UNIVERSITY OF CAPE TOWN

DEPARTMENT OF COMPUTER SCIENCE

CS/IT Honours Project Final Paper 2022

Title: Semantically Meaningful Density-Based Clustering of Cultural Heritage Sites

Author: Leah Gluckman

Project Abbreviation: CHSEG

Supervisor(s): Patrick Marais and Luc Hayward

Category	*Min*	*Max*	Chosen
Requirement Analysis and Design	*0*	*20*	0
Theoretical Analysis	*0*	*25*	0
Experiment Design and Execution	*0*	*20*	20
System Development and Implementation	*0*	*20*	10
Results, Findings and Conclusions	*10*	*20*	20
Aim Formulation and Background Work	*10*	*15*	10
Quality of Paper Writing and Presentation	*10*		10
Quality of Deliverables	*10*		10
Overall General Project Evaluation (*this section allowed only with motivation letter from supervisor*)	*0*	*10*	0
Total marks		**80**	**80**

Semantically Meaningful Density-Based Clustering of Cultural Heritage Sites

Leah Gluckman
University of Cape Town
Cape Town, South Africa
glclea001@myuct.ac.za

Abstract

Constructing 3D models from laser scans of Cultural Heritage (CH) sites is a vital area of research for CH preservation. These scans produce point clouds which can be visualised as 3D models. However, the construction process is challenging because laser scans of CH sites contain unwanted information, such as foliage, people and scaffolding, that do not belong to the CH site and which one must remove. Removing this noisy data is called point cloud cleaning - a predominantly manual, laborious and time-consuming task which must be automated and made more efficient. This research paper applies the DBSCAN, mean-shift and OPTICS density-based clustering methods to the point cloud classification problem and evaluates them on three separate datasets with varying feature sets. The aim is that the algorithms divide point clouds into clusters that can be grouped into semantically meaningful partitions and classified using binary classification labels of *keep* (belonging to the CH site) and *discard*. This paper also evaluates whether the density-based algorithms produce more semantically meaningful clusters than k-means, a partitional clustering algorithm, based on cluster validity metrics. The experimental results show that the mean-shift clustering algorithm can accurately and efficiently partition a CH point cloud into clusters that can classify into *keep* and *discard*. In addition, mean-shift outperforms k-means when evaluated using classification evaluation metrics on two datasets. However, DBSCAN and OPTICS produce poor classification results and do not produce more semantically meaningful clusters than k-means. Density-based clustering methods can classify CH clouds into *keep* and *discard* clusters. However, they may not be a viable solution to point cloud cleaning because they produce numerous partitions rather than one for each classification label.

Keywords

Point Clouds, Point Cloud Cleaning, Machine Learning, Classification, Clustering, Density-Based Clustering, Semantic Segmentation, Feature Extraction

1 Introduction

The construction of 3D models from laser scans of Cultural Heritage (CH) sites is a vital but challenging area of research for the preservation of CH. 3D models of CH sites are required to record and visualise the three-dimensional structure of historical sites. Preservationists can use such models to decipher where the CH site is deteriorating and needs improvement. In addition, the models are a means of transmitting a testimony of past human activity to future generations. However, constructing these models is not simple. Terrestrial Laser Scanning (TLS) scans CH sites, resulting in data that can produce 3D point clouds. Besides the CH site, these scans contain noise such as foliage, scaffolding, people, and animals that are not part of the CH site and which one must remove. This removal process is called point cloud cleaning – a mainly manual, labour-intensive, and time-consuming process. There is a need to automate this process and improve its efficiency.

This research project explores the application of unsupervised density-based clustering algorithms to the semantic segmentation of 3D point clouds of CH sites. The goal is to assess whether clustering can simplify and accelerate the arduous task of point cloud cleaning. The project aims to implement, analyse, and compare the performance of three density-based clustering algorithms - DBSCAN, mean-shift and OPTICS - at accurately and efficiently producing semantically meaningful clusters of three point cloud datasets. It does this by comparing the Silhouette Coefficients (SC) and Davies Bouldin (DB) indexes of these algorithms against a baseline partitional clustering algorithm, k-means. These three datasets include a raw dataset with $(X, Y, Z, intensity)$ features, a dataset with human-understandable features generated with CloudCompare, and a dataset with Machine Learning (ML)-generated features created with PointNet++. The clustering operates on a set of per-point vectors corresponding to the feature sets of each dataset. In addition, the project aims to use the resulting partitions to implement binary classification labelling that labels each or several clusters as belonging to the CH site (*keep*) or erroneous (*discard*).

The structure of this paper comprises an explanation of key terms and related works surrounding this topic, a detailed description of the design and implementation pipeline of the project, the evaluation metrics, and a discussion of the experimental results.

2 Background

Point clouds are a set of data points in 3D space defined by three spatial coordinates (X, Y, Z). They can be visualised as 3D models and are thus a popular and effective representation of objects and structures from the real world. Point clouds may contain additional information describing different properties of points, such as (R, G, B) colour and *intensity*, called features. These features develop from a 3D object's specific local or global geometric characteristics and are invariant to transformations. Descriptive features make it possible to perform classification on point clouds.

Point cloud cleaning involves removing noise and unwanted objects from point clouds to enable the construction of accurate 3D models with limited to no extraneous information. Classification, also known as semantic segmentation, is the process of categorising a dataset, such as a point cloud, into a set of classes with similar properties. It is the task of assigning meaningful, human-understandable labels to the clusters produced during segmentation. After a segmentation process, such as clustering, partitions a 3D dataset into

regions, classification assigns a class label to each region, providing each one with some semantics. A class label is a category such as 'tree', 'wall' or 'ground plane'. Point cloud classification is a ML approach that aims to automate the laborious, time-consuming point cloud cleaning process of assigning class labels to points in a point cloud. In the case of CH sites, it seeks to accurately and efficiently predict the class labels of unseen points within a point cloud by classifying 3D points as either relevant to the CH or not [19].

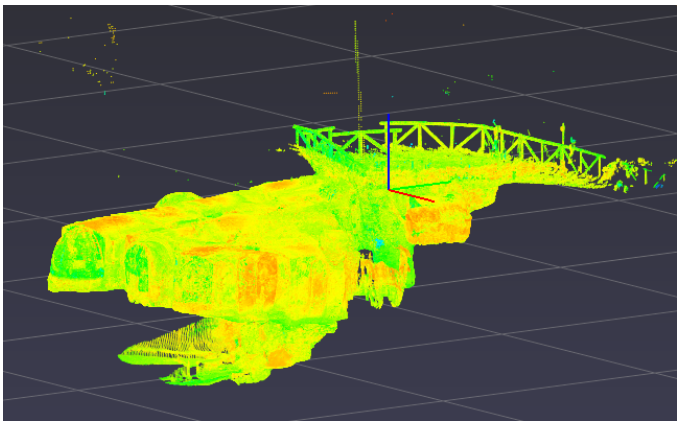


Figure 1: Visualisation of the *intensity* feature of the point cloud of an underground church CH site (the raw dataset)

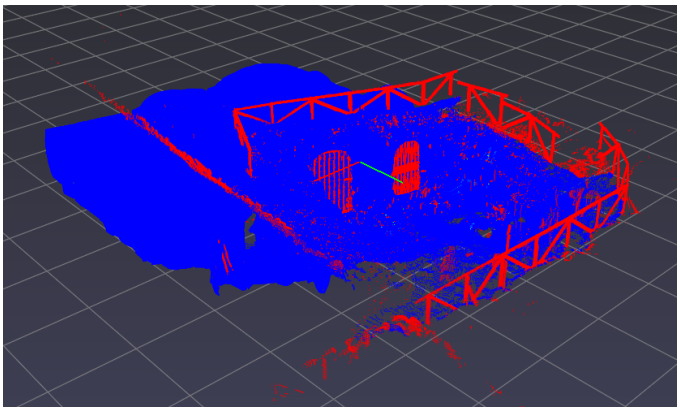


Figure 2: Visualisation of the actual ground truth of the point cloud of an underground church CH site. Parts of the church (*keep* regions) are depicted in blue, and structures that need to be removed (*discard* regions) are depicted in red.

Point cloud classification has been applied as an automatic method of classifying points as relevant or not in the context of point cloud cleaning by Marais et al. [15]. However, point clouds, particularly those of CH sites, have limited labelled training data, a requirement of supervised learning. In addition, point clouds of CH scans often comprise large datasets that contain a significant amount of 3D point data. Datasets also vary, each comprising a different set of objects, ruins, and buildings. CH site scans can range from well-preserved structures to eroding and highly irregular ruins. The variance in CH data makes a general classification model hard to construct. It is the main reason a single model cannot be used across all sites naively, as is possible in urban environments.

Unsupervised learning algorithms find patterns in unlabelled data. They aim to predict a target variable without ever seeing examples. Clustering is an unsupervised ML approach that partitions data into homogeneous subgroups, called clusters, such that each contains objects that bear more similarity to one another than those in other clusters. The aim is to maximise intra-cluster similarity whilst minimising inter-cluster similarity. Clustering algorithms are easy to understand, and their lack of a pre-processing step makes them simple to implement. They do not require any labelled training data and, therefore, are a suitable solution to the problem of point cloud classification because point clouds, particularly those of CH sites, have limited labelled training examples.

Clustering can create numerous context-based and variance-resistant groupings from data. However, clustering methods cannot always discern meaningful and relevant clusters in datasets with substantial density discrepancies or high-dimensional data – both common to point clouds. In addition, high-dimensional data, such as point clouds with large feature sets, is subject to the *curse of dimensionality*. This curse implies that as the number of dimensions increases, the degree of significant differentiation between distinct and similar points decreases [4]. This, along with the significant amount of noise present in high-dimensional data, poses a challenge to the effectiveness of clustering. Hence, unsupervised clustering is not guaranteed to produce semantically meaningful clusters. Density-based clustering methods, however, can identify arbitrarily-shaped clusters in high-density regions, making them a suitable approach to the point cloud classification task.

This paper's primary research question is: 'do density-based clustering methods produce more semantically meaningful clusters on CH site point clouds than a baseline k-means algorithm, based on the SC and DB index cluster validity metrics?' A secondary research question is: 'what is the upper bound accuracy when using the resulting clusters to perform binary classification using *keep* and *discard* labels on a point cloud?'

2.1 Related Work

Researchers explore point cloud classification through various approaches that aim to classify 3D points in laser scans as either relevant or not. Marais et al. [15] implement a Random Forest classifier to perform point cloud classification using an intuitive binary point labelling system of *keep* and *discard* labels. Their classifier achieves a 98% average accuracy and successfully predicts the class labels of points in the point cloud dataset. Whilst this is a supervised classification approach, we use their binary labelling approach to classify produced clusters into *keep* and *discard* and aim to achieve a similar accuracy using unsupervised learning. Shan and Sampath [20] implement a k-means algorithm to find points belonging to a specific roof segment. Their algorithm successfully clusters the point clouds, despite the presence of outliers. This work aims to achieve similar success in clustering and classifying a point cloud using k-means clustering. In addition, Aljumaily et al. [2] apply the DBSCAN clustering algorithm to point clouds and achieve meaningful and relevant clusters, despite the high-dimensional nature and

substantial density discrepancies within the dataset. This suggests that density-based clustering algorithms should succeed in partitioning the CH point clouds into semantically meaningful classes. Similarly, Tonini and Abellan [23] effectively classify feature points within a dataset with DBSCAN and identify arbitrarily shaped clusters. Ester et al. [6] evaluate the efficiency and effectiveness of DBSCAN in discovering clusters of arbitrary shapes in large spatial databases with noise. They find that the algorithm outperforms the partitional clustering algorithm, CLARANS, in both efficiency and clustering efficacy. However, their experiment does not investigate the clustering algorithm's ability at clustering high-dimensional spaces. We explore the effectiveness and efficiency of DBSCAN at clustering point clouds with high-dimensional feature sets, aiming to achieve a similar result - whereby the density-based algorithm outperforms the partitional algorithm, k-means.

Grilli et al. [10] analyse the impact of several geometric covariance features on the classification of CH point clouds. They extract various covariance features computed within a 3D point's local neighbourhood and input the resulting point cloud into a classifier, intending to classify the main linear, volumetric and planar structures. Their research discovers *verticality*, *surface variation*, *planarity* and *sphericity* as highly relevant features for CH point clouds.

3 Design and Implementation

The implementation pipeline includes stages of data acquisition, pre-processing, feature extraction, density-based clustering, classification, visualisation, evaluation, and analysis. The implementation was conducted on an 11th Generation Intel Core i7-1165G7 and Google Collaboratory Pro with a Tesla T4 GPU and 25 GB of RAM.

3.1 Datasets

A point cloud of a registered church with 21,901,958 points was acquired from the Visual Computing Group (VCG) at CNR-ISTI [5] and used as the raw dataset. Whilst we only test on one large-scale point cloud, it contains many challenging regions and represents the upper end of complexity in CH point clouds. We extract features from it through two different applications to create two additional rich feature sets to test the clustering and classification methods on. The final datasets used are:

(1) Dataset with $(X, Y, Z, intensity)$ features.
(2) Dataset with (X, Y, Z) and 22 human-interpretable geometric features, generated with CloudCompare.
(3) Dataset with (X, Y, Z) and 126 ML-generated features, created with PointNet++.

This research paper refers to these datasets as the raw dataset, dataset 2 and dataset 3, respectively. We perform several pre-processing steps before feeding the datasets into the clustering algorithms.

3.1.1 Downsampling The raw point cloud contains over 21 million points and is too large for the clustering algorithms or hardware to handle. Thus, all the datasets are voxel downsampled. Voxel downsampling constructs a uniformly downsampled point cloud by grouping points into voxels, boxes of 3D points, and averaging all the points per voxel to produce a single point [21]. We downsample all the datasets using a voxel of 0.085. Except dataset 3, which we downsample with a voxel of 0.075 before extracting features since it is too large to feed into PointNet++ with the given hardware and software resources. We then downsample it further with a voxel of 0.085 after feature extraction but before clustering. The OPTICS clustering algorithm, discussed later, cannot run on that downsampled dataset 3. Thus, we downsample dataset 3 with a voxel of 0.19 before inputting it into the OPTICS clustering algorithm.

3.1.2 File Conversion We store the downsampled point clouds in *ply* formats and dataset 2 after feature extraction in a *las* format. We then convert these datasets to *npy* file formats to be able to hold them in NumPy [1] arrays. In addition, PointNet++ requires input point clouds to be in the form (X, Y, Z, R, G, B), but the raw point cloud is in the format $(X, Y, Z, intensity)$. We implement a data conversion to store the *intensity* values in the (R, G, B) values.

3.1.3 Ground Truth Rounding The ground truth values are either 0 or 1, indicating whether a point is *keep* (part of the CH site) or *discard*. However, voxel downsampling of the point clouds results in ground truth values being between 1 and 0. Thus, we use the *ceiling* function to round all ground truth values greater than 0 to 1, thereby ensuring that the ground truth values are only in the binary set {0,1}. We create two NumPy arrays from the input point cloud: one including and one excluding the ground truth labels. We use the former for clustering and the latter to classify the point cloud using the resulting clusters.

3.2 Feature Extraction

Feature extraction is obtaining rich descriptors – features – from a dataset. We extract features from the raw dataset using two methods to create an additional two datasets for testing: one containing human-interpretable geometric features and the other containing ML-generated features. The goal is to feed datasets with a higher dimensional feature space into the clustering algorithms to increase the algorithms' chance of partitioning the point cloud into semantically meaningful clusters.

3.2.1 Cloud Compare Feature Extraction CloudCompare is a 3D point cloud processing software that can visualise, edit, and extract features from 3D data. We use it to extract a set of 22 human-interpretable geometric features from the registered church point cloud we acquired from VCG, computed within a local neighbourhood radius of 0.049006 of a 3D point [10]. This set of features includes covariance, density, curvature, roughness and moment features (see Table 1). We extract the covariance features because they are the most relevant to CH sites – all CH sites comprise features such as *planarity* and *linearity*. Through extracting such features, we aim to enable the clustering algorithms to detect the geometrics features in the church dataset, such as the walls, ceilings and doorways. Some covariance features, however, are more relevant than others. *Surface variation*, *sphericity*, *planarity* and *linearity* are features of most, if not all, points in a CH site like a church. Other covariance features, including *eigenentropy* and *sum of eigenvalues*, are the least relevant covariance features. However, we still include them in the feature selection for dataset 2.

[1] NumPy is an open-source Python package for scientific computing with comprehensive mathematical functions and powerful n-dimensional arrays [11]

Feature (0.049006 local radius)	Type of Geometric Feature
Sum of eigenvalues Omnivariance Eigenentropy Anisotropy Planarity Linearity PCA1 PCA2 Surface variation Sphericity Verticality 1st eigenvalue 2nd eigenvalue 3rd eigenvalue	Covariance
Mean curvature Gaussian curvature Normal change rate	Curvature
Roughness	Roughness
Number of neighbours Surface density Volume density	Density
1st order moment	Moment

Table 1: Table of Geometric Features Generated with CloudCompare

3.2.2 PointNet++ Feature Extraction PointNet++ is a neural network architecture that processes point datasets in a metric space [18]. It performs hierarchical feature learning, segmentation, and classification of point sets. This work utilises PointNet++ for its feature learning component. PointNet++ learns a spatial encoding of every point in the dataset and then aggregates the point features to form a global point cloud signature [18]. It leverages point neighbourhoods at numerous scales, recursively extracts local features from these neighbourhoods, and combines them into large groups. This process repeats until it obtains the final feature set. We use PointNet++ to extract 126 ML-generated features from the raw registered church point cloud dataset. These features are difficult to map to geometric features like *planarity* but may contain information that improves the clustering algorithms' performance at creating semantically meaningful partitions.

3.3 Density-Based Clustering

Clustering partitions a dataset into homogeneous subgroups called clusters such that each comprises similar points, thereby maximising inter-cluster similarity. In addition, distinct clusters contain unrelated points such that intra-cluster similarity is minimised. There are several classes of clustering algorithms, including partitional, hierarchical, and density-based classes. This research paper investigates the performance of three density-based clustering algorithms at partitioning the three point cloud datasets with varying feature sets into semantically meaningful clusters that can classify using binary classification labels of *keep* and *discard*. A partitional clustering algorithm, k-means, is used as a baseline for comparison. We implement the clustering algorithms using the Python library Scikit-learn [1] and tune the hyperparameters to aid the algorithms' performance.

Density-based clustering algorithms identify clusters as maximal sets of density-connected points in space [7]. The density of points within each partition is significantly greater than outside it. Thus, low-density regions (noise) separate clusters. Density-based clustering methods can identify arbitrarily shaped clusters and effectively discover noise [13]. Both of these components are prevalent in CH point clouds. Unlike partitional clustering methods, density-based algorithms require no prior information about the number of clusters. These factors make density-based algorithms a more desirable solution to the problem of point cloud classification than hierarchical and partitional clustering algorithms.

3.3.1 K-means Clustering Algorithm K-means is a partitional clustering algorithm that divides a dataset into k clusters. It is easy to understand and fast to implement. However, one must specify the number of clusters in advance, and the algorithm does not necessarily account for outliers. K-means is used as a baseline for comparison against the density-based clustering methods, using cluster validity metrics, visual inspection, and classification evaluation metrics.

3.3.2 DBSCAN Clustering Algorithm Density-Based Spatial Clustering of Applications (DBSCAN) is a density-based clustering method capable of identifying arbitrary-shaped clusters. Such clusters are prevalent in point clouds since TLS scans contain data of arbitrary shapes and densities, and thus a density-based segmentation approach is relevant. The DBSCAN clustering algorithm requires at least two parameters whose values significantly impact the clustering result: *eps*, the searching radius, and *minPts*, the minimum number of points per cluster. It is difficult to determine the values of these parameters, particularly *eps*, which makes the clustering process difficult and can hinder the results of DBSCAN. However, with the correct parameters, the algorithm can effectively group areas of high density into clusters and identify outliers. It discovers clusters as dense regions separated by objects of low density or noise [13]. For every point, the algorithm searches for the minimum points within a given radius to assign to a cluster. If it cannot find enough points within that radius it labels the point as noise.

3.3.3 OPTICS Clustering Algorithm Ordering Points to Identify the Clustering Structure (OPTICS) constructs an augmented ordering of the dataset, representing its density-based clustering structure [3]. OPTICS is similar to DBSCAN, except that it does not assign cluster memberships. Instead, it stores the order of processing points and information DBSCAN uses to determine cluster memberships. For each point OPTICS processes, it computes two distance values: reachability-distance and core-distance. Not being limited by a single global parameter setting is its main advantage. It stores information about the density-based clusters that correspond to a broad range of parameter settings [3]. OPTICS is, therefore, a versatile approach for interactive and automatic cluster analysis [3]. OPTICS discovers core high-density samples and creates clusters from them. The implementation used in this work differs from

[1] Scikit-learn is a free software ML library for the Python programming language [17].

the original OPTICS implementation since it initially performs k-nearest-neighbourhood searches on every point to identify core sizes [17]. Then, when it constructs the cluster order, it computes the distances to unprocessed points only. The main disadvantage of this implementation is that the time complexity is O(n^2).

3.3.4 Mean-Shift Clustering Algorithm Mean-shift clustering iteratively assigns points in a dataset to clusters by shifting each point to the highest point density region in the space (the cluster centroid). The algorithm requires a *bandwidth* parameter whose selection heavily influences the algorithm's performance and the overall density estimation. The *bandwidth* corresponds to the largest range over which the algorithm can discover an equal number of clusters [24]. The algorithm works in a Euclidean feature space and identifies clusters around centroids that correspond to dense areas within the feature space [25]. Mean-shift clustering is guaranteed to converge but will stop iterating when the change in centroids becomes very small [17].

3.4 Classification

Classification is the task of assigning semantically meaningful labels to clusters produced during segmentation. The goal is to classify irrelevant clusters as noise or *discard* and relevant ones belonging to the CH site as *keep*. The raw dataset contains ground truth labels of *keep* and *discard* that annotate whether a point in the point cloud is part of the CH site or not. Unsupervised clustering algorithms segment the point clouds into clusters. Thus, the ground truth labels are not input into the algorithms. However, we use them after the clustering stage to implement binary classification of *keep* and *discard* on the clustered point cloud.

The clusters formed by the various clustering algorithms are input into the classification stage of the pipeline. We align each point with its corresponding ground truth label and assign each cluster the majority ground truth class of the points within it. For instance, a cluster containing more points labelled *keep* than *discard*, is labelled *keep*, and vice versa. The final cluster class labels provide an upper bound on the classification accuracy, given the produced clusters.

3.5 Visualisation

Visualisation is a critical step in the implementation pipeline. It enables visual inspection of the clustering and classification output. Python's Point Processing Toolkit, PPTK, is used to interactively visualise the point clouds and their attributes, including ground truth labels, features, and resulting clusters.

4 Evaluation

This work utilises several clustering and classification evaluation metrics to quantify whether the clustering algorithms produce semantically meaningful clusters and determine the quality of each algorithm. In addition, the metrics evaluate the success or failure of the algorithms in creating partitions that can classify as *keep* and *discard*. We use the Scikit-learn Python library to implement the metrics, including SC, DB index, precision, F1 score, Intersection Over Union (IOU) score, recall and Mean Squared Error (MSE).

4.1 Clustering Evaluation Metrics

A clustering algorithm's primary goal is to maximise intra-cluster similarity whilst minimising inter-cluster similarity. It is vital to test the validity of the created clusters to determine whether that goal holds.

4.1.1 Silhouette Coefficient Good clustering algorithms produce a high cohesion within and high separation between clusters [16]. The SC combines these two metrics to provide a simple qualification framework and computes a similarity metric using the distance between points or clusters. One calculates it by combining the average distance between a point and every other point in the same and nearest cluster. The SC ranges between -1 and 1. A positive value signifies highly dense, correct clusters and high separation between clusters [16]. A negative value suggests overlapping clusters, implying that several data points are assigned the incorrect class label. A zero value means the algorithm uniformly distributes data points throughout the space. Thus, the SC can determine an algorithm's success rate and whether it produces semantically meaningful clusters. This metric is also helpful for visually portraying the similarities and differences within and between clusters.

$$SC = \frac{b - a}{max(a, b)}$$

4.1.2 Davies Bouldin Index The DB index evaluates the density of data points in clusters and the degree of separation between different clusters. Very dense clusters that are well-spaced from others imply good clusters. The smaller the DB index, the better the clustering. This result is because small DB indexes indicate that clusters bear little to no similarity, meaning they are well-separated and compact.

4.2 Classification Evaluation Metrics

Performance metrics determine how applicable the produced clusters are for classifying points into binary classes. They describe the algorithm's ability to label points in a point cloud as part of the CH site or irrelevant and noise. All the classification metrics implemented take a macro (unweighted) average over the predicted labels, meaning that each label has equal weight and label imbalance is unaccounted for. A truth table (see Table 2) illustrates the four possible classification performance outcomes of clustering algorithms that describe their precision, accuracy, and recall. These outcomes comprise 'True Positive' (TP), 'True Negative' (TN), 'False Positive' (FP), and 'False Negative' (FN). TP is when the sample is positive and correctly classified as positive, which in this case is *discard*. FP is when the sample is negative, but the algorithm classifies it as positive. TN and FN behave similarly.

		Actual Class	
		Positive	**Negative**
Predicted Class	**True**	TP	FN
	False	FP	TN

Table 2: Truth table for evaluating clustering algorithms' classification performance, illustrating that the predicted label does not always equate to the actual class.

4.2.1 F1 Score The F1 score averages a model's performance by calculating the weighted average of precision and recall as a single metric. It is a number between 0 and 1, with scores closer to 1 being more desirable [12].

4.2.2 Intersection Over Union Score The IOU score determines how well the model perfectly separates points belonging to CH sites and noise. When evaluating noisy data, the score may be worse (lower) since clustering only the relevant data points becomes increasingly challenging [9].

4.2.3 Precision The precision metric determines the number of TPs within the cluster over the total positive records [8]. It reflects the clustering algorithm's accuracy in classifying a point as positive (*discard*). This metric is useful because it proves how reliably an algorithm can predict TPs. The more TP predictions an algorithm makes, the higher its precision percentage.

$$Precision = \frac{TP}{TP + FP}$$

4.2.4 Recall Recall determines the ratio of correctly classified points [22]. It reveals how well the algorithm can detect TP data points.

$$Recall = \frac{TP}{TP + FN}$$

4.2.5 Mean Squared Error MSE calculates the difference between the actual ground truth value and that predicted by the classifier, in this case, the clustering algorithm [14]. It is a positive-valued number that decreases as the error tends to zero.

4.3 Visual Inspection

The human eye is adept at detecting disparities in visual data. This work uses visual inspection as a heuristic for evaluating the quality of the clustering methods and the binary classification results. The goal is to achieve clusters that appear convincing to the human eye. Visually, it is difficult to evaluate the clustering since most algorithms produce thousands of partitions that look like many dots on the point cloud. This result is undesirable for the classification process since, ideally, we want to have as few clusters as possible so that a human operator can easily select those classified as *discard* to remove from the point cloud. By contrast, visual inspection proves useful in evaluating the classification prediction. It is clear whether the correct parts of the point cloud classify as *keep* or *discard*.

5 Results and Discussion

5.1 K-means Clustering Results

The k-means clustering algorithm performs significantly better than expected on all three datasets. Choosing the value of k proved the most difficult part of the algorithm. Initially, the algorithm was run up to 100 times on each dataset, incrementing the value of k on each iteration and calculating the SC and DB index on the clustering result. The optimal clustering validity scores provided a starting point for choosing values of k to run on the algorithm. However, after thorough testing, we disregarded these values of k as they did not perform well. For low values of k, which tend to have higher SC and DB indexes, k-means can only classify the outliers as *discard*. It captures no fine detail in the church CH site.

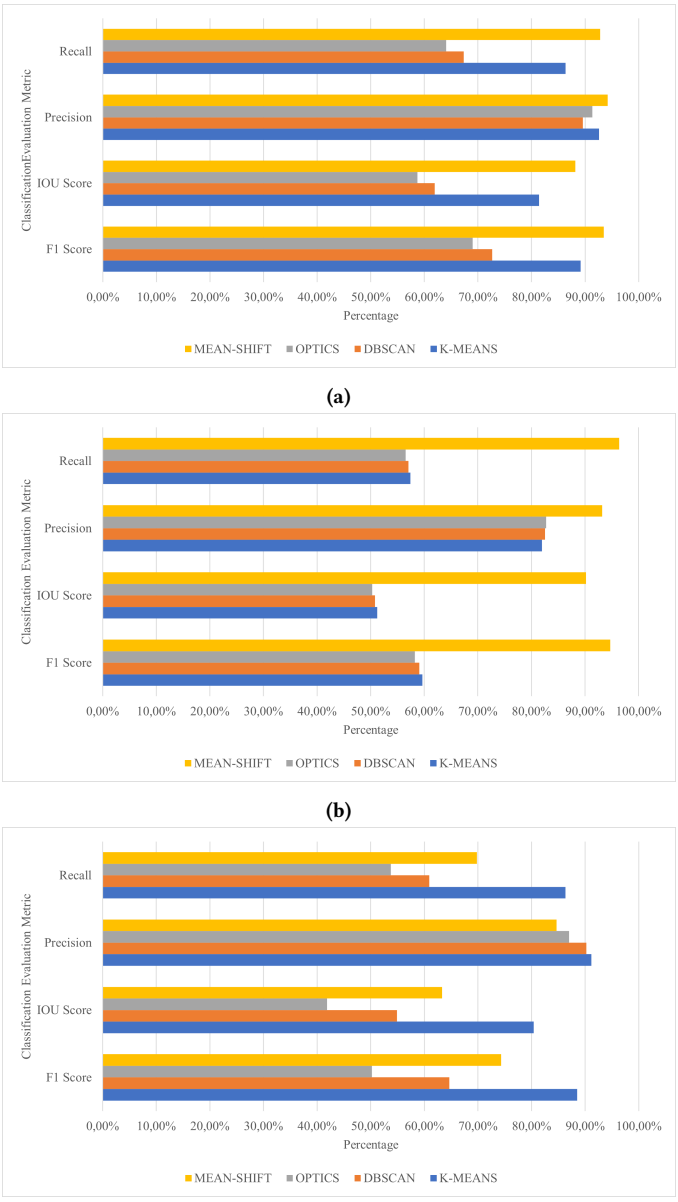


Figure 3: Three Bar Graphs Comparing the Classification Evaluation Metric Percentages for four clustering algorithms on (a) the Raw Dataset, (b) Dataset 2, and (c) Dataset 3

As the value of k increases, more fine detail is detected, and the algorithm can classify more accurately.

For the raw dataset, the optimal value of k is 13, based on the highest SC and lowest DB index values. However, the result of this clustering leads to poor classification results. Mean-shift obtains good classification results and creates roughly 30000 clusters (see Figure 6a). The decision was therefore made to choose much larger values of k and evaluate the classification metrics using the results

of that clustering. For 50 clusters on the raw dataset, k-means classifies more noise as *discard* than 13 clusters. For 500 clusters, it manages to classify part of the scaffolding outside the church as *discard*. More clusters result in better classification results, visually and according to the evaluation metrics. 5000 clusters give the best k-means results on the raw dataset. The final predicted classification result resembles the actual truth labels (see Figure 7a). The classification evaluation metrics reiterate this, as k-means achieves over 80% for precision, recall, IOU score and F1 score. In addition, Figure 3a shows that k-means outperforms the density-based clustering algorithms DBSCAN and OPTICS in all metrics and Figure 4 shows that it has the lowest MSE percentage.

There is a noticeable trade-off between the number of clusters and the relative speed of the algorithm, as more clusters mean longer computation time. However, the computation time of the clustering and classification stages using k-means clustering is still faster and more efficient than a manual point cloud cleaning process.

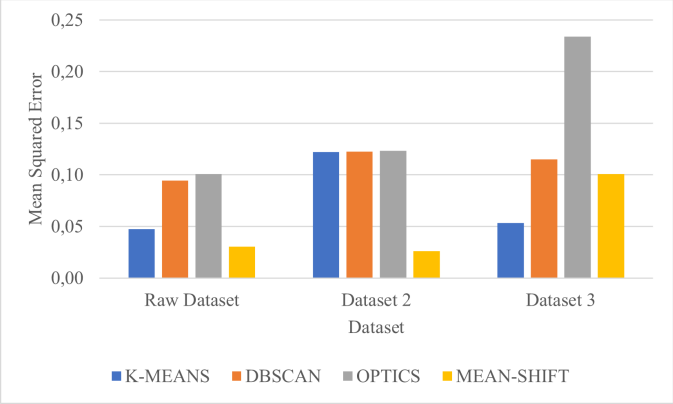


Figure 4: Bar Graph Comparing the MSE Results across four clustering algorithms and three datasets

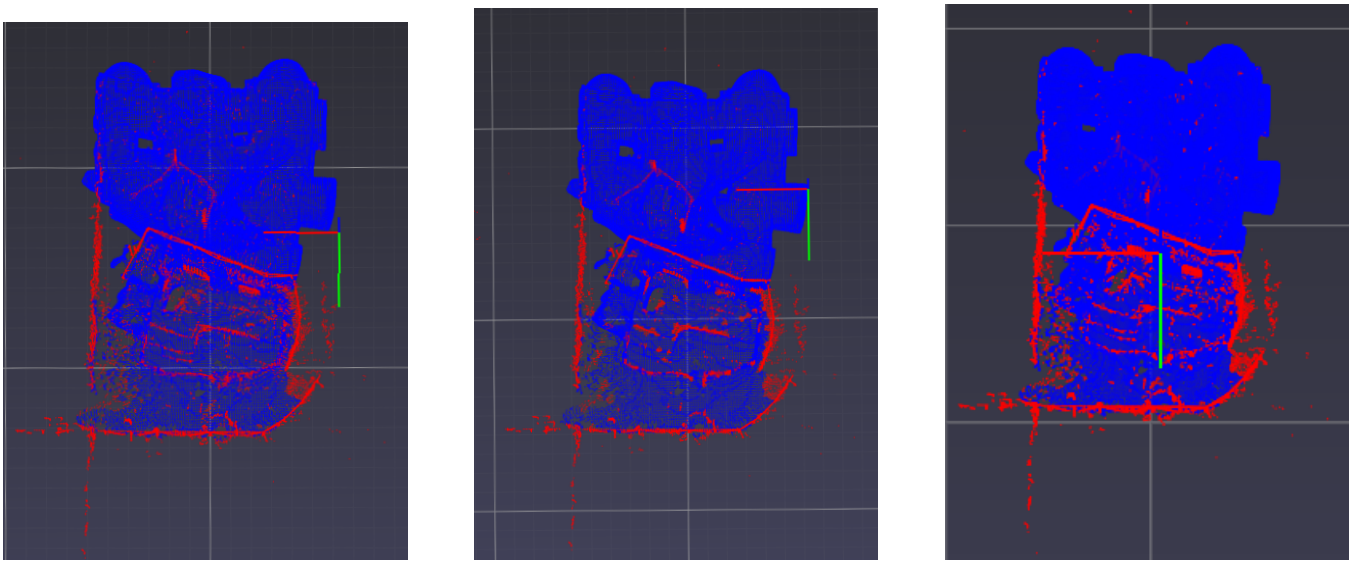
In contrast to the high classification results on the raw dataset, k-means does not perform well on dataset 2 and achieves similar results to DBSCAN and OPTICS (see Figure 3b). The initial SC computation suggests that k should be between 2 and 12 clusters. However, the classification results using these k values are poor since it labels all points in the point cloud as *keep*. We selected a large value of k to cluster dataset 2, and the classification result remained entirely *keep* until k increased to 1000 or greater. Unfortunately, even in its best case, k-means cannot effectively classify the regions of *discard* in dataset 2. With 6000 clusters, the algorithm achieves over 50% for all the classification evaluation metrics, besides precision, for which it obtains 81%, as seen in Figure 3b. Through visual inspection in Figure 8a, one can see that k-means with 6000 clusters can classify the noise exterior to the church structure and some points of *discard* within the church, but not enough that makes it noticeable. The algorithm does not perform well on a dataset with geometric features.

We initially chose k values of 7 and 13 to cluster dataset 3 with k-means, based on the highest SC achieved when choosing values of k up to 100. These values of k produce clusters that look like large chunks of the point cloud and are not semantically meaningful. Increasing the value of k creates smaller and smaller partitions, with the final clustering result (with k equal to 3000) looking like a pattern of small clusters. The SC and DB index values of 3000 clusters are lower than that of 7 clusters (see Tables 3 and 4), but the classification results are significantly improved. Hence, we use the cluster validity metrics as a guideline and starting point, but the final clustering results are not dependent upon them. For 7 clusters, the algorithm only classifies a portion of the noise exterior to the church as *discard* and everything else as *keep*. By contrast, the classification result of clustering using 3000 clusters results in a point cloud that visually resembles the actual ground truth (see Figure 9a). Figure 3c compares the classification evaluation metric percentages across the different clustering methods on dataset 3 and illustrates that all of the k-means classification evaluation metrics are greater than 86%, with the precision score being 91%. Its MSE is the lowest of the clustering methods (see Figure 4), indicating that the algorithm is the best predictor of correct cluster classes. Thus, k-means can accurately partition dataset 3, with over 100 ML-generated features, into clusters that can classify as *keep* and *discard*. This algorithm proves effective as a point cloud cleaning mechanism.

5.2 Mean-Shift Clustering Results

The mean-shift clustering algorithm performs the best out of the density-based algorithms on all three datasets when evaluated using the classification evaluation metrics. When clustering the raw dataset, the SC ranges between 0.18 and 0.27, depending on the choice of the *bandwidth* parameter. The results suggest poor clustering because SCs close to 0 indicate that points distribute uniformly throughout the space. In addition, such scores imply that the produced clusters are not highly dense or well-separated from others. However, this is a poor cluster validity metric for testing density-based clustering methods since thousands of clusters get produced, and thus they are not well-separated or dense. Despite the poor cluster validity metrics, the mean-shift algorithm achieves over 88% for the classification metrics when clustering the raw dataset, outperforming all the other clustering methods, as seen in Figure 3a. These results prove that this algorithm can accurately partition points in a point cloud into semantically meaningful classes of *keep* and *discard*. Visually, the classification result resembles the actual truth labels (see Figure 7c). We achieve the best classification results on the raw dataset with low *bandwidth* values of either 0.09 or 0.1 and poor classification results with higher *bandwidth* values of 1 and greater such as where the algorithm only manages to classify outliers and noise as *discard*.

Figure 3b and 4 show that, when testing dataset 2, the clustering algorithm achieves close to 100% prediction accuracy when evaluated using precision, recall, F1 score, and IOU score and an MSE close to 0%. We initially selected 0.09 as the *bandwidth* value based on the results of this algorithm on the raw dataset. However, an exhaustive test of different *bandwidth* values determined that the optimal *bandwidth* for clustering the dataset is 1. The visualised clusters in Figure 8c do not have much meaning because there are over 80000 clusters (see Figure 6b), so there are no distinct partitions that one can visually inspect. However, one can observe that



(a) Actual Ground Truth Binary Classification

(b) K-means Clustering Binary Classification Prediction

(c) Mean-shift Clustering Binary Classification Prediction

Figure 5: Visualisation of actual ground truth labels (a), and labels predicted by k-means (b) and mean-shift (c) clustering on the raw dataset

the areas coloured dark blue are *discard* regions. This number of clusters is undesirable for the classification process, as a human operator will need to manually select each *discard* cluster to remove from the point cloud. The visualisation of the classification result is more overt. There is a clear distinction between *keep* and *discard* clusters, and one can hardly discern the difference between the predicted classification result and the actual classification. Not only does the algorithm correctly classify the structures that one must clearly remove as *discard*, such as the scaffolding, but it also manages to label the specs of noise within the church as *discard*. One can conclude that the mean-shift clustering algorithm is adept at clustering a point cloud into groups that can classify into *keep* and *discard*.

The selection of the *bandwidth* parameter significantly impacts the clustering and classification result of the mean-shift algorithm on dataset 3. Choosing the wrong *bandwidth* value (greater than 1) results in the algorithm only classifying parts of the noise as *discard*. Contrastingly, selecting a *bandwidth* value of 1 makes the algorithm capable of labelling most of the scaffolding, inside and outside the church, as *discard* (see Figure 9c). The classification result is not as accurate as the other datasets, but the algorithm performs optimally out of the three density-based clustering algorithms.

5.3 DBSCAN Clustering Results

Estimating the parameters for DBSCAN, namely *eps* and *minPts*, proved a difficult task as there is no precise method for determining these parameters. Other research papers suggest that two times the number of dimensions in the feature set determines the *minPts* parameter. We use this approach as a starting point. Thus, the initial chosen values of *minPts* were 8, 50, and 252 for each respective dataset, calculated by doubling the number of features in each. We calculate the average distance between each point in the dataset and its *minPts* nearest neighbours and set *eps* to the maximum curvature value of all the calculated distance values - the elbow point.

For the raw dataset, *minPts* of 7 achieves the best classification results, and, in Figure 7b, one can see the *discard* points in the scaffolding and inside the church. However, the SC is close to negative 1 for all tested values of *minPts* and *eps*. We use these metrics as a guideline for parameter selection but they are not an accurate estimation of the cluster validity of density-based algorithms. The SC is a distance-based metric and is, thus, less competent at evaluating density-based than partitional clusters. This is because it does not perform well on arbitrary-shaped partitions produced by DBSCAN [1]. Table 3 shows that k-means, the partitional algorithm, consistently achieves the optimal SC out of all the clustering algorithms. In addition, the density-based clustering algorithms of DBSCAN and mean-shift are close to -1 for all datasets, even when the classification results are high. Visually, as seen in Figure 7b, the algorithm does not create distinct clusters or ones that appear semantically meaningful. The clusters look like points scattered around the CH site and are arbitrarily shaped rather than dense and well-separated. However, when grouping them for classification, they can accurately discern the regions of *keep* and *discard*.

After classifying each cluster in dataset 2 as either *keep* or *discard*, one can see in Figure 8b that these scattered points become red specs of *discard* within the areas that should classify as *discard*. There are no distinct *discard* regions. The algorithm performs worse on the point cloud with geometric features than without. This outcome is interesting because dataset 2 contains density-based geometric features, such as *surface density*, which one would assume a density-based clustering algorithm like DBSCAN would use to its

advantage. There is a chance that the inclusion of geometric features other than the density-based features hinders the advantage the density features could have on the algorithm's performance. Future work can research the effect of different geometric features on the clustering result. Figure 3b shows that DBSCAN achieves recall, F1 score, and IOU score percentages between 50 and 60%, but precision achieves 82%. This implies that the algorithm accurately predicts most of the *keep* but not the *discard* labels, which is clear from the visualised classification result. DBSCAN obtains similar results to the baseline k-means algorithm but performs significantly worse than the mean-shift clustering algorithm on dataset 2.

DBSCAN can detect part of the scaffolding outside and inside the church as *discard* when clustering and classifying dataset 3 (see Figure 9b). It achieves a 90% precision score when classifying the dataset using a *minPts* of 7. Trial and error determined the selection of the parameters for this dataset by testing using the parameters that performed well in the previous datasets and adjusting values where it was deemed necessary.

5.4 OPTICS Clustering Results

The OPTICS algorithm takes the longest to run out of all the algorithms tested in this work. It is, thus, difficult to tune the algorithm's parameters. The best result is achieved on the raw dataset when using the following parameters: *minPts* = 7, *min_cluster_size* = 5, *xi* = 0.05, *max_eps* = 10. Similarly to the DBSCAN algorithm, SCs are negative for all tested parameters. However, this is not a good cluster validity indication since the SC does not perform well on clusters of arbitrary shape.

As previously mentioned, OPTICS performs similarly to the DBSCAN algorithm in dataset 2. However, its SC and DB indexes are worse, as seen in Table 4. The algorithm achieves its best classification results with an SC of -0.57 and a DB index of 2.1. These values suggest that the goal of maximising intra-cluster similarity and minimising inter-cluster similarity is unsuccessful. Through visual inspection of the clustering in Figure 8d, one can observe that the algorithm manages to detect the domes in the church as spherical geometric structures. Thus, the algorithm can produce relatively meaningful clusters based on the additional geometric features in the dataset. It is imperceptible which regions are to be kept or discarded from the CH site, as there are no distinct clusters beside the spherical structures. All other partitions are visualised as groups of points scattered around the point cloud and bear no semantics. The classification result looks similar for all parameters selected – points of discard scattered around the point cloud in the regions that should classify as *discard* but not enough points classify as *discard* for one to make out any distinct objects and structures. The F1 and precision scores are 58% and 82%, respectively, indicating that most of the predicted *discard* clusters are correctly classified, but not enough are classified as *discard*. Hence, the recall accuracy is significantly lower than precision.

The OPTICS algorithm is incapable of clustering dataset 3 or producing any result from it. Thus, the dataset is voxel downsampled by 0.19 and fed into the OPTICS clustering algorithm. Even using an increasingly downsampled dataset, the algorithm takes over an hour to run and achieves poor classification results. In the best case, OPTICS manages to detect the exterior noise and spots in the scaffolding, but no fine detail inside the church, as *discard* (see Figure 9d). The algorithm performs poorly on the dataset as it cannot produce clusters that can semantically classify the point cloud into *keep* and *discard*.

	K-means	DBSCAN	OPTICS	Mean-shift
Raw Dataset	**0.30**	-0.71	-0.60	0.19
Dataset 2	**0.55**	-0.35	-0.57	0.08
Dataset 3	**0.27**	-0.66	-0.62	0.179

Table 3: Table comparing the SCs across four clustering algorithms and three datasets. The optimal SC for each dataset is made bold.

5.5 Discussion

The mean-shift clustering algorithm performs optimally out of the density-based clustering algorithms in all three datasets regarding cluster validity and classification evaluation metrics (see Figures 3a, 3b, 3c and 4, and Tables 3 and 4). It achieves an SC and DB index close to k-means clustering on the raw dataset and dataset 3. In addition, its classification evaluation metrics are significantly higher than k-means on the raw dataset and dataset 2 for precision, recall, F1 score and IOU score, achieving results close to 100%, and its MSE is closer to 0. Thus, the mean-shift clustering algorithm produces clusters on the raw dataset and dataset 2 that are more semantically meaningful than those produced by the baseline k-means algorithm.

DBSCAN performs worse than expected on all three datasets since other research papers in the field deem it a viable solution to detecting arbitrary-shaped clusters in high-density spaces, such as point clouds of CH sites. This result may be due to the selected *minPts* and *eps* parameters, and future work can investigate improved parameter selection methods. However, DBSCAN was tested with multiple combinations of hyperparameters, implying that DBSCAN is not the optimal clustering algorithm to classify point clouds into *keep* and *discard*. It produces the worst classification results on dataset 2, despite the selection of density-based features in that dataset.

OPTICS is the most inefficient clustering algorithm implemented in this work. It performs slowly on large datasets, such as point clouds, making parameter tuning difficult. In addition, it achieves poor classification results, such that one cannot distinctly discern the *keep* and *discard* regions. Researchers should avoid this clustering algorithm as a point cloud classification approach.

K-means outperforms all the density-based clustering algorithms on dataset 3, based on both clustering and classification evaluation metrics, as seen in Figure 3c, Figure 4 and Table 3. Visually, k-means' classification of dataset 3 resembles the actual ground truth labels (see Figure 9a). The density-based algorithms perform poorly on dataset 3, as they are only capable of classifying noise as *discard* and fail to classify discernible structures in the point cloud, such as the scaffolding, as *discard*. Mean-shift performs optimally out of the density-based algorithms on dataset 3 but can only classify noise and parts of the scaffolding as *discard*. One can conclude that density-based algorithms cannot partition dataset 3 into semantically meaningful clusters that can classify as *keep* and *discard*.

The density-based algorithms do not produce more semantically meaningful clusters than those produced by the baseline k-means algorithms based on the SC and DB index. K-means achieves the optimal SC and DB index across all three datasets. OPTICS and DBSCAN consistently achieve SCs closer to -1 and DB indexes greater than k-means on all datasets, suggesting that the clusters are not well-separated or dense. However, the SC and DB index metrics are unsuitable cluster validity metrics for evaluating density-based clustering algorithms. These metrics are distance-based and, thus, perform well on partitional algorithms, like k-means, but do not perform well on arbitrary-shaped clusters produced by density-based methods.

Moreover, the SC and DB index metrics are poor determiners of k for k-means clustering on points clouds. They achieve optimal results for low values of k. However, for small numbers of clusters, the classification prediction labels the entire point cloud as *keep*. By contrast, with high values of k of the order 1000, k-means produces clustering that effectively and accurately classifies the point cloud into *keep* and *discard*.

Although increasing the number of clusters improves the classification results for most of the clustering methods, having large numbers of clusters is undesirable for the task of point cloud cleaning. Ideally, there should be as few resulting clusters as possible - one large *keep* and one large *discard* cluster - so that a human operator can manually select the *discard* cluster(s) and remove it from the point cloud. Having thousands of clusters or more, as illustrated in Figures 6a, 6b and 6c in the Appendix, is ineffective as someone has to remove each *discard* one. The idea is to minimise the amount of human selection required. Thus, the practical utilisation of the density-based algorithms we implemented is not as viable as the results suggest due to the number of produced clusters.

ML-generated and geometric human-interpretable features affect clustering algorithms' performance at producing semantically-meaningful clusters. K-means successfully classifies the point cloud into *keep* and *discard* clusters on the raw dataset and dataset 3 but produces poor classification results on dataset 2. Thus, the k-means clustering algorithm does not perform well on datasets with geometric features.

6 Conclusions

Constructing 3D models from TLS scans is challenging because point clouds, particularly CH site ones, contain unwanted points, such as noise, people, scaffolding and foliage, which one must remove. We need to automate the tedious point cloud cleaning process and create an algorithm to accurately and efficiently predict the class labels of points within a point cloud. We extract features through CloudCompare and PointNet++ to create three datasets from the point cloud of a registered church CH site with varying feature sets: raw, human-interpretable, and ML-generated features. This work evaluates the performance of the OPTICs, mean-shift and DBSCAN density-based clustering algorithms at partitioning the datasets into semantically meaningful clusters that can classify using binary classification labels of *keep* and *discard*. Moreover, we compare these algorithms against a baseline k-means partitional clustering algorithm and deduce several conclusions from the results.

The DBSCAN, mean-shift and OPTICS density-based clustering algorithms do not produce more semantically meaningful clusters than k-means, the baseline partitional clustering algorithm, based on the SC and DB index cluster validity metrics. However, these metrics are distance-based and are, therefore, insufficient for evaluating density-based clustering algorithms which produce arbitrary-shaped clusters. Visual inspection and classification evaluation metrics prove more suitable for evaluating the clustering and classification results of the algorithms.

Clustering algorithms can augment the process of point cloud cleaning by automating it and improving its efficiency. The algorithms can, therefore, reduce the need for manual and labour-intensive cleaning. In particular, density-based clustering algorithms can partition point clouds into clusters that can be grouped into semantically meaningful clusters and classified as *keep* and *discard*. However, these algorithms produce an excessive number of partitions which is less desirable for the classification task than having one for each class label. A human operator must manually select and remove each *discard* cluster, which is inefficient.

DBSCAN and OPTICS cannot accurately classify the entire point cloud into *keep* and *discard*. They are suboptimal and perform worse than the partitional k-means algorithm in classification and clustering evaluation metrics. However, they can still produce clusters that can classify using binary classification labels. One can use their classification results as a starting point for point cloud cleaning, thereby speeding up the process. The mean-shift and k-means clustering algorithms can accurately and efficiently produce clusters on two datasets that can classify a CH site point cloud into regions that are part of the CH and those that are not. Thus, the mean-shift clustering algorithm is the optimal density-based algorithm for the point cloud classification task.

This work has several limitations, including the dataset sizes, the algorithms' speed and the available hardware and software resources. Thus, we utilise voxel downsampling to reduce the dataset sizes before inputting them into the clustering algorithms.

Future work can investigate improved parameter selection methods for the clustering algorithms to achieve better clustering and classification results. In addition, it can analyse the effect of different features on the performance of clustering algorithms and aim to select features that enable the algorithms to perform optimally.

7 Acknowledgements

Acknowledgement goes to Patrick Marais and Luc Hayward for their invaluable guidance throughout the research project. Thank you for always being available to answer questions, provide advice or discuss ideas.

References

[1] Charu C. Aggarwal. 2015. (1 ed.). Springer, New York. https://doi.org/10.1007/978-3-319-14142-8

[2] Harith Aljumaily, Debra F Laefer, and Dolores Cuadra. 2017. Urban Point Cloud Mining Based on Density Clustering and MapReduce. *Journal of Computing in Civil Engineering* 31, 5 (2017), 04017021. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000674

[3] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. 1999. OPTICS: Ordering Points to Identify the Clustering Structure. *SIGMOD Rec.* 28, 2 (jun 1999), 49–60. https://doi.org/10.1145/304181.304187

[4] Ira Assent. 2012. Clustering high dimensional data. *WIREs Data Mining and Knowledge Discovery* 2, 4 (2012), 340–350. https://doi.org/10.1002/widm.1062

[5] Paolo Brivio. 2022. *Visual Computing Lab.* Visual Computing Lab. Retrieved May 22,2022 from http://vcg.isti.cnr.it/

[6] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise.. In *kdd*, Vol. 96. 226–231.

[7] Roberto Ferrara, Salvatore G.P. Virdis, Andrea Ventura, Tiziano Ghisu, Pierpaolo Duce, and Grazia Pellizzaro. 2018. An automated approach for wood-leaf separation from terrestrial LIDAR point clouds using the density based clustering algorithm DBSCAN. *Agricultural and Forest Meteorology* 262 (2018), 434–444. https://doi.org/10.1016/j.agrformet.2018.04.008

[8] César Ferri, José Hernández-Orallo, and R Modroiu. 2009. An experimental comparison of performance measures for classification. *Pattern recognition letters* 30, 1 (2009), 27–38.

[9] Ahmad Gamal, Ari Wibisono, Satrio Bagus Wicaksono, Muhammad Alvin Abyan, Nur Hamid, Hanif Arif Wisesa, Wisnu Jatmiko, and Ronny Ardhianto. 2020. Automatic LIDAR building segmentation based on DGCNN and euclidean clustering. *Journal of Big Data* 7, 1 (2020), 1–18.

[10] E Grilli, EM Farella, A Torresani, and F Remondino. 2019. Geometric feature analysis for the classification of cultural heritage point clouds. In *27th CIPA International Symposium "Documenting the past for a better future"*, Vol. 42. 541–548.

[11] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. 2020. Array programming with NumPy. *Nature* 585, 7825 (Sept. 2020), 357–362. https://doi.org/10.1038/s41586-020-2649-2

[12] Yasen Jiao and Pufeng Du. 2016. Performance measures in evaluating machine learning based bioinformatics predictors for classifications. *Quantitative Biology* 4, 4 (2016), 320–330.

[13] Amin Karami and Ronnie Johansson. 2014. Choosing DBSCAN parameters automatically using differential evolution. *International Journal of Computer Applications* 91, 7 (2014), 1–11.

[14] Yangguang Liu, Yangming Zhou, Shiting Wen, and Chaogang Tang. 2014. A strategy on selecting performance metrics for classifier evaluation. *International Journal of Mobile Computing and Multimedia Communications (IJMCMC)* 6, 4 (2014), 20–35.

[15] Patrick Marais, Matteo Dellepiane, Paolo Cignoni, and Roberto Scopigno. 2019. Semi-automated Cleaning of Laser Scanning Campaigns with Machine Learning. *ACM Journal on Computing and Cultural Heritage* 12, 3 (2019), 1–29. https://doi.org/10.1145/3292027

[16] Julio-Omar Palacio-Niño and Fernando Berzal. 2019. Evaluation Metrics for Unsupervised Learning Algorithms. *CoRR* abs/1905.05667 (2019), 1–9. arXiv:1905.05667 http://arxiv.org/abs/1905.05667

[17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[18] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. 2017. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. https://doi.org/10.48550/ARXIV.1706.02413

[19] Heinz Rüther, Christoph Held, Roshan Bhurtha, Ralph Schroeder, and Stephen Wessels. 2012. From Point Cloud to Textured Model, the Zamani Laser Scanning Pipeline in Heritage Documentation. *South African Journal of Geomatics* 1, 1 (Jan. 2012), 44–59.

[20] Jie Shan and Aparajithan Sampath. 2008. Building extraction from LiDAR point clouds based on clustering techniques. In *Topographic laser ranging and scanning: principles and processing*. CRC press, Boca Raton, FL, 421–444.

[21] Editorial Team. 2018–2021. *Point Cloud.* Open3D. http://www.open3d.org/docs/release/tutorial/geometry/pointcloud.html

[22] Alaa Tharwat. 2021. Classification assessment methods. *Applied Computing and Informatics* 17 (2021), 168–192. Issue 1.

[23] Marj Tonini and Antonio Abellan. 2014. Rockfall detection from terrestrial LiDAR point clouds: A clustering approach using R. *Journal of Spatial Information Science* 1, 8 (2014), 95–110.

[24] Kuo-Lung Wu and Miin-Shen Yang. 2007. Mean shift-based clustering. *Pattern Recognition* 40, 11 (2007), 3035–3052. https://doi.org/10.1016/j.patcog.2007.02.006

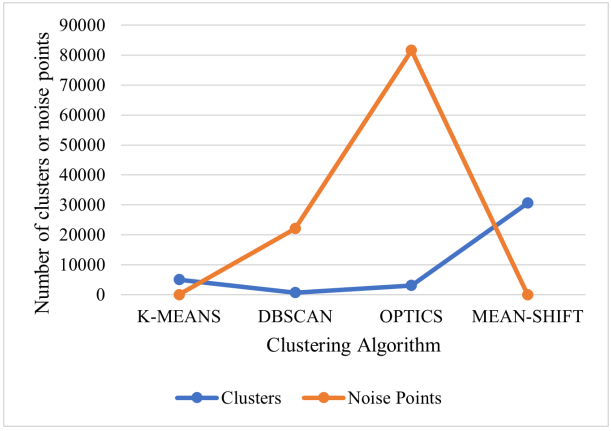
[25] Zhang Ximin, Wan Wanggen, Xiao Li, and Ma Junxing. 2014. Mean shift clustering segmentation and RANSAC simplification of color point cloud. In *2014 International Conference on Audio, Language and Image Processing*. 837–841. https://doi.org/10.1109/ICALIP.2014.7009912

A Supplementary Information

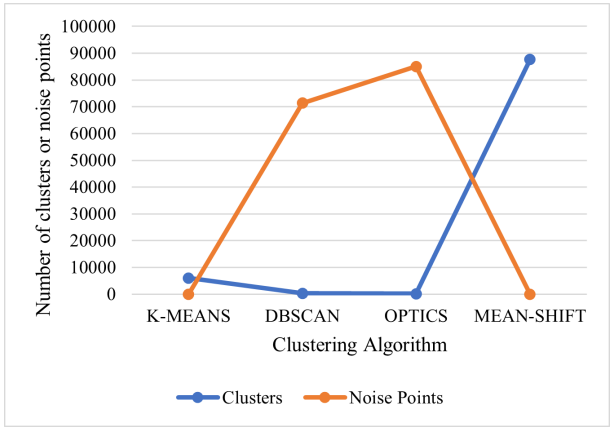
A.1 Tables and Graphs

	K-means	DBSCAN	OPTICS	Mean-shift
Raw Dataset	**0.84**	1.35	1.26	**0.84**
Dataset 2	**0.41**	1.21	2.10	n/a
Dataset 3	**1.10**	1.54	1.15	1.19

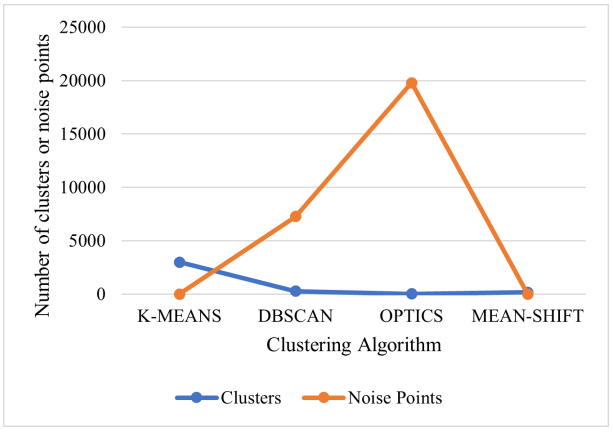
Table 4: Table comparing the DB indexes across four clustering algorithms and three datasets. The optimal DB index for each dataset is made bold. *n/a* signifies that the DB index could not be computed because it utilised too much RAM.



(a)



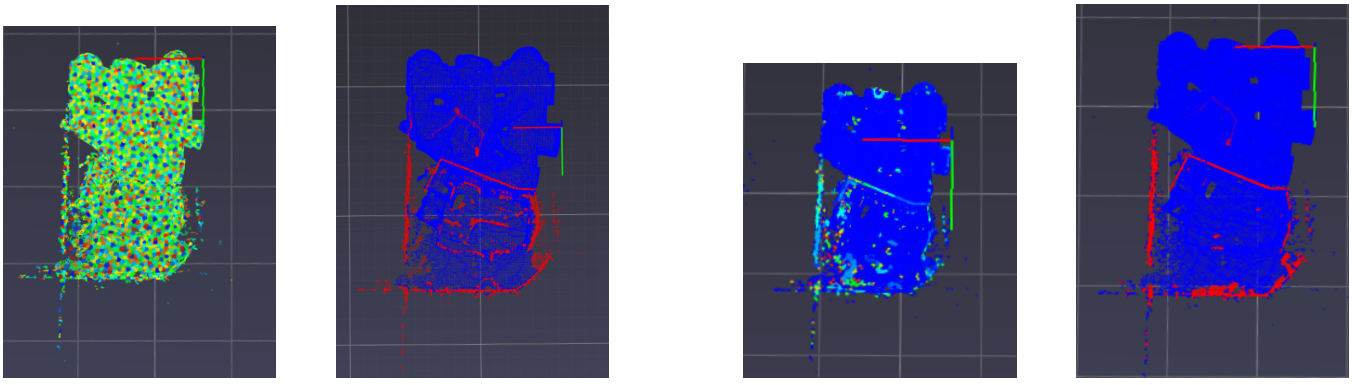
(b)



(c)

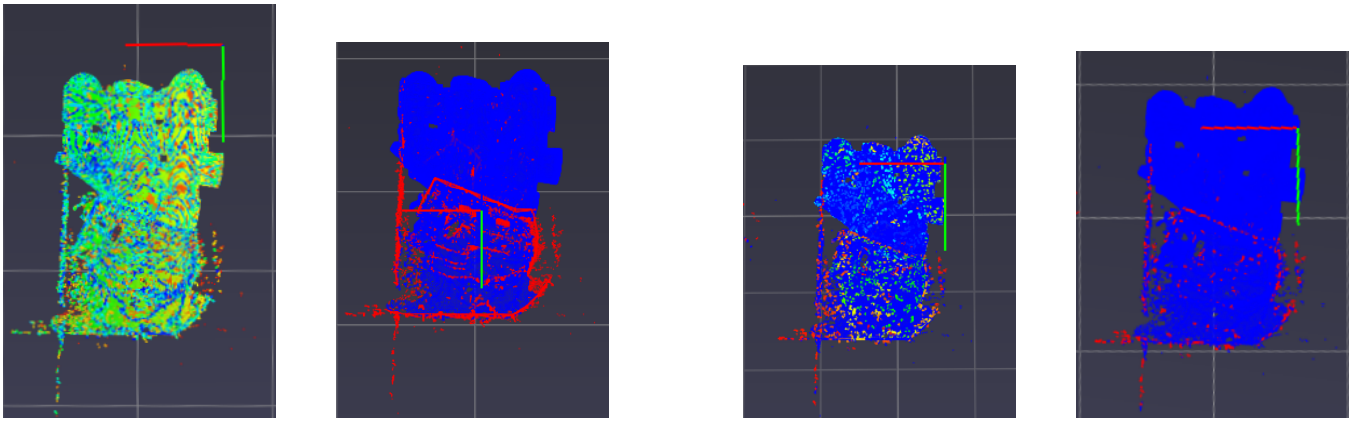
Figure 6: Three Line Graphs Illustrating the number of clusters and number of noise points computed by each of four clustering algorithms on (a) the Raw Dataset, (b) Dataset 2, and (c) Dataset 3

A.2 Visualised Clustering and Classification Results



(a) K-Means Clustering

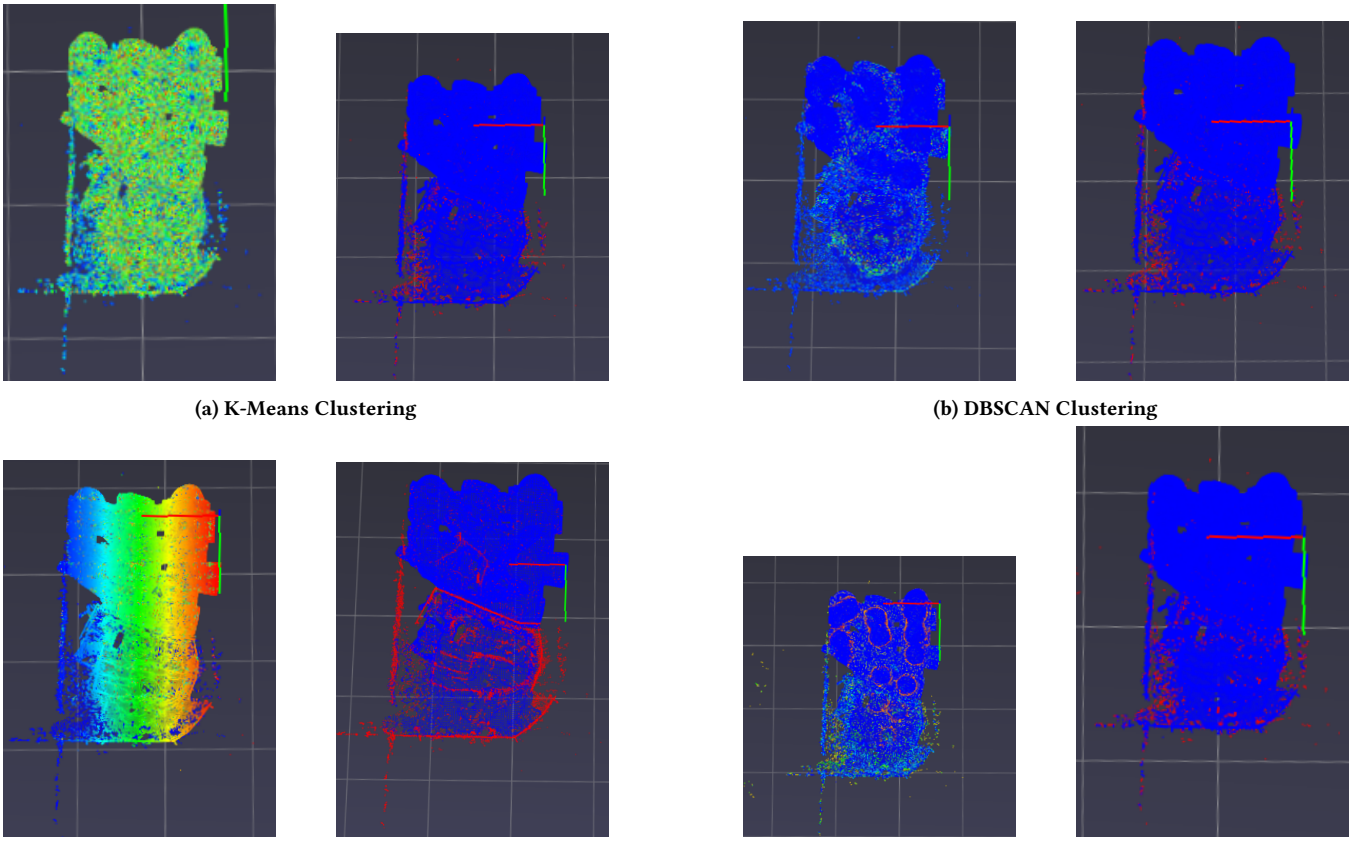
(b) DBSCAN Clustering



(c) Mean-Shift Clustering

(d) OPTICS Clustering

Figure 7: Clustering (left image of every figure) and Classification (right image of every figure) Results of four clustering algorithms on the Raw Dataset



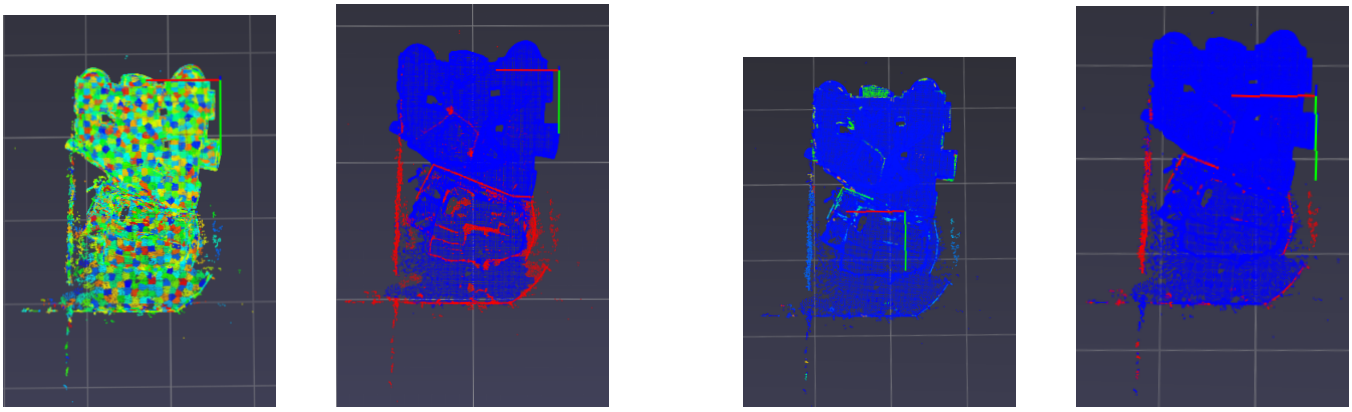
(a) K-Means Clustering

(b) DBSCAN Clustering

(c) Mean-Shift Clustering

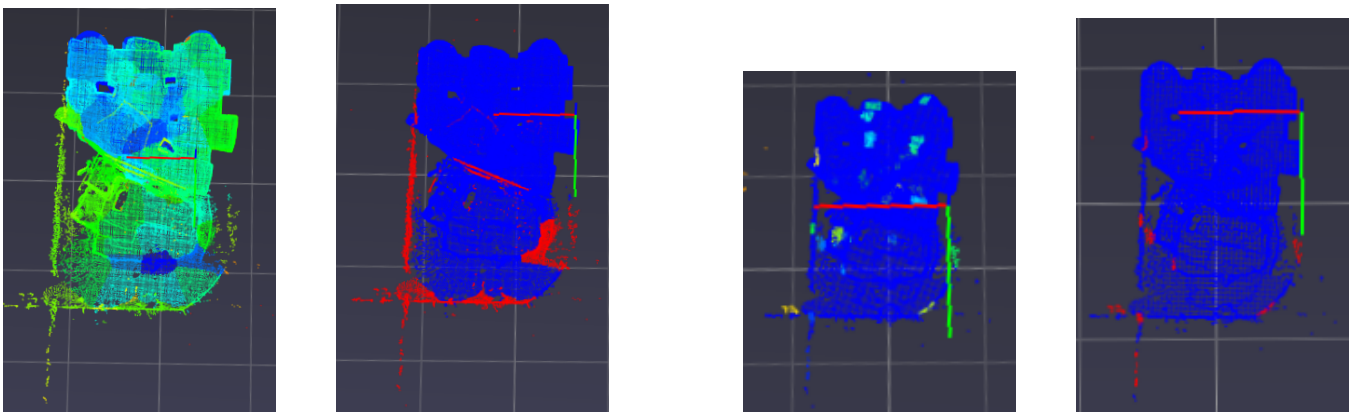
(d) OPTICS Clustering

Figure 8: Clustering (left image of every figure) and Classification (right image of every figure) Results of four clustering algorithms on Dataset 2



(a) K-Means Clustering

(b) DBSCAN Clustering



(c) Mean-Shift Clustering

(d) OPTICS Clustering

Figure 9: Clustering (left image of every figure) and Classification (right image of every figure) Results of four clustering algorithms on Dataset 3