Automated Conversion of 3D Point Clouds to Finite Element Analysis Compatible Geometrical 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

List of Tables

List of Figures

Abstract

Automated Conversion of 3D Point Clouds to Finite Element Analysis Compatible Geometrical Models

Matthew S. Brown

Antonios Kontsos, Ph. D.

1. Introduction

1.1 Motivation

Currently, methods to collect and convert point cloud data into usable models (CAD, finite element, etc.) are few and far in between. This is because the market for unmanned vehicles and automated systems has not become mainstream – until now. 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.

The common usages for this equipment is for automated awareness and unmanned navigation, but there is much more that can be done with the data collected. Setting a precident for a way to convert point clouds collected by UAVs to solid models would be enormously helpful.

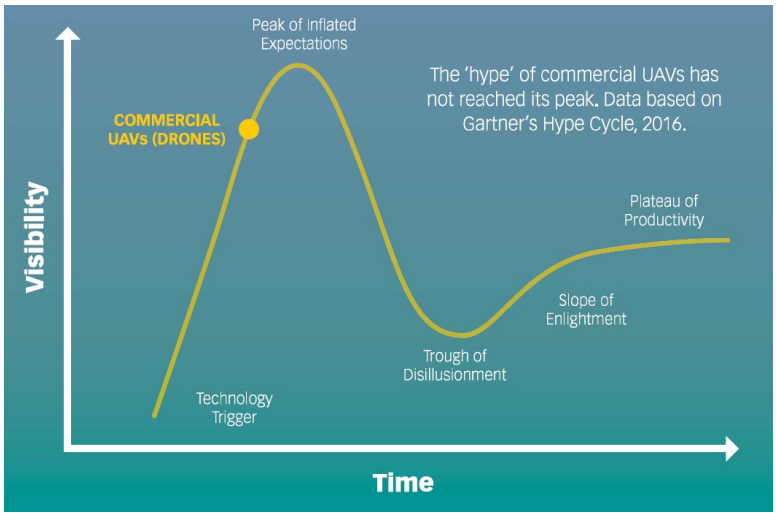


Figure 1.1: According a Gartner hype cycle analysis from 2016, UAVs have not reached the peak of expections [1].

1.2 Problem Statement and Approach

Currently, there are no 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 usable CAD models. The process involves five main steps: Data collection, data pre-processing, point cloud segmentation, initial meshing, and mesh optimization.

The data collection phase encompasses the actual scanning of the object of interest, wether 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 is an analysis of, for each step, what methods work, what methods do not, and what descriptors of the datasets can be used as parameters to measure the success of the methods. The proposed steps in the algorithm are applied to three sets of point cloud data with different properties.

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 except 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: An L-beam with dimensions 5x5x1/4".

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 [2].

1.3.2 Building Informational Modeling

[3]

1.3.3 Aerial Scanning for GPS Overlay

1.3.4 Semi-Automated Point Cloud to FEA modeling

[4] [5] [6] [7]

1.4 Thesis Outline

[[I know there's supposed to be an outline of the structure, but this feels like a clunkier version of the table of contents]]

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 implicit methods where the collection tool does not return direct xyz point data, such as stereogrammetry and structure from motion, to explicit methods where the direct return is a 3-dimensional position output, such as LiDAR, laser scanning and ultrasonic sensing. Each collection technique has its own set of parameters, accuracy ratings, and speed of collection / calculation.

2.1.1 Implicit 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 [8].

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.

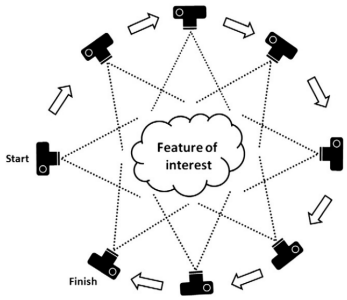


Figure 2.1: Structure from Motion requires collecting a series of overlapping images to generate point clouds [9].

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 [9].

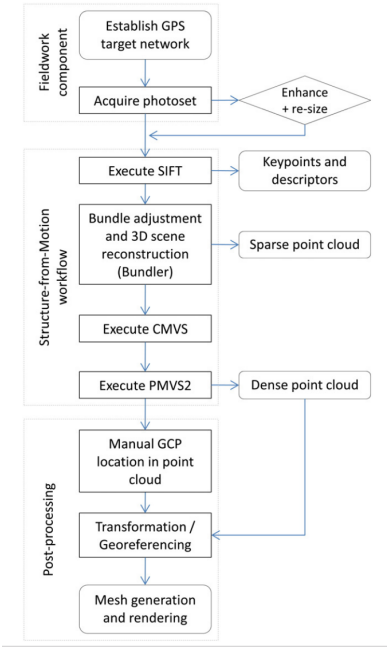


Figure 2.2: A broad overview of the SfM pipeline [9].

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 is.

Figure 2.3 shows the entire Structure from Motion process from start to finish.

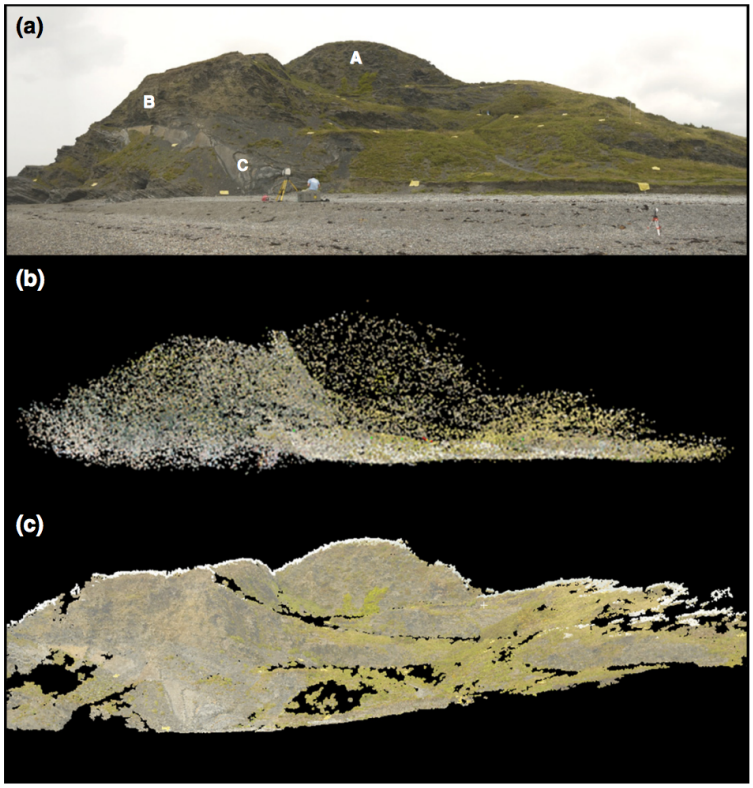


Figure 2.3: Structure from Motion point cloud based on a series of panoramic images take from the ground. a) panorama of the scene b) sparse point cloud reconstruction c) image overlay [9].

2.1.2 Explicit Methods: LiDAR, Laser Scanning, and Ultrasonic Sensing

Explicit collection methods are those that directly return a distance value from a sensor. If the scanner itself is non-static, as is the case for laser-scanners and LiDAR

devices, the sensor also returns some timestamped orientation parameters to provide a local reference frame.

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 θ and a distance magnitude.

Figure 2.4: Data collected from a single rotation of the [[insert sensor name]] LiDAR device.

High end LiDAR devices, such as the HDL-32E used to collect the data in this thesis, have a series of lasers distributed through a range of angles. This allows the LiDAR to provide a much more comprehensive pointset of it's 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 return the timestamp of the data, the rotation angle of the sensor, 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 $\pm 2cm$ or lower.

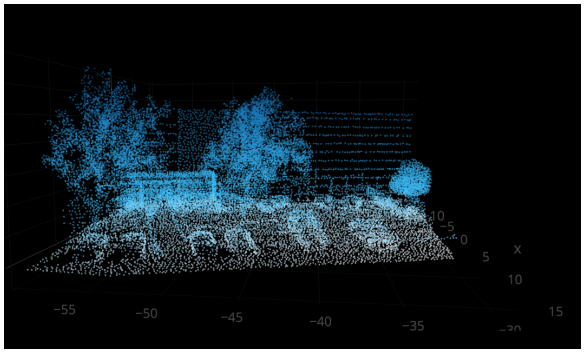


Figure 2.5: Data collected from an HDL-32E LiDAR device [10].

Laser Scanning

[[add]]

Ultrasonic Sensing

[[add]]

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 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 RANSAC and Intrinsic Shape Signatures (defined in the next section).

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 it's 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 $r_{density}$:

$$w_i = \frac{1}{||p_j \mid |p_j - p_i| < r_{density}||} \tag{2.1}$$

   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}:

$$cov(p_i) = \sum |p_j - p_i| < r_{frame} \frac{w_j (p_j - p_i)(p_j - p_i)^T}{\sum |p_j - p_i| < r_{frame} w_j} \tag{2.2}$$

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 [13].

3D Shape Feature Extraction The goal of the extraction is to create a view invariant feature vector providing us with 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 pi is created and divided into 66 distinct partitions in angular space (θ, ψ). A distinctive

feature vector with 66 values is then computed by summing the radial distances ρ_i in each bin [13].

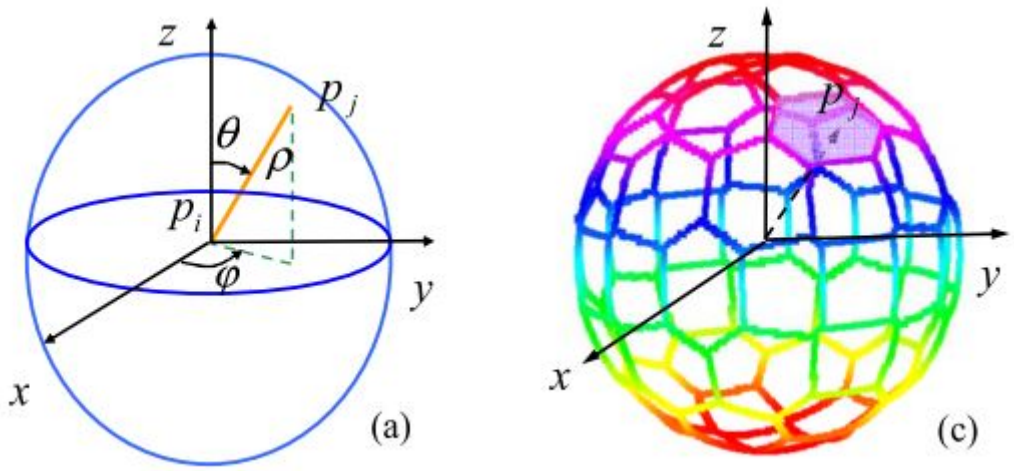


Figure 2.6: Feature vector calculation via spherical bin decomposition [13].

2.2.2 Filtering

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

$$P_x = p_i \mid \sum_{j=1}^{n} |p_j - p_i| \leq (r_{density} + d) \geq k \tag{2.3}$$

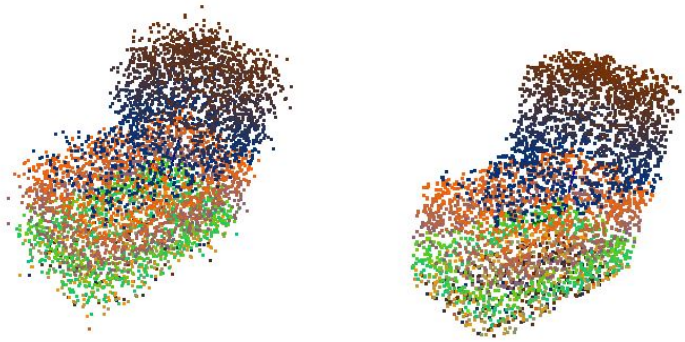


Figure 2.7: The effects of noise filtering on a simulated L block with 10% induced noise.

2.2.3 Down-Sampling

Down-sampling is the process of fixing a point clouds mean density to a voxel of size n. This is done by iterating the voxel throughout the clouds 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 2.4

$$p_{new}(i) = \frac{\sum_{p(i)\in V}^{N} p(i)}{N} \tag{2.4}$$

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 os this equation is the average position of the points inside the voxel.

Figure 2.8 shows the results of a simulated shape of initial point-to-point average

distance of 0.05 inches downsampled with a voxel size of 0.25 in^3.

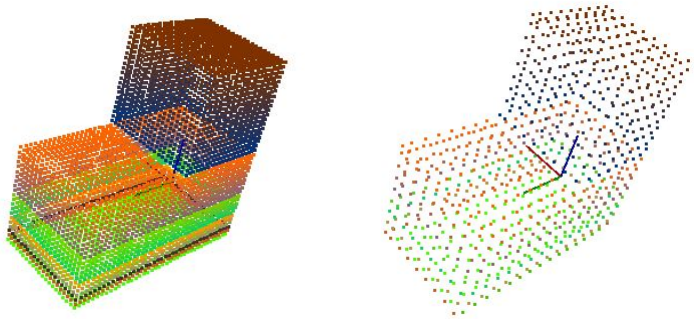


Figure 2.8: A simulated L block point cloud down-sampled with voxel size = 0.25 in^3.

Downsampling is crucial for processing poinclouds. The obvious benefit is decreased computation time, but it also allows for us to control 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.

2.2.4 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 is occlusion. In the field, it is nearly impossible to fully avoid occluded datasets when scanning an object via LiDAR equipped UAVs.

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

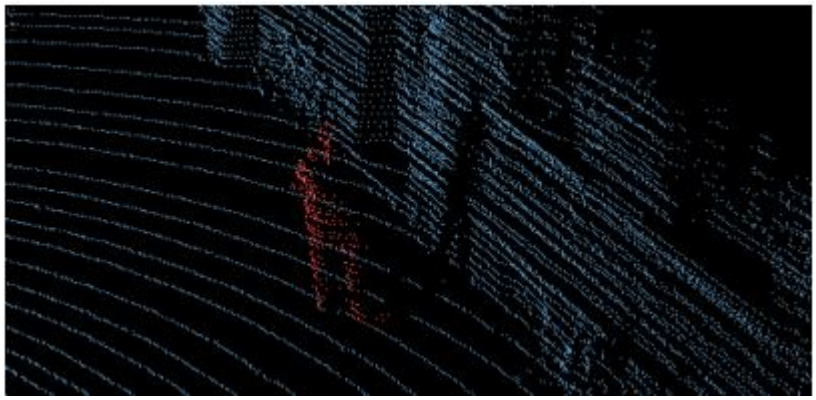


Figure 2.9: Section of a building occluded by an object in the foreground [14].

Range Segementation

In urban scanning situations, it is nearly impossible to avoid scan noise. This can come in the form of humans passing by, parked cars, street lights, other buildings, trees, bushes, and a slew of other objects that may come between the scanner and the desired object. 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 θ and ϕ, 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.

$$C_s^i = \frac{\sum_{b \in C_s^i} b \times h_s(b)}{\sum_{b \in C_s^i} h_s(b)} \tag{2.5}$$

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

$$d(C_s^i, C_r^j) = |C_s^i - C_r^j| \tag{2.6}$$

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 neighbors. This approach requires structured data that is also time-stamped. In the equation below, u represents the (ϕ, θ) coordinates of a point the merged dataset, and Ω represents the full range image of the merged dataset, and η represents the orthogonal projection of each pixel in the range image. The aim is to solve the following disocclusion problem.

$$\begin{cases} \frac{\partial u}{\partial t} - \Delta u = 0 \in \Omega \times (0, T) \\ u(0, x) = u_0(0) \in \Omega \end{cases} \tag{2.7}$$

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 [14].

2.3 Machine Learning for Object Recognition and Segmentation

2.3.1 Supervised Methods — Neural Networks

Definitions It is difficult to define how a Convolutional Neural Network works without first defining a convolution. To detect distinctive features in a dataset from a machines 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 equaling 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.

$$C_{new} = \sum_{i}^{N} w(i)x_i \tag{2.8}$$

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.

$$C_{new} = \frac{\sum_i^N x_i}{N} \tag{2.9}$$

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:

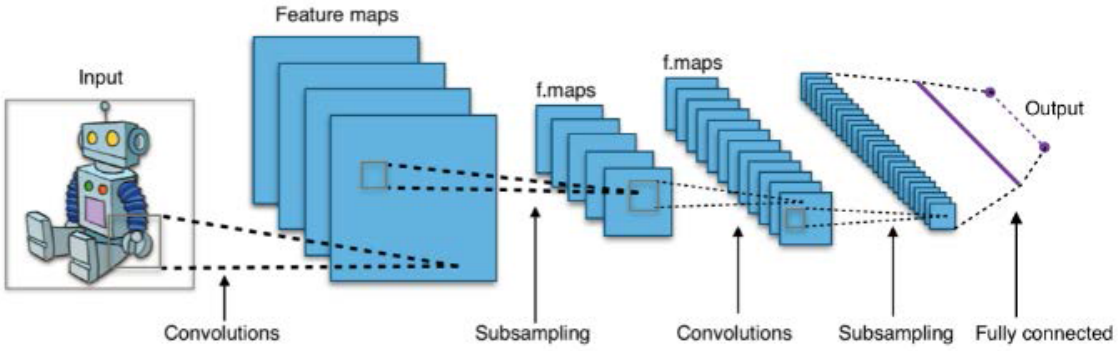


Figure 2.10: Map of a convolutional neural network with two hidden layers [ref]

In the case of point cloud processing, the input is a raw xyz set of some size n x 3. The first step in the network is a series of convolutions of the dataset. Each kernel in the layer contains m x m x 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 draw 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.

$$\begin{bmatrix} C_1 \\ \vdots \\ C_n \end{bmatrix} = \begin{bmatrix} w_{1,1} & \dots & w_{1,m+1} \\ \vdots & \ddots & \dots \\ w_{n,1} & \dots & w_{n,m+1} \end{bmatrix} \begin{bmatrix} f_1 \\ \vdots \\ f_n \\ 1 \end{bmatrix} \tag{2.10}$$

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 ith 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:

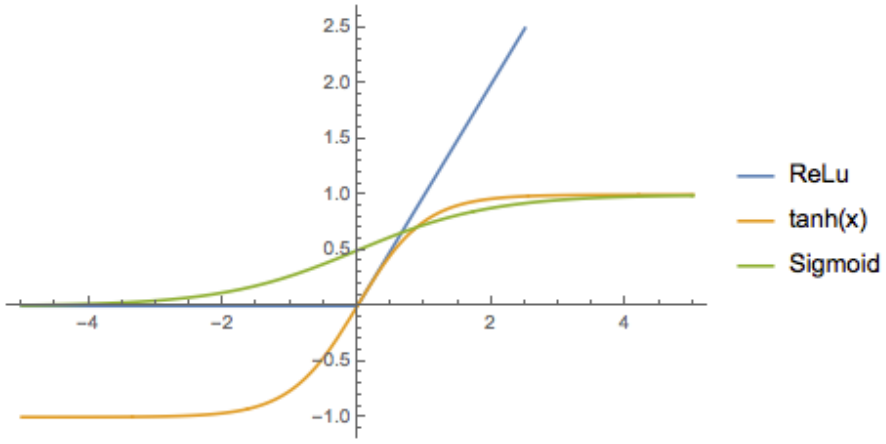


Figure 2.11: Visualization of commonly used non-linear activation functions

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:

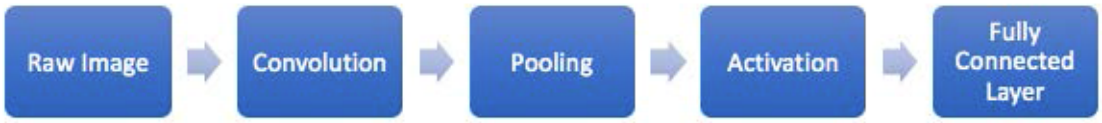


Figure 2.12: Block diagram of a CNN′s skeleton structure

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 systems classification and the true classification of each image is used to update the weights iteratively until the machines 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:

$$x_{k+1} = x_k - \gamma \nabla F(x_k) \tag{2.11}$$

$F(x)$ is described as the residual function. It is the diference between the result of the cost function $C(x)$, and the current x variable values.

$$F(x) = \begin{bmatrix} C_1 \\ \vdots \\ C_n \end{bmatrix} - \begin{bmatrix} w_{1,1} & \dots & w_{1,m+1} \\ \vdots & \ddots & \dots \\ w_{n,1} & \dots & w_{n,m+1} \end{bmatrix} \begin{bmatrix} f_1 \\ \vdots \\ f_n \\ 1 \end{bmatrix} \tag{2.12}$$

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 [15]. 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.

2.3.2 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 σ between centroid position c_n and c_{n+1}.

$$argmin_S \sum i = 1^k \sum x \in S_i |x - c_i|^2 \tag{2.13}$$

K-means clustering is arguably the most well known clustering method, and is useful for a massive variety of nave 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 it's 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.

$$c_k = \frac{\sum xw_k(x)^m x}{\sum xw_k(x)^m} \tag{2.14}$$

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:

$$w_{ij} = \frac{1}{\sum_{k=1}^{c} (\frac{|x_i - c_j|}{|x_i - c_k|})^{\frac{2}{m-1}}} \tag{2.15}$$

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 it's exit criteria. This allows for all points in the dataset to continually effect the location of every cluster centroid. FCM finds it's 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 it's 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.

Aggomerative 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.

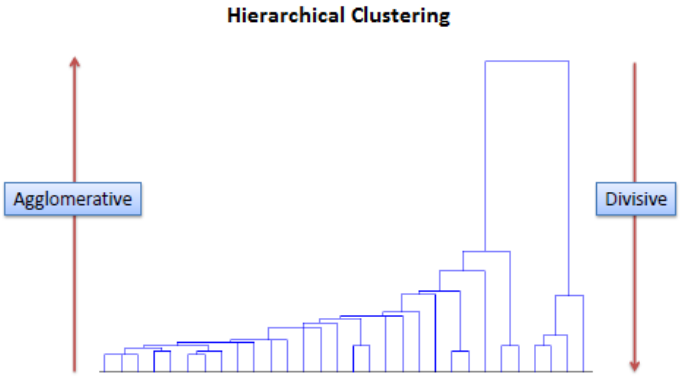


Figure 2.13: Comparison of divisive and agglomerative hierarchical clustering

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 it's 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 it's unsupervised brethren, and does not require an expected bin-size.

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.

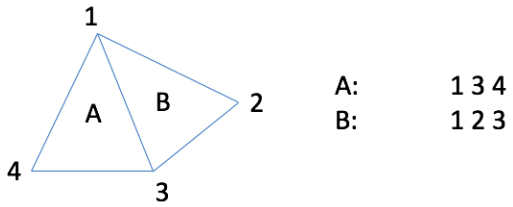


Figure 2.14: Visualization of necessary components in a surface mesh.

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 — and the target application for this thesis — is finite element analysis, a numerical method for solving structural analysis problems through discretization of individual 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).

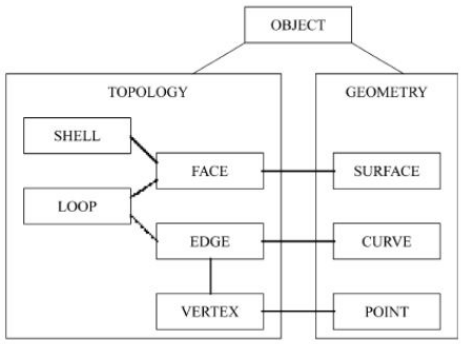


Figure 2.15: An overview of the components involved in a boundary representation of a solid object [16].

Non-Uniform Rational Basis Spline (NURBS)

NURBS curves are a method of interpolating a discrete set of vertex points into a continous 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 ast the following:

$$f(u) = \frac{\sum_{i=0}^{K-1} w_i B_{i,n}(u) P_i}{\sum_{i=0}^{K-1} w_i B_{i,n}(u)} \tag{2.16}$$

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 2.17 to the piecewise B-Spline equation shown in Equation 2.16.

$$R_i(u) = \frac{w_i B_{i,n}(u)}{\sum_{i=0}^{K-1} w_i B_{i,n}(u)} \tag{2.17}$$

Leaving us with the following relationship:

$$f(u) = \sum_{i=0}^{K-1} R_{i,d}(u) P_i \tag{2.18}$$

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

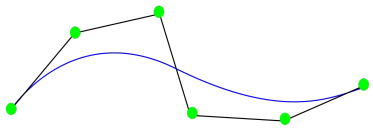


Figure 2.16: A 2-dimensional example of a nurbs spline. Green points represent the control points, P, black lines represent their connectivity, and the blue line represents one possible nurbs spline.

2.4.1 CAD to Finite Element Conversion Compatibility

Most commercial CAD software subscribes to the boundary representation model to represent solid modeling objects. Converting a boundary representation – a list of surface features and piece-wise shape functions – to a solid volume of finite elements that accurately describe the connectivity of the object requires a series of user input parameters. Element type, linear material parameters (typically ASTM material standards), local mesh density, and load and boundary conditions.

The crucial compatibility factor for B-REP solids is the absence of critical regions which cause stress singularities during the analysis. These regions can take the shape of sharp corners, spires, or point loads. They will not cause the mesh to fail, but return disproportionately massive stresses when compared with the stresses in the surrounding mesh.

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, $\{P_k\}$ — $k = 1, \ldots, K$, 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.

$$V_k = \{P_i \mid |p - P_i| < |p - P_j|, \forall j \neq i\} \tag{2.19}$$

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 [17].

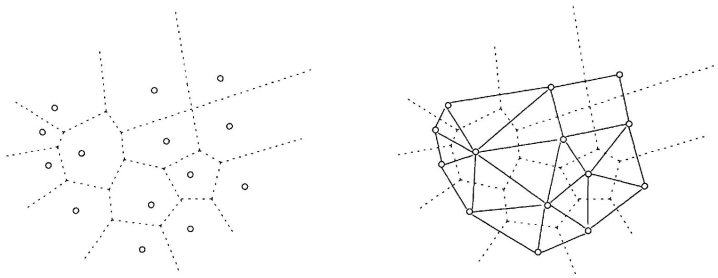


Figure 2.17: Visual represention of a 2-dimensional delaunay triangulation process. The figure on the left shows the voronoi diagram of the vertex set, and the image on the right shows the triangulation result [17].

Advancing Front / Marching Triangles

The Advancing Front method is common in computer graphics software especially in procedurally generated games because of its 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 triangluation between the boundary vertices and unreferenced points in space. Each new triangulation object is created by adding a single unreferenced point at a time [18].

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 pk 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 ∈ P_idx
                p' = MCC(p, P_idx, r)
                store p' in P_new_idx
                p'.origin = p.origin
        if ( idx > 0 )
                remove P_idx
        idx = new_idx
```

Using the Ball Pivoting Algorithm [19], 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 thoughout 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 minimual noise reduction in the process [20].

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.

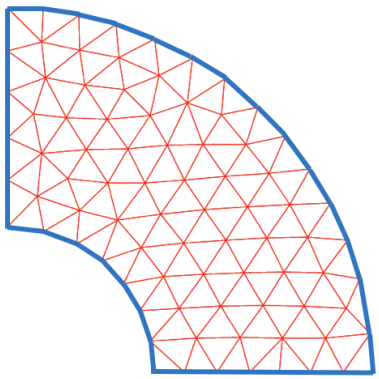


Figure 2.18: A visualization of the difference between an edge and a half edge. Half edges are outlined in blue.

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.

[[specifics]]

[21].

Fairing The fairing algorithm smooths the newly refined surface into a continous shape based on a minimization of a linear bi-Laplacian system with boundary constraints [22]. [[specifics]]

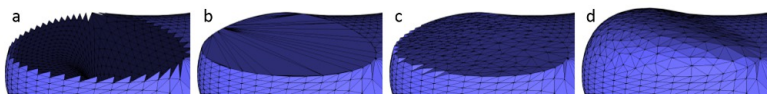


Figure 2.19: Results of the main steps of the algorithm. From left to right: (a) the hole, (b) the hole after its triangulation, (c) after triangulation and refinement, (d) after triangulation, refinement and fairing [23].

2.4.2 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 volumetric mesh.

Criteria for Mesh Quality and Failure Criteria for Volumetric Conversion

Failure Criteria The first criterion for surface mesh compatibility with volumetric conversion is water-tightness. Meaning, there are no gaps, holes, or unbounded edges 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.

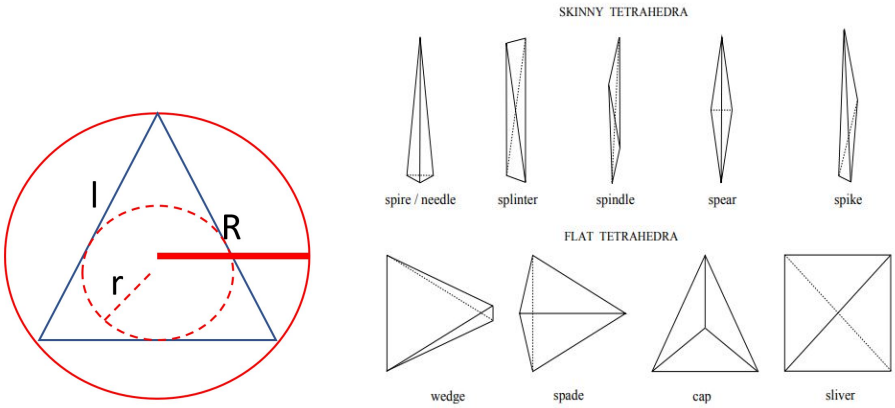


Figure 2.20: Definitions for triangulation quality measures, and typical shapes with poor quality. R = circumsphere radius, r = inscribed sphere radius, l = edge length.

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 volumetric conversion, typically looking like those shown in figure 2.20.

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	$Q_4 = \frac{l_{min}}{l_{max}}$
Volume ratio	$Q_5 = \frac{V}{l_{max}^3}$

Voronoi Relaxation / Lloyd's Algorithm

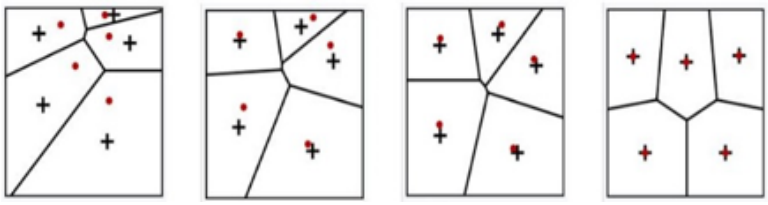


Figure 2.21: From right to left: Voronoi relaxation over 1 iteration, 5 iterations, 10 iterations, and 15 iterations

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 shapes topology significant due to its heavy smoothing capabilities [24].

Optimal Delaunay Triangulation

Delaunay triangulation, as described in Section 2.4.1, optimizes the connectivity for a finite point set through voronoi propagation. To calculate the optimal triangula-

tion of the pointset defined by a discrete set of vertices, the vertices are unfixed from their intial 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 Ω and $1 \leq p \leq \inf$, and triangulation $\tau^* \in P_N$ the following criteria is used:

$$Q(\tau^*, f, p) = \inf_{\tau \in P_N}(\tau, f, p) \tag{2.20}$$

[[specifics]]

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 are necessary to oust any slivers remaining in the triangulation. Slivers are defined as any triangulation with an angle less than α, 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 [25].

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 [26]. [[specifics]]

3. Objective, Hypothesis, and Technical Approach

3.1 Research Objective

The objective of this thesis is to develop and validate a software pipeline capable of segmenting point cloud data of some expected density ρ and within some accuracy tolerance σ into clusters of meaningful shapes, and convert the resulting clusters into surface meshes valid for conversion to simply connected volumes capable of being inputs to Finite Element simulations.

3.2 Hypothesis Statement

The 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 volumetrically adaptable surface meshes.

The second hypothesis attempts to nail down an explicit definition of what it means for a mesh to be volumetrically 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 it's real world counterpart, and criteria for a surface mesh to be successful 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 finite element analysis on real objects with reliable results. The following chapters quantify the effect of parameter modification at each step of the algorithm.

While this thesis does not provide vetted parameters for assured volume compatibility, there is significant exploration into what parameters work and what cause a mesh to fail. Interestingly enough, a mesh that is aesthetically pleasing is demonstrated to be a successfully convertible mesh.

3.3 Technical Approach

To fully evaluate the effect of each step in the algorithm and test the robustness of the proposed algorithm, it is applied to a series 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 proof 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 forseeable future. They do, however, serve as a proof of concept for the meshing pipeline as a form of optimal results.

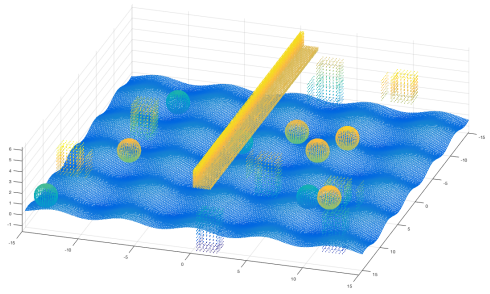


Figure 3.1: Virtual lab point cloud with zero induced noise and occlusion. there are 114,808 points in the cloud.

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, σ, is applied to the mesh. This σ 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 $\sigma = 2cm$.

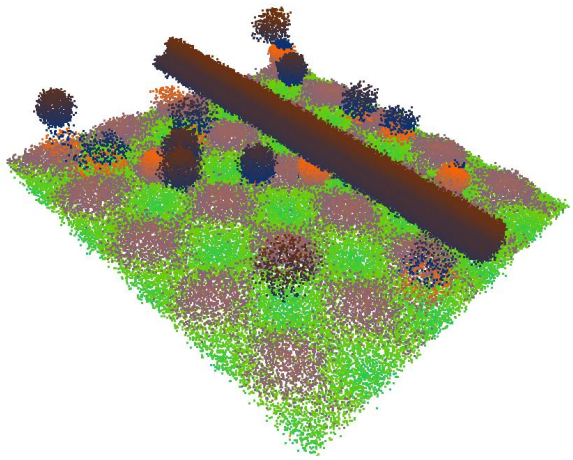


Figure 3.2: Simulated high density lab point cloud with 2 centimeter noise induction.

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° to -30.67°

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 ϕ, its angle about the z-axis, θ, 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 to 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, 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 calculation 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

Figure 3.4: Flow chart of point cloud concatenation process.

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.

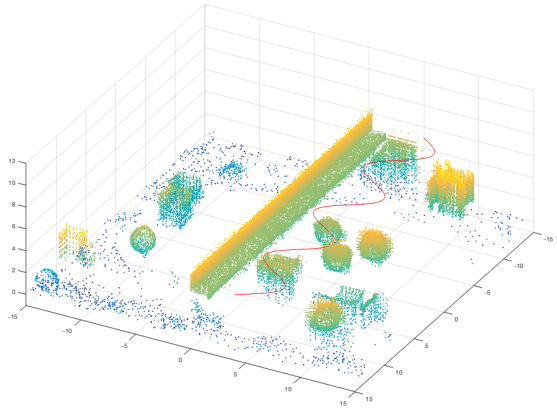


Figure 3.5: Simulated LiDAR scan of the virtual lab point cloud at 100% sampling rate. The red trail shows the collection path of the sensor.

3.3.4 Stereogrammetry and Photogrammetry

Stereogrammetry and Photogrammetry are both highly dependent on the camera quality, speed of motion, and feature density of the image. As these methods depend

on matching descriptive features between images, the resulting cloud is typically far more sparse than what would be obtained with a LiDAR device.

3.3.5 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.

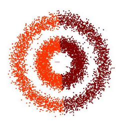
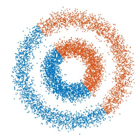
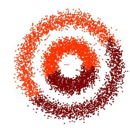
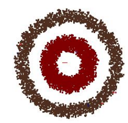
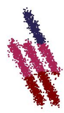
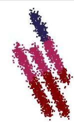
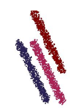
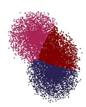
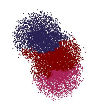
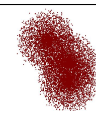
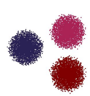
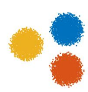
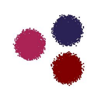
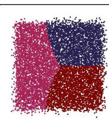
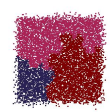
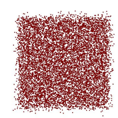
As a result, the methods evaluated in this thesis are reserved purely to those which are classified as unsupervised. Selected based on potential success, popularity, and ease of implementation, the following segmentation methods are analyzed in Chapter 4: K-Means Clustering, Fuzzy C-Means Clustering, 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.

Comparison of Methods

Table 3.1 shows a brief comparison of how each of the methods described above act on a series of 2-dimensional control point clouds with various shape characterisics.

Table 3.1: Comparision of unsupervised clustering methods on various simulated point clouds

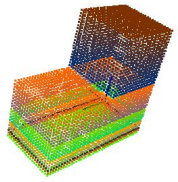
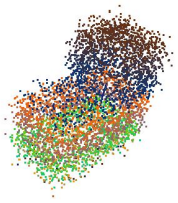
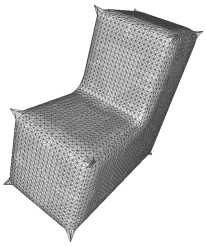
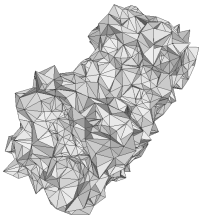
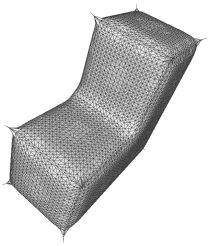
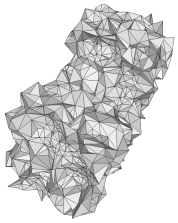
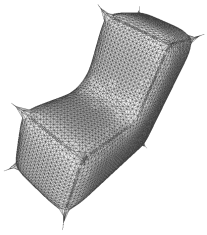
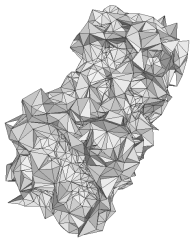
K-means	Fuzzy C-means	Agglomerative	Euclidean
			
			
			
			
			

3.3.6 Initial Mesh Exploration

There are two algorithms for the initial mesh construction intended for evaluation in this thesis: Scale Space Reconstruction and Avancing Front. Advancing front provides no robustness to noise or occlusion in the dataset, but guarantees no overlapping faces, so could be viable with the addition of hole patching and the optimization stage. Scale Space Reconstruction, on the other hand, is designed specifically for handling noisey, variable density datasets. It does not prevent overlapping elements, so removal of these elements must be added to the pipeline for this case. It should also be noted that because of Scale Space Reconstruction's casting steps (described

in Section 2.4), it is far more computationally expensive than the Advancing Front method. Table 3.2 provides some visiualization of the effect of casting a mesh to varying scales.

Table 3.2: Effect of increasing scale in a scale space reconstruction

Input Clouds		
Scale Space 3		
Scale Space 5		
Scale Space 10		

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

3.3.7 Optimization Steps

The goal of the optimization process is to remedy or remove any elements in the mesh that violate the criteria for volumetric conversion compatibility. These criteria are defined in Section 2.4.2. 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 2.4.2.

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 2.4.2, 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.

3.3.8 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 volumetric 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 it's response to the expected response of the object.

3.3.9 Occlusion Correction

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:

$$C = \sum_{i=0}^{N} p(i) - p_{proj}(i) \tag{3.1}$$

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, τ.

$$p_{proj}(i) = arg_{min} \frac{p(i) \bullet l(j)}{||l(j)||} \in p_{normal}(i) \bullet l_{normal}(j) \geq \tau \tag{3.2}$$

$$x_{k+1} = x_k - \frac{C(x_k)}{C'(x_k)} \tag{3.3}$$

Using a Newton Raphson interative regression, shown in equation 3.3 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 it's 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 thoughout 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.

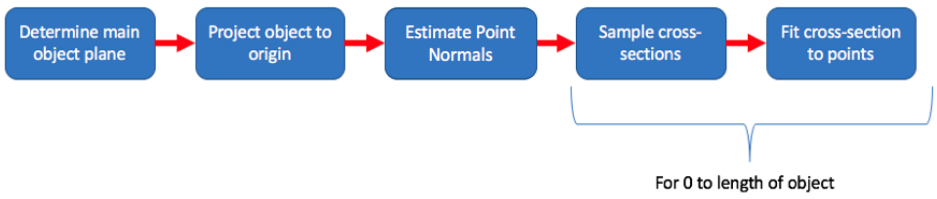


Figure 3.6: A flow chart of the informed shape estimation algorithm.

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 allows for avoidance of heavy amounts of noise.

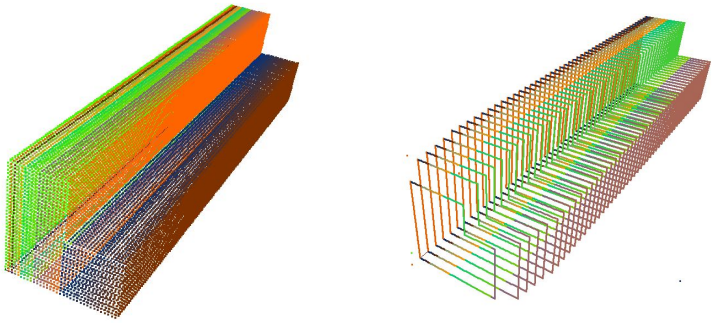


Figure 3.7: Demonstration of the Shape Estimation algorithm on a simulated beam of known dimensions 10x10x5” (left). The iterator converges to dimensions 10.67x10.67x4.87”.

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.

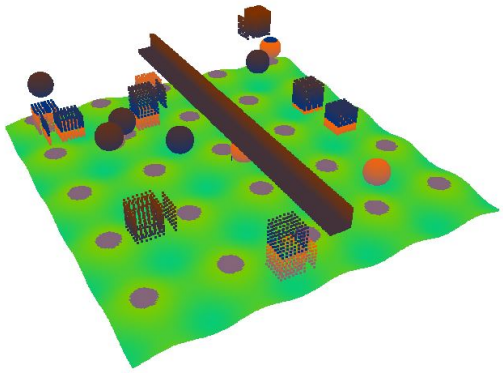


Figure 4.1: Simulated laboratory point cloud data with zero noise induced.

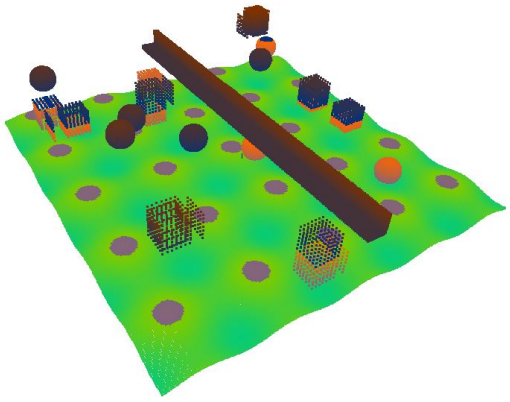


Figure 4.2: Zero noise simulated data after being filtered with voxel size of $0.1in^3$, 15 minimum point neighbors at a distance of 1 standard deviation from the mean point to point distance.

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.

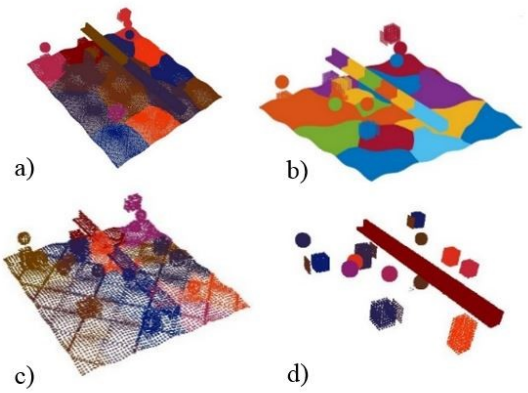


Figure 4.3: Resulting cloud cluster from a) k-means, 17 centroid b) Fuzzy c-means, 17 centroids c) Agglomerative clustering, 17 bins and d) Euclidean clustering with radius of 0.5 in and a maximum cluster size of 50,000 points.

It is clear from the results in Figure 4.3 that Euclidean clustering far outshines it's 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 cloudsize 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.

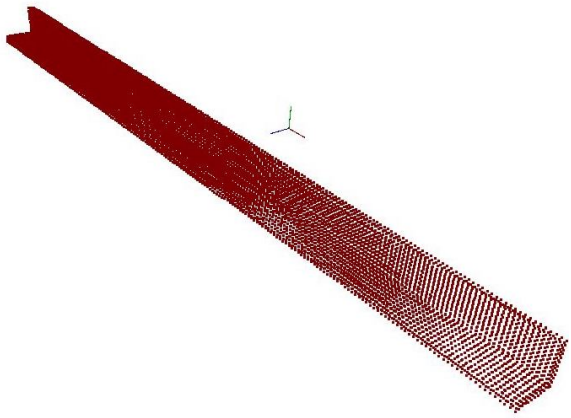


Figure 4.4: Euclidean clustering with radius of 0.5 in, minimum cluster size of 3,000 points and a maximum cluster size of 50,000 points.

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 4.6 shows the effects of Scale Space reconstruction at scale $S = 2$. The predominantly visible effect is the change in topology of the beam. because of the

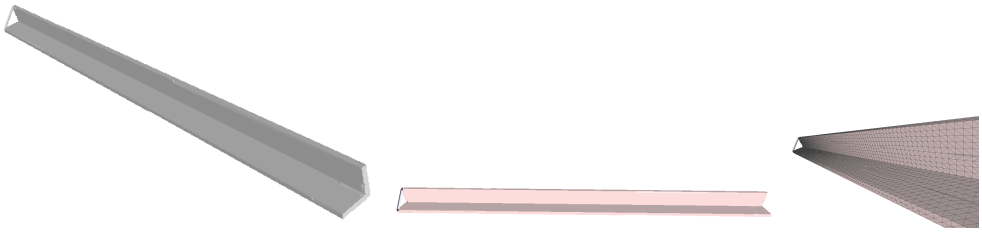


Figure 4.5: Result of initial meshing using the advancing front method.

localized fnunction 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.

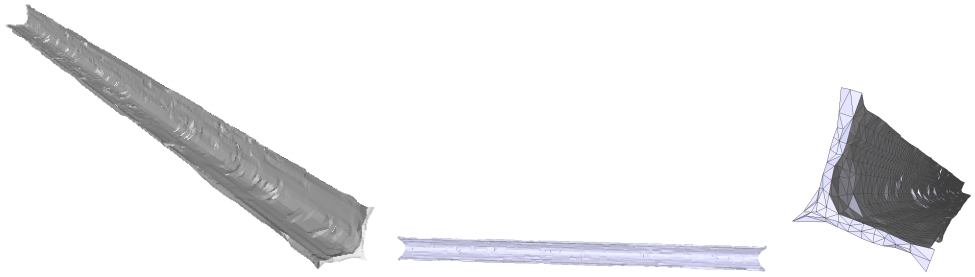


Figure 4.6: Result of initial meshing using the scale space reconstruction method with $S = 2$.

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.

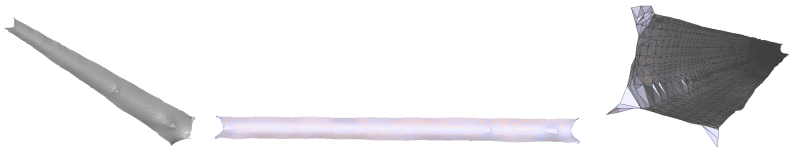


Figure 4.8: Result of initial meshing using the scale space reconstruction method with $S = 15$.

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

The advancing front case is the only case to be volumetrically compatible at first pass. This is due to the input set being completely noiseless. None of the Scale Space

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

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

Quality values shown represent the absolute minima in each category.

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 2.4.2, 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 2.20).

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.

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

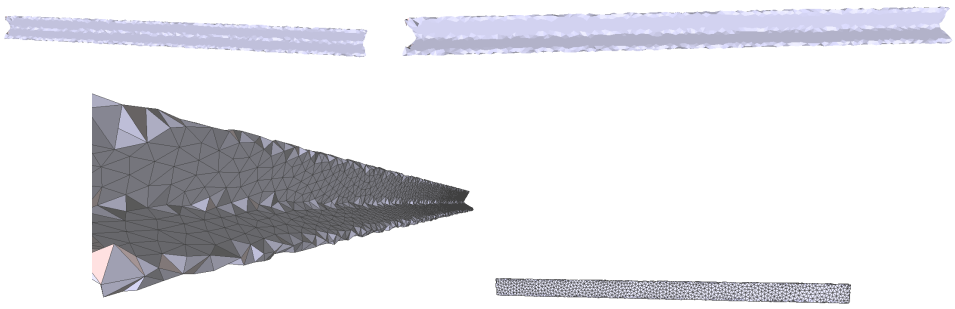


Figure 4.9: Result of Advancing Front mesh after 200 iterations of voronoi relaxation.

process aim to remove these slivers.

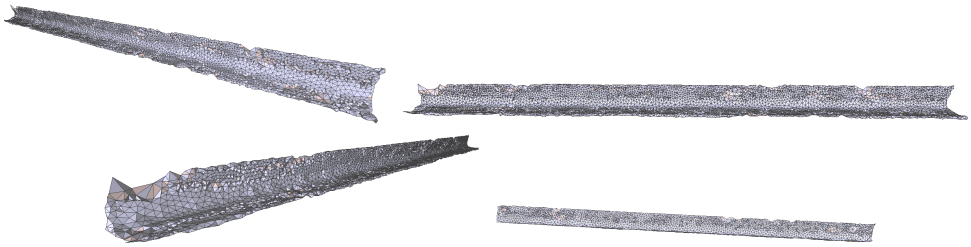


Figure 4.10: Scale space reconstruction $S = 2$ after 200 iterations of voronoi relaxation.

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.

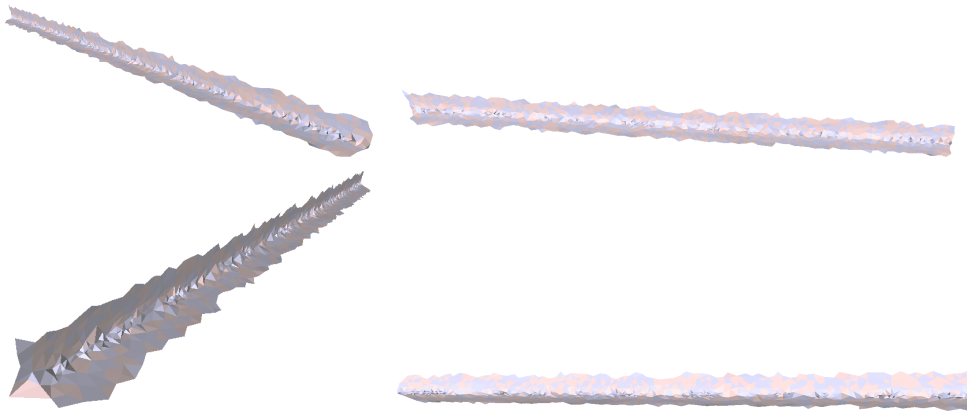


Figure 4.11: Scale space reconstruction $S = 4$ after 200 iterations of voronoi relaxation.

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 2.4.2. 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.

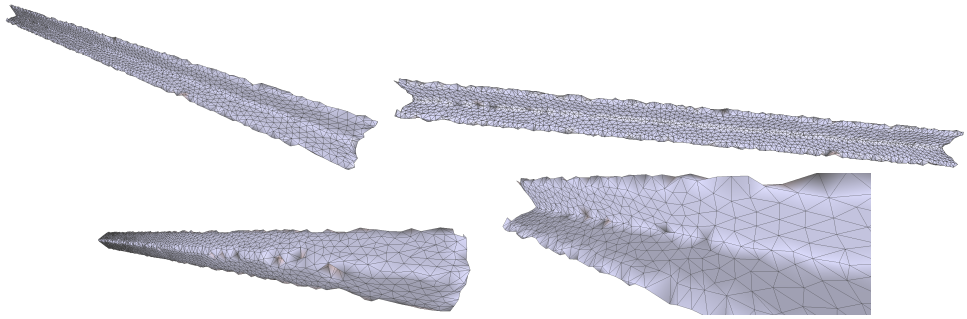


Figure 4.12: Result of Advancing Front mesh after ODT with a 30 second clock cap.

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.

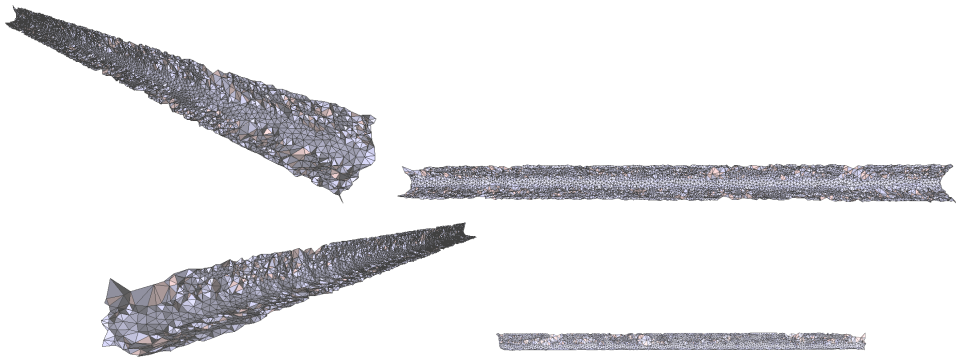


Figure 4.13: Scale space reconstruction $S = 2$ after ODT with a 30 second clock cap.

It is clear at this point that for low thickness objects in zero noise situations, 4^{th} 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.

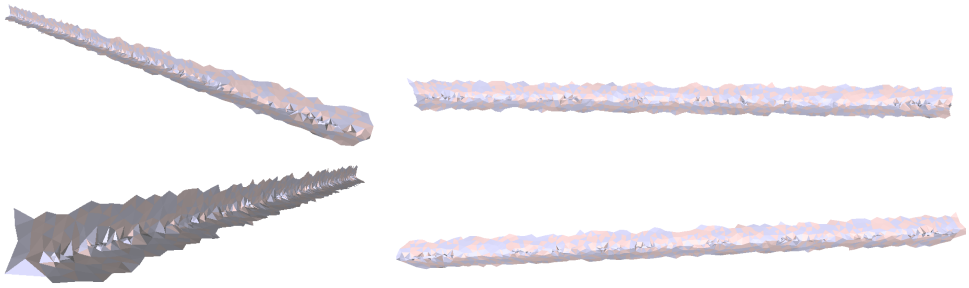


Figure 4.14: Scale space reconstruction $S = 4$ after ODT with a 30 second clock cap.

Effects of Mesh Perturbation

Perturbation serves two purposes: To relax any remaining slivers in the mesh (slivers defined in Section 2.4.2) 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.

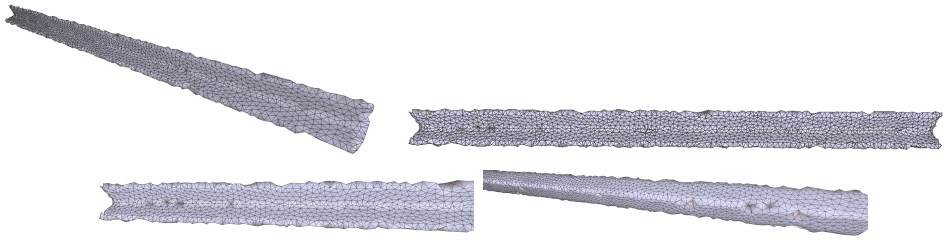


Figure 4.15: Result of Advancing Front mesh after perturbation with a 30 second clock cap.

The next figure is the reconstruction at scale space 2. Again the mesh is severely pock-marked, but far less so than it's 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 it's constraints for quality.

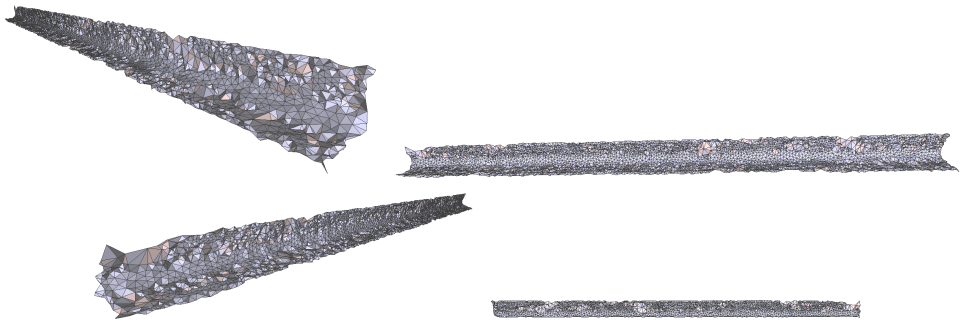


Figure 4.16: Scale space reconstruction $S = 2$ after perturbation with a 30 second clock cap.

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 4.17, 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]].

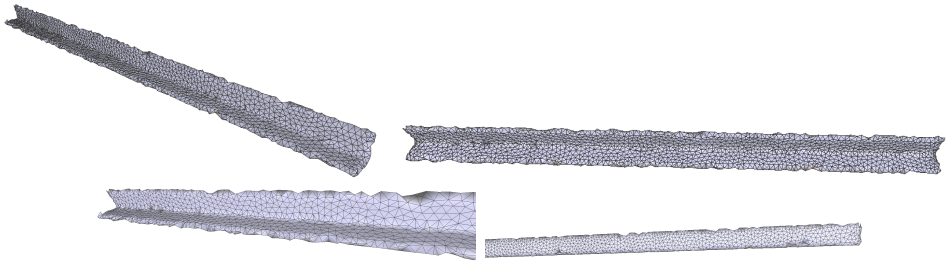


Figure 4.17: Result of Advancing Front mesh after exudation with a 30 second clock cap.

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 4.18 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.

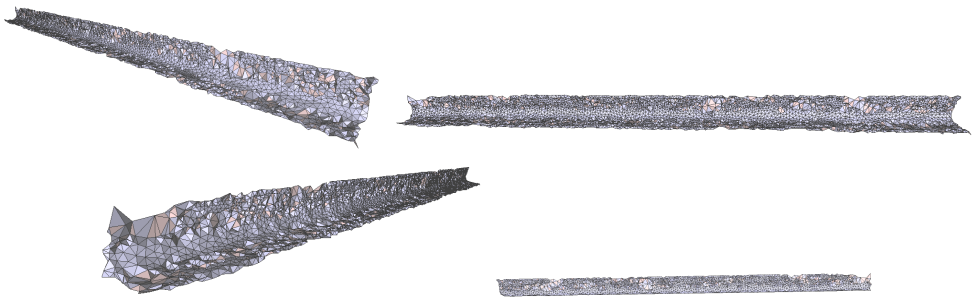


Figure 4.18: Scale space reconstruction $S = 2$ after exudation with a 30 second clock cap.

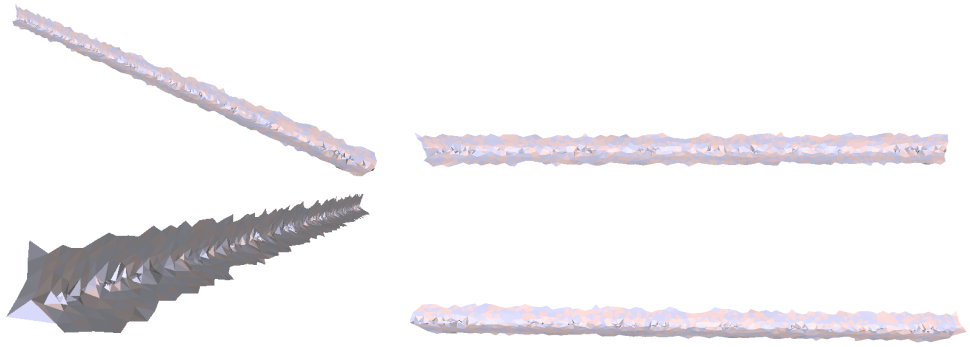


Figure 4.19: Scale space reconstruction $S = 4$ after exudation with a 30 second clock cap.

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	Successful Conversion
Initial Advacing Front	0.0548	0.2916	0.0270	0.0274	1	Yes
Optimized Advancing Front	0.3096	0.1910	1.333	0.1548	1	Yes
Initial Scale Space 2	0.0577	0.0665	0.0082	0.0587	1	No
Optimized Scale Space 2	0.0072	0.1085	0.0035	0.0037	1	No
Initial Scale Space 4	0.0038	0.0185	0.0005	0.0035	1	No
Optimized Scale Space 4	0.0031	0.0786	0.0016	0.0016	1	No

Quality values shown 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 therefor 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 4.20 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 downsampled due to redundant point collection. The algorithm built here does not include the same target point more than once.

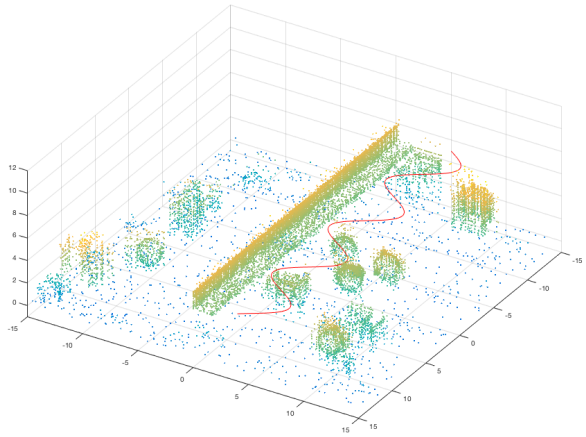


Figure 4.20: Simulated LiDAR scan of the virtual lab point cloud at 20% sampling rate. The red trail shows the collection path of the sensor.

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 **??**.

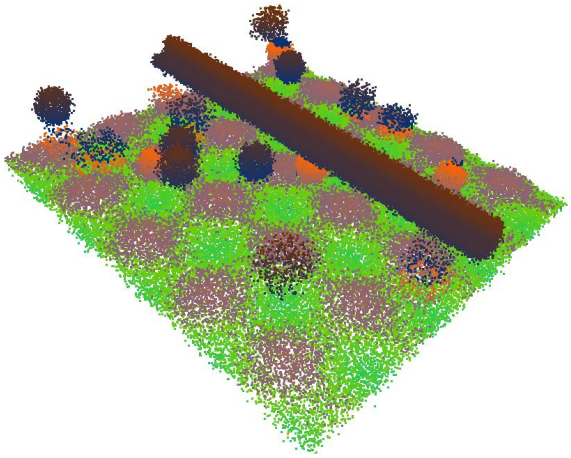


Figure 4.21: Simulated point cloud with 2 *cm* magnitude gaussian noise induced.

As all three axes are introduced to this uncertainty, the total noise in this simulation 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 4.22, the resulting point cloud is significantly cleaner than it's untreated counterpart. Separating the cloud into clearly defined shapes is crucial for the unsupervised clustering algorithms to accurately and meaningfully segment the point cloud.

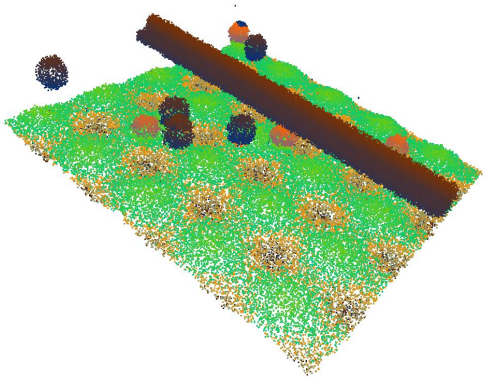


Figure 4.22: 2 cm gaussian noise simulated data after being filtered with voxel size of $0.1in^3$, 15 minimum point neighbors at a distance of 1 standard deviation from the mean point to point distance.

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 it's 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.

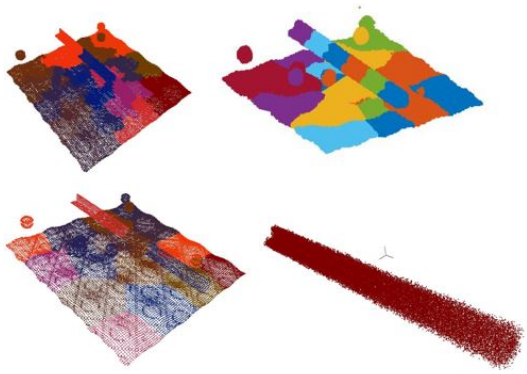


Figure 4.23: Resulting cloud cluster from a) k-means, 17 centroid b) Fuzzy c-means, 17 centroids c) Agglomerative clustering, 17 bins and d) Euclidean clustering with radius of 0.5 cm minimum cluster size of 3,000 and maximum cluster size of 50,000 points.

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 4.24 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 it's ability to compensate for noise. Figure 4.25 shows the initial meshing results using a scale space of 2. Note the uniformity in the mesh, and the singular surface area the

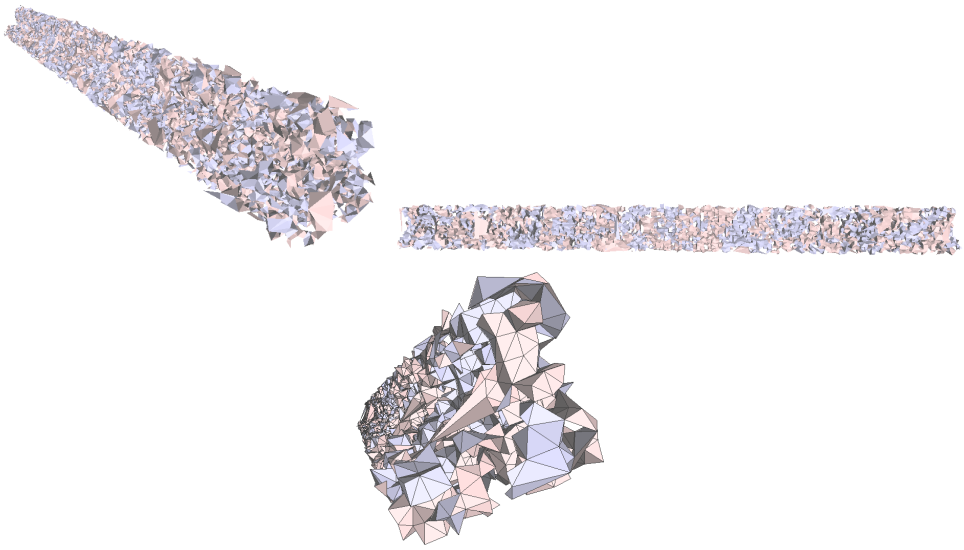


Figure 4.24: Result of initial meshing using the advancing front method.

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

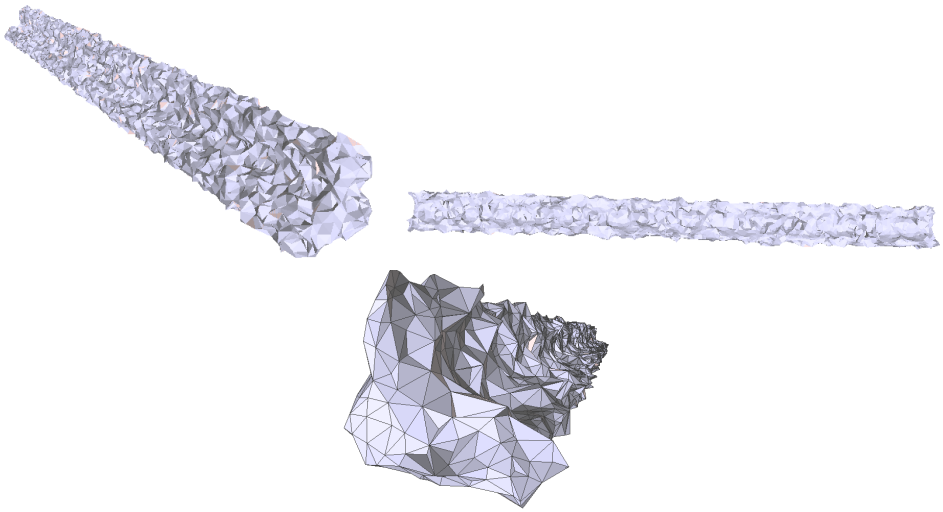


Figure 4.25: Result of initial meshing using the scale space reconstruction method with $S = 2$.

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.

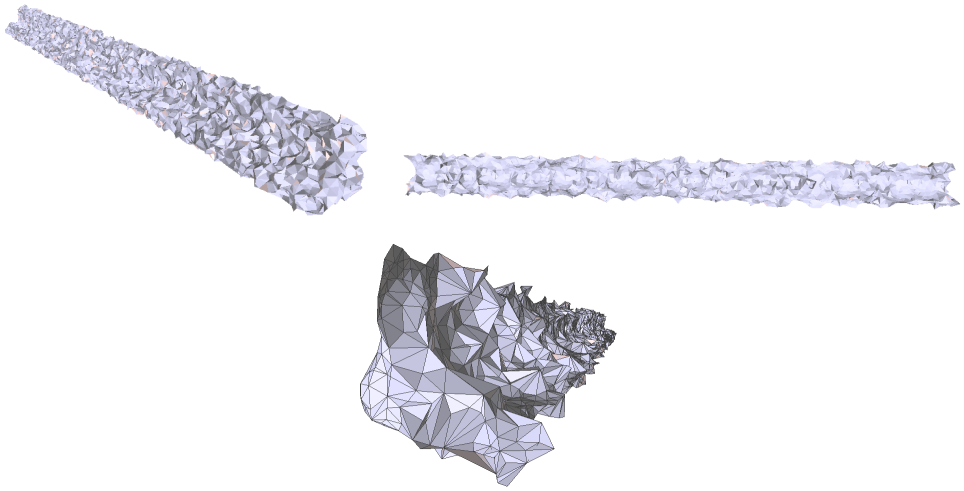


Figure 4.26: Result of initial meshing using the scale space reconstruction method with $S = 4$.

The Advancing Front reconstruction is once again ranked the highest in every quality metric used, but is visibly not suitable for volumetric 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	Successful Conversion
Advacing Front	0.1518	0.0877	0.0133	0.0924	314	No
Scale Space 2	0.0402	0.0594	0.0088	0.0259	1	Yes
Scale Space 4	0.0080	0.0592	0.0035	0.0043	1	No

Quality values shown 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 it's 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 4.27, the reconstruction at scale space 2 is relatively smooth compared the the raw point cloud, and exists as a singular bounded surface area.

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

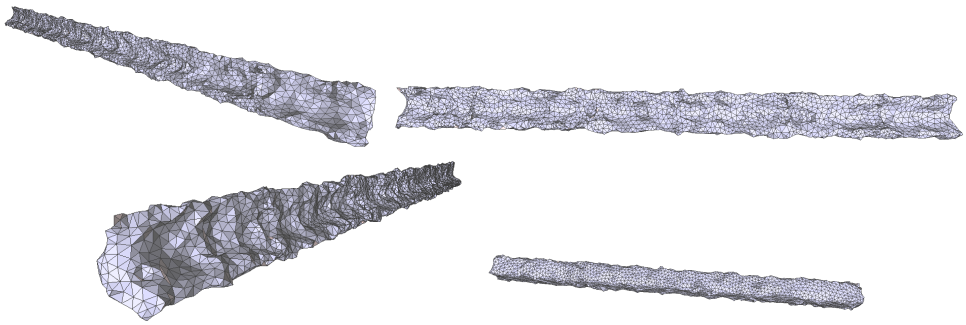


Figure 4.27: Scale space reconstruction $S = 2$ after 200 iterations of voronoi relaxation.

sharp edges.

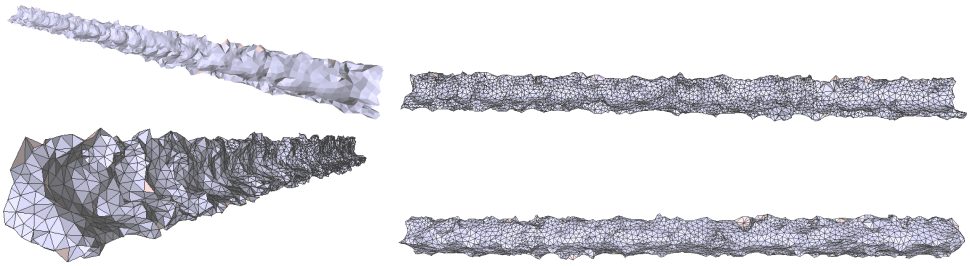


Figure 4.28: Scale space reconstruction $S = 4$ after 200 iterations of voronoi relaxation.

Effects of Delaunay Optimization

Delaunay optimization has some more noticable effects on the noisy point cloud than on the clean point cloud – unsurprising. These effects are most visible in Figure 4.29 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 4.30 shows

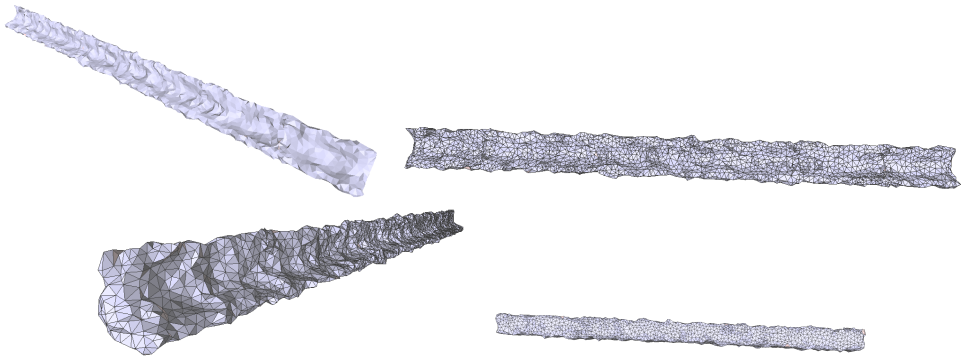


Figure 4.29: Scale space reconstruction $S = 2$ after ODT with a 30 second clock cap.

the extent ODT can go to to increase the quality in a mesh. The portion in the top 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 it's surrounding edges.

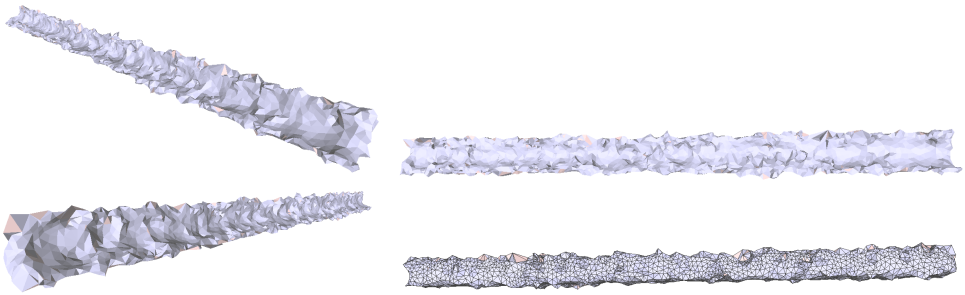


Figure 4.30: Scale space reconstruction $S = 4$ after ODT with a 30 second clock cap.

Because the reconstructions at scale 2 and 4 are already quite high in quality metrics, ODT did not shift many vertices noticably. 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 4.31 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.

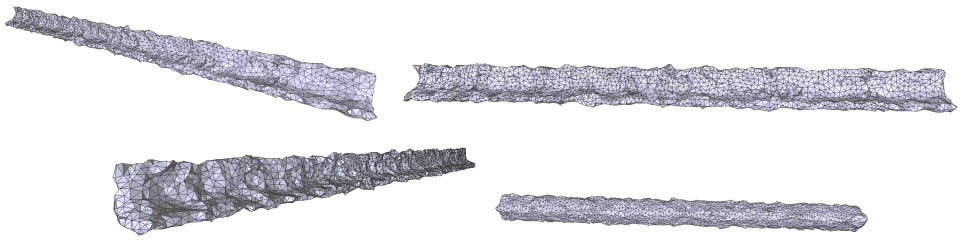


Figure 4.31: Scale space reconstruction $S = 2$ after perturbation with a 30 second clock cap.

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

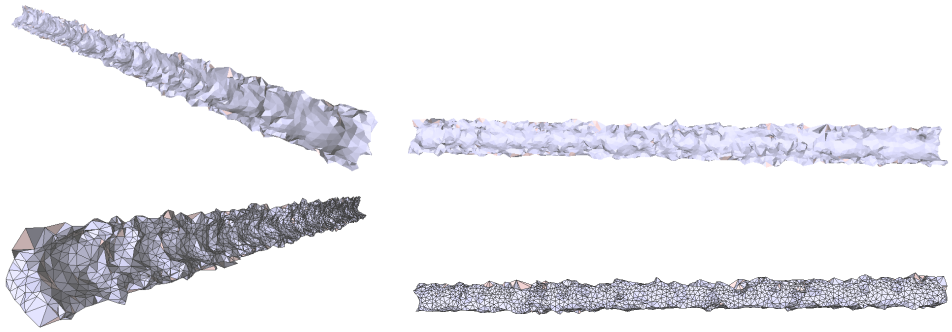


Figure 4.32: Scale space reconstruction $S = 4$ after perturbation with a 30 second clock cap.

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 4.33 shows the final optimized mesh at a scale space of 2.

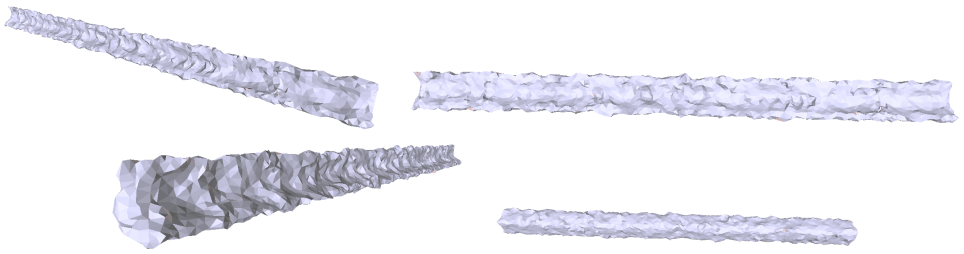


Figure 4.33: Scale space reconstruction $S = 2$ after exudation with a 30 second clock cap.

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 4.34, an evenly distributed set of tetrahedra defining the relatively smooth surface area of a beam with a parabolic cross-section is visible.

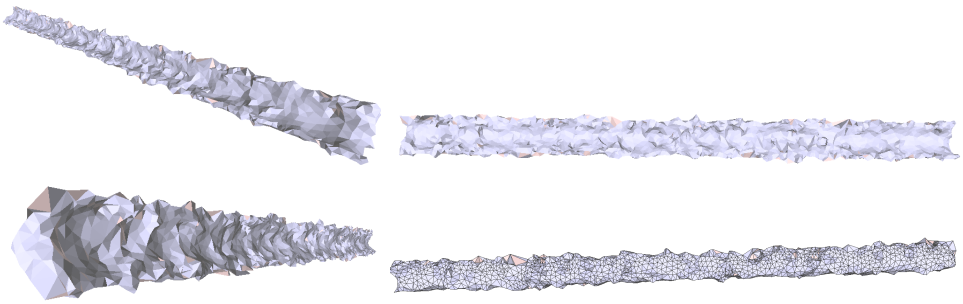


Figure 4.34: Scale space reconstruction $S = 4$ after exudation with a 30 second clock cap.

4.2.4 Quality Analysis

At the end of the optimization phase, both Scale Space reconstructions are compatible with volumetric 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 quantitatively 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	Successful Conversion
Initial Scale Space 2	0.0402	0.0594	0.0088	0.0259	1	Yes
Optimized Scale Space 2	0.1367	0.1077	0.0147	0.0955	1	Yes
Initial Scale Space 4	0.0080	0.0592	0.0035	0.0043	1	No
Optimized Scale Space 4	0.1166	0.1444	0.0552	0.0595	1	Yes

Quality values shown 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.

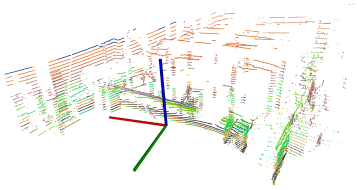


Figure 4.35: Individual frame from LiDAR scan. Image contains 32,000 points.

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 4.36 shows the results of the concatenation before any filters are applied.

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 4.37. 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.

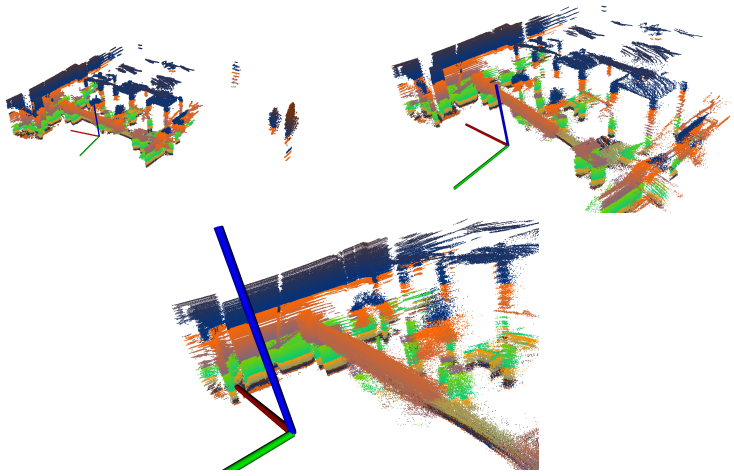


Figure 4.36: Concatenation of 100 individual LiDAR scan frames.

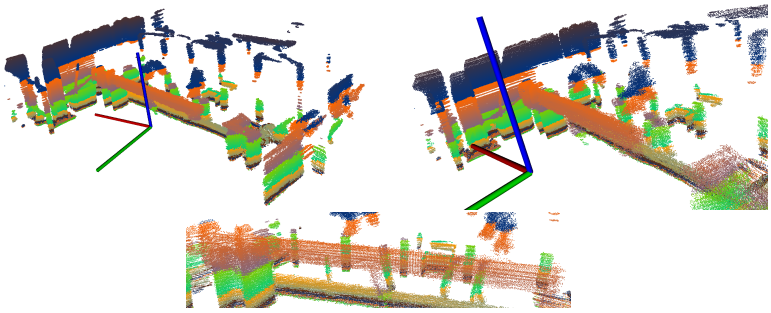


Figure 4.37: Concatenated LiDAR scans after noise filtering and down sampling.

4.3.1 Segmentation Method Comparison

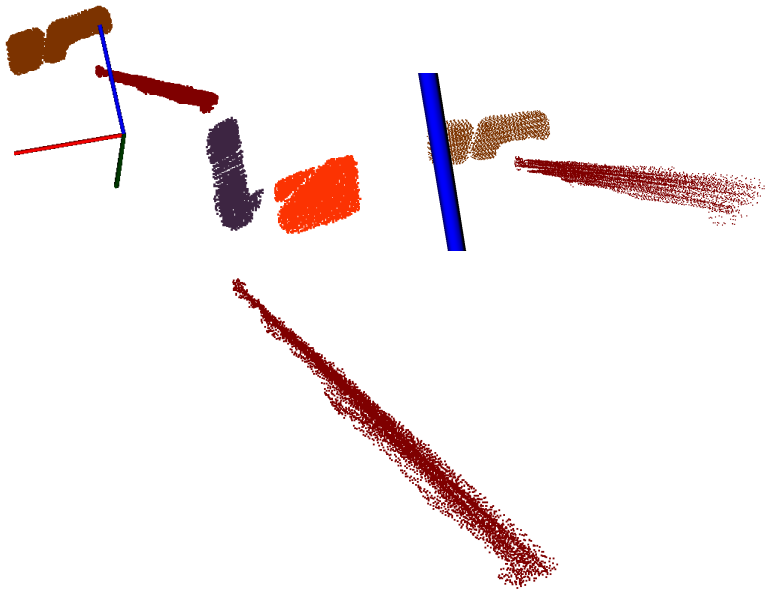


Figure 4.38: Results after euclidean clustering.

4.3.2 Initial Meshing Methods

The effects of the collection path and method are apparent in Figure 4.38. 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 volumetrically compatible mesh topology with incomplete or inaccurate datasets is invaluable in the sensing field.

Figure 4.39 shows the results of a raw advancing front meshing procedure on the segmented beam point cloud. Without any compensation in the form of noise reduc-

tion 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.

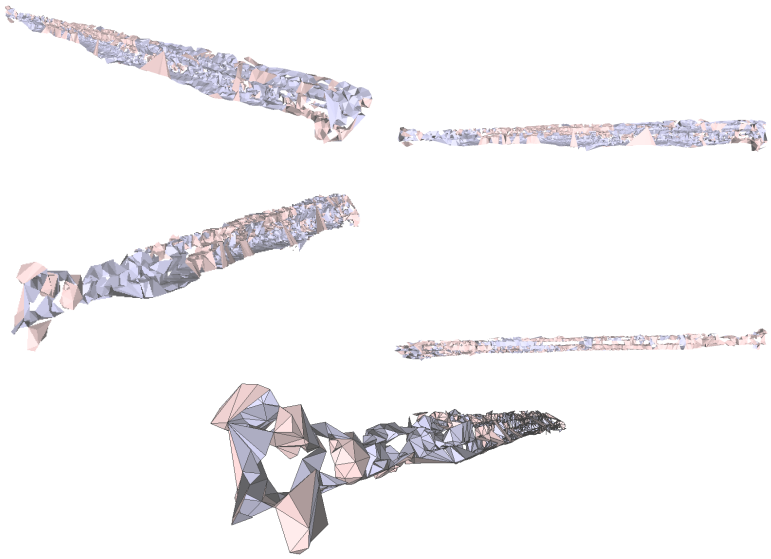


Figure 4.39: Advancing front reconstruction of resulting beam point cloud.

Figure 4.40 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 volumetric 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 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 volumetric conversion.

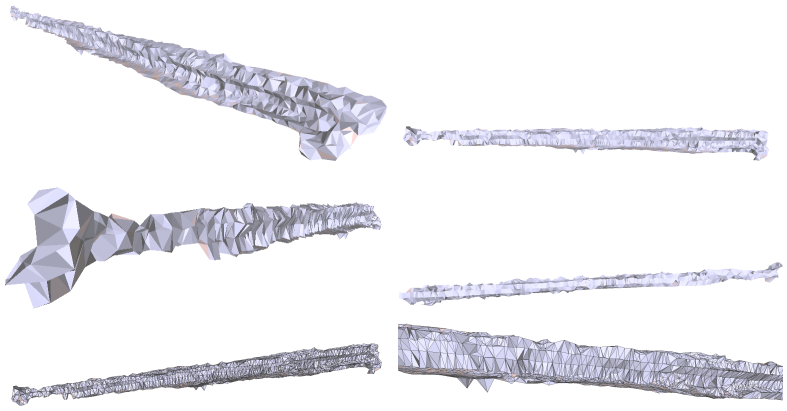


Figure 4.40: Scale space reconstruction at scale S=2 of resulting beam point cloud.

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.

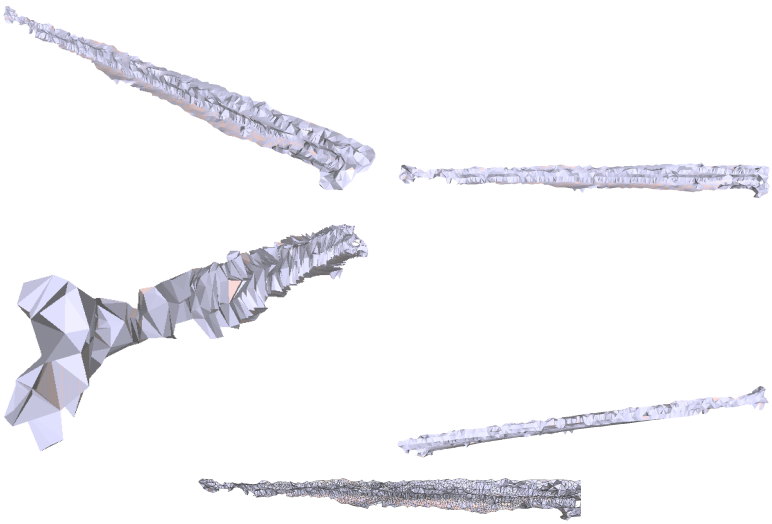


Figure 4.41: Scale space reconstruction at scale S=4 of resulting beam point cloud.

Table 4.5: Quality analysis of initial meshing methods.

Mesh Type	Q_1	Q_2	Q_3	Q_4	Bounded Surfaces	Successful Conversion
Advacing Front	0.1446	0.1072	0.0419	0.0849	28	No
Scale Space 2	0.0175	0.0524	0.0066	0.0093	2	No
Scale Space 4	0.0117	0.0562	0.0057	0.0059	2	No

Quality values shown 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 volumetric 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 it's 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. This is the last step incorporating the advancing front mesh, as it is clear this method is not optimal for non-ideal point clouds. Figure 4.42 shows the results of voronoi relaxation on the advancing front mesh.

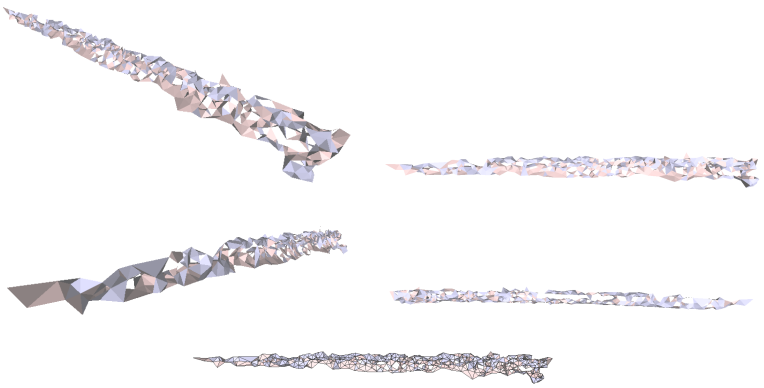


Figure 4.42: Lloyd + Advancing front reconstruction of resulting beam point cloud.

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

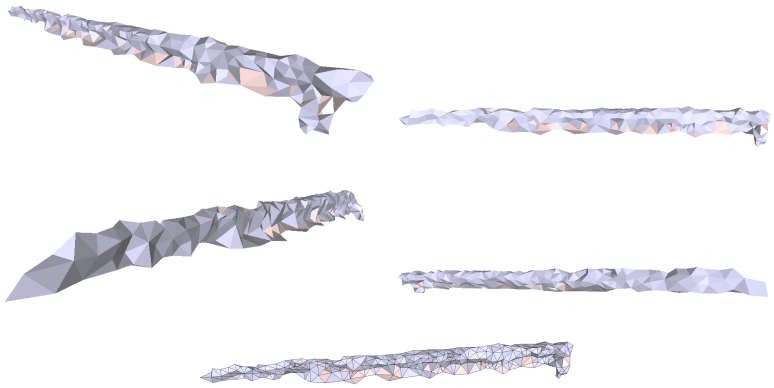


Figure 4.43: Lloyd + Scale space reconstruction at S=2 of resulting beam point cloud.

surface defines a rectangular prism, the horizontal section of the beam merges into it's vertical face. [[Second image on the left]] shows the phenomenon in detail, in Figure 4.44.

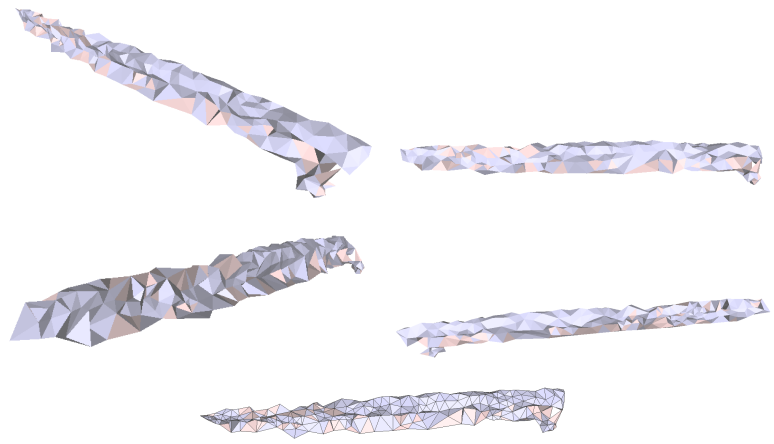


Figure 4.44: Lloyd + Scale space reconstruction at S=4 of resulting beam point cloud.

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]]

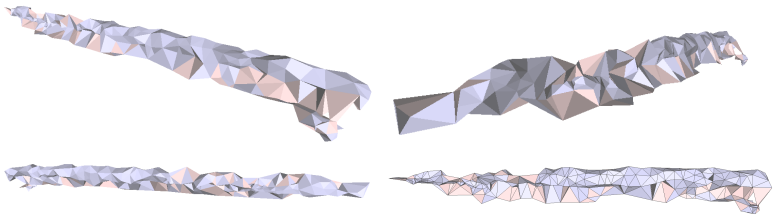


Figure 4.46: Lloyd + ODT + Scale space reconstruction at S=4 of resulting beam point cloud.

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 verticale 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.

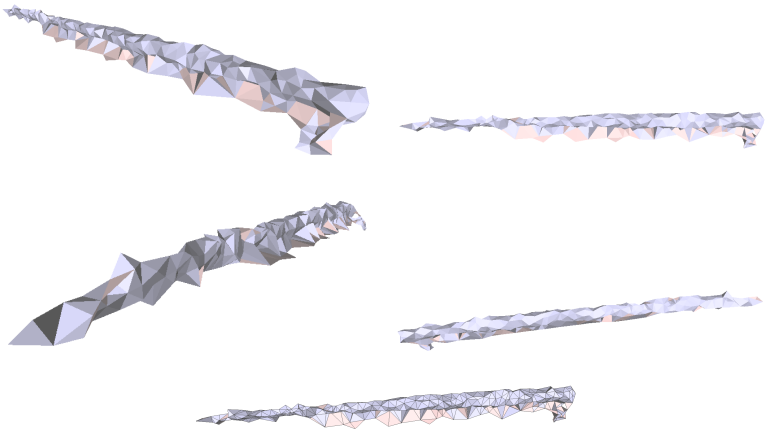


Figure 4.47: Lloyd + ODT + Perturb + Scale space reconstruction at S=2 of resulting beam point cloud.

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

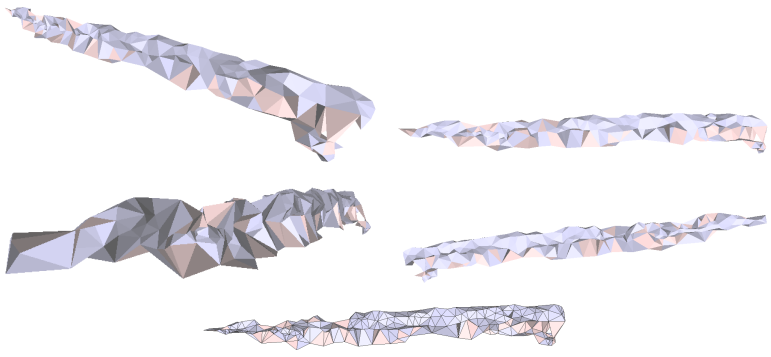


Figure 4.48: Lloyd + ODT + perturb + Scale space reconstruction at S=4 of resulting beam point cloud.

Exudation

Exudation represents the final step in the optimization process, and the complete surface mesh to be exported to a volumetric 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 volumetrically 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.

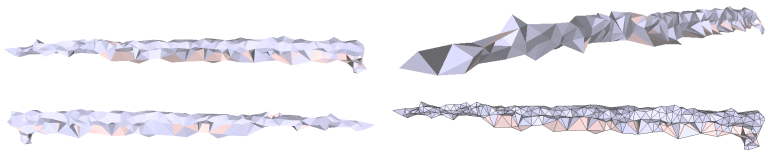


Figure 4.49: The full suite of optimization steps on the LiDAR reconstruction at S=2.

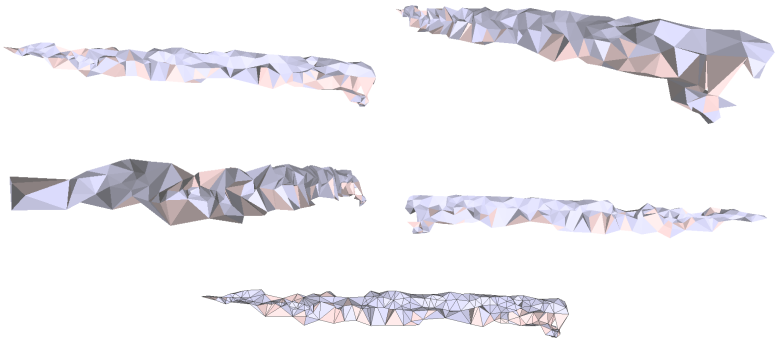


Figure 4.50: The full suite of optimization steps on the LiDAR reconstruction at S=4.

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 volumetric 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	Successful Conversion
Initial Scale Space 2	0.0175	0.0524	0.0066	0.0093	2	No
Optimized Scale Space 2	0.2629	0.1941	0.1099	0.1418	2	No
Initial Scale Space 4	0.0117	0.0562	0.0057	0.0059	2	No
Optimized Scale Space 4	0.2143	0.1229	0.0558	0.1087	2	No

Quality values shown represent the absolute minima in each category.

The end result of the algorithm on the experimental data, when fully automated, is failure. However, this 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	Successful Conversion
Optimized Scale Space 2	0.2629	0.1941	0.1099	0.1418	1	No
Optimized Scale Space 4	0.2143	0.1229	0.0558	0.1087	1	Yes

Quality values shown represent the absolute minima in each category.

Why the mesh at Scale Space 2 fails 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 meshes volumetric compatibility, they are not the only metrics that determine a meshes compatibility.

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

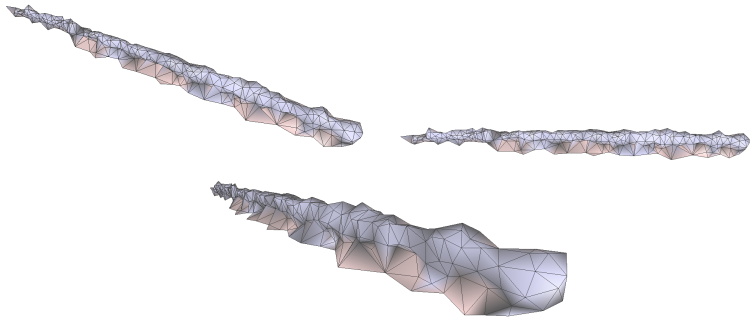


Figure 4.51: Manually corrected optimized scale space 2 reconstruction. [[Compare with Figure 4.49.]]

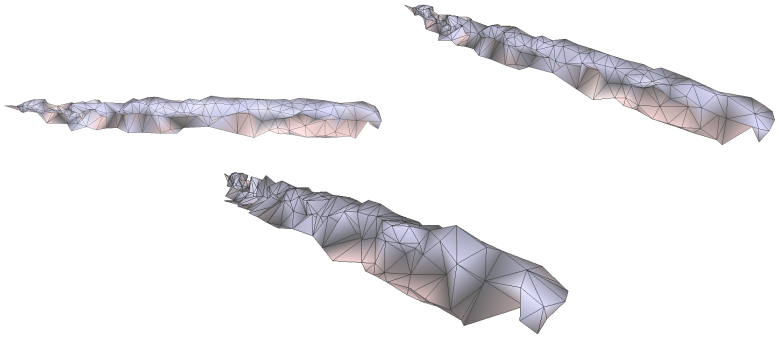


Figure 4.52: Manually corrected optimized scale space 4 reconstruction. [[Compare with Figure 4.50.]]

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 fails.

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	Successful Conversion
Zero noise AF	0.3096	0.191	1.333	0.1548	1
Zero noise SS2	0.0072	0.1085	0.0035	0.0037	0
Zero noise SS4	0.0031	0.0486	0.0016	0.0016	0
2cm noise SS2	0.1367	0.1077	0.0147	0.0955	1
2cm noise SS4	0.1166	0.1444	0.0553	0.0595	1
Experimental SS2	0.2629	0.1941	0.1099	0.1418	0
Experimental SS4	0.2143	0.1229	0.0558	0.1087	0

AF – Advancing Front, SS – Scale Space

Evaluating these metrics based on successful conversion, 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.

6. Future Work

6.1 Informed Shape Estimation

While the simulated point clouds provide information for the entirety of the L-beam's shape, the true LiDAR collection data does not. This section walks through the results of the Informed Shape Estimation algorithm, defined in Section 3.3.9. For each dataset, the original pointcloud of the isolated beam, the initial meshing method, the estimated point cloud, and the fully optimized mesh using the estimated cloud are shown.

[[Still need to generate this data]]

Bibliography

[1] GARTNER. Gartner hype cycle. [Online]. Available: https://www.gartner.com/technology/research/methodologies/hype-cycle.jsp

[2] S. Siebert and J. Teizer, "Mobile 3d mapping for surveying earthwork 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

[3] 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 Photgrammetry, Remote Sensing and Spatial Information Siences*, vol. XL, no. 5/W4, 2015.

[4] G. Castellazzi, A. M. DAltri, 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.

[5] U. Clarenz, M. Rumpf, and A. Telea, "Finite elements on point based surfaces," in *Eurographics Symposium on Point-Based Graphics*, Conference Paper.

[6] H. Hu, H. Hu, T. M. Fernandez-Steeger, M. Dong, and R. Azzam, "Numerical modeling of lidar-based geological model for landslide analysis," *Automation in construction*, vol. 24, pp. 184–193.

[7] 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

[8] 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.

[9] 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

[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] 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.

[14] 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.

[15] 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.

[16] *Modelling Background*. London: Springer London, 2006, pp. 11–27.

[17] N. P. Weatherill and O. Hassan, "Efficient threedimensional 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:https://doi.org/10.1002/nme.1620371203

[18] 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

[19] 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.

[20] J. Digne, "An implementation and parallelization of the scale-space meshing algorithm," *Image Processing On Line*, 2015.

[21] 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

[22] 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

[23] 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

[24] S. Lloyd, "Least squares quantization in pcm," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[25] J. Tournois, R. Srinivasan, and P. Alliez, "Perturbing slivers in 3d delaunay meshes," in *18th International Meshing Roundtable*, HAL, Ed. HAL, Conference Paper.

[26] 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.