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

List of Tables

List of Figures

Abstract
Automated Conversion of 3D Point Clouds to FEA Compatible Meshes
Matthew S. Brown
Antonios Kontsos, Ph. D.

This paper outlines a method to utilize machine learning in conjunction with advanced meshing techniques to autonomously segment raw point cloud data and reconstruct the resulting segments into simply connected volumetric meshes.

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, Jos 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 subsec:UAVscanning

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

[8]

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

2.2 Sensor Fusion

2.3 Point Cloud Pre-processing Methods

2.3.1 Registration

Position Data

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:

- An intrinsic reference frame

- A highly discriminative feature vector encoding the 3D shape characteristics

Intrinsic Reference Frame Calculation

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}:

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

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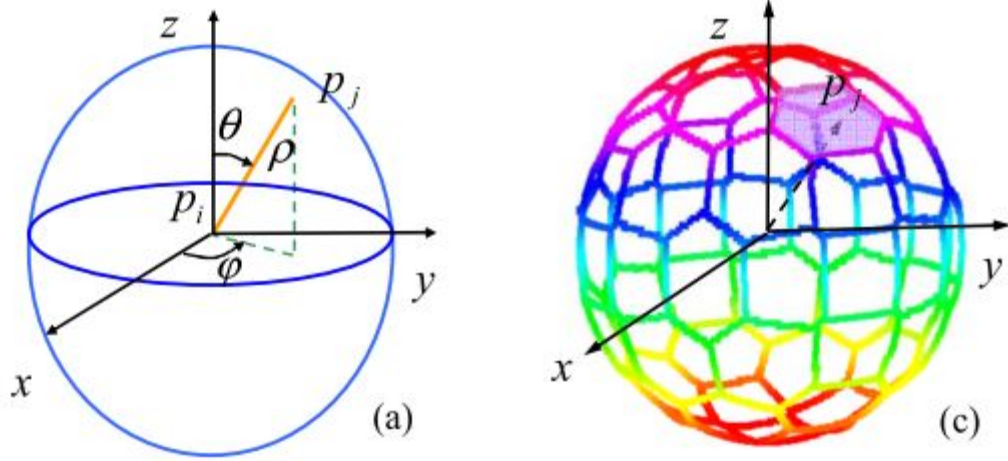
Figure 2.1: Feature vector calculation via spherical bin decomposition [9].

2.3.2 Flitering

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.

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

2.3.3 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
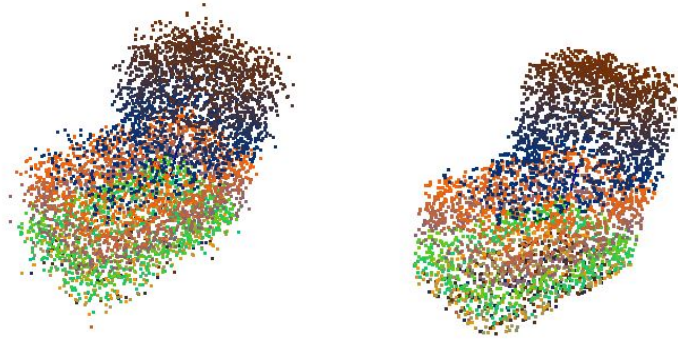
Figure 2.2: The effects of noise filtering on a simulated L block with 10% induced noise.

voxel.

2.3.4 Dealing with 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.

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,

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

Figure 2.4: Section of a building occluded by an object in the foreground [10].
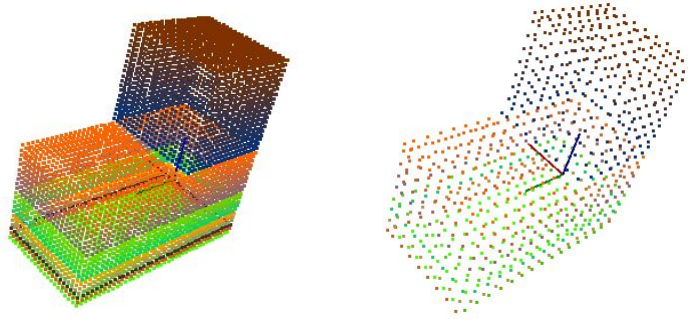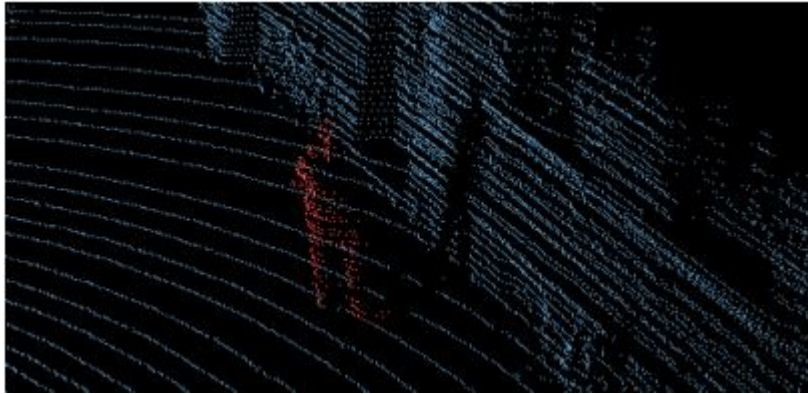
trees, bushes, and a slew of other objects that may come between the scanner and the desired object. Biasuttia et. al. propose a way to cast these interruptions to the surface they desire to map by converting the xyz point cloud to a range image — a three parameter map of distance r from the device plotted 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.

$$C_s^i = \frac{\sum_{b \in C_s^i} b \times h_s(b)}{\sum_{b \in C_s^i} h_s(b)} \tag{2.4}$$

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

$$d(C_s^i, C_r^j) = |C_s^i - C_r^j| \tag{2.5}$$

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.

$$\begin{cases} \frac{\partial u}{\partial t} - \Delta u = 0 \in \Omega \times (0, T) \\ u(0, x) = u_0(0) \in \Omega \end{cases} \tag{2.6}$$

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:

$$C = \sum_{i=0}^{N} p(i) - p_{proj}(i) \tag{2.7}$$

Where N represents the number of points in the set, $p(i)$ represents the ith 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.

$$p_{proj}(i) = arg_{min}\frac{p(i) \bullet l(j)}{||l(j)||} \in p_{normal}(i) \bullet l_{normal}(j) \geq \tau \tag{2.8}$$

$$x_{k+1} = x_k - \frac{C(x_k)}{C'(x_k)} \tag{2.9}$$

Using a Newton Raphson interative regression, shown in equation 2.9 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.
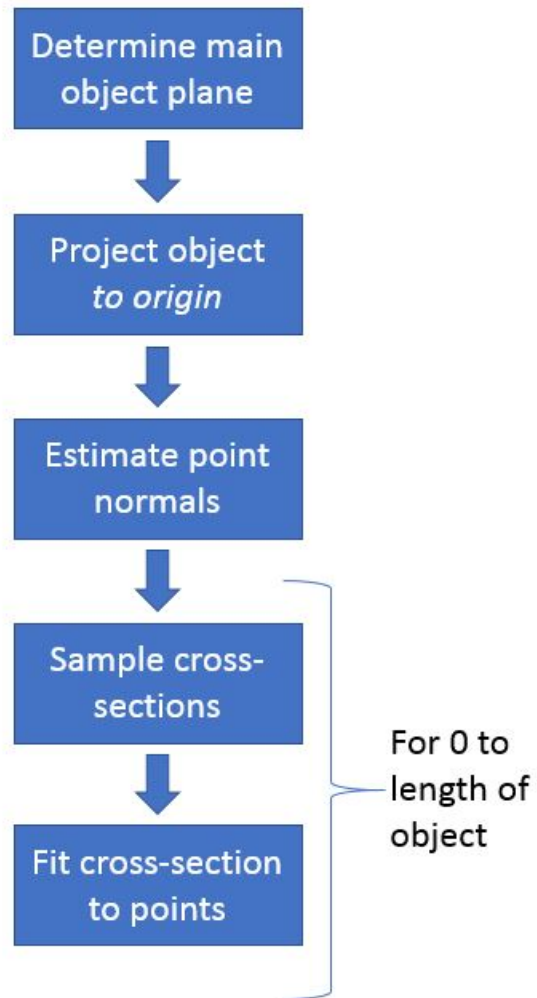
Figure 2.5: A flowchart of the informed shape estimation algorithm.

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.

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 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 ith point and weight of N points located inside the voxel.

$$C_{new} = \sum_{i}^{N} w(i)x_i \tag{2.10}$$

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.

$$C_{new} = \frac{\sum_i^N x_i}{N} \tag{2.11}$$

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:

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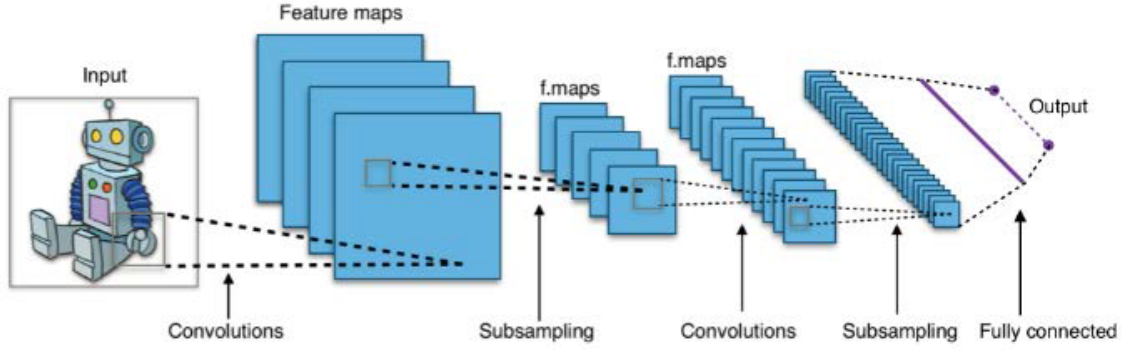
Figure 2.6: Map of a convolutional neural network with two hidden layers [ref]

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.

$$
\begin{bmatrix} C_1 \\ \vdots \\ C_n \end{bmatrix} = \begin{bmatrix} w_{1,1} & \cdots & w_{1,m+1} \\ \vdots & \ddots & \cdots \\ w_{n,1} & \cdots & w_{n,m+1} \end{bmatrix} \begin{bmatrix} f_1 \\ \vdots \\ f_n \\ 1 \end{bmatrix} \tag{2.12}
$$

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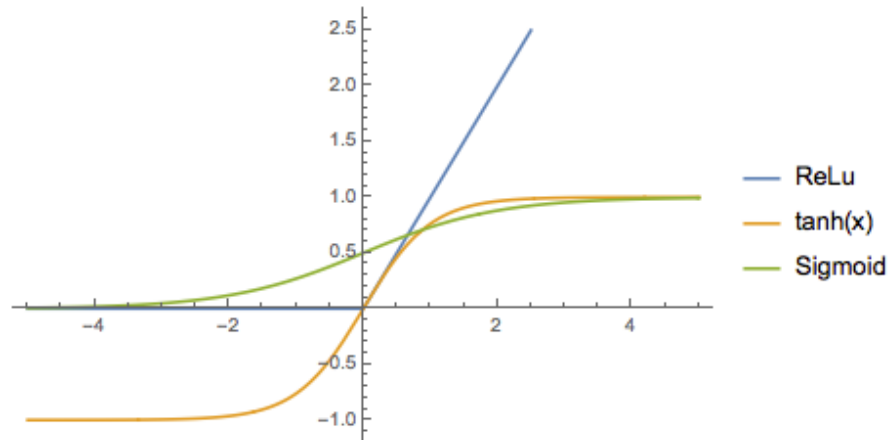
Figure 2.7: Visualization of commonly used non-linear activation functions
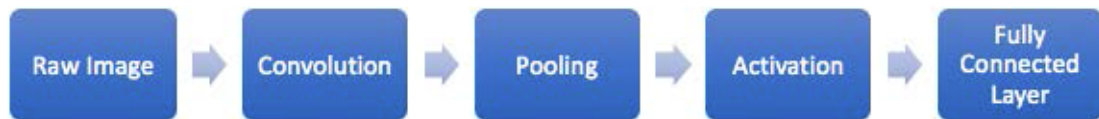

Figure 2.8: Block diagram of a CNN′s skeleton structure

ones utilized in this paper incorporate a de-linearizing element dubbed the activation function. The activation function applies a nonlinear operation to the values in the convolution layer, which allows for the CNN to become a very powerful nonlinear fit function. Typical activation functions include the hyperbolic tangent function, the sigmoid function, and most popularly the rectifier function. Each of these functions are show in the figure below:

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

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

$$x_{k+1} = x_k - \gamma \nabla F(x_k) \tag{2.13}$$

$$F(x) = \begin{bmatrix} C_1 \\ \vdots \\ C_n \end{bmatrix} - \begin{bmatrix} w_{1,1} & \ldots & w_{1,m+1} \\ \vdots & \ddots & \ldots \\ w_{n,1} & \ldots & w_{n,m+1} \end{bmatrix} \begin{bmatrix} f_1 \\ \vdots \\ f_n \\ 1 \end{bmatrix} \tag{2.14}$$

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].

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.

Methods Used

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}.

$$argmin_S \sum i = 1^k \sum x \in S_i |x - c_i|^2 \tag{2.15}$$

[PROS, CONS, AND GENERAL USAGE CASES]

Fuzzy C-means Clustering

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

$$c_k = \frac{\sum x w_k(x)^m x}{\sum x w_k(x)^m} \tag{2.16}$$

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

$$w_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{|x_i - c_j|}{|x_i - c_k|} \right)^{\frac{2}{m-1}}} \tag{2.17}$$

[PROS, CONS, AND GENERAL USAGE CASES]

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.

Hierarchical Clustering
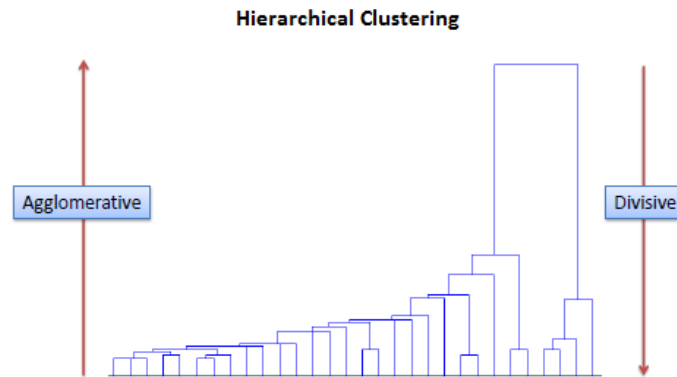
Figure 2.9: Comparison of divisive and agglomerative hierarchical clustering

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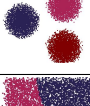
Perhaps the simplest of the algorithms listed above, Euclidean distance clustering operates on the pretense that objects are separated significantly enough spatially from another that clustering points based on their proximity to other points in the cloud is sufficient to properly segment the dataset.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

Comparison of Methods

2.4.3 Converting a Discrete Point Cloud to a Bounded Area Surface Mesh

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

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

K-means	Fuzzy C-means	Agglomerative	Divisive	Euclidean

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.

$$V_k = \{P_i \mid |p - P_i| < |p - P_j|, \forall j \neq i\} \qquad (2.18)$$

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
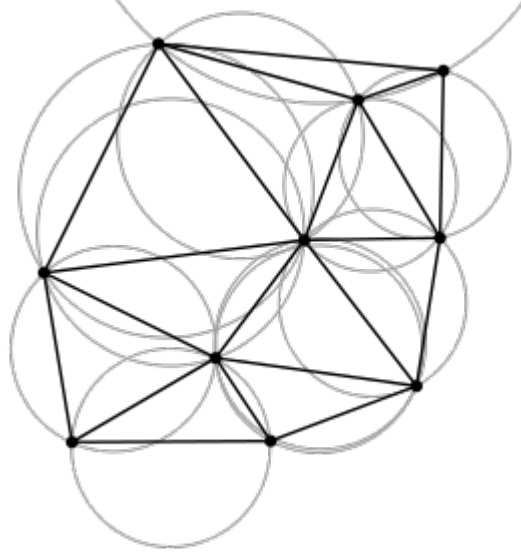
Figure 2.10: Visual represention of a 2-dimensional delaunay triangulation process triangulation [13].

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. [DEFINITION, USAGE, AND 2D EXAMPLE] [7].

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

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
```

Algorithm 3: Back Projection to Final Mesh

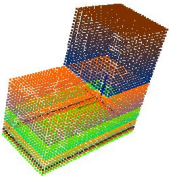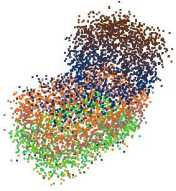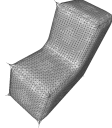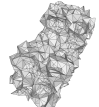**Input:** A point set P, a number of iterations N, and a radius r
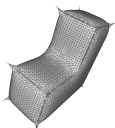
Output: A modified point set P_N

Hole Patching, Fairing, and 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.

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		

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
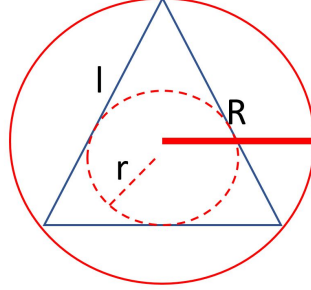
Figure 2.11: Definitions for triangulation quality measures. R = circumsphere radius, r = inscribed sphere radius, l = edge length.

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	$Q_5 = \frac{V}{l^3_{max}}$

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.

Low quality values in mesh return problematic polyhedrons for volumetric conversion, typically looking like those shown in figure 2.12.

Voronoi Relaxation / Lloyd's Algorithm

Voronoi relaxation operates on the same principle as k-means clustering. At each iteration, the centroid of each Voronoi region is calculated, and the concurrent vertex is moved to the centroidal location. At convergence, the resulting mesh is uniform

SKINNY TETRAHEDRA

spire / needle   splinter   spindle   spear   spike

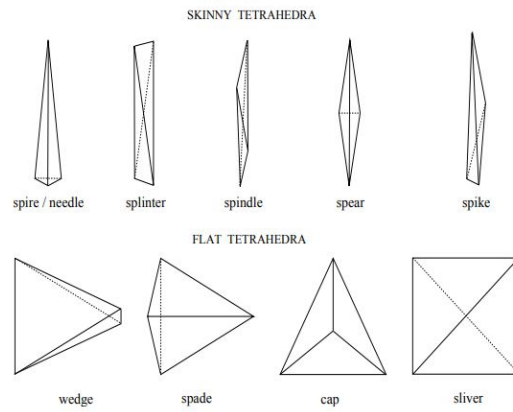FLAT TETRAHEDRA

wedge   spade   cap   sliver

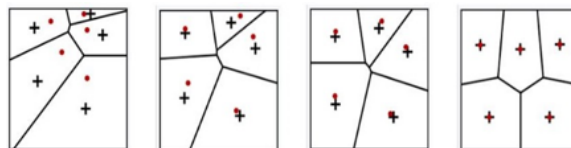Figure 2.12: Common examples of poor quality tetrahedra

Figure 2.13: From right to left: Voronoi relaxation over 1 iteration, 5 iterations, 10 iterations, and 15 iterations

in tetrahedron/triangulation size. Voronoi relaxation can modify a shapes topology significant due to its heavy smoothing capabilities [14].

Optimal Delaunay Triangulation

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 [5].

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 [6].

3. Hypothesis Statement

4. Technical Approach

5. Results

5.1   Simulated Data: Zero Noise

5.1.1   Segmentation Method Comparison

5.1.2   Initial Meshing Methods

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

5.2.1   Segmentation Method Comparison

5.2.2   Initial Mesh Methods

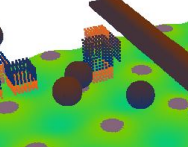Figure 5.1: Simulated laboratory point cloud data with zero noise induced.

Figure 5.2: Zero noise simulated data after being filtered with voxel size of $0.1cm^3$, 15 minimum point neighbors at a distance of 1 standard deviation from the mean point to point distance.
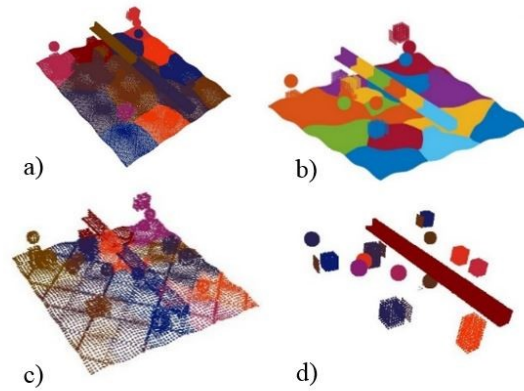
Figure 5.3: Resulting cloud cluster from a) k-means, 17 centroid b) Fuzzy c-means, 17 centroids c) Agglomerative clustering, 17 bins and d) Euclidean clustering with radius of 0.5 cm and a maximum cluster size of 50,000 points.
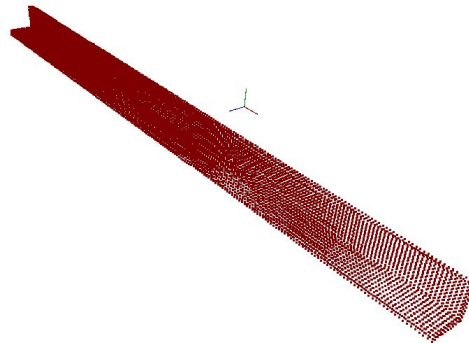

Figure 5.4: Euclidean clustering with radius of 0.5 cm, minimum cluster size of 3,000 points and a maximum cluster size of 50,000 points.
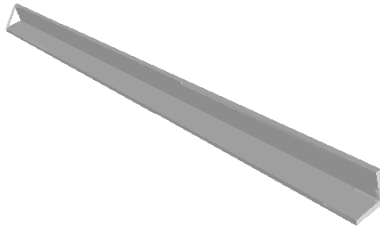
Figure 5.5: Result of initial meshing using the advancing front method.

Figure 5.6: Result of initial meshing using the scale space reconstruction method with $S = 2$.
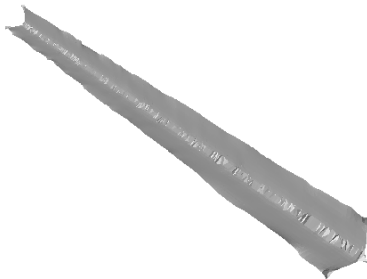

Figure 5.7: Result of initial meshing using the scale space reconstruction method with $S = 4$.
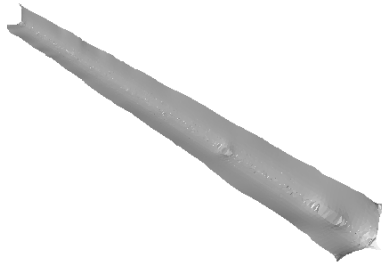
Figure 5.8: Result of initial meshing using the scale space reconstruction method with $S = 15$.
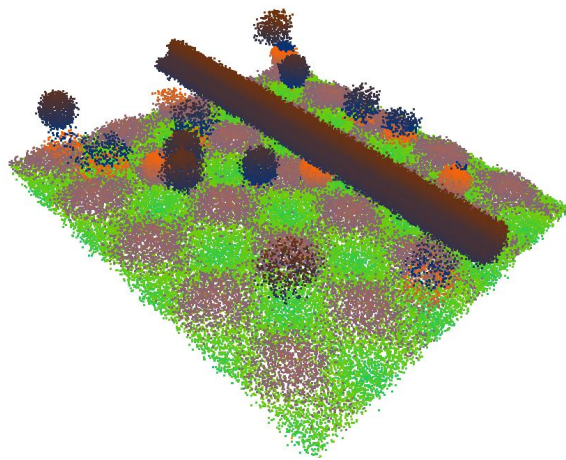

Figure 5.9: Simulated point cloud with 2cm magnitude gaussian noise induced.
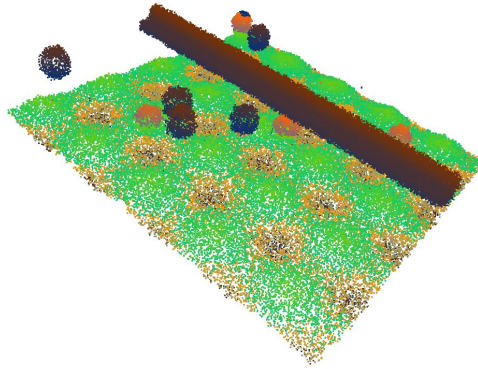
Figure 5.10: 2 cm gaussian noise simulated data after being filtered with voxel size of $0.1cm^3$, 15 minimum point neighbors at a distance of 1 standard deviation from the mean point to point distance.
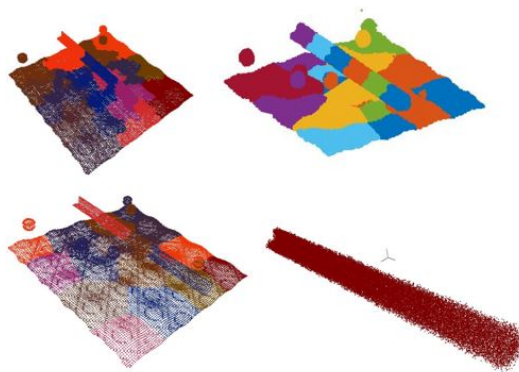

Figure 5.11: Resulting cloud cluster from a) k-means, 17 centroid b) Fuzzy c-means, 17 centroids c) Agglomerative clustering, 17 bins and d) Euclidean clustering with radius of 0.5 cm minimum cluster size of 3,000 and maximum cluster size of 50,000 points.

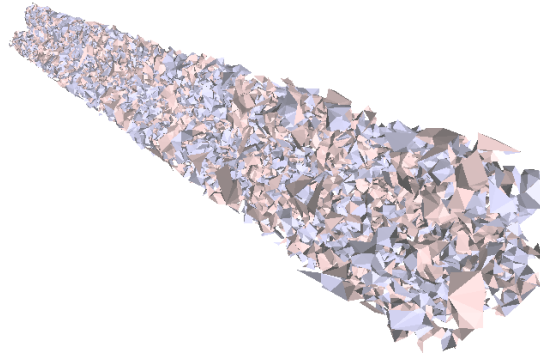Figure 5.12: Result of initial meshing using the advancing front method.
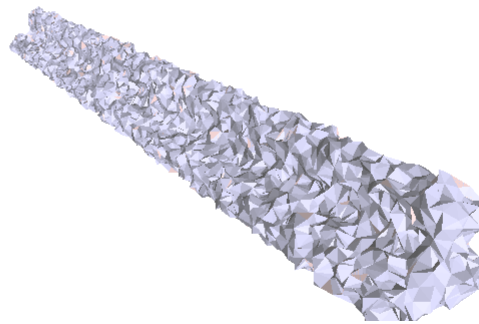

Figure 5.13: Result of initial meshing using the scale space reconstruction method with $S = 2$.
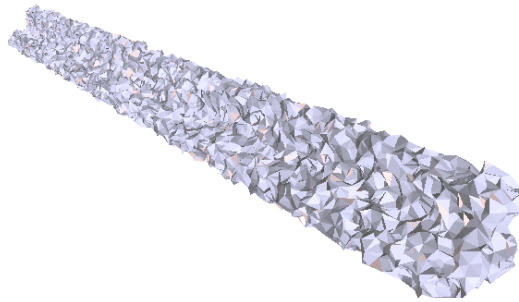
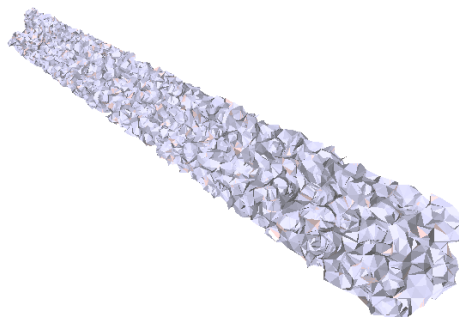Figure 5.14: Result of initial meshing using the scale space reconstruction method with $S = 4$.

Figure 5.15: Result of initial meshing using the scale space reconstruction method with $S = 15$.

6. Conclusion

7. 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 parpall 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. DAltri, G. Bitelli, I. Selvaggi, and A. Lambertini, "From laser scanning to finite element analysis of complex buildings by using a semiautomatic 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] 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

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