Automated Conversion of 3D Point Clouds to FEA Compatible Meshes

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

Abstract **x**

1 Introduction **0**

1.1 Related Work 0

1.1.1 Geographical Topography Mapping 0

1.1.2 Building Informational Modeling 1

1.1.3 Aerial Scanning for GPS Overlay 1

1.1.4 Semi-Automated Point Cloud to FEA modeling 1

2 Background **2**

2.1 Collection of Point Cloud Data 2

2.1.1 Implicit collection methods: Stereogrammetry and Structure from Motion 2

2.1.2 Explicit Methods: LiDAR, Laser Scanning, and Ultrasonic Sensing 2

2.2 Sensor Fusion 2

2.3 Point Cloud Pre-processing Methods 3

2.3.1 Registration 3

2.3.2 Filtering 5

2.3.3 Down-sampling 6

2.3.4 Handling Occlusion 7

2.4 Machine Learning for Object Recognition and Segmentation 12

2.4.1 Supervised Methods — Neural Networks 12
2.4.2 Unsupervised Methods 17
2.4.3 Converting a Discrete Point Cloud to a Bounded Area Surface Mesh 22
2.4.4 Mesh Optimization 32

3 Objective, Hypothesis, and Technical Approach 36
3.1 Research Objective 36
3.2 Hypothesis Statement 36
3.3 Technical Approach 37
3.3.1 General: Zero Noise 38
3.3.2 General: Estimated Noise and Occlusion 38
3.3.3 LiDAR: Velodyne HDL-32E 39
3.3.4 Stereogrammetry and Photogrammetry 40
3.3.5 Segmentation Method Exploration 40
3.3.6 Initial Mesh Exploration 40
3.3.7 Optimization Steps 40
3.3.8 Final Mesh Verification 40

4 Results 41
4.1 Simulated Data: Zero Noise 41
4.1.1 Segmentation Method Comparison 43
4.1.2 Initial Meshing Methods 44
4.1.3 Optimization 49

4.2 Simulated Data: Applied Gaussian noise $\pm 2cm$ 52
4.2.1 Segmentation Method Comparison 53
4.2.2 Initial Mesh Methods 53
4.2.3 Optimization 58
4.3 HDL-32E LiDAR Scan Data 59
4.3.1 Segmentation Method Comparison 60
4.3.2 Initial Meshing Methods 61
4.3.3 Optimization 64

5 Conclusion **69**

6 Future Work **70**

Bibliography **71**

List of Tables

2.1 Comparison of Unsupervised clustering methods 22
2.2 Effect of increasing scale in a scale space reconstruction 30
2.3 Quantification of mesh quality 33

List of Figures

2.1 Intrinsic Shape Signature feature vectors 5
2.2 Effects of noise filter on simulated point cloud objects 6
2.3 Effects of down-sampling on simulated point cloud objects 7
2.4 Demonstration of occlusion 8
2.5 Flow chart of informed shape estimation algorithm 12
2.6 High level flow chart of a convolutional neural network 14
2.7 Common CNN non-linear activation functions 16
2.8 Block diagram of CNN flow 16
2.9 Comparison of divisive and agglomerative clustering methods 20
2.10 Simple example of surface mesh components 23
2.11 Components of a boundary representation. 24
2.12 Example of a 2-dimensional nurbs spline. 25
2.13 2D delaunay triangulation 27
2.14 Example of an edge vs. a half edge 31
2.15 Definitions for triangulation quality measures 33
2.16 Demonstration of voronoi relaxation over several iterations 33
3.1 Velodyne LiDAR Operation 39
3.2 Comparison of simulated and real HDL-32E scan data 40
4.1 Simulated laboratory point cloud data with zero noise induced 42
4.2 Zero noise simulated data after being filtered with voxel size of 0.1 in^3, 15 minimum point neighbors at a distance of 1 standard deviation from the mean point to point distance 42
4.3 Comparison of unsupervised segmentation techniques on a simulated dataset. 43
4.4 Euclidean distance segmentation with ideal parameters. 44
4.5 Initial meshing using a raw advancing front approach 45
4.6 Initial meshing using a scale space reconstruction with $S = 2$ 46
4.7 Initial meshing using a scale space reconstruction with $S = 4$ 47
4.8 Initial meshing using a scale space reconstruction with $S = 15$ 47
4.9 Advancing Front mesh after 200 iterations of voronoi relaxation 49
4.10 Scale space reconstruction $S = 2$ after 200 iterations of voronoi relaxation 50
4.11 Scale space reconstruction $S = 4$ after 200 iterations of voronoi relaxation 51
4.12 Comparison of mesh before and after perturbation 51
4.13 Scale space reconstruction $S = 2$ after 200 iterations of voronoi relaxation and with perturbation 52
4.14 Simulated point cloud with 2cm magnitude gaussian noise induced 52

4.15 2 cm gaussian noise simulated data after being filtered with voxel size of 0.1 in^3, 15 minimum point neighbors at a distance of 1 standard deviation from the mean point to point distance 53
4.16 Comparison of unsupervised segmentation techniques on simulated dataset with induced noise. 53
4.17 Initial meshing using a raw advancing front approach 54
4.18 Initial meshing using a scale space reconstruction with $S = 2$ 55
4.19 Initial meshing using a scale space reconstruction with $S = 4$ 56
4.20 Initial meshing using a scale space reconstruction with $S = 15$ 57
4.21 Scale space reconstruction $S = 2$ after 200 iterations of voronoi relaxation 58
4.22 Scale space reconstruction $S = 4$ after 200 iterations of voronoi relaxation 58
4.23 Individual LiDAR scan frame 59
4.24 Concatenation of 100 individual LiDAR scan frames 60
4.25 Concatenated LiDAR scans after noise filtering and down sampling 60
4.26 Euclidean clustering results on HDL-32E LiDAR scans 61
4.27 Advancing front reconstruction of segmented LiDAR data 62
4.28 Scale space reconstruction at scale S=2 of segmented LiDAR data 63
4.29 Scale space reconstruction at scale S=4 of segmented LiDAR data 64
4.30 Lloyd + Advancing front reconstruction of segmented LiDAR data 65
4.31 Lloyd + Scale space reconstruction at S=2 of segmented LiDAR data.. 66
4.32 Lloyd + Scale space reconstruction at S=4 of segmented LiDAR data.. 66
4.33 Lloyd + ODT +Scale space reconstruction at S=2 of segmented LiDAR data 66
4.34 Lloyd + ODT + Scale space reconstruction at S=4 of segmented LiDAR data 67
4.35 Lloyd + ODT + perturb +Scale space reconstruction at S=2 of segmented LiDAR data 67
4.36 Lloyd + ODT + perturb + Scale space reconstruction at S=4 of segmented LiDAR data 68

Abstract

Automated Conversion of 3D Point Clouds to FEA Compatible Meshes

Matthew S. Brown

Antonios Kontsos, Ph. D.

Technology today is increasing at a such a rate that structurally obselete objects are still in regular use throughout industry. While today's design and construction methods include CAD drawings, finite element models, and countless other digital analysis tools, many objects in the field do not have any support data available due to their age. In an attempt to solve this problem, this paper outlines a method utilizing machine learning in conjunction with advanced meshing techniques to autonomously segment raw point cloud data and reconstruct the resulting segments into simply connected meshes. We seek to quantify traits in unordered point clouds and surface meshes necessary for fully volumetric convertible surface meshes by evaluating readily available metrics and methods for determining mesh quality. This thesis proposes a method to convert real objects in the field into analyzable models with minimal user interface in the process.

1. Introduction

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. All these processes, however, do not provide a general method to automate the entire pipeline between point-cloud collection and simple CAD geometry. We present a solution to this problem in the form of a robust and autonomous process for filtering, segmentation, and meshing of raw 3D point clouds.

1.1 Related Work

[Introductory filler]

1.1.1 Geographical Topography Mapping

In 2009, José Lerma and his team of archaeologists began using Terrestrial laser scanning in tandem with close proximity photogrammetry to render high resolution 3D surface models of ancient caves in Spain. Lerma et. al. are general in their description of their meshing method, which is most likely due to their non-computer programming oriented backgrounds. However, their pipeline involves sensor fusion between their laser scanner, which returns a pure point cloud with an origin at the center of the instrument, and deduced point clouds from photogrammetry data. The result is an impressive, high resolution surface mesh that accurately captures the

features relevant to an archaeologist, but provide no useful information in terms of structural health monitoring [1].

In 2014, Sebastian Siebert implemented similar technology mounted to UAVs to provide 3D mapping of earthwork projects for surveyors [2].

1.1.2 Building Informational Modeling

[3]

1.1.3 Aerial Scanning for GPS Overlay

1.1.4 Semi-Automated Point Cloud to FEA modeling

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

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

[[fill in]] [8]

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

[[fill in]]

2.2 Sensor Fusion

[[fill in]]

2.3 Point Cloud Pre-processing Methods

2.3.1 Registration

Position Data

[[fill in]]

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 in this paper, we will 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, we evaluate the relationship of a point, p_i, with the points inside of it's neighborhood. 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 [9].

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, we build a sphere of some desired radius r centered at pi and divide it 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 [9].

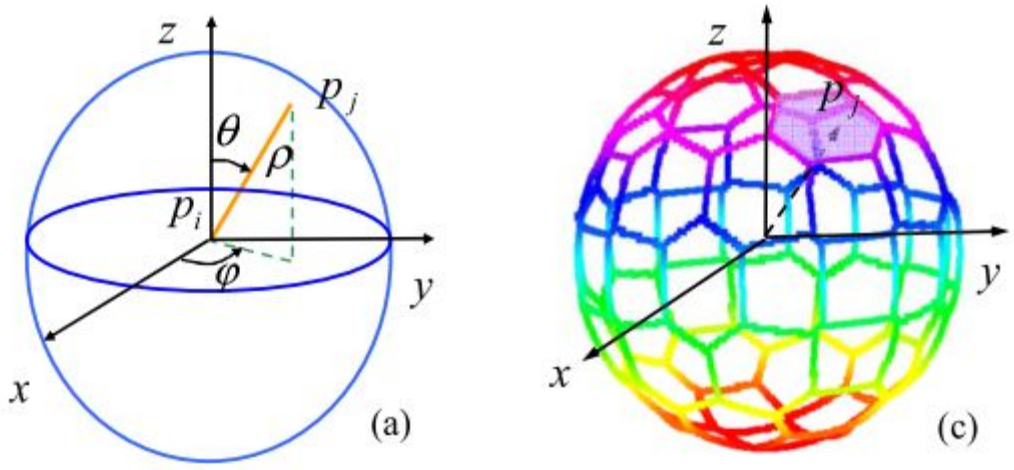


Figure 2.1: Feature vector calculation via spherical bin decomposition [9].

2.3.2 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.

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

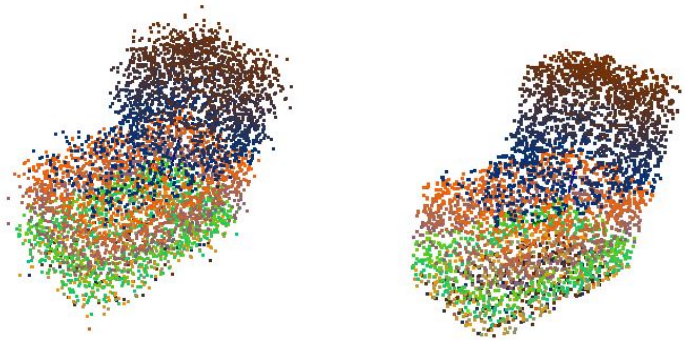


Figure 2.2: The effects of noise filtering on a simulated L block with 10% induced noise.

2.3.3 Down-sampling

Down-sampling is the process of fixing a point cloud's mean density to a voxel of size n. This is done by iterating the voxel throughout the cloud's entire volume and replacing all points occupying a voxel with a single point in the mean position of the voxel. For an n-dimensional feature-set, the downsampling equation can be defined by Equation 2.4

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

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 2.3 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.

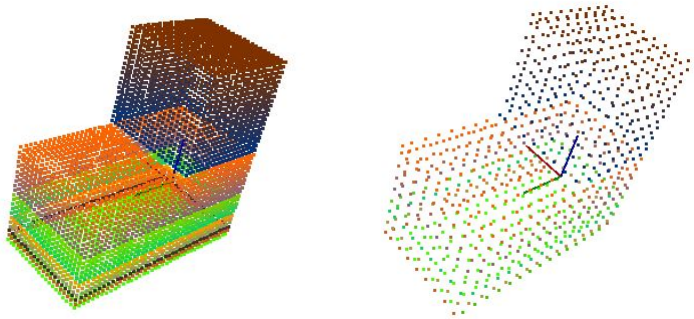


Figure 2.3: A simulated L block point cloud down-sampled with voxel size = 0.25 in^3.

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 we can force the input cloud to an expected density. Section 2.4.3 describes the meshing steps in more detail.

2.3.4 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 we cannot see inside the box. 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.

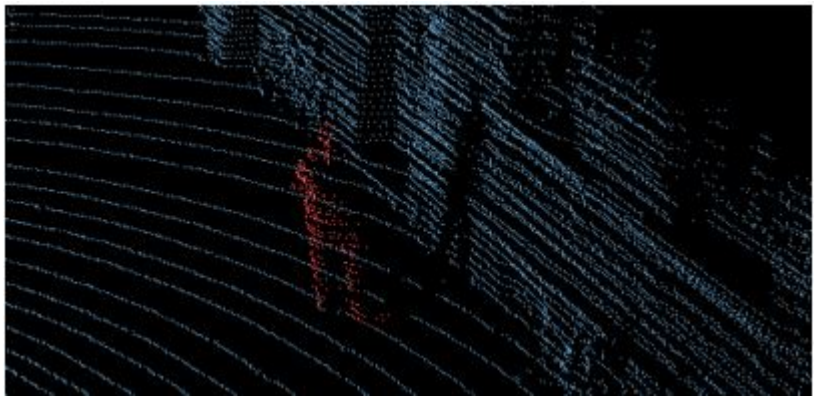


Figure 2.4: Section of a building occluded by an object in the foreground [10].

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 necessary first step in removing excess noise / unnecessary information from the cloud [10].

Informed Shape Estimation

Occlusion is the cause of two main problems. The first being the unnecessary information provided by objects blocking the instruments view to our 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:

$$C = \sum_{i=0}^{N} p(i) - p_{proj}(i) \tag{2.8}$$

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.

$$p_{proj}(i) = arg_{min} \frac{p(i) \bullet l(j)}{||l(j)||} \in p_{normal}(i) \bullet l_{normal}(j) \geq \tau \tag{2.9}$$

$$x_{k+1} = x_k - \frac{C(x_k)}{C'(x_k)} \tag{2.10}$$

Using a Newton Raphson interative regression, shown in equation 2.10 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.

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.

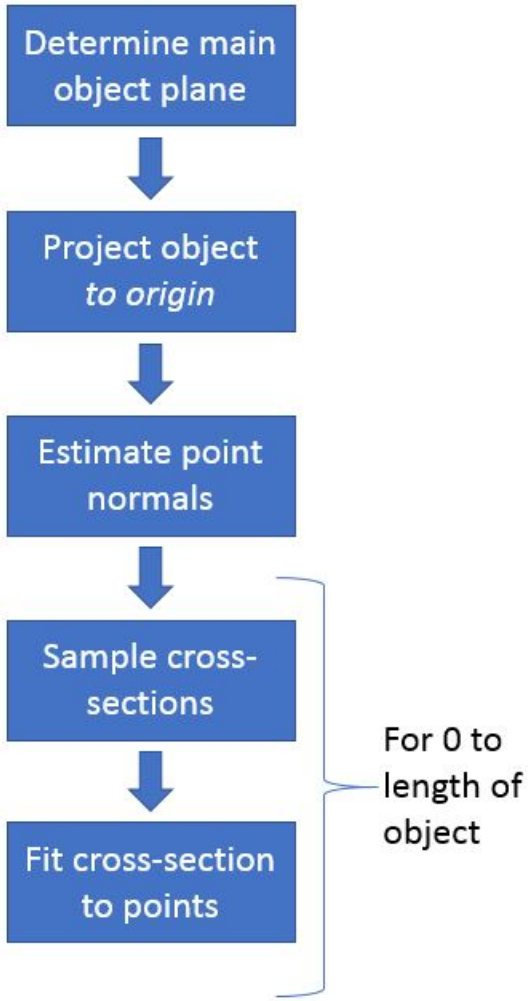


Figure 2.5: A flowchart of the informed shape estimation algorithm.

2.4 Machine Learning for Object Recognition and Segmentation

2.4.1 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 machine's perspective – the dataset needs to be modified to enunciate those features. Key features in machine vision include edges, corners, and areas with distinctive geometry. Convolutions are the key to bringing these features to the forefront of the image. A convolution kernel is a weighted square matrix of dimensions m, and depth 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.

$$C_{new} = \sum_{i}^{N} w(i)x_i \tag{2.11}$$

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.

$$C_{new} = \frac{\sum_{i}^{N} x_i}{N} \tag{2.12}$$

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:

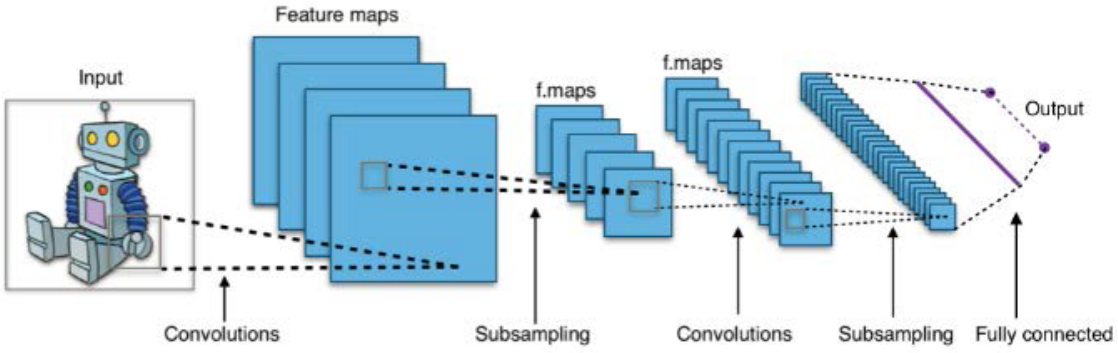


Figure 2.6: Map of a convolutional neural network with two hidden layers [ref]

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.

$$\begin{bmatrix} C_1 \\ \vdots \\ C_n \end{bmatrix} = \begin{bmatrix} w_{1,1} & \dots & w_{1,m+1} \\ \vdots & \ddots & \dots \\ w_{n,1} & \dots & w_{n,m+1} \end{bmatrix} \begin{bmatrix} f_1 \\ \vdots \\ f_n \\ 1 \end{bmatrix} \tag{2.13}$$

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:

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

For the system to be accurate, the weights for each convolution and connection

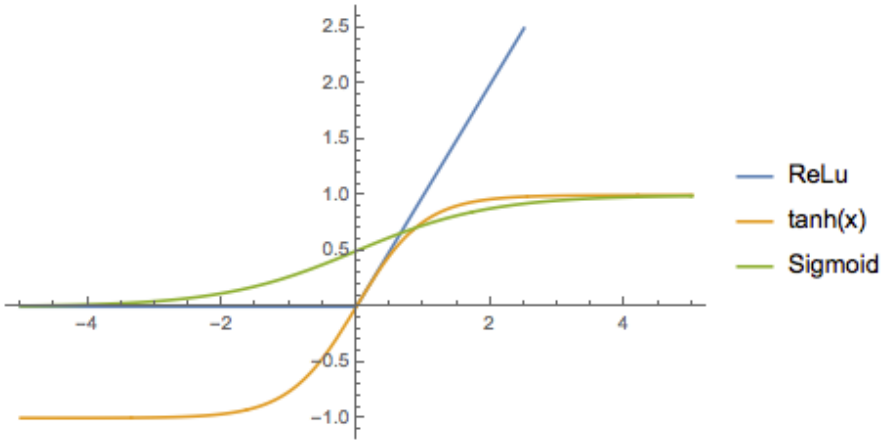


Figure 2.7: Visualization of commonly used non-linear activation functions

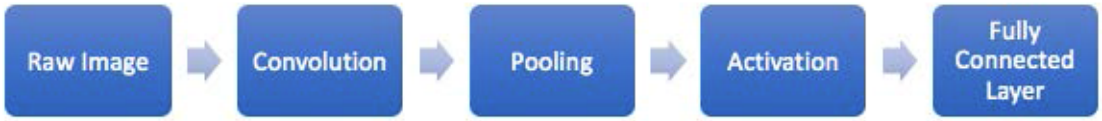


Figure 2.8: Block diagram of a CNN′s skeleton structure

must be trained. This training is done through a process called backpropagation. An image-set of known classifications is fed to the untrained system, and the error between the system's classification and the true classification of each image is used to update the weights iteratively until the machine's class prediction closely resembles ground truth. Most algorithms use numerical solving methods to sharply diminish the number of iterations required for convergence. The "gradient descent" method is commonly used in the machine learning world due to its rapid convergence properties and low computational complexity. The method is shown below:

$$x_{k+1} = x_k - \gamma \nabla F(x_k) \tag{2.14}$$

$F(x)$ 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.

$$F(x) = \begin{bmatrix} C_1 \\ \vdots \\ C_n \end{bmatrix} - \begin{bmatrix} w_{1,1} & \dots & w_{1,m+1} \\ \vdots & \ddots & \dots \\ w_{n,1} & \dots & w_{n,m+1} \end{bmatrix} \begin{bmatrix} f_1 \\ \vdots \\ f_n \\ 1 \end{bmatrix} \tag{2.15}$$

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 [11]. Because CNNs – and all supervised learning methods – require a large amount of training data, they do not fit within the scope of the proof of concept inside of this paper. However, supervised learning has proven to be one of the most effective forms of classification, and will be addressed in the future work section of this thesis.

2.4.2 Unsupervised Methods

With our goal being to isolate specific objects in a structural health monitoring setting, the ideal segmentation method is a supervised learning algorithm, such as a convolutional neural network, that semantically parses the point cloud based on a training set of pre-defined cloud objects [12]. However, due to time limitations, and

a lack of training data relevant to our objects of interest, this is not possible. Instead we explore a series of unsupervised clustering methods on the assumption that objects are distinct enough in relative cloud neighborhoods to be properly segmented.

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}.

$$argmin_S \sum i = 1^k \sum x \in S_i |x - c_i|^2 \tag{2.16}$$

K-means clustering is arguably the most well known clustering method, and is useful for a massive variety of naïve classification problems. In any situation where the bin-size is known and the data is clearly separated in some feature space $\mathbb{R}^n$, k-means proves to be an effective clustering method. It's ease of implementation comes at the cost of computate. As with most unsupervised clustering methods, the major downside to k-means is 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.

$$c_k = \frac{\sum xw_k(x)^m x}{\sum xw_k(x)^m} \tag{2.17}$$

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:

$$w_{ij} = \frac{1}{\sum_{k=1}^{c} (\frac{|x_i - c_j|}{|x_i - c_k|})^{\frac{2}{m-1}}} \tag{2.18}$$

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 Dvisive 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 our 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.

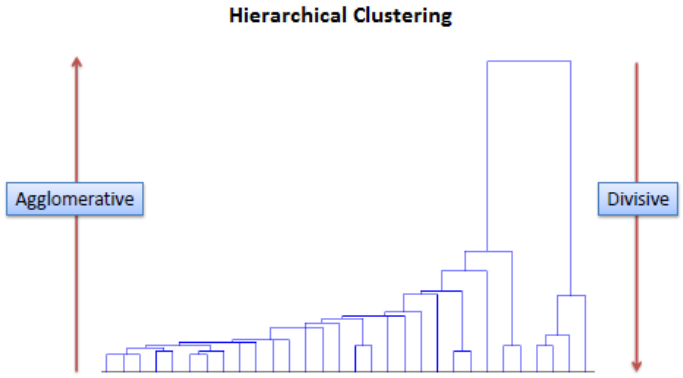


Figure 2.9: Comparison of divisive and agglomerative hierarchical clustering

[[GENERAL USAGE, PROS, CONS]]

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

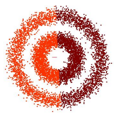
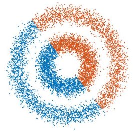
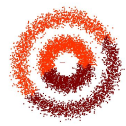
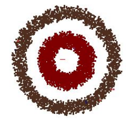
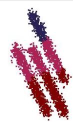
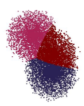
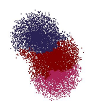
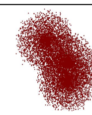
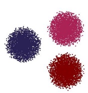
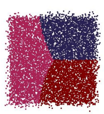
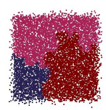
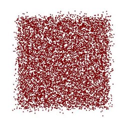
Perhaps the simplest of the algorithms listed above, Euclidean distance clustering operates on the pretense that objects are separated significantly enough spatially from another that clustering points based on their proximity to other points in the cloud is sufficient to properly segment the dataset. 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.

Comparison of Methods

Observing the 2-dimensional sample data, it is clear Euclidean clustering has a significant edge over the other clustering methods. We will see in Chapter 4 how it fairs upon 3-dimensional clouds.

Table 2.1: Comparision of unsupervised clustering methods on various simulated point clouds

K-means	Fuzzy C-means	Agglomerative	Divisive	Euclidean
				
				
				
				
				

2.4.3 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.

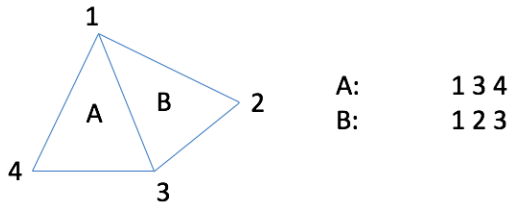


Figure 2.10: Visualization of necessary components in a surface mesh.

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.

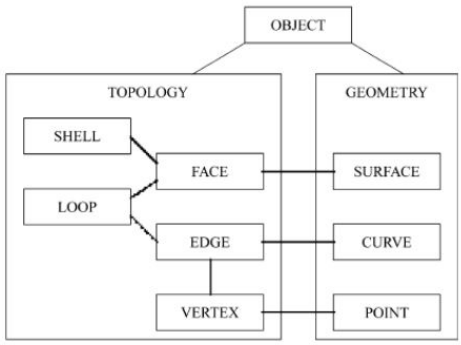


Figure 2.11: An overview of the components involved in a boundary representation of a solid object [13].

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).

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:

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

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

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

Leaving us with the following relationship:

$$f(u) = \sum_{i=0}^{K-1} R_{i,d}(u) P_i \tag{2.21}$$

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

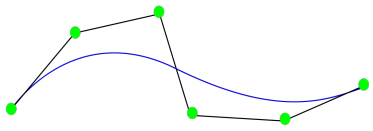


Figure 2.12: A 2-dimensional example of a nurbs spline. Green points represent the control points, P, black lines represent their connectivity, and the blue line represents one possible nurbs spline.

Delaunay Triangulation

Surface meshing is the science 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.

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

If every point pair sharing a Voronoi boundary are connected, the result is a triangulation object encasing the pointset. This object is referred to as a Delaunay triangulation [14].

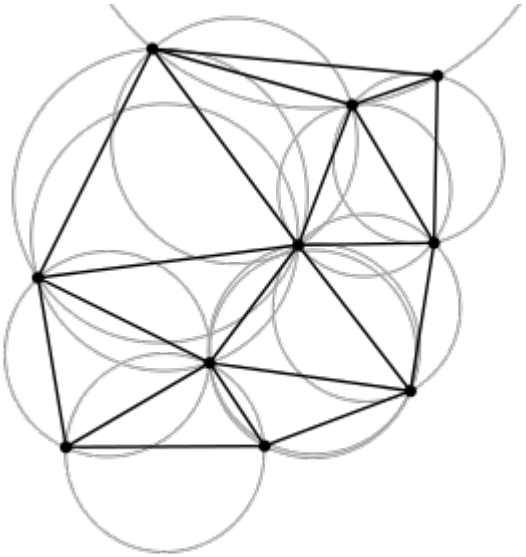


Figure 2.13: Visual represention of a 2-dimensional delaunay triangulation process

Advancing Front / Marching Triangles

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

In the Advancing Front method, a mesh is constructed by progressively adding tetrahedra starting at the boundaries of the previous surface elements. At each iteration, the mesh boundary is propagated further across the set of surface points. New tetrahedra are determined based on the minimal delaunay 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 [15].

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 it's 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 $v_0 = arg_{min} eigenvector(C)$

$$\text{set } p' = p - \langle p - p_{bar}, v_0 \rangle v_0$$

$$p' \cdot n = \frac{p-p'}{|p-p'|} \cdot sign(\langle p - p', p \cdot n \rangle)^*$$

Algorithm 2: Scale Space Iterator

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

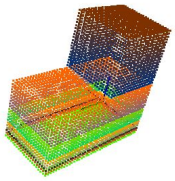
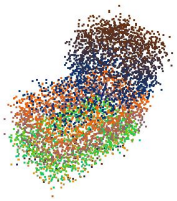
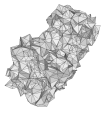
Output: A modified point set P_N

```
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 [16], 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 [17].

Table 2.2: Effect of increasing scale in a scale space reconstruction

	 Clean Point Cloud	 10% Induced Noise
Scale Space 3		
Scale Space 5		
Scale Space 10		

Hole Patching, Fairing, and Refinement

Surface meshes, defined in Section 2.4.3, 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. We will define two types of mesh edges 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 our surface area.

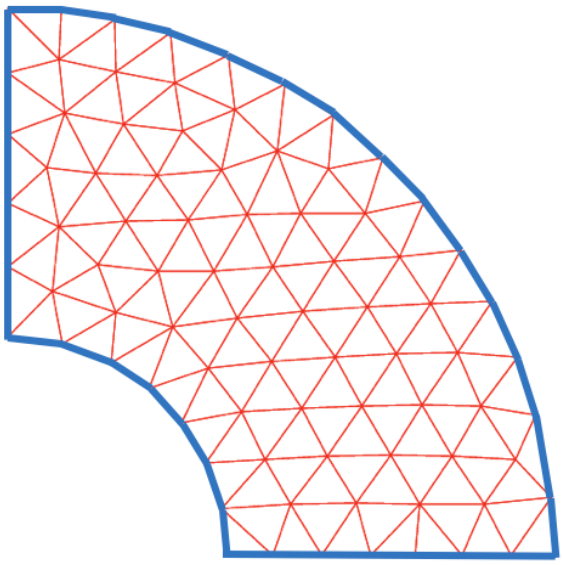


Figure 2.14: A visualization of the difference between an edge and a half edge. Half edges are outlined in blue.

To remedy this, we run a delaunay triangulation search 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]]

2.4.4 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

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.

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. We define these criteria 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.

It is easy to categorize meshes by quality via visual inspection, but in order to resolve low quality areas autonomously, we must define mathematically what it means to be "low quality." Table 2.3 lists some of the ratios we will use in this thesis 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 2.15.

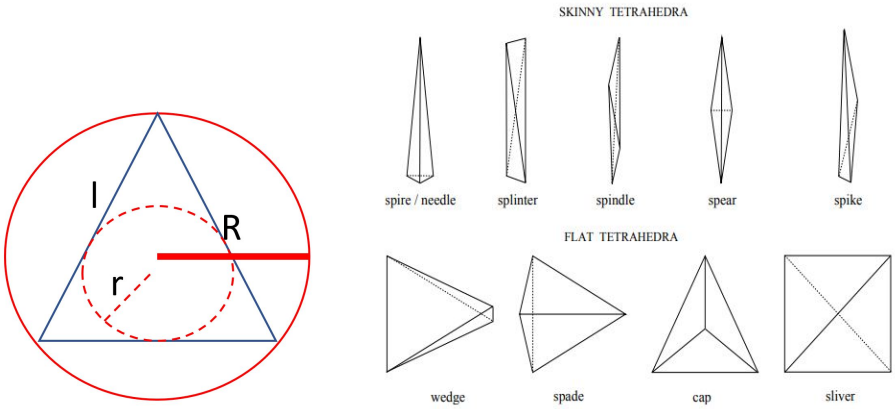


Figure 2.15: Definitions for triangulation quality measures, and typical shapes with poor quality. R = circumsphere radius, r = inscribed sphere radius, l = edge length.

Table 2.3: 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

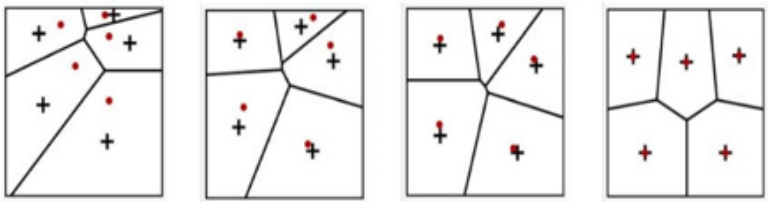


Figure 2.16: From right to left: Voronoi relaxation over 1 iteration, 5 iterations, 10 iterations, and 15 iterations

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 shape's topology significant due to its heavy smoothing capabilities [19].

Optimal Delaunay Triangulation

Delaunay triangulation, as described in Section 2.4.3, optimizes the connectivity for a finite point set through voronoi propagation. To calculate the optimal triangulation 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 $\tau^* \in P_N$ we use the following criteria:

$$Q(\tau^*, f, p) = \underset{\tau \in P_N}{inf}(\tau, f, p) \tag{2.23}$$

[[what does this mean / what does it mean to be optimal]]

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 [20].

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 it's 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 [21].

3. Objective, Hypothesis, and Technical Approach

Raw point clouds can be autonomously segmented into intuitive objects with or without supervised machine learning methods. The resulting objects can be converted from discrete point clouds to CAD models and volumetric meshes fully compatible with finite element analysis methods.

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. The research described in this thesis is inarguably very wide in application and scope, but is geared specifically towards Structural Health Monitoring and Non-Destructive Test and Analysis. The results of this Master's thesis are a proving ground for rapid scanning, meshing, and analysis of real-world objects.

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.

3.3 Technical Approach

To fully evaluate the effect of each step in the algorithm and test the robustness of the proposed algorithm, we apply it to a series point clouds, but 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

We explore the pipeline 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 it's 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. If the pipeline cannot work for the optimal case, it is pointless to attempt it's use on suboptimal clouds.

[[add figures of each pointcloud set (also generate each pointcloud set)]]

3.3.2 General: Estimated Noise and Occlusion

As the instruments within the scope of this thesis vary in expense, acquireability, and technical specifications, it is necessary to simulate their expected results. We can do this by modeling each instrument's acquisition technique, and applying that technique to the simulated room point cloud at a high density. The following figure set outline the collection methods of each technique, and their simulated accuracy.

[[Add all these figures]]

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°

Like all LiDAR devices, the HDL-32E does not return cartesian co-ordinate cloud, but instead a spherical co-ordinate cloud, where each point is defined by ϕ, it's angle about the z-axis, θ, it's 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.

Figure 3.1: Visualization of how to the Velodyne HDL-32E operates.

Figure 3.2 shows a comparison between a single frame of simulated HDL-32E LiDAR data and a true scan with the device. The simulation proves to be quite similar in results to the true scans, which will prove to be useful for future endeavors

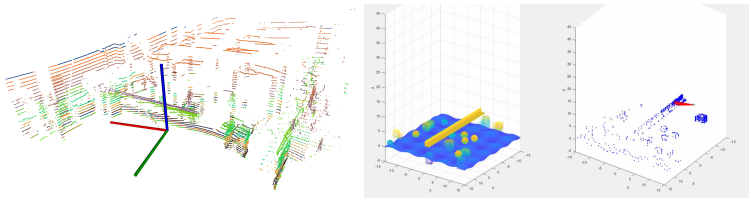


Figure 3.2: Comparison of simulated and real HDL-32E scan data.

beyond this one.

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

3.3.6 Initial Mesh Exploration

3.3.7 Optimization Steps

3.3.8 Final Mesh Verification

4. Results

4.1 Simulated Data: Zero Noise

The first iteration of our 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 [[insert number of points]], average point to point distance is [[insert point to point distance]], and beam point-to-point distance is [[insert beam point to point distance]]. Beam dimensions are 3.048m (10ft) with a cross-section of the following specifications:

Width: 12.7cm (5in)

Height: 12.7cm (5in)

Thickness: 0.635cm (0.25in)

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 we apply a filter 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

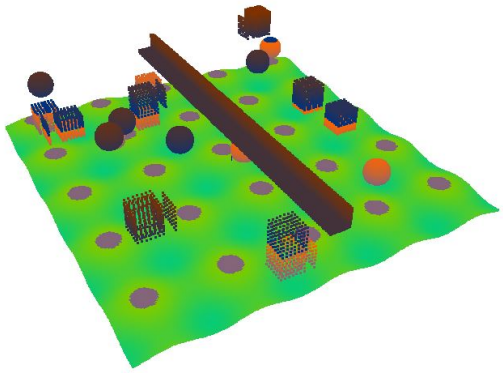


Figure 4.1: Simulated laboratory point cloud data with zero noise induced.

squared.

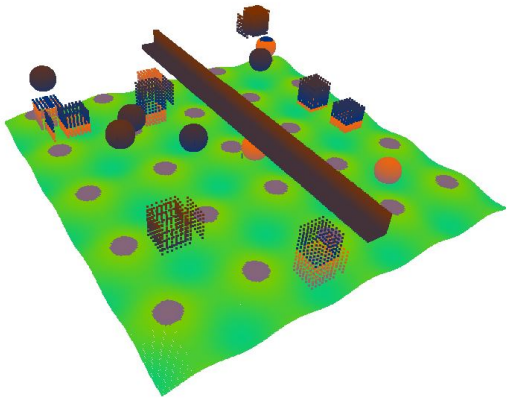


Figure 4.2: Zero noise simulated data after being filtered with voxel size of $0.1in^3$, 15 minimum point neighbors at a distance of 1 standard deviation from the mean point to point distance.

The resulting cloud size is [[insert image 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, we wish to set a higher value to encapsulate gaps in our surfaces which we will encounter in the non-ideal cases. Setting the distance to 0.5 inches proves to have satisfactory results.

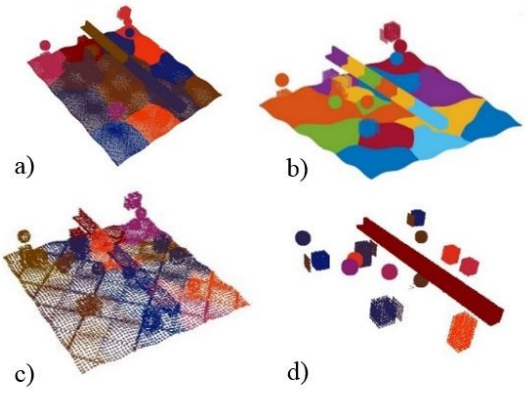


Figure 4.3: Resulting cloud cluster from a) k-means, 17 centroid b) Fuzzy c-means, 17 centroids c) Agglomerative clustering, 17 bins and d) Euclidean clustering with radius of 0.5 in and a maximum cluster size of 50,000 points.

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 the segments we do not wish to see, criteria on the minimum and

maximum cloudsize can be imposed. In Figure 4.4 we impose a minimum cloud size of 3,000 points, and a maximum of 50,000.

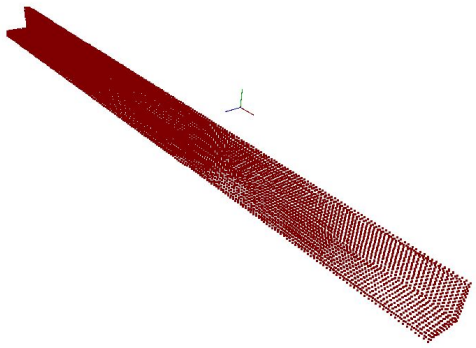


Figure 4.4: Euclidean clustering with radius of 0.5 in, minimum cluster size of 3,000 points and a maximum cluster size of 50,000 points.

The filtering and segmentation parameters derived from this zero noise situation are adequate for isolating our beam from the rest of the point cloud. We will now evaluate the steps in the meshing technique.

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 we will see when we introduce noise that Advancing Front quickly fails to produce a reliable initial mesh for successful optimization.

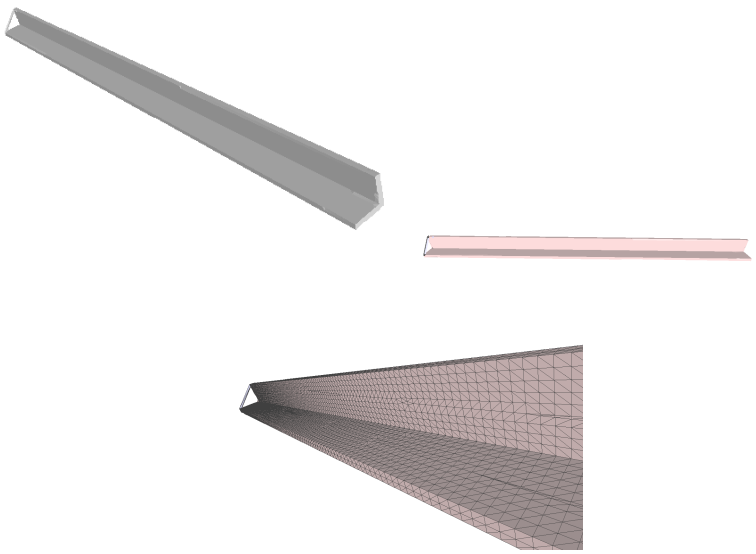


Figure 4.5: Result of initial meshing using the advancing front method.

In Figure 4.6 we see 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 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.

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, our 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

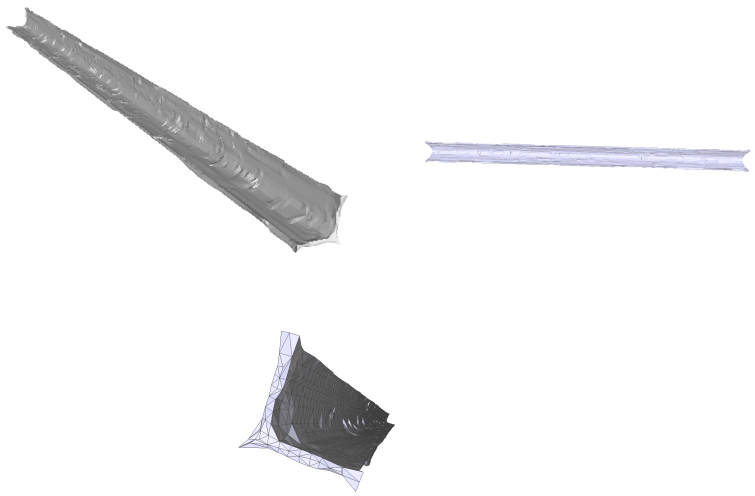


Figure 4.6: Result of initial meshing using the scale space reconstruction method with $S = 2$.

curve.

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 our beam to as high a scale as shown here is not viable for initial meshing.

In the zero noise case, Advancing Front reconstruction proves to be the most accurate to our beam's dimensions. However, as we will see 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,

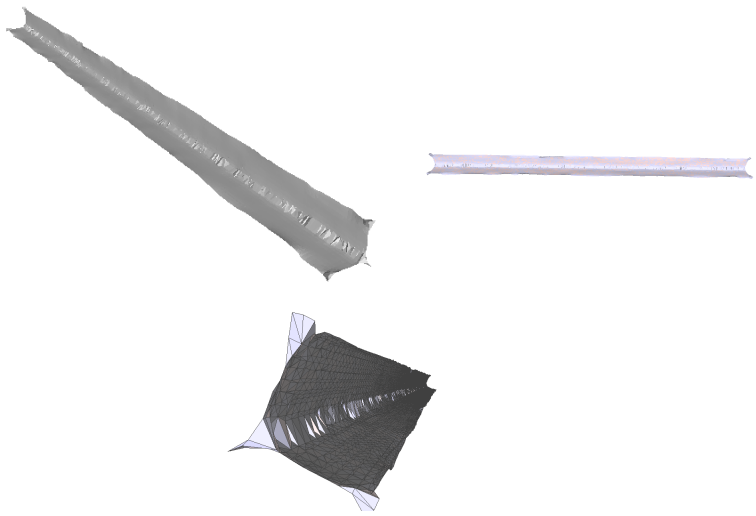


Figure 4.7: Result of initial meshing using the scale space reconstruction method with $S = 4$.

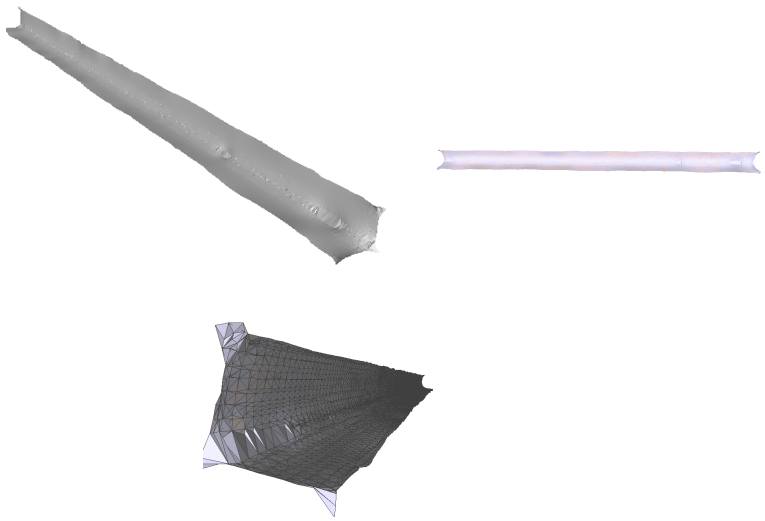


Figure 4.8: Result of initial meshing using the scale space reconstruction method with $S = 15$.

and therefor directly represents the shape defined by the input cloud.

4.1.3 Optimization

Effects Voronoi Relaxation

Voronoi relaxation, defined in Section 2.4.4, 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 2.3, with the exception of volume ratio, which can be visualized as slivers in the mesh (Figure 2.15).

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

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

Voronoi Relaxation has interesting effects of our scale space meshes. In Figure 4.10, we see a relatively smooth, pock-marked version of our scale-space mesh. Notice on the left-most corner of our beam there are a few tall, narrow shapes. These

shapes are the slivers remaining in the mesh. The next steps in the optimization process aim to remove these slivers.

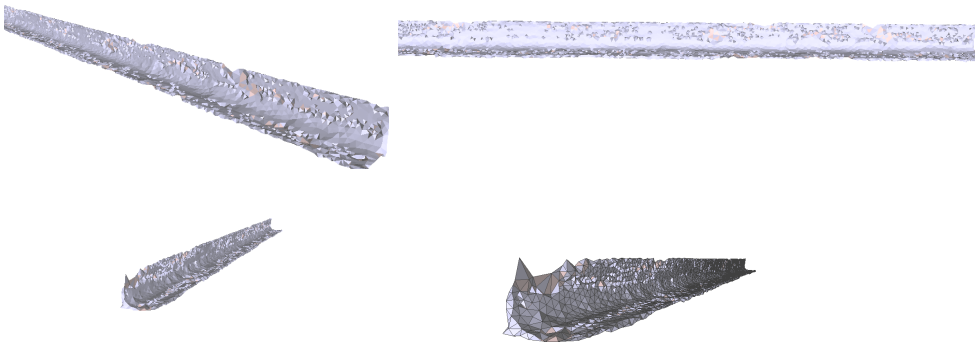


Figure 4.10: Scale space reconstruction $S = 2$ after 200 iterations of voronoi relaxation.

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

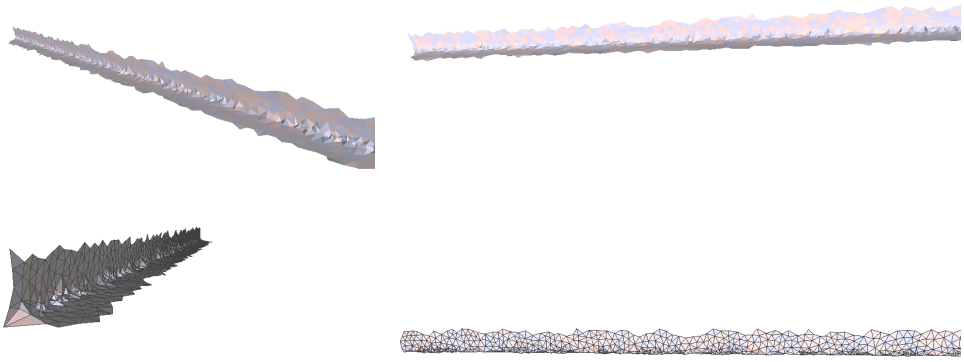


Figure 4.11: Scale space reconstruction $S = 4$ after 200 iterations of voronoi relaxation.

Effects of Delaunay Optimization

Effects of Mesh Perturbation

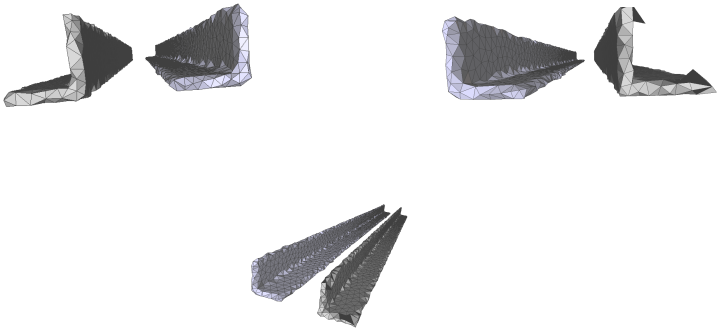


Figure 4.12: Comparison of mesh before and after perturbation.

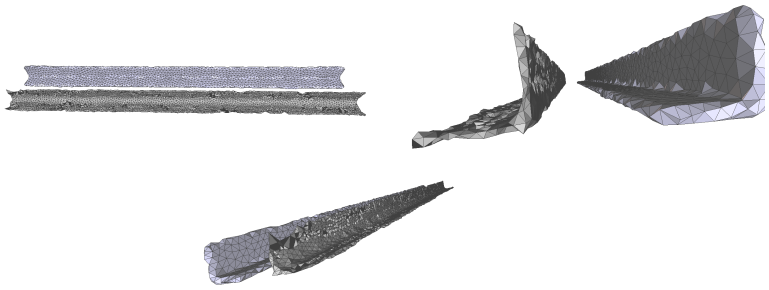


Figure 4.13: Scale space reconstruction $S = 2$ after 200 iterations of voronoi relaxation and with perturbation.

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

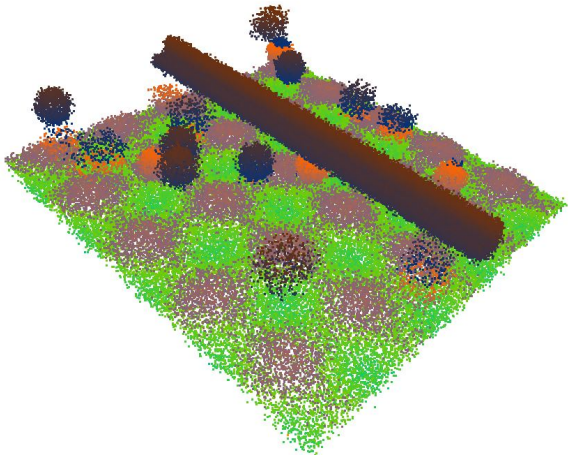


Figure 4.14: Simulated point cloud with 2cm magnitude gaussian noise induced.

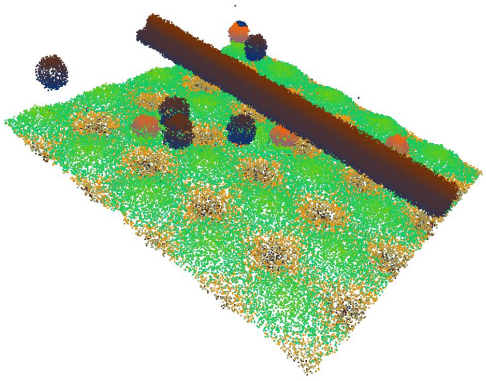


Figure 4.15: 2 cm gaussian noise simulated data after being filtered with voxel size of $0.1in^3$, 15 minimum point neighbors at a distance of 1 standard deviation from the mean point to point distance.

4.2.1 Segmentation Method Comparison

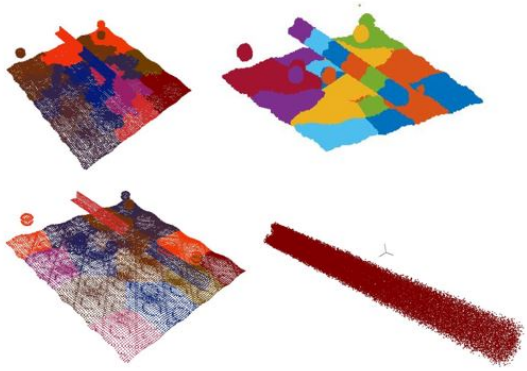


Figure 4.16: Resulting cloud cluster from a) k-means, 17 centroid b) Fuzzy c-means, 17 centroids c) Agglomerative clustering, 17 bins and d) Euclidean clustering with radius of 0.5 cm minimum cluster size of 3,000 and maximum cluster size of 50,000 points.

4.2.2 Initial Mesh Methods

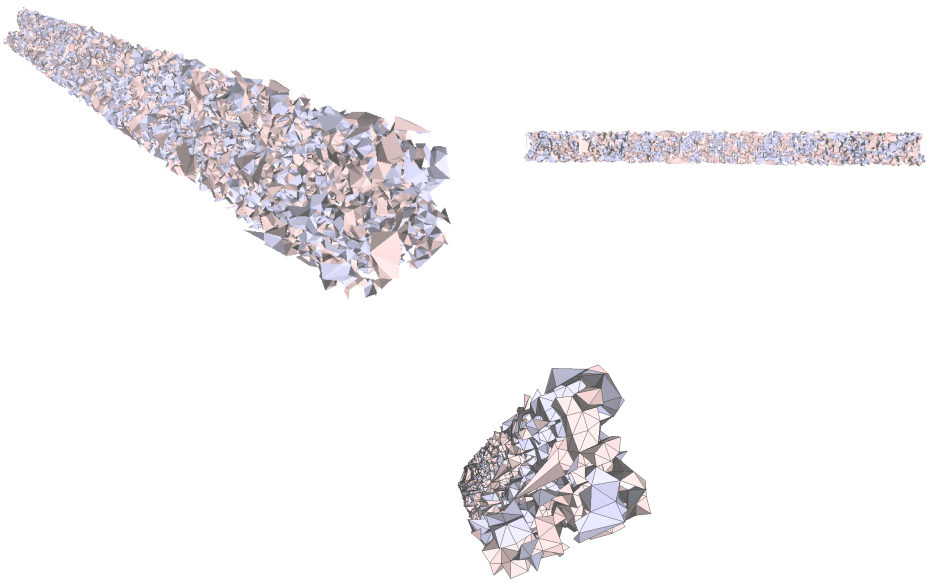


Figure 4.17: Result of initial meshing using the advancing front method.

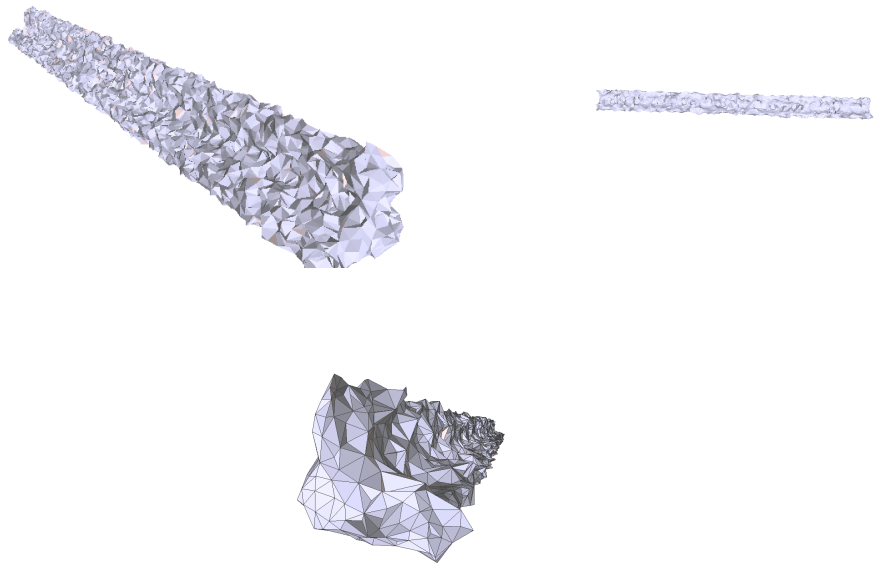


Figure 4.18: Result of initial meshing using the scale space reconstruction method with $S = 2$.

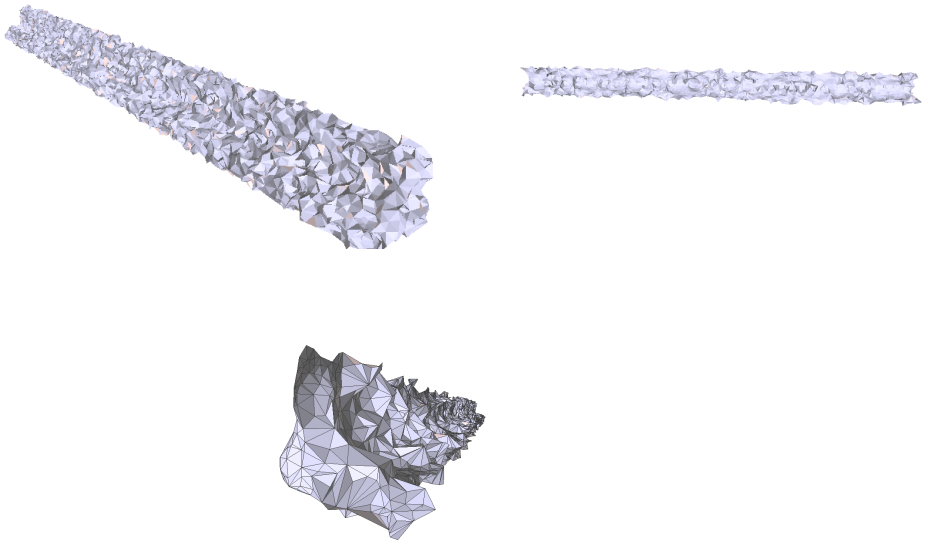


Figure 4.19: Result of initial meshing using the scale space reconstruction method with $S = 4$.

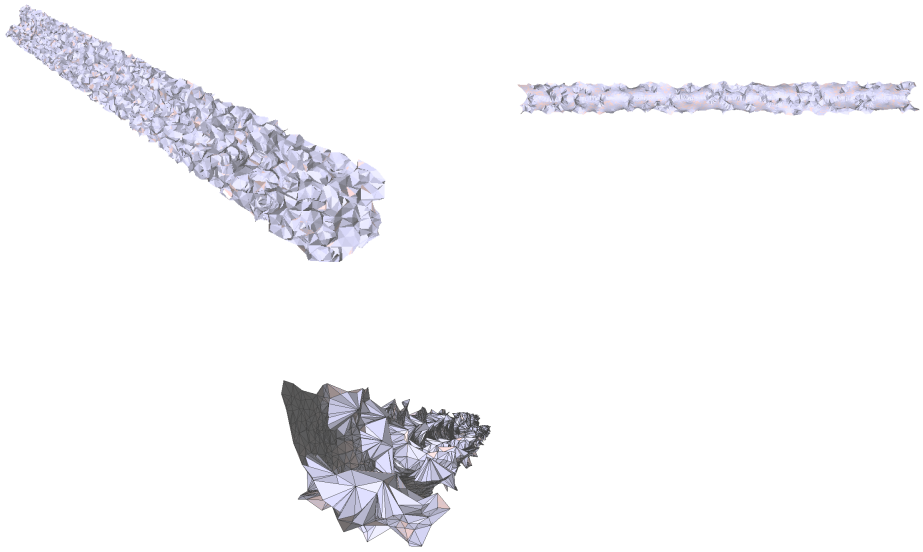


Figure 4.20: Result of initial meshing using the scale space reconstruction method with $S = 15$.

4.2.3 Optimization

Effects of Voronoi Relaxation

Due to the unbounded nature of the surface mesh generated via the Advancing Front method, shown in Figure 4.17, the criteria for the relaxation algorithm to converge is not met. The vertex points expand into n-dimensional space indefinitely until the algorithm crashes.

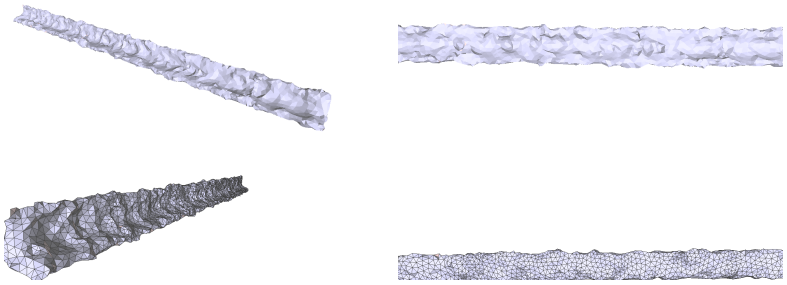


Figure 4.21: Scale space reconstruction $S = 2$ after 200 iterations of voronoi relaxation.

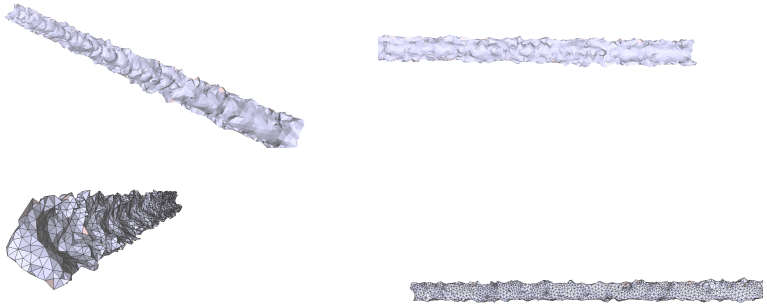


Figure 4.22: Scale space reconstruction $S = 4$ after 200 iterations of voronoi relaxation.

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.

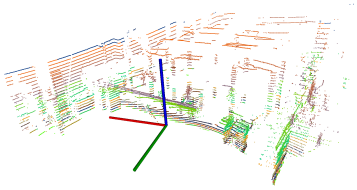


Figure 4.23: Individual frame from LiDAR scan. Image contains 32,000 points.

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 4.24 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 4.25. The result of the concatenation and filtering is sufficient to move on to the segmentation step. It is hard to see in the full concatenation, but the effects of concatenating

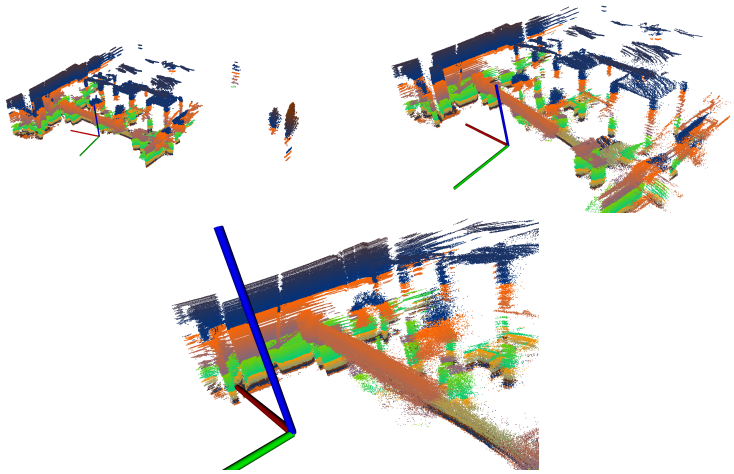


Figure 4.24: Concatenation of 100 individual LiDAR scan frames.

multiple frames without using instrument location/pose data becomes apparent when isolating the beam. We will see these effects in the next section.

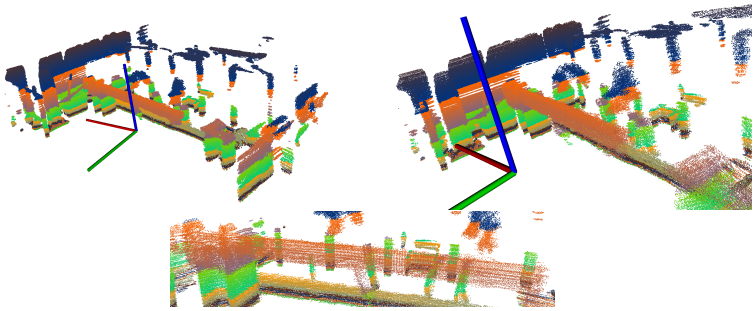


Figure 4.25: Concatenated LiDAR scans after noise filtering and down sampling.

4.3.1 Segmentation Method Comparison

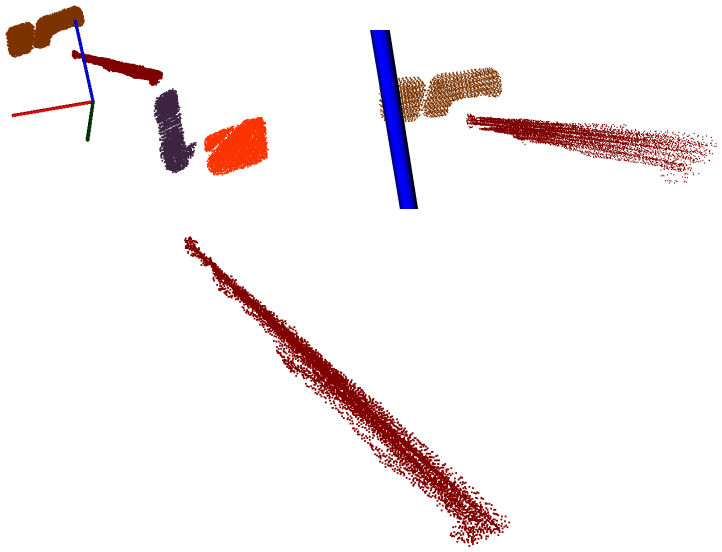


Figure 4.26: Results after euclidean clustering.

4.3.2 Initial Meshing Methods

The effects of the collection path and method are apparent in Figure 4.26. 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 4.27 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.

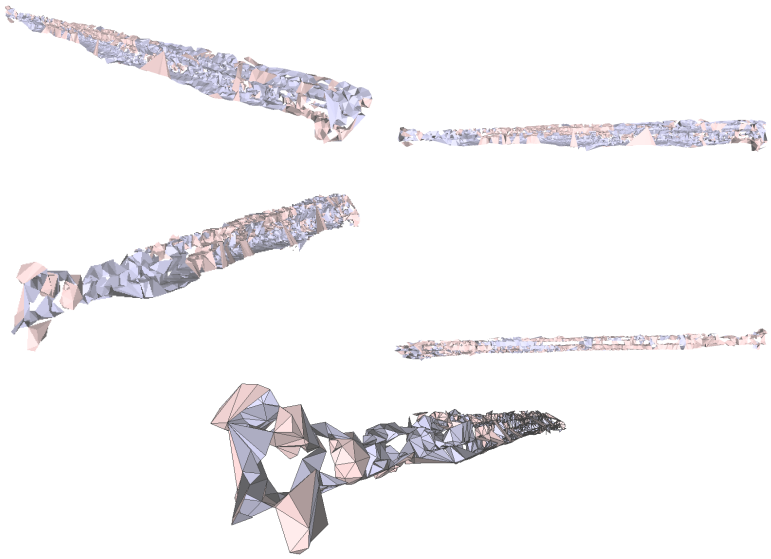


Figure 4.27: Advancing front reconstruction of resulting beam point cloud.

Figure 4.28 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.

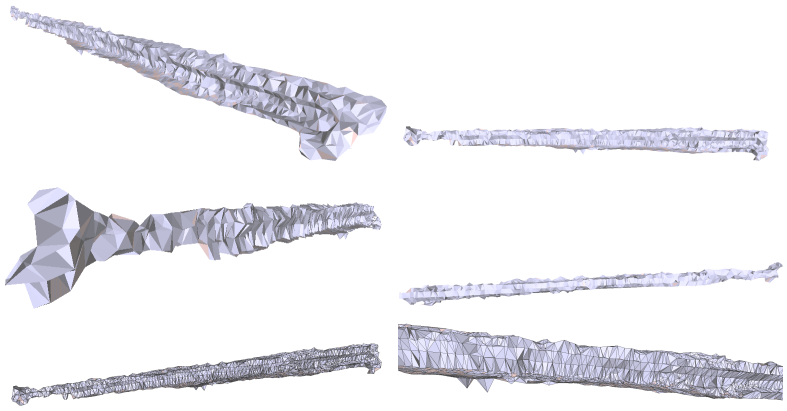


Figure 4.28: Scale space reconstruction at scale S=2 of resulting beam point cloud.

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.

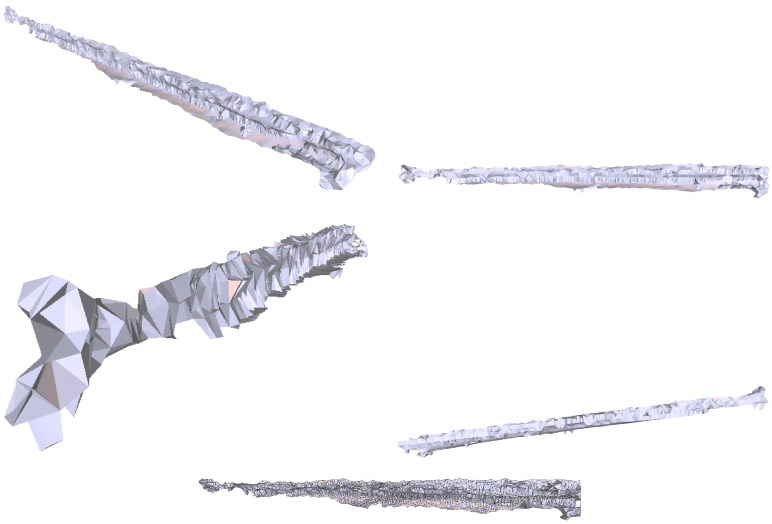


Figure 4.29: Scale space reconstruction at scale S=4 of resulting beam point cloud.

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 LLoyd Optimization

As in previous sections, the first step in the optimization process is Lloyd optimization – otherwise known as voronoi relaxation. This is the last step we will incorporate the advancing front mesh, as it is clear this method is not optimal for non-ideal point clouds. Figure 4.30 shows the results of lloyd optimization on the advancing front mesh.

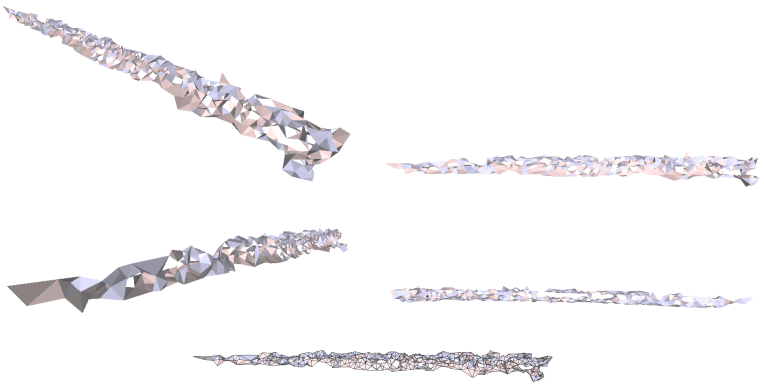


Figure 4.30: Lloyd + Advancing front reconstruction of resulting beam point cloud.

The effect of Lloyd optimization on a mesh is the redistribution of surface area throughout the mesh triangulations. The exit conditions for relaxation is convergence of mesh centroids. Figure 4.31 demonstrates what a lloyd optimization does for the LiDAR collected point cloud after being reconstructed at a scale space of 2.

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

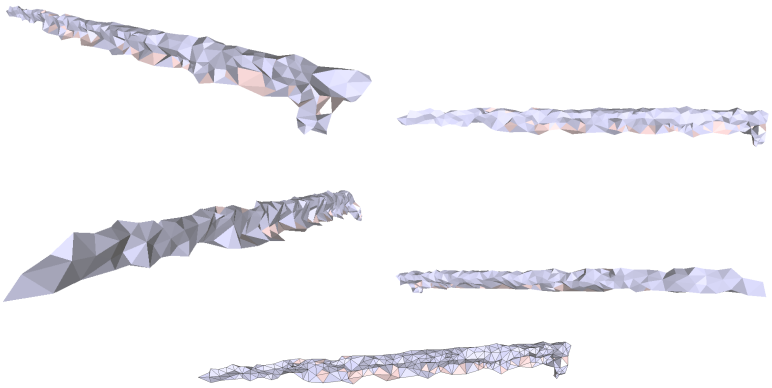


Figure 4.31: Lloyd + Scale space reconstruction at S=2 of resulting beam point cloud.

Figure 4.32.

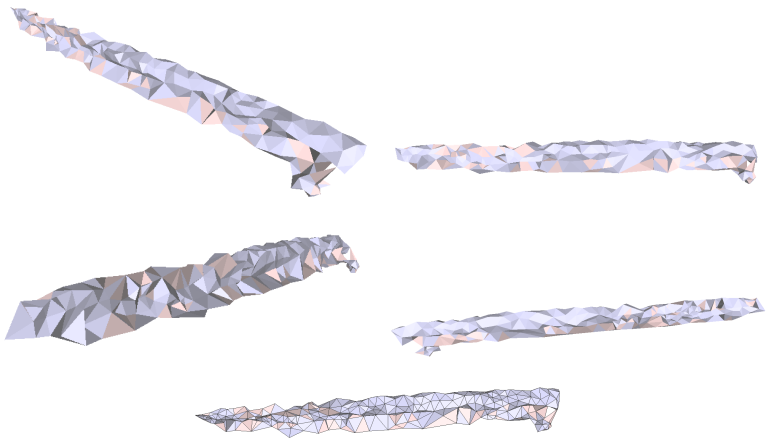


Figure 4.32: Lloyd + Scale space reconstruction at S=4 of resulting beam point cloud.

Optimized Delaunay Triangulation

Figure 4.33: Lloyd + ODT + Scale space reconstruction at S=2 of resulting beam point cloud.

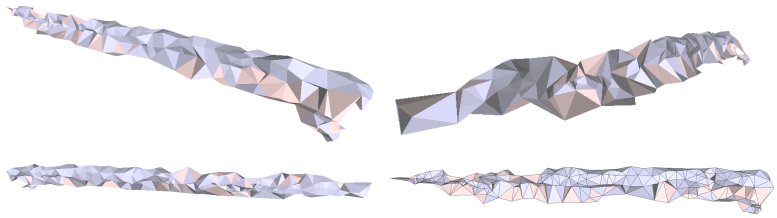


Figure 4.34: Lloyd + ODT + Scale space reconstruction at S=4 of resulting beam point cloud.

Perturbation

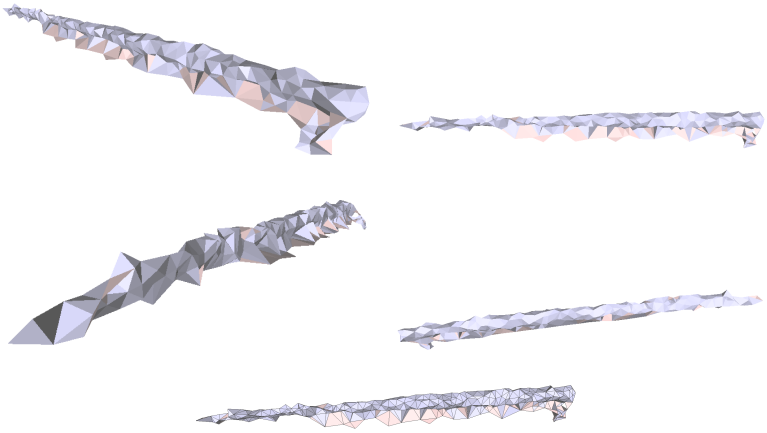


Figure 4.35: Lloyd + ODT + Perturb + Scale space reconstruction at S=2 of resulting beam point cloud.

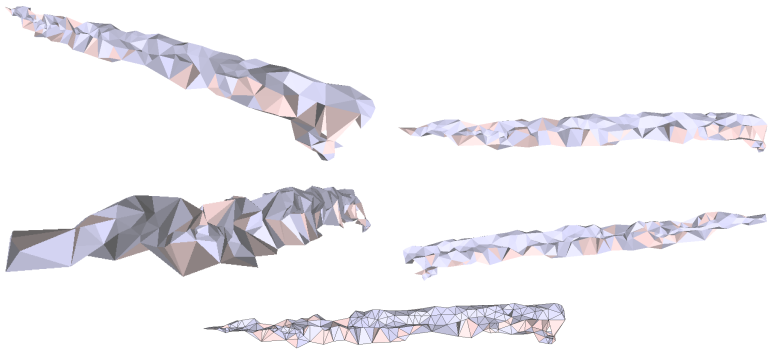


Figure 4.36: Lloyd + ODT + perturb + Scale space reconstruction at S=4 of resulting beam point cloud.

Exudation

5. Conclusion

6. Future Work

Bibliography

[1] J. L. Lerma, S. Navarro, M. Cabrelles, and V. Villaverde, "Terrestrial laser scanning and close range photogrammetry for 3d archaeological documentation: the upper palaeolithic cave of parpalló as a case study," *Journal of Archaeological Science*, vol. 37, no. 3, pp. 499–507, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0305440309003781

[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. D'Altri, G. Bitelli, I. Selvaggi, and A. Lambertini, "From laser scanning to finite element analysis of complex buildings by using a semi-automatic procedure," *Sensors*, vol. 15, 2015.

[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] 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.

[10] 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.

[11] 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.

[12] H. Jing and Y. Suya, "Point cloud labeling using 3d convolutional neural network," in *2016 23rd International Conference on Pattern Recognition (ICPR)*, Conference Proceedings, pp. 2670–2675.

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

[14] N. P. Weatherill and O. Hassan, "Efficient three-dimensional delaunay triangulation with automatic point creation and imposed boundary constraints," *International Journal for Numerical Methods in Engineering*, vol. 37, no. 12, pp. 2005–2039, 1994. [Online]. Available: http:https://doi.org/10.1002/nme.1620371203

[15] 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

[16] 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.

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

[18] J. Brandts, S. Korotov, and M. Krizek, *A Geometric Toolbox for Tetrahedral Finite Element Partitions*. Institute of Mathematics, ELTE University, Hungary: Bentham Books, 2011, vol. 1, book section 6.

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

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

[21] 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.