Verification of As-Built Footing Geometry Using LiDAR Point Cloud Data

McGill University

MECH 309

Lucas Bessai	261050265
Aidan Kimberley	261004905

Introduction

In the construction industry, verifying that as-built structures align with their design specifications is critical for ensuring structural integrity and project quality. This problem often necessitates comparing design geometries to as-built geometries derived from LiDAR point cloud data. LiDAR scans provide detailed 3D representations of constructed components, capturing many spatial data points. However, the inherent noise and density of such data make it challenging to extract critical geometric features, such as planes and corner points, necessary for verification of the construction.

In the problem explored in this report, CEF and FCC have disagreements over the building quality of a concrete footing. They desire verification of whether the structure was built within the tolerances specified by the design drawings. We are a third party that has been tasked with finding the answer.

This report details the algorithm we developed to assess point could data and compare it to design geometry. Using SVD's, a modified K-means algorithm, and a series of plane intersection Ax=b problems, we created an algorithm that can take point cloud data and design geometry as inputs and 'learn' the planes and corner points that make up the as build structure.

After applying this algorithm on the data from the concrete footing in question, we determined that CEF was correct in their assertion that the as built geometry did not match the drawing specifications.

Method

Define Input Geometry:

The first stage of learning the as built geometry of the footing is defining the design geometry as an input. This is necessary for the comparison of the as built footing to the design specifications. This involves inputting the ideal corner points and normal vectors for the walls of the footing. Using the corner points, the centroids of the ideal planes can be found as the average of the corners of a given plane:

$$c = mean(P_1,\ P_2,\ P_3,\ P_4,)$$

The offsets of each ideal plane can then be calculated by taking the dot product of the normal vector and the centroid. This dot product could also be done with any point on the plane to get the same result; however, the centroid is more useful to use as it represents the average position on the plane:

$$d = n^T c$$

K-means Clustering:

With the ideal geometry define the next stage was to cluster the point cloud data into specific planes to compare the geometries. We did this clustering using a modified k-means algorithm. The classical k-means algorithm first takes an input of the desired number of clusters (groups) then takes the data set objects, x, and uses an initial guess of a representative value, z, to assign each x to a cluster. c, is the column matrix that holds the cluster assignments for every x in the dataset. The representative that is closest to x will define its assignment:

$$C_i = \min_{j=1,2,\ldots,k} \left\| x_i - z_j \right\|^2$$

In the above equation, i ranges from 1 to the number of objects in the data set, N, and j ranges from 1 to the number of desired clusters, k. Once each object has been assigned to a cluster, the representatives are redefined to better approximate the data points their respective cluster. Thus, z is redefined as the average data point:

$$z_j = \frac{1}{G_j} \sum_{i \epsilon G_j} x_i$$

Where, in the above equation, G_j, represents the set of data points in the cluster j. This classical k-means process is done iteratively, where the redefined representatives are retested on the dataset, reassigning each data point to the adjusted representative that it is closest to. The iteration stops when the change in the cost function that defines the system is sufficiently small.

$$J(z, C) = \frac{1}{N} \sum_{i=1}^{N} \left\| x_i - z_j \right\|^2$$

In the above equation $J(z, C)$ is the cost function.

The problem with using classical k-means for this geometry clustering problem is that the walls are of different scale. Thus, using a representative, z, that is a position (and thus same dimension as the position data objects, x) would lead to misclassification of points near the edges of the wall intersections on the footing. As such, the classification using classical k-means would only be accurate for a given radius around the initial representative guess. To fix this problem, we modified the k-means algorithm to use a higher dimensional z. With a higher dimensional representative, more information about how the objects relate to the clusters can be carried and, naturally, the edge cases can be sorted effectively.

The alteration to z involved making it a position (three dimensional) as well as a normal (three dimensional). The specific position is the centroid a plane, c. This centroid is accompanied by the normal to that plane, n. Together the centroid and normal define a plane rather than a position and carry six dimensions. Clustering each position data point using the plane representative, x, follows essentially the same process as the classical k-means algorithm:

1. First, we figure out which plane a given object, x_i, is closest to using the dot product of the position with the normal of that plane after it has been centered around the origin. It is important that the plane is centered because otherwise the dot product of the normal with a point exactly on the plane would give the offset of the plane rather than zero. This centered dot product (otherwise known as the residual) is done instead of the squared norm in the classical k-means.
2. Second, we redefine the normal and centroid by fitting a plane to each cluster using an SVD. The mean of the all the positions in a given cluster represents the adjusted centroid, and the last row of the left singular matrix (the direction of least variance of the data) of the cluster data matrix, V^T, is the adjusted normal vector. Compared to classical k-means, the average of the objects in each cluster is taken to find the centroid; however, an additional step of computing the right singular matrix is required to find the normal.
3. Lastly, this process is iterated until the change in the cost function is sufficiently small. The cost function in this case is the average residual of all the points in the dataset. We know

that the points have been classified correctly at this point since the residuals are all very small. The small residuals indicates that planes are sufficiently close to every point. This value can be quantified by taking the norm of the residuals. If this norm is bellow 0.1 the fit is considered strong.

The modified k-means process can effectively illustrate with the following pseudo code:

while $\Delta J(n, c, C) > \epsilon$:

$$C_i = \min_{j=1,2,3} \left(n_j^T (x_i - c_j) \right)$$

$$n_j, c_j = plane\_fit(G_j)$$

$$J(n, c, C) = \frac{\sum_{j=1}^{k=3} \sum_{i=1}^{N} n_j^T (x_i - c_j)}{N}$$

Where, in the above code, $J(n, c, C)$ is the cost function that is dependent on the normal, n, the centroid, c, and the assignment matrix, C. ϵ represents the threshold at which sufficiently low change in the cost function is deemed to have happened. For our code we chose a value of $\epsilon = 0.0001$. The loop converged in nine iterations.

To conclude on clustering, the predefined ideal geometry discussed above was used to make the initial guess of the representatives (normal and centroid for the front, side, and top walls). This is a strong initial guess since it is assumed that the footing was data will closely approximate the ideal geometry if it was built to, or close to specification. This guess is why the clustering converged relatively quickly. To further illustrate the process, the initial interactions give clusters that do not effectively sort the data into the correct walls:

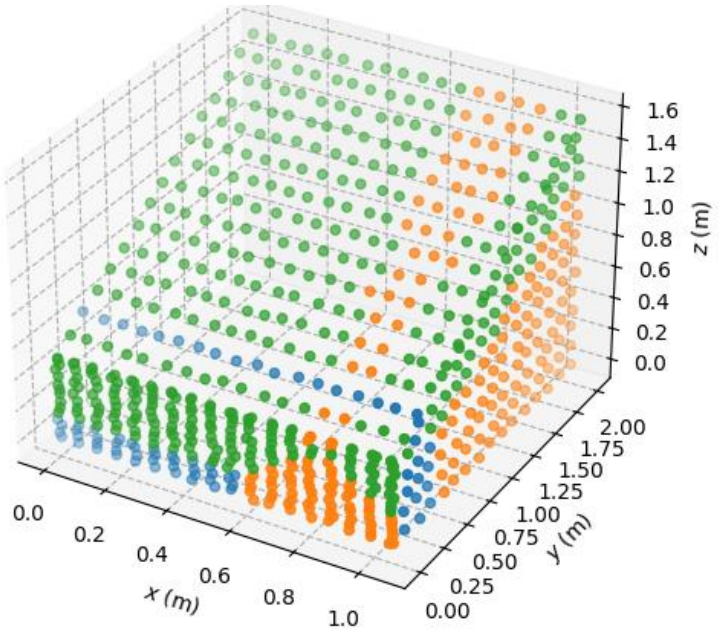

Figure 1 First Iteration of Clustering Algorithm

As the iterations continue, it is clear when we plot the data of each cluster that the position points have been classified correctly.

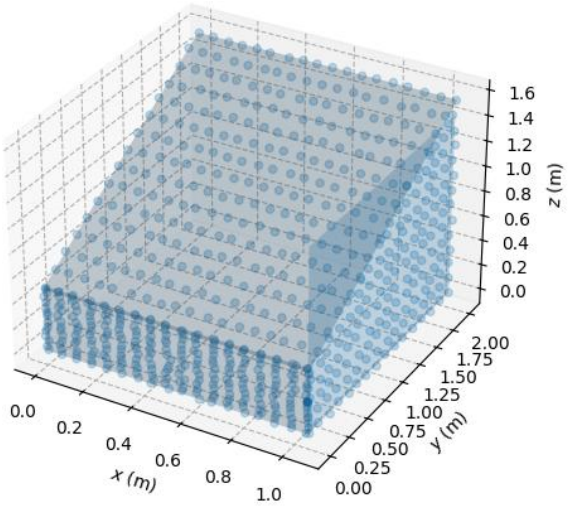

Figure 2 Clustering after sufficient iterations

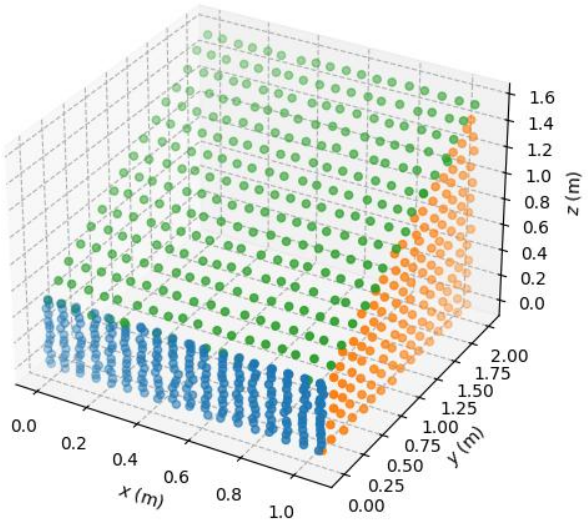

Figure 3 Plane and Points Plot

Outlier Removal:

After all points have been correctly assigned to their respective planes, we iteratively removed outliers using a three time the standard deviation rule, $P < 3\sigma$, for defining an outlier. The three standard deviation bounds are illustrated by the dotted red lines, only points within these bounds are kept. The outlier algorithm converged in two or less iterations for each plane. As is illustrated by the bellow graphs, the planes fit the data well.

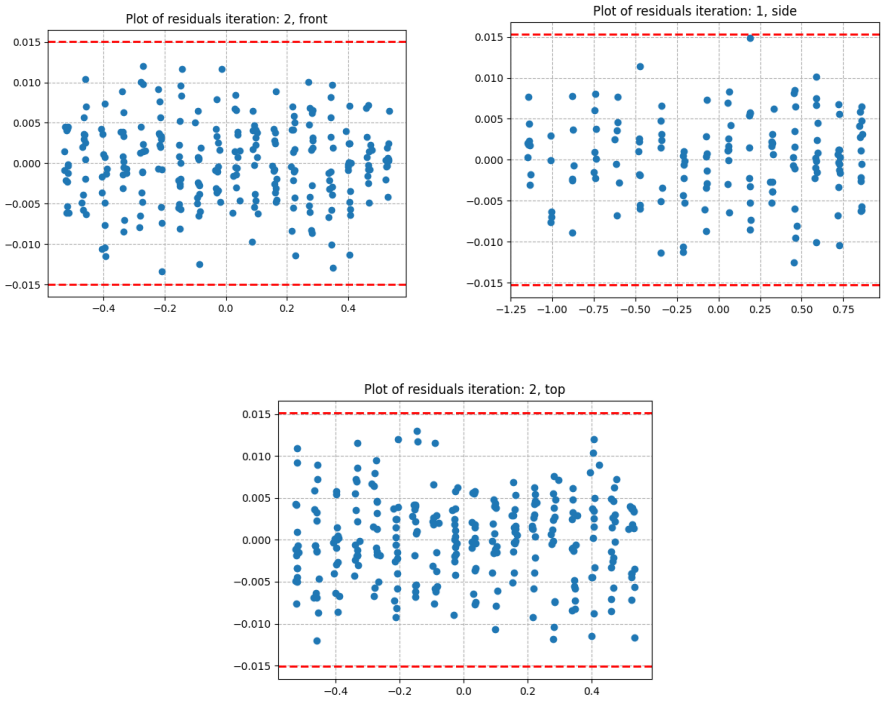

Figure 4 Plot of Residuals for each Wall

Corner Points Determination:

The next task for our algorithm is to determine the corner points of the geometry to compare the dimensions of the 'learned' as built geometry to the design. We did this by formulating an Ax=b problem to find the intersection of three planes. A plane can be defined by the following equation:

$$ax + by + cz + d = 0$$

Where $a, b,$ and c represent the components of the normal vector, $x, y,$ and z represent the coordinates of a point, and d represent the offset of the plane from the origin. A corner point in the context of a three-dimensional planar geometry is the intersection of three planes. Thus, the point that satisfied the equations of three planes in the geometry will be an intersection point and thus a corner. This can be represented by the following system of linear equations:

$$\begin{aligned} a_1x + b_1y + c_1z &= d_1 \ \ (plane\ 1) \\ a_2x + b_2y + c_2z &= d_2 \ \ (plane\ 2) \\ a_3x + b_3y + c_3z &= d_3 \ \ (plane\ 3) \end{aligned}$$

This can be turned into an $Ax = b$ problem where:

$$A = \begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{bmatrix}, \qquad x = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \qquad b = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix}$$

The normal components, A, as well as the offsets, b are known. As such, we solve for x which is the intersection between the three planes.

It is necessary that we use the fit plane to the clusters to find the corners of the geometry rather than the point in the dataset that are closest to a given corner due to the risk of noise in specific point measurement causing large errors. Using the fit plane represents a better approximation of the actual location of the as built corner. As is clear in Figure 3, the error in individual points can be up to 15mm which is far too large to use for corner point determination when the design tolerance is 25mm. The error in the fit planes will be much less due to the normal distribution of noise over all points.

A problem with using the planes that make up the geometry is that there are many combinations of planes that intersect but are not corners of the geometry. The mathematical objects that represent the planes extend infinitely. For example, the top slanted wall of the footing could intersect with the base and side wall of the footing if the planes were to be extended. Thus, to find the real corner points of a general planar geometry we must iterate through all combinations of three planes in the geometry, solve the $Ax = b$ problem to find the intersection at each iteration, and then filter out the intersections points that either don't exist or are outside the bounds the of geometry. This can be illustrated with the following pseudo code:

For i, j, k combinations in planes in geometry:

$Ap = d$

If p outside of geometry bounds, discard

Where, in the above code, p is the intersection point at a given iteration. The bounds for the geometry are defined by the using the minimum and maximum $x, y,$ and z of the data that defines the geometry. This makes a cube boundary cube around the geometry:

$$\begin{Bmatrix} x_{boundary} = (x_{min}, x_{max}) \\ y_{boundary} = (y_{min}, y_{max}) \\ z_{boundary} = (z_{min}, z_{max}) \end{Bmatrix}$$

By using the data set to define the elimination boundaries, it necessitates that the corner point from the fit plane approximation will be within the bounds since the fit plane corner will be in the middle of the standard distribution of the data. This effectively eliminates other extended plane intersections outside the geometry. Although large outliers in the data set may greatly effect the cube bounds of the geometry, this is not a large concern for the application of this algorithm with the footing analysis data (LiDAR data that has been post processed, does not have excessively large outliers). More complex planar geometries may also have plane intersections within the cube boundary. For this problem, another filter that adjusts for specific angles of faces is needed; however, this was not necessary for the case of analysing the footing.

As-Built Deviation:

Finally, the last task of the algorithm is to compute the geometric differences between the as built footing and the design geometry. This is simply done taking the absolute error of the matrix of as built corner points of the footing and the inputted design corner points.

$$\delta = \left| P_{design} - P_{as\ built} \right|$$

Where δ in the above equation is the deviation. The norm of the deviation can be taken to find a one-dimensional value for the deviation.

$$\delta_{1D} = \|\delta_{3D}\|$$

Mathematical Basis

Plane Fitting:

Given a set of points with x, y, and z values which we have shifted to the centroid of the point cloud, we can define the following matrix:

$$A = \begin{bmatrix} x_1 & y_1 & z_1 \\ \vdots & \vdots & \vdots \\ x_n & y_n & z_n \end{bmatrix}$$

The goal is find a best fit normal vector $\vec{n}$ such that:

$$\|A\vec{n}\|^2{}_2$$

Is minimized. We can do this via an SVD. An SVD decomposes A into three matrices, $A = U\Sigma V^T$. For our problem we care about the relationship between the rows of A, and so we vectors of the V matrix which are called the right singular vectors.

$$V^T = \begin{bmatrix} - & v_1^T & - \\ - & v_2^T & - \\ - & v_3^T & - \end{bmatrix}$$

The V matrix represents the orthonormal basis for the row space of A which makes descriptive for how the rows, which are the points (x, y, z), relate to each other. Because of this, the right singular vectors represent orthogonal direction in the 3D space of points. In the SVD each singular vector

corresponds to a singular value in Σ, all of which are the square roots of the eigenvalues. Therefore, the smallest right singular vector corresponds to the smallest eigenvalue, meaning the direction of least stretch for the points, which gives the direction of $\vec{n}$.

After assigning points to clusters, we center them around zero and create a matrix of these points, compute an SVD, and use the smallest right singular vector as the normal vector of fit.

Corner Point Computation:

The solution to the $Ax = b$ problem can be found many ways In our algorithm we used numpy.linalg.solve which uses LU decomposition to find x in most cases.

Results

Normal Vectors of the Fitted Planes

The normal vectors and the plane offsets of the three as-built footing faces (front, side, and top), computed numerically with the aforementioned SVD-based plane fitting and outlier elimination process, are shown below:

$$\boldsymbol{n}_{front}^T = [-0.0009 \quad 1.0000 \quad 0.0017], \qquad d_{front} = 0.0002$$

$$\boldsymbol{n}_{side}^T = [-1.0000 \quad -0.0069 \quad -0.0012], \quad d_{side} = 1.0519$$

$$\boldsymbol{n}_{top}^T = [-0.0014 \quad 0.4653 \quad -0.8851], \quad d_{top} = 0.4426$$

The normal and offsets of the as-designed footing are shown below:

$$\boldsymbol{n}_{front,design}^T = [0 \quad 1 \quad 0], \qquad d_{front} = 0$$

$$\boldsymbol{n}_{side,design}^T = [-1 \quad 0 \quad 0], \qquad d_{side} = 1$$

$$\boldsymbol{n}_{top,design}^T = [0 \quad 0.4472 \quad 0.8944], \qquad d_{top} = 0.4426$$

Computed Corner Points

The corner points for the as-built geometry, numerically computed, are listed below with the as-designed (ideal) corner points :

	As-built			**As-designed**		
$front-left\ bottom$	0.0000	−0.0001	0.0000	0	0	0
$front-right\ bottom$	1.0519	0.0008	0.0000	1	0	0
$back-left\ bottm$	1.0498	2.0000	0.0000	0	2	0
$back-right\ bottm$	1.0498	2.0000	0.0000	1	2	0
$front-left\ top$	0.0000	−0.0011	0.4997	0	0	0.5
$front-right\ top$	1.0512	−0.0003	0.4984	1	0	0.5
$back-left\ top$	0.0000	2.0000	1.5514	0	2	1.5
$back-right\ top$	1.0476	2.0000	1.5496	1	2	1.5
	[m]			[m]		

Computed Error on Corner Points

The absolute error of the as-built corner point with the as-designed corner points are shown below:

Error			Norm of Error	
$\begin{bmatrix} 0 & 0.0001 & 0 \\ 0.0519 & 0.0008 & 0 \\ 0 & 0 & 0 \\ 0.0498 & 0 & 0 \\ 0 & 0.0012 & 0.0003 \\ 0.0512 & 0.0002 & 0.0015 \\ 0 & 0 & 0.0514 \\ 00.0476 & 0 & 0.0496 \end{bmatrix}$	[m]		$\begin{bmatrix} 0.0001 \\ 0.0519 \\ 0 \\ 0.0227 \\ 0.0003 \\ 0.0454 \\ 0.0311 \\ 0.0458 \end{bmatrix}$	[m]

The max error is 51.9mm which is greater than the 25mm tolerance specified in the design. Therefore, the footing is not built as specified. Note that noise in corner point computation has a diminished effect because we used the 'learned' fit planes to find the corners rather than specific point measurements. The footing was built with a width that was too large and second height that was too large. This is illustrated in the plot bellow where orange is the as built corners and blue is the design corners.

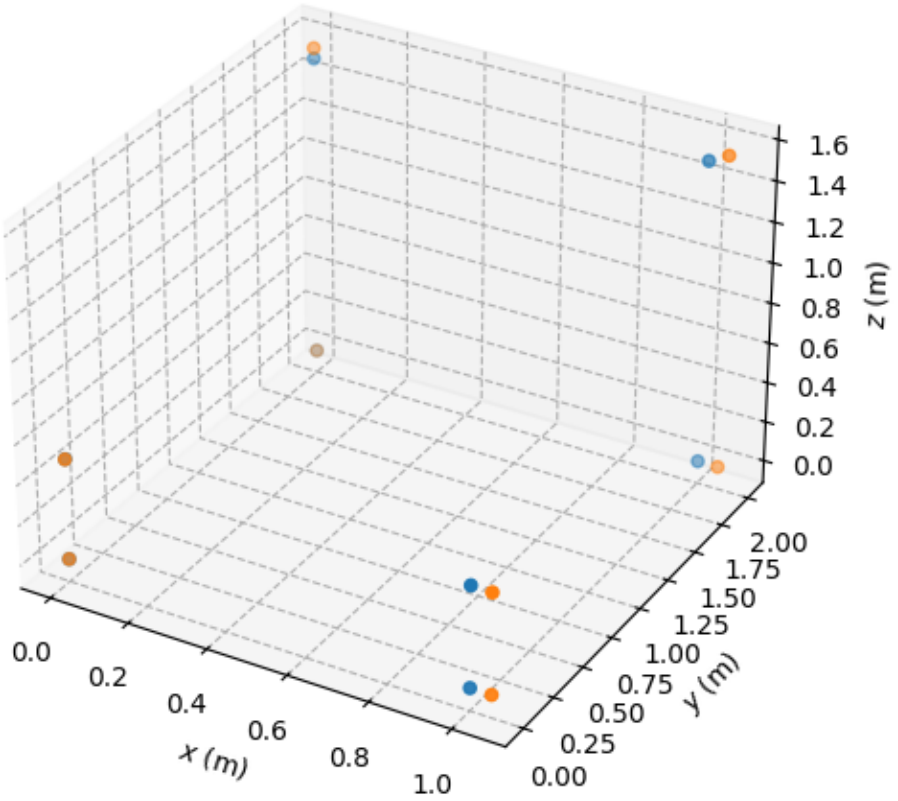

Figure 5 Corner Points Plot

Discussion and Conclusion

Our algorithm successfully processed LiDAR point cloud data, clustering with plane fitting, and removing statistical outliers. Comparing the computed corner points to the design specifications revealed deviations exceeding the design tolerances. Specifically, the top face exhibited a

noticeable tilt, resulting in incorrect Z-coordinates at the back corners. Additionally, the side face was offset too much resulting in the footing being too wide. These findings confirm that the as-built geometry does not align with the design specifications, thus CEF is right to question the integrity of FCC's analysis of the footing quality. We conclude that corrective action against FCC is justified.

References

[1] James Forbes, *Project Description – ET.3 & CS.2*, McGill University, 2024

[2] Brunton S. et al., *Data Driven Science and Engineering*, Cambridge University Press, 2019