Figure 4.18, many of these pock-mark artifacts are visibly smoother to some extent, but still exist in the mesh. Interestingly, Exudation has increased

the angle of attack of the lip in the bottom left corner of the L-beam. This can be attributed to [[ the way exudation operates, and why ]].

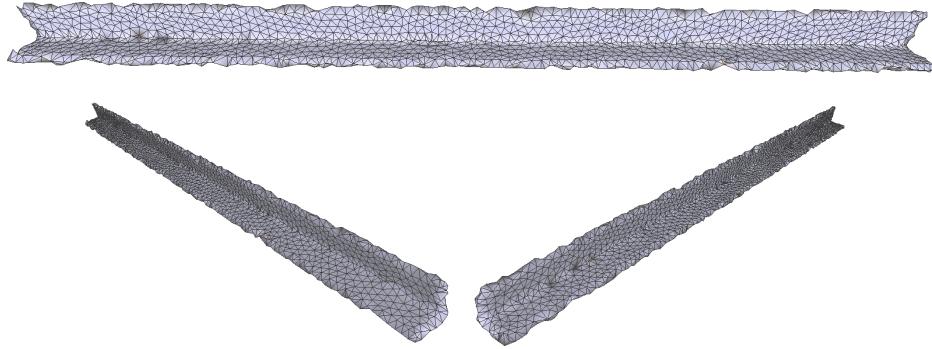


Figure 4.18: Result of Advancing Front mesh after exudation with a 30 second clock cap.

The scale space reconstruction in the zero noise case is interesting. In future work, minimum volume parameters suitable for zero-noise cases will be quantified. Figure 4.19 shows a poor looking mesh as the final product of the algorithm. Non-uniform areas can be seen distributed throughout the edges, and the mesh is filled with pockmarks, which damage the overall quality level of the mesh.

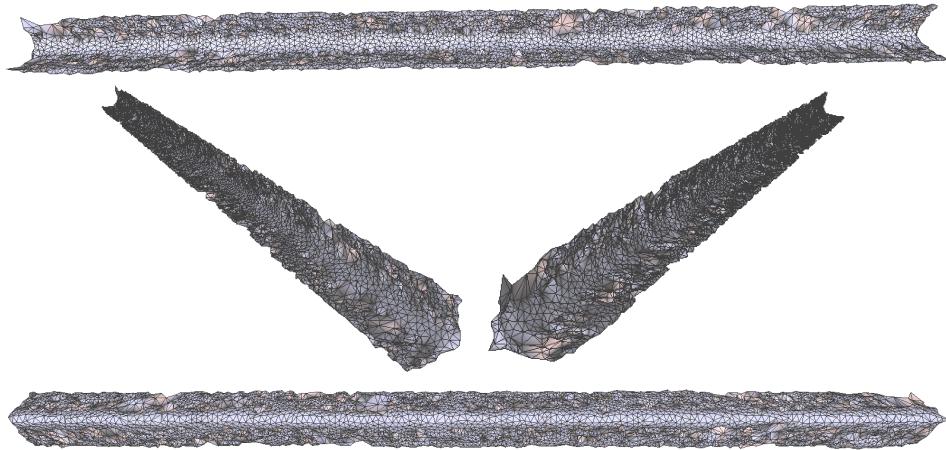


Figure 4.19: Scale space reconstruction  $S = 2$  after exudation with a 30 second clock cap.

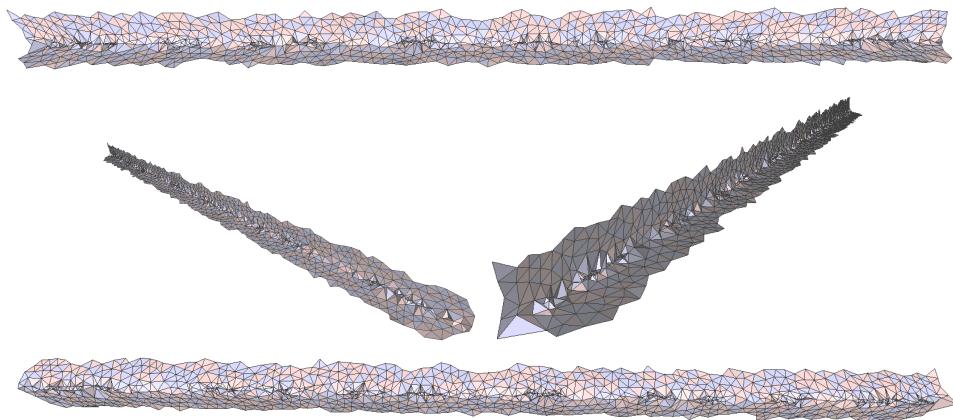


Figure 4.20: Scale space reconstruction  $S = 4$  after exudation with a 30 second clock cap.

#### 4.1.4 Quality Analysis

After applying the optimization steps, the meshes – with the exception of the Advancing Front method – do not increase in quality. In fact, they actually decrease. This phenomenon cannot be attributed to the thickness of the test beam with 100%

certainty, but this is the probable cause of failure, as the Scale Space reconstructions cast both sides of the beam to the same surface.

Table 4.2: Exit quality analysis of simulated zero noise surface mesh

Mesh Type	$Q_1$	$Q_2$	$Q_3$	$Q_4$	Bounded Surfaces	B-REP Compatible	FE Compatible
Initial Advacing Front	0.0548	0.2916	0.0270	0.0274	1	Yes	Yes
Optimized Advancing Front	0.3096	0.1910	1.333	0.1548	1	Yes	Yes
Initial Scale Space 2	0.0577	0.0665	0.0082	0.0587	1	No	No
Optimized Scale Space 2	0.0072	0.1085	0.0035	0.0037	1	Yes	No
Initial Scale Space 4	0.0038	0.0185	0.0005	0.0035	1	No	No
Optimized Scale Space 4	0.0031	0.0786	0.0016	0.0016	1	No	No

Quality values shown are defined in Section 2.4.1 and represent the absolute minima in each category.

Due to this casting issue, none of the reconstructions except for the Advancing Front method pass the conversion test.

## 4.2 Simulated Data: Applied Gaussian noise $\pm 2cm$

This section delves into the effect of noise introduced into the simulation data on the meshing and segmentation algorithms. Specifically, the aim is to determine how well each step of the process holds up to noise introduction, and make definitive decisions on the pipeline and parameters required at every step.

To simulate noise in the LiDAR scans, it is a simple matter of adding the measured uncertainty to distance metrics collected by the simulated sensor. It should be noted that in the current iteration of the simulator, no continuous surfaces are created, and the simulation relies on nearest cross product distances of points in the cloud to determine the sensor data return. An unfortunate artifact of this method is the inability to guarantee correct occlusion effects, as simulated rays can be cast through a continuous object created by a discrete pointset of non-infinite density.

In the experimental test run in this thesis, only one in every five LiDAR scan frames are used in the post-processing steps. The true LiDAR scan data is therefore reduced to 20% of the original sample set. This is mainly due to redundancy and unwieldy cloud sizes (700,000 points per second becomes unruly in a very short period of time). Figure 4.21 shows the result of the simulation at 20% sampling rate. There are 18,973 points in the cloud shown below. It is likely the true scan data will contain many more points before being down-sampled due to redundant point collection. The algorithm built here does not include the same target point more than once.

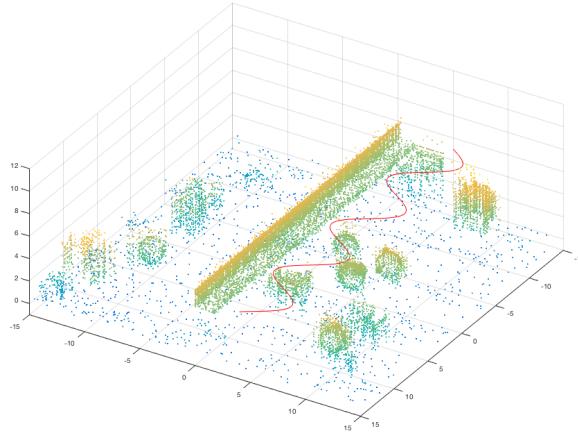


Figure 4.21: Simulated LiDAR scan of the virtual lab point cloud at 20% sampling rate. The red trail shows the collection path of the sensor.

Because the simulated scans do not capture occlusion effects, and are visibly low in noise, it is not suitable for testing of the algorithms robustness to noise. Instead, 2 cm gaussian noise is applied individually to the  $x$ ,  $y$ , and  $z$  coordinates of each point in the cloud. The result of this induction is shown in Figure ??.

As all three axes are introduced to this uncertainty, the total noise in this simu-

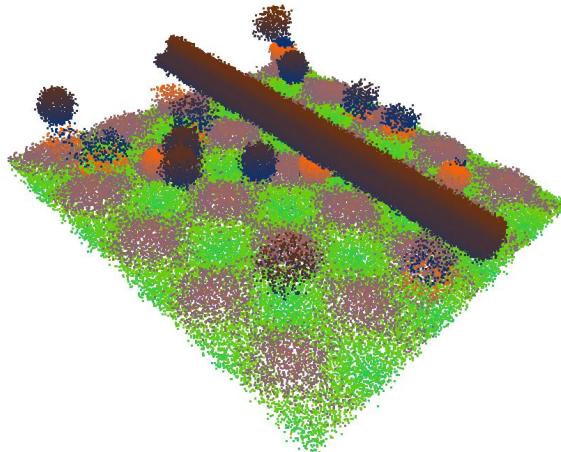


Figure 4.22: Simulated point cloud with  $2\text{ cm}$  magnitude gaussian noise induced.

lation dataset is greater than expected in true LiDAR scans. However, the level of occlusion in this data set is zero. All shapes are completely accounted for at every viewing angle.

The first step in the process is again to filter and sub-sample the point cloud. In Figure 4.23, the resulting point cloud is significantly cleaner than its untreated counterpart. Separating the cloud into clearly defined shapes is crucial for the unsupervised clustering algorithms to accurately and meaningfully segment the point cloud.

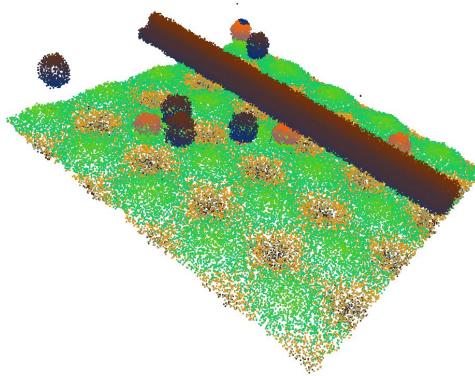


Figure 4.23: 2 cm gaussian noise simulated data after being filtered with voxel size of  $0.1in^3$ , 15 minimum point neighbors at a distance of 1 standard deviation from the mean point to point distance.

#### 4.2.1 Segmentation Method Comparison

The same segmentation methods shown in the noiseless case are compared once again in the induced noise case. As true experimental datasets are never entirely noiseless, the analysis on the induced noise case provides much more significant insight on the true abilities of each segmentation method to meaningfully cluster the data.

Using the same parameters as the noiseless case, euclidean clustering once again proves its vast superiority to retrieve the shape of interest accurately and cleanly. For the remaining sections in this chapter, only beam segments created by euclidean clustering are analyzed.

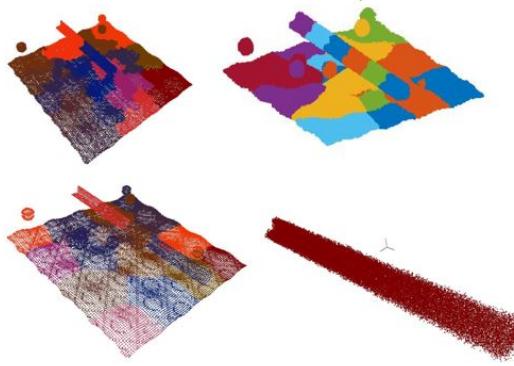


Figure 4.24: Resulting cloud cluster from a) k-means, 17 centroid b) Fuzzy c-means, 17 centroids c) Agglomerative clustering, 17 bins and d) Euclidean clustering with radius of 0.5 cm minimum cluster size of 3,000 and maximum cluster size of 50,000 points.

#### 4.2.2 Initial Mesh Methods

While Advancing Front touted itself as the superior method in the zero noise case, it has no measures to prevent corruption of the mesh in the case where noise is apparent in the dataset. Figure 4.25 shows a completely disjointed mesh object, with many small bounded surface areas, and a large number of disconnected half edges. Because of these small bounded areas, the hole filling methods implemented in the algorithm are unable to remesh the object into a comprehensive surface area. It can be said with certainty that the Advancing Front method fails the search criteria for non-ideal pointclouds.

Where Advancing Front fails, the Scale Space reconstruction method excels in its ability to compensate for noise. Figure 4.26 shows the initial meshing results using a scale space of 2. Note the uniformity in the mesh, and the singular surface area the

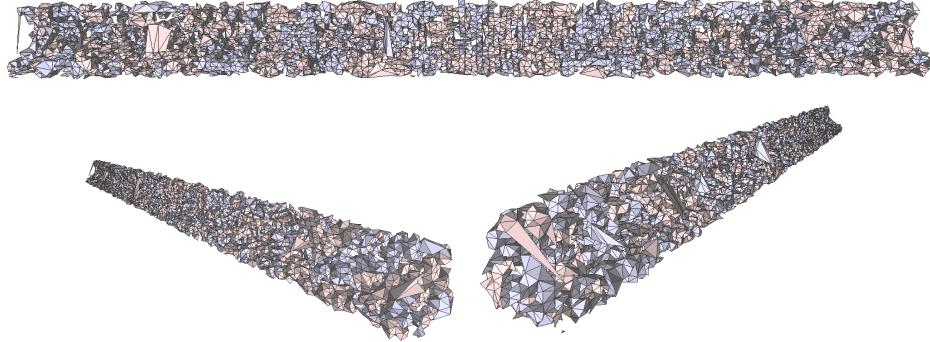


Figure 4.25: Result of initial meshing using the advancing front method.

mesh defines. This is unsurprising, as the scale space reconstruction method is built specifically for high noise and high variability cases.

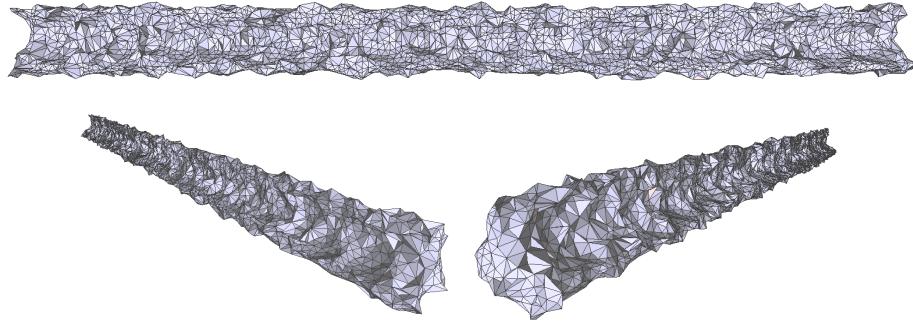


Figure 4.26: Result of initial meshing using the scale space reconstruction method with  $S = 2$ .

The scale space reconstruction, when the casting scale is increased to 4, once again shows a unified singular surface area, but suffers from the effects of the beam's lack of thickness seen in Figure 4.6 and Figure 4.7. At this point, it is too early to make a definitive statement about whether scale space 4 is too drastic of a casting, but this method is re-evaluated at the end of the optimization process.

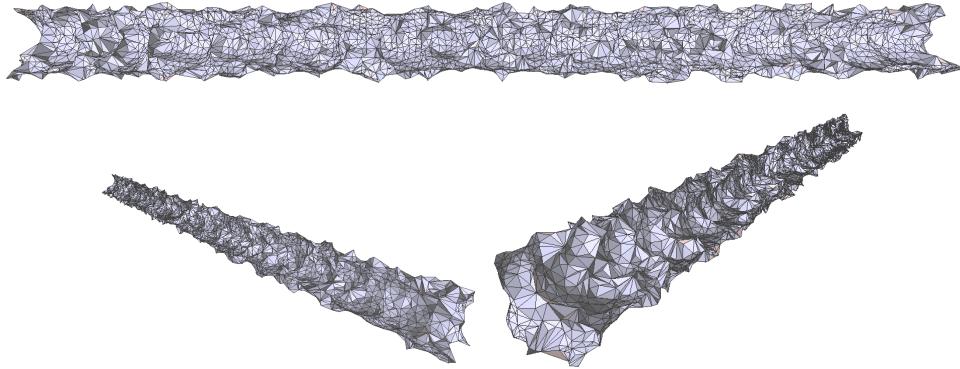


Figure 4.27: Result of initial meshing using the scale space reconstruction method with  $S = 4$ .

The Advancing Front reconstruction is once again ranked the highest in every quality metric used, but is visibly not suitable for B-REP conversion. Quality metrics are good for determining the level of information each triangulation provides to the mesh, but do not directly coincide with suitability for conversion. The metric which makes clear the incompatibility of conversion is the number of individual bounded surfaces in the mesh. Any number above 1 means there are volumes connected at a single point. Unfortunately, these are difficult to remedy without complete removal of shapes, as the surfaces individually do not contain any half-edges, which are fixed during the hole-filling section of the algorithm.

Table 4.3: Initial quality analysis of simulated 2 centimeter noise surface mesh.

Mesh Type	$Q_1$	$Q_2$	$Q_3$	$Q_4$	Bounded Surfaces	B-REP Compatible	FE Compatible
Advacing Front	0.1518	0.0877	0.0133	0.0924	314	No	No
Scale Space 2	0.0402	0.0594	0.0088	0.0259	1	Yes	No
Scale Space 4	0.0080	0.0592	0.0035	0.0043	1	No	No

Quality values shown are defined in Section 2.4.1 and represent the absolute minima in each category.

#### 4.2.3 Optimization

Before traversing into the individual optimization sections, it is important to note that the advancing front method has been removed from the analysis due to its clear inability to create a singular, comprehensive surface area. It is a waste of processing power and space, to further analyze this meshing technique.

#### Effects of Voronoi Relaxation

As before, a voronoi relaxation algorithm is applied to the mesh, with exit criteria of convergence, or a maximum of 200 iterations. Clearly visible in Figure 4.28, the reconstruction at scale space 2 is relatively smooth compared the the raw point cloud, and exists as a singular bounded surface area.

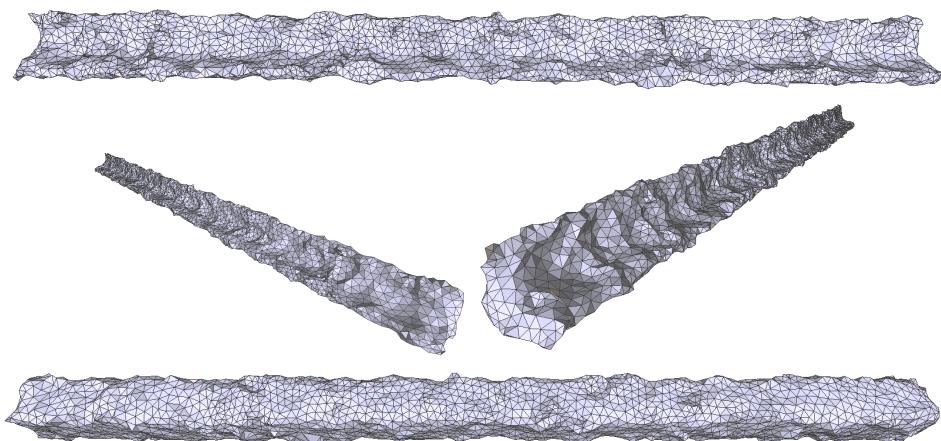


Figure 4.28: Scale space reconstruction  $S = 2$  after 200 iterations of voronoi relaxation.

Voronoi relaxation applied on the reconstruction at scale space 4 has done some work to remedy the highly visible issues with the initial mesh. On the top left corner,

the section no longer has zero implied volume, as the mesh has ballooned out slightly. However, there are still a series of discontinuities and sharp edges existant in the mesh. The next sections show how ODT, perturbation, and exudation affect these sharp edges.

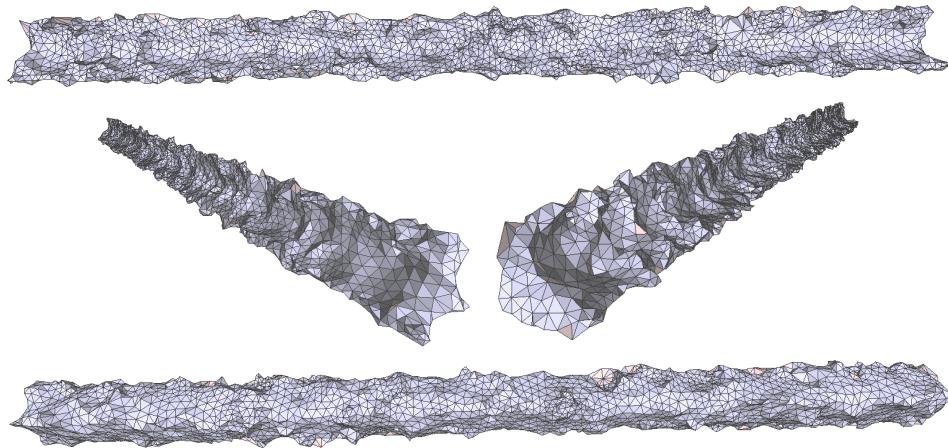


Figure 4.29: Scale space reconstruction  $S = 4$  after 200 iterations of voronoi relaxation.

### Effects of Delaunay Optimization

Delaunay optimization has some more noticeable effects on the noisy point cloud than on the clean point cloud – unsurprising. These effects are most visible in Figure 4.30 on the lefthand side of the mesh. The vertices of the mesh are pushed further in the normal direction of the surface area of the shape, creating a more even mesh, that is arguably more accurate to the true shape of the input cloud.

The effects listed above are even more visible in scale space 4. Figure 4.31 shows the extent ODT can go to to increase the quality in a mesh. The portion in the top

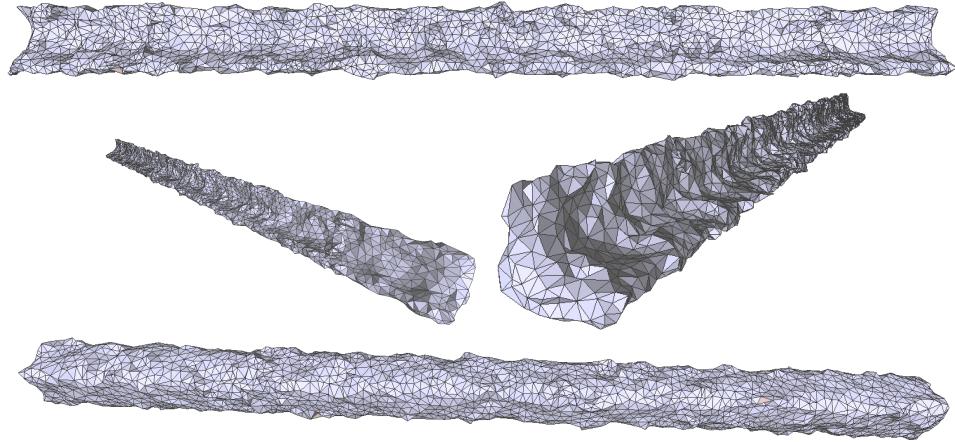


Figure 4.30: Scale space reconstruction  $S = 2$  after ODT with a 30 second clock cap.

left corner of the mesh – initial a nearly 2-dimensional area – is now defined as a shape with definite volume, and only slightly thinner than its surrounding edges.

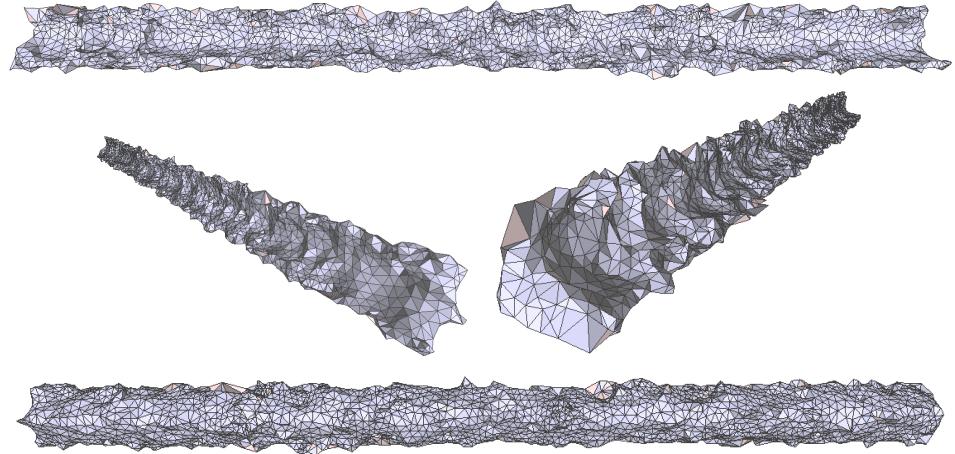


Figure 4.31: Scale space reconstruction  $S = 4$  after ODT with a 30 second clock cap.

Because the reconstructions at scale 2 and 4 are already quite high in quality metrics, ODT did not shift many vertices noticeably. For this reason, the same number

of blemishes and pockmarks ODT created in the previous section (Figure 4.13 and Figure 4.14) are not seen here. There are, most likely, a number of slivers still incorporated in the meshes.

### Effects of Mesh Perturbation

The algorithm will now attempt to remove the remaining slivers using perturbation with a 30 second time limit if convergence is not met. Figure 4.32 shows the mesh at scale space 2 after perturbation. In this case, perturbation does not return any overtly visible changes. It is likely that none or very few of the tetrahedra in the mesh met the criteria defining a sliver. This being the case, changes in the mesh would be very difficult to detect by visual inspection.

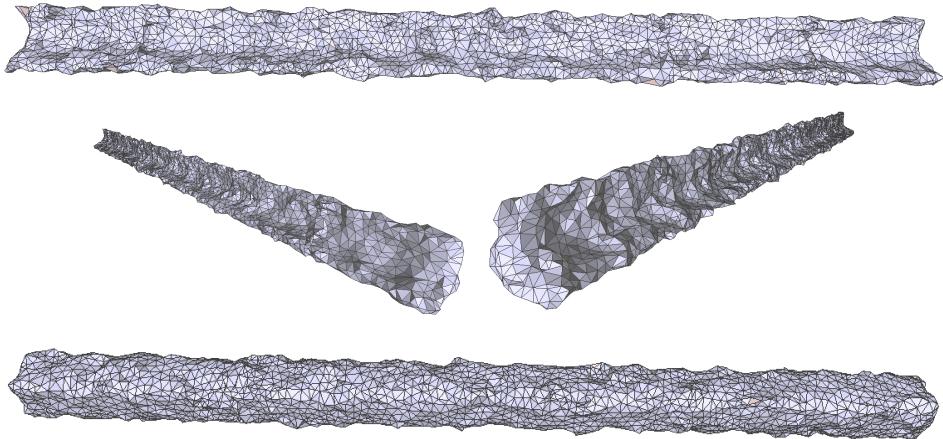


Figure 4.32: Scale space reconstruction  $S = 2$  after perturbation with a 30 second clock cap.

Changes in scale space 4 reconstruction are subtle as well [[details]].

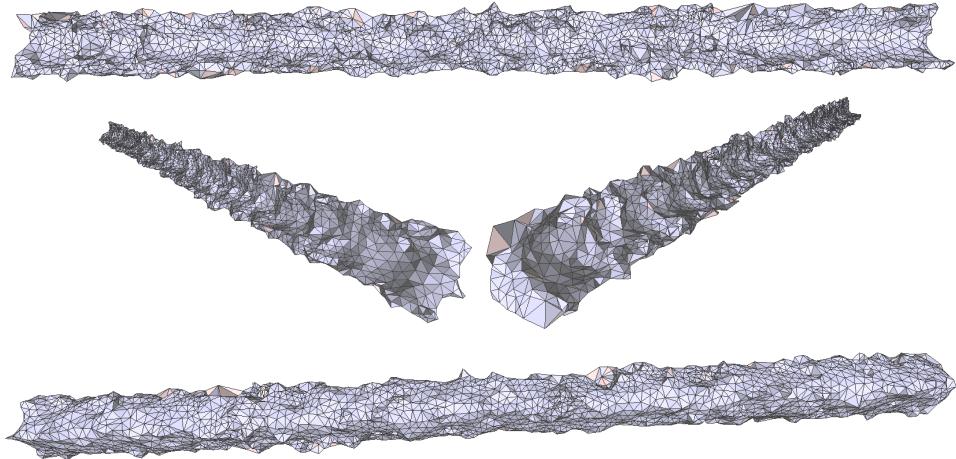


Figure 4.33: Scale space reconstruction  $S = 4$  after perturbation with a 30 second clock cap.

### Exudation

Changes due to exudation are more visible than perturbation in this case. Looking again at the lefthand side of the beam, there is a more uniform slope in the shape of the beam. While this modifies the topology of the mesh in a way that does not meet the true definition of the beam, it provides a smoother and more uniform mesh shape. Figure 4.34 shows the final optimized mesh at a scale space of 2.

Scale space 4 exaggerates the parabolic shape of the beam even more than scale space 2 – again, unsurprisingly, as heavier interpolation is applied as the reconstruction is moved to higher scales. Viewing the final mesh in Figure 4.35, an evenly distributed set of tetrahedra defining the relatively smooth surface area of a beam with a parabolic cross-section is visible.

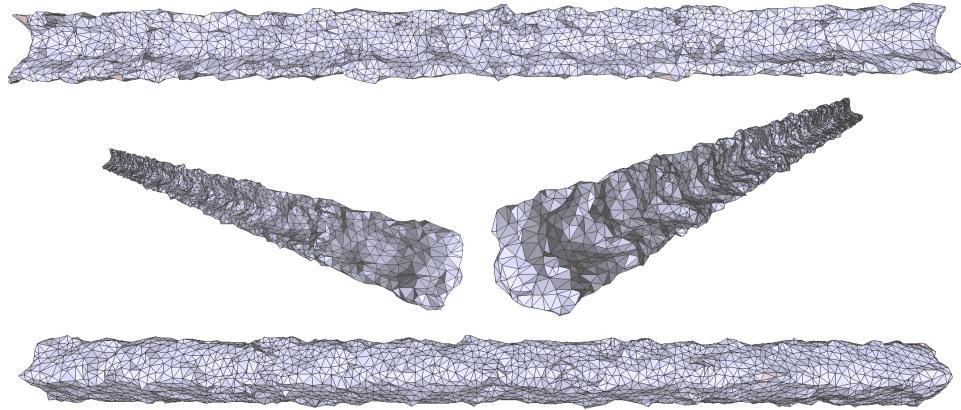


Figure 4.34: Scale space reconstruction  $S = 2$  after exudation with a 30 second clock cap.

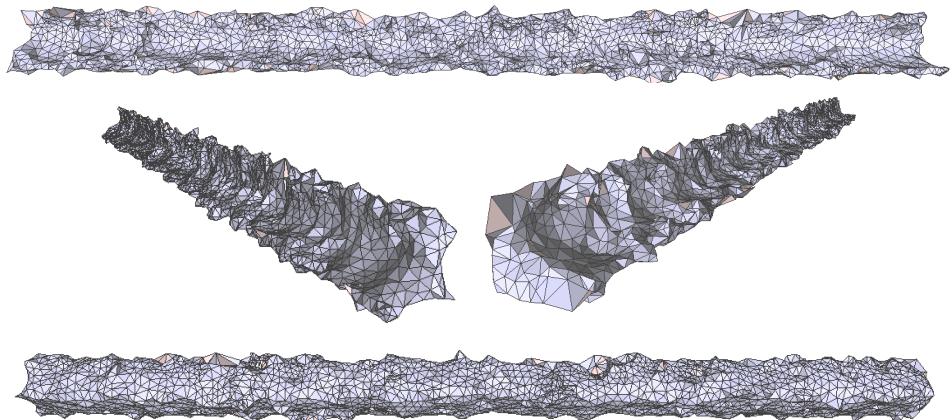


Figure 4.35: Scale space reconstruction  $S = 4$  after exudation with a 30 second clock cap.

#### 4.2.4 Quality Analysis

At the end of the optimization phase, both Scale Space reconstructions are compatible with B-REP conversion. In both methods there is also an uptrend in all quality metrics of an entire order of magnitude. At this point, the ability to quanti-

tatively say whether or not a mesh will meet criteria is not available, but the range of values this can happen is beginning to narrow.

Table 4.4: Exit quality analysis of simulated 2 centimeter noise surface mesh.

Mesh Type	$Q_1$	$Q_2$	$Q_3$	$Q_4$	Bounded Surfaces	B-REP Compatible	FE Compatible
Initial Scale Space 2	0.0402	0.0594	0.0088	0.0259	1	Yes	No
Optimized Scale Space 2	0.1367	0.1077	0.0147	0.0955	1	Yes	Yes
Initial Scale Space 4	0.0080	0.0592	0.0035	0.0043	1	No	No
Optimized Scale Space 4	0.1166	0.1444	0.0552	0.0595	1	Yes	Yes

Quality values shown are defined in Section 2.4.1 and represent the absolute minima in each category.

### 4.3 HDL-32E LiDAR Scan Data

To test the true robustness of the algorithm, a real scan of a real object of potential interest is necessary. The object for the following test is a 10 foot long beam, with a width and height of 5 inches, and a thickness of 0.25 inches.

Before viewing the test results, note the thickness of the beam is smaller than the rated accuracy of the LiDAR device. This combined with sensor mobility limited to the xy plane at a height similar to that of the beam proved to cause a severely limited view of the entire beam.

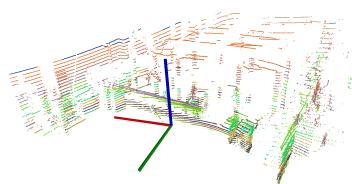


Figure 4.36: Individual frame from LiDAR scan. Image contains 32,000 points.

Concatenating the frames recorded over the entire 30 second test would result in a

point cloud of 210 million points, so instead the concatenation was created out of 100 frames deemed to be the most critical in terms of information provided to the beam's shape characteristics. Figure 4.37 shows the results of the concatenation before any filters are applied.

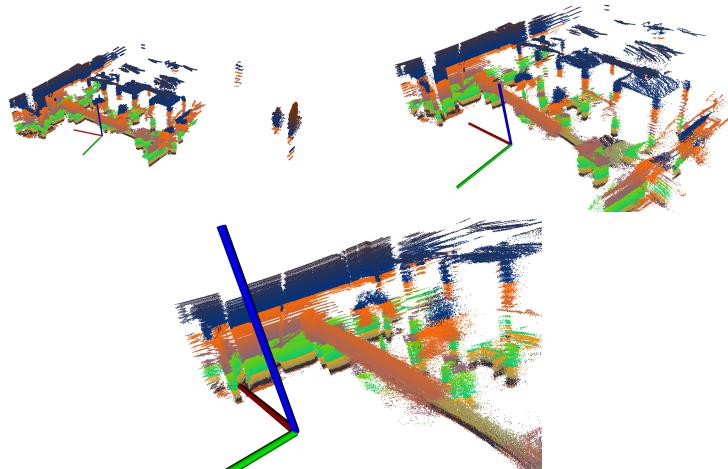


Figure 4.37: Concatenation of 100 individual LiDAR scan frames.

The same filtering method as the previous sections is applied to the concatenated point cloud, resulting in the much cleaner looking point cloud shown in Figure 4.38. The result of the concatenation and filtering is sufficient to move on to the segmentation step. The full concatenation images do not sufficiently show the accrued uncertainty in the point cloud, but the effects of concatenating multiple frames without using instrument location/pose data becomes apparent when the beam is isolated.

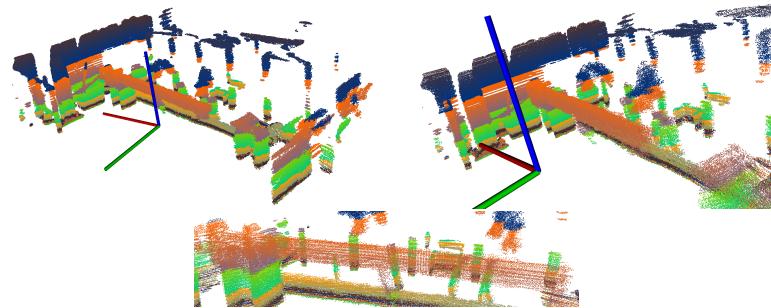


Figure 4.38: Concatenated LiDAR scans after noise filtering and down sampling.

#### 4.3.1 Segmentation Method Comparison

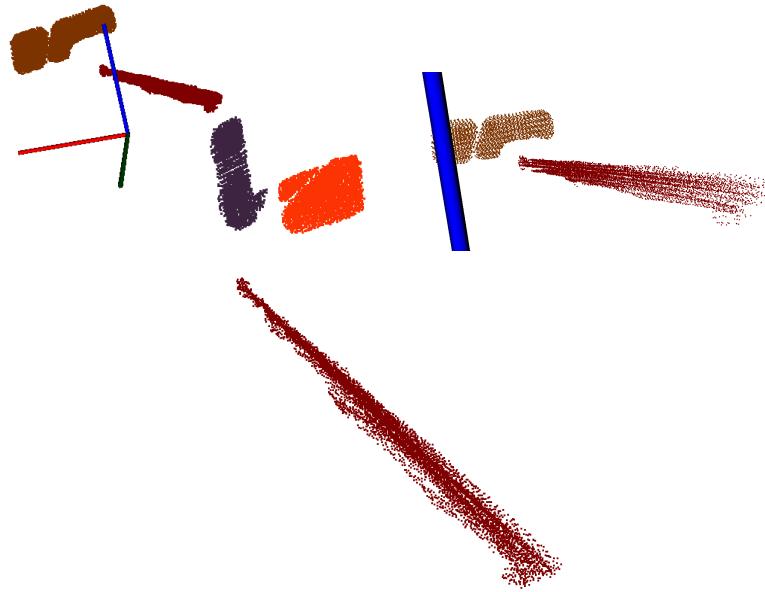


Figure 4.39: Results after euclidean clustering.

### 4.3.2 Initial Meshing Methods

The effects of the collection path and method are apparent in Figure 4.39. The radial data collection of the LiDAR device combined with the xy planar motion lock of the test path creates an uneven horizontal beam surface, almost occupying a sinusoidal wave space. This, along with the lack of thickness data provided by the LiDAR sensor, greatly affects the final topology of the mesh. Lack of information and noise-induction in datasets is a routine and unavoidable factor in discrete sensing, so an algorithm that can optimize a CAD compatible mesh topology with incomplete or inaccurate datasets is invaluable in the sensing field.

Figure 4.40 shows the results of a raw advancing front meshing procedure on the segmented beam point cloud. Without any compensation in the form of noise reduction or shape complexity reduction, the resulting surface mesh from this method is not one comprehensive surface area, but a series of tangentially connected shapes. Because many of these smaller shapes are bounded within themselves, they do not contain half edges, and are not picked up by the hole filling algorithms implemented in the pipeline. For non-ideal datasets, noise compensation is required for a comprehensive mesh to be created.

Figure 4.41 shows the results of the LiDAR collected dataset using the scale space reconstruction method, at a scale of 2. While the mesh does not represent the beam accurately, and does not meet all of the criteria for a B-REP conversion compatibility, the topology represents one continuous surface, and is far smoother than the original input data. This mesh can be seen to fail by visual inspection. In the [[bottom right

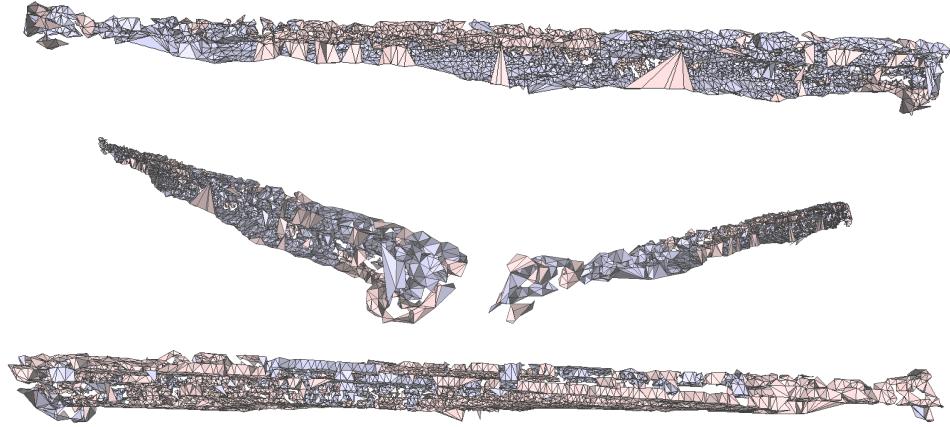


Figure 4.40: Advancing front reconstruction of resulting beam point cloud.

image (label these images individually)], two spires can be seen at the bottom of the beam. Beyond this, the horizontal section of the beam is still represented as a volumeless surface. This is not compatible with CAD conversion.

Reducing the input point cloud to scale space 4 provides similar results to scale space 2, the difference being in the smoothing level. This extra smoothing is most visible in the [[first image on the left]]. There are far less ridges in the vertical face than in scale space 2. While moving to higher scale spaces almost always guarantees a smoother reconstruction, it comes at the price of moving farther away from the true collected topology of the object.

None of the initial meshing methods create an instantaneously convertible surface mesh in this case. There are a number of reasons for this that include artifacts from the cloud concatenation process, noise in the collection data, and a lack of depth data provided for the beam.



Figure 4.41: Scale space reconstruction at scale  $S=2$  of resulting beam point cloud.

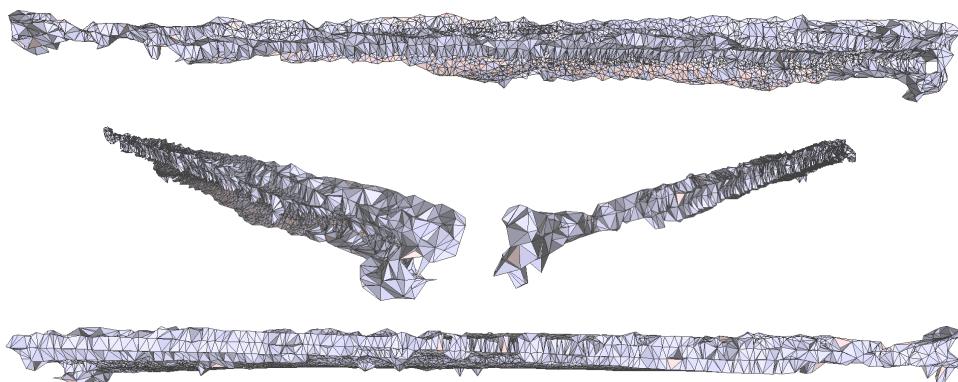


Figure 4.42: Scale space reconstruction at scale  $S=4$  of resulting beam point cloud.

Table 4.5: Quality analysis of initial meshing methods.

Mesh Type	$Q_1$	$Q_2$	$Q_3$	$Q_4$	Bounded Surfaces	B-REP Compatible	FE Compatible
Advacing Front	0.1446	0.1072	0.0419	0.0849	28	No	No
Scale Space 2	0.0175	0.0524	0.0066	0.0093	2	No	No
Scale Space 4	0.0117	0.0562	0.0057	0.0059	2	No	No

Quality values shown are defined in Section 2.4.1 and represent the absolute minima in each category.

### 4.3.3 Optimization

Unlike the simulated cases, none of the initial methods for the LiDAR collection dataset produce fully compatible meshes on a first pass. This is undoubtedly due to the incorporation of occlusion in the dataset. The steps involved in the optimization process smooth the input meshes which meet the baseline criteria of having a single surface area to solid conversion compatible surface meshes, but the topology of the resulting mesh is not indicative of the true beam shape. To maintain the true shape, more informative point cloud data is necessary, or informed shape estimation must be used to rebuild the mesh to be more indicative of its true to life properties.

#### Effects of Voronoi Relaxation

As in previous sections, the first step in the optimization process is voronoi relaxation – otherwise known as voronoi relaxation. The effect of voronoi relaxation on a mesh is the redistribution of surface area throughout the mesh triangulations. The exit conditions for relaxation is convergence of mesh centroids. Figure 4.43 demonstrates what a voronoi relaxation does for the LiDAR collected point cloud after being reconstructed at a scale space of 2.

After running the voronoi relaxation on the reconstruction at scale 4, the differences in the meshes between scales becomes more apparent. The evenly distributed surface defines a rectangular prism, the horizontal section of the beam merges into its vertical face. [[Second image on the left]] shows the phenomenon in detail, in Figure 4.44.

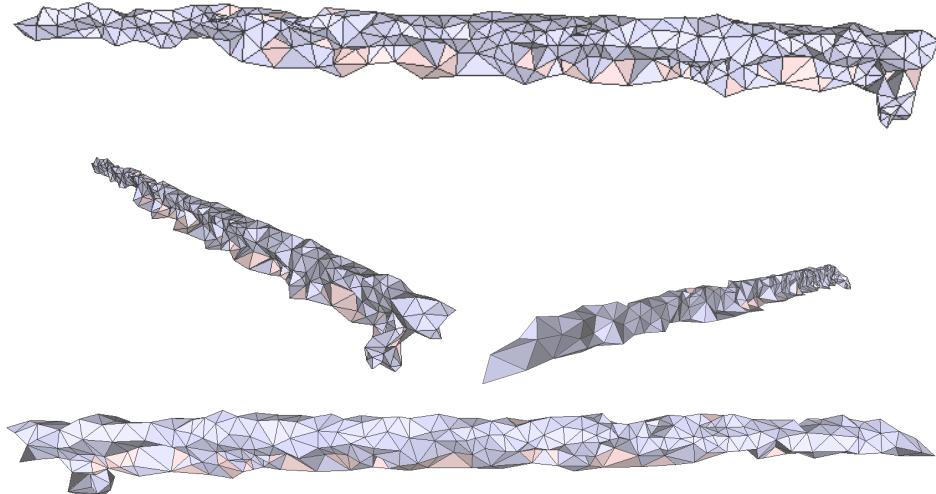


Figure 4.43: Lloyd + Scale space reconstruction at  $S=2$  of resulting beam point cloud.

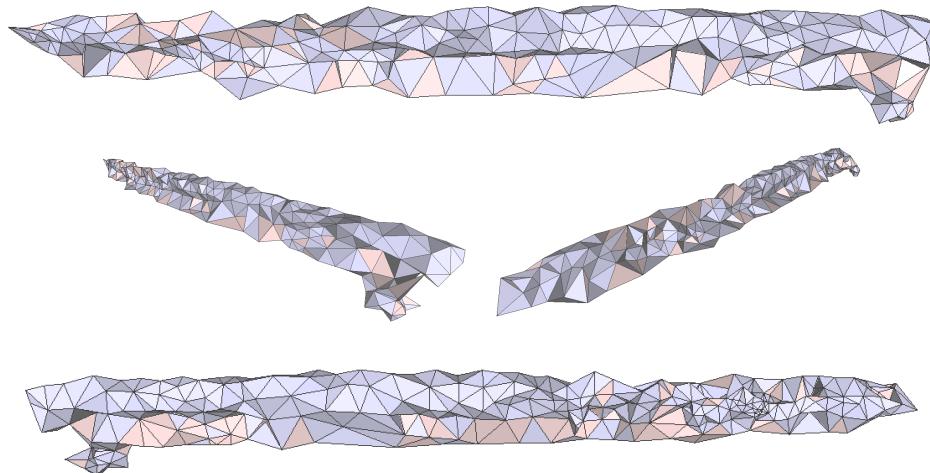


Figure 4.44: Lloyd + Scale space reconstruction at  $S=4$  of resulting beam point cloud.

### Optimized Delaunay Triangulation

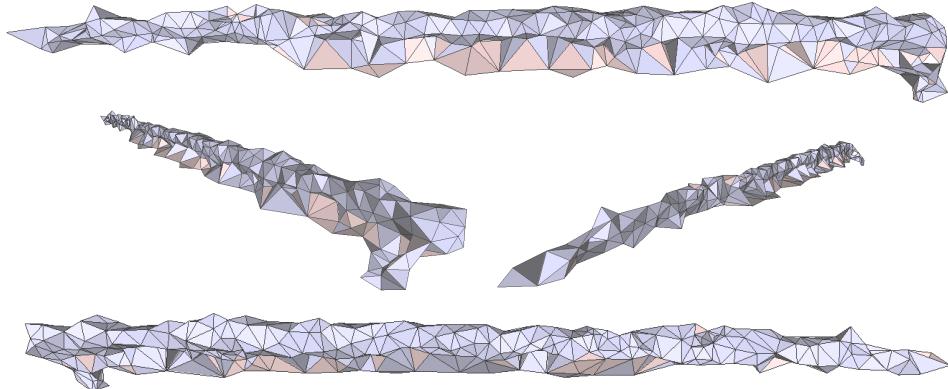


Figure 4.45: Lloyd + ODT + Scale space reconstruction at  $S=2$  of resulting beam point cloud.

The differences in the mesh before and after ODT is applied are subtle, but the changes are visible. In Figure 4.46 the mesh can be seen after the ODT optimization step. The main differences here are in the [[define the main differences in terms of what ODT does]]

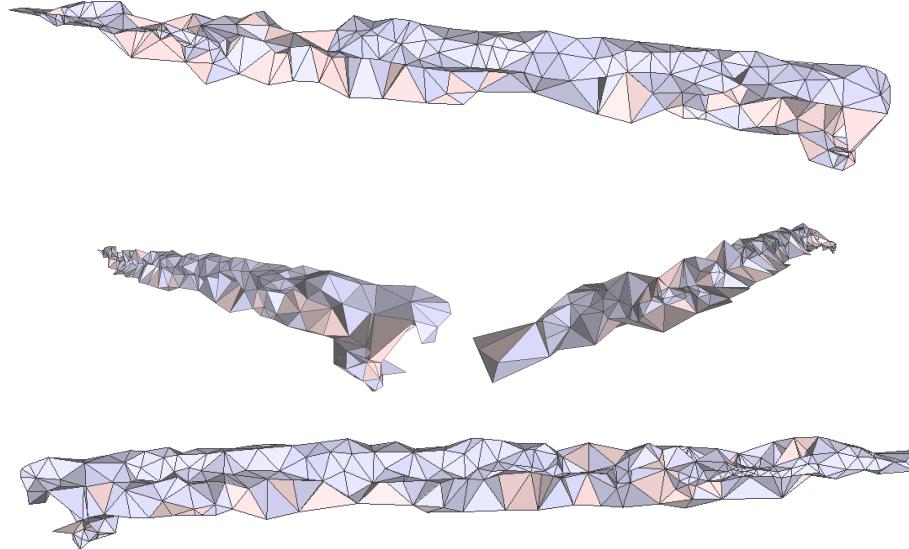


Figure 4.46: Lloyd + ODT + Scale space reconstruction at  $S=4$  of resulting beam point cloud.

### Perturbation

The final steps in the optimization process, perturbation and exudation, aim to resolve any remaining poor quality tetrahedra in the surface topology. At this point, the only shapes designated as poor quality left in the mesh are slivers, identified in Table 2.1 as tetrahedra with low volume ratios. In Figure 4.47, a ballooning effect on the mesh can be seen, especially visible in the vertical face of the beam. At this point, even at scale space 2, the optimization process has almost entirely removed the horizontal portion of the beam.

The reconstruction at scale space 4 again shows similar results to the scale space 2 reconstruction, with the exception of the lower forward edge of the beam shown in

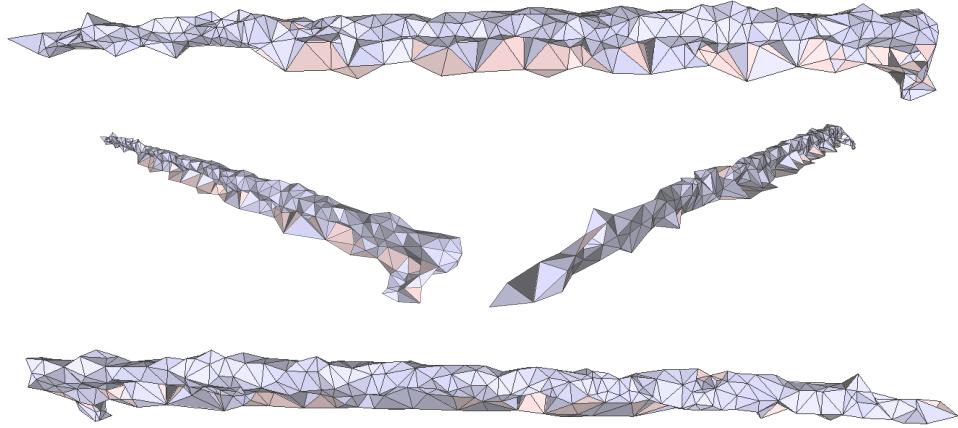


Figure 4.47: Lloyd + ODT + Perturb + Scale space reconstruction at  $S=2$  of resulting beam point cloud.

[[the upper lefthand corner image]] of Figure 4.48. An artifact of the casting space, this corner does not extrude from the beam's surface as far as the case of scale space 2.

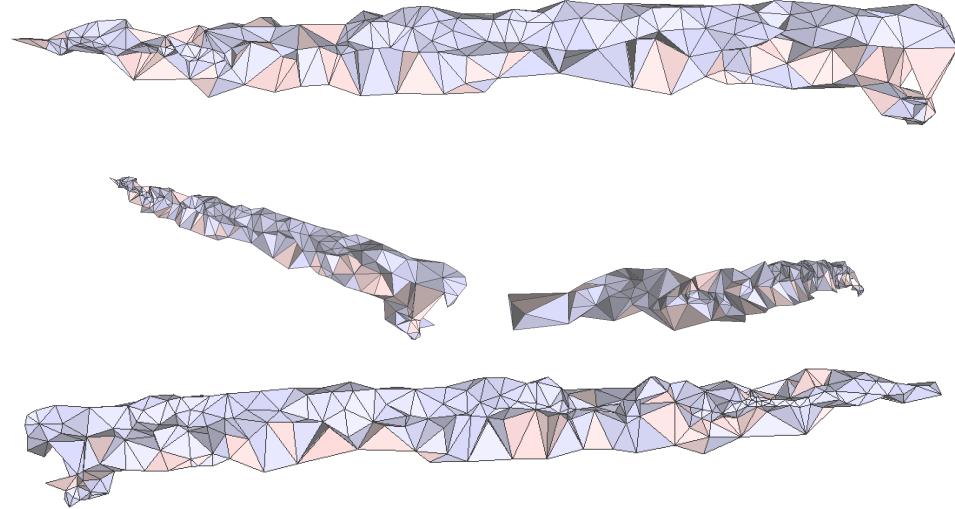


Figure 4.48: Lloyd + ODT + perturb + Scale space reconstruction at  $S=4$  of resulting beam point cloud.

### Exudation

Exudation represents the final step in the optimization process, and the complete surface mesh to be exported to a surface-to-solid conversion software. Analyzing the final results, it is clear – unsurprisingly – that the quality and completeness of the input point cloud is crucial in developing a mesh topology that is both B-REP compatibility and accurately defining of the shapes true parameters. After the exudation step, the meshes shown in Figures 4.49 and 4.50 have the quality and failure metrics shown in Table 4.6.

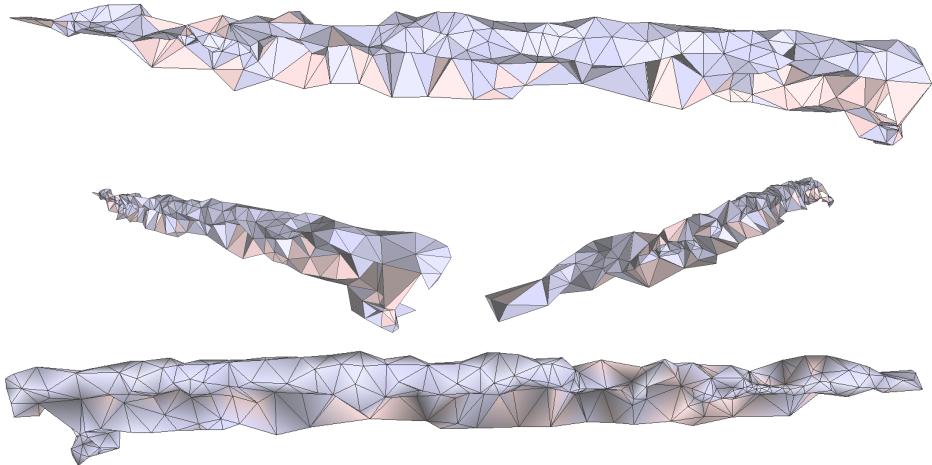


Figure 4.49: The full suite of optimization steps on the LiDAR reconstruction at  $S=2$ .

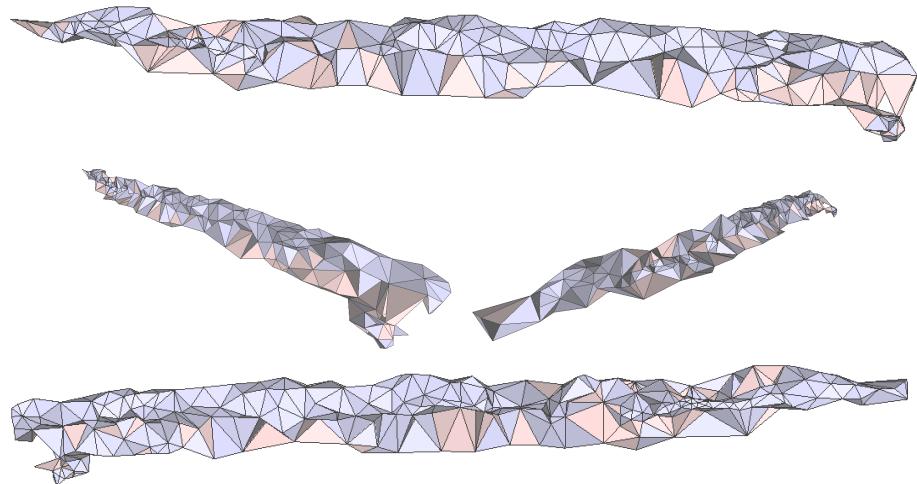


Figure 4.50: The full suite of optimization steps on the LiDAR reconstruction at  $S=4$ .

#### 4.3.4 Quality Analysis

The quality of each Scale Space reconstruction is improved vastly after optimization. However, the meshes are not singular volumes, and contain two individual

shapes each, connected at a single point. None of the meshes convert successfully to solid models as they are presented at this point.

Table 4.6: Exit quality analysis of experimental beam scan.

Mesh Type	$Q_1$	$Q_2$	$Q_3$	$Q_4$	Bounded Surfaces	B-REP Compatible	FE Compatible
Initial Scale Space 2	0.0175	0.0524	0.0066	0.0093	2	No	No
Optimized Scale Space 2	0.2629	0.1941	0.1099	0.1418	2	Yes	No
Initial Scale Space 4	0.0117	0.0562	0.0057	0.0059	2	No	No
Optimized Scale Space 4	0.2143	0.1229	0.0558	0.1087	2	Yes	No

Quality values shown are defined in Section 2.4.1 and represent the absolute minima in each category.

The end result of the algorithm on the experimental data, when fully automated, is failure in terms of full finite element compatibility, but successful for B-REP conversion. Finite element failure in these cases is not due to overlapping tetrahedra or an unbounded area, but instead because of the single point linkage between the two bounded surfaces both optimized meshes produce. By removing the small surface located at the bottom of the far right end of the beam, the resulting quality metrics are shown in Table 4.7 below.

Table 4.7: Exit quality analysis of experimental beam scan after manual modification.

Mesh Type	$Q_1$	$Q_2$	$Q_3$	$Q_4$	Bounded Surfaces	B-REP Compatible	FE Compatible
Optimized Scale Space 2	0.2629	0.1941	0.1099	0.1418	1	Yes	No
Optimized Scale Space 4	0.2143	0.1229	0.0558	0.1087	1	Yes	Yes

Quality values shown are defined in Section 2.4.1 and represent the absolute minima in each category.

Why the mesh at Scale Space 2 fails the finite element conversion at this point is still unclear. It is possible that there are unobservable sections of low or zero volume, or that there are overlapping sections inherent in the mesh. Interestingly, All of the quality metrics for Scale Space 2 are higher than those at Scale Space 4, so all expected parameters are met. Scale Space 2 serves as proof that although

the metrics used to evaluate mesh quality in this thesis are valuable in estimating a mesh's surface-to-solid compatibility, they are not the only metrics that determine the ability to do so.

The modified surface meshes are shown in Figures 4.51 and 4.52.

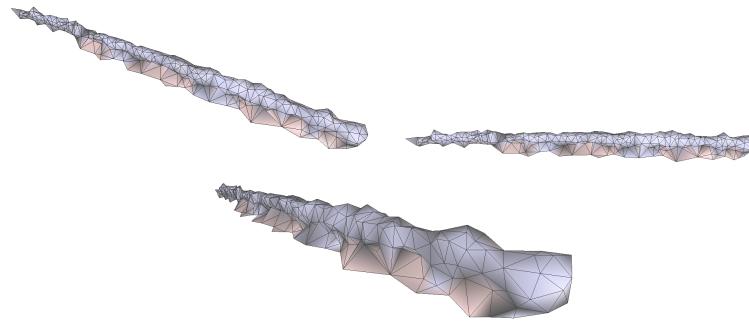


Figure 4.51: Manually corrected optimized scale space 2 reconstruction. [[Compare with Figure 4.49.]]

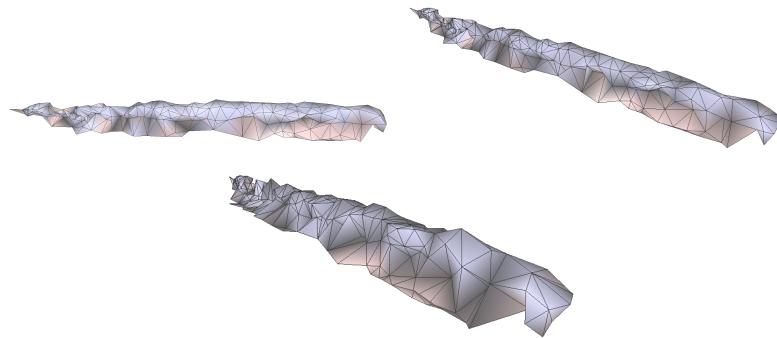


Figure 4.52: Manually corrected optimized scale space 4 reconstruction. [[Compare with Figure 4.50.]]

## 5. Conclusion

Analyzing the results from the data collected above, the first conclusion that can be made is simple: If there is more than one bounded volume in the connected mesh, conversion to a singular solid volume fails. The other quality metrics recorded are found to tie more closely with finite element conversion compatibility than with solid volume compatibility.

The rest of the quality metrics are shown in Table 5.1.

Table 5.1: Comparison of final mesh parameters.

Mesh	$Q_1$	$Q_2$	$Q_3$	$Q_4$	Bounded Surfaces	B-REP Compatible	FE Compatible
Zero noise AF	0.3096	0.191	1.333	0.1548	1	Yes	Yes
Zero noise SS2	0.0072	0.1085	0.0035	0.0037	1	Yes	No
Zero noise SS4	0.0031	0.0486	0.0016	0.0016	1	No	No
2cm noise SS2	0.1367	0.1077	0.0147	0.0955	1	Yes	Yes
2cm noise SS4	0.1166	0.1444	0.0553	0.0595	1	Yes	Yes
Experimental SS2	0.2629	0.1941	0.1099	0.1418	2	Yes	No
Experimental SS4	0.2143	0.1229	0.0558	0.1087	2	Yes	No

AF – Advancing Front, SS – Scale Space

Evaluating these metrics based on successful conversion to finite elements, the trends in the data can be interpreted as the following:

Table 5.2: Analysis of significant quality metrics

Metric	Minimum Success Value	Maximum Failure Value*
$Q_1$	0.1166	0.0072
$Q_2$	0.1077	0.1085
$Q_3$	0.0553	0.0035
$Q_4$	0.0595	0.0037

\*Excludes experimental data meshed at scale space 2.

As seen in the experimental meshing section, these parameters are not the be-all

end-all of surface-to-solid compatibility, but they are definitely related to a mesh's ability to be converted to a solid volume.

### 5.1 Finite Element Models of Compatible Meshes

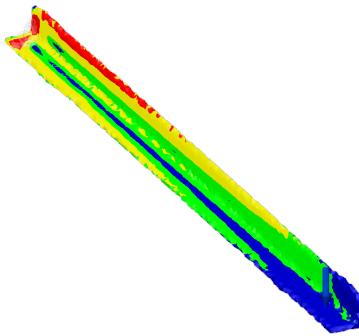


Figure 5.1: The zero noise beam, using the advancing front reconstruction, is subjected to a static cantilever type load of [[load]].

### 5.2 Algorithm Programmability

The algorithm built is highly customizable. At each step in the pipeline, the defining parameters – and whether or not that step is implemented at all – are entirely controllable as input values.

Table 5.3: Full algorithm flow with all user defined parameters available.

Algorithm Step	Default method and parameters
Individual frame filtering	Noise filter: Minimum neighbors $N_1$ , standard deviation $\sigma_1$ Down-sample: voxel size $v_1$
Frame Concatenation	Intrinsic Shape Signature Stitching: frame step size $s$
Concatenated cloud filtering	Noise filter: Minimum neighbors $N_2$ , standard deviation $\sigma_2$ Down-sample: voxel size $v_2$
Object segmentation	K-means: Number of bins $k$ Fuzzy C-means: Number of bins $k$ Agglomerative Clustering: Number of bins $k$ Euclidean Clustering*
Initial Meshing Method	Advancing Front Scale Space Reconstruction: scale space $S$
Hole filling and refinement	Refinement ratio $r$
Optimization steps	Voronoi relaxation** Optimal Delaunay triangulation** Perturbation** Exudation**

\*2-norm distance  $d$ , min cluster size  $c_{min}$ , max cluster size  $c_{max}$

\*\*max iterations  $i$ , max time  $t$ , convergence tolerance  $\Delta$

## 6. Future Work

### 6.1 Guaranteed FEA compatibility

[[]]

### 6.2 Informed Shape Estimation

Occlusion is the cause of two main problems. The first being the unnecessary information provided by objects blocking the instruments view to the desired target, which is dealt with via the diffusion of objects from the foreground into the background via the range segmentation method show in the previous section. The second — far more relevant to the approach outlined in this thesis — is the lack of complete information provided by the sensor. This can be most easily explained by making an analogy to photography. If one takes a picture of the front of a box, there is no possible way to say with certainty what the back of the box looks like. In 3D point processing, the problem is the same. Informed shape estimation attempts to tackle this problem by applying a shape with known parameters to the object point cloud and modifying the shape parameters to minimize the following cost function:

$$C = \sum_{i=0}^N p(i) - p_{proj}(i) \quad (6.1)$$

Where  $N$  represents the number of points in the set,  $p(i)$  represents the  $i^{th}$  point in the set, and  $p_{proj}(i)$  represents the closest point on the applied shape who's unit

normal vector matches within some tolerance,  $\tau$ .

$$p_{proj}(i) = \arg\min \frac{p(i) \bullet l(j)}{\|l(j)\|} \in p_{normal}(i) \bullet l_{normal}(j) \geq \tau \quad (6.2)$$

$$x_{k+1} = x_k - \frac{C(x_k)}{C'(x_k)} \quad (6.3)$$

Using a Newton Raphson interative regression, shown in equation 6.3 the cross section parameters are modified to minimize the cost required to projection points to the estimated surface. This technique is iterated throughout the long axis of the object, allowing for crucial information such as deformation to be retained. The power in this method is in its ability to fill information in heavily occluded areas, at the cost of having a narrow scope. Informed shape estimation makes the following assumptions:

1. Objection deformation is planar. Calculating the length plane removes the information involving multi-axial deformation.
2. Cross-section is undamaged throughout the length of the object. Information regarding damage which alters the objects cross section will be lost when points are projected to the estimated cross section.
3. The target object can be defined via simple parameters such as length, height, width, and thickness.

In many object scanning situations, it is not unreasonable to assume the cross-section of the object is known. Forcing the point cloud to conform to a uniform shape

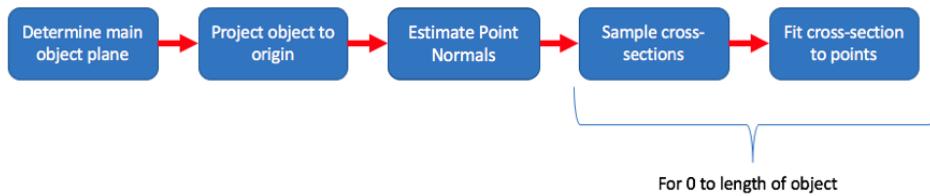


Figure 6.1: A flow chart of the informed shape estimation algorithm.

allows for avoidance of heavy amounts of noise.

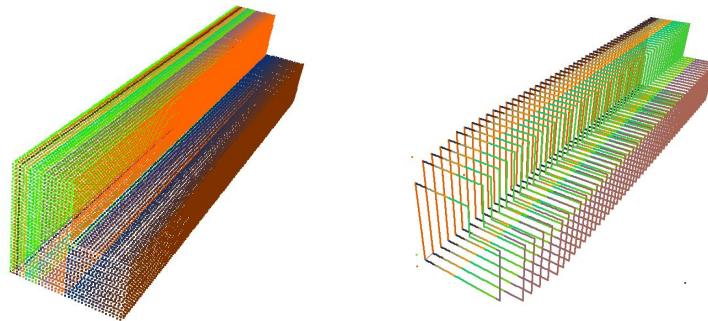


Figure 6.2: Demonstration of the Shape Estimation algorithm on a simulated beam of known dimensions 10x10x5" (left). The iterator converges to dimensions 10.67x10.67x4.87".

## Bibliography

- [1] S. Siebert and J. Teizer, “Mobile 3d mapping for surveying earth-work projects using an unmanned aerial vehicle (uav) system,” *Automation in Construction*, vol. 41, pp. 1–14, 2014. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0926580514000193>
- [2] L. Barazzetti, F. Banfi, R. Brumana, G. Gusmeroli, D. Oreni, M. Previtali, F. Roncoroni, and G. Schiantarelli, “Bim from laser clouds and finite element analysis: Combining structural analysis and geometric complexity,” *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XL, no. 5/W4, 2015.
- [3] G. Castellazzi, A. M. D’Altri, G. Bitelli, I. Selvaggi, and A. Lambertini, “From laser scanning to finite element analysis of complex buildings by using a semi-automatic procedure,” *Sensors*, vol. 15, 2015.
- [4] U. Clarenz, M. Rumpf, and A. Telea, “Finite elements on point based surfaces,” in *Eurographics Symposium on Point-Based Graphics*, Conference Paper.
- [5] H. Hu, H. Hu, T. M. Fernandez-Steger, M. Dong, and R. Azzam, “Numerical modeling of lidar-based geological model for landslide analysis,” *Automation in construction*, vol. 24, pp. 184–193.
- [6] J. Dan and W. Lancheng, “Direct generation of die surfaces from measured data points based on springback compensation,” *The International Journal of Advanced Manufacturing Technology*, vol. 31, no. 5, pp. 574–579, 2006. [Online]. Available: <https://doi.org/10.1007/s00170-005-0225-4>
- [7] M. Smith, J. Carrivick, and D. Quincey, “Structure from motion photogrammetry in physical geography,” *Progress in Physical Geography*, vol. 40, no. 2, pp. 247–275, 2015.

- [8] M. Westoby, J. Brasington, N. Glasser, M. Hambrey, and J. Reynolds, “structure-from-motion’ photogrammetry: A low-cost, effective tool for geoscience applications,” *Geomorphology*, vol. 179, pp. 300 – 314, 2012. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0169555X12004217>
- [9] G. Vosselman and H.-G. Maas, “1.1 basic measurement principles of laser scanners,” 2010. [Online]. Available: <https://app.knovel.com/hotlink/khtml/id:kt00BRXVO4/airborne-terrestrial/basic-measurement-principles>
- [10] V. LiDAR. Hdl-32e. [Online]. Available: <http://velodynelidar.com/hdl-32e.html>
- [11] S. Y. Chen, “Kalman filter for robot vision: A survey,” *IEEE Transactions on Industrial Electronics*, vol. 59, no. 11, pp. 4409–4420, Nov 2012.
- [12] ——, “Kalman filter for robot vision: A survey,” *IEEE Transactions on Industrial Electronics*, vol. 59, no. 11, pp. 4409–4420, Nov 2012.
- [13] M. A. Fischler and R. C. Bolles, “Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography,” *Commun. ACM*, vol. 24, no. 6, pp. 381–395, Jun. 1981. [Online]. Available: <http://doi.acm.org/10.1145/358669.358692>
- [14] R. Raguram, J.-M. Frahm, and M. Pollefeys, “A comparative analysis of ransac techniques leading to adaptive real-time random sample consensus,” in *Computer Vision – ECCV 2008*, D. Forsyth, P. Torr, and A. Zisserman, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 500–513.
- [15] Y. Zhong, “Shape signatures: A shape descriptor for 3d object recognition,” in *IEEE 12th International Conference on Computer Vision Workshops*, IEEE, Ed., vol. 12, Conference Paper.
- [16] P. Biasuttia, J.-F. Aujola, M. Bredif, and A. Bugeaub, “Disocclusion of 3d lidar point clouds using range images,” *ISPRS Annals of the Photogrammetry*, vol. e IV-1/W1, no. 75, 2017.
- [17] Choudhury, D. R. Parhi\*, and Sasanka, “Analysis of smart crack detection methodologies in various structures,” *Journal of Engineering and Technology Research*, vol. 3, no. 5, pp. 139–147, 2011.

- [18] *Modelling Background.* London: Springer London, 2006, pp. 11–27.
- [19] N. P. Weatherill and O. Hassan, “Efficient three-dimensional delaunay triangulation with automatic point creation and imposed boundary constraints,” *International Journal for Numerical Methods in Engineering*, vol. 37, no. 12, pp. 2005–2039, 1994. [Online]. Available: <http://doi.org/10.1002/nme.1620371203>
- [20] D. J. Mavriplis, “An advancing front delaunay triangulation algorithm designed for robustness,” *Journal of Computational Physics*, vol. 117, no. 1, pp. 90–101, 1995. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0021999185710479>
- [21] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin, “The ball-pivoting algorithm for surface reconstruction,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 5, no. 4, pp. 349–359, Oct 1999.
- [22] J. Digne, “An implementation and parallelization of the scale-space meshing algorithm,” *Image Processing On Line*, 2015.
- [23] P. Liepa, “Filling holes in meshes,” in *Proceedings of the 2003 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, ser. SGP ’03. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2003, pp. 200–205. [Online]. Available: <http://dl.acm.org/citation.cfm?id=882370.882397>
- [24] M. Botsch and L. Kobbelt, “A remeshing approach to multiresolution modeling,” in *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, ser. SGP ’04. New York, NY, USA: ACM, 2004, pp. 185–192. [Online]. Available: <http://doi.acm.org/10.1145/1057432.1057457>
- [25] Z. Ji, L. Liu, and G. Wang, “Non-iterative global mesh smoothing with feature preservation,” 01 2009.
- [26] S. Loriot, J. Tournois, and I. O. Yaz, “Polygon mesh processing,” in *CGAL User and Reference Manual*, 4.12 ed. CGAL Editorial Board, 2018. [Online]. Available: <https://doc.cgal.org/4.12/Manual/packages.htmlPkgPolygonMeshProcessingSummary>
- [27] S. Lloyd, “Least squares quantization in pcm,” *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

- [] L. Chen and J. chao Xu, “Optimal delaunay triangulations,” *Journal of Computational Mathematics*, vol. 22, no. 2, p. 10, 2004.
- [28] J. Tournois, R. Srinivasan, and P. Alliez, “Perturbing slivers in 3d delaunay meshes,” in *18th International Meshing Roundtable*, HAL, Ed. HAL, Conference Paper.
- [29] S.-W. CHENG, T. K. DEY, H. EDELSBRUNNER, M. A. FACELLO, and S.-H. TENG, “Sliver exudation,” *Journal of the ACM*, vol. 47, no. 5, p. 21, 2000.