Automatic Segmentation of Structural Elements from 3D Point Cloud: The Case of Occupied Multi-Storey Building

ジョラム スタンスラウス ンティヤクンゼ

出版情報：Kyushu University, 2023, 博士（工学）, 課程博士
バージョン：
権利関係：

Automatic Segmentation of Structural Elements from 3D Point Cloud

The Case of Occupied Multi-Storey Building

(占有されている高層ビルの 3D 点群からの構造要素の自動セグメンテーション)

JORAM STANSLAUS NTIYAKUNZE
ジョラム スタンスラウス ンティヤクンゼ

September, 2023

Abstract

Point clouds obtained from built assets have become increasingly popular in various tasks including heritage conservation, renovations, building maintenance, progress monitoring in construction sites, and urban planning. One of the desirable features of point clouds is the entailment of a highly detailed and accurate geometric representation of the acquired scene or object, which makes it suitable for many. However, raw point clouds contain unordered 3D points and hence lack semantic information about objects, which can limit their usability for further applications. In addition, point clouds acquired from the existing buildings contain diverse and large amounts of occlusions which further complicates its use. Thus, it is essential to extract relevant information about the building features from the point clouds about the building features.

The main goal of this thesis is to propose a robust method of segmenting structural elements from point clouds without being affected by the noise or occlusions present in the environment. The contribution lies in three major aspects: the development of octree-based infrastructure to extract the important features from a point cloud; the identification and merging of the candidate surface patches for the structural elements; and the classification of the identified patches into segments of generic structural elements.

The proposed segmentation method starts with extracting small-sized surface patches from the created voxels, and empirically joining the adjoining patches sharing similar local features. This process returns patches including disconnected patches due to the presence of occlusions and noise in the scene. In the second part, a robust technique is developed to merge the disconnected patches that represent similar surfaces of structural elements. The third part includes classifying and segmenting the merged into class elements which are: floor slabs, floor beams, walls, and columns. A novel classification method was used utilizing the concept of spatial dependency of elements in the structural systems of buildings.

The proposed method is tested on a large point cloud with high levels of occlusions and noise. The evaluation of test results has indicated a good general performance, especially for segmenting the floor slabs where 100% was scored on all the quantitative metrics. The floor beams and walls have shown promising performance with rates above 60% for precision and 80% for recall. The comparison tests were done against the state-of-the-art segmentation methods including the deep learning method where the proposed method has shown relatively good results, particularly for beams that normally are difficult to segment. The performance of the proposed segmentation was highly affected by the lack of a good number of paired planar patches due to occlusion(s) located on one side of the pair-set. Other limitations are discussed and potential areas for improving the study are narrated.

Acknowledgment

I would like to express my deep gratitude to my academic supervisor, Assistant Professor Tomo Inoue, for his invaluable guidance and unwavering support throughout the journey of my doctoral research. His extensive knowledge and expertise in building technology have been instrumental in shaping this thesis. His dedication to academic excellence and insightful feedback have immensely contributed to the development and refinement of this study.

I would also like to extend my gratitude to my co-supervisors, Professor Kenichi Tanoue and Professor Tomokazu Yoshioka. Their profound knowledge and experience in architectural and structural designs of have been pivotal in guiding the completion of this research. Their valuable insights, constructive criticism, and thought-provoking discussions have significantly enriched the quality and depth of this study.

I would also like to acknowledge the support the Shigeru Aoki Architects and Associates provided as the industry collaborator for this research. This thesis is the outcome of the case study involving an occupied building that was commissioned for design and supervision by Shigeru Aoki Architects and Associates. I am truly grateful to Shigeru Aoki Architects and Associates for granting me permission to access the building and handing-out of as-built drawings necessary for conducting this research.

Furthermore, I would like to acknowledge the support and encouragement received from the academic mentors and research colleagues. Their valuable inputs, technical assistance, and shared experience, and discussions have been instrumental in overcoming challenges and ensuring the smooth progress of this research.

Last but not least, I would like to express my heartfelt gratitude to my family, friends, and loved ones for their constant encouragement, understanding, and patience throughout this arduous journey. Their unwavering support has been a source of motivation and inspiration in the successful completion of this study.

Table of Contents

List of Abbreviations

List of Figures

List of Tables

Chapter 1. Introduction

1.1. Motivation

1.2. Objectives and contributions

1.3. Research questions

1.4. Related work

1.4.1. Spatial subdivision of 3D point clouds

1.4.2. Geometric fitting in 3D Point Clouds

1.4.3. Methods for filtering noise in point clouds

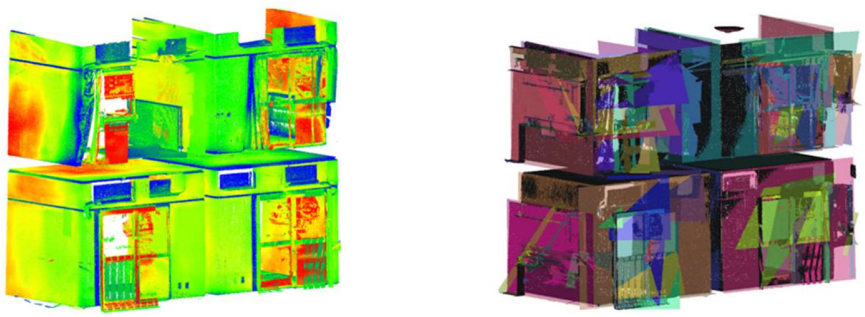
1.4.4. Segmentation methods for point clouds

1.5. Structure and organization

1.1. Motivation

Segmentation of structural elements from 3D point clouds is a crucial task in various applications, including reconstruction of 3D man-made infrastructures, progress monitoring in construction sites, change detection in urban scenes, and building inspection. The point cloud captured by 3D laser scanners, photogrammetry, or video-grammetry techniques, provides high-detailed and accurate geometric information about the scanned environment (Vincke et al., 2019; Q. Wang & Kim, 2019). The measured 3D points have spatial coordinates of geometric surfaces and topologies which can be resourceful in various applications such as 3D remodeling, urban planning, and virtual reality. However, the raw point cloud data contains a vast amount of unorganized and noisy points (Poux & Billen, 2019), making it challenging and time-consuming to extract meaningful information for subsequent processing such as object detection, classification, and segmentations (Hajian & Becerik-Gerber, 2010).

The research on point cloud segmentation has gained significant attention in recent years, and various techniques have been proposed to address its efficiency and overall accuracy in different environments. Conventional methods often rely on handcrafted features, such as surface normal, curvature, or color, to perform classification and segmentation which includes geometric fitting of underlying shapes in point clouds such as planar, circular, and spherical. Some of the widely used algorithms for geometric fitting in 3D point clouds are: Random Sample Consensus (RANSAC), 3D-Hough transform (3D-HT) (Deschaud, 2010; Maalek et al., 2015; Schnabel et al., 2007), and region-growing (Deschaud, 2010). Nonetheless, implementing these algorithms has been demonstrated to be ineffective and expensive in terms of time and computation, when applied to large point clouds with a greater degree of noise and clutter (Vo et al., 2015). Specifically, the RANSAC method frequently generates erroneous (spurious) planes that do not match any object in the building scene as illustrated in Figure 1.1. In addition, these methods have been reported to underperform in capturing complex and heterogeneous information in the point cloud data (Vo et al., 2015). To solve this problem, much attention has been devoted to deep learning methods of segmenting point cloud data.



(a) (b)

Figure 1. 1 Geometric fitting on 3D point cloud using RANSAC: (a) Original point cloud (b) Result after plane fitting using RANSAC with erroneous planes representing in different RGB colors.

Deep learning-based methods, especially convolutional neural networks (CNNs), have shown promising results in point cloud segmentation. CNNs can analyze hierarchical attributes of the 3D point cloud data and capture both local and global features of building scenes. One of the pioneering works in this direction is PointNet (Qi, Su, et al., 2017), which treats point clouds as a set of unordered points and directly applies multilayer perceptron (MLPs) to each point feature. Another state-of-the-art work is PointCNN (Y. Li et al., 2018) which introduces a novel point convolution operation that respects the local permutation invariance of the point cloud data. Several subsequent works have extended the PointNet and PointCNN frameworks to address specific challenges in point cloud segmentation.

However, the deep learning methods still have some drawbacks in the use of point clouds. They demand large numbers of training datasets, which can be computationally expensive, and labeled data are limited especially for complex structures (J. Zhang et al., 2019). In the study conducted in (Poux & Billen, 2019), the PointNet has demonstrated low performance in classifying the standard building components such as ceilings, floors, and walls when no color attributes are used despite the great performance it has shown in classifying other indoor objects such as chairs, tables, and bookshelves.

The lack of instance segmentation in indoor point clouds is also a significant drawback for deep learning techniques (J. Zhang et al., 2019). Without this information, it can be challenging for applications such as 3D remodeling and structural analysis which prefer accurate and detailed classifications of objects within the point cloud as illustrated in Figure 1.2. Thereon, it is often enforced to perform additional processes that require prior knowledge about building structures to extract distinct details and high-level geometric representation of building features (Xu, 2019).

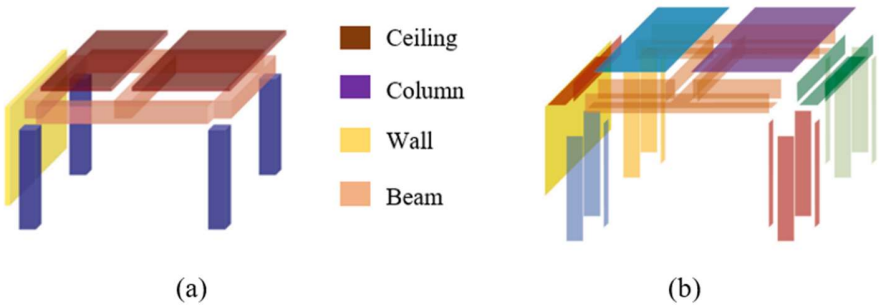


Figure 1. 2 Segmentation result of the 3D model (Hyunsoo & Changwan, 2021): (a) Semantic segmentation with moderate classification (b) Instance segmentation with detailed classification.

On that account, several studies have been conducted attempting to use the contextual knowledge of structural and architectural forms of buildings in the segmentation process. Son & Kim (2017) developed an approach to segment the 3D points into meaningful parts comprising floors, columns, walls, girders, beams, and slabs using the local concave and convex connectivity between the structural elements, and Hyunsoo & Changwan (2021) exploited the similar approach to perform an automated as-built remodeling for point cloud with the limited amount of missing data. Maalek et al. (2015) introduced a robust segmentation method that performs complete linkage of planar and linear features associated with columns and reinforcements for construction site progress monitoring and structural dimension compliance control. Hamid-Lakzaeian (2020)

developed a multi-planar algorithm as a means of distinguishing the structural from the non-structural elements for a building with multi-planar facades. The proposed algorithm considered the positions of clusters of points with respect to their neighbors to segment the principal facades of architecturally ornate, multi-planar buildings. Although these methods have shown success, they have mostly been tested in construction-related scenarios where the building features are relatively more exposed and visible to sensors compared to those of existing and occupied buildings.

Occupied buildings typically offer limited visibility of permanent structures due to obstructions caused by the presence of secondary objects in the scene such as fittings, furniture, and fixtures. These occlusions may have similar dimensions and orientations to standard building elements such as walls, slabs, and beams (Xiong et al., 2013), as shown in Figure 1.3. The occlusions can result in point clouds missing critical data, which can lead to inaccurate segmentations due to disconnected surface patches, particularly during the geometric fitting process (Hyunsoo & Changwan, 2021).

(a) (b) (c)

Figure 1. 3 Various forms of occlusions are present in an occupied building: (a) large wardrobe concealing one face of the wall; (b) upper cabinet resembling a short-span beam; (c) door leaning and occluding a wall surface.

From the aforementioned research, it can be seen that three major tasks should be addressed in the segmentation process: (1) Geometric fitting in a large point cloud; (2) Classifying the structural elements from the noise and occlusions present in the point cloud; (3) Segmenting the detected structural elements from one another. To address the mentioned tasks, this study proposes a segmentation method that overcomes the challenge of missing data caused by noise and occlusions while robustly segmenting structural points from the nearby non-structural objects. This proposed method involves identifying principal pairs of detected planar patches and merging the remotely disconnected patches representing the same structural surfaces. By incorporating patch-pairing, we can assimilate the surface symmetry of common structural components into the segmentation pipeline.

The method proposed in this thesis can manage point clouds with varying degrees of missing data, as well as handling point clouds of different scales and sample distributions. A case study is conducted on a multistory building with planar surfaces, the method applied in the study can be adopted in segmenting other concrete-based structures such as bridges, tunnels.

1.2. Objectives and contributions

In this thesis, we present a framework for segmenting structural elements from a multi-storey building using 3D point cloud with novel algorithms and methods in the fields of detection, classification, and segmentation of point cloud. We aim to develop robust methods and techniques to segment structural elements from large point clouds containing high amount of secondary and temporary objects which usually complicates the segmentation process.

The tasks included in this thesis consists of three main parts. The first part is about the repartitioning of the point cloud into a more simplified and manageable units (voxels) that can allow easier processing of point cloud and deduction of the underlying features in the neighborhood of points. this is done by spatially decomposing the point cloud into well-defined voxels for reliable estimation and computation of the underlying local features in the point clouds. This helps to alleviate the problem large size of point cloud while maintaining the attributes of the original point cloud. The second part involves the extraction of candidate planar patches associated with structural elements amidst the noise and clutters present in the neighborhood. The proposed technique in this part helps to counter the challenge of the present secondary and temporary objects in the point cloud during the segmentation pipeline. The final part deals with segmentation of structural elements from the detected planar patches obtained in the preceding part. In this part, we introduce novel techniques for classifying the detected planar surfaces into well-defined structural elements using the spatial dependency and topology of elements in a structural system of buildings.

In Figure 1.4, we illustrate an overview of the proposed methods, containing former works from other researches and our proposed method and their developing workflow.

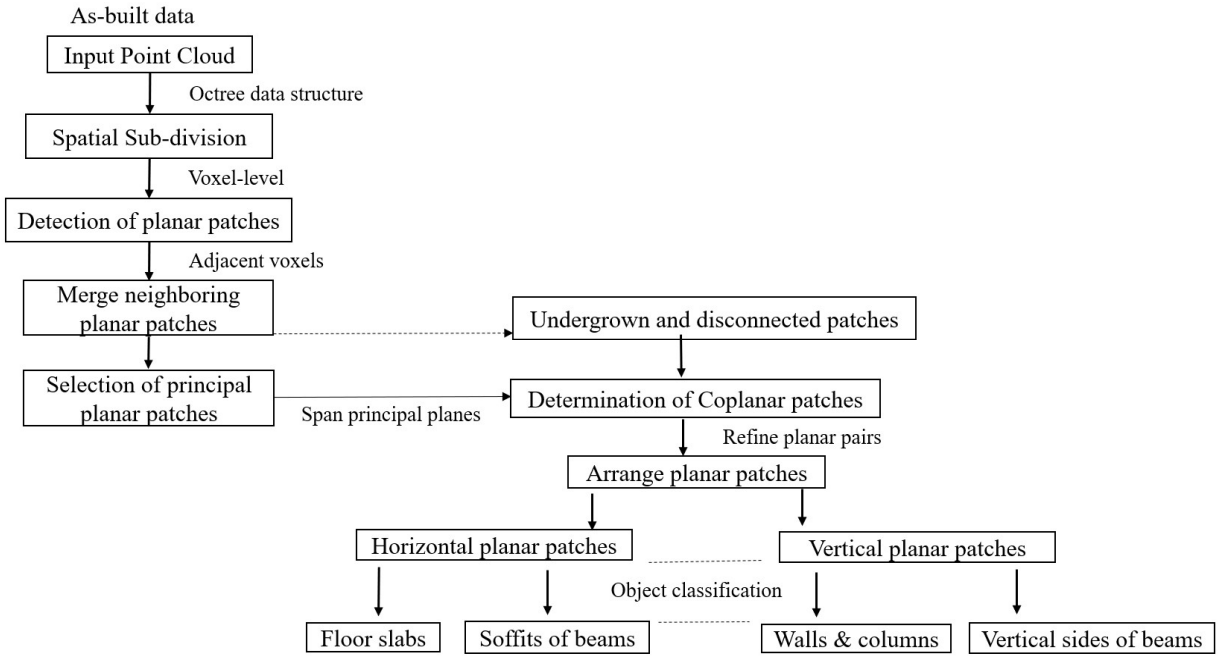


Figure 1. 4 Overview of the proposed segmentation method.

1.3. Research questions

More specifically, in this work, the proposed segmentation method associated to large point cloud scene with high degree of noise, clutters and is to answer the following questions, which have not yet been considered or fully addressed in state-of-the-art literature:

(a) How to efficiently and accurately detect local features associated with planar surfaces from a 3D point cloud that contains a high amount of noise and artifacts?
(b) How to effectively determine if the remote and disconnected planar patches in a point cloud belong to the same surface of a particular structural element?
(c) What are the necessary aspects to consider in segmenting structural elements from the non-structural elements and secondary objects present in the point cloud scene?

For finding answers to the aforementioned research questions, before introducing our objectives and contributions, we first provide a detailed survey and review of related work in the following section

1.4. Related work

Numerous researchers have studied about segmentation works using point clouds. According to the three main tasks mentioned in the section 1.1, we will provide detailed reviews and discussions concerning methods and algorithms relating to spatial subdivision of 3D point clouds, geometric fitting, noise handling in point in point cloud and state of the art algorithms developed to segment building features from a point cloud.

1.4.1. Spatial subdivision of 3D point clouds

Spatial subdivision is a process of partitioning a point cloud into smaller subsets or regions based on their spatial locations. This process is commonly used in point cloud processing for applications such as data compression, segmentation, and visualization.

There are several methods available for spatial subdivision of point clouds, and the two most common methods are octrees, and k-d trees. These methods differ in their complexity, efficiency, and accuracy, and are selected based on the specific application requirements.

(a) Octree

Octree is a hierarchical data structure that is commonly used for spatial subdivision of point clouds. The octree subdivides the space recursively into eight equal-sized cubes, or octants, until a termination condition is met. Each octant can either contain points or be empty, and if it contains points, it can be further subdivided into eight smaller octants.

Octree structures are particularly useful for point clouds with non-uniform densities or complex shapes, as they adapt to the local density of points and can handle irregular shapes. They also allow efficient querying of points within a given region, which is useful for tasks such as point cloud segmentation or feature extraction (L. Han et al., 2021; Xu et al., 2021).

(b) K-d tree

K-d tree is a data structure that is commonly used for spatial subdivision of point clouds. The k-d tree is a binary tree, where each node corresponds to an axis-aligned hyperplane that partitions the space into two half-spaces. The left subtree of the node represents the points to the left of the hyperplane, while the right subtree represents the points to the right of the hyperplane. The hyperplane direction is determined by associating every node with one of the k dimensions, and the hyperplane is perpendicular to that dimension's axis (Bentley, 1975).

Even though both methods are widely used for spatial subdivision of point clouds, however, octree is mostly favored. The advantage of using octree is that it can handle point clouds with varying densities and shapes more efficiently than k-d tree. Another advantage of octree over the k-d tree is in its search for the nearest neighbor because the hierarchical structure of octree allows for efficient traversal of cells at different levels of detail, reducing the search space and improving the processing speed (Limberger & Oliveira, 2015).

1.4.2. Geometric fitting in 3D Point Clouds

Geometric fitting of point clouds is the process of finding a mathematical model that approximates the shape of a set of points in 3D space. This is a fundamental task in computer vision and robotics, where it is used for applications such as object recognition, pose estimation, and scene reconstruction. It is achieved by evaluating the connected points according to their geometric features (e.g., spatial positions and normal vectors) in a local or global scale through certain geometric models. Points meeting the criteria of fitting the same geometric model (either in the spatial or parametric domain) are extracted from the point cloud as single individual segment.

In 3D point clouds, the three most widely used methodologies to extract geometric features are Random Sample Consensus (RANSAC), 3D-Hough transform (3D-HT), and region-growing.

(a) RANSAC

RANSAC is a popular algorithm for geometric fitting in point clouds, especially when the data contains a significant amount of noise or outliers. The RANSAC algorithm works by randomly selecting a subset of points from the point cloud and using them to fit a model. The model is then evaluated on the remaining points in the cloud, and those that are consistent with the model (i.e., within some predefined threshold) are considered as inliers (Fischler & Bolles, 1981). The algorithm is repeated for a fixed number of iterations, and the best model is chosen based on the number of inliers it generates. The RANSAC algorithm is effective in fitting various types of geometric models to point clouds, such as lines, planes, circles, and spheres. The algorithm can be easily extended to handle more complex models by modifying the model-fitting function and the threshold for inlier detection.

The RANSAC is used to classify building planar features from a 3D point cloud (Hamid-Lakzaeian, 2020; Hong et al., 2015) due to its computational efficiency in outlier settings. Several variants of RANSAC exist to optimize execution speed and accuracy in higher outliers. (Hamid-Lakzaeian (2020) Enhanced RANSAC with an octree structure to detect multi-planar building facades, and in (L. Li et al., 2017) the point cloud is subdivided into cells to detect minimal planar

patches. However, RANSAC is highly sensitive to parameter tuning depending on the noise levels (Araújo & Oliveira, 2020).

(b) 3D-Hough transform

Hough transform (HT) is more prominent in detecting parameterized features in linear and circular shapes in 2D compared to 3D datasets (Borrmann et al., 2011). To facilitate the speed and accuracy of HT in 3D settings, in Kultanen et al. (1990) a randomized version of HT was developed where planes and curves are detected by mapping points into one point in Hough space and randomly voting points based on a distance from the candidate primitive. Then, for each point in the point cloud, the algorithm computes the set of parameters that define the primitive that contains that point. The accumulator array is updated at each step to accumulate the evidence for each set of parameters. Once all points have been processed, the accumulator array is searched for peaks, which correspond to the most likely sets of parameters for the primitive. In (Borrmann et al. (2011), different variations of 3D-HT were evaluated to detect planes from a simulated laser scan model, while in (Díaz-Vilariño et al., 2015) HT was used to extract circular and rectangular columns from the projected points in the x-y plane; however, the implementation of 3D-HT is still disfavored due to its overall high computation cost (Borrmann et al., 2011; Vo et al., 2015).

(c) Region growing

Region growing is a technique used in geometric fitting of point clouds to segment a point cloud into regions that belong to a common geometric primitive. The basic idea of region growing is to start with a seed point, and then iteratively grow a region of points that are geometrically similar to the seed point, based on some criteria, such as the closest point, orientation, and surface curvature. If a neighboring point satisfies the similarity criteria, it is added to the growing region, and the algorithm continues to search for more neighboring points that are similar to the region. The algorithm stops when there are no more points that satisfy the similarity criteria or when the region reaches a predefined size or shape.

Region-growing is widely implemented in 3D point clouds to detect geometry features due to its high resilience in contaminated data (Deschaud, 2010; Dimitrov & Golparvar-Fard, 2015; Maalek et al., 2015, 2019; Nurunnabi et al., 2012; Son & Kim, 2017; Xiong et al., 2013). The main limitation of region-growing is the selection of a seeding point that can affect the quality of segmentation (Kang et al., 2020). Also, during clustering, the algorithm can result in the over- or under-segmentation of regions leading to the spontaneous creation of region borders. Due to these problems, Maalek et al. (2015) suggested that a hybrid and clustering integrations can be used to overcome these limitations. In Rabbani et al. (2006), the region-growing was modified with a smoothness-constraint approach where close points with similar surface curvature are merged. This approach is deployed in the point cloud library (PCL) (Rusu & Cousins, 2011) and applied in (Son & Kim, 2017; Xiong et al., 2013) to detect planar patches.

More recently, robust methods of Principal Component Analysis (PCA) have also been applied to segment planes in the presence of a high proportion of outliers. In Maalek et al. (2019), a statistical-based PCA that uses a fast-minimum covariance determinant to extract multiple planar objects from 3D point sets was applied. Nevertheless, the mentioned approaches are yet to solve

the problem of missing data due to extensive occlusions and separate nearly identical structural and non-structural elements in a point cloud scene.

1.4.3. Methods for filtering noise in point clouds

Noise filtration is an important preprocessing step in 3D point cloud processing, which aims to remove unwanted noise and outliers from the data. Noise in 3D point clouds can be caused by various factors, such as sensor noise, registration errors, occlusions, and surface irregularities. Filtration techniques can help to improve the quality and accuracy of the data and make subsequent processing steps more robust and efficient. Different categories have been created to filter noise in point clouds and they can be mainly categorized into two methods: statistical-based and neighborhood-based noise filtering (X. F. Han et al., 2017).

(a) Statistical-based noise filtering

Statistical-based noise filtering is a type of noise filtering technique used in point cloud processing that aims to remove random noise from the data. This method is based on the assumption that the noise in the data is random and can be modeled by a statistical distribution, such as a Gaussian or Laplacian distribution (Rusu & Cousins, 2011).

To perform noise filtering in a point cloud, the distances to neighboring points within a given neighborhood are first calculated, assuming a Gaussian distribution. The mean value, μ, and standard deviation, σ, are then determined based on these distances. Points that are within a selected k number of neighboring points are considered as inliers and retained in the data set, while points that are further away are considered as outliers and excluded from the data set.

There are several statistical-based noise filtering methods available for point clouds, including mean and median deviation filtering. Mean filtering is a simple technique that replaces each point with the average of its neighboring points within a defined neighborhood size. Median filtering is another popular technique that replaces each point with the median value of its neighboring points within a defined window size. Both mean and median filtering can effectively remove random noise from the data, but they may also smooth out important features and edges.

Statistical-based noise filtering techniques are computationally efficient and easy to implement, but they may not be effective in removing structured noise or outliers. They are suitable for applications that require simple and fast preprocessing, such as visualization and point cloud compression.

(b) Neighborhood-based noise filtering

Neighborhood-based noise filtering is a type of noise filtering technique used in point cloud processing that aims to remove noise by identifying and filtering out points that do not conform to the geometric properties of the neighboring points (X. F. Han et al., 2017).

In neighborhood-based noise filtering, a local neighborhood around each point in the point cloud is defined, typically using a fixed distance or a fixed number of neighboring points. The geometric properties of the points in the neighborhood, such as their spatial positions, normal vectors, or curvature, are then evaluated and compared to a predefined model or threshold.

1.4.4. Segmentation methods for point clouds

Segmentation of structural elements from a point cloud involves the process of identifying and separating distinct structural elements such as floors, beams, walls, columns, roofs from the overall 3D point cloud data. The segmentation process is generally a complex procedure that requires advanced algorithms and computational resources. However, the overall process usually follows several operations which can divided into four main steps; *preprocessing*, *extraction of geometric features*, *point classification* and finally, the *segmentation of structural elements*.

- Preprocessing of point cloud: this activity involves cleaning up the point cloud which normally contains noise, outliers and other artifacts that may affect the segmentation process. It also includes aligning the point cloud to a common coordinate system.
- Feature estimation and extracting 3D geometric attributes: Once the data has been cleaned up, the next process is to identify all the relevant features from the point cloud and extract the respective geometric properties for further processing.
- Point Classification: this operation deals with assigning labels or categories to individual points in a point cloud dataset based on their characteristics of the geometric features extracted in the previous step. The labels can represent different classes of objects, and for the built environment many researches have concentrated in classifying the generic building components such as floors, walls and its openings, columns, beams, and roofs.
- Segmentation of structural elements: Once the points have been labelled, the next step is to segment the structural elements from the rest of the data. Various techniques are used to such as clustering, semantic segmentation using machine learning methods such as unsupervised means and deep learning methods, convex hulls, and plane fitting methods.

The segmentation process can be divided in two main groups: the hardcoded knowledge-based methods and deep learning methods.

(a) Hardcoded knowledge-based methods

Hardcoded knowledge-based segmentation methods refers to a techniques used to segment the structural elements from the point cloud based on the predefined rules or patterns (Q. Wang & Kim, 2019). The use of prior knowledge about the building components integrated with the features extracted from the point cloud has been used to separate structural points from non-structural points from the point cloud.

In this approach, specific patterns and characteristics are used to identify structural elements within the point cloud data. For example, certain geometric shapes or patters may be indicative of certain elements such as floor or walls and these can be used to segment the points accordingly. The process typically involves identifying individual points or groups of points that represent particular features and patterns and using that to segment them into distinct elements.

Several studies have applied the hardcoded knowledge-based context to segment point clouds, and they can be divided into the following types: density-based clustering, and geometric modelling.

The density-based clustering is commonly used technique used to discover the unknown clusters without the need to know the number of clusters. One popular algorithm used is Density-Based Spatial Clustering of Applications with Noise (DBSCAN) (Ester et al., 1996), which is in point cloud processing and data mining. These methods rely on the concept of density of samples, which is the number of points within a certain Euclidean distance from a given point. (X. Chen et al., 2022) segmented indoor objects from a point cloud scene by developing a density clustering model which used an exponential function and local density distribution along the z-direction. The function allowed to obtain a cutoff distance between clusters in point cloud. Majority of studies that applied this technique have relied on the variation of point densities along the z-axis to cluster and identify horizontal components such as floors and ceilings (J. Chen et al., 2017; Cho et al., 2004; Hong et al., 2015; Valero et al., 2016). Cho et al. (2004) built a pseudo-grid that virtually contains points in accordance with their z-values and filter out noise based on the predefined local maxima and point distribution. Hong et al. (2015) also used the pseudo points to estimate the ceiling-to-floor height to perform vertical modelling. The authors used the local maxima and minima of z-values from the pseudo isolate the ceilings and floors and then, used RANSAC to modelling.

Geometric modeling for point cloud involves estimating the geometric shapes such as planes, spheres, cylinders, or linear features that can best describe a set of 3D points in a point cloud. Geometrical modelling is one among the fundaments challenges in modelling the built assets, which often uses plane-based approach since most building elements are composed of planar surfaces (Q. Wang & Kim, 2019). As mentioned previously some of the commonly applied algorithms for geometric modeling (fitting) RANSAC, 3D HT and region-growing. Convex hull has also been applied to detect the outer boundaries of geometric shapes by tracing and fit the closest points given a set of points. Convex hull can be applied to solve the problem of edge detection and generate meaningful boundaries of various shapes using the alpha shapes (Edelsbrunner et al., 1983; Fayed & Mouftah, 2009; Sampath & Shan, 2005). Similar approach was used by (Maalek et al., 2019) to detect the column boundaries on the segmented planes. However, the downside to the geometric fitting method is its sensitivity towards geometry shape which is difficult to generalize due to uniqueness of different building objects and their shape profiles. Hence this method is normally used in the availability of as-planned BIM or Computer Aided Design (CAD) model for comparison purposes.

The downside of hardcoded-based knowledge method is that it can be effective in some cases, however it may require significant manual effort to define the patterns and characteristics used for segmentation and it may not always be applicable to all types of point cloud scenes. Moreover, deep learning methods may be able to achieve more adaptable segmentation results by automatically learning patterns and features from the data itself.

(b) Point-based Deep Learning methods

In this part, we review the basics and techniques of deep learning (DL) networks in segmenting the point cloud data. DL has made significant advancements in both image processing and point cloud processing, but the techniques and challenges associated with each field are different. Processing 3D point clouds requires specialized algorithms that can handle the additional

dimensionality, unique data structure, and properties of point clouds, such as point scarcity, irregularity, and varying density. In contrast, image processing algorithms are designed to handle the regular grid-like structure of pixels in a 2D structure. Another difference is the type of information captured by each representation. Images capture color and intensity information in a 2D, while point clouds capture geometric information about the scene or object, such as shape and depth. For that reason, we have seen more applications and advances in deep learning algorithms in images.

Deep learning techniques have been adapted and developed specifically for point clouds, such as *point-based* and *convolution neural networks* (CNN), which have achieved groundbreaking results in segmenting building components. DL methods achieves the segmentation of point cloud associated with tasks such as the classification, semantic segmentation, and instance segmentation. There are numerous deep learning techniques used to segment point clouds, for this research we will introduce three prevalent networks: PointNet, PointNet++, and Dynamic Graph CNN.

(i) PointNet

PointNet as proposed by (Qi, Su, et al., 2017) is a deep learning architecture that directly processes the 3D point cloud data with while being invariant to permutations of the input points. Its architecture consist several layers of neural networks that takes advantage of the transformation networks (T-Net) and a number of the multi-layer perceptron (MLP) to extract features from the unordered set of n points, $P = \{p_1, p_2, \ldots, p_n\}$. The PointNet architecture allows to perform both classifications based on the pre-trained data and semantic segmentation by combining the local and global features.

PointNet has several advantages over other DL architectures due to its robustness to rotation and translation and ability to handle variable-sized point clouds with low memory requirements.

(ii) PointNet++

PointNet++ is a modified version of the PointNet (Qi, Su, et al., 2017), where a hierarchical neural network is introduced to capture local and global features of the input point cloud with high complexity (Qi, Yi, et al., 2017). The architecture is improved to divide the point cloud into a hierarchy of nested clusters. For each hierarchy, the network processes the points within that the cluster to increment more information about the global features for the entire point cloud. This allows efficient propagation of feature learning to speed up the aggregation of local and global features.

(iii) Dynamic Graph CNN (DGCNN)

DGCNN which is proposed by (Y. Wang et al., 2019) is a CNN-based architecture that uses the dynamic graph structure that allows to capture local and global features of the input point cloud. The network treats each point of the point cloud as a node of a graph, with vertices (edges) formed by the relations between neighboring points.

DGCNN is built to perform in two-stage process where in the first stage, the network applies a shared MLP to each point to extract the local features. Then, using max pooling the local features

are aggregated to produce a global feature representation. In the next stage, the network implements the dynamic graph convolution layer on a local neighborhood for each point to gain the geometric relationships between nearby points in the point cloud. This process also, used the max pooling to aggregate the resulting features as in the first stage.

1.5. Structure and organization

This thesis is structured as follows. Chapter 2 addresses the theoretical aspects of spatial subdivision of point cloud using voxel structure, and feature estimation of point clouds. Chapters 3 and chapter 4 are the core chapters for this thesis and describes the employed methods and how the methodological steps are interrelated as a means for attaining the goals presented above. Chapter 5 presents the experiments, experiment results and introduction to the methods used to evaluate the experimental results. Chapter 6 presents the evaluation of the experiment, and discussions of the overall methods. Chapter 7 closes the thesis by drawing the main conclusions and making suggestions for future works.

References

Araújo, A. M. C., & Oliveira, M. M. (2020). A robust statistics approach for plane detection in unorganized point clouds. *Pattern Recognition*, *100*. https://doi.org/10.1016/j.patcog.2019.107115

Bentley, J. L. (1975). Multidimensional Binary Search Trees Used for Associative Searching. *Communications of the ACM*, *18*(9), 509–517. https://doi.org/10.1145/361002.361007

Borrmann, D., Elseberg, J., Lingemann, K., & Nüchter, A. (2011). The 3D Hough Transform for plane detection in point clouds: A review and a new accumulator design. *3D Research*, *2*(2), 1–13. https://doi.org/10.1007/3DRes.02(2011)3

Chen, J., Fang, Y., & Cho, Y. (2017). Unsupervised recognition of volumetric structural components from building point clouds. *Congress on Computing in Civil Engineering, Proceedings*, *December*, 34–42. https://doi.org/10.1061/9780784480823.005

Chen, X., Wu, H., Lichti, D., Han, X., Ban, Y., Li, P., & Deng, H. (2022). Extraction of indoor objects based on the exponential function density clustering model. *Information Sciences*, *607*, 1111–1135. https://doi.org/10.1016/j.ins.2022.06.032

Cho, W., Jwa, Y., Chang, H., & Lee, S. (2004). Pseudo-Grid Based Building Extraction Using Airborne LIDAR Data. *In: International Archives of the Photogrammetry and Remote Sensing*, 378–383. https://www.isprs.org/proceedings/xxxv/congress/comm3/papers/298.pdf

Deschaud, J. (2010). A Fast and Accurate Plane Detection Algorithm for Large Noisy Point Clouds Using Filtered Normals and Voxel Growing. *Symposium A Quarterly Journal In Modern Foreign Literatures*. http://campwww.informatik.tu-muenchen.de/3DPVT2010/data/media/e-proceeding/papers/paper111.pdf

Díaz-Vilariño, L., Conde, B., Lagüela, S., & Lorenzo, H. (2015). Automatic detection and segmentation of columns in as-built buildings from point clouds. *Remote Sensing*, *7*(11), 15651–15667. https://doi.org/10.3390/rs71115651

Dimitrov, A., & Golparvar-Fard, M. (2015). Segmentation of building point cloud models

including detailed architectural/structural features and MEP systems. *Automation in Construction*, *51*(C), 32–45. https://doi.org/10.1016/j.autcon.2014.12.015

Edelsbrunner, H., Kirkpatrick, D., & Seidel, R. (1983). On the shape of a set of points in the plane. *IEEE Transactions on Information Theory*, *29*(4), 551–559. https://doi.org/10.1109/TIT.1983.1056714

Ester, M., Kriegel, H.-P., Sander, J., & Xu, X. (1996). A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, 226–231.

Fayed, M., & Mouftah, H. T. (2009). Localised alpha-shape computations for boundary recognition in sensor networks. *Ad Hoc Networks*, *7*(6), 1259–1269. https://doi.org/10.1016/j.adhoc.2008.12.001

Fischler, M. A., & Bolles, R. C. (1981). RANSAC: Random Sample Paradigm for Model Consensus: A Apphcatlons to Image Fitting with Analysis and Automated Cartography. *Graphics and Image Processing*, *24*(6), 381–395.

Hajian, H., & Becerik-Gerber, B. (2010). Scan to BIM: Factors affecting operational and computational errors and productivity loss. *2010 - 27th International Symposium on Automation and Robotics in Construction, ISARC 2010*, *Isarc*, 265–272. https://doi.org/10.22260/isarc2010/0028

Hamid-Lakzaeian, F. (2020). Point cloud segmentation and classification of structural elements in multi-planar masonry building facades. *Automation in Construction*, *118*(October 2018), 103232. https://doi.org/10.1016/j.autcon.2020.103232

Han, L., Yongmei, L., Chaoguang, M., & Yong, F. (2021). A novel 3D point cloud segmentation algorithm based on multi-resolution supervoxel and MGS. *International Journal of Remote Sensing*, *42*(22), 8492–8525. https://doi.org/10.1080/01431161.2021.1978583

Han, X. F., Jin, J. S., Wang, M. J., Jiang, W., Gao, L., & Xiao, L. (2017). A review of algorithms for filtering the 3D point cloud. *Signal Processing: Image Communication*, *57*(November 2016), 103–112. https://doi.org/10.1016/j.image.2017.05.009

Hong, S., Jung, J., Kim, S., Cho, H., Lee, J., & Heo, J. (2015). Semi-automated approach to indoor mapping for 3D as-built building information modeling. *Computers, Environment and Urban Systems*, *51*, 34–46. https://doi.org/10.1016/j.compenvurbsys.2015.01.005

Hyunsoo, K., & Changwan, K. (2021). 3D as-built modeling from incomplete point clouds using connectivity relations. *Automation in Construction*, *130*(January). https://doi.org/10.1016/j.autcon.2021.103855

Kang, C. L., Wang, F., Zong, M. M., Cheng, Y., & Lu, T. N. (2020). RESEARCH on IMPROVED REGION GROWING POINT CLOUD ALGORITHM. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, *42*(3/W10), 153–157. https://doi.org/10.5194/isprs-archives-XLII-3-W10-153-2020

Kultanen, P., Xu, L., & Oja, E. (1990). Randomized Hough transform (RHT). *Proceedings - International Conference on Pattern Recognition*, *1*(July), 631–635. https://doi.org/10.1109/icpr.1990.118177

Li, L., Yang, F., Zhu, H., Li, D., Li, Y., & Tang, L. (2017). An improved RANSAC for 3D point cloud plane segmentation based on normal distribution transformation cells. *Remote Sensing*, *9*(5). https://doi.org/10.3390/rs9050433

Li, Y., Bu, R., Sun, M., Wu, W., Di, X., & Chen, B. (2018). PointCNN : Convolution On X - Transformed Points. *32nd Conference on Neural Information Processing Systems (NeurIPS)*, 820–830.

Limberger, F. A., & Oliveira, M. M. (2015). Real-time detection of planar regions in unorganized point clouds. *Pattern Recognition*, *48*(6), 2043–2053. https://doi.org/10.1016/j.patcog.2014.12.020

Maalek, R., Lichti, D. D., & Ruwanpura, J. (2015). Robust classification and segmentation of planar and linear features for construction site progress monitoring and structural dimension compliance control. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *2*(3W5), 129–136. https://doi.org/10.5194/isprsannals-II-3-W5-129-2015

Maalek, R., Lichti, D., & Ruwanpura, J. (2019). Automatic Recognition of Common Structural Elements from Point Clouds for Automated Progress Monitoring and Dimensional Quality Control in Reinforced Concrete Construction. *Journal of Remote Sensing*, *11*(9), 1102. https://doi.org/10.3390/rs11091102

Nurunnabi, A., Belton, D., & West, G. (2012). Robust segmentation for multiple planar surface extraction in laser scanning 3D point cloud data. *Proceedings - International Conference on Pattern Recognition*, *Icpr*, 1367–1370.

Poux, F., & Billen, R. (2019). Voxel-based 3D Point Cloud Semantic Segmentation: Unsupervised geometric and relationship featuring vs deep learning methods. *ISPRS International Journal of Geo-Information*, *8*(5). https://doi.org/10.3390/ijgi8050213

Qi, C. R., Su, H., Mo, K., & Guibas, L. J. (2017). PointNet: Deep learning on point sets for 3D classification and segmentation. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, *2017-Janua*, 77–85. https://doi.org/10.1109/CVPR.2017.16

Qi, C. R., Yi, L., Su, H., & Guibas, L. J. (2017). PointNet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems*, *2017-Decem*, 5100–5109.

Rabbani, T., van den Heuvel, F. a, & Vosselman, G. (2006). (impo)(Fashuai exper+Sudan recom)Segmentation of point clouds using smoothness constraint. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences - Commission V Symposium "Image Engineering and Vision Metrology," 36*(5), 248–253. http://www.isprs.org/proceedings/XXXVI/part5/paper/RABB_639.pdf

Rusu, R. ., & Cousins, S. (2011). 3D is here: Point Cloud Library (PCL). *N Proceedings of the 2011 IEEE International Conference on Robotics and Automation*.

Sampath, A., & Shan, J. (2005). *Building Boundary Tracing and Regularization*. *2051*(July), 805–812.

Schnabel, R., Wahl, R., & Klein, R. (2007). Efficient RANSAC for point-cloud shape detection. *Computer Graphics Forum*, *26*(2), 214–226. https://doi.org/10.1111/j.1467-8659.2007.01016.x

Son, H., & Kim, C. (2017). Semantic as-built 3D modeling of structural elements of buildings based on local concavity and convexity. *Advanced Engineering Informatics*, *34*(October), 114–124. https://doi.org/10.1016/j.aei.2017.10.001

Valero, E., Adán, A., & Bosché, F. (2016). Semantic 3D Reconstruction of Furnished Interiors Using Laser Scanning and RFID Technology. *Journal of Computing in Civil Engineering*, *30*(4). https://doi.org/10.1061/(asce)cp.1943-5487.0000525

Vincke, S., Hernandez, R. D. L., Bassier, M., & Vergauwen, M. (2019). Immersive visualisation of construction site point cloud data, meshes and bim models in a vr environment using a gaming engine. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, *42*(5/W2), 77–83. https://doi.org/10.5194/isprs-archives-XLII-5-W2-77-2019

Vo, A. V., Truong-Hong, L., Laefer, D. F., & Bertolotto, M. (2015). Octree-based region growing for point cloud segmentation. *ISPRS Journal of Photogrammetry and Remote Sensing*, *104*, 88–100. https://doi.org/10.1016/j.isprsjprs.2015.01.011

Wang, Q., & Kim, M. (2019). Applications of 3D point cloud data in the construction industry: A fifteen-year review from 2004 to 2018. *Advanced Engineering Informatics*, *39*(September 2018), 306–319. https://doi.org/10.1016/j.aei.2019.02.007

Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., & Solomon, J. M. (2019). Dynamic graph Cnn for learning on point clouds. *ACM Transactions on Graphics*, *38*(5). https://doi.org/10.1145/3326362

Xiong, X., Adan, A., Akinci, B., & Huber, D. (2013). Automatic creation of semantically rich 3D building models from laser scanner data. *Automation in Construction*, *31*, 325–337. https://doi.org/10.1016/j.autcon.2012.10.006

Xu, Y. (2019). Reconstruction of building objects from point clouds of built environment and construction sites [Technical University of Munich]. In *phD thesis*. https://www.semanticscholar.org/paper/Reconstruction-of-building-objects-from-point-of-Xu/f6a6f74a97b37e3b59e75032343c55cad51cdab7#citing-papers

Xu, Y., Tong, X., & Stilla, U. (2021). Voxel-based representation of 3D point clouds: Methods, applications, and its potential use in the construction industry. *Automation in Construction*, *126*(February), 103675. https://doi.org/10.1016/j.autcon.2021.103675

Zhang, J., Zhao, X., Chen, Z., & Lu, Z. (2019). A Review of Deep Learning-Based Semantic Segmentation for Point Cloud. *IEEE Access*, *7*, 179118–179133. https://doi.org/10.1109/ACCESS.2019.2958671

Chapter 2. Theoretical basics

2.1. Introduction

2.2. Voxel-based Partitioning of 3D Point Cloud

2.2.1. Grid-based Voxelization

2.2.2. Octree-based Voxelization

2.3. Feature estimation for 3D Geometric Shapes in Point Clouds

2.3.1. Eigen-based features

2.3.2. Height-based features

2.3.3. Projecton-area-based features

2.3.4. Surface-based features

2.1. Introduction

In this chapter, specific methods and techniques about the basic voxel-based data structure, geometric feature extraction, and geometric modeling of point clouds. The methods and techniques will be extensively used in the following methodology work. An appropriate notation is also introduced which will be used throughout the remaining chapters of this thesis

2.2. Voxel-based Partitioning of 3D Point Cloud

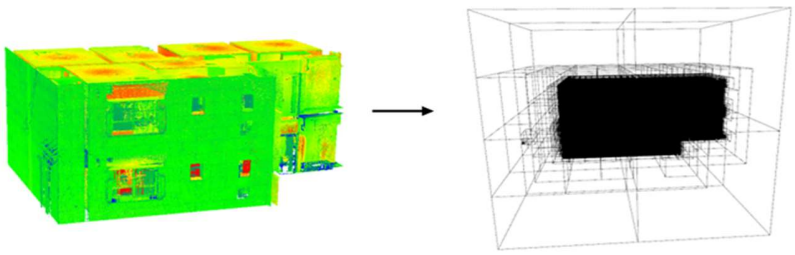
Voxelization of the point cloud involves discretizing the input point cloud into smaller and more manageable regular grids of cubic voxels. Each voxel represents a small portion of the point cloud containing a particular number of points falling within its grid boundaries. This approach is commonly used in the processing and analysis of 3D scenes in computer graphics, computer vision, and robotic applications. The points are bucketed in the voxels, and they can each be indexed to represent a particular voxel.

There are two main methods of voxelization: Grid-based voxelization and Octree-based Voxelization.

2.2.1. Grid-based Voxelization

A grid-based voxelization is a technique of subdividing a point cloud into regular grids of 3D space creating cubicle cells of equal sizes. Then, for each cell of a grid, a voxel unit is generated to encompass all points found within its space. This approach is simple and computationally efficient that it does not require constructing complex data structures such as *octree* or *k-d tree* to parse the entire point cloud. Another advantage, of grid voxelization, is that it is easy to regulate its resolution by adjusting the grids.

However, the disadvantage of this approach is the creation of multiple empty cells or voxels with relatively fewer points, particularly in low dense areas. This is attributed to the discretization of the entire point cloud contrary to the octree-based voxelization. An illustration of grid-based voxelization is presented in Fig. 2.1 The voxels can be referred from the grid indexes by the center or boundary coordinates.



(a) (b)

Figure 2. 1 Voxelization of the point cloud using grid-based approach: (a) Original point cloud; (b) voxelized point cloud.

2.2.2. Octree-based Voxelization

This method involves partitioning the point cloud into voxels using an octree structure. An octree is a tree-based structure in which each node (parent-voxel) is subdivided into eight equally sized children (sub-voxels). The octree is built recursively by subdividing each voxel into eight smaller cubes (voxels) until a certain termination criterion is met. The criteria used are often, the minimum number of points in a tree or the maximum depth of the tree. If a cube has yet to meet the criterion, it is further subdivided into eight even smaller cubes until, and so on until the maximum depth is reached or minimum number of points are left. Once the octree is formed, the voxels are constructed to occupy the points based on the center of each octree node constructed. Fig. 2.2 illustrates the operational diagram for this approach.

This approach differs from grid-based voxelization since it does not need to operate on empty spaces which save the computation cost. However, when working with large point clouds it may prove significantly expensive in memory consumption and computation when needing to traverse through each node of the octree structure.

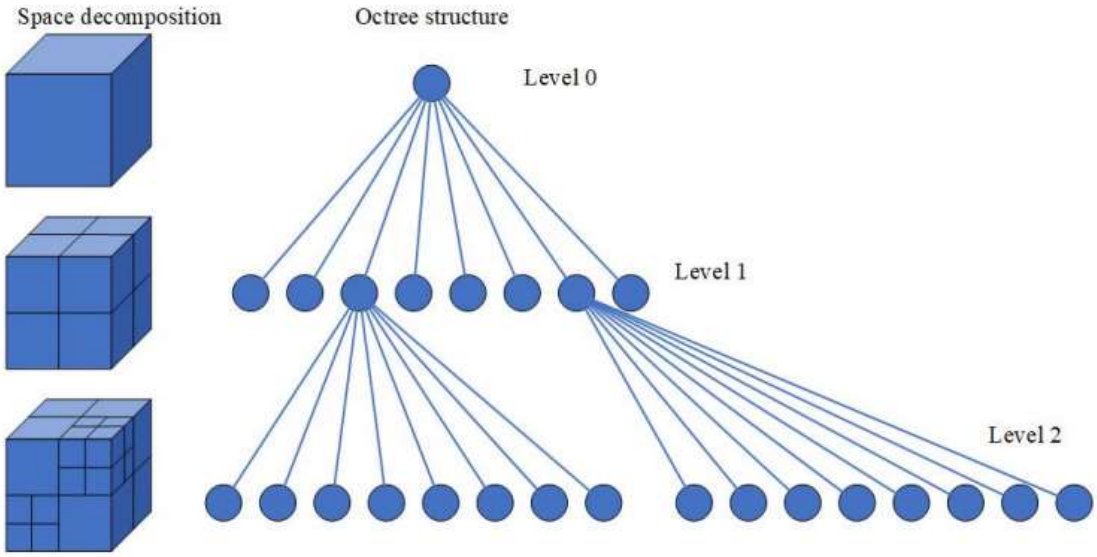


Figure 2. 2 Illustration of the octree structure (Guan et al., 2012).

2.3. Feature estimation for 3D Geometric Shapes in Point Clouds

Estimation of features from a 3D point cloud involves identifying the underlying attributes, structure, and patterns of points that can be used to describe geometric shapes and characteristics of 3D scenes.

There are a number of features that can be analyzed from a point cloud, and while they are significant in describing the point cloud, estimating all of them without relevancy can be rendered redundant and computationally expensive (Chehata et al., 2009; Ni et al., 2017). It is necessary to specify the features of interest depending on the application and nature of the point cloud. Some studies have developed procedures for selecting the features from the point cloud (Chehata et al., 2009; Guan et al., 2012; Weinmann et al., 2015).

In this study, we have focused on the geometry features to discover the geometry structures in the point clouds. According to (Ni et al., 2017) the geometric features in point clouds can be summarized into fourfold: *Eigen-based, Height-based*, *Projection-area-based*, and *Surface-based* features. These features provide powerful and efficient ways to capture and analyze the local structure and variability of the point clouds.

2.3.1. Eigen-based features

Eigen features are essentially descriptors that capture the geometric properties of a point cloud for shape analysis, feature extraction and object recognition. The common geometric descriptors derived using the Eigen features including curvature, normal orientation and local shape.

Eigen-based features are derived from the eigenvalues and eigenvectors of the covariance matrix ($\boldsymbol{C}$) of the suitable neighborhood points in the point cloud (Chehata et al., 2009). These features are operating under the concept that a local 3D structure and orientation of a respective point-neighborhood N is aligned and spread in principle axes. The local structure around a given point p from a point cloud is described from the selection of a suitable closest neighborhood $N(p)$ from the point p. Several approaches are used to define the neighborhood structure $N(p)$ including spherical (Guo et al., 2015; I. Lee & Schenk, 2002), cylindrical (Filin & Pfeifer, 2005; Guo et al., 2015) shapes, and using a fixed number of $K \in \mathbb{N}$ closest neighbors (lars Linsen & Prautzsch, 2001). The use of K neighbors optimizes the neighborhood size and hence, increases the distinctiveness of local features irrespective of the local shape.

Given a local neighborhood structure within k points $\{p_1, p_2, \ldots, p_k\}$, the covariance matrix $\boldsymbol{C}$ can be derived according with Equation (2.1).

$$\mathrm{C} = \frac{1}{k-1}\sum_{i=1}^{k}(p_{\mathrm{i}} - p)(p_{\mathrm{i}} - p)^{\mathrm{T}} \tag{2.1}$$

Where p is the mean vector representing the centroid location of the neighborhood structure in accordance with Equation (2.2).

$$p = \sum_{i=1}^{k} p_{\mathrm{i}} \tag{2.2}$$

The eigenvalues and eigenvectors of the C matrix provide essential information about the local geometry within the $N(p)$. The eigenvalues represent the variance of the point cloud along each principal axis, while the eigenvectors represent the orientation of these axes (Feng & Guo, 2018; Weinmann et al., 2015). Given a neighborhood structure and the derived eigenvectors($\overrightarrow{v_{\mathrm{i}}}$) where $\overrightarrow{v_{\mathrm{i}}} \in \{\overrightarrow{v_0}, \overrightarrow{v_1}, \overrightarrow{v_2}\}$, with the corresponding eigenvalues (λ_i) where $\lambda_i \in \{\lambda_0, \lambda_1, \lambda_2\}$ several eigen-based features can be estimated.

With ordering the eigenvalues in a decreasing values such that $\lambda_0 > \lambda_1 > \lambda_2 > 0$, the following 3D shape features can be estimated to describe the local neighborhood structure; its surface normal vector, $\overrightarrow{v_2}$, its anisotropy, λ_{an}, its omnivariance, λ_{om}, its eigen entropy, λ_{en} (Poux & Billen, 2019),

its linearity, λ_{ln} , its planarity, λ_{pl} , and its surface variation, λ_{sv}, (Chehata et al., 2009; Poux & Billen, 2019; Weinmann et al., 2015). These features can be computed as according.

$$\lambda_{an} = \frac{(\lambda_0 - \lambda_2)}{\lambda_0} \tag{2.3}$$

$$\lambda_{ln} = \frac{(\lambda_0 - \lambda_1)}{\lambda_0} \tag{2.4}$$

$$\lambda_{pl} = \frac{(\lambda_1 - \lambda_2)}{\lambda_1} \tag{2.5}$$

$$\lambda_{sv} = \frac{\lambda_2}{\sum_{i=0}^{2} \lambda_0} \tag{2.6}$$

$$\lambda_{om} = \sqrt[3]{\prod_{i=0}^{2} \lambda_0} \tag{2.7}$$

$$\lambda_{en} = -\sum_{i=0}^{2} \lambda_i * \ln(\lambda_i) \tag{2.8}$$

2.3.2. Height-based features

Height-based features in point cloud are deduced from the vertical elevation information of the points in a point cloud. These features are particularly useful for applications that require information about the height variations of a surface or topography for instance for buildings is used for surface quality inspection for concrete slab or floor flatness (Bosché & Biotteau, 2015; Bosché & Guenet, 2014; Puri et al., 2018; Q. Wang et al., 2016) and surface deformation analysis (Nuttens et al., 2014; Sánchez-Aparicio et al., 2018; Q. Wang et al., 2016), urban planning, tree detection and terrain modelling (Chehata et al., 2009; Feng & Guo, 2018; Guan et al., 2012).

Some of the common height-based features are briefly described below:

- Elevation or Vertical-profile-based features: This feature helps to measure the vertical distance between the ground and off-ground objects by obtaining the difference between a particular point of a terrestrial lidar data and the lowest point in a generated cylindrical with predetermined radius (Chehata et al., 2009; Guan et al., 2012; Guo et al., 2015). This measurement can be used to estimate the height of buildings, trees, or power lines.
- Surface roughness: These features are used to describe variability across a surface. It is normally used to identify areas of rough terrains and surface deformity such as rocky outcrops or uneven surfaces (Fan & Atkinson, 2018; Rodriguez-Cuenca et al., 2015).
- Elevation variance: This is the variance of the measurements of height above the ground. It provides are higher value to standardize minimal differences for comparison purposes (Chehata et al., 2009).

- Local curvature: This is the measure of the rate of change of a surface at a particular point, and it is often used in the analyze the peak terrains and identify vertical features (Steinle & Voegtle, 2001).

2.3.3. Projection-area-based features

Projection area based features are a type of features derived from the projection of the point cloud or portion of points onto different planes or surfaces for analysis. (Johnson & Hebert, 1999) used the projection area features to perform simultaneous recognition of multiple 3D shaped objects by creating a spin image. (Guo et al., 2015) also used the same approach where the 2D accumulator was used as an image screen to collect bins containing projected points. The projection area is then calculated using the non-bare bins containing the projection points. The projection area-based features can be classified in twofold (Guo et al., 2015):

- *Horizontal-projection-area features*: these features are derived from the 2D accumulator for area that is parallel to the XY projection plane. Object such as roof, ground and power lines can be identified using this approach.
- *Minimum vertical-projection-area features*: for these features, the direction perpendicular to the Z-axis where the bins distributed along the XY plane are used to calculate the areas. Vegetation and buildings are commonly identified using this approach.

2.3.4. Surface-based features

These are features related to flatness of the surface which is often associated to planarity, surface curvature and normal (Guo et al., 2015) developed two measurement indices to identify surface-based features:

- Planarity index: this is a measurement that indicates whether a point belongs to a particular plane, such as the floor or a wall. It is calculated by fitting a plane to a particular range of points and measure the amount of deviation. By calculating the plane index for each point in the point cloud, it is possible to segment the point cloud into different planes.
- Surface index: Similarly, the surface index is the value that indicates if a point belongs to a particular type of surface or not. It is calculated using the surface curvatures and normal. The surface index is normally regarded as the average value of all points to a plane's surface.

References

Bosché, F., & Biotteau, B. (2015). Terrestrial laser scanning and continuous wavelet transform for controlling surface flatness in construction - A first investigation. *Advanced Engineering Informatics*, *29*(3), 591–601. https://doi.org/10.1016/j.aei.2015.05.002

Bosché, F., & Guenet, E. (2014). Automating surface flatness control using terrestrial laser scanning and building information models. *Automation in Construction*, *44*, 212–226. https://doi.org/10.1016/j.autcon.2014.03.028

Chehata, N., Guo, L., & Mallet, C. (2009). Airborne LiDAR feature selection for urban classification using random forests. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *38 (Part 3*(W8). https://hal.science/hal-02384719/

Fan, L., & Atkinson, P. M. (2018). A new multi-resolution based method for estimating local surface roughness from point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing*, *144*, 369–378. https://doi.org/10.1016/j.isprsjprs.2018.08.003

Feng, C. C., & Guo, Z. (2018). Automating parameter learning for classifying terrestrial LiDAR point cloud using 2D land cover maps. *Remote Sensing*, *10*(8). https://doi.org/10.3390/rs10081192

Filin, S., & Pfeifer, N. (2005). Neighborhood systems for airborne laser data. *Photogrammetric Engineering and Remote Sensing*, *71*(6), 743–755. https://doi.org/10.14358/PERS.71.6.743

Guan, H., Yu, J., Li, J., & Luo, L. (2012). Random Forests-Based Feature Selection for Land-Use Classification Using Lidar Data and Orthoimagery. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *XXXIX-B7*(September), 203–208. https://doi.org/10.5194/isprsarchives-xxxix-b7-203-2012

Guo, B., Huang, X., Zhang, F., & Sohn, G. (2015). Classification of airborne laser scanning data using JointBoost. *ISPRS Journal of Photogrammetry and Remote Sensing*, *100*, 71–83. https://doi.org/10.1016/j.isprsjprs.2014.04.015

Johnson, A. E., & Hebert, M. (1999). Using spin images for efficient object recognition in cluttered 3D scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *21*(5), 433–449. https://doi.org/10.1109/34.765655

Lee, I., & Schenk, T. (2002). Perceptual organization of 3D surface points. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *34*(3A), 193–198.

Linsen, lars, & Prautzsch, H. (2001). Local Versus Global Triangulations. *Proceedings of Eurographics*, 257–263. https://doi.org/10.2312

Ni, H., Lin, X., & Zhang, J. (2017). Classification of ALS point cloud with improved point cloud segmentation and random forests. *Remote Sensing*, *9*(3). https://doi.org/10.3390/rs9030288

Nuttens, T., Stal, C., De Backer, H., Schotte, K., Van Bogaert, P., & De Wulf, A. (2014). Methodology for the ovalization monitoring of newly built circular train tunnels based on laser scanning: Liefkenshoek Rail Link (Belgium). *Automation in Construction*, *43*, 1–9.

https://doi.org/10.1016/j.autcon.2014.02.017

Poux, F., & Billen, R. (2019). Voxel-based 3D Point Cloud Semantic Segmentation: Unsupervised geometric and relationship featuring vs deep learning methods. *ISPRS International Journal of Geo-Information*, *8*(5). https://doi.org/10.3390/ijgi8050213

Puri, N., Valero, E., Turkan, Y., & Bosché, F. (2018). Assessment of compliance of dimensional tolerances in concrete slabs using TLS data and the 2D continuous wavelet transform. *Automation in Construction*, *94*(May), 62–72. https://doi.org/10.1016/j.autcon.2018.06.004

Rodriguez-Cuenca, B., Garcia-Cortes, S., Ordonez, C., & Alonso, M. C. (2015). A study of the roughness and curvature in 3D point clouds to extract vertical and horizontal surfaces. *International Geoscience and Remote Sensing Symposium (IGARSS)*, *2015-Novem*(July), 4602–4605. https://doi.org/10.1109/IGARSS.2015.7326853

Sánchez-Aparicio, L. J., Del Pozo, S., Ramos, L. F., Arce, A., & Fernandes, F. M. (2018). Heritage site preservation with combined radiometric and geometric analysis of TLS data. *Automation in Construction*, *85*(March 2017), 24–39. https://doi.org/10.1016/j.autcon.2017.09.023

Steinle, E., & Voegtle, T. (2001). Automated extraction and reconstruction of buildings in laserscanning data for disaster management. *In: Proceeding of the Workshop on Automatic Extraction of Man-Made Objects from Aerial and Space Images*, 309–318.

Wang, Q., Kim, M. K., Sohn, H., & Cheng, J. C. P. (2016). Surface flatness and distortion inspection of precast concrete elements using laser scanning technology. *Smart Structures and Systems*, *18*(3), 601–623. https://doi.org/10.12989/sss.2016.18.3.601

Weinmann, M., Jutzi, B., Hinz, S., & Mallet, C. (2015). Semantic point cloud interpretation based on optimal neighborhoods, relevant features and efficient classifiers. *ISPRS Journal of Photogrammetry and Remote Sensing*, *105*, 286–304. https://doi.org/10.1016/j.isprsjprs.2015.01.016

Chapter 3. Detection and Merging of planar patches

3.1. Introduction

This chapter introduces the initial methods of the proposed segmentation pipeline. The chapter covers the spatial decomposition of point cloud and placing the points in the voxels, estimating of eigen-based features of points populated in the voxels, detection of minimal planar patches and merging of minimal planar patches.

3.2. Octree Decomposition of Point Cloud

Octree-based voxelization is a popular technique of partitioning the point cloud using the octree structure and bucketing the points in voxel units. The first step is to construct the octree structure, this is typically done by recursively subdividing the point cloud into octants until the termination criteria is met. In this case, we adopted the minimum number of points as our termination criterion as used. Each leaf node of the octree represents a smaller region of input point cloud and contain list of all points that fall within a respective space region in accordance with the spatial frame system in use. Once the octree is generated, each leaf node is converted into a voxel in the voxel-grid. The capitalizes the benefits of both systems of voxelization, where the voxel-grids provides uniform space partitioning and octree structure reduces the memory consumption by disregarding empty voxels and very sparse scenes which carries no geometry value.

This approach also benefits in redistributing the dense regions of the point cloud which can improve accuracy and efficiency of the subsequent processing steps. Also, it supports efficient query by using voxel-grid to locate specific regions of interest, which it can be useful for applications such as collision detection in computer graphics (Vo et al., 2015; Xu, 2019).

3.2.1. Generating an Octree Structure

Working with octree requires a proper choice of resolution level which ensures proper subdivision 3D space with regard to trade-off between the accuracy of results and requirements for both computational time and memory (Pulli et al., 1997). In this work, the input point cloud is equally partitioned into eight smaller regions (voxels) until a predefined maximum depth tree, Ψ_{max}, is reached. The use of maximum tree depth (Ψ_{max}) as our resolution strategy is a prevalent technique working with octree, (Poux & Billen, 2019; Pulli et al., 1997). This strategy allows for more efficient and effective data processing, specifically a deeper octree will result more levels and detailed partitioning of the point cloud data with increased homogeneity of data and finer details for precise analysis.

However, this approach requires a user-defined depth which is important since setting a depth tree which is too high can lead to over-segmentation and increase the computational complexity of processing the data. In consideration to this challenge, we set other criteria to complement the tree depth criteria, where the minimum number of points per each voxel, ε_{min} is used to have an optimum level of point cloud subdivision. The octree will recursively subdivide whichever the value of Ψ_{max} and ε_{min} is reached earlier.

The advantage of limiting the number of points in each voxel, is that the computational complexity of operations such as geometry fitting, segmentation and surface reconstruction can be significantly reduced, as fewer point need to be processed at each step. Another advantage of using minimum

number of points per voxel ensures that the resulting voxels are bucketed with points, which can help to preserve details and geometry features of the underlying surface or object of interest.

However, setting a lower number of points per voxel can also lead to creation of exorbitant number of smaller voxels which can result computational challenges and increased memory usage especially for larger point clouds. In addition, having a very small sample size would lead to an erroneous data estimation and over-fitting, i.e., model skewing towards one datum (Osborne & Costello, 2004; Shaukat et al., 2016), while a too large sample size would otherwise lead to diminishing returns (Forcino, 2012). As, the result the optimal balance of voxel size and point density is carefully chosen for general adaptation. In this case, a proposed minimum number of points per voxel is ε_{min}=30, which is proven to be efficient in large point clouds (Limberger & Oliveira, 2015).

The octree is constructed in which we start with creating the root octant node by inserting a cube encompassing the entire point cloud. Afterwards, the root node is subdivided sequentially into eight equal children nodes (cubes) and assign the points to each node. The x, y, z coordinate of each point in the point cloud is used to determine which octant is contained in space. The division continues for all non-empty nodes until the Ψ_{max} or ε_{min} is reached first. For storage referencing, the points are indexed similar to (Poux & Billen, 2019), which allow to efficiently access the points within a particular region of space as illustrated in Figure 3.1.

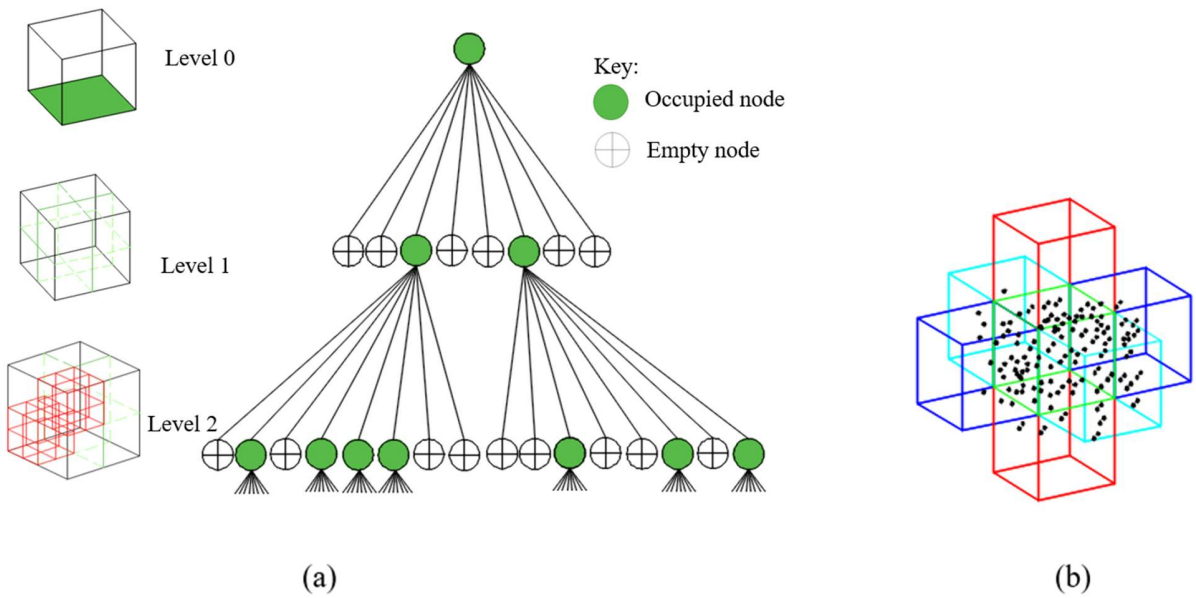


Figure 3. 1 Point cloud subdivision: (a) octree structural framework; (b) abutting voxels for joining proximate planar patches.

3.2.2. Voxelization of Point Cloud

As described in the previous sub-section, all points are fitted into respective node/voxel-space based on the -x, -y, -z coordinates. This subsection discusses about the build-up of voxels using the voxel-grids. The use of voxel-grids provides an effective referencing of voxels through grid indexes (Qian-Yi et al., 2018).

The voxel-grid is constructed using a similar approach to (Nourian et al., 2016), where the bounding box is initially fitted to contain the entire point cloud. The size of the bounding box is decided by the local minima and maxima of all points present in the point cloud. For each subsequent round of partitioning of octree, the bounding box is equally subdivided creating the voxel-grids. The voxel-grid are represented by 3D arrays of -x, -y, -z values of points falling within the volume space of voxel-grids. After the voxel-grid are constructed, all points are assigned to voxels which is done by iterating through all points and determining which voxel it belongs to by dividing its coordinates by the size of the voxel and rounding down to the nearest integer.

3.3. Feature estimation and Extraction of Planar Patches

This section deals with estimation of 3D geometry attributes, where in our case we compute the eigen-based features within voxels to detect the underlying surface planarity. As mentioned earlier, the octree-based voxels help to reduce the complexity of the point cloud data while retaining the most important information. The eigen-based features are computed to determine the underlying surface planarity using the Principal Component Analysis (PCA). PCA is a prominent technique used for dimensionality reduction and feature extraction in 3D computer vision. Features in the n-dimensional features ($\Re^n$) are transformed to a lower (n-i) dimensional space (i.e. $i = \{1,2,\ldots, n-1\}$), for easy analysis and interpretation. In this study we attempt to reduce the underlying planarity features which are in 3D space to a 2D space: $\Re^3 \rightarrow \Re^2$, to reveal the surface planarity within voxels.

3.3.1. Estimation of Eigen-based Features and PCA

The eigen-based features are computed to determine the underlying surface planarity using the Principal Component Analysis (PCA). PCA is a prominent technique used for dimensionality reduction and feature extraction in 3D computer vision.

Given voxel V_i, populated with n points is used to compute the covariance matrix, C, for a particular neighborhood of points. However, prior to calculation of the covariance matrix, all points in the voxels are initially standardized which is important to ensure that x, y, z values have the same scale. This is important especially for points that are located further-off the origin in one direction of either x or y or z. The process of standardization involves subtracting the mean (μ) and dividing by the standard deviation (σ) of each variable.

Once the points are standardized, the next step is to compute the covariance matrix as (eqn.). The covariance matrix represents the square matrix of that summarizes the relationships between pairs of variables. From the covariance matrix, the eigen decomposition is performed to obtain the eigenvector and eigenvalues. The eigenvectors give the directions of maximum variance of points in the voxels, and the eigenvalues represent the amount of variance relative to each of the eigenvector.

After the eigen decomposition, the next step is to choose the number of principal components to keep. All the points in the voxels are split into three main components representing their principal directions. The first principal component, *PC1* represents the direction of maximum variance, and the second, *PC2* and third principal components, *PC3* are directions of decreasing variance orthogonal to the first. This is done by examining the proportion of variance explained by each

principal component and selecting the two main components that carries the most of variance in the data as illustrated in the Figure 3.2.

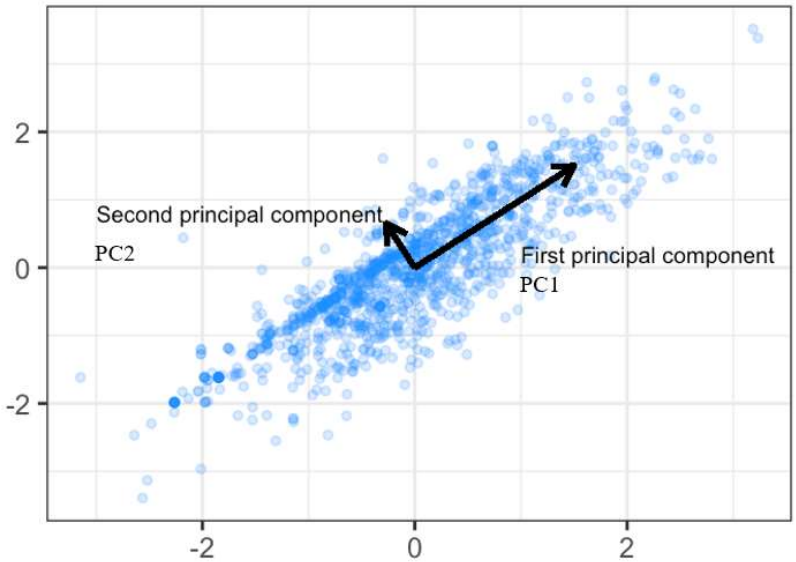


Figure 3. 2 Points variations around the two selected principal components points variations around the two selected principal components.

The eigenvalues obtained from the covariance matrix helps to interpret various attributes about the points in the voxels. The eigenvalues arranged in their decreasing values such that $\lambda_0 > \lambda_1 > \lambda_2 > 0$ The eigenvectors with the smallest eigenvalues correspond to the direction of minimum variance in the data (Narváez & Narváez, 2006). This direction is interpreted as the direction of planarity, as it represents the axis along which the points are most likely to lie on the flat surface. On the other case, the local surface normal is represented by the direction of the eigenvector with the largest eigenvalue.

3.3.2. Extraction of Minimal Planar Patches

A set of *n* points (math) from the voxel, V_t that describes the principal direction are fitted with a plane to extract the planar patch. The approach to fit a plane is presented a total least square (TLS) problem (Gene & Charles, 1980). In this method, plane fitting is performed in voxels by minimizing the total distance between each point and the plane, including the sensor's noise error herein referred as *residual value (r)*. The plane parameters are estimated by the normal vectors which is the eigenvector of least eigenvalue.

The general equation of a plane in 3D space is:

$$Ax + By + Cz + D = 0$$

Where A, B, and C are the plane's normal vector components, and D is the plane's distance for the origin point (0, 0, 0). Given a set of *n* the points with 3D coordinates (x_i, y_i, z_i), the plane fitting problem can be formulated as finding the values of parameters A, B, C and D that minimizes the total sum of square distances between the points and the plane. To enhance the accuracy of plane

fitting, the TLS is complemented with the sensor's standard noise error. The best-fit plane is then obtained as illustrated in the Figure 3.3, and all the points found on the plane are assigned accordingly. This activity returns a set of *k* minimal planar patches from each voxel.

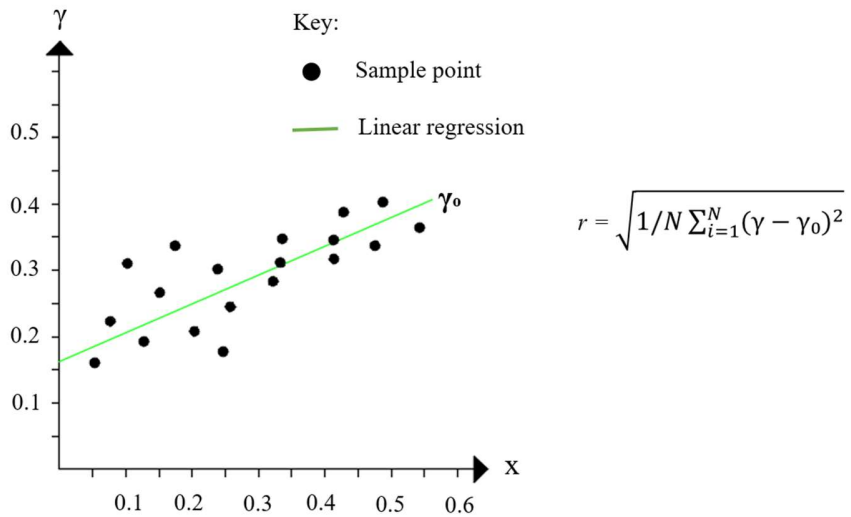


Figure 3. 3 Point distribution around the fitted plane on the x-y plane, and the estimated root mean square error.

3.4. Merging of Minimal Planar Patches using Region-growing

While the previous section describes the overall process of detecting the minimal planar patches within voxels, this section presents the merging of the planar patches across voxels based on proximity. The idea here is to join the minimal planar patches representing the same surfaces of structural components using the similar features used to detect them in the first place. To merge the planar patches across voxels, we first identify the neighboring voxels which are connected using predetermined criteria as described in section 3.4.1 Connectivity criteria for merging patches. Once the connected voxels are identified, the planar patches are checked if they share the same attributes for joining, and for similar patches they are merged by fitting a plane to the connected voxels using the TLS approach.

3.4.1. Connectivity of Voxels for merging planar patches

Selecting the connection between adjacent voxels is significant in merging the minimal planar patches considering high noise levels inherited in most point cloud data (Laine, 2013). In this study, the 26-voxel connectivity is selected, where every voxel has at least one vertex-neighbor i.e., that share a face, an edge, or a corner with it. This is considered as the most comprehensive type of connectivity and provide a more detailed representation of the spatial relationships between voxels compared to simpler connectivity rules, such as the 6-faces or 18-faces rules. However, its computational cost is higher due to the larger number of neighbors that need to be considered for each combination (Nourian et al., 2016).

A combination of 26-connected voxels each enclosing a set planar patch (math), such that their surface normal vectors (math) are drawn for comparison and merging if they have similar

orientation. This approach is similar to (Y. T. Su et al., 2016) except that the proximity is based on the connected voxels instead of the ϵ-distance threshold of the edging points between adjacent voxels. Therefore, using the voxel-proximity approach allows the consideration of merging two planar patches sourced from the same surface and are slightly disconnected due to occlusion or noise that otherwise the ϵ-distance threshold would disqualify them. Some studies have used *graph-based* methods to establish feature relationship among the connected voxels (Poux & Billen, 2019; Y. T. Su et al., 2016). In their strategies, each voxel was positioned as a node to create a voxel-to-voxel proximity relationship, in particular the (Poux & Billen, 2019) managed to embed each voxel with all related eigen-based features to extract the geometrical differences while using the log-Euclidean Riemannian metric to measure similarity between the adjacent voxels. In this study we opt to use surface normal vector to compare between the connected set of 26 voxels.

Figure 3.4 illustrates the connectivity analysis, showing two planar patches from voxels which have surface normal vectors oriented in the same directions.

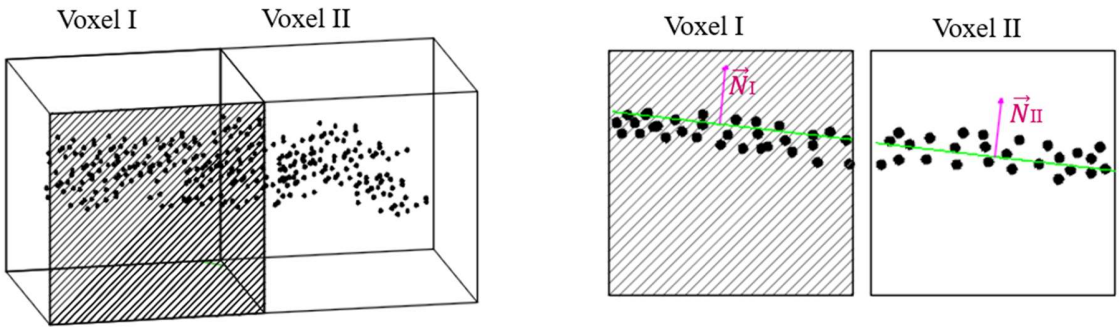


Figure 3. 4 Adjoining voxels containing planar patches with similar normal orientations as candidates for the merging.

3.4.2. Merging of Minimal Planar Patches

Once a set of k connected voxels where $k \leq 26$, having the same normal orientations are identified, the next step is to join together their corresponding planar patches as described in (Ntiyakunze & Inoue, 2023).

The merging process is done by finding the best-fit plane for the pair set using the TLS as explained in the section 3.3.2. Then, the process is repeated for all connected voxels until all suitable planar patches are merged. The residual value (r) is also used to refine the merged patches to ensure they are within the allowable noise levels depending on the sensory equipment used to obtain the point cloud.

It is important to note that merging planar patches across voxels can be challenging task, especially in the presence of high levels of noise or partial occlusions (Poux & Billen, 2019). Noise can be referred to the irregularities, inaccuracies, or inconsistencies in the point cloud data that are caused by various factors in the scene being captured and/or unintentionally introduced during the processing of the point cloud (Araújo & Oliveira, 2020). Noise caused by the nature of scene are related to the occlusions present in the scene. When objects are occluded, the sensors cannot capture data in certain areas, resulting in missing or incomplete data. Some conditions present in

scene during the scanning process can also add noises to the point cloud such as the ambient lighting and weather conditions. Noise can also arise from the devices and technique used to acquire the point cloud. This can relate to the measurement errors (limitations) of the sensors associated to the resolution angle, distance of the sensor and the surface being scanned, signal reflection resulting to duplicate data. Additionally, noise errors can be introduced when processing the point cloud often during registration of scans (Mostegel et al., 2017).

Merging patches at the voxel level significantly limits dealing with the overwhelming amount of noise which increases the accuracy and the overall computation time. Other techniques such as regularization or outlier rejection can also be used to address the noise challenges and improve the accuracy of the merging process.

3.5. Classification of merged planar patches

The newly merged planar patches, $S_i = \{S_1, S_2, \ldots, S_n\}$ are grouped into classifications based on the sizes of the patches. Here, the surface areas of patches, A_s, are used to define the patch's size instead of the number of points in a patch due to the expected variability of point densities. Patches with surface areas (A_s) equal to or more than 0.15m^2 are identified as *complete* (in our experiment) and are clustered together in a new set R consisting of n number of complete patches (i.e. $R_i = \{R_1, R_2, \ldots, R_n\}$ for $i = 1,2,\ldots n$). The rest of the patches are labeled incomplete also referred in the study as undergrown patches, which are clustered in a new set $\acute{R}$ comprising of m number of incomplete patches (i.e. $\acute{R}_j = \{\acute{R}_1, \acute{R}_2, \ldots, \acute{R}_m\}$ for $j = 1,2,\ldots m$ }. Points left unassigned after the refinement process are also appended to the set $\acute{R}$. The threshold value where $A_s \geq 0.15\text{m}^2$ is set to represent the expected minimum surface of any structural element that can be scanned in the scene.

However, this approach only merges planar patches based on their proximity defined by their connected/abutting voxels as shown in Figure 3.1(b), and in the condition of high levels of noise and occlusions which is common in scenes of occupied buildings, other coplanar patches or points can be distant (Hyunsoo & Changwan, 2021). This phenomenon is often described as missing points, which causes the coplanar surface patches to be scattered and disconnected. The next section discusses about how to overcome this challenge by searching for candidate coplanar patches or points.

3.6. Merging of remote disconnected coplanar patches

Disconnected coplanar patches refers to clusters of surface patches of various sizes that lie on the same plane but are separated from one another. One of the critical challenges in working with point cloud is to identify the disconnected coplanar patches and substantiate that they belong to the same surface of an object. Several algorithms have been developed to tackle this challenge. One of such algorithm is RANSAC (Random Sample Consensus) (Fischler & Bolles, 1981). RANSAC is an iterative algorithm used to find a subset of inlier observations where in this case are coplanar patches and provide a robust parameter estimation despite a pool of outlier points. However, the implementation of RANSAC consumes high computation time (Nguyen et al., 2013), and do not perform well with large point clouds and high levels of outliers and noise. In Figure 3.5, illustrates the results of RANSAC approach trying to connect various disconnected coplanar patches in a

noisy point cloud. It can be observed that dummy planes were established that do not represent any actual surface of the building.

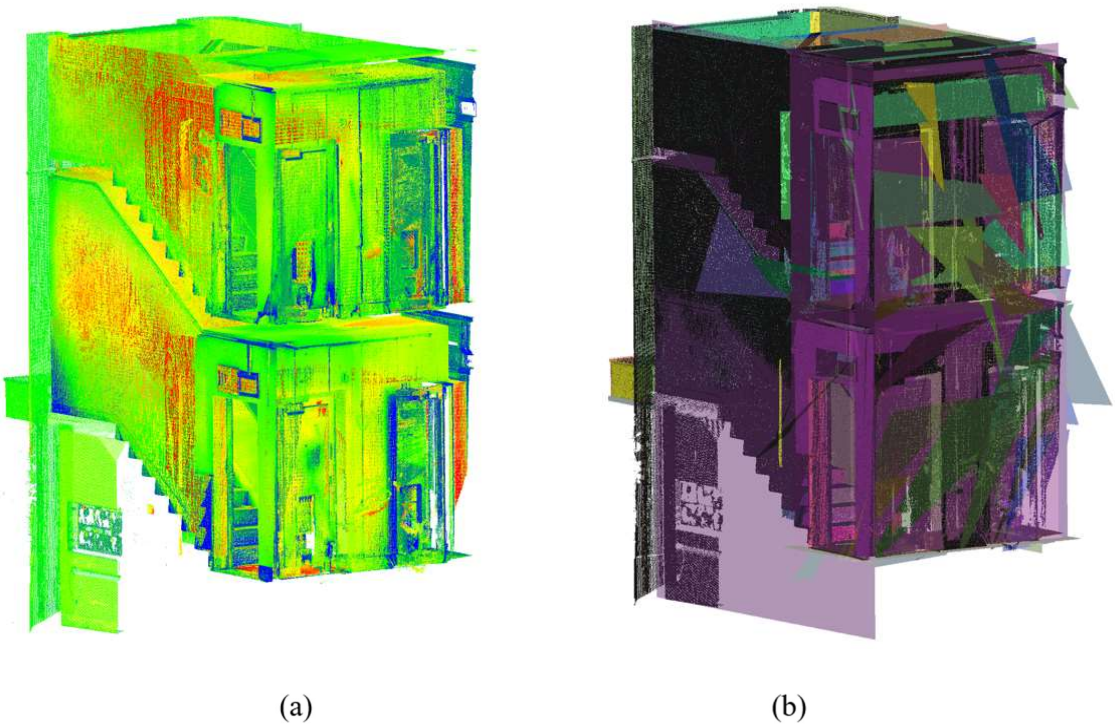


Figure 3. 5 Geometric fitting on 3D point cloud using RANSAC: (a) Original point cloud (b) Result after plane fitting using RANSAC with erroneous planes representing in different RGB colors.

3.6.1. Missing data in Point Clouds

The disconnected coplanar patches can be caused by noise or occlusions that are present either during the acquisition or preprocessing of point cloud, and this can result to missing data or holes on certain areas of surfaces particularly in walls, floors or ceilings (Hyunsoo & Changwan, 2021). Several studies have attempted to process point clouds containing missing data in their remodeling pipeline. In a study done by (Hyunsoo & Changwan, 2021), the authors encountered with the point cloud with incomplete sets of points on some objects that lead to missing portions of planes. They developed a method involving building a network of voxelized planar patches and then enlarging the voxel sizes within defined limitations to identify the overlapping surface patches. The process was repeated until no more overlaps were found and the network was updated recursively. The method managed to connect various discontinuities between the proximity elements.

(Klasing et al., 2008) Applied a k-d tree structure to search for nearby clusters of points representing the same flat surface with the help of radically bounded nearest neighbor (RBNN) strategy. The approach managed to reduce for searching the nearby clusters and its average computational complexity was still at $0(\frac{n}{K(average)}\log(n))$. In a study done in (Nurunnabi et al., 2012), a predetermined value of mean square error (MSE) was applied to estimate the lowest difference of the squared error (DMSE) between the pair sets of undergrown disconnected patches

that shares the similar features and in close distance. In the experiments, a fixed value of DMSE = $1.0e^{-04}$ was used as a criterion to merge the candidate disconnected pairs of patches.

Noises due to devices are usually estimated with reference to instrument specifications, while scene-related noises are difficult to measure due to the heterogeneousness of clutters. Noises due to non-permanent and permanent objects present in indoor scenes can obscure the segmentation process of structural elements especially when they are in the vicinity as demonstrated in Figure 3.6.

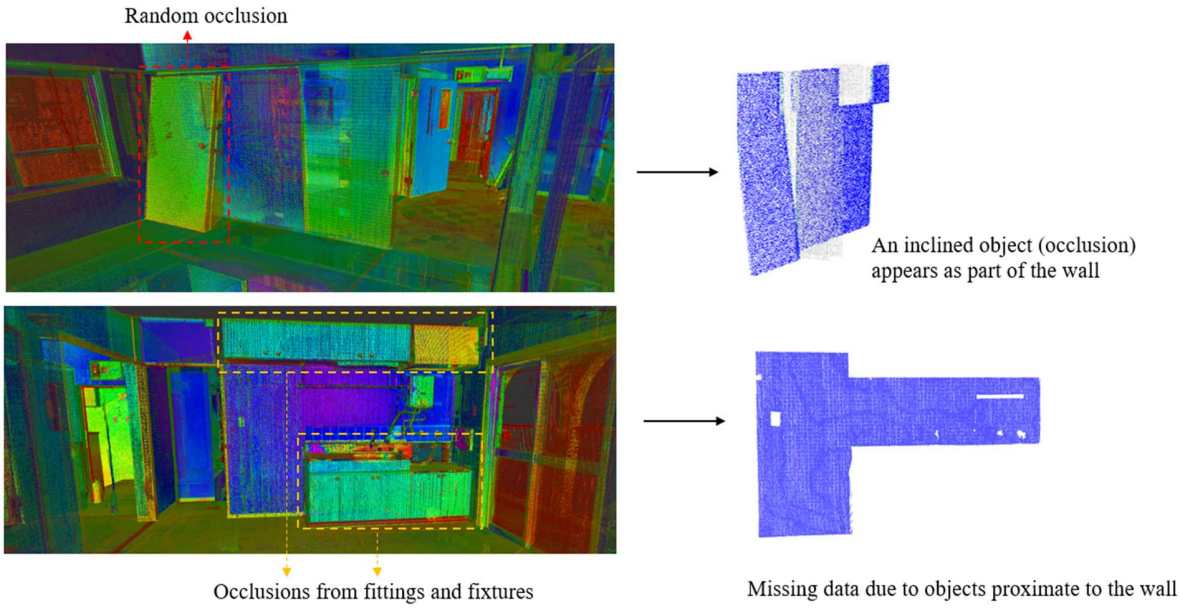


Figure 3. 6 Presence of occlusions along the walls.

Nonetheless, the mentioned studies have made use of patches or regions of coplanar points that are in proximity, which will be difficult to deliver when working with widely distant disconnected patches. Point clouds acquired in highly occluded sites bears significant amount of missing data points that requires a robust method to overcome them. Figure 3.7 shows the amount of missing data or holes created due to occlusion in point cloud.

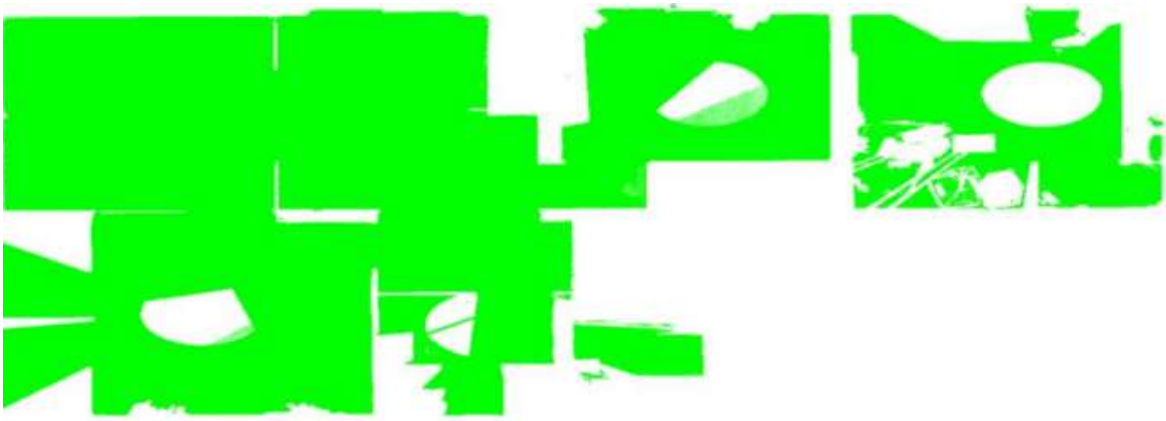


Figure 3. 7 Gaps on planar surfaces due to missing data caused by extensive occlusions.

To address the issue of large missing data, we developed a novel approach consisting of linearly interpolating the undergrown patches and unassigned points using the *patch-pair* correspondence referred in this study as the *principal patches,* $\sum$. The principal patches are empirically drawn from the set R. The use of pairs of principal patches helps to introduce the concepts of surface topology of structural elements that occurs in the pairs of opposing planar faces.

Afterward, the planes fitted onto the selected pairs of principal patches are orderly and systematically spanned to collect coplanar patches and points using the statistical inferences drawn from the known samples i.e., *principal patches*. We further scrutinize the candidate coplanar patches by checking for the presence of outliers and later validate their coplanarity.

3.6.2. Criteria for selecting pairs of principal planar patches

In this subsection, we deal with patches from the set R_n $\{R_i = R_1, R_2, \ldots, R_n\}$ to determine the principal patches. At this point, there is a considerable number of undesirable planar patches due to noises and clutters, hence we initialize the filtration process by uncovering the sets of parallel-pairs of principal patches. The essence of pairing the patches is to infer the surface topology of generic structural elements such as floor slabs, columns, walls and beams, which are bounded by pairs of opposing surfaces in a standard offset distance corresponding to the thicknesses or depths.

To determine the principal patches ($\sum$), we first extract the following attributes from patch set, R_n: *normal vectors* ($\vec{n}_i$), *patch's surface area* (A_s), and *patch's centroid*, μ. The centroid (μ) of a particular patch, R_i containing k points set Ω, ($\Omega = \{P_1, P_2, P_3, \ldots, P_k\}$) can be defined as in Equation (3.1).

$$\mu = \sum_{i=1}^{k} (x_i, y_i, z_i) \tag{3.1}$$

These attributes helps to uncover the surface and spatial properties of patches in order to find the similarities and proximity between patches. Four criteria are developed to determine the principal pairs of patches, $\sum_i$ & $\sum_{i+1}$ (for $\sum \in R_n$). Selection of the pairs of principal patches are based on the concept of parallel patches that are in vicinity, and in addition the proposed criteria enhance the structural-based knowledge into the selection process.

(a) Criteria 1 - Orthogonal distance between planar patches

Using a predetermined orthogonal distance (t) between the two pairs of candidate patches (R_i & R_{i+1}) reduces the computation resources for searching the correspondence pair sets. The orthogonal distance is estimated by finding any point, P_i (i.e. $P_i = \{x_i, y_i, z_i\}$) on the patch's plane (Π_i) and compute the shortest distance between this point (P_i) and other patch's plane (Π_{i+1}) along the direction of the normal vectors. Conditions for normal vectors and their absolute orientations are discussed in detail in the *criteria 2*.

The threshold distance (t) between the candidate's pair patches (R_i & R_{i+1}) is used to describe the maximal (tolerance) of the orthogonal offset distance between the two parallel patches, that stipulates the standard thickness of the common structural elements in buildings. The value of t,

where $t \leq 50\ cm$ between the two neighboring parallel planar patches, is used to assign the candidate pairs of principal patches ($\sum_i$ & $\sum_{i+1}$).

(b) Criteria 2 – Angular deviation between the planar patches

Applying the angular deviation, θ between the two candidate planar patches is used to refine the selection of principal patches in their parallelism. The angular value (θ) between the pair patches (R_i & R_{i+1}) which is defined by the dot product of their respective normal vectors computed as in Equation (3.2).

$$arccos\ (\vec{n}_i\ .\ \vec{n}_{i+1}) \leq \theta \tag{3.2}$$

where $\vec{n}_i$, $\vec{n}_{i+1}$ are the estimated normal vectors of R_i, R_{i+1} planar patches respectively. We consider the maximum angular deviation (θ) of not more than 5°.

(c) Criteria 3 – Size difference between the pairs of planar patches

This part deals with the comparison of patch coverage by their surface areas. The two parallel patches (R_i & R_{i+1}) with surface areas, $A_{s(i)}$ and $A_{s(i+1)}$, respectively, are compared against one another to enhance the selection of true patches representing structural elements as opposed to occluding objects in the vicinity.

To calculate the surface area of each patch, we use the points contained in the patch (R) instead of its fitted plane (Π) to obtain accurate estimates. Hence, for a given patch, R_i containing k number of points, $\{P_i = P_1, P_2,...,P_k\}$, a polygon is created to connect all points using a convex hull based on a Quickhull algorithm (Mei et al., 2012). Convex hull polygon represents the smallest convex (set of triangles) that encloses all the points of the planar patch. Now, after obtaining the polygon, its area is calculated using a shoelace formula (Y. Lee & Lim, 2017), The shoelace method works by computing the aggregate sum of the signed areas of all the triangles that make up the polygon as provided in Equation (3.3).

$$surface\ area = \frac{1}{2} \left| \sum_{i=1}^{n-1} X_i Y_i{}^{+}{}_1 + X_n Y_1 - \sum_{i=1}^{n-1} X_i + Y_i - X_1 Y_n \right| \tag{3.3}$$

Where the X and Y are the corresponding coordinates of the triangle's vertices.

Once the area of each patch is determined, we set a condition regarding a difference in their surface areas (d_A) two adjacent and parallel patches using Equation (3), such that the ratio $d_A/A_{s(i)}$ or $d_A/A_{s(i+1)}$ (whichever is larger between $A_{s(i)}$ and $A_{s(i+1)}$) is not more than 20%. We opted to use the surface areas instead of the number of points in a patch, which is influenced by point density that varies depending on the distance between the sensor and an object (Rabbani et al., 2006).

$$d_A = |A_{s(i)} - A_{s(i+1)}| \tag{3.4}$$

(d) Criteria 4 – Overlap between the pairs of planar patches

The above criteria deal with the adjacency and matching sizes between two parallel planar patches, however, due to the expected high-level noises, the two patches may still represent surfaces from two different objects. To solve this, we introduce a patch-alignment criterion

whereby the two parallel patches must coherently overlap by more than 50% of their surface areas (A_s) as displayed in Figure 3.8.

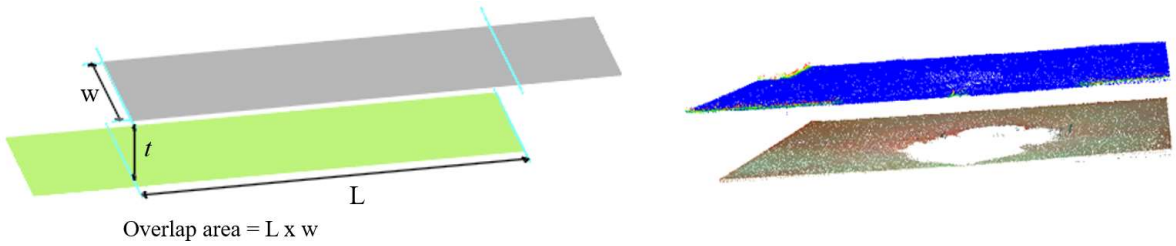


Figure 3. 8 Presence of occlusions along the walls.

The overall procedures in section 3.6.2.a–d return a set of n number of parallel patches referred as the *principal patches*, $\sum$, for $\sum_n = \{\sum_1, \sum_2, \ldots, \sum_n\}$ and their associated fitted-planes ($\mathbb{\Pi}_n$).

3.7. Spanning of Principal Planes

Given a particular principal patch ($\sum_i$), we span its corresponding best-fit plane ($\mathbb{\Pi}_i$) to locate the disconnected and remote coplanar patches. Span of the plane ($\mathbb{\Pi}_i$) referred as $Span(\mathbb{\Pi}_i)$ (such that $Span(\mathbb{\Pi}_i) \in \Re^3$) is initialized by finding set of linear combination of vectors on the plane, $\mathbb{\Pi}_i$. Let $\vec{u}_i$ be a set of spanning vectors of the plane, $\mathbb{\Pi}_i$, such that $\vec{u}_i = \{\vec{u}_1, \vec{u}_2, \ldots, \vec{u}_n \mid \vec{u}_i \text{ in } \Re^3\}$. Each spanning vector corresponds to a particular point on the plane i.e. $P_i = \{X_i, Y_i, Z_i\}$ as illustrated on the Figure 3.9. The span of $\vec{u}_1, \vec{u}_2, \ldots, \vec{u}_n$ is the collection of all linear combinations of $\vec{u}_1, \vec{u}_2, \ldots, \vec{u}_n$ denoted as $Span(\vec{u}_1, \vec{u}_2, \ldots, \vec{u}_n)$ which is commutative as described in Equation (3.5) The linear combination ($\boldsymbol{W}$) of these vectors with their corresponding scalar values (coefficients), $\{C_i = C_1, C_2, \ldots, C_n \mid C_i \text{ is a real number}\}$ is shown in Equation (3.6).

$$Span(\vec{u}_1, \vec{u}_2) = Span(\vec{u}_2, \vec{u}_3) = Span(\vec{u}_1, \vec{u}_3) \text{ for each } \mathrm{Span}(\vec{u}_i) = \mathrm{Span}\begin{Bmatrix} x_i \\ y_i \\ z_i \end{Bmatrix} \tag{3.5}$$

$$\boldsymbol{W} = C_1\vec{u}_1 + C_2\vec{u}_2 + \ldots + C_n\vec{u}_n \tag{3.6}$$

Spanning of a plane is instructed to occur in various directions in a vector spaces, $\boldsymbol{V}$ (for $\boldsymbol{V} \in \Re^3$) and the extent of the span is modified by adjusting the scalars values (C_i). The spanning is terminated at the *local maxima* and *local minima* in each direction of *-x, -y, -z* of the input point cloud (i.e. *min-x, max-x, min-y, max-y, min-z and max-z*). Placing the spanning constraints helps to avoid the unnecessary expense of computing the infinity planes ($\mathbb{\Pi}_\alpha$), which reduces the search for approximate coplanar patches and/or points.

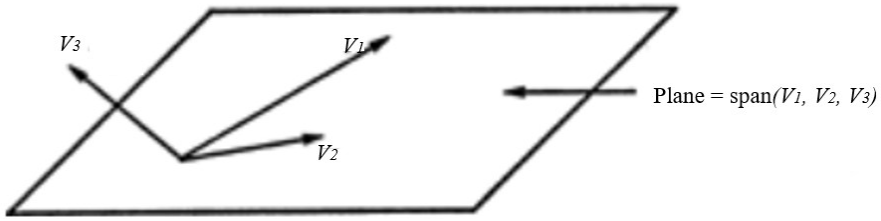


Figure 3. 9 Spanning vectors on a plane.

3.7.1. Locating coplanar points onto the spanned planes

This section deals with assigning points and undergrown patches from the set $\acute{R}$ onto the spanned principal planes (*Span*($\mathbb{\Pi}$)) in order to determine the remaining coplanar patches or points. First, we check for the points contained in the set $\acute{R}$ that lie on any of the spanned plan ($\mathbb{\Pi}_i$). This can be done by substituting the coordinates of the point into the equation of the plane and see if satisfies the equation of plane. For a point, $Pi = \{x_i, y_i, z_i) \mid P_i \in \text{set } \acute{R} \in \Re^3$, should satisfy the general equation of a particular spanned plane, Ax + By + Cz = D such that $Ax_i + By_i + Cz_i = D$ as well.

However, it is often uncommon for points to lie perfectly on the fitted plane due to the expected regression errors, and the other hand, not all points found on the vector space (***V***) of the spanned plane (*Span*($\mathbb{\Pi}$)) or closely to it, are deemed to represent the similar object surface. Therefore, several strategies are proposed to locate true coplanar.

The process is initiated by calculating the closest points from the spanned plane and check if they are actually coplanar with reference to the spanned principal plane. A distance threshold (δ_{dist}) is used as a measure of points, P_i contained in the set $\acute{R}$ to the spanned plane (i.e., $P_i = \{x_i, y_i, z_i) \mid P_i \in \text{set } \acute{R} \in \Re^3$. This is a preliminary screening step to locate all the closest candidate points fitting the distance criteria. Points falling within the allowable distance threshold (*f*) are provisionally assigned to the closest spanned plane (*Span*($\mathbb{\Pi}_i$)). Afterward, the assigned points are screened for outliers and measure their coplanarity based on the neighborhood (N) using PCA.

3.7.2. Points-to-Plane-Assignment

This part describes the initial step of assigning the nearby points to the spanned plane, and in this case the closest distance to the plane is computed which is the orthogonal distance. Hence, points in the set $\acute{R}$ are drawn and measured their orthogonal distances (*f*) to the spanned planes such that. $f\colon P_i \rightarrow Span(\mathbb{\Pi}_i) \mid P_i \in \text{set } \acute{R} \in \Re^3$, where *f* denotes the orthogonal distance of a particular point P_i to a spanned plane, *Span*($\mathbb{\Pi}_i$), within a pre-signed allowable distance (δ_{dist}). The distance (*f*) of an arbitrary point, P_i, to a plane represented in a general form: Ax + By + Cz + D = 0 where (A, B, C) denotes the unit normal vector, $\vec{N}$, is computed as Equation (3.7);

$$f = \frac{|\mathrm{Ax1} + \mathrm{By1} + \mathrm{Cz1}|}{\sqrt{\mathrm{A}^2 + \mathrm{B}^2 + \mathrm{C}^2}} \tag{3.7}$$

The threshold value, δ_{dist}, is used to set a tolerance distance of a candidate point to the spanned plane, which is partially related to the concept of the *maximum distance* of sample-to-plane *(MDP)* as applied in (Araújo & Oliveira, 2020; H. Chen et al., 2022). The value of δ_{dist}, determined by inferring the sample distribution from the principal patches ($\sum_i$) using a median estimator. The median estimator is used in the process instead of the mean value (μ) because the median value is more useful when dealing with skewed data and outliers due to its robustness and insensitive to extreme values, providing a better representation of the central tendency of the dataset (Rousseeuw & Croux, 1993).

Given a set Q containing k points ($p_i = p_1, p_2, \ldots, p_k$) from a particular principal patch ($\sum_i$) with its corresponding best-fit plane ($\mathbb{\Pi}_i$), for each point in the set Q, we compute the *point-to-plane* distances (f), forming a new set of measurements, $Đ = \{đ_1, đ_2, \ldots, đ_q\}$ provided using Equation (3.7). To incorporate the noise errors, we adopt the use of *median absolute deviation (MAD)* to estimate the deviation of points around the plane (planar patch) as used in (Araújo & Oliveira, 2020).

From the set Đ, the MAD is computed using Equation (3.8) and used it to calculate the value of δ_{dist}. δ_{dist} is designed to set the maximum limit over the median value using the standard deviation (σ), which is equivalent to MAD in our case as shown in Equation (3.9).

$$MAD = \Omega \times median\ (|p_i - median(Đ)|) \tag{3.8}$$

Where p_i represents a point in the set Q and $\Omega = 1.4826$, representing a constant value in the normal distribution for the consistent coherence between the MAD and standard deviation values (Limberger & Oliveira, 2015).

$$\delta_{dist.} = [Median(Đ) + p\ MAD(Đ)] \tag{3.9}$$

Using a probability distribution, we use the second quantile (2σ), $p = 2$, as shown in Figure 3.10, which results to a confidence interval of 95.45% to select true samples (inliers) around the median value.

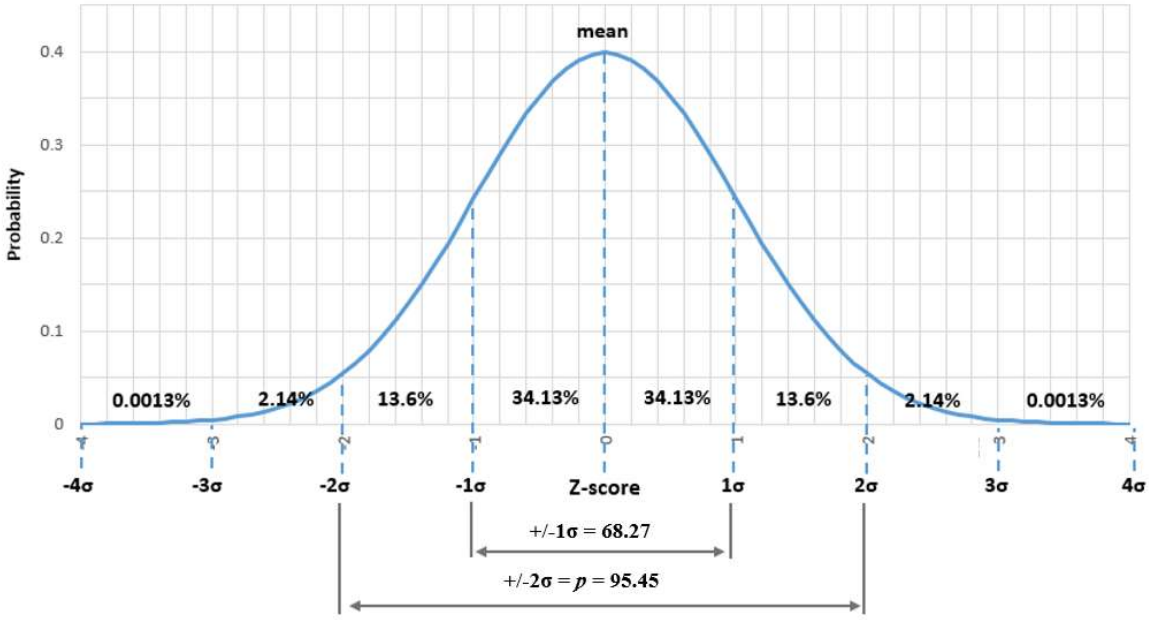


Figure 3. 10 Probability distribution curve

Points that lie within the δ_{dist} are assigned to the corresponding plane ($Span(\mathbb{I}_i)$) as the *provisional coplanar points*. In addition, the points falling between two spanned planes and are within the allowable distance threshold (δ_{dist}) are assigned to the nearest plane judged by the smallest orthogonal distance. The output of this process is the number of sparse regions of points over the course of spanned planes are shown in Figure 3.11. This is due to the inherited amount of noise and multi-dense properties in those regions that created the set $\acute{R}$ in the first place. The next step involves the screening for outliers in regions based on the clustering method.

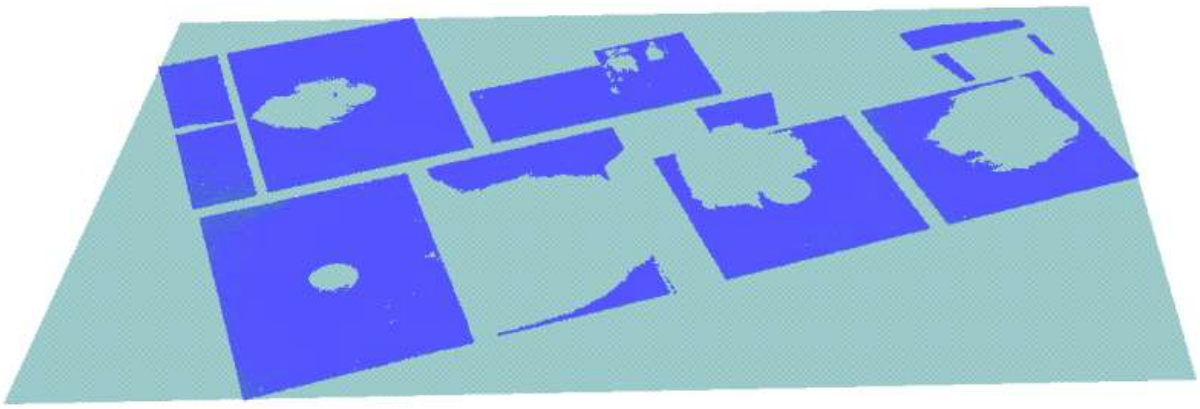


Figure 3. 11 Disconnected coplanar regions as the result of occlusion present in the scene.

3.7.3. Testing for outliers

In the raw point cloud dataset, the measurement of noise errors is commonly used to detect outliers by setting the allowable offsets of samples-to-plane. However, it is difficult to measure the

noise levels for raw 3D point clouds, as it is biased toward the sensors used and the noises may contaminate all three components (xyz) of the coordinate system (Huang & Menq, 2001). Our approach is independent of the sensor's noise levels; instead, we apply the knowledge acquired from the dataset. We use the statistical distribution and neighborhood structure of the samples to eliminate the outliers. Intuitively, the provisionally assigned coplanar points detected in Section 3.5.3a form K number of sparse points across the spanned planes. For this reason, we used the *Density-Based Spatial Clustering of Applications with Noise* (DBSCAN) approach to detect the outliers.

The DBSCAN algorithm is mainly used for clustering analysis, but it can also be used to detect outliers for n-dimensional datasets. The approach works by partitioning the provisional point sets assigned to the spanned planes into distinct groups of points based on their density defined by a certain neighborhood and then identifying any points that do not belong to any of these groups as *outliers*.

In DBSCAN, a point is considered a core point (Č) if there are at least a certain number of points, *MinPts,* neighboring points within a given radius, *eps*. A neighboring point is within the radius if its distance from the core point is less than or equal to *eps*. Points that are not core points but still within the radius of a core point (*eps*) are considered to be border points. Finally, any points that are neither core points nor border points are considered as *outliers* (X. Zhang et al., 2022).

To detect outliers, the algorithm first identifies the core points (Č) and border points in the from the input point set. Here, all the provisionally assigned points from each spanned plane, $Span(\mathbb{\Pi}_i)$ are drawn separately, and used as input in the algorithm. The next step is to form clusters by connecting the core points and their associated border points. Any points that are not assigned to any cluster are classified as outliers and unassigned/removed from the spanned plane.

The advantage of using DBSCAN for outlier detection is that it is not affected by the shape of the dataset, can handle noise and it performs well even when the points are unordered which is the case when dealing with the point cloud datasets (Sawant, 2014). Also, it performs well in large datasets as the case of this study. However, it requires parameter tuning (i.e. *MinPts* and *eps*) which can be challenging in practice especially working with diverse point cloud. For optimum selection of the parameters, the Varied DBSCAN (VDBSCAN) approach (Liu et al., 2007) is employed to explicitly select parameters which are proportional to the density variations of the dataset.

In the VDBSCAN, each point is assigned a specific density threshold that determines the *MinPts* required for the point to be considered a core point (Č). This threshold is calculated based on the distance to the k-nearest neighbor or each point, where k is a user-defined parameter and a guide towards determining the *eps*. Specifically, the threshold is set as the average distance between a point and its k-nearest neighbors. By assigning different density thresholds to each point, VDBSCAN can identify clusters with varying densities. Point in dense regions will have a higher

density threshold and require more k-neighbors to be considered a core point (Č), while points in sparse regions will have a relatively lower density threshold and require fewer neighbors to be considered a core point.

The VDBSCAN algorithm proceeds in a similar way to the traditional DBSCAN (Ester et al., 1996), starting with a random non-visited point and expanding the neighborhood cluster around it. However, in the VDBSCAN the density threshold of each point is used to determine its neighbors, instead of a fixed parameter. The algorithm continues until all points have been visited and assigned to a cluster.

Therefore, the unique values of *eps* are obtained at different densities and the corresponding minimum numbers of neighboring points (*MinPts*) are subsequently determined as shown in Equation (3.10).

$$MinPts_{\mathrm{i}} = \frac{1}{n}\sum_{i=1}^{n} Pi \tag{3.10}$$

where P_i ($i = 1, 2, \ldots, n$) is the number of points in the *eps* neighborhood for point *i*.

Points that are left *un*-clustered are identified as *outliers* and eliminated from the spanned plane. This process outputs a number of *k* clusters, $C_i = \{C_1, C_2, \ldots, C_k\}$, as inliers to spanned planes, and the next step involves testing if the cluster and the points are coplanar.

3.7.4. Validation for Coplanarity

For each cluster detected in the previous process (C_i), the check for their coplanarity is performed using PCA by measuring their principal directions with respect to the directions of their assigned spanned planes (i.e., $Span(\mathbb{\Pi}_i)$). Similar to the approach in Section 3.3, the covariance matrix, **C**, of neighboring samples, *N*, in the cluster (C_i) decomposes and returns three eigenvectors $\vec{v}_0$, $\vec{v}_1$, $\vec{v}_2$, and their corresponding eigenvalues $\lambda_2 \geq \lambda_1 \geq \lambda_0$. The plane is fitted along the direction of the vectors representing the highest variation, i.e., $\vec{v}_1, \vec{v}_2$. Eigenvector, $\vec{v}_0$ defines normal vectors $\vec{n}$ of clusters.

Then, the orientations of each of the cluster's normal vector,$\vec{n}_{\mathrm{i}}$ are matched with the principal plane's normal vector, $\vec{N}$. The cluster is identified as coplanar if the angle between the two normal vectors ($\vec{n}$ and $\vec{N}$) is within a maximal normal deviation (MND) of 2° (*in our experiment*) as provided in equation (3.11). Otherwise, the cluster and all its elements (i.e., points) are rejected for further processing. The process is iterated for all the assigned clusters relative to their spanned planes.

$$\arccos\,(\vec{n}_i \,.\, \vec{N}_i) \leq 2^\circ \tag{3.11}$$

We provide a smaller MND threshold for the clusters i.e. 2° compared to the MND of 5° that is adopted earlier in selecting the principal patches, this is because larger patches (i.e. principal patches,) have admit relatively more normal deviations (Araújo & Oliveira, 2020).

3.8. Refining the pairs of principal patches

Once the coplanar points are distinctly assigned to their corresponding spanned principal planes, *Span*($\mathbb{\Pi}$), the next process involves refining and arranging the planes including their associated patches in order to make sure that the pairs of principle planes are properly aligned and matched in terms of coverage of points.

There are several methods used to refine planar patches. One common approach is to use iterative closest point (ICP) algorithms (Besl & McKay, 1992), which is a classical approach that iteratively refine the relative position and orientation of the patches based on the correspondence between their points. Another approach is to used feature-based methods, which rely on extracting and matching distinctive features in the patches such as corner, edges, or orientations (Förstner, n.d.; Kaiser et al., 2020; Verginis, 2013).

In our approach each pair of the spanned principal planes, i.e., *Span*($\mathbb{\Pi}_i$) & *Span*($\mathbb{\Pi}_{i+1}$), is traversed to determine the number of points and their associated patches (clusters). Then, we compared the points coverage between the planes in each pair set using the criteria applied in Section 3.5.1b. The qualified pair sets are then subjected to a plane classification procedure as described in Section 4.1.

References

Araújo, A. M. C., & Oliveira, M. M. (2020). A robust statistics approach for plane detection in unorganized point clouds. *Pattern Recognition*, *100*. https://doi.org/10.1016/j.patcog.2019.107115

Besl, P. J., & McKay, N. D. (1992). A Method for Registration of 3-D Shapes. In *IEEE Transactions on Pattern Analysis and Machine Intelligence* (Vol. 14, Issue 2, pp. 239–256). https://doi.org/10.1109/34.121791

Chen, H., Liang, M., Liu, W., Wang, W., & Liu, P. X. (2022). An approach to boundary detection for 3D point clouds based on DBSCAN clustering. *Pattern Recognition*, *124*. https://doi.org/10.1016/j.patcog.2021.108431

Ester, M., Kriegel, H.-P., Sander, J., & Xu, X. (1996). A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, 226–231.

Fischler, M. A., & Bolles, R. C. (1981). RANSAC: Random Sample Paradigm for Model Consensus: A Apphcatlons to Image Fitting with Analysis and Automated Cartography. *Graphics and Image Processing*, *24*(6), 381–395.

Forcino, F. L. (2012). Multivariate assessment of the required sample size for community paleoecological research. *Palaeogeography, Palaeoclimatology, Palaeoecology*, *315–316*, 134–141. https://doi.org/10.1016/j.palaeo.2011.11.019

Förstner, W. (n.d.). *Efficient and Accurate Registration of Point Clouds... - Google Scholar*. https://scholar.google.com/scholar?hl=en&as_sdt=0%2C39&q=Efficient+and+Accurate+Registration+of+Point+Clouds+with+Plane+to+Plane+Correspondences&btnG=

Gene, G., & Charles, van L. (1980). An Analysis of the Total Least Squares Problem. *SIAM Journal on Numerical Analysis*, *17*(6). https://doi.org/10.1137/0717073

Huang, J., & Menq, C. H. (2001). Automatic data segmentation for geometric feature extraction from unorganized 3-D coordinate points. *IEEE Transactions on Robotics and Automation*, *17*(3), 268–279. https://doi.org/10.1109/70.938384

Hyunsoo, K., & Changwan, K. (2021). 3D as-built modeling from incomplete point clouds using connectivity relations. *Automation in Construction*, *130*(January). https://doi.org/10.1016/j.autcon.2021.103855

Kaiser, A., Zepeda, J. A. Y., & Boubekeur, T. (2020). *Plane Pair Matching for Efficient 3D View Registration*. *February*. http://arxiv.org/abs/2001.07058

Klasing, K., Wollherr, D., & Buss, M. (2008). A clustering method for efficient segmentation of 3D laser data. *Proceedings - IEEE International Conference on Robotics and Automation*, *June 2014*, 4043–4048. https://doi.org/10.1109/ROBOT.2008.4543832

Laine, S. (2013). A topological approach to voxelization. *Computer Graphics Forum*, *32*(4), 77–86. https://doi.org/10.1111/cgf.12153

Lee, Y., & Lim, W. (2017). Shoelace Formula: Connecting the Area of a Polygon and the Vector Cross Product. *The Mathematics Teacher*, *110*(8), 631–636.

https://doi.org/10.5951/mathteacher.110.8.0631

Limberger, F. A., & Oliveira, M. M. (2015). Real-time detection of planar regions in unorganized point clouds. *Pattern Recognition*, *48*(6), 2043–2053. https://doi.org/10.1016/j.patcog.2014.12.020

Liu, P., Zhou, D., & Wu, N. (2007). VDBSCAN: Varied Density Based Spatial Clustering of Applications with Noise. *International Conference on Service Systems and Service Management*, 1–4. https://doi.org/10.1109/ICSSSM.2007.4280175

Mei, G., Tipper, J. C., & Xu, N. (2012). An algorithm for finding convex hulls of planar point sets. *Proceedings of 2nd International Conference on Computer Science and Network Technology, ICCSNT 2012*, 888–891. https://doi.org/10.1109/ICCSNT.2012.6526070

Mostegel, C., Prettenthaler, R., Fraundorfer, F., & Bischof, H. (2017). Scalable surface reconstruction from point clouds with extreme scale and density diversity. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, *2017-Janua*, 2501–2510. https://doi.org/10.1109/CVPR.2017.268

Narváez, E. A. L., & Narváez, N. E. L. (2006). Point cloud denoising using robust principal component analysis. *GRAPP 2006 - Proceedings of the 1st International Conference on Computer Graphics Theory and Applications*, *53*, 51–58. https://doi.org/10.5220/0001358900510058

Nguyen, H. H., Kim, J., Lee, Y., Ahmed, N., & Lee, S. (2013). Accurate and fast extraction of planar surface patches from 3D point cloud. *Proceedings of the 7th International Conference on Ubiquitous Information Management and Communication, ICUIMC 2013*, *October 2015*. https://doi.org/10.1145/2448556.2448640

Nourian, P., Gonçalves, R., Zlatanova, S., Ohori, K. A., & Vu Vo, A. (2016). Voxelization algorithms for geospatial applications: Computational methods for voxelating spatial datasets of 3D city models containing 3D surface, curve and point data models. *MethodsX*, *3*, 69–86. https://doi.org/10.1016/j.mex.2016.01.001

Ntiyakunze, J., & Inoue, T. (2023). Segmentation of Structural Elements from 3D Point Cloud Using Spatial Dependencies for Sustainability Studies. *Sensors*, *23*(4). https://doi.org/10.3390/s23041924

Nurunnabi, A., Belton, D., & West, G. (2012). Robust segmentation for multiple planar surface extraction in laser scanning 3D point cloud data. *Proceedings - International Conference on Pattern Recognition*, *Icpr*, 1367–1370.

Osborne, J. W., & Costello, A. B. (2004). Sample size and subject to item ratio in principal components analysis. *Practical Assessment, Research and Evaluation*, *9*(11).

Poux, F., & Billen, R. (2019). Voxel-based 3D Point Cloud Semantic Segmentation: Unsupervised geometric and relationship featuring vs deep learning methods. *ISPRS International Journal of Geo-Information*, *8*(5). https://doi.org/10.3390/ijgi8050213

Pulli, K., Duchamp, T., Hoppe, H., McDonald, J., Shapiro, L., & Stuetzle, W. (1997). Robust meshes from multiple range maps. *Proceedings of the International Conference on Recent Advances in 3-D Digital Imaging and Modeling*, 205–211.

https://doi.org/10.1109/im.1997.603867

Qian-Yi, Z., Jaesik, P., & Vladlen, K. (2018). *Open3D: A Modern Library for 3D Data Processing* (0.16.0). https://doi.org/10.48550

Rabbani, T., van den Heuvel, F. a, & Vosselman, G. (2006). (impo)(Fashuai exper+Sudan recom)Segmentation of point clouds using smoothness constraint. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences - Commission V Symposium "Image Engineering and Vision Metrology," 36*(5), 248–253. http://www.isprs.org/proceedings/XXXVI/part5/paper/RABB_639.pdf

Rousseeuw, P., & Croux, C. (1993). Alternatives to the Median Absolute Deviations. *Journal of the American Statistical Association*, *88*(424), 1273–1283. https://doi.org/10.1080/01621459.1993.10476408

Sawant, K. (2014). Adaptive Methods for Determining DBSCAN Parameters. *IJISET - International Journal of Innovative Science, Engineering & Technology*, *1*(4), 329–334. www.ijiset.com

Shaukat, S. S., Rao, T. A., & Khan, M. A. (2016). Impact of sample size on principal component analysis ordination of an environmental data set: Effects on eigenstructure. *Ekologia Bratislava*, *35*(2), 173–190. https://doi.org/10.1515/eko-2016-0014

Su, Y. T., Bethel, J., & Hu, S. (2016). Octree-based segmentation for terrestrial LiDAR point cloud data in industrial applications. *ISPRS Journal of Photogrammetry and Remote Sensing*, *113*, 59–74. https://doi.org/10.1016/j.isprsjprs.2016.01.001

Verginis, C. (2013). *Plane registration using uncertainties*. *April 2013*. https://doi.org/10.13140/RG.2.1.2743.5600

Vo, A. V., Truong-Hong, L., Laefer, D. F., & Bertolotto, M. (2015). Octree-based region growing for point cloud segmentation. *ISPRS Journal of Photogrammetry and Remote Sensing*, *104*, 88–100. https://doi.org/10.1016/j.isprsjprs.2015.01.011

Xu, Y. (2019). Reconstruction of building objects from point clouds of built environment and construction sites [Technical University of Munich]. In *phD thesis*. https://www.semanticscholar.org/paper/Reconstruction-of-building-objects-from-point-of-Xu/f6a6f74a97b37e3b59e75032343c55cad51cdab7#citing-papers

Zhang, X., Shen, X., & Ouyang, T. (2022). Extension of DBSCAN in Online Clustering: An Approach Based on Three-Layer Granular Models. *Applied Sciences (Switzerland)*, *12*(19). https://doi.org/10.3390/app12199402

Chapter 4. Segmentation of Structural Elements

4.1. Introduction

This chapter focusses in segmenting the extracted planar patches in their pairwise groupings. The segmentation process is initialized by clustering the patch pairs based on their spatial locations and geometric properties such as size, orientations, group the patches belong to the same structural element. The following step involves analyzing the clusters to identify the structural elements they present, such as floors, beams, columns, or walls, using the knowledge about the structural forms specifically in their spatial dependencies, and the geometry rules in classification of points. The segmentation process also goes in line with validating the identified points matching the structural elements with their geometric and topological consistency with the overall building model, for instance checking the consistency of height differences between floors with the heights of vertical elements such as walls or columns or location of beams and ceilings along the z-values.

4.2. Plane arrangements

Once the planar patches are implicitly validated and grouped in pairwise planes ($\mathbb{\Pi}$), they are now arranged with respect to the standard orientations of the generic reinforced concrete (RC) structural elements in consideration: *horizontal elements* (floors, suspended slabs, and beams) and *vertical elements* (walls and columns). Horizontal elements refer to the elements that provide lateral stability to the building and resist the forces caused by wind or seismic activity, such as floor slabs, roofs, floor beams, cantilever beams, foundation beams, foundation footings and bracing systems. The vertical structural elements refer to the elements that provide vertical support to the building and transfer loads from the upper levels (roofs) to the foundation, and they include columns and walls. This study deals only with the *floors, suspended slabs,* and *beams* as the horizontal members, and for the vertical members are *columns* and *walls*.

The arrangement of the plane pairs is setup with accordance to the orientation of the planes referring to their normal directions ($\vec{n}$), which is categorized in the following manner:

- Horizontal Planes, $\mathbb{\Pi}_h$ (including the nearly horizontal planes): comprises pairs of planes with their normal vectors, $\vec{n}_h$, oriented at an angle, $\Theta < 45°$ to the *z-axis*;
- Vertical Planes, $\mathbb{\Pi}_v$ (including nearly vertical planes): comprises pairs of planes with normal vectors, $\vec{n}_v$, oriented at an angle, $\Theta \geq 45°$ to the *z-axis*.

The set of two planes in a pair-set is denoted as i and j, such that $\mathbb{\Pi}_{h(i,j)}$ refers to the horizontal plane pairs and $\mathbb{\Pi}_{v(i,j)}$ refers to the vertical plane pairs. After all pairs of planes are arranged in their respective orientations i.e. *horizontal* or *vertical*, we start to classify the planes by matching their local feature properties with the physical (*global*) attributes of structural elements. From each plane-pairs, the following local geometric properties are extracted and used in the segmentation process: the normal direction ($\vec{n}_i$), the centroid ($\mathcal{M}_i$), the surface area of planes *($A_p = l \times w$, based on the extreme edges*), and the distance from the origin (). Whereas the centroid, $\mathcal{M}_i$ of any particular plane ($\mathbb{\Pi}_i$) represents the center point of a plane, i.e. $\mathcal{M}_i=\{x_i, y_i, z_i\}$. Whereas the global properties (attributes) of the structural elements are defined in accordance with the general

standards and the domain knowledge about the structural systems of buildings with the RC framework.

4.3. Segmentation of Floor Slabs

As defined in (Ram, 2014), reinforced concrete (RC) slabs are structural member panels, horizontal *or nearly so*, bounded between and supported by beams, columns, walls, or the ground. In a typical multi-story building, floor slabs are located between each level of the building forming a series of platforms or levels. The floor slabs of the topmost level of the building is called the roof slab, while the floor slab of the bottommost level in level to the ground surface is referred in this study as a ground slab.

Since this study deals with the multi-storey building, therefore the segmentation process involves separating different levels of floor slabs that involves the floor slab and the suspended slabs. The latter are the slabs which are not directly supported by the ground but is instead suspended from the structural elements of the building, such as beams and columns. This creates a space between to form storey. This study uses a bottom-up approach in the segmentation process, whereby it starts with identifying the planar patches representing the surfaces of the ceilings and floors, which are the composing surfaces of the suspended slabs, followed by segmenting the suspended slabs. From a typical suspended floor slab, the ceiling and floor surfaces are distinguished by their positions as illustrated in the Figure 4.1(a). The ceiling surface is the underside of the floor slab, which is exposed during the scanning from the room below and depending on the design it can be *uncovered* or *covered* with another suspended layer usually consisting of plasterboard or ceiling tiles. For this study, we deal with the uncovered ceilings. The floor surface, on the other hand, is the upper surface of the floor slab, which is visible from the room above.

The available horizontal pairs of planes $\mathbb{\Pi}_{h(i,j)}$ are used in this case to detect the ceilings and floors. For this study, we express the slabs according to their floor levels starting from the *first floor* (ground level) and the pairs of planes are denoted in the following manner:

- $\mathbb{\Pi}_{h(0,j)}$, for planes representing *floors* (upper skin of the slab), where $j = \{1, 2,..., n\}$ corresponding to the floor levels;
- $\mathbb{\Pi}_{h(i,0)}$, for planes representing *ceilings* (bottom skin of the slab), where $i = \{1, 2,..., n\}$ at a corresponding floor level.

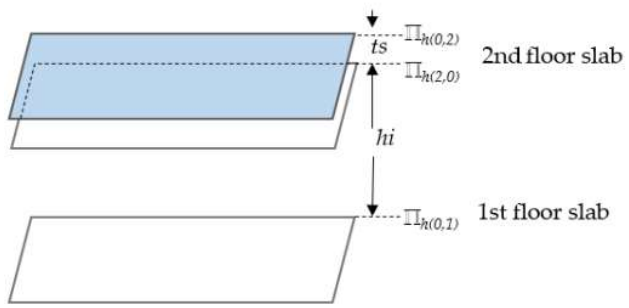
To detect the floor slabs, the horizontal planes are first ranked according to their vertical positions with reference to the z-component (z_i) of their centroid points, $\mathcal{M}_i$. This process creates a z-order histogram that comprises of planes describing their elevation level with the lowest plane assigned to the *first floor*, ($\mathbb{\Pi}_{h(0,1)}$) as illustrated in Figure 10a. The subsequent floors are determined in ascending order as shown in Figure 4.10(b).

The next step is to detect the *second-floor slab,* which involves identifying the associated planes representing the *ceiling* ($\Pi_{h(1,0)}$), and the corresponding *floor surface* ($\Pi_{h(0,2)}$). We prescribe criteria to determine the candidate planes for the ceiling and floor. We first set the orthogonal distance, h, between the *first-floor plane* ($\Pi_{h(0,1)}$), and the succeeding parallel (*or nearly parallel*) plane in the z-order. The distance, h, stipulates the allowable minimum *storey height*, where $h \geq 2.1$ m for a standard residential building, and the succeeding parallel plane is labeled as the candidate ceiling plane ($\Pi_{h(1,0)}$).

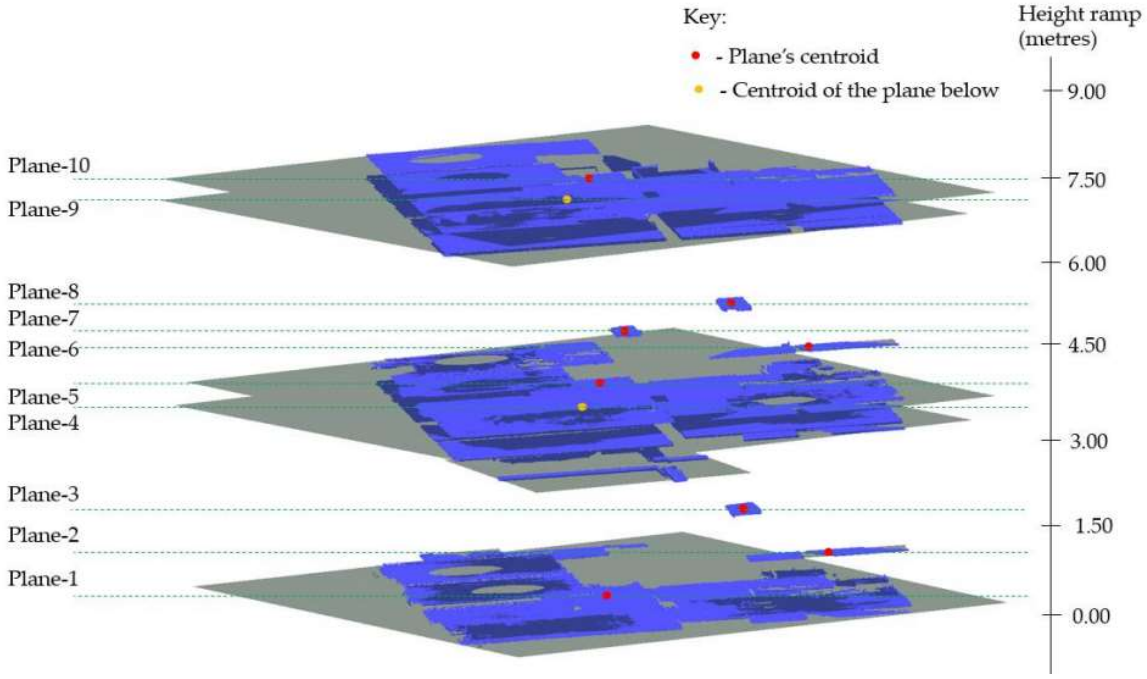
If the candidate plane ($\Pi_{h(1,0)}$) conforms to the h-threshold, we further analyze the plane in combination with its succeeding plane defined as the candidate plane for the *floor* ($\Pi_{h(0,2)}$). The two planes ($\Pi_{h(1,0)}$ & $\Pi_{h(0,2)}$) are set to be within the predetermined distance (t_s) stipulating the slab thickness (particularly for the *two-way slabs*), where $t_s < 50$ cm.

In conjunction with the t_s-threshold, the difference in surface areas (A_p) between the two planes, $\Pi_{h(2,0)}$ & $\Pi_{h(0,2)}$ is set to be less than 15% (*in our experiment*). The area difference accounts for dissimilarity in surface exposure between the ceilings and floors, whereby floor beams conceal the ceiling surfaces as opposed to walls on the floors. Once all the conditions are met, the candidate planes $\Pi_{h(1,0)}$ and $\Pi_{h(0,2)}$ are eventually designated for the *ceiling* and *floor,* respectively, for the *second-floor slab* and the offset distance h_i is signed as the first story.

The operation continues sequentially to detect the remaining floor slabs and their corresponding ceilings and floors. This approach helps to automatically obtain the respective story heights, h_i, that we can use to segment the vertical components enclosed between the floors and ceilings, where $i = \{1, 2, \ldots, n\}$ for each i indicates a particular story level.



(a)



(b)

Figure 4. 1 Vertical alignment of horizontal planes defining the respective floors and ceilings: (a) Planes representing the 1st and 2nd-floor slabs; (b) Horizontal patches including the detected occluding objects positioned along the z-axis.

4.4. Segmentation of Floor Beams

Concrete beams are horizontal or sloped composite structural members that transfer transverse loads from slabs to columns and/or load-bearing walls below (Nilson et al., 2016). The orientation and form of a beam will depend on the span of the opening it is supporting, the load it needs to bear, and the available headroom, that together play a role in designing the form or profile of the section and its overall dimensions (i.e. width and depth of the beam). The common beams forms include rectangular, circular or I-shaped. This study deals with the rectangular floor RC beams composed of three exposed planar faces, which are: *the soffit of beams*, which is the horizontal underside or ceiling of the beam, and the two parallel *vertical sides* as shown in Figure 4.2.

Segmentation of beams from the 3D point cloud is a complex task that requires a careful processing of the data. The method used in this study develops an approach for detecting surface patches representing floor beams amidst of high levels of noise and artifacts. Two main steps are proposed to segment the floors beams: the soffits of beams, and the vertical sides of beams.

First, the horizontal patches representing the soffits of beams are detected using a ray-casting technique where the beams of virtual rays are casted from the top with reference to the floor slab that the beam in consideration is attached to. The detected planes for the slab's ceilings ($\mathbb{\Pi}_{h(i,0)}$) are

used in this case to locate the candidate surface patches for the beam's soffits, where the rays casted from the virtual source located just above the ceiling's plane. Then, the rays are traced across their path across the ceiling's plane as the immediate first layer, and the recording is done whenever the rays hit the surface patches sitting below the ceiling's plane. For each of the recorded hit, the target surface patch is further analyses if it corresponds to the beam's soffits. Then, the detected surface patches for the soffits are used to locate the corresponding vertical sides of beams using the grouped pairs of vertical planes ($\mathbb{\Pi}_{v(i,j)}$). The next subsection discusses in detail about the segmentation process.

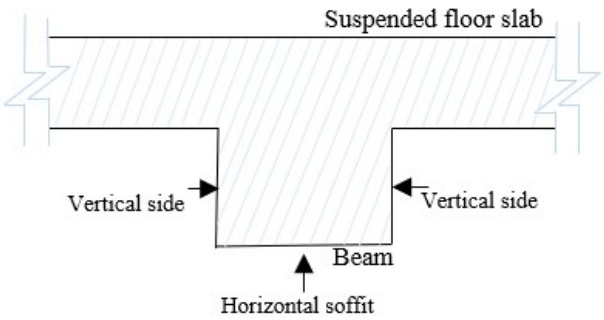


Figure 4. 2 Cross-section of the concrete beam under the floor slab.

4.4.1. Assortment of the candidate planar patches for floor beams

This part involves of assortment and preliminary screening of all points that approximately match the local position of floor beams. All the points and planar patches that still remains in the sets $\acute{R}$ and R, are collected. Where at this stage, the sets $\acute{R}$ and R (*refer to section 3.3.3*) are left with the points (and patches) that are not assigned to any of the paired planes (i.e. $\mathbb{\Pi}_h$ and $\mathbb{\Pi}_v$). All the collected points are grouped in one cluster ω, such that $\omega = \{p_1, p_2, \ldots, p_n\}$ where $p_1, p_2, \ldots, p_n$ are the points in the cluster ω.

The points from the cluster ω, are further classified into groups of k number of smaller clusters with respect to which floor level they belong, such that ω, $\omega_i = \{\omega_1, \omega_2, \ldots, \omega_k\}$ where ω_i comprises of sub-clusters obtained depending on the number of storeys (floor levels) in the building. This is done by determining the z-value of each point in the cluster ω, and find the exact floor level they belong with respect to the segmented floor slabs (floor and ceiling surfaces). Furthermore, the points contained in each of the sub-clusters (ω_i), are filtered to be remained with points that are only 1.00 metres below its corresponding ceiling. Again, the z-values of each point in the sub-clusters ω_i are used to compare with the vertical position of the respective planes for ceilings ($\mathbb{\Pi}_{h(i,0)}$). Such that, for a given ceiling's plane with a general equation, aX + bY + c(Z) + d = 0, and a point, $p_i = \{x_i, y_i, z_i\}$ drawn from a sub-cluster ω_i, where $Z - z_i \leq 1.0$ metres. A restraint of 1.00metres below the ceiling level is placed to reduce the searching process into only the points in proximity to the floor slab where the beam is expected to exist.

4.4.2. Ray casting to segment soffits of floor beams

After the preliminary screening of the point cloud to obtain the candidate points related to the soffits of beams presented in their particular sub-clusters (ω_i) arranged in accordance to the floor levels, the next step is to determine the true points representing the floor beams.

The points contained from each of the sub-cluster (ω_i) are taken and the planes Π, are fitted to these points in their respective sub-clusters in a predetermined constraint. For each plane (Π) to be fitted, its normal vector ($\vec{n}$) should not deviate by more than 10° from the normal vector ($\vec{N}$) of a ceiling's plane that is on the same floor level as the sub-cluster, i.e. $arccos(\vec{n} \,.\, \vec{N}) \leq 10°$. A residual ($r$) as shown in the Figure 3.3 is designed to respond to the estimated noise errors of the scanner. This process aims to tentatively assign each point contained in the sub-clusters to a particular provisional plane in order to allow the proceeding operations that involves rasterizing the point cloud and ray casting.

In order to apply ray casting, the input dataset is rasterized to create a visibility grid-cells during ray casting. Ray casting is a technique used in computer vision and 3D graphics to create viewpoints of objects present in the scene. For point clouds, ray casting can be used to determine which point are visible from a particular viewpoint. In this case, the ray casting approach is adopted to detect 3D points related to floor beams using the ceiling as the viewpoint. The input dataset in the ray casting process is comprised of the segmented ceiling planes ($\mathbb{\Pi}_{h(i,0)}$) and the sub-clusters (ω_i) including its provisional fitted planes, however the operation is conducted separately for each sub-cluster and the corresponding ceiling plane herein referred as a *batch*.

Hence given a batch composing of a particular sub-cluster, ω_i containing n number of fitted-planes $\sum_{i=i}^{n} \Pi$, with its constituents points, and the associated ceiling plane, $\mathbb{\Pi}_{h(i,0)}$, rasterization is done inspired by an approach used in (Ochmann et al., 2019; Xiong et al., 2013) to create single-layered voxelization referred as *occupancy bitmaps*. In order to process computations of ray casting efficiently, the octree-based voxels are used for storing the bitmaps. A resolution of 10 x 10cm bitmaps is selected in the study which provides a desired level of details and efficient computational resources. This operation returns m number of occupancy bitmaps, $S_i = \{S_1, S_2, \ldots, S_m\}$, where each bitmap is given by its center c_j, referring to the local coordinate system.

The bitmaps are labeled as either *occupied*, $S_¢$, if it has at least one point, or *empty*, S_e if it has no point inside. Next, a series of m rays, r_i, for $i = \{1, 2, \ldots, m\}$, originating from the camera's position $\mathbf{p}_r$, which is positioned above the ceiling's plane, are cast onto the occupancy bitmaps and the intersection of rays are computed thereafter. Intersections are traced from the position of the camera as the starting point $\mathbf{p}_r$ and direction vector of rays are determined, $\mathbf{u}_r$. The direction vector is typically a normalized vector that points in the direction of the ray, in our case is aimed down the negative z-axis. If the ray does not hit any bitmap on the plane, we proceed to observe if any intersection (*hit*) is made on the voxels below the ceiling's plane in the direction of the ray vector. Prior to determining the exact position where the bitmap and the ray intersect, we first need to

calculate the distance of the camera's position and the intersection point $\mathbf{p_o}$, on the plane with bitmap. Figure 4.3 illustrates the vector direction $\mathbf{u}_r$, of the ray, from the camera's origin point $\mathbf{p}_r$, to the point $\mathbf{p_o}$, on the plane.

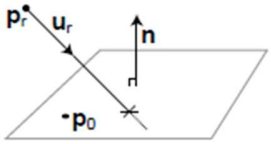


Figure 4. 3 Direction vector for ray onto the plane

The plane where the intersection point $\mathbf{p_o}$ is located can be defined by the point $\mathbf{p}_o$ and a unit surface normal vector **n** (i.e. a unit vector perpendicular to the plane's) i.e. the plane: ($\mathbf{p}_o$, **n**). Together these give the position of the plane in the space and its orientation. In order to find out where the ray ($\mathbf{p}_r$, $\mathbf{u}_r$) intersect the plane, the distance of the point **x** is computed from the plane ($\mathbf{p}_o$, **n**). To begin with, the vector from $\mathbf{p}_o$ to the point **x** is constructed, i.e., $\mathbf{x} - \mathbf{p}_o$ as shown in Figure 4.4.

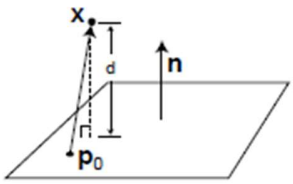


Figure 4. 4 Distance between the intersection point and ray source

The projection of the vector onto **n** gives the perpendicular distance (d) of **x** from the plane. The distance, d is calculated in Equation (4.1). In the setup, if $d > 0$, then x is above the plane by the distance d, if $d < 0$ then x is below the plane by the distance $|d|$, and if $d = 0$ then **x** is exactly on the plane ($\mathbf{p}_o$, **n**).

$$d = \mathbf{n} \,.\, (\mathrm{x} - \mathbf{po}) \tag{4.1}$$

Given a standard parametric equation of a ray (L. Linsen et al., 2007), any distance along the ray measured by parameter t whereby all the points **y** lying on the ray has to satisfy the Equation (4.2). Now, the parametric equation, can now be combined with the distance (d) to find the ray plane intersection calculated in Equation (17).

$$\mathbf{y}(t) = \mathbf{p}_r + t\mathbf{u}_r,\ t \geq 0 \tag{4.2}$$

$$0 = \mathrm{n} \,.\, [(\mathbf{p}_r + t\mathbf{u}_r) - \mathbf{p}_o] \tag{4.3}$$

$$t(\mathrm{n} \,.\, \mathbf{u}_r) = -\mathrm{n} \,.\, (\mathbf{p}_r - \mathbf{p}_o) \tag{4.4}$$

Equation (4.4) gives the value of t for which the ray intersects the plane. And the value of all intersection points **y**, are computed by inserting the value of t in the Equation (4.2).

The process is repeated for all intersections made happening below the ceilings plane ($\mathbb{\Pi}_{h(i,0)}$) and identify all the respective bitmaps where the intersections occurred. The bitmaps are used to extract the points contained in them, and from each bitmap (S_i), N number of points, p (i.e. $\sum_{i=1}^{N} p_i$), are drawn in their respective provisional planes (Π). For the neighborhood of N points, the best-fit planes are refitted, this time the TLS approach (Gene & Charles, 1980) is applied to build more accurate surface planes. On implementing the TLS, since the expected width of the floor beam is larger than bitmap's i.e. 10cm width, each plane fitted on one bitmap (S_i) is expanded and merged with plane of the abutting bitmap (S_{i+1}) on the condition that they are aligned along the direction of the ceiling's plane ($\mathbb{\Pi}_{h(i,0)}$) right above them as illustrated in the Figure 4.5.

Points aligned in the best-fit planes are identified and assigned to represent soffits of beams in their respective patches (clusters). As the result, a new set of k horizontal patches are segmented, $\mathcal{P}_{h(i)} = \{\mathcal{P}_{h(1)}, \mathcal{P}_{h(2)}, \ldots, \mathcal{P}_{h(n)}\}$ and used to identify points representing the vertical sides of floor beams as discussed in the next part.

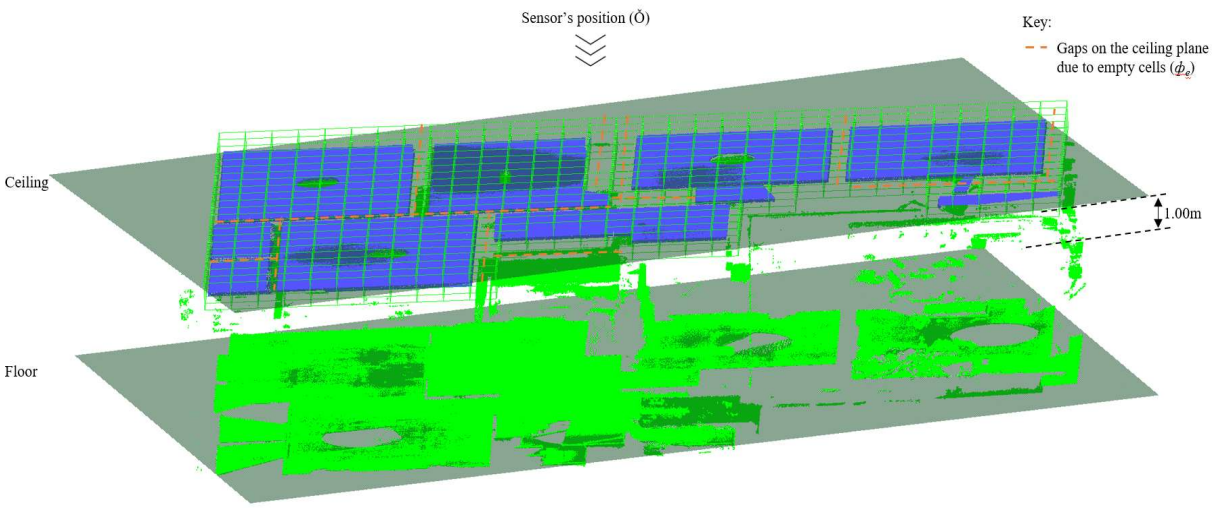


Figure 4. 5 Alignment and position of the overhead ceiling and points below the ceiling plane for detecting candidate points for the corresponding beam below.

4.4.3. Segmentation of vertical sides of floor beams

Following that, using the planes for beam soffits ($\mathcal{P}_{h(n)}$) and the associated ceiling plane, $\mathbb{\Pi}_{h(i,0)}$, we find the local positions of the neighboring pairs of vertical planes, $\mathcal{P}_{V(i)}$, which correspond to the vertical sides of beams. A set of vertical *paired planes* ($\mathbb{\Pi}_v$) is used to determine the planes $\mathcal{P}_{V(i)}$ if they qualify for the following conditions:

- The normals, $\vec{n}_v$, of a particular pair of vertical planes ($\mathbb{\Pi}_v$), should be perpendicular (*or nearly so*) to the normal direction, $\vec{n}_h$, of a soffit's plane, $\mathcal{P}_{h(i)}$;
- The orthogonal distance between the pair of vertical planes (i.e., $\mathbb{\Pi}_{V(i)} \leftrightarrow \mathbb{\Pi}_{V(i+1)}$) is approximately equal to the breadth of the associated soffit's plane, $\mathcal{P}_{h(i)}$;

- The height of vertical planes (Π_v) should correspond to the distance between the ceiling's plane ($\Pi_{h(i,0)}$) and the associated plane for the beam's soffit ($\mathcal{P}_{h(i)}$).

The pair set of vertical planes conforming to the above conditions are eliminated from the list of Π_v and designated to represent the vertical sides of the floor beams. Henceforth, for each of the revealed beams, we unveil a set of three planes ($\mathcal{P}_{h(i)}$, $\mathcal{P}_{v(i)}$, & $\mathcal{P}_{v(i+1)}$) representing the typical surfaces of beams under the floor ceiling.

4.5. Segmentation of Walls and Columns

The vertical *pair-planes* (Π_v) are primarily used to identify the points representing the surfaces of the walls and columns. The study is focused on the load bearing vertical elements i.e. RC columns and solid walls, which construes a structural path starting from the floor slabs or beams and ultimately anchored to the floor. For the purpose of this study, columns shall also include the attached and isolated piers. A tolerance, t_w, (10 cm $\leq t_w \geq$ 50 cm) is defined for wall thickness, where the value of t_w = 10 cm is applied to avoid selecting non-structural walls such as drywalls.

The segmentation process is initialized with clustering the remaining vertical planes (Π_v) into several sub-clusters referred herein as *vertical sub-spaces* where each plane is grouped into its respective floor (storey) level based on the series of floor slabs detected earlier as illustrated in the Figure 4.6. From each vertical plane $\Pi_{v(i)}$, containing the n number of points p_i (i.e. $\sum_{i=1}^{n} p_i$), the highest and lowest points along the z-axis are identified and use them to find which floor level that particular plane ($\Pi_{v(i)}$) is located with reference of the identified planes for ceilings $\Pi_{h(0,j)}$, and floors $\Pi_{h(i,0)}$. The created vertical sub-spaces helps to establish the storey heights h_i (i.e. $h_i = \{h_1, h_2, \ldots, h_n\}$ which describes the height between the floor and the nearest subsequent upper ceiling. For instance, given two successive floor slabs (i.e. Floor slab$_1$ as the lower slab and Floor slab$_2$ as the upper slab), where the for floor slab$_1$ is composed of the floor plane $\Pi_{h(0,j)}$, with a centroid point $\mathcal{M}_1 = x_1, y_1, z_1$, and the floor slab$_2$ is composed with a floor ceiling plane $\Pi_{h(i,0)}$, with a centroid point $\mathcal{M}_2 = x_2, y_2, z_2$, the storey height ($h_i$) between the floor slab$_1$ and floor slab$_2$ can be computed as shown in Equation (4.5).

$$h_i = z_2 - z_1 \tag{4.5}$$

Where the z_1, z_2 are the z-values of centroid points $\mathcal{M}_1$ and $\mathcal{M}_2$.

Next, the established storey heights (h_i) from each of the vertical sub-space are used to set the maximum requirements h_{max}, for the heights of the candidate wall planes (Π_w) housed in the respective sub-space. This implies that the candidate planes for walls and columns should not exceed the space between two successive floor slabs. Whereas, for each storey the minimum height requirement h_{min}, is refers to the height between the soffits of the floor beam and the immediate floor surface below it. Hence, to determine the h_{min} the storey height (h_i) is subtracted from the depth of the corresponding floor beam (h_d), as formulated in Equation (4.6). Therefore, for each

vertical sub-space, the set of vertical pairs of planes are identified whose heights lie within the height thresholds (h_{max} - h_{min}).

$$h_{min} = h_i - h_d \tag{4.6}$$

This approach uses a *pair-wise* plane search, which is suitable for detecting internal walls that comprise a pair of planes that locally lie on the same vertical sub-space. Henceforth, it can be ineffective in the classification of the external walls where one skin is in the interior and another skin is on the exterior of the building (façades), where the latter inherently do not fall within the floor partitions (vertical sub-spaces) as shown in Figure 4.6. To segment the planar surfaces representing the external walls, first the identification of the planes for the façades (exterior skin) $\mathbb{\Pi}_{w(f)}$, is done and their corresponding interior planes, $\mathbb{\Pi}_{w(g)}$, separately, and pair them afterward using the thickness threshold, t_w.

In principle, a common multi-story building has facades comprised of extended planar faces with disconnected surface patches due to large openings, recesses, protrusions (Hamid-Lakzaeian, 2020), and occlusions from the external features. From the set R, we identify the remaining unpaired vertical patches and find the combination of coincidental planar patches, $\bar{r}_i = \{\bar{r}_1, \bar{r}_2, \ldots, \bar{r}_n\}$ and fit the planes accordingly. To identify the points representing the façades, the combination of $\bar{r}_n$ patches is located beyond or along the edge points of the ceiling ($\mathbb{\Pi}_{h(i,0)}$) and floor planes ($\mathbb{\Pi}_{h(0,j)}$) in their x- and y-components. In doing so, obtain the coplanar patches can be obtained to represent the exterior surfaces (façades) of the external walls across the perimeter of the building. Using the set $\bar{r}_i$, we identify the planes that are perpendicular (or nearly so) to the planes for the façades ($\mathbb{\Pi}_{w(f)}$) within the thickness threshold (t_w). The identified planes are then checked against the requirements for the wall height ($h_{max} - h_{min}$) and assigned as planes for the interior skin of the external wall ($\mathbb{\Pi}_{w(g)}$).

The walls are formed by two parallel planes (skins), while the columns are formed by two pairs of planes in the opposing direction. The length-to-thickness ratio (4:1) described in (AIJ, 2002; Barker et al., 1976) is used to distinguish between the vertical planes forming the walls and columns. The lengths of planes in the two pair sets not exceeding four times (i.e., 4×) the perpendicular distance between them (thickness) are classified as planes for columns. Those having lengths over 4× the thickness are classified as planes for walls.

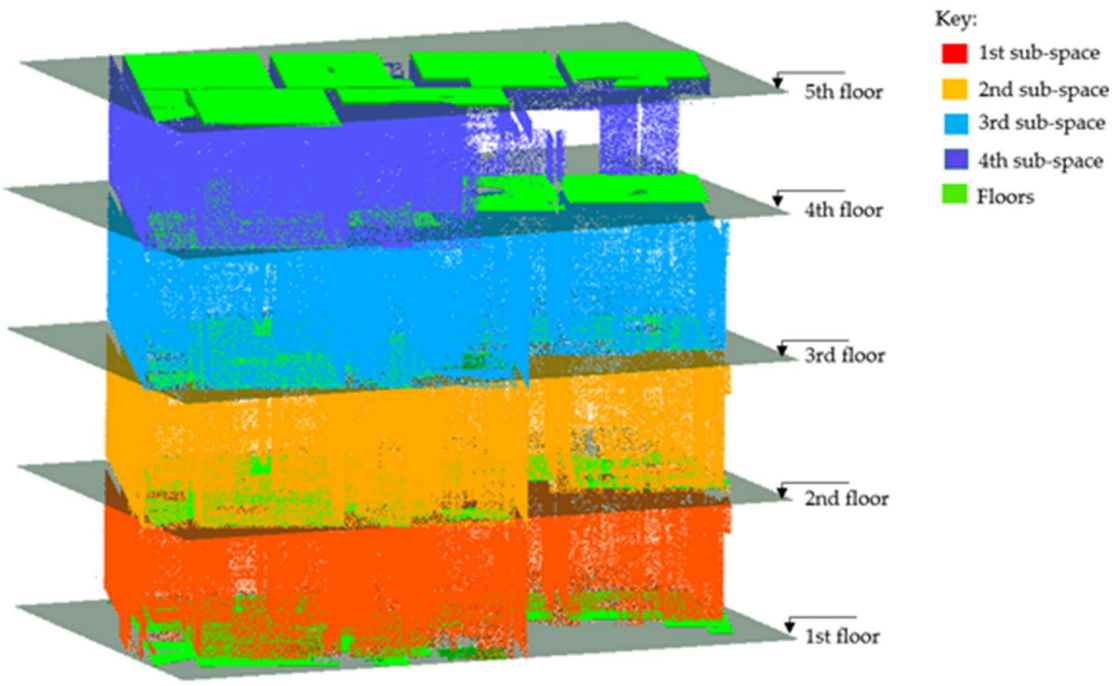


Figure 4. 6 Vertical subspaces by floor levels.

References

AIJ. (2002). *Standard for Structural Design of Reinforced Concrete Boxed-Shaped Wall Structures*.

Barker, H. A., Fairbairns, R. F., Lamont, R. C., Maher, B., Neild, A. P., & Willard, C. G. (1976). *Standard Method of Measurement of Building Works for East Africa* (First Edit). Quantity Surveyors, AAK.

Gene, G., & Charles, van L. (1980). An Analysis of the Total Least Squares Problem. *SIAM Journal on Numerical Analysis*, *17*(6). https://doi.org/10.1137/0717073

Hamid-Lakzaeian, F. (2020). Point cloud segmentation and classification of structural elements in multi-planar masonry building facades. *Automation in Construction*, *118*(October 2018), 103232. https://doi.org/10.1016/j.autcon.2020.103232

Linsen, L., Müller, K., & Rosenthal, P. (2007). Splat-based ray tracing of point clouds. *15th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision 2007, WSCG'2007 - In Co-Operation with EUROGRAPHICS, Full Papers ProceedingsWSCG Proceedings*, *January 2007*, 51–58.

Nilson, A. H., Darwin, D., & Dolan, C. W. (2016). *Design Concrete Sructures*.

Ochmann, S., Vock, R., & Klein, R. (2019). Automatic reconstruction of fully volumetric 3D building models from oriented point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing*, *151*, 251–262. https://doi.org/10.1016/j.isprsjprs.2019.03.017

Ram, G. . (2014). *Principle of Structural Design* (Second Edi). CRC Press.

Xiong, X., Adan, A., Akinci, B., & Huber, D. (2013). Automatic creation of semantically rich 3D building models from laser scanner data. *Automation in Construction*, *31*, 325–337. https://doi.org/10.1016/j.autcon.2012.10.006

Chapter 5. Experiments and Results

5.1. Introduction

This chapter involves the introduction of the experiment design, the experimental dataset, instruments, and software applied, evaluation metrics, experimental procedures, and description of the results.

5.2. Experimental design

For testing the performance of the methods introduced in sections 3-4, the experiments are performed and assessed using three major parameters: quantitative evaluation, qualitative analysis, and comparative evaluation.

The quantitative evaluation of the experiments done on the segmentation of point clouds into structural components provides a numerical analysis of the performance of the proposed segmentation methods. The evaluation is based on a set of quantitative metrics such as precision, recall, F1-score, and the intersection over Union (IoU), which are discussed in detail in section 6. The evaluation is conducted at different stages of the segmentation pipelines from the detection of the minimal planar patches to the final segmentation of building components. The evaluation is also performed to check the effectiveness of different segmentation algorithms depending on the type of structural elements in the study. This is necessary because certain algorithms have been more effective in identifying particular elements than others.

The qualitative analysis is done to provide a subjective analysis of the performance of segmentation approaches applied in the study. The evaluation is based on the visual inspection of the segmentation results and analytical observation based on several factors that can affect the accuracy of the results for instance nature of the occlusions present on the site, for instance, the type and size of fittings and furniture, disparities between elements of the similar type, finishing applied specially to walls and ceilings, the existence of moving clutters, and the like. Furthermore, a correlation is made between the qualitative and quantitative evaluations. Overall, the qualitative evaluation provides additional insights into the performance of different methods proposed in the study and help to understand the strengths and limitation of each method.

The comparative evaluation of the experiments is performed between the proposed segmentation methods and the standard methods of segmenting the point clouds into building components. Our proposed segmentation methods are directly compared with the popular techniques in the industry that have deployed clustering and deep learning segmentation techniques. A similar raw dataset is applied in this instance for all the segmentation methods selected in the comparison, and the results are compared in their accuracies and efficiencies. Comparative evaluation can help to validate the proposed segmentation method's effectiveness. Comparing the results of the proposed method with other state-of-the-art methods enhances to determine whether the proposed method is better, worse, or similar in terms of accuracy, speed, and other relevant metrics.

In general, the evaluations are used as a guide for further improvements in our algorithms, leading to more accurate and efficient segmentation of structural elements from the point cloud.

5.3. Experiment dataset and equipment

In order to test the efficiency of the proposed methods a large 3D point cloud with a high degree of noise and occlusions with unmodified scenes. The points cloud is acquired from a 46-year-old five-story RC-framed building as shown in Figure 5.1. The building is comprised of 30 fully furnished apartment units with similar floor layouts (refer *Appendix I*), located in Onojo, Fukuoka.

The point cloud was generated by combining multiple registered scans acquired at known positions for indoor and outdoor scenes. The scans were captured using a Zoller + Fröhlich (Z + F) Imager® 5016 terrestrial laser scanner (Zoller + Fröhlich, 2022). Outline activities for laser scanning the building which include the scanning planning, setting up the scanner, activation of the inbuilt software, calibrating the scanner, methods for scanning the building, and the initial processing of the scans are discussed in detail in Ntiyakunze et al. (2022). The procedures discussed in the paper provide a basic guide for conducting laser scanning in a highly cluttered environment, and they can be adapted to suit various requirements of different sites.

Figure 5. 1 Rear view of Hikari Building

The building was scanned from several scan positions (refer *Appendix II*) to capture all the necessary geometries to obtain an unmodified point cloud. For experimental purposes, the original point cloud (see Figure 5.2a), is randomly cut into two portions, where the portion$_1$ (see Figure 5.2b) is used primarily for the general experimentation of the proposed segmentation procedures and portion$_2$ is used for comparative testing and evaluation (see Figure 5.2c). The original point cloud contains a huge file size and high similarities between floor levels and room conditions, hence the portions used for the experiment still provide an accurate presentation of the scenery and provide manageable data size. The scans are indexed and registered on Autodesk ReCap Pro and generate the point cloud. The scanning results for these scenes are illustrated in Table 5.1.

Table 5. 1 The number of scan stations, the total number of points, and registration precision.

Experiment	Scan stations	No. of Points	Average resolution range	Average range error
Original Point cloud	236	374,405,029	0.1mm	0.25 mm rms

Test data (Portion$_1$)	N/A	66,997,720	0.1mm	0.25 mm rms
Comparison dataset (Portion$_2$)	N/A	7.360,289	0.1mm	0.25 mm rms

The proposed segmentation method was implemented on Open3D 0.16.0 (Qian-Yi et al., 2018), CloudCompare v2.12.3 (Kyiv, Ukraine) (64-bit), and Python 3.9.12. All the experiments were performed on a 3.70 GHz Intel(R) Core (TM) i9-10900K processor and 32 GB of RAM and an NVIDIA Geforce RTX 3070. All the baseline methods were performed on Open3D 0.16.0 and Python 3.9.12, whereas CloudCompare v2.12.3 was used for geometric fitting and visualization of results.

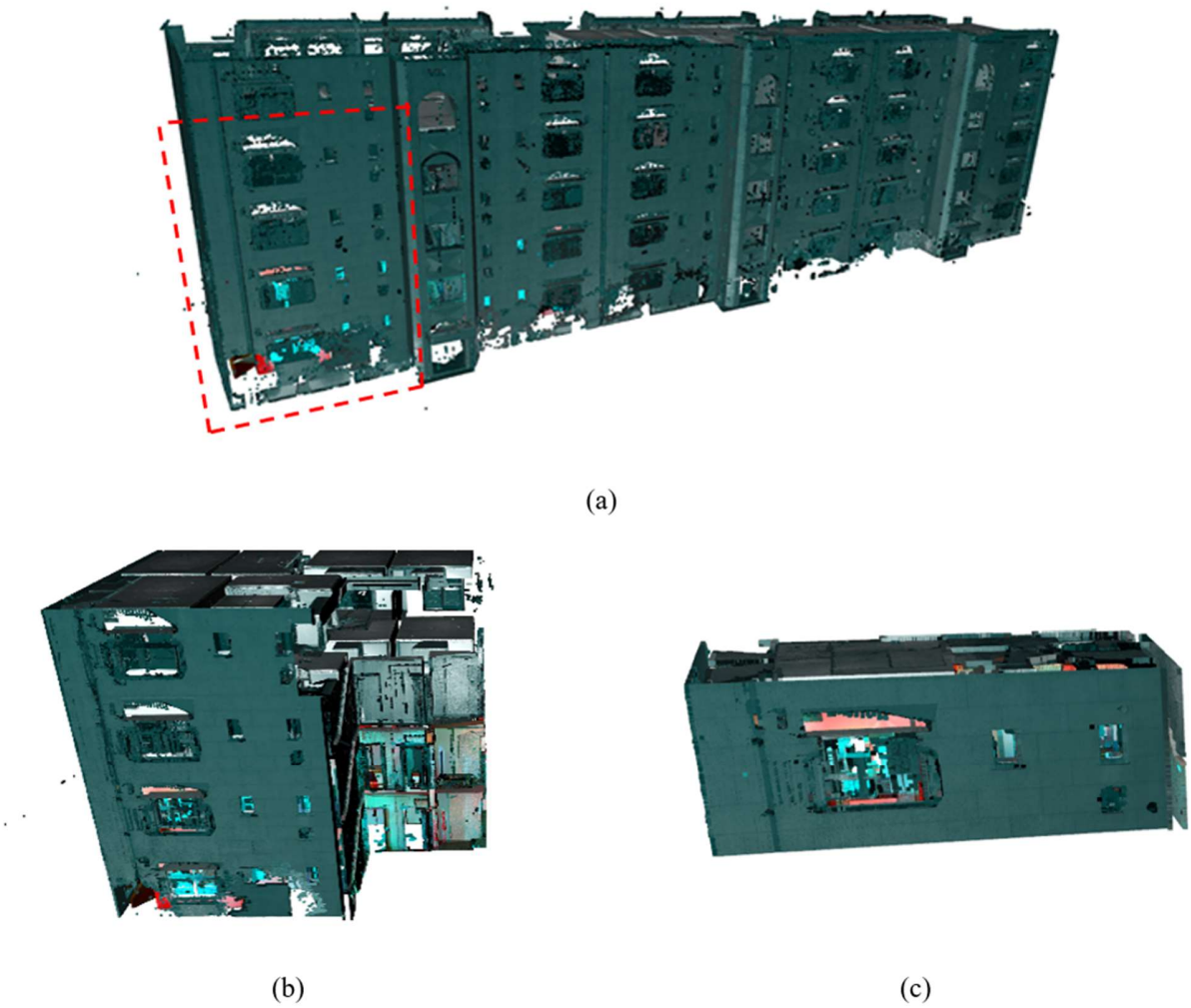


Figure 5. 2 Input point cloud for the experiments: (a) Original point cloud; (b) Portion$_1$; (c) Portion$_2$

5.4. Evaluation metrics

Setting appropriate evaluation metrics is essential for measuring the quantitative performance of the proposed segmentation methods, identifying their strengths and weaknesses, and ensuring the accuracy and reliability of the overall approach. The basic approach in selecting the evaluation metrics is to assess the level of agreement between the segmentation results and the real-world dataset (ground-truth data) which is used for referencing. The quantities used in the reference dataset are collected manually from the site with the assistance of the original point cloud data and scan images.

For performance evaluation of the proposed methods regarding segmentation and classification outcomes, we utilize the following evaluation metrics: precision (*pr*), recall (*r*), F_1-score, and Intersection-Over-Union (IoU). The precision (*pr*), recall (*r*), F_1-score, and Intersection-Over-Union (IoU) are selected as a measure for assessing the accuracy and effectiveness of our method.

The precision (*pr*) stands for the percentage of correct segmentations among all the predicted segmentations, which pertaining to the confusion matrix is the ability of the method not to label as positive a sample that is negative. On the other hand, the recall (*r*) corresponds to the percentage of correctly identified positive instances among all the actual positive instances. F_1-score combines precision and recall providing an overall measure of accuracy. The F_1-score is calculated as the harmonic mean of precision and recall, with a value of 1 indicating perfect accuracy and a value of 0 indicating poor accuracy. IoU is the measure of the overlap between the predicted segmentation and the ground truth data. The IoU metric is typically preferred over the F_1-score when there the class frequencies are severally unbalanced which is common in indoor point cloud datasets (Poux & Billen, 2019). The overall accuracy (OA) which is an aggregates measure of the overall performance of the experiment in predicting all class elements is avoided to be used as an evaluation metric in the study because of the undue bias declining towards dominant elements where in this case the performance accuracy for walls can significantly influence the overall accuracy. Hence, the F_1-score is preferred to penalize the single effect of bad classification of one element over others.

$$\text{Precision } (pr) = \frac{\text{TP}}{\text{TP+FP}} \tag{5.1}$$

$$\text{Recall } (r) = \frac{\text{TP}}{\text{TP+FN}} \tag{5.2}$$

$$F_1\text{-score} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}} \tag{5.3}$$

$$\text{IoU} = \frac{\text{TP}}{\text{FP} + \text{FN} + \text{TP}} \tag{5.4}$$

Where TP represents true positives, which refers to the number of class elements (slabs, beams, or walls) that were detected and found in the existing building; FP represents false positives, which refers to the number of detected class elements that were not found in the existing building; FN represents false negatives, which refers to the number of undetected class elements that were not detected by the proposed method. We also use true negative (TN) to evaluate our model's ability to correctly predict the negative classes. An illustration of the TP, FP, FN, and TN is provided in Figure 5.3.

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

Figure 5. 3 Confusion matrix

The selected evaluation metrics are popular measures in the segmentation and classification tasks in point clouds and images. Poux & Billen (2019) used the confusion matrix of size n x n to extract the evaluation metrics to test the segmentation approach using graph theory, where n is the number of labels. Other state of art studies also used the same evaluation strategy for the plane detection problem (Awrangjeb & Fraser, 2014; Nurunnabi et al., 2012; Vo et al., 2015; Xiong et al., 2013) also used the same metrics to test their plane detection approaches.

5.5. Preprocessing of the dataset

Initially, the input point cloud is manually cropped to remove the unwanted objects appearing separated from the site of interest including nearby buildings, roads, and trees. Then, voxel-based down-sampling is used to mainly sparse the points in highly dense regions while preserving the overall shape and local structure. This process helps to save computational resources which reduce the computation time and memory use. The input point cloud $\boldsymbol{P}$, with 66,997,720 points is uniformly sub-sampled using a 0.05mm voxel size, generating a point cloud, $\boldsymbol{p}$ with 30,129,150 points, such that, $\boldsymbol{p} \subseteq \boldsymbol{P}$.

5.6. Experimental Procedures and Results

5.6.1. Decomposition of the 3D Point Cloud using Octree-based voxelization

The input point cloud ($\boldsymbol{p}$) is spatially subdivided to produce a voxelized point cloud as presented in Figure 15. The decomposition of the point cloud was initiated at the internal node level 0 with 30,129,150 points into 8 children and recursively decomposes guided by early termination criteria. The termination is based on two criteria: the maximum depth and the minimum number of samples

in each voxel, whichever arrived earlier. The maximum depth selected was $\Psi_{max} = 6$, and the minimum number of points per voxel is $\mathcal{E}_{min} = 30$, used to subdivide the input point cloud as shown in Figure 5.4, into smaller levels for reduced computations in the proceeding processes.

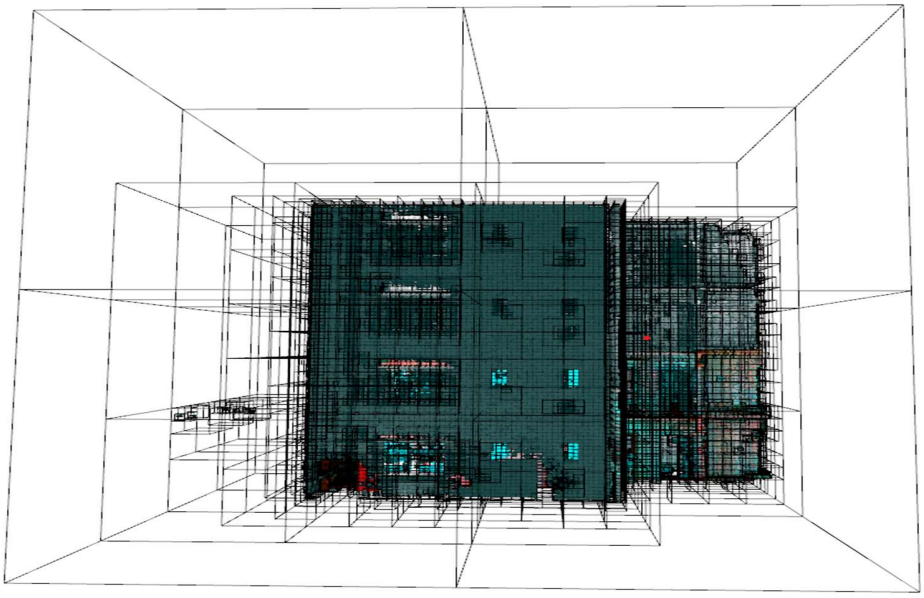


Figure 5. 4 Spatial sub-division of the input point cloud.

A small output of the resulting octree is presented in Table 5.2 which shows the number of selective internal nodes at their specific levels, the number of points at each node, and their respective grid indices (refer *Appendix III* for additional information about the octree structure created from the input point cloud). Internal nodes are created and voxel grids were fitted at all nodes from level 0 to level 5 as the termination point.

Table 5. 2 Octree structure created and the respective voxel-grids at internal node level 6

7: Internal node at depth 5 has 2 children and 1406 points ([-6.8281192 29.97262225 6.10515596])

5: Leaf node at depth 6 has 645 points with origin [-6.81233795 29.97262225 6.12093721]

7: Leaf node at depth 6 has 761 points with origin [-6.81233795 29.9884035 6.12093721]

6: Internal node at depth 5 has 6 children and 1124 points ([-6.7965567 29.97262225 6.10515596])

0: Leaf node at depth 6 has 22 points with origin [-6.7965567 29.97262225 6.10515596]

1: Leaf node at depth 6 has 4 points with origin [-6.78077545 29.97262225 6.10515596]

4: Leaf node at depth 6 has 138 points with origin [-6.7965567 29.97262225 6.12093721]

5: Leaf node at depth 6 has 7 points with origin [-6.78077545 29.97262225 6.12093721]

6: Leaf node at depth 6 has 363 points with origin [-6.7965567 29.9884035 6.12093721]

7: Leaf node at depth 6 has 590 points with origin [-6.78077545 29.9884035 6.12093721]

7: Internal node at depth 5 has 4 children and 1569 points ([-6.7649942 29.97262225 6.10515596])

4: Leaf node at depth 6 has 56 points with origin [-6.7649942 29.97262225 6.12093721]

5: Leaf node at depth 6 has 167 points with origin [-6.74921295 29.97262225 6.12093721]

6: Leaf node at depth 6 has 508 points with origin [-6.7649942 29.9884035   6.12093721]

7: Leaf node at depth 6 has 838 points with origin [-6.74921295 29.9884035   6.12093721]

5: Internal node at depth 5 has 2 children and 1463 points ([-6.8281192 30.00418475 6.10515596])

5: Leaf node at depth 6 has 716 points with origin [-6.81233795 30.00418475 6.12093721]

7: Leaf node at depth 6 has 747 points with origin [-6.81233795 30.019966   6.12093721]

7: Internal node at depth 5 has 2 children and 1655 points ([-6.8281192 30.03574725 6.10515596])

5: Leaf node at depth 6 has 822 points with origin [-6.81233795 30.03574725 6.12093721]

7: Leaf node at depth 6 has 833 points with origin [-6.81233795 30.0515285   6.12093721]

4: Internal node at depth 5 has 4 children and 2729 points ([-6.7965567 30.00418475 6.10515596])

4: Leaf node at depth 6 has 344 points with origin [-6.7965567 30.00418475 6.12093721]

5: Leaf node at depth 6 has 740 points with origin [-6.78077545 30.00418475 6.12093721]

6: Leaf node at depth 6 has 567 points with origin [-6.7965567 30.019966   6.12093721]

7: Leaf node at depth 6 has 1078 points with origin [-6.78077545 30.019966   6.12093721]

5: Internal node at depth 5 has 4 children and 731 points ([-6.7649942 30.00418475 6.10515596])

4: Leaf node at depth 6 has 220 points with origin [-6.7649942 30.00418475 6.12093721]

5: Leaf node at depth 6 has 53 points with origin [-6.74921295 30.00418475 6.12093721]

6: Leaf node at depth 6 has 455 points with origin [-6.7649942 30.019966   6.12093721]

7: Leaf node at depth 6 has 3 points with origin [-6.74921295 30.019966   6.12093721]

6: Internal node at depth 5 has 4 children and 2849 points ([-6.7965567 30.03574725 6.10515596])

4: Leaf node at depth 6 has 691 points with origin [-6.7965567 30.03574725 6.12093721]

5: Leaf node at depth 6 has 746 points with origin [-6.78077545 30.03574725 6.12093721]

6: Leaf node at depth 6 has 708 points with origin [-6.7965567 30.0515285   6.12093721]

7: Leaf node at depth 6 has 704 points with origin [-6.78077545 30.0515285   6.12093721]

7: Internal node at depth 5 has 4 children and 3290 points ([-6.7649942 30.03574725 6.10515596])

4: Leaf node at depth 6 has 811 points with origin [-6.7649942 30.03574725 6.12093721]

5: Leaf node at depth 6 has 371 points with origin [-6.74921295 30.03574725 6.12093721]

6: Leaf node at depth 6 has 707 points with origin [-6.7649942 30.0515285   6.12093721]

7: Leaf node at depth 6 has 1401 points with origin [-6.74921295 30.0515285   6.12093721]

4: Internal node at depth 4 has 3 children and 6009 points ([-6.8596817 29.94105975 6.13671846])

3: Internal node at depth 5 has 4 children and 2880 points ([-6.8281192 29.97262225 6.13671846])

1: Leaf node at depth 6 has 723 points with origin [-6.81233795 29.97262225 6.13671846]

3: Leaf node at depth 6 has 705 points with origin [-6.81233795 29.9884035   6.13671846]

5: Leaf node at depth 6 has 726 points with origin [-6.81233795 29.97262225 6.15249971]

7: Leaf node at depth 6 has 726 points with origin [-6.81233795 29.9884035   6.15249971]

5: Internal node at depth 5 has 1 children and 18 points ([-6.8281192 29.94105975 6.16828096])

7: Internal node at depth 5 has 4 children and 3111 points ([-6.8281192 29.97262225 6.16828096])

1: Leaf node at depth 6 has 840 points with origin [-6.81233795 29.97262225 6.16828096]

3: Leaf node at depth 6 has 738 points with origin [-6.81233795 29.9884035   6.16828096]

5: Leaf node at depth 6 has 809 points with origin [-6.81233795 29.97262225 6.18406221]

7: Leaf node at depth 6 has 724 points with origin [-6.81233795 29.9884035   6.18406221]

5: Internal node at depth 4 has 7 children and 17646 points ([-6.7965567 29.94105975 6.13671846])

1: Internal node at depth 5 has 3 children and 85 points ([-6.7649942 29.94105975 6.13671846])

0: Leaf node at depth 6 has 1 points with origin [-6.7649942 29.94105975 6.13671846]

4: Leaf node at depth 6 has 26 points with origin [-6.7649942 29.94105975 6.15249971]

5: Leaf node at depth 6 has 58 points with origin [-6.74921295 29.94105975 6.15249971]

2: Internal node at depth 5 has 7 children and 2385 points ([-6.7965567 29.97262225 6.13671846])

0: Leaf node at depth 6 has 341 points with origin [-6.7965567 29.97262225 6.13671846]

2: Leaf node at depth 6 has 46 points with origin [-6.7965567 29.9884035   6.13671846]

3: Leaf node at depth 6 has 780 points with origin [-6.78077545 29.9884035   6.13671846]

4: Leaf node at depth 6 has 469 points with origin [-6.7965567 29.97262225 6.15249971]

5: Leaf node at depth 6 has 161 points with origin [-6.78077545 29.97262225 6.15249971]

6: Leaf node at depth 6 has 8 points with origin [-6.7965567 29.9884035   6.15249971]

7: Leaf node at depth 6 has 580 points with origin [-6.78077545 29.9884035   6.15249971]

3: Internal node at depth 5 has 8 children and 3772 points ([-6.7649942 29.97262225 6.13671846])

0: Leaf node at depth 6 has 29 points with origin [-6.7649942 29.97262225 6.13671846]

1: Leaf node at depth 6 has 538 points with origin [-6.74921295 29.97262225 6.13671846]

2: Leaf node at depth 6 has 787 points with origin [-6.7649942 29.9884035   6.13671846]

3: Leaf node at depth 6 has 483 points with origin [-6.74921295 29.9884035   6.13671846]

4: Leaf node at depth 6 has 419 points with origin [-6.7649942 29.97262225 6.15249971]

5: Leaf node at depth 6 has 1023 points with origin [-6.74921295 29.97262225 6.15249971]

6: Leaf node at depth 6 has 405 points with origin [-6.7649942 29.9884035   6.15249971]

7: Leaf node at depth 6 has 88 points with origin [-6.74921295 29.9884035   6.15249971]

4: Internal node at depth 5 has 4 children and 1473 points ([-6.7965567 29.94105975 6.16828096])

0: Leaf node at depth 6 has 18 points with origin [-6.7965567 29.94105975 6.16828096]

1: Leaf node at depth 6 has 351 points with origin [-6.78077545 29.94105975 6.16828096]

4: Leaf node at depth 6 has 405 points with origin [-6.7965567 29.94105975 6.18406221]

5: Leaf node at depth 6 has 699 points with origin [-6.78077545 29.94105975 6.18406221]

5: Internal node at depth 5 has 4 children and 2891 points ([-6.7649942 29.94105975 6.16828096])

0: Leaf node at depth 6 has 572 points with origin [-6.7649942 29.94105975 6.16828096]

1: Leaf node at depth 6 has 744 points with origin [-6.74921295 29.94105975 6.16828096]

4: Leaf node at depth 6 has 770 points with origin [-6.7649942 29.94105975 6.18406221]

5: Leaf node at depth 6 has 805 points with origin [-6.74921295 29.94105975 6.18406221]

6: Internal node at depth 5 has 4 children and 3242 points ([-6.7965567 29.97262225 6.16828096])

0: Leaf node at depth 6 has 776 points with origin [-6.7965567 29.97262225 6.16828096]

1: Leaf node at depth 6 has 971 points with origin [-6.78077545 29.97262225 6.16828096]

4: Leaf node at depth 6 has 743 points with origin [-6.7965567 29.97262225 6.18406221]

5: Leaf node at depth 6 has 752 points with origin [-6.78077545 29.97262225 6.18406221]

7: Internal node at depth 5 has 4 children and 3798 points ([-6.7649942 29.97262225 6.16828096])

0: Leaf node at depth 6 has 1042 points with origin [-6.7649942 29.97262225 6.16828096]

1: Leaf node at depth 6 has 1154 points with origin [-6.74921295 29.97262225 6.16828096]

4: Leaf node at depth 6 has 712 points with origin [-6.7649942 29.97262225 6.18406221]

5: Leaf node at depth 6 has 890 points with origin [-6.74921295 29.97262225 6.18406221]

6: Internal node at depth 4 has 4 children and 16944 points ([-6.8596817 30.00418475 6.13671846])

1: Internal node at depth 5 has 4 children and 2809 points ([-6.8281192 30.00418475 6.13671846])

1: Leaf node at depth 6 has 718 points with origin [-6.81233795 30.00418475 6.13671846]

3: Leaf node at depth 6 has 695 points with origin [-6.81233795 30.019966   6.13671846]

5: Leaf node at depth 6 has 691 points with origin [-6.81233795 30.00418475 6.15249971]

7: Leaf node at depth 6 has 705 points with origin [-6.81233795 30.019966   6.15249971]

3: Internal node at depth 5 has 4 children and 3292 points ([-6.8281192 30.03574725 6.13671846])

1: Leaf node at depth 6 has 855 points with origin [-6.81233795 30.03574725 6.13671846]

3: Leaf node at depth 6 has 906 points with origin [-6.81233795 30.0515285   6.13671846]

5: Leaf node at depth 6 has 775 points with origin [-6.81233795 30.03574725 6.15249971]

7: Leaf node at depth 6 has 756 points with origin [-6.81233795 30.0515285   6.15249971]

5: Internal node at depth 5 has 6 children and 4713 points ([-6.8281192 30.00418475 6.16828096])

1: Leaf node at depth 6 has 650 points with origin [-6.81233795 30.00418475 6.16828096]

2: Leaf node at depth 6 has 451 points with origin [-6.8281192 30.019966   6.16828096]

3: Leaf node at depth 6 has 1322 points with origin [-6.81233795 30.019966   6.16828096]

5: Leaf node at depth 6 has 674 points with origin [-6.81233795 30.00418475 6.18406221]

6: Leaf node at depth 6 has 283 points with origin [-6.8281192 30.019966   6.18406221]

7: Leaf node at depth 6 has 1333 points with origin [-6.81233795 30.019966   6.18406221]

7: Internal node at depth 5 has 6 children and 6130 points ([-6.8281192 30.03574725 6.16828096])

0: Leaf node at depth 6 has 379 points with origin [-6.8281192 30.03574725 6.16828096]

1: Leaf node at depth 6 has 1571 points with origin [-6.81233795 30.03574725 6.16828096]

2: Leaf node at depth 6 has 427 points with origin [-6.8281192 30.0515285   6.16828096]

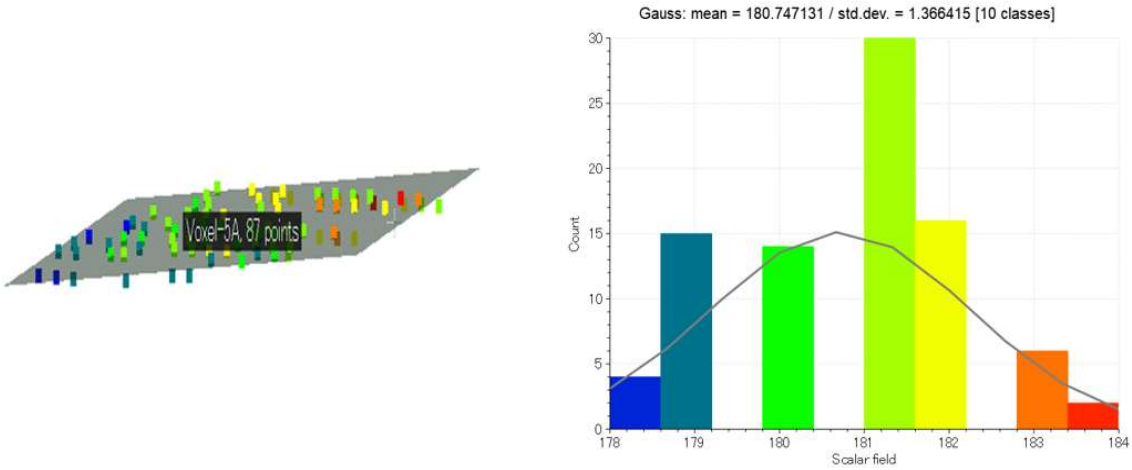
3: Leaf node at depth 6 has 1590 points with origin [-6.81233795 30.0515285   6.16828096]

5: Leaf node at depth 6 has 1143 points with origin [-6.81233795 30.03574725 6.18406221]

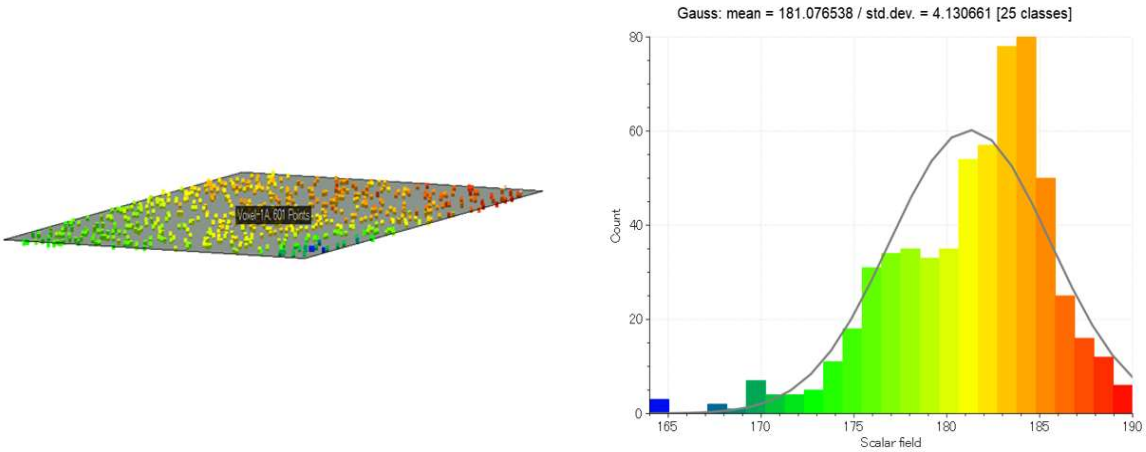
7: Leaf node at depth 6 has 1020 points with origin [-6.81233795 30.0515285   6.18406221]

5.6.2. Planar Patch Generation

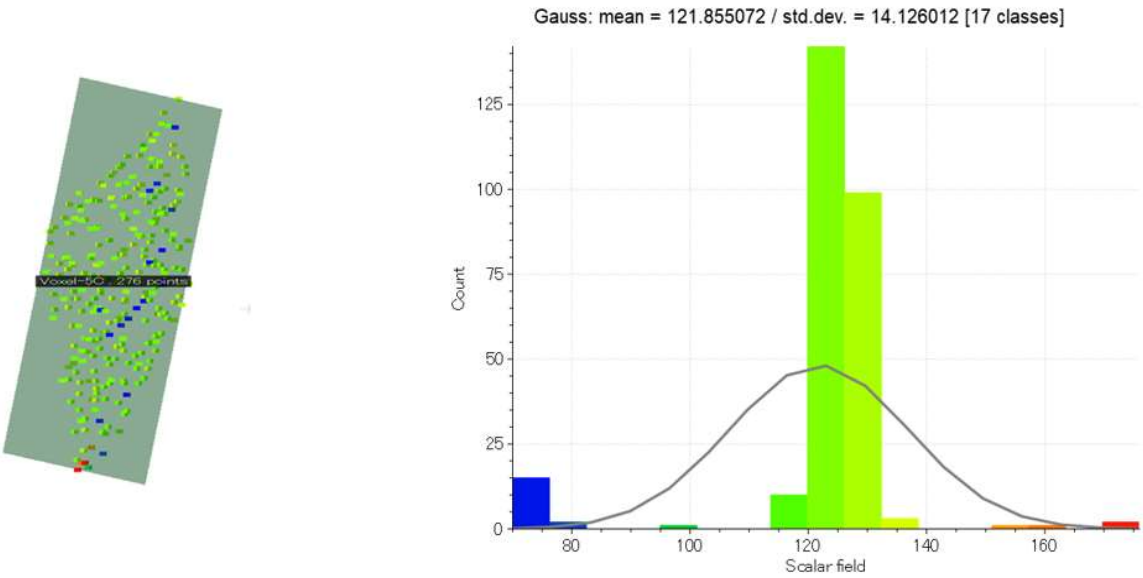
Firstly, the eigen-based features are computed using the PCA and fit a plane in the principal direction of the points, whose normal is given by the smallest eigenvalue (λ_0). The TLS approach was used to fit planes at the voxel level. For this case, only the voxels with more than 30 points were processed. Non-empty voxels and voxels with less than 30 points were excluded from this process. Some of the plane fitting results are illustrated in Figures 5.5(a) – 5.5(e), where points in their respective voxels are fitted. Points are presented in colors referring to the scalar values described in the mixed histogram and Gaussian distribution charts. The charts show how much each point in the plane deviates from the sample mean value.



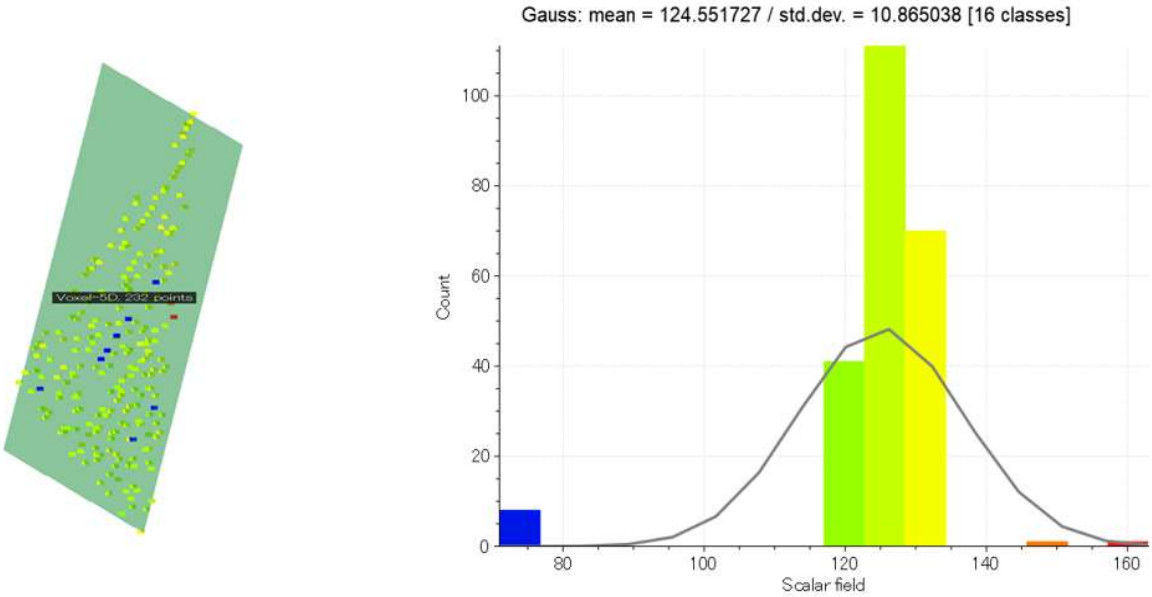
(a)


Gauss: mean = 181.076538 / std.dev. = 4.130661 [25 classes]
Count
Scalar field

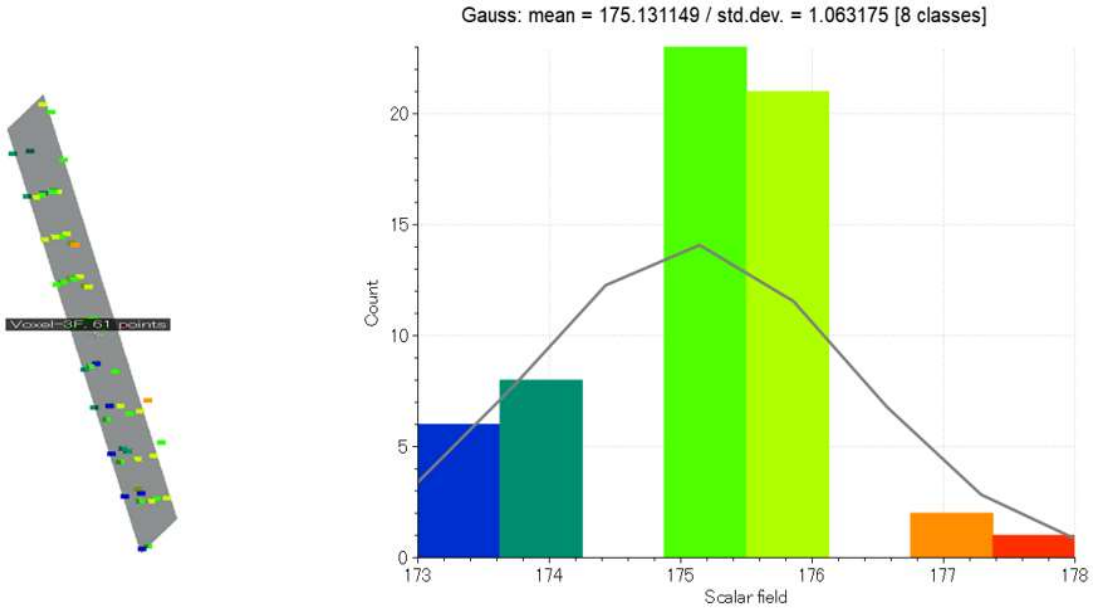
(b)


Gauss: mean = 121.855072 / std.dev. = 14.126012 [17 classes]
Count
Scalar field

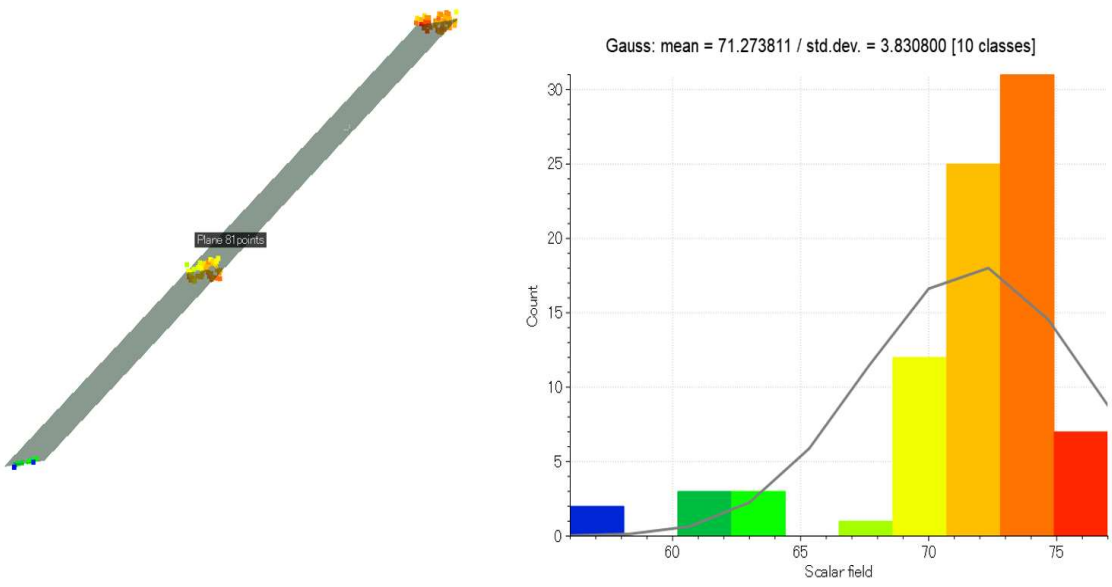
(c)


Gauss: mean = 124.551727 / std.dev. = 10.865038 [16 classes]
Voxel=5D, 232 points
Count
Scalar field
0
20
40
60
80
100
80
100
120
140
160

(d)


Gauss: mean = 175.131149 / std.dev. = 1.063175 [8 classes]
Voxel=3F, 61 points
Count
Scalar field
0
5
10
15
20
173
174
175
176
177
178

(e)



(f)

Figure 5. 5 Plane fitting at the voxel level: (a) Plane on 87 points; (b) Plane on 601 points; (c) Plane on 276 points; (d) Plane on 232 points; (e) Plane on 61 points; (f) Plane on 81 points.

5.6.3. Merging of planar patches

The planar patches from the adjoining voxels using the 26-face connectivity approach are merged provided that the difference in their normal vectors, $\vec{n}_i$ & $\vec{n}_{I+I}$ lie within a maximal angular value of 5°. The minimal planar patches detected in the previous step are recursively merged on their proximities as illustrated in Figure 5.6 (a) - (b) where about five minimal patches from the 5 connected patches are merged, summing up a plane fitted to 941 points.

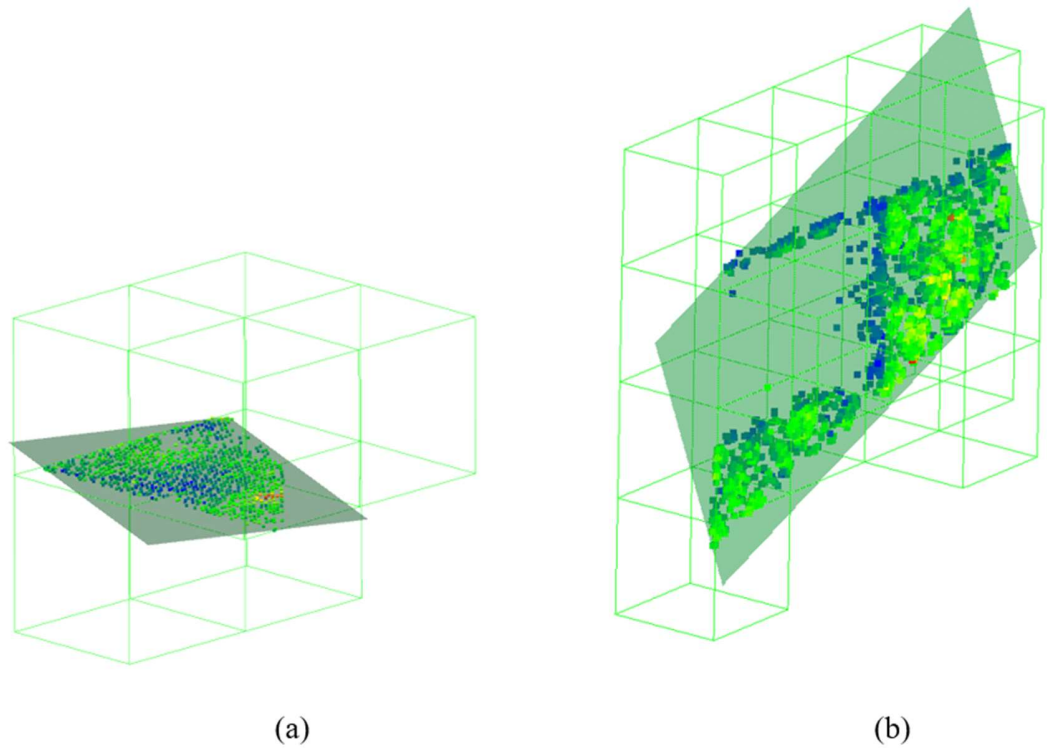


Figure 5. 6 Merging of adjacent planar patches across the connected voxels: (a) Fully connected patches; (b) Patches with sparsely distributed points.

Next, the merged patches with surface areas (A_s) ≥ 0.15m^2, are together appended in a set R which is used for determining the principal pairs of planes. Otherwise, the merged patches with surface areas (A_s) ≤ 0.15m^2 are clustered in the set $\acute{R}_i$, and labeled as undergrowth patches.

5.6.4. Determination of principal patches

The planar patches contained in set R are carefully analyzed in order to obtain the principal patches (∑). The criteria of selecting principal patches are used to filter out patches that may not present the surfaces of building elements, including differences of normal orientation between the two patches to form a pair-set as illustrated in Figure 5.7(a) and 5.7(b), where the difference the normal of two patches in each pair set is less than 5 degrees.

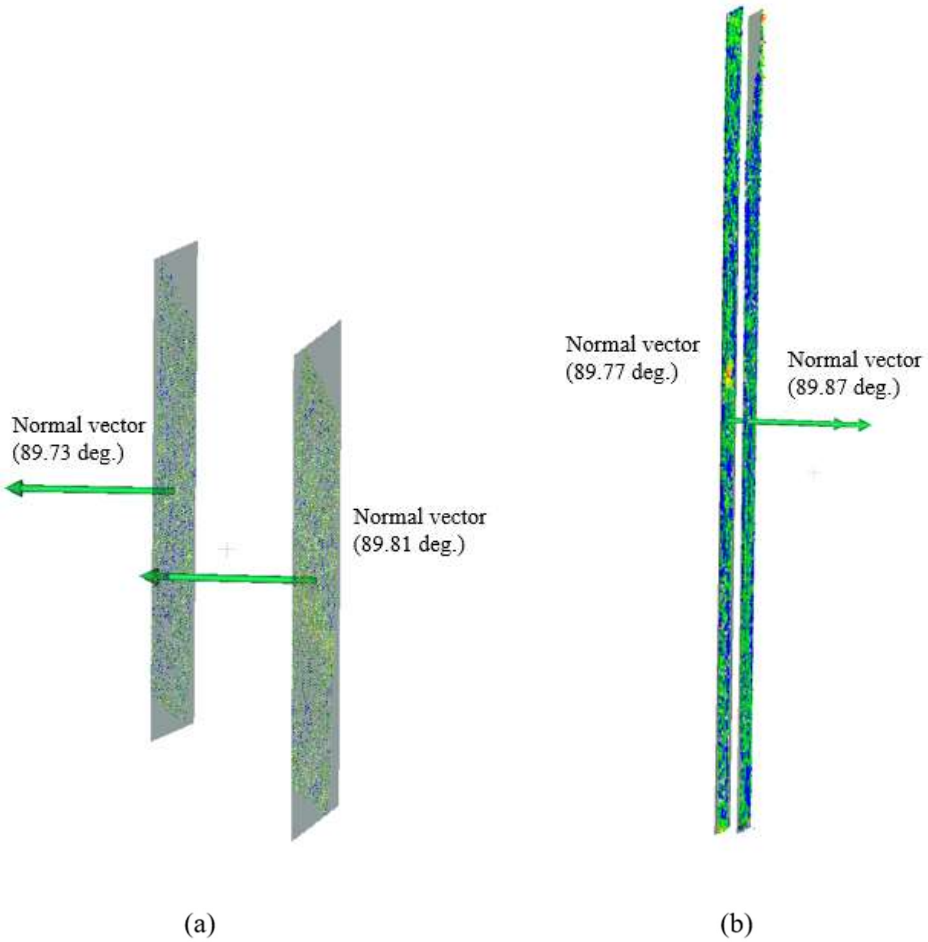
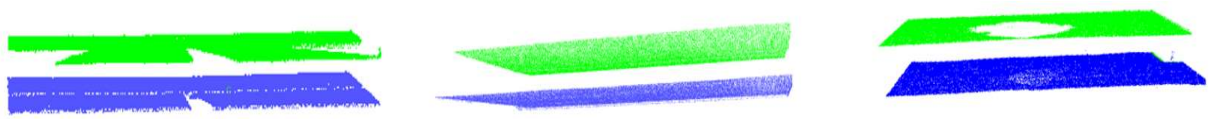
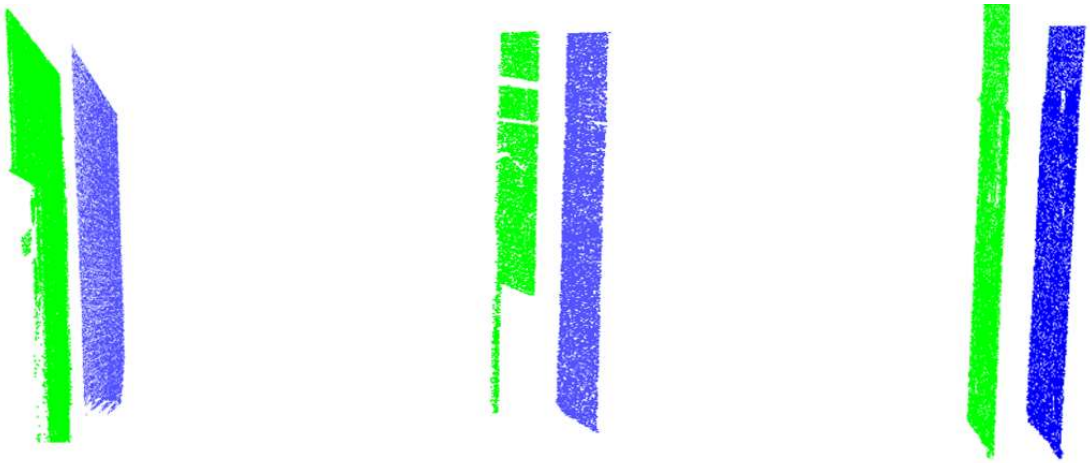


Figure 5. 7 Pairs of principal patches with close normal orientations: (a) Adjacent patches with normal's angular deviation of only 0.08 degrees; (b) Adjacent patches with normal's angular deviation of only 0.10 degrees.

As a result, pairs of principal patches are determined as illustrated as shown in Figure 5.8, where Figure 5.8(a) represents the horizontally oriented principal pairs of patches planes, and Figure 5.8(b) represents the vertically oriented principal pairs of patches.



(a)



(b)

Figure 5. 8 Pairs of generated principal planar patches: (a) Horizontal patches; (b) Vertical patches.

5.6.5. Search for the disconnected coplanar points using the principal patches

The principal patches ($\sum$) are then spanned using the linear combinations ($\boldsymbol{W_i}$) of their respective span vectors ($\vec{u}_i$) found from each of the planes (Π_i) of the principal patches. The spanned planes ($Span(\Pi_i)$), are then used to allocate their nearest corresponding points from the set $\acute{R}$. The points falling within the distance threshold (δ_{dist}) from the $Span(\Pi_i)$ are provisionally assigned as the coplanar point. Next, the VDBSCAN approach was applied to exclude the outliers, followed up by the screening for coplanarity. Finally, the points are fixed to the $Span(\Pi_i)$ and refined for the proceeding segmentation process which involves the classification of planes into structural elements. Figure 5.9 and 5.10 illustrates the spanned principal planes and their corresponding coplanar points drawn from the incomplete set of patches i.e., set $\acute{R}$.

Prior to the classification process, the spanned planes are updated, now to include the principal patches ($\sum$) and their newly assigned coplanar points from the set $\acute{R}$. Then, the updated planes in their entirety are arranged and clustered into horizontal (Π_h) and vertical (Π_v) with respect to their orientations pertaining to the conditions described in previously. Thereon, the planes are segmented to embody the structural elements of regular buildings in the following order: *floor slabs*, *floor beams*, *walls*, and *columns*.

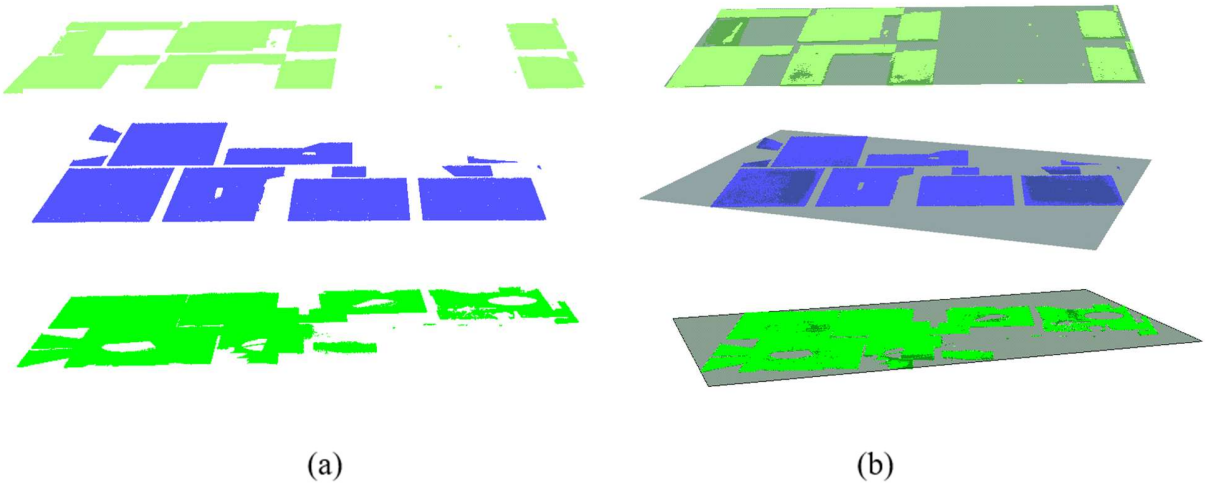


Figure 5. 9 Results of spanning the horizontal principal planes and the subsequent assignment of disconnected coplanar patches: (a) disconnected patches; (b) planes fitted after spanning the principle patches and assigning the corresponding coplanar patches.

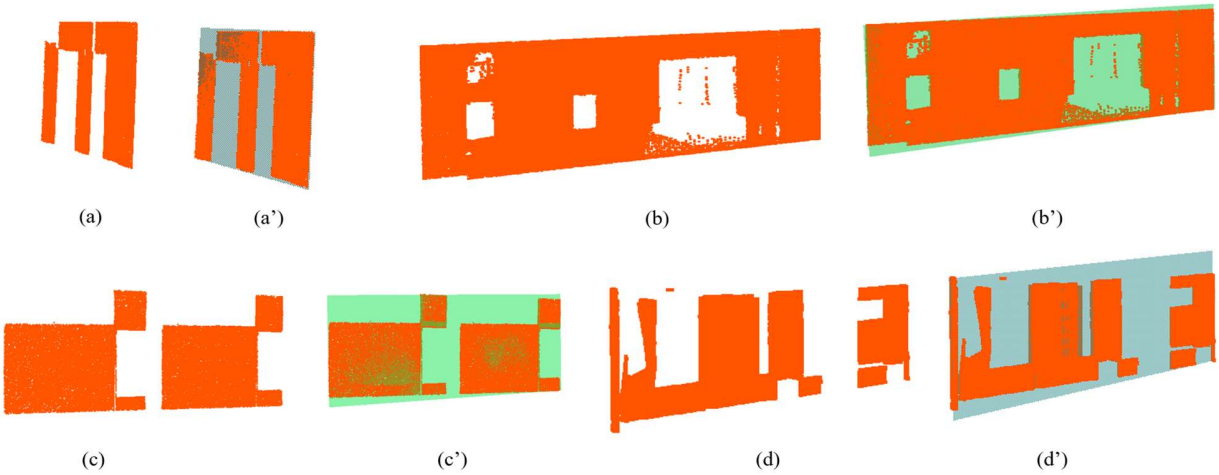


Figure 5. 10 Results of spanning the vertical principal planes and the subsequent assignment of disconnected coplanar patches: (a-d) disconnected patches; (a'-d') planes fitted after spanning the patches and assigning the corresponding coplanar patches.

The point assignment to the spanned plane is followed up by the removal of outlier points and minimal clusters generated using the DBSCAN approach as shown in Figure 5.11(a). The elimination of minimal clusters helps to avoid the inclusion of small clusters as coplanar regions as presented in Figure 5.11(b) since the assignment of points in the Span(□) may overlook these regions.

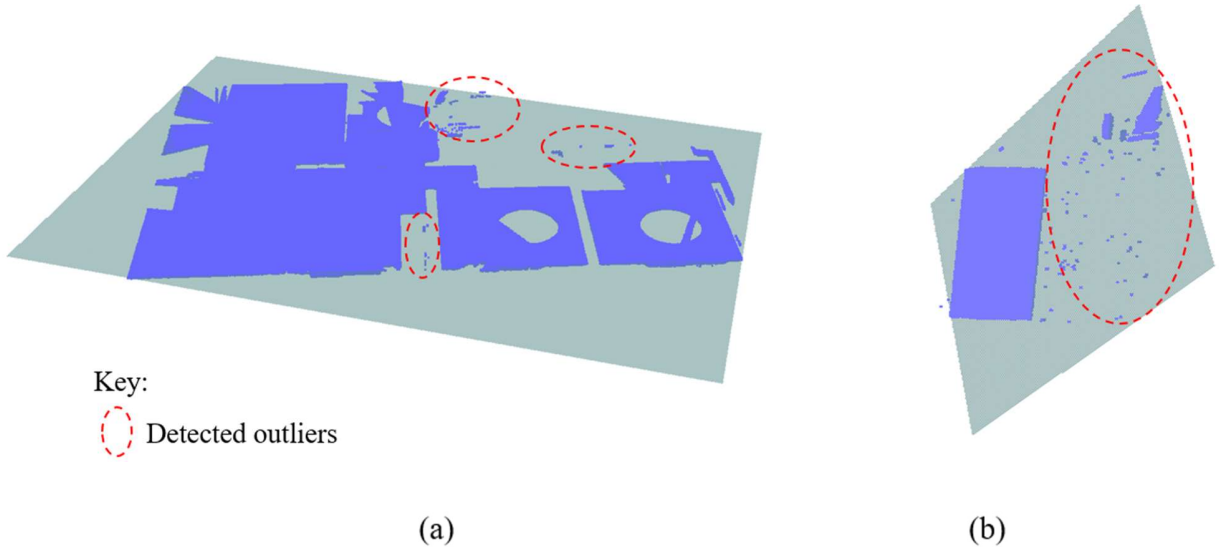


Figure 5. 11 Point assignment to the spanned principal planes in searching for coplanar points and patches: (a) Outliers detected; (b) Surface patches and points from the clutter objects converge with spanned plane.

5.6.6. Segmentation of floor slabs

The horizontal planes ($\mathbb{\Pi}_h$) are used in this case to segment floor slabs from the detection of the planes representing the surfaces of the ceilings ($\mathbb{\Pi}_{h(i,0)}$) and floors ($\mathbb{\Pi}_{h(0,j)}$). The parameter thresholds stipulated for story heights (h_i) and slab thickness (t_s) are used to subjectively provide the logical sequencing of horizontal planes corresponding to the floor slabs. The first floor denoted the particular plane as $\mathbb{\Pi}_{h(0,1)}$, following that, the plane representing the ceiling for the second floor is determined ($\mathbb{\Pi}_{h(1,0)}$) using the height threshold (h_i). Then, the process is recursively repeated in the vertical direction to determine the subsequent ceilings and floors as illustrated in Figure 5.12. As a result, we identified and clustered five horizontal planes corresponding to the floor surfaces and four horizontal planes for the ceiling surfaces that resulted in a total of four-floor slabs, where;

- $\mathbb{\Pi}_{h(0,1)}$, represents the floor surface for the 1st floor slab;
- $\mathbb{\Pi}_{h(1,0)}$, represents the ceiling surface for the 2nd floor slab;
- $\mathbb{\Pi}_{h(0,2)}$, represents the floor surface for the 2nd floor slab;
- $\mathbb{\Pi}_{h(2,0)}$, represents the ceiling surface for the 3rd floor slab;
- $\mathbb{\Pi}_{h(0,3)}$, represents the floor surface for the 3rd floor slab;
- $\mathbb{\Pi}_{h(3,0)}$, represents the ceiling surface for the 4th floor slab;
- $\mathbb{\Pi}_{h(0,4)}$, represents the floor surface for the 4th floor slab;
- $\mathbb{\Pi}_{h(4,0)}$, represents the ceiling surface for the 5th floor slab;

- $\mathbb{\Pi}_{h(0,5)}$, represents the floor surface for the 5th floor slab.

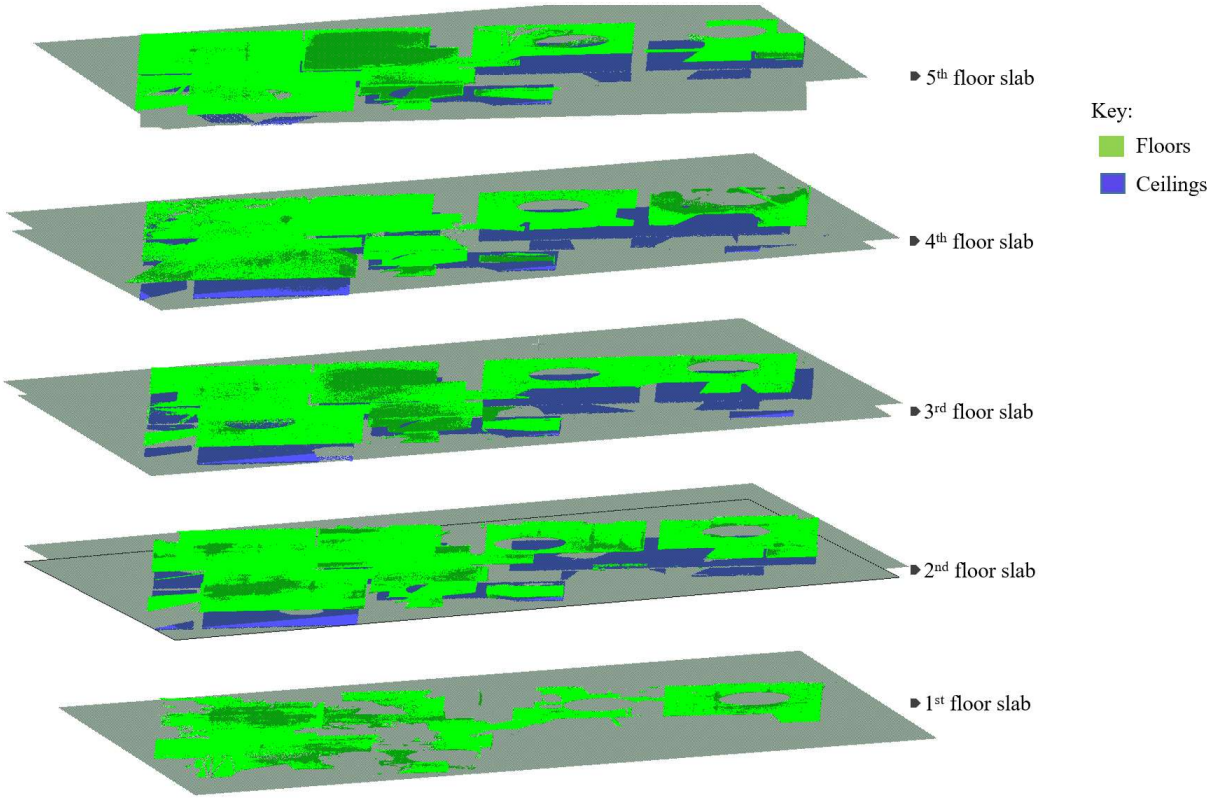


Figure 5. 12 Floor slabs detected in the experiment are represented by the pairs of ceiling and floor planes.

5.6.7. Segmentation of floor beams

Methods proposed in this research are designed to classify surface patches and planes into floor beams through the detection of the three main surfaces of floor beams: *soffits* and the two opposing *vertical sides*. The beam's soffits are detected from the *non-principal patches* in cluster R and the points remaining in clusters $\acute{R}$. The ray casting operation is conducted using Open3D software, where the input dataset is inserted in the process consisting of separate batches. The batches are each comprised of the detected ceiling plane ($\mathbb{\Pi}_{h(i,0)}$) and the corresponding unassigned points from the sub-clusters (ω_i) that are within 1.00 metres below the ceiling plane. The bitmaps (S_i) are created using flat layer octree build-up and then, the beam of rays (r_i) that are equivalent to the number of bitmaps created is cast onto the batch in the direction of the ceiling's normal vector ($\vec{N}$) i.e. $\sum_{i=i}^{n} r$ & $\sum_{i=i}^{n} S$, where n denotes the number of rays and bitmaps created from the ceiling plane.

The intersection of rays $\mathbf{y}(t)$, is computed from each batch, and the bitmaps (voxels) hit below the planes are identified and used to extract the points bucketed in them. These points are refitted with new planes in the direction of the ceiling plane in a similar batch. The points are segmented

and labeled to represent the soffits of floor beams. After that, the vertical sides of beams are detected using the detected soffit points. Figure 5.13 illustrated the combination of the segmented patches for soffits and vertical sides of floor beams.

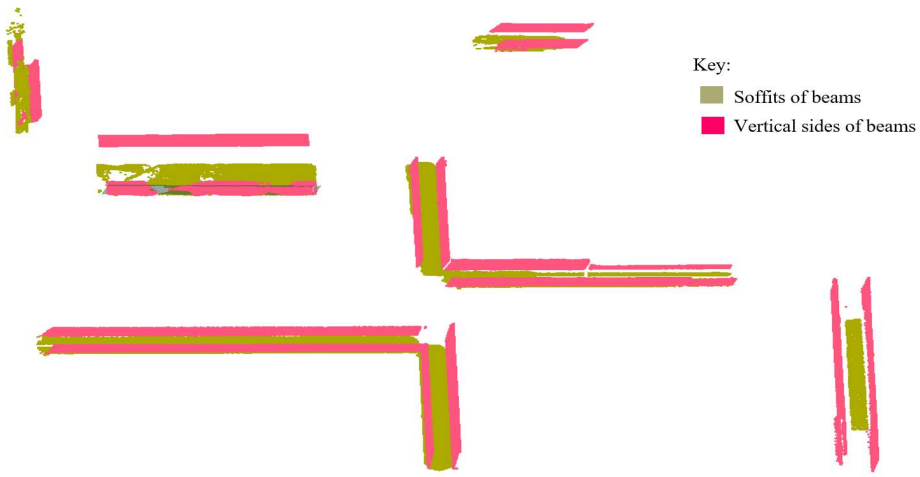


Figure 5. 13 The detected soffits and vertical sides of the 2nd-floor beams.

5.6.8. Segmentation of Walls

The vertical pairs of planes ($\mathbb{\Pi}_v$) are used to detect the walls and columns found between the created vertical sub-spaces which are created between the segmented floor slabs. The vertical sub-spaces are created between the successive floor slabs, such that each vertical sub-space pertains to a storey height (h_i). As a result, a total of four (4) vertical subspaces are created corresponding to storey heights.

The created vertical subspace, h_{max}, $h_{min,}$ and the thickness (t_w) are used to determine the wall planes from the pool of vertical pairs of planes ($\mathbb{\Pi}_v$). The threshold t_w is used to empirically separate the load-bearing walls and columns from other vertical objects including the light partitions and occlusions along the walls. As a result, a total of 44 walls are determined and segmented. Figure 5.14 illustrates the walls detected from the first storey.

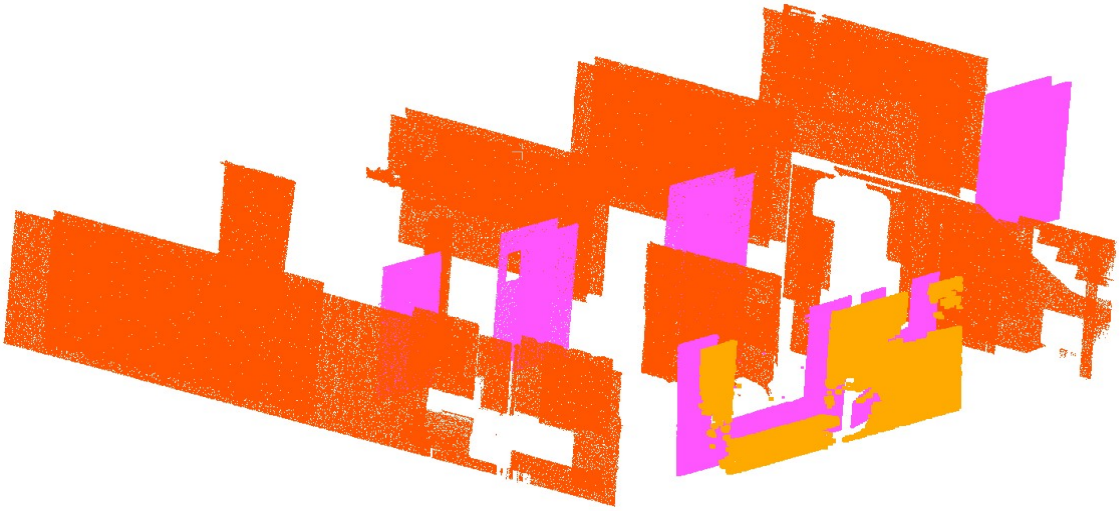


Figure 5. 14 The detected walls at the first floor

References

Awrangjeb, M., & Fraser, C. S. (2014). Automatic segmentation of raw LIDAR data for extraction of building roofs. *Remote Sensing*, *6*(5), 3716–3751. https://doi.org/10.3390/rs6053716

Ntiyakunze, J., Inoue, T., Jian, M., & Aoki, S. (2022). Study of Efficiency Improvement of Site Survey using DX in Renovation projects. - Refinement of Multi-Storey Apartment Building -. *37th Proceedings of Symposium on Building Construction and Management of Projects*, 205–212.

Nurunnabi, A., Belton, D., & West, G. (2012). Robust segmentation for multiple planar surface extraction in laser scanning 3D point cloud data. *Proceedings - International Conference on Pattern Recognition*, *Icpr*, 1367–1370.

Poux, F., & Billen, R. (2019). Voxel-based 3D Point Cloud Semantic Segmentation: Unsupervised geometric and relationship featuring vs deep learning methods. *ISPRS International Journal of Geo-Information*, *8*(5). https://doi.org/10.3390/ijgi8050213

Qian-Yi, Z., Jaesik, P., & Vladlen, K. (2018). *Open3D: A Modern Library for 3D Data Processing* (0.16.0). https://doi.org/10.48550

Vo, A. V., Truong-Hong, L., Laefer, D. F., & Bertolotto, M. (2015). Octree-based region growing for point cloud segmentation. *ISPRS Journal of Photogrammetry and Remote Sensing*, *104*, 88–100. https://doi.org/10.1016/j.isprsjprs.2015.01.011

Xiong, X., Adan, A., Akinci, B., & Huber, D. (2013). Automatic creation of semantically rich 3D building models from laser scanner data. *Automation in Construction*, *31*, 325–337. https://doi.org/10.1016/j.autcon.2012.10.006

Zoller + Fröhlich. (2022). *Z + F Imager, 5016 Terrestrial Laser Scanner Datasheet and Key Peformance Specifications*.

Chapter 6. Evaluation and Discussions

6.1. Introduction

This chapter covers the evaluation of the experimental results in terms of the overall quality and quantitative performance which measures up the observed results against the ground truth inferred from the existing building. The chapter also covers the comparative evaluation that includes the comparison of the proposed segmentation method versus the state-of-the-art segmentation methods. In addition, it discusses the limitations of the proposed method which includes the major problems.

6.2. Quantitative Evaluation

The quality of the classification process is evaluated by comparing the observed class elements from the experiment results against those of the ground truth inferred from the existing building. Quantitative evaluation of the classification of points into several elements (*floor slabs, beams,* and *walls*) is performed using the five metrics: precision (*pr*), recall (*r*), F_1-score, Intersection-Over-Union (IoU), and overall accuracy (*OA*).

Some elements can induce dominance over other elements due to their large quantities and for this different metrics of measurement are used for quantification. The floor slabs are enumerated in their composed ceiling and floor surfaces. As for the floor beams and walls, the linear measurement along the x and or y-direction to get their dimensions in running lengths. This is due to the distinct nature of beams and wall layouts where they are represented in partial lengths between corners or columns and often occupy several arrangements and orientations dividing the floor or ceiling spaces. For the walls, the detected pairs of vertical planes are measured by their average extreme dimensions excluding the openings.

The overall measurements of different class elements from the ground truth data and those obtained from the experimental results are provided in Table 6.1.

Table 6. 1 Measurements of structural elements from ground truth data and experimental results.

Object	Floor slabs			Floor beams			Walls
	Ceilings	Floors	[1]Slabs	Soffits	Vertical sides	[1]Beams	
Ground truth data	4	5	5	213	213	213	242
Experimental results	4	5	5	284	242	242	310

Results from Table 7.1 shows a resemblance in the quantities associated with floor slabs between the ground truth data and experimental results. However, the measurements obtained from the output of the experiment have indicated significant false measurements for the classes of floor beams and walls, where the variation increase is 14% and 28% for the floor beams and walls respectively. Furthermore, it is noted that the experimental results have produced different measurements between the soffits and the vertical sides of the beam i.e. 284m for soffits and 242m for the vertical sides. This is illustrated in Figure 6.1 that the soffits are observed to be longer compared to the vertical sides, and this is attributed to two factors; the order of detection of beams

where the soffits are detected first and then used to find the vertical sides under limited conditions. Hence, since the detection of vertical sides is preceded by the detection of soffits, the amount detected for the vertical sides is predestined to have lesser or equal scores compared to the soffits; and another factor is the inherent undersize of planar patches forming the vertical sides.

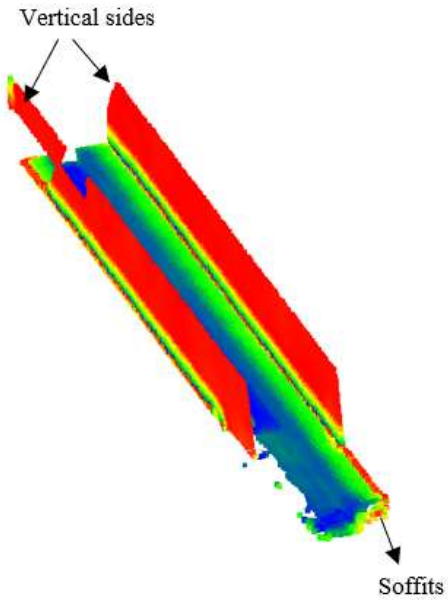


Figure 6. 1 Part of the floor beam with the different sizes of the soffits and vertical sides.

The overall results of the experiment using $Portion_1$ are presented in Table 6.2, where TP stands for True Positives, FN for False Positives, FN for False Negatives, and TN for True Negatives.

Table 6. 2 Comparison of the overall experimental results and ground truth data.

Metrics	**Floor slabs**			**Floor beams**			**Walls**
	Ceilings	Floors	[1]Slabs	Soffits	Vertical sides	[1]Beams	
TP	4	5	5	192.00	149.76	149.76	202.00
FP	0	0	0	92.16	92.16	92.16	108.00
FN	0	0	0	20.80	20.80	20.80	40.00
TN	0	8	8	23.40	23.40	23.40	28.00
Precision	1	1	1	0.68	0.62	0.62	0.65
Recall	1	1	1	0.90	0.88	0.88	0.83
F_1-score	1	1	1	0.77	0.73	0.73	0.73
IoU	1	1	1	0.63	0.57	0.57	0.58

[1] Slabs and [1] Beams are subject to the identification of all their constituent parts.

From Table 6.2, the floor slabs and beams are represented by their constituent surface planes, (i.e., the ceilings and floors for the floor slabs, and the soffits and vertical sides for the beams).

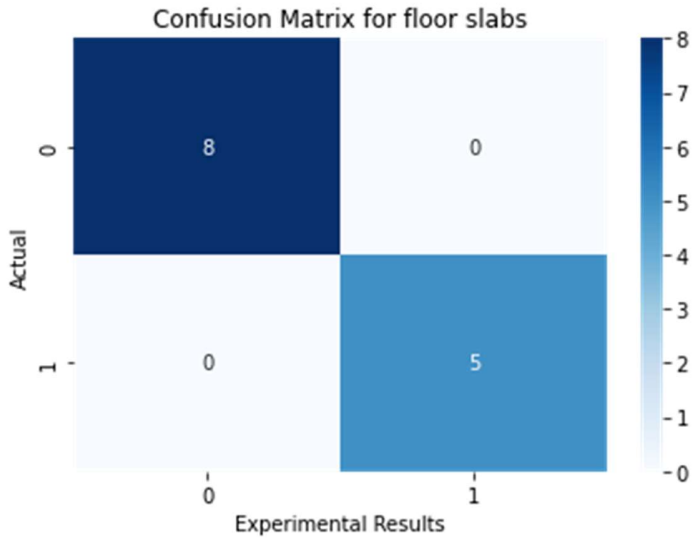


Figure 6. 2 Confusion Matrix for the Segmented Versus Actual Floor Slabs

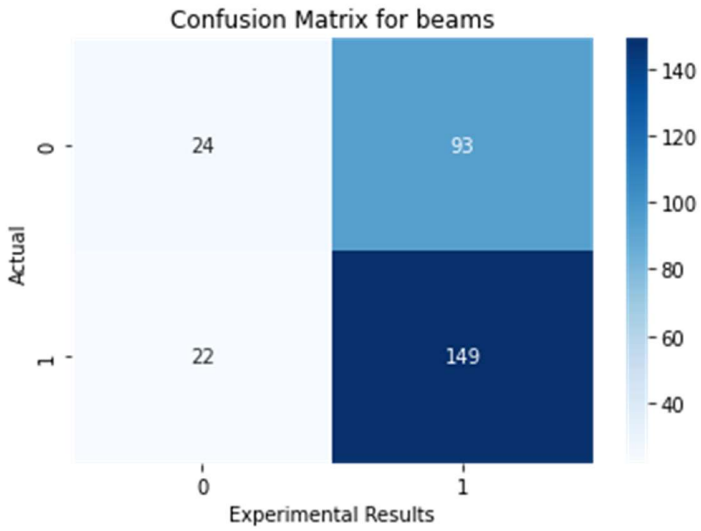


Figure 6. 3 Confusion Matrix for the Segmented Versus Actual Floor Beams

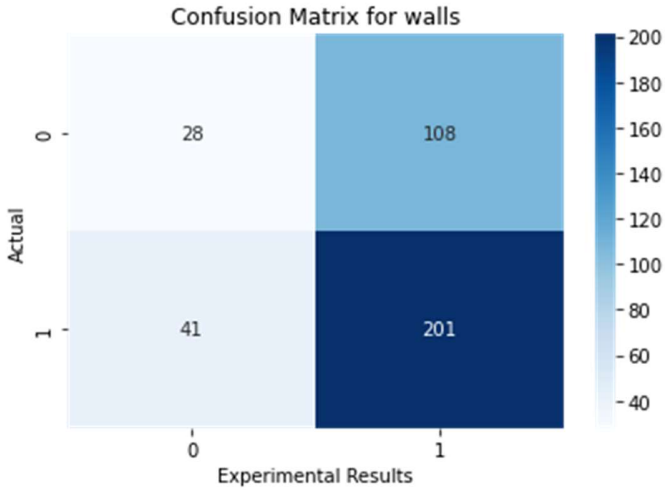


Figure 6. 4 Confusion Matrix for the Segmented Versus Actual Walls

As Table 6.2 and Figure 6.2. show, all the ceilings and floors are perfectly detected and segmented using the proposed method, with a score of 1 in all the quantitative metrics. Therefore, all the floor slabs are detected with reference to the ground truth data. The first-floor slab appeared in one planar surface which is identified as the floor surface, while the rest of the floor slabs are presented as pairs of surfaces of ceilings and floors. In the evaluation process, the ceilings and floors are considered to their full dimensions while disregarding the size or number of gaps (referred to as *holes*) between the coplanar patches as shown in Figure 6.5. However, the holes which are due to the missing data led by occlusions can significantly influence the segmentation results when working with relatively smaller floor slabs. The experiment is based on the interior slabs and the exterior recessed slabs only; the protruding balconies are not considered due to the hidden ceilings and excessive occlusions.

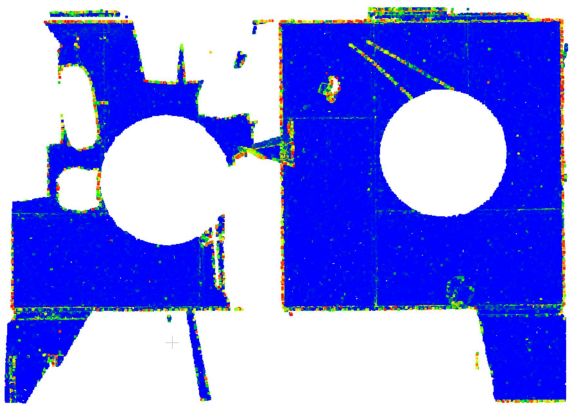


Figure 6. 5 Holes present on the surfaces of the point cloud

The TN values cover the performance of the proposed method to correctly predict the negative class. In this case, the countertops, which are captured by the scanner are formulated as the negative

class for the floor surface, and the model has correctly classified them identified as not floors. Figure 6.6. demonstrates the presence of the countertops and their detected planar patches.

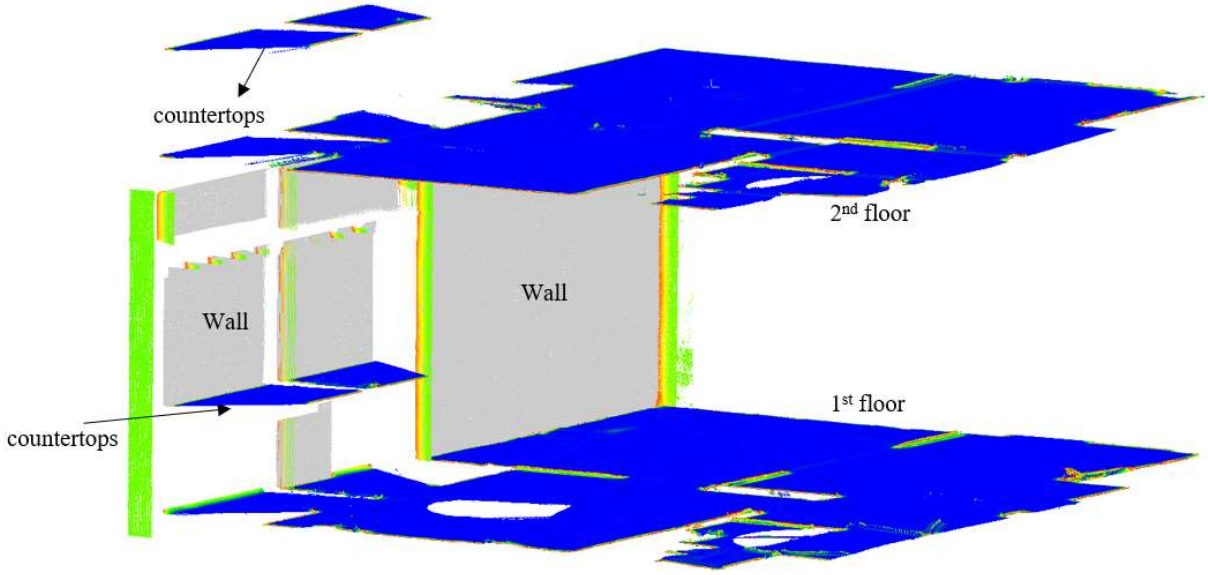


Figure 6. 6 Detected planar patches for the countertops, floors, and walls.

The floor beams showed positive performance, as shown in Table 6.2 and Figure 6.3, amidst a high level of clutter on the ceiling surfaces. Beams as a whole element are scored over 0.60 in terms of precision, recall, and F_1-score, and the IoU which measures the worst-case performance, is slightly above 0.5. As mentioned previously, there is a slight difference accounted for in the detected soffits and vertical sides of beams, which among other reasons mentioned is also caused by the presence of occlusions along the sides of the beams.

According to our approach, the vertical sides can only be detected if both planes are in a pair set of vertical planes, so if one face is occluded it may affect the detection of both faces as a pair, which also hindered classifying the object as a beam as shown in Figure 6.7.

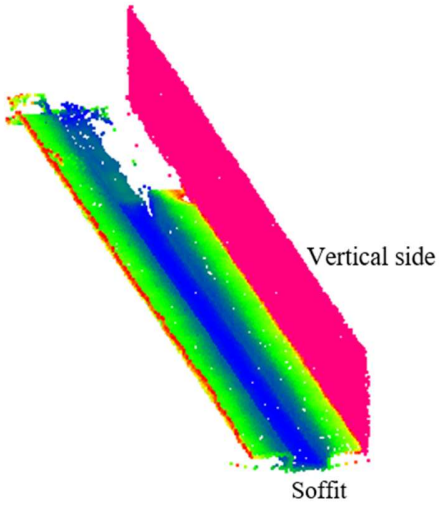


Figure 6. 7 Incomplete detection of the beam due to the missing one vertical side.

Some overhead (upper) cabinets with breadths not exceeding 50 cm are falsely identified as beams contributing to the value of the FP as shown in Figure 6.8. In general, the detection of beams is affected by the amount of surfaces exposed to the scanners. In particular, the segmentation of the isolated beams performed better compared to the segmentation of *revealed beams* with walls attached under them as demonstrated in Figure 6.9, and this is due to the difference in the extent of the exposed undersides (soffits) of the beams.

Figure 6. 8 The overhead cabinet falsely classified as a beam.

Figure 6. 9 Different types of exposure of floor beams.

In accordance with Table 6.2 and Figure 6.4, the quality of the proposed method approach in segmenting the walls also showed positive results. The approach has accurately not labeled the non-wall points as walls are presented with a precision score of 0.65, and the recall measure, which is related to the ability of our classification process to find all the wall points, recorded a score of 0.83. The worst-case performance of the segmentation and classification approach is marked at 0.58 IoU, which is still acceptable as it is above a threshold of 0.5. In general, the method has shown good performance in classifying the wall elements despite the substantial amounts of temporary and fixed objects on the faces of the walls.

The existing building contains a substantial amount of non-load-bearing walls, whereby most of the interior walls are made up of *shôji* (traditional Japanese sliding walls or doors) and glass partitions as shown in Figure 6.10 (a) & (b). Since the drywall partitions have the same heights (2.50–3.20 m) as those of load-bearing walls (solid walls), the classification method had to rely mainly on the differences in their thicknesses to distinguish the load-bearing from the non-load-bearing walls. A thickness tolerance (t_w) applied in the classification (for 10 cm $\leq t_w \geq$ 50 cm) managed to dismiss most of the drywall partitions except for a few partitions with thickness >10 cm, which accounted for the FP. The value of the FP was also attributed to the incorrect classification of certain wardrobes, which displayed dimensions (height and width) that covered the full extent of one face of a wall and a breadth (w) not exceeding 50 cm as illustrated in Figure 6.11. The value of FN is contributed mostly by the amount of undetected external walls. This is due to the restriction of the scanner's access in the narrow spaces leading to the limited exposure of the exterior faces of the external walls. The presence of trees in the vicinity of the building also partly limited the detection of the external walls.

(a) (b)

Figure 6. 10 Drywall partitions mistaken; (a) Glasswork partitions; (b) shoji

Figure 6. 11 The Wardrobe falsely classified as wall

6.3. Qualitative analysis

6.3.1. Influence of voxel resolutions

The voxel resolutions which refer to the size of the 3D grid cells used to contain a small dataset of point cloud, can have a significant influence on the accuracy and efficiency of the plane detection pipeline. In the experiment, where the octree system is adopted to decompose the point cloud, the voxel size is maintained at the same size at each corresponding internal level. The resolution is controlled by the two factors simultaneously: the maximum depth (Ψ_{max}) = 6, and the minimum number of points per voxel ($\mathcal{E}_{min}$) = 30.

The resolution performed well in fitting planes on actual planar surfaces, however, spurious planes were created in cases with highly scattered points in voxels as illustrated in Figure 6.12. Nonetheless, more often these spurious planes were discarded due to a lack of a merging partner nearby since they did not represent any real surface of a building component, hence they were considered undergrown patches and inserted in the set $\acute{R}$.

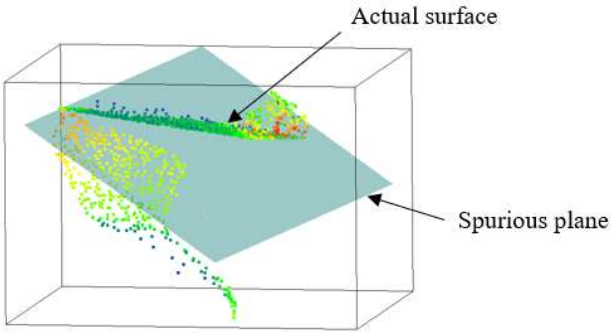


Figure 6. 12 Spurious plane detected in the voxel due to sparsity of data.

6.3.2. Analysis of the Classification of detected planes

Figure 14b shows an input (subsampled) point cloud P with a total of 30,129,150 points, and after the spatial decomposition, the planar patches are derived and classified accordingly into class elements. As the result of the segmentation and classification processes into structural elements, the points remaining unclassified ($\bar{P}$) are found to represent about 35% of the original points (P) such that $\bar{P} \subseteq P$.

Analysis was conducted on the remaining points to gain insight into the relevancy of the unclassified points. Manual inspection of the remaining planar patches is done to determine if they may represent any of the surfaces of structural elements. Among them, about 20% were found to be part of the class objects largely associated with the external walls which also contributed in large proportion to the FN value for the walls (i.e., undetected walls). This is mainly due to a lack of the corresponding pair plane on the other side, and this was sensitive since the wall can be classified it is represented wholly by two planes. Furthermore, among the remaining unclassified pairs of vertical planes ($\mathbb{II}_v$), 90% of them are found to represent the drywall partitions and doors which were successfully rejected because of the thickness (t_w) threshold. The building did not have any exposed columns that required the segmentation process.

It is also observed that the planar patches associated with floor surfaces were disposed to a high number of occlusions that led to missing points in the planes. Clutters present on the floor were not removed during scanning and the segmentation method is tasked to correctly identify them and properly distinguish them from the floor points. In most cases, the missing points appeared as holes on the surface patches as illustrated in Figure 6.13, this is caused by the shadow effect of the scanner for each of the scanning positions. This may have affected the segmentation results for floors if the floor surfaces were smaller.

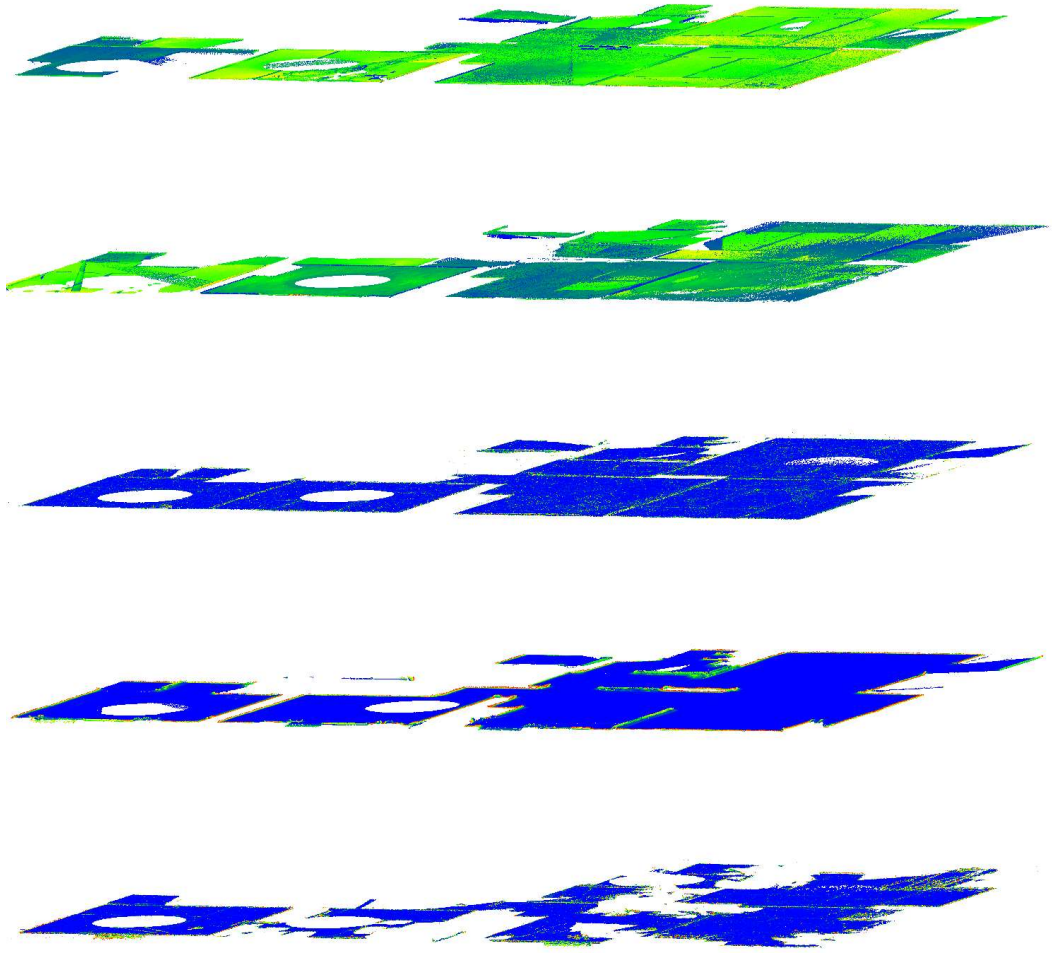


Figure 6. 13 Surface patches for floors with missing points and holes.

6.4. Comparative evaluation

To compare the performance of the proposed segmentation method with the recent outstanding methods related to clustering and deep learning methods, a different set of point clouds was cut from the original dataset for an independent and unbiased evaluation. A relatively smaller point cloud is used in this section to allow adaptation of each method used in the comparison. Two methods are selected for the comparison; the Density-based clustering approach as applied in (X. Chen et al. (2022), and an optimized DGCNN with the neighbor network as proposed in (Hyunsoo & Changwan (2021). These methods have been tested for indoor point cloud scenes to segment various building components and have shown good performance in their respective experiments.

6.4.1. Comparison dataset

The test dataset Pts, consists of a point cloud cut from the building on the second floor as shown in Figure 6.14, which includes scenes from the living room, bedroom, washing room, and entrance porch. The dataset (Pts) is already downsampled resulting in a total of 7,360,289 points.

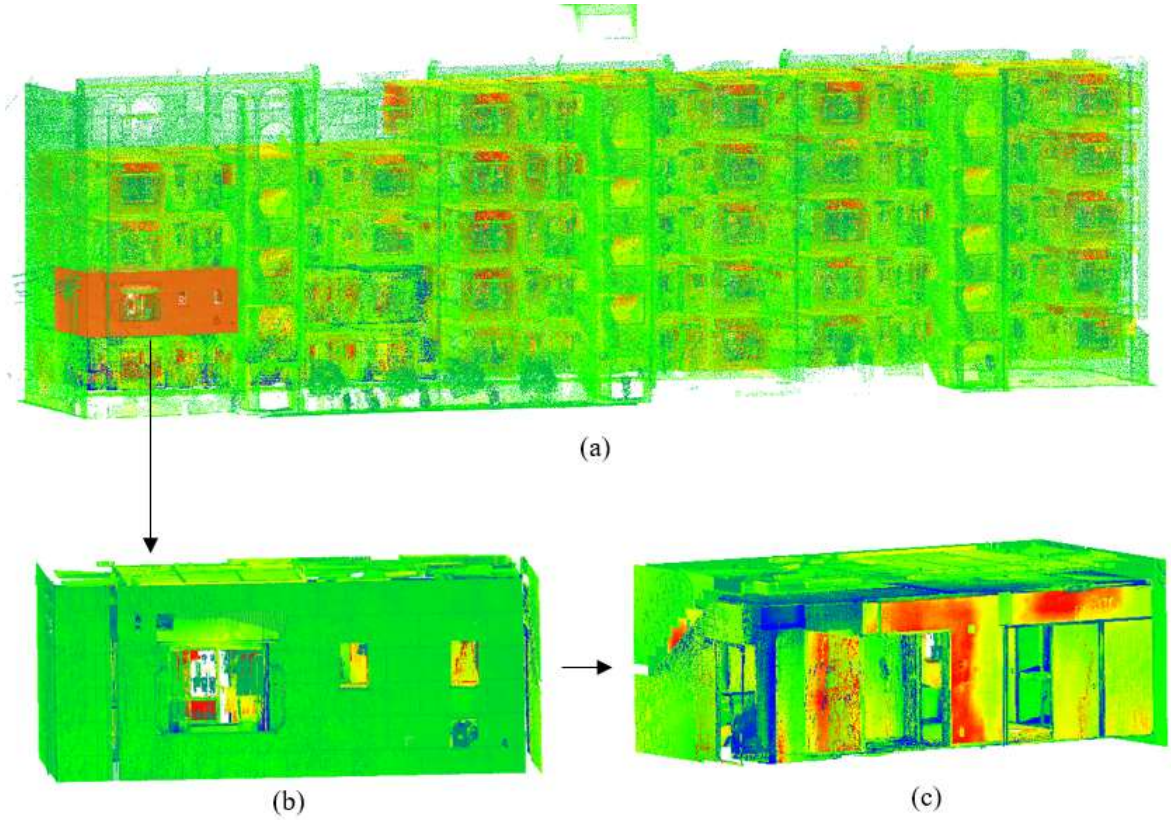


Figure 6. 14 Comparison dataset: (a) Original Point Cloud; (b) Portion of the original dataset for experiment testing – Rearview; (c) Cut portion of the original dataset for experiment testing – Front view.

6.4.2. Proposed segmentation method

Similar procedures are used in the segmentation of the dataset as described in subsection 5.6, where the point cloud is first voxelized, followed up by detecting and merging planar patches. Lastly, the pairs of refined patches are classified accordingly into classes of elements which are floor, ceiling, beams, and walls. The results of the segmentation are presented in Figure 6.15. The color seen in the figures represents the scalar values and normal orientations for the planar patches.

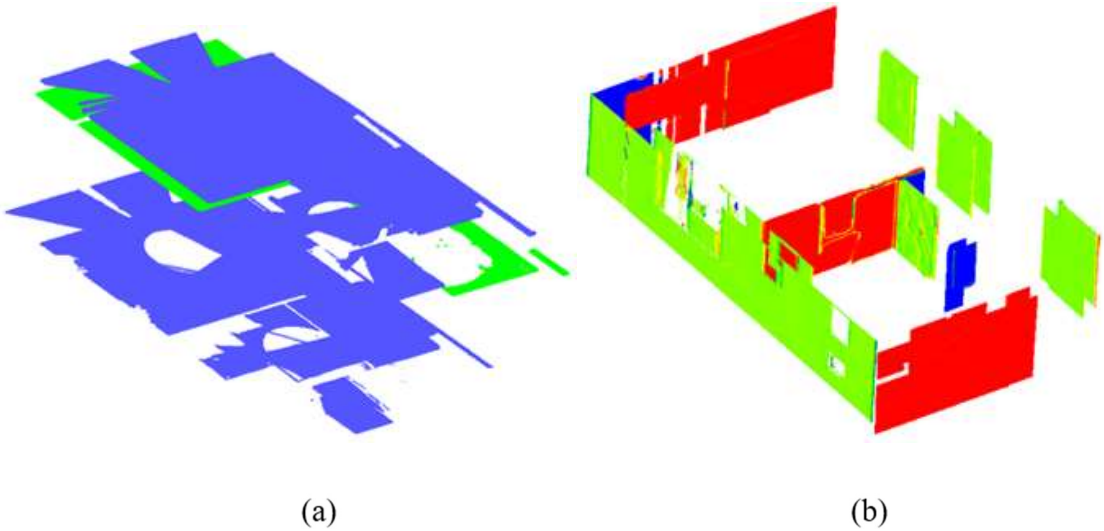


Figure 6. 15 Segmentation results based on the proposed method: (a) Segmented surfaces for floors and ceiling; (b) Segmented walls.

6.4.3. Segmentation based on density-based clustering method (X. Chen et al., 2022)

In this paper, a density clustering model based on the exponential function is utilized to determine the local density within a cutoff distance. The boundary-to-indoor point cloud distances are initially used to create a constraint condition for wall density clustering. Subsequently, the density clustering of the z direction and local density model is employed to extract the ceiling and floor.

In implementing the method proposed in this study for the comparative dataset (Pts), first the density of the input point cloud (Pts) is estimated using the developed exponential function model. The model involves assigning a density value to each point cloud based on its distance from neighboring points. Then, the wall density clustering is performed where the wall points are clustered based on their density values using wall constraint conditions which are room boundaries. The distances between the boundary and indoor point clouds are used to create these constraint conditions. Points with densities above a threshold are clustered to form walls. As for the ceilings and floors, the density values of points in the z-direction are computed and grouped to form the ceiling and floor. The resulting segmentation model is shown in Figure 6.16.

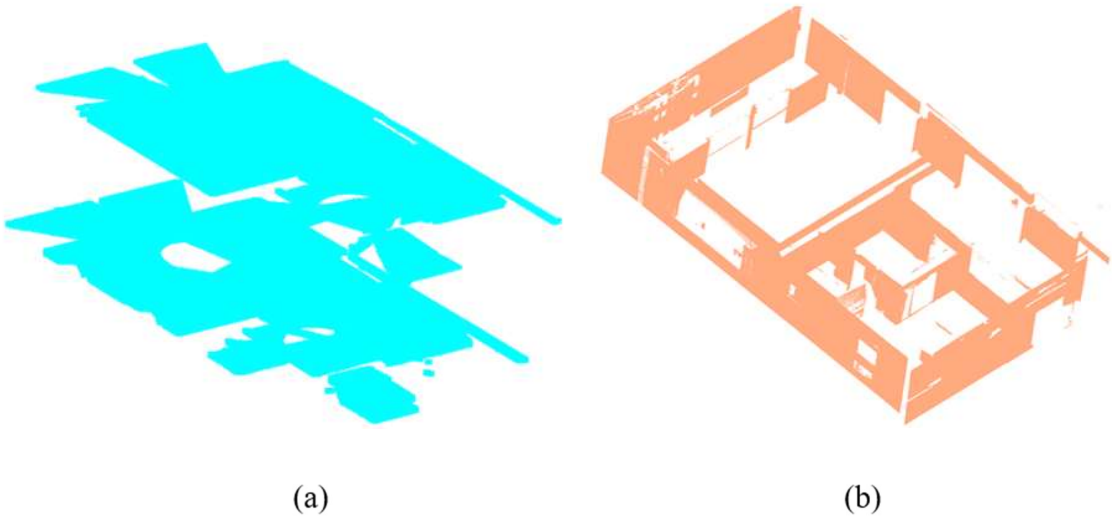


Figure 6. 16 Segmentation results based on the density-based clustering method: (a) Segmented surfaces for floors and ceiling; (b) Segmented walls.

6.4.4. Segmentation based on the optimized DGCNN with the neighbor network (Hyunsoo & Changwan, 2021)

The authors in this study proposed a method for creating 3D as-built models from incomplete point clouds by utilizing connectivity relations between neighboring elements. The methods involve using a DGCNN which is a deep learning method that can learn local features from neighboring points, unlike PointNet, which learns from the local and global features for all input points, regardless of the order of the input point clouds. The DGCNN enabled semantic segmentation at the point level to label each point including those coming from incomplete points. Then, from the segmented objects, the authors proposed to extract planar patches to break down the labeled objects into finer details since the DGCNN obtains broad segmentation results. The method involves segmenting the point cloud into smaller parts and using a connectivity graph to identify and connect the segmented parts. The authors also present an algorithm for resolving conflicts between connected segments. The proposed method is tested on both synthetic and real-world point clouds and is found to produce accurate as-built models with minimal manual intervention.

In implementing the method proposed in the study in the comparison evaluation, the input point cloud (Pts) is segmented into labeled classes using DGCNN, where the training of the dataset was conducted using the Stanford 3D Indoor Space (S3DIS) data (Armeni et al., 2017). Then, the planar patches are determined using the smoothness-constraint region growing (Rabbani et al., 2006) as suggested in the paper. Once the planar patches are obtained for each corresponding segment, the connectivity graph which describes the neighboring network is built where each sub-graph corresponds to a segment of the input point cloud. The connectivity which is based on the neighboring segments and planar patches managed to associate and combine the patches with

missing points. The methods resulted in the segmentation of several elements even those with incomplete points, the results of the segmentation method are illustrated in Figure 6.17.

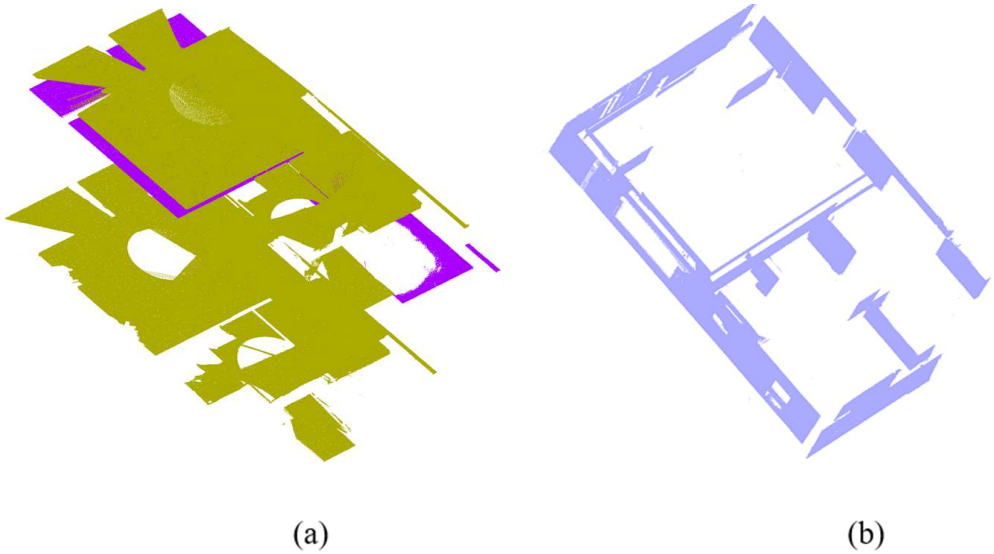


Figure 6. 17 Segmentation results based on the optimized DGCNN with the neighbor network: (a) Segmented surfaces for floors and ceiling; (b) Segmented walls.

6.4.5. Evaluation of Results

Table 6.3 provides combined evaluation results from the three methods used in the comparison evaluation.

Table 6. 3 Comparison of different segmentation methods.

Objects	**Floor slabs**			**Floor beams**			**Walls**		
Metrics	**pr**	**r**	**F_1-score**	**pr**	**r**	**F_1-score**	**pr**	**r**	**F_1-score**
Proposed method (A)	1	1	1	0.75	0.75	0.75	0.60	0.75	0.67
Density-based clustering (B)	1	1	1	0.25	0.33	0.29	0.70	0.64	0.67
DGCNN with the neighbor network (C)	1	1	1	0.67	0.67	0.67	0.70	0.70	0.70

Key: pr - precision, r - recall

From Table 6.3, all three methods have generally performed well in segmenting the floor slab and wall classes. The proposed method (method A) has an edge advantage in segmenting beams, while method B has shown a low performance in this category due to low scores in all metrics related to beams. Method B is prone to a lower recall value due to overreliance on the local density which fails to distinguish objects in the vicinity which is the case of floor beams and ceilings.

Method C shows slight advantages in segmenting wall class compared to other methods, which is attributed to the pairwise strategy of planes which is required in method A. This means that for a wall to be identified and segmented then both planes forming a wall must be detected.

6.5. Limitations of the Study

This study presents a method to segment raw point clouds into meaningful segments of structural elements. The approaches proposed in this study have shown the potential to classify structural elements in a large and noisy point cloud including other complex settings. However, the proposed method is fallen short in various situations.

First, the method has entirely focused on surface planarity and developed a strategy to detect and perform classification based on that. However, there is a significantly large stock of buildings built with non-planar members such as circular columns and beams. In that way, the proposed method should develop more adaptable techniques for incorporating different profiles and shapes of structural members. Also, the method was observed to be vulnerable to elements with relatively smaller surfaces such as the vertical sides of beams which is attributed to the reliance on planar patches in identifying these elements. Supplement techniques for dealing with smaller surfaces should be developed to solve this challenge.

Secondly, the research has not given enough attention to the points falling between the intersection lines of two or more patches and properly assign them to the respective patch as illustrated in Figure 6.18. This is an important aspect that is referred to as the boundary regularization process and is important for 3D remodeling (Macher et al., 2017; Z. Su et al., 2022). Assigning points falling on the intersection line to an appropriate surface patch will further improve the segmentation process.

Thirdly, from the experimental results, we have observed that our method is not performing well to classify the unpaired surface patches as the result of occlusions. Ways which can improve the approach towards classifying the single planes are essential, especially dealing with buildings with small rooms where the presence of small objects can occlude a larger part of its walls.

Finally, additional experiments with different settings should be performed to further validate the robustness of the proposed method. The proposed method has only been tested on a regular rectangular building with minimal variations of floor layouts and configuration of the main components. Further tests on other environments are necessary to evaluate the performance of the proposed method and recognize other areas of improvement. Also, other structural members should be incorporated such as stairs, outdoor suspended slabs (e.g., canopies), and cantilever beams.

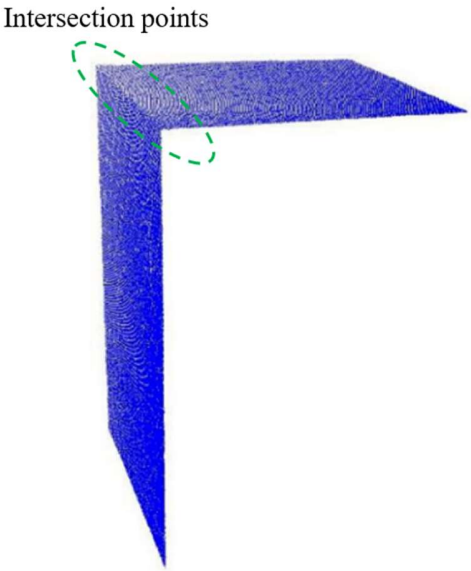


Figure 6. 18 Intersection points between two colliding planar patches.

References

Armeni, I., Sax, S., Zamir, A. R., & Savarese, S. (2017). *Joint 2D-3D-Semantic Data for Indoor Scene Understanding*. http://arxiv.org/abs/1702.01105

Chen, X., Wu, H., Lichti, D., Han, X., Ban, Y., Li, P., & Deng, H. (2022). Extraction of indoor objects based on the exponential function density clustering model. *Information Sciences*, *607*, 1111–1135. https://doi.org/10.1016/j.ins.2022.06.032

Hyunsoo, K., & Changwan, K. (2021). 3D as-built modeling from incomplete point clouds using connectivity relations. *Automation in Construction*, *130*(January). https://doi.org/10.1016/j.autcon.2021.103855

Macher, H., Landes, T., & Grussenmeyer, P. (2017). From point clouds to building information models: 3D semi-automatic reconstruction of indoors of existing buildings. *Applied Sciences (Switzerland)*, *7*(10), 1–30. https://doi.org/10.3390/app7101030

Rabbani, T., van den Heuvel, F. a, & Vosselman, G. (2006). (impo)(Fashuai exper+Sudan recom)Segmentation of point clouds using smoothness constraint. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences - Commission V Symposium "Image Engineering and Vision Metrology," 36*(5), 248–253. http://www.isprs.org/proceedings/XXXVI/part5/paper/RABB_639.pdf

Su, Z., Gao, Z., Zhou, G., Li, S., Song, L., Lu, X., & Kang, N. (2022). Building Plane Segmentation Based on Point Clouds. *Remote Sensing*, *14*(1). https://doi.org/10.3390/rs14010095

Chapter 7. Conclusion and Outlook

7.1 Introduction
7.2 Conclusion
7.3 Outlook

7.1. Introduction

The final chapter of this paper serves two objectives - firstly, to demonstrate the key conclusions resulting from the proposed methods presented in this study, and secondly, to suggest potential avenues for future research based on the limitations associated with the proposed method. The conclusions are drawn based on their relevance to the objectives pursued in the thesis and the three research questions outlined in chapter 1.

7.2. Conclusion

Research question 1: How to efficiently and accurately detect local features associated with planar surfaces from a large 3D point cloud that contains a high amount of noise and artifacts?

The task here is three-fold: dealing with the large point cloud, solving the challenge of noises and clutter situations, and detecting surface planarity while facing the mentioned challenges. The first solution to handle the large size point cloud is to divide the point cloud and deal with small portions one at a time, but rather in our approach, the point cloud is recursively divided, and each portion is connected to the original source along with the rest of other portions on the same level and any of the portions that would result from the proceeding divisions. Hence, the adoption of an octree-based voxelization helped to solve this by subdividing the point cloud into smaller sets using the octree structure where each octree (node) is connected to its parent octree (node). Then, points are allocated to each octree and stored in a voxel where it's easier to estimate the local attributes of points. The division is regulated to allow the optimum number of points to be stored in each voxel for statistical computations. The proposed partition involves a maximum depth of $\Psi_{max} = 6$, and the minimum number of points per voxel is $\varepsilon_{min} = 30$, whichever is reached earlier. This approach minimized the number of samples to deal with so as is the amount of noise and clutters, hence the PCA and plane fitting operations were less challenged since they are sensitive to outliers (noises). Hence instead of performing computations on the 66,997,720 points all at once, much lesser points are processed simultaneously. Therefore, by utilizing voxelization, the detection process is made more efficient and accurate while reducing the impact of noise and clutter in the original point cloud data.

The proposed method was able to detect about 1,487,525 minimal planar patches which are subsequently merged based on empirical similarities and proximities. It was noted that fake or spurious patches were also detected in the process which required more computation power to deal with, even though they were later rectified by discarding them due to lack of pairing partners. However, this challenge should further be improved to reduce the computation memory and time.

Research question 2: How to effectively determine the remote and disconnected planar patches in a point cloud belonging to the same surface of a particular structural element?

The main challenge in dealing with surface patches that are infected with missing points is to find which patches are deduced from a similar object. Point clouds containing occlusions, which are common for those acquired from the built assets, often result in disconnected patches often referred to as incomplete patches. The study has proposed a pairwise grouping of planar patches

by incorporating the spatial coordination of parallel patches in the neighborhood designed to assimilate the surface topology of structural elements. The grouped pairs of planes herein referred to as principal patches, are empirically spanned to allocate the candidate points representing the same surface. The process then continues to dispatch outliers and validate the planarity of the provisional inlier points by inferring the main direction of the respective principal patches. The proposed method managed to allocate and accommodate true coplanar points and patches that belong to a similar surface of the global representation. However, several points were still left out unassigned to either of the spanned planes which contributed to a higher FN value (false negative) of floor beams especially for the vertical sides. This was caused by the small sizes of patches corresponding to the vertical sides of beams which were difficult to detect.

Research question 3: What are the necessary aspects to consider in segmenting structural elements from the non-structural elements and secondary objects present in the point cloud scene?

To answer this question, this paper presents the concept of spatial dependency of structural systems of buildings which describes the connectivity of common building elements in terms of their locations, orientations, and functional connectivity. The concept is aggregated with the spatial information derived from the detected planar patches to establish meaning linkage and connection with the global attributes of structural elements. The features of planar patches associated with their spatial proximity, orientations, and locations helped to determine whether they belong to a particular structural element. The proposed method has been able to classify and segment points in their respective planar patches. This allowed for accurate segmentation of the structural elements in the point cloud scene.

The performance of the proposed method was assessed both qualitatively and quantitatively. The test results generally produced positive results, with each element being satisfactorily segmented. The floor slabs were 100% detected and segmented. The constituent surfaces composing the floor slabs (ceilings and floors) were also perfectly identified. The floor beams showed a promising performance with a precision rate of 62% and the ability to positively recall 88% of beams. Good results were also observed in detecting the walls, where the approach was performed at a precision rate of 65% and with the ability to recall 83% of walls, while the worst-case scenario was observed at 58% only. Most of the class elements were appropriately labeled. We also compared our proposed method with other recent and reputable segmentation methods related to built assets and the performance results in segmenting walls and floors were closely similar, while our proposed method performed better in segmenting floor beams

7.3. Outlook

The main objective of this thesis was to identify and reconstruct building objects from the construction site and built environment. While this goal has been achieved, there is still room for further improvement. In the future, the following areas would be explored and examined.

The octree-voxelization structure should be designed to be more adaptable. The resolution of the voxels which covers the size and population of samples in the voxel should be guided with

lesser of tuning parameters, which also applies to other selected thresholds used in the study. The proposed methods have been shown to incline much about setting the parameters which would not fit every situation.

The use of surface planarity could be further optimized. The proposed method has identified structural elements by identifying planar surfaces in the point cloud data while disregarding other potential characteristics or attributes that could be used to identify and segment the elements. This may limit the accuracy or completeness of the segmentation results, especially if some structural elements are not predominantly planar or if non-structural elements share similar planar surfaces with the structural ones.

One of the primary objectives of this study is to enhance automation in the segmentation pipeline, by utilizing more efficient algorithms and methods to further enhance the performance. However, deep learning methods have increasingly been reported to have more significant advantages over hand-crafted features, resulting in better distinctiveness and less time-consuming. The utilization of newly developed neural networks and our proposed method can assist in overcoming the limitations of dataset acquisition by filling in missing points caused by occlusion which is a critical challenge to overcome.

At last, further developments should be consummated in the direction of efficient and parallel operations which can permit an increase in time efficiency. The current operations have involved different equipment and programs from point cloud registration, feature estimation, plane fitting, and merging of planar patches, to the resulting segmentation. This way it can increase the speed and reduce the time required to process the point cloud data. By doing so, the method can become more practical and useful for real-world applications where processing large amounts of data in a short amount of time is critical. The author believes that by making these improvements, the method can achieve better performance and become a more valuable tool for researchers and practitioners in the field.

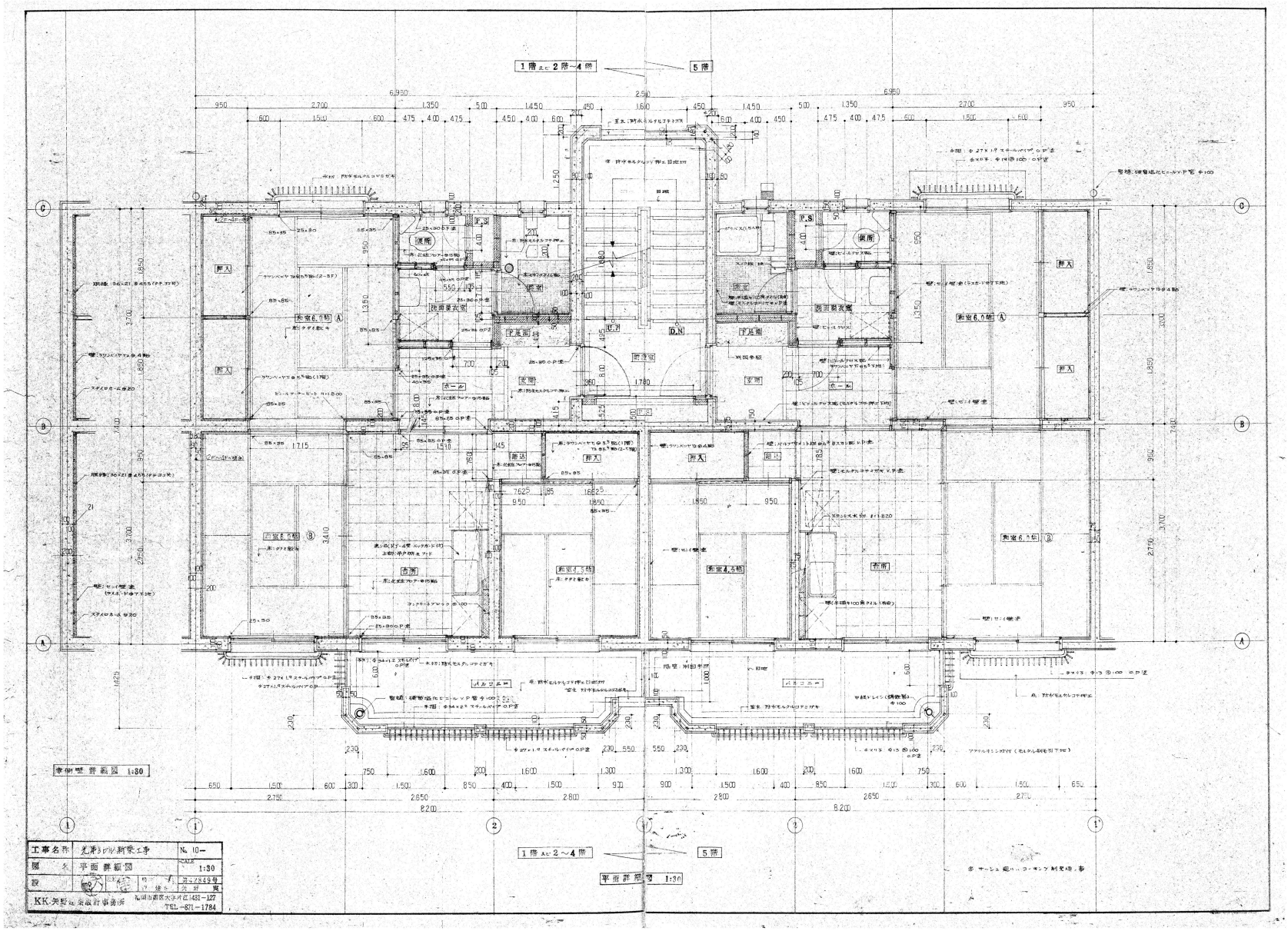
Appendices

Appendix I – As-Built Floor Plan

Appendix II – Site Images

Appendix III – Octree Subdivisions

Appendix I – As-Built Floor Plan


1階及び2階～4階
5階
便所
洗面脱衣室
和室6.0帖
和室6.5帖
和室4.5帖
和室4.5帖
浴室
玄関
ホール
押入
廊下
バルコニー
平面詳細図 1:30
1階及び2～4階
5階
平面詳細図 1:30
工事名称 尼崎市民病院新築工事
No. 10
図名 平面詳細図
1:30
KK.矢野建築設計事務所
尼崎市潮江大字西1431-127
TEL-871-1784

Appendix II – Site Images

External

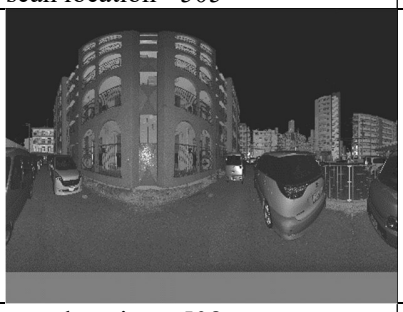
scan location - 501

scan location - 502

scan location - 503

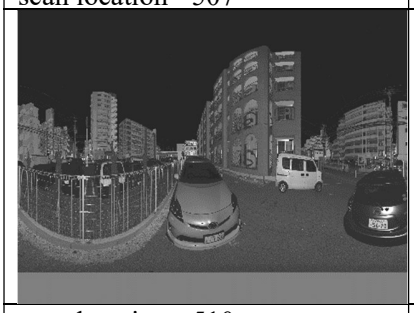
scan location - 504

scan location - 505

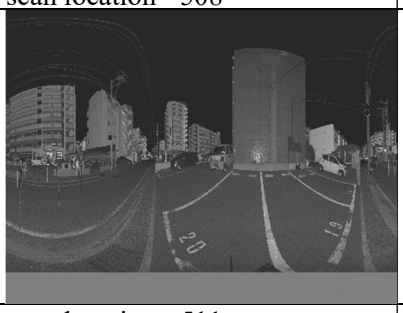


scan location - 506

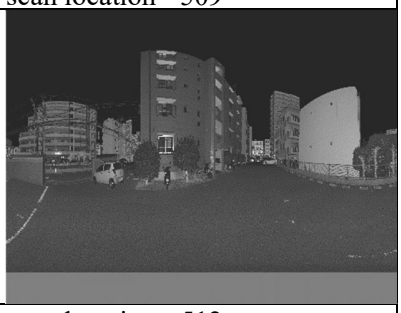
scan location - 507



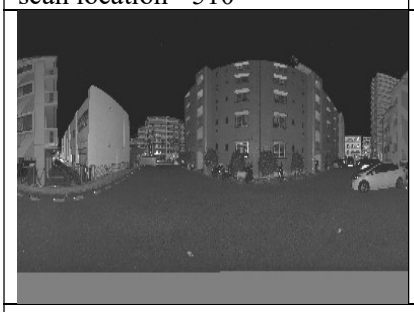
scan location - 508



scan location - 509

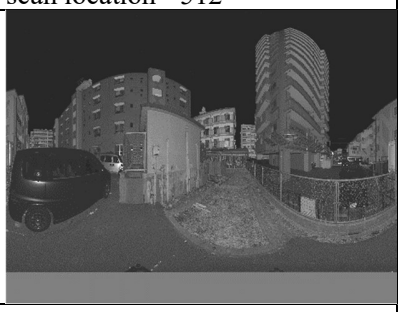


scan location - 510



scan location - 511

scan location - 512



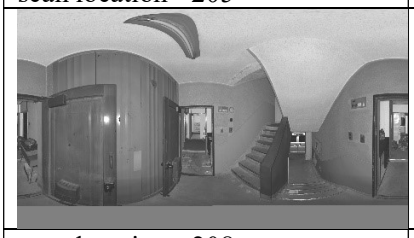
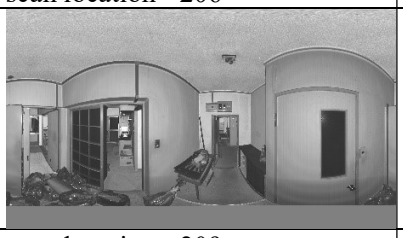
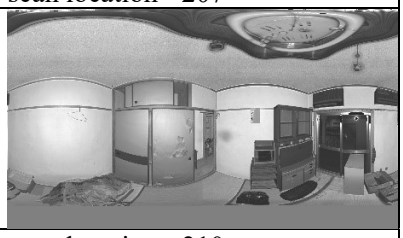
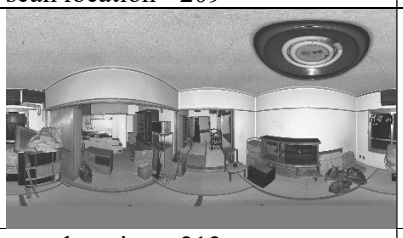
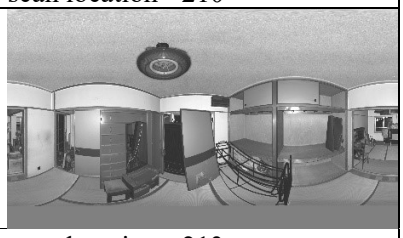
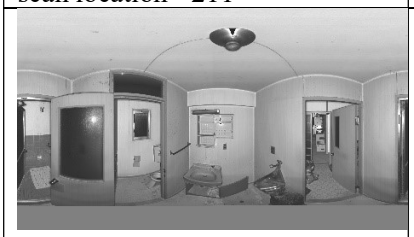
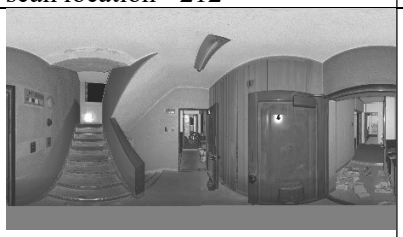
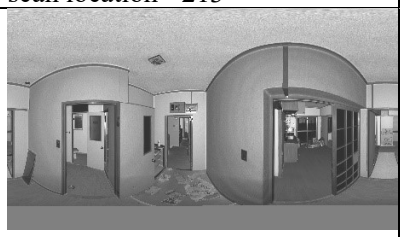
Internal

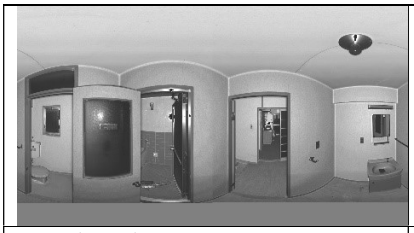
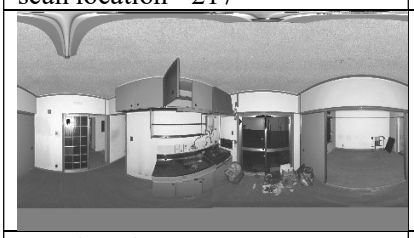
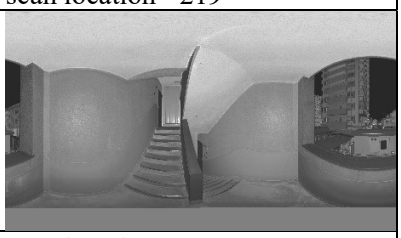
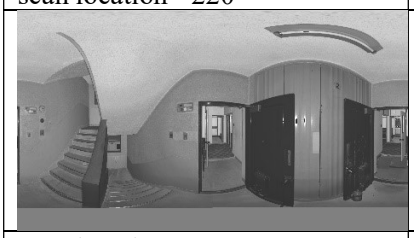
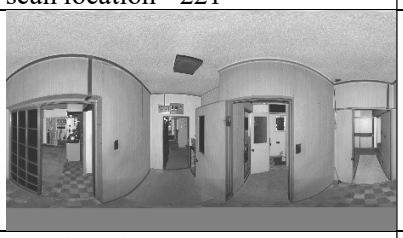
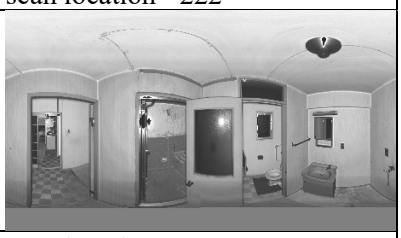
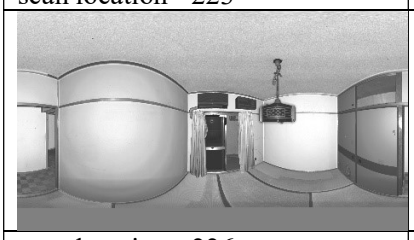
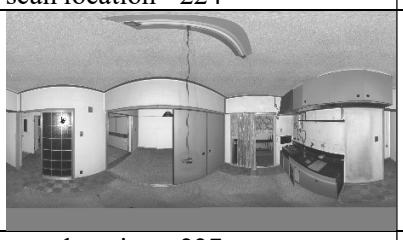
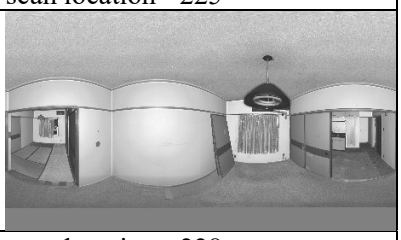
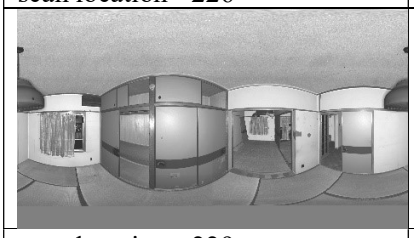
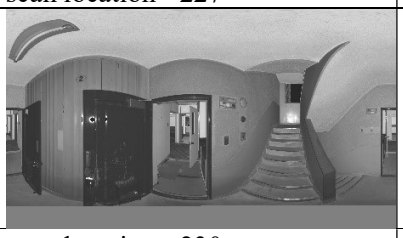
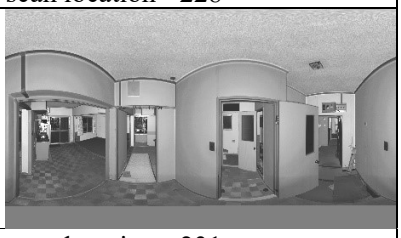
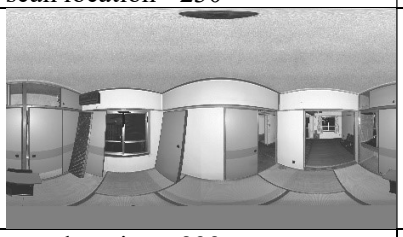
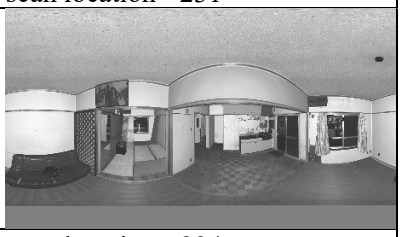
scan location - 3

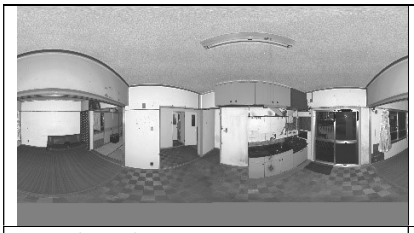
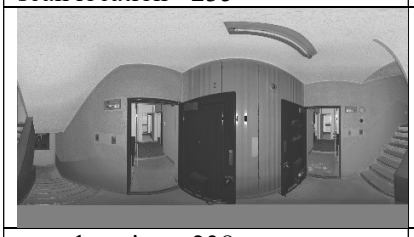
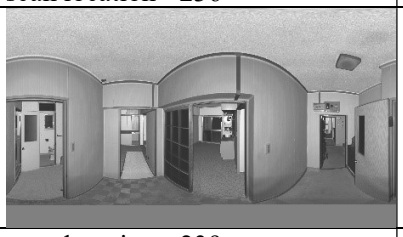
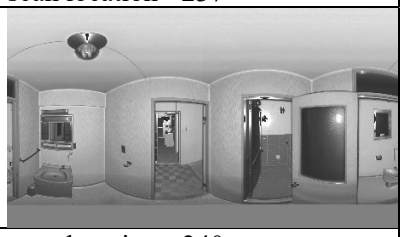
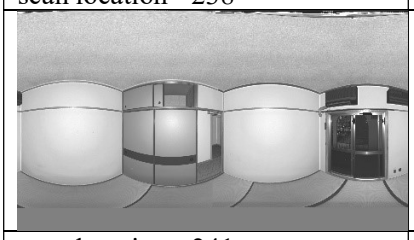
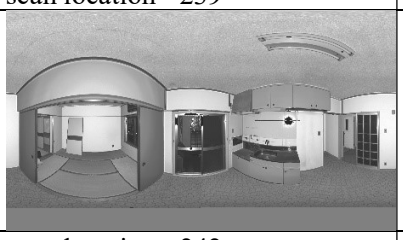
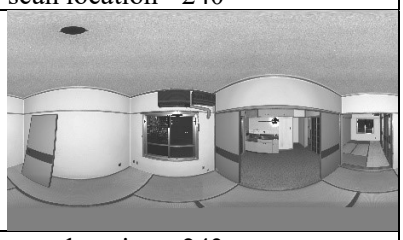
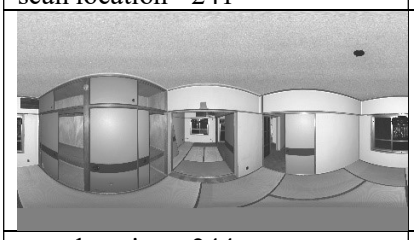
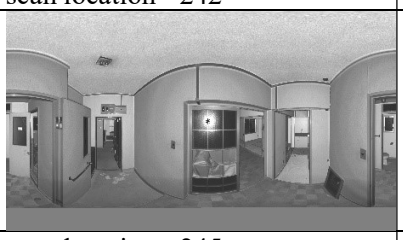
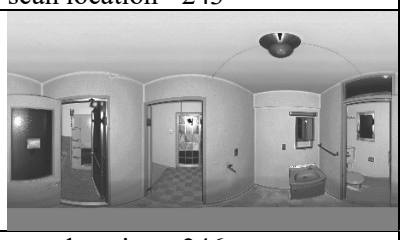
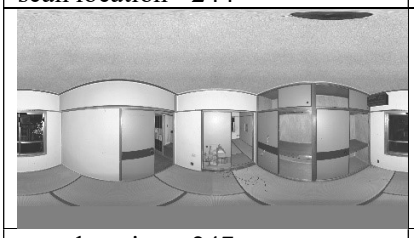
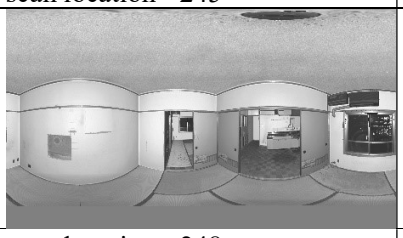
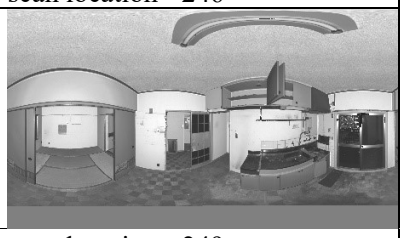
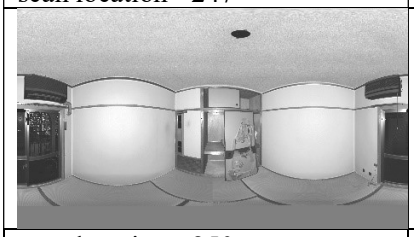
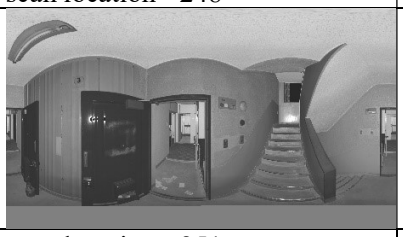
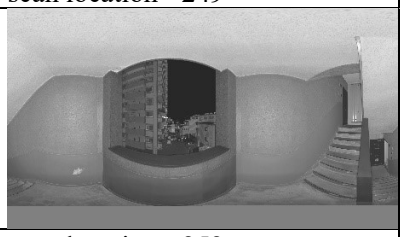
scan location - 4

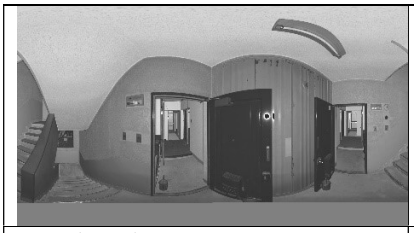
scan location - 5

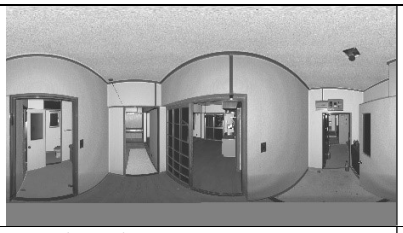
scan location - 6	scan location - 7	scan location - 8
		
scan location - 9	scan location - 10	scan location - 11
		
scan location - 12	scan location - 13	scan location - 14
		
scan location - 15	scan location - 16	scan location - 17
		
scan location - 18	scan location - 19	scan location - 20
		
scan location - 21	scan location - 22	scan location - 23
		
scan location - 24	scan location - 25	scan location - 26

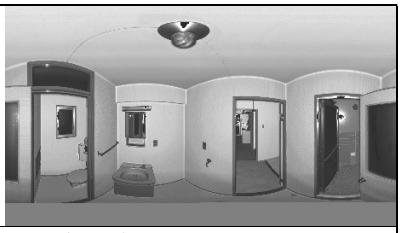
		
scan location - 27	scan location - 28	scan location - 29
		
scan location - 30	scan location - 31	scan location - 32
		
scan location - 205	scan location - 206	scan location - 207
		
scan location - 208	scan location - 209	scan location - 210
		
scan location - 211	scan location - 212	scan location - 213
		
scan location - 214	scan location - 215	scan location - 216

		
scan location - 217	scan location - 218	scan location - 219
		
scan location - 220	scan location - 221	scan location - 222
		
scan location - 223	scan location - 224	scan location - 225
		
scan location - 226	scan location - 227	scan location - 228
		
scan location - 229	scan location - 230	scan location - 231
		
scan location - 232	scan location - 233	scan location - 234

		
scan location - 235	scan location - 236	scan location - 237
		
scan location - 238	scan location - 239	scan location - 240
		
scan location - 241	scan location - 242	scan location - 243
		
scan location - 244	scan location - 245	scan location - 246
		
scan location - 247	scan location - 248	scan location - 249
		
scan location - 250	scan location - 251	scan location - 252


scan location - 253

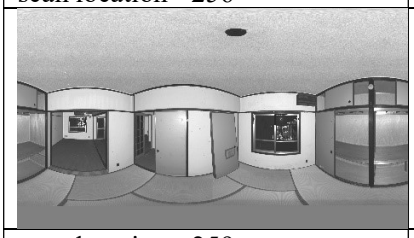

scan location - 254

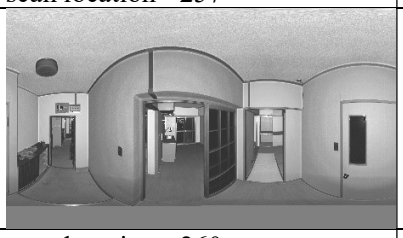

scan location - 255

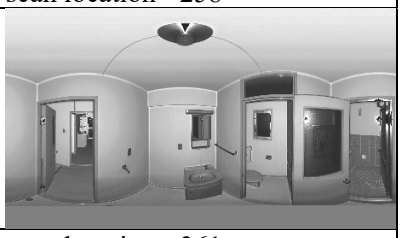

scan location - 256

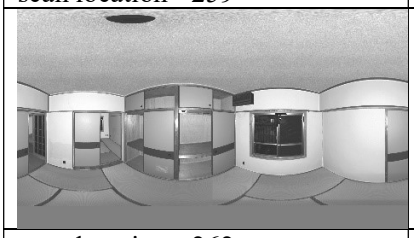

scan location - 257

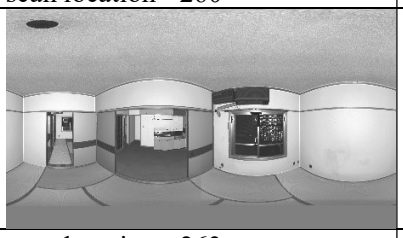

scan location - 258

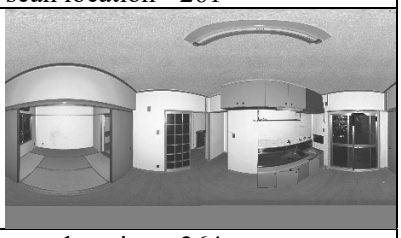

scan location - 259

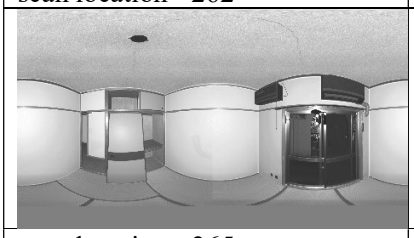

scan location - 260

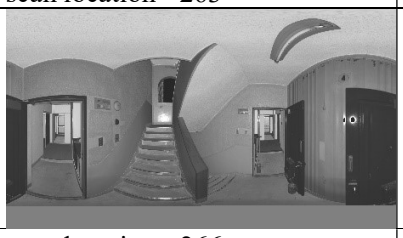

scan location - 261

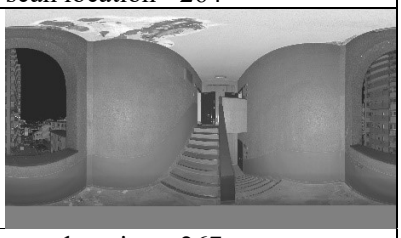

scan location - 262

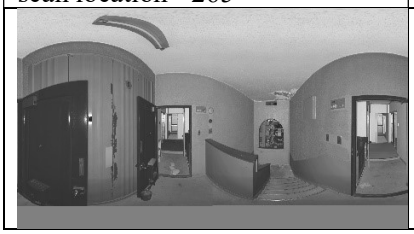

scan location - 263

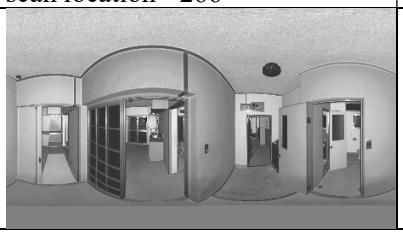

scan location - 264


scan location - 265


scan location - 266


scan location - 267





Appendix III - Octree Subdivisions

OCTREE.TRAVERSE(F_traverse)

```
0: Internal node at depth 0 has 8 children and 30129150 points ([-6.9859317  29.94105975  6.07359346])
  0: Internal node at depth 1 has 8 children and 5029004 points ([-6.9859317  29.94105975  6.07359346])
    0: Internal node at depth 2 has 4 children and 324915 points ([-6.9859317  29.94105975  6.07359346])
      1: Internal node at depth 3 has 8 children and 60815 points ([-6.8596817  29.94105975  6.07359346])
        0: Internal node at depth 4 has 1 children and 1406 points ([-6.8596817  29.94105975  6.07359346])
          7: Internal node at depth 5 has 2 children and 1406 points ([-6.8281192  29.97262225  6.10515596])
            5: Leaf node at depth 6 has 645 points with origin [-6.81233795 29.97262225  6.12093721]
            7: Leaf node at depth 6 has 761 points with origin [-6.81233795 29.9884035   6.12093721]
        1: Internal node at depth 4 has 2 children and 2693 points ([-6.7965567  29.94105975  6.07359346])
          6: Internal node at depth 5 has 6 children and 1124 points ([-6.7965567  29.97262225  6.10515596])
            0: Leaf node at depth 6 has 22 points with origin [-6.7965567  29.97262225  6.10515596]
            1: Leaf node at depth 6 has 4 points with origin [-6.78077545 29.97262225  6.10515596]
            4: Leaf node at depth 6 has 138 points with origin [-6.7965567  29.97262225  6.12093721]
            5: Leaf node at depth 6 has 7 points with origin [-6.78077545 29.97262225  6.12093721]
            6: Leaf node at depth 6 has 363 points with origin [-6.7965567  29.9884035   6.12093721]
            7: Leaf node at depth 6 has 590 points with origin [-6.78077545 29.9884035   6.12093721]
          7: Internal node at depth 5 has 4 children and 1569 points ([-6.7649942  29.97262225  6.10515596])
            4: Leaf node at depth 6 has 56 points with origin [-6.7649942  29.97262225  6.12093721]
            5: Leaf node at depth 6 has 167 points with origin [-6.74921295 29.97262225  6.12093721]
            6: Leaf node at depth 6 has 508 points with origin [-6.7649942  29.9884035   6.12093721]
            7: Leaf node at depth 6 has 838 points with origin [-6.74921295 29.9884035   6.12093721]
        2: Internal node at depth 4 has 2 children and 3118 points ([-6.8596817  30.00418475  6.07359346])
          5: Internal node at depth 5 has 2 children and 1463 points ([-6.8281192  30.00418475  6.10515596])
            5: Leaf node at depth 6 has 716 points with origin [-6.81233795 30.00418475  6.12093721]
            7: Leaf node at depth 6 has 747 points with origin [-6.81233795 30.019966    6.12093721]
          7: Internal node at depth 5 has 2 children and 1655 points ([-6.8281192  30.03574725  6.10515596])
            5: Leaf node at depth 6 has 822 points with origin [-6.81233795 30.03574725  6.12093721]
            7: Leaf node at depth 6 has 833 points with origin [-6.81233795 30.0515285   6.12093721]
        3: Internal node at depth 4 has 4 children and 9599 points ([-6.7965567  30.00418475  6.07359346])
          4: Internal node at depth 5 has 4 children and 2729 points ([-6.7965567  30.00418475  6.10515596])
            4: Leaf node at depth 6 has 344 points with origin [-6.7965567  30.00418475  6.12093721]
            5: Leaf node at depth 6 has 740 points with origin [-6.78077545 30.00418475  6.12093721]
            6: Leaf node at depth 6 has 567 points with origin [-6.7965567  30.019966    6.12093721]
            7: Leaf node at depth 6 has 1078 points with origin [-6.78077545 30.019966    6.12093721]
          5: Internal node at depth 5 has 4 children and 731 points ([-6.7649942  30.00418475  6.10515596])
            4: Leaf node at depth 6 has 220 points with origin [-6.7649942  30.00418475  6.12093721]
            5: Leaf node at depth 6 has 53 points with origin [-6.74921295 30.00418475  6.12093721]
            6: Leaf node at depth 6 has 455 points with origin [-6.7649942  30.019966    6.12093721]
            7: Leaf node at depth 6 has 3 points with origin [-6.74921295 30.019966    6.12093721]
          6: Internal node at depth 5 has 4 children and 2849 points ([-6.7965567  30.03574725  6.10515596])
            4: Leaf node at depth 6 has 691 points with origin [-6.7965567  30.03574725  6.12093721]
            5: Leaf node at depth 6 has 746 points with origin [-6.78077545 30.03574725  6.12093721]
            6: Leaf node at depth 6 has 708 points with origin [-6.7965567  30.0515285   6.12093721]
            7: Leaf node at depth 6 has 704 points with origin [-6.78077545 30.0515285   6.12093721]
          7: Internal node at depth 5 has 4 children and 3290 points ([-6.7649942  30.03574725  6.10515596])
            4: Leaf node at depth 6 has 811 points with origin [-6.7649942  30.03574725  6.12093721]
            5: Leaf node at depth 6 has 371 points with origin [-6.74921295 30.03574725  6.12093721]
            6: Leaf node at depth 6 has 707 points with origin [-6.7649942  30.0515285   6.12093721]
            7: Leaf node at depth 6 has 1401 points with origin [-6.74921295 30.0515285   6.12093721]
        4: Internal node at depth 4 has 3 children and 6009 points ([-6.8596817  29.94105975  6.13671846])
          3: Internal node at depth 5 has 4 children and 2880 points ([-6.8281192  29.97262225  6.13671846])
            1: Leaf node at depth 6 has 723 points with origin [-6.81233795 29.97262225  6.13671846]
            3: Leaf node at depth 6 has 705 points with origin [-6.81233795 29.9884035   6.13671846]
            5: Leaf node at depth 6 has 726 points with origin [-6.81233795 29.97262225  6.15249971]
            7: Leaf node at depth 6 has 726 points with origin [-6.81233795 29.9884035   6.15249971]
```

5: Internal node at depth 5 has 1 children and 18 points ([-6.8281192 29.94105975 6.16828096])
7: Internal node at depth 5 has 4 children and 3111 points ([-6.8281192 29.97262225 6.16828096])
1: Leaf node at depth 6 has 840 points with origin [-6.81233795 29.97262225 6.16828096]
3: Leaf node at depth 6 has 738 points with origin [-6.81233795 29.9884035   6.16828096]
5: Leaf node at depth 6 has 809 points with origin [-6.81233795 29.97262225 6.18406221]
7: Leaf node at depth 6 has 724 points with origin [-6.81233795 29.9884035   6.18406221]
5: Internal node at depth 4 has 7 children and 17646 points ([-6.7965567 29.94105975 6.13671846])
1: Internal node at depth 5 has 3 children and 85 points ([-6.7649942 29.94105975 6.13671846])
0: Leaf node at depth 6 has 1 points with origin [-6.7649942 29.94105975 6.13671846]
4: Leaf node at depth 6 has 26 points with origin [-6.7649942 29.94105975 6.15249971]
5: Leaf node at depth 6 has 58 points with origin [-6.74921295 29.94105975 6.15249971]
2: Internal node at depth 5 has 7 children and 2385 points ([-6.7965567 29.97262225 6.13671846])
0: Leaf node at depth 6 has 341 points with origin [-6.7965567 29.97262225 6.13671846]
2: Leaf node at depth 6 has 46 points with origin [-6.7965567 29.9884035   6.13671846]
3: Leaf node at depth 6 has 780 points with origin [-6.78077545 29.9884035   6.13671846]
4: Leaf node at depth 6 has 469 points with origin [-6.7965567 29.97262225 6.15249971]
5: Leaf node at depth 6 has 161 points with origin [-6.78077545 29.97262225 6.15249971]
6: Leaf node at depth 6 has 8 points with origin [-6.7965567 29.9884035   6.15249971]
7: Leaf node at depth 6 has 580 points with origin [-6.78077545 29.9884035   6.15249971]
3: Internal node at depth 5 has 8 children and 3772 points ([-6.7649942 29.97262225 6.13671846])
0: Leaf node at depth 6 has 29 points with origin [-6.7649942 29.97262225 6.13671846]
1: Leaf node at depth 6 has 538 points with origin [-6.74921295 29.97262225 6.13671846]
2: Leaf node at depth 6 has 787 points with origin [-6.7649942 29.9884035   6.13671846]
3: Leaf node at depth 6 has 483 points with origin [-6.74921295 29.9884035   6.13671846]
4: Leaf node at depth 6 has 419 points with origin [-6.7649942 29.97262225 6.15249971]
5: Leaf node at depth 6 has 1023 points with origin [-6.74921295 29.97262225 6.15249971]
6: Leaf node at depth 6 has 405 points with origin [-6.7649942 29.9884035   6.15249971]
7: Leaf node at depth 6 has 88 points with origin [-6.74921295 29.9884035   6.15249971]
4: Internal node at depth 5 has 4 children and 1473 points ([-6.7965567 29.94105975 6.16828096])
0: Leaf node at depth 6 has 18 points with origin [-6.7965567 29.94105975 6.16828096]
1: Leaf node at depth 6 has 351 points with origin [-6.78077545 29.94105975 6.16828096]
4: Leaf node at depth 6 has 405 points with origin [-6.7965567 29.94105975 6.18406221]
5: Leaf node at depth 6 has 699 points with origin [-6.78077545 29.94105975 6.18406221]
5: Internal node at depth 5 has 4 children and 2891 points ([-6.7649942 29.94105975 6.16828096])
0: Leaf node at depth 6 has 572 points with origin [-6.7649942 29.94105975 6.16828096]
1: Leaf node at depth 6 has 744 points with origin [-6.74921295 29.94105975 6.16828096]
4: Leaf node at depth 6 has 770 points with origin [-6.7649942 29.94105975 6.18406221]
5: Leaf node at depth 6 has 805 points with origin [-6.74921295 29.94105975 6.18406221]
6: Internal node at depth 5 has 4 children and 3242 points ([-6.7965567 29.97262225 6.16828096])
0: Leaf node at depth 6 has 776 points with origin [-6.7965567 29.97262225 6.16828096]
1: Leaf node at depth 6 has 971 points with origin [-6.78077545 29.97262225 6.16828096]
4: Leaf node at depth 6 has 743 points with origin [-6.7965567 29.97262225 6.18406221]
5: Leaf node at depth 6 has 752 points with origin [-6.78077545 29.97262225 6.18406221]
7: Internal node at depth 5 has 4 children and 3798 points ([-6.7649942 29.97262225 6.16828096])
0: Leaf node at depth 6 has 1042 points with origin [-6.7649942 29.97262225 6.16828096]
1: Leaf node at depth 6 has 1154 points with origin [-6.74921295 29.97262225 6.16828096]
4: Leaf node at depth 6 has 712 points with origin [-6.7649942 29.97262225 6.18406221]
5: Leaf node at depth 6 has 890 points with origin [-6.74921295 29.97262225 6.18406221]
6: Internal node at depth 4 has 4 children and 16944 points ([-6.8596817 30.00418475 6.13671846])
1: Internal node at depth 5 has 4 children and 2809 points ([-6.8281192 30.00418475 6.13671846])
1: Leaf node at depth 6 has 718 points with origin [-6.81233795 30.00418475 6.13671846]
3: Leaf node at depth 6 has 695 points with origin [-6.81233795 30.019966   6.13671846]
5: Leaf node at depth 6 has 691 points with origin [-6.81233795 30.00418475 6.15249971]
7: Leaf node at depth 6 has 705 points with origin [-6.81233795 30.019966   6.15249971]
3: Internal node at depth 5 has 4 children and 3292 points ([-6.8281192 30.03574725 6.13671846])
1: Leaf node at depth 6 has 855 points with origin [-6.81233795 30.03574725 6.13671846]

```
        3: Leaf node at depth 6 has 906 points with origin [-6.81233795 30.0515285   6.13671846]
        5: Leaf node at depth 6 has 775 points with origin [-6.81233795 30.03574725  6.15249971]
        7: Leaf node at depth 6 has 756 points with origin [-6.81233795 30.0515285   6.15249971]
    5: Internal node at depth 5 has 6 children and 4713 points ([-6.8281192  30.00418475  6.16828096])
        1: Leaf node at depth 6 has 650 points with origin [-6.81233795 30.00418475  6.16828096]
        2: Leaf node at depth 6 has 451 points with origin [-6.8281192  30.019966    6.16828096]
        3: Leaf node at depth 6 has 1322 points with origin [-6.81233795 30.019966    6.16828096]
        5: Leaf node at depth 6 has 674 points with origin [-6.81233795 30.00418475  6.18406221]
        6: Leaf node at depth 6 has 283 points with origin [-6.8281192  30.019966    6.18406221]
        7: Leaf node at depth 6 has 1333 points with origin [-6.81233795 30.019966    6.18406221]
    7: Internal node at depth 5 has 6 children and 6130 points ([-6.8281192  30.03574725  6.16828096])
        0: Leaf node at depth 6 has 379 points with origin [-6.8281192  30.03574725  6.16828096]
        1: Leaf node at depth 6 has 1571 points with origin [-6.81233795 30.03574725  6.16828096]
        2: Leaf node at depth 6 has 427 points with origin [-6.8281192  30.0515285   6.16828096]
        3: Leaf node at depth 6 has 1590 points with origin [-6.81233795 30.0515285   6.16828096]
        5: Leaf node at depth 6 has 1143 points with origin [-6.81233795 30.03574725  6.18406221]
        7: Leaf node at depth 6 has 1020 points with origin [-6.81233795 30.0515285   6.18406221]
  7: Internal node at depth 4 has 2 children and 3400 points ([-6.7965567  30.00418475  6.13671846])
    1: Internal node at depth 5 has 4 children and 1170 points ([-6.7649942  30.00418475  6.13671846])
        1: Leaf node at depth 6 has 35 points with origin [-6.74921295 30.00418475  6.13671846]
        2: Leaf node at depth 6 has 36 points with origin [-6.7649942  30.019966    6.13671846]
        3: Leaf node at depth 6 has 324 points with origin [-6.74921295 30.019966    6.13671846]
        7: Leaf node at depth 6 has 775 points with origin [-6.74921295 30.019966    6.15249971]
    3: Internal node at depth 5 has 4 children and 2230 points ([-6.7649942  30.03574725  6.13671846])
        0: Leaf node at depth 6 has 196 points with origin [-6.7649942  30.03574725  6.13671846]
        1: Leaf node at depth 6 has 1739 points with origin [-6.74921295 30.03574725  6.13671846]
        3: Leaf node at depth 6 has 157 points with origin [-6.74921295 30.0515285   6.13671846]
        5: Leaf node at depth 6 has 138 points with origin [-6.74921295 30.03574725  6.15249971]
3: Internal node at depth 3 has 8 children and 56934 points ([-6.8596817  30.06730975  6.07359346])
  0: Internal node at depth 4 has 1 children and 805 points ([-6.8596817  30.06730975  6.07359346])
    5: Internal node at depth 5 has 2 children and 805 points ([-6.8281192  30.06730975  6.10515596])
        5: Leaf node at depth 6 has 632 points with origin [-6.81233795 30.06730975  6.12093721]
        7: Leaf node at depth 6 has 173 points with origin [-6.81233795 30.083091    6.12093721]
  1: Internal node at depth 4 has 4 children and 10515 points ([-6.7965567  30.06730975  6.07359346])
    4: Internal node at depth 5 has 5 children and 2350 points ([-6.7965567  30.06730975  6.10515596])
        2: Leaf node at depth 6 has 2 points with origin [-6.7965567  30.083091    6.10515596]
        4: Leaf node at depth 6 has 703 points with origin [-6.7965567  30.06730975  6.12093721]
        5: Leaf node at depth 6 has 740 points with origin [-6.78077545 30.06730975  6.12093721]
        6: Leaf node at depth 6 has 373 points with origin [-6.7965567  30.083091    6.12093721]
        7: Leaf node at depth 6 has 532 points with origin [-6.78077545 30.083091    6.12093721]
    5: Internal node at depth 5 has 4 children and 3089 points ([-6.7649942  30.06730975  6.10515596])
        4: Leaf node at depth 6 has 680 points with origin [-6.7649942  30.06730975  6.12093721]
        5: Leaf node at depth 6 has 737 points with origin [-6.74921295 30.06730975  6.12093721]
        6: Leaf node at depth 6 has 908 points with origin [-6.7649942  30.083091    6.12093721]
        7: Leaf node at depth 6 has 764 points with origin [-6.74921295 30.083091    6.12093721]
    6: Internal node at depth 5 has 2 children and 931 points ([-6.7965567  30.09887225  6.10515596])
        5: Leaf node at depth 6 has 358 points with origin [-6.78077545 30.09887225  6.12093721]
        7: Leaf node at depth 6 has 573 points with origin [-6.78077545 30.1146535   6.12093721]
    7: Internal node at depth 5 has 4 children and 4145 points ([-6.7649942  30.09887225  6.10515596])
        4: Leaf node at depth 6 has 1558 points with origin [-6.7649942  30.09887225  6.12093721]
        5: Leaf node at depth 6 has 736 points with origin [-6.74921295 30.09887225  6.12093721]
        6: Leaf node at depth 6 has 1076 points with origin [-6.7649942  30.1146535   6.12093721]
        7: Leaf node at depth 6 has 775 points with origin [-6.74921295 30.1146535   6.12093721]
  2: Internal node at depth 4 has 2 children and 29 points ([-6.8596817  30.13043475  6.07359346])
  3: Internal node at depth 4 has 2 children and 6645 points ([-6.7965567  30.13043475  6.07359346])
    5: Internal node at depth 5 has 4 children and 3522 points ([-6.7649942  30.13043475  6.10515596])
```

```
    4: Leaf node at depth 6 has 647 points with origin [-6.7649942  30.13043475  6.12093721]
    5: Leaf node at depth 6 has 1124 points with origin [-6.74921295 30.13043475  6.12093721]
    6: Leaf node at depth 6 has 198 points with origin [-6.7649942  30.146216    6.12093721]
    7: Leaf node at depth 6 has 1553 points with origin [-6.74921295 30.146216    6.12093721]
  7: Internal node at depth 5 has 3 children and 3123 points ([-6.7649942  30.16199725  6.10515596])
    5: Leaf node at depth 6 has 1564 points with origin [-6.74921295 30.16199725  6.12093721]
    6: Leaf node at depth 6 has 13 points with origin [-6.7649942  30.1777785   6.12093721]
    7: Leaf node at depth 6 has 1546 points with origin [-6.74921295 30.1777785   6.12093721]
4: Internal node at depth 4 has 4 children and 8434 points ([-6.8596817  30.06730975  6.13671846])
  1: Internal node at depth 5 has 3 children and 976 points ([-6.8281192  30.06730975  6.13671846])
    1: Leaf node at depth 6 has 498 points with origin [-6.81233795 30.06730975  6.13671846]
    3: Leaf node at depth 6 has 117 points with origin [-6.81233795 30.083091    6.13671846]
    5: Leaf node at depth 6 has 361 points with origin [-6.81233795 30.06730975  6.15249971]
  3: Internal node at depth 5 has 2 children and 3 points ([-6.8281192  30.09887225  6.13671846])
  5: Internal node at depth 5 has 8 children and 3979 points ([-6.8281192  30.06730975  6.16828096])
    0: Leaf node at depth 6 has 424 points with origin [-6.8281192  30.06730975  6.16828096]
    1: Leaf node at depth 6 has 863 points with origin [-6.81233795 30.06730975  6.16828096]
    2: Leaf node at depth 6 has 622 points with origin [-6.8281192  30.083091    6.16828096]
    3: Leaf node at depth 6 has 228 points with origin [-6.81233795 30.083091    6.16828096]
    4: Leaf node at depth 6 has 16 points with origin [-6.8281192  30.06730975  6.18406221]
    5: Leaf node at depth 6 has 850 points with origin [-6.81233795 30.06730975  6.18406221]
    6: Leaf node at depth 6 has 317 points with origin [-6.8281192  30.083091    6.18406221]
    7: Leaf node at depth 6 has 659 points with origin [-6.81233795 30.083091    6.18406221]
  7: Internal node at depth 5 has 8 children and 3476 points ([-6.8281192  30.09887225  6.16828096])
    0: Leaf node at depth 6 has 606 points with origin [-6.8281192  30.09887225  6.16828096]
    1: Leaf node at depth 6 has 204 points with origin [-6.81233795 30.09887225  6.16828096]
    2: Leaf node at depth 6 has 537 points with origin [-6.8281192  30.1146535   6.16828096]
    3: Leaf node at depth 6 has 186 points with origin [-6.81233795 30.1146535   6.16828096]
    4: Leaf node at depth 6 has 129 points with origin [-6.8281192  30.09887225  6.18406221]
    5: Leaf node at depth 6 has 885 points with origin [-6.81233795 30.09887225  6.18406221]
    6: Leaf node at depth 6 has 79 points with origin [-6.8281192  30.1146535   6.18406221]
    7: Leaf node at depth 6 has 850 points with origin [-6.81233795 30.1146535   6.18406221]
5: Internal node at depth 4 has 7 children and 11203 points ([-6.7965567  30.06730975  6.13671846])
  0: Internal node at depth 5 has 2 children and 14 points ([-6.7965567  30.06730975  6.13671846])
  1: Internal node at depth 5 has 2 children and 678 points ([-6.7649942  30.06730975  6.13671846])
    2: Leaf node at depth 6 has 286 points with origin [-6.7649942  30.083091    6.13671846]
    6: Leaf node at depth 6 has 392 points with origin [-6.7649942  30.083091    6.15249971]
  2: Internal node at depth 5 has 3 children and 114 points ([-6.7965567  30.09887225  6.13671846])
    1: Leaf node at depth 6 has 12 points with origin [-6.78077545 30.09887225  6.13671846]
    3: Leaf node at depth 6 has 96 points with origin [-6.78077545 30.1146535   6.13671846]
    7: Leaf node at depth 6 has 6 points with origin [-6.78077545 30.1146535   6.15249971]
  3: Internal node at depth 5 has 5 children and 4033 points ([-6.7649942  30.09887225  6.13671846])
    0: Leaf node at depth 6 has 1250 points with origin [-6.7649942  30.09887225  6.13671846]
    2: Leaf node at depth 6 has 1185 points with origin [-6.7649942  30.1146535   6.13671846]
    4: Leaf node at depth 6 has 761 points with origin [-6.7649942  30.09887225  6.15249971]
    6: Leaf node at depth 6 has 608 points with origin [-6.7649942  30.1146535   6.15249971]
    7: Leaf node at depth 6 has 229 points with origin [-6.74921295 30.1146535   6.15249971]
  4: Internal node at depth 5 has 4 children and 186 points ([-6.7965567  30.06730975  6.16828096])
    1: Leaf node at depth 6 has 38 points with origin [-6.78077545 30.06730975  6.16828096]
    3: Leaf node at depth 6 has 74 points with origin [-6.78077545 30.083091    6.16828096]
    5: Leaf node at depth 6 has 48 points with origin [-6.78077545 30.06730975  6.18406221]
    7: Leaf node at depth 6 has 26 points with origin [-6.78077545 30.083091    6.18406221]
  5: Internal node at depth 5 has 4 children and 1763 points ([-6.7649942  30.06730975  6.16828096])
    0: Leaf node at depth 6 has 11 points with origin [-6.7649942  30.06730975  6.16828096]
    2: Leaf node at depth 6 has 892 points with origin [-6.7649942  30.083091    6.16828096]
    4: Leaf node at depth 6 has 18 points with origin [-6.7649942  30.06730975  6.18406221]
```

6: Leaf node at depth 6 has 842 points with origin [-6.7649942 30.083091 6.18406221]
7: Internal node at depth 5 has 6 children and 4415 points ([-6.7649942 30.09887225 6.16828096])
0: Leaf node at depth 6 has 1134 points with origin [-6.7649942 30.09887225 6.16828096]
2: Leaf node at depth 6 has 587 points with origin [-6.7649942 30.1146535 6.16828096]
3: Leaf node at depth 6 has 435 points with origin [-6.74921295 30.1146535 6.16828096]
4: Leaf node at depth 6 has 1035 points with origin [-6.7649942 30.09887225 6.18406221]
6: Leaf node at depth 6 has 748 points with origin [-6.7649942 30.1146535 6.18406221]
7: Leaf node at depth 6 has 476 points with origin [-6.74921295 30.1146535 6.18406221]
6: Internal node at depth 4 has 3 children and 2698 points ([-6.8596817 30.13043475 6.13671846])
1: Internal node at depth 5 has 1 children and 1 points ([-6.8281192 30.13043475 6.13671846])
5: Internal node at depth 5 has 8 children and 2675 points ([-6.8281192 30.13043475 6.16828096])
0: Leaf node at depth 6 has 517 points with origin [-6.8281192 30.13043475 6.16828096]
1: Leaf node at depth 6 has 167 points with origin [-6.81233795 30.13043475 6.16828096]
2: Leaf node at depth 6 has 460 points with origin [-6.8281192 30.146216 6.16828096]
3: Leaf node at depth 6 has 2 points with origin [-6.81233795 30.146216 6.16828096]
4: Leaf node at depth 6 has 82 points with origin [-6.8281192 30.13043475 6.18406221]
5: Leaf node at depth 6 has 829 points with origin [-6.81233795 30.13043475 6.18406221]
6: Leaf node at depth 6 has 577 points with origin [-6.8281192 30.146216 6.18406221]
7: Leaf node at depth 6 has 41 points with origin [-6.81233795 30.146216 6.18406221]
7: Internal node at depth 5 has 2 children and 22 points ([-6.8281192 30.16199725 6.16828096])
7: Internal node at depth 4 has 4 children and 16605 points ([-6.7965567 30.13043475 6.13671846])
1: Internal node at depth 5 has 7 children and 4195 points ([-6.7649942 30.13043475 6.13671846])
0: Leaf node at depth 6 has 454 points with origin [-6.7649942 30.13043475 6.13671846]
1: Leaf node at depth 6 has 486 points with origin [-6.74921295 30.13043475 6.13671846]
2: Leaf node at depth 6 has 3 points with origin [-6.7649942 30.146216 6.13671846]
3: Leaf node at depth 6 has 945 points with origin [-6.74921295 30.146216 6.13671846]
4: Leaf node at depth 6 has 6 points with origin [-6.7649942 30.13043475 6.15249971]
5: Leaf node at depth 6 has 1178 points with origin [-6.74921295 30.13043475 6.15249971]
7: Leaf node at depth 6 has 1123 points with origin [-6.74921295 30.146216 6.15249971]
3: Internal node at depth 5 has 5 children and 3410 points ([-6.7649942 30.16199725 6.13671846])
1: Leaf node at depth 6 has 838 points with origin [-6.74921295 30.16199725 6.13671846]
2: Leaf node at depth 6 has 143 points with origin [-6.7649942 30.1777785 6.13671846]
3: Leaf node at depth 6 has 693 points with origin [-6.74921295 30.1777785 6.13671846]
5: Leaf node at depth 6 has 881 points with origin [-6.74921295 30.16199725 6.15249971]
7: Leaf node at depth 6 has 855 points with origin [-6.74921295 30.1777785 6.15249971]
5: Internal node at depth 5 has 4 children and 4642 points ([-6.7649942 30.13043475 6.16828096])
1: Leaf node at depth 6 has 1141 points with origin [-6.74921295 30.13043475 6.16828096]
3: Leaf node at depth 6 has 1010 points with origin [-6.74921295 30.146216 6.16828096]
5: Leaf node at depth 6 has 1126 points with origin [-6.74921295 30.13043475 6.18406221]
7: Leaf node at depth 6 has 1365 points with origin [-6.74921295 30.146216 6.18406221]
7: Internal node at depth 5 has 6 children and 4358 points ([-6.7649942 30.16199725 6.16828096])
1: Leaf node at depth 6 has 876 points with origin [-6.74921295 30.16199725 6.16828096]
2: Leaf node at depth 6 has 26 points with origin [-6.7649942 30.1777785 6.16828096]
3: Leaf node at depth 6 has 1003 points with origin [-6.74921295 30.1777785 6.16828096]
5: Leaf node at depth 6 has 1316 points with origin [-6.74921295 30.16199725 6.18406221]
6: Leaf node at depth 6 has 151 points with origin [-6.7649942 30.1777785 6.18406221]
7: Leaf node at depth 6 has 986 points with origin [-6.74921295 30.1777785 6.18406221]
5: Internal node at depth 3 has 8 children and 95270 points ([-6.8596817 29.94105975 6.19984346])
0: Internal node at depth 4 has 4 children and 7462 points ([-6.8596817 29.94105975 6.19984346])
1: Internal node at depth 5 has 2 children and 479 points ([-6.8281192 29.94105975 6.19984346])
1: Leaf node at depth 6 has 224 points with origin [-6.81233795 29.94105975 6.19984346]
5: Leaf node at depth 6 has 255 points with origin [-6.81233795 29.94105975 6.21562471]
3: Internal node at depth 5 has 4 children and 3223 points ([-6.8281192 29.97262225 6.19984346])
1: Leaf node at depth 6 has 799 points with origin [-6.81233795 29.97262225 6.19984346]
3: Leaf node at depth 6 has 807 points with origin [-6.81233795 29.9884035 6.19984346]
5: Leaf node at depth 6 has 787 points with origin [-6.81233795 29.97262225 6.21562471]

```
    7: Leaf node at depth 6 has 830 points with origin [-6.81233795 29.9884035   6.21562471]
  5: Internal node at depth 5 has 2 children and 557 points ([-6.8281192  29.94105975  6.23140596])
    1: Leaf node at depth 6 has 276 points with origin [-6.81233795 29.94105975  6.23140596]
    5: Leaf node at depth 6 has 281 points with origin [-6.81233795 29.94105975  6.24718721]
  7: Internal node at depth 5 has 4 children and 3203 points ([-6.8281192  29.97262225  6.23140596])
    1: Leaf node at depth 6 has 794 points with origin [-6.81233795 29.97262225  6.23140596]
    3: Leaf node at depth 6 has 816 points with origin [-6.81233795 29.9884035   6.23140596]
    5: Leaf node at depth 6 has 877 points with origin [-6.81233795 29.97262225  6.24718721]
    7: Leaf node at depth 6 has 716 points with origin [-6.81233795 29.9884035   6.24718721]
1: Internal node at depth 4 has 8 children and 23433 points ([-6.7965567  29.94105975  6.19984346])
  0: Internal node at depth 5 has 4 children and 2706 points ([-6.7965567  29.94105975  6.19984346])
    0: Leaf node at depth 6 has 683 points with origin [-6.7965567  29.94105975  6.19984346]
    1: Leaf node at depth 6 has 639 points with origin [-6.78077545 29.94105975  6.19984346]
    4: Leaf node at depth 6 has 725 points with origin [-6.7965567  29.94105975  6.21562471]
    5: Leaf node at depth 6 has 659 points with origin [-6.78077545 29.94105975  6.21562471]
  1: Internal node at depth 5 has 4 children and 3349 points ([-6.7649942  29.94105975  6.19984346])
    0: Leaf node at depth 6 has 791 points with origin [-6.7649942  29.94105975  6.19984346]
    1: Leaf node at depth 6 has 788 points with origin [-6.74921295 29.94105975  6.19984346]
    4: Leaf node at depth 6 has 733 points with origin [-6.7649942  29.94105975  6.21562471]
    5: Leaf node at depth 6 has 1037 points with origin [-6.74921295 29.94105975  6.21562471]
  2: Internal node at depth 5 has 4 children and 2893 points ([-6.7965567  29.97262225  6.19984346])
    0: Leaf node at depth 6 has 708 points with origin [-6.7965567  29.97262225  6.19984346]
    1: Leaf node at depth 6 has 737 points with origin [-6.78077545 29.97262225  6.19984346]
    4: Leaf node at depth 6 has 736 points with origin [-6.7965567  29.97262225  6.21562471]
    5: Leaf node at depth 6 has 712 points with origin [-6.78077545 29.97262225  6.21562471]
  3: Internal node at depth 5 has 4 children and 2836 points ([-6.7649942  29.97262225  6.19984346])
    0: Leaf node at depth 6 has 692 points with origin [-6.7649942  29.97262225  6.19984346]
    1: Leaf node at depth 6 has 685 points with origin [-6.74921295 29.97262225  6.19984346]
    4: Leaf node at depth 6 has 733 points with origin [-6.7649942  29.97262225  6.21562471]
    5: Leaf node at depth 6 has 726 points with origin [-6.74921295 29.97262225  6.21562471]
  4: Internal node at depth 5 has 4 children and 2791 points ([-6.7965567  29.94105975  6.23140596])
    0: Leaf node at depth 6 has 715 points with origin [-6.7965567  29.94105975  6.23140596]
    1: Leaf node at depth 6 has 669 points with origin [-6.78077545 29.94105975  6.23140596]
    4: Leaf node at depth 6 has 725 points with origin [-6.7965567  29.94105975  6.24718721]
    5: Leaf node at depth 6 has 682 points with origin [-6.78077545 29.94105975  6.24718721]
  5: Internal node at depth 5 has 4 children and 2752 points ([-6.7649942  29.94105975  6.23140596])
    0: Leaf node at depth 6 has 624 points with origin [-6.7649942  29.94105975  6.23140596]
    1: Leaf node at depth 6 has 705 points with origin [-6.74921295 29.94105975  6.23140596]
    4: Leaf node at depth 6 has 685 points with origin [-6.7649942  29.94105975  6.24718721]
    5: Leaf node at depth 6 has 738 points with origin [-6.74921295 29.94105975  6.24718721]
  6: Internal node at depth 5 has 4 children and 2891 points ([-6.7965567  29.97262225  6.23140596])
    0: Leaf node at depth 6 has 684 points with origin [-6.7965567  29.97262225  6.23140596]
    1: Leaf node at depth 6 has 697 points with origin [-6.78077545 29.97262225  6.23140596]
    4: Leaf node at depth 6 has 725 points with origin [-6.7965567  29.97262225  6.24718721]
    5: Leaf node at depth 6 has 785 points with origin [-6.78077545 29.97262225  6.24718721]
  7: Internal node at depth 5 has 4 children and 3215 points ([-6.7649942  29.97262225  6.23140596])
    0: Leaf node at depth 6 has 725 points with origin [-6.7649942  29.97262225  6.23140596]
    1: Leaf node at depth 6 has 732 points with origin [-6.74921295 29.97262225  6.23140596]
    4: Leaf node at depth 6 has 877 points with origin [-6.7649942  29.97262225  6.24718721]
    5: Leaf node at depth 6 has 881 points with origin [-6.74921295 29.97262225  6.24718721]
2: Internal node at depth 4 has 4 children and 17083 points ([-6.8596817  30.00418475  6.19984346])
  1: Internal node at depth 5 has 6 children and 4594 points ([-6.8281192  30.00418475  6.19984346])
    1: Leaf node at depth 6 has 679 points with origin [-6.81233795 30.00418475  6.19984346]
    2: Leaf node at depth 6 has 343 points with origin [-6.8281192  30.019966    6.19984346]
    3: Leaf node at depth 6 has 1356 points with origin [-6.81233795 30.019966    6.19984346]
    5: Leaf node at depth 6 has 690 points with origin [-6.81233795 30.00418475  6.21562471]
```

6: Leaf node at depth 6 has 321 points with origin [-6.8281192 30.019966   6.21562471]
7: Leaf node at depth 6 has 1205 points with origin [-6.81233795 30.019966   6.21562471]
3: Internal node at depth 5 has 8 children and 5047 points ([-6.8281192 30.03574725 6.19984346])
0: Leaf node at depth 6 has 17 points with origin [-6.8281192 30.03574725 6.19984346]
1: Leaf node at depth 6 has 1243 points with origin [-6.81233795 30.03574725 6.19984346]
2: Leaf node at depth 6 has 109 points with origin [-6.8281192 30.0515285   6.19984346]
3: Leaf node at depth 6 has 1244 points with origin [-6.81233795 30.0515285   6.19984346]
4: Leaf node at depth 6 has 246 points with origin [-6.8281192 30.03574725 6.21562471]
5: Leaf node at depth 6 has 942 points with origin [-6.81233795 30.03574725 6.21562471]
6: Leaf node at depth 6 has 177 points with origin [-6.8281192 30.0515285   6.21562471]
7: Leaf node at depth 6 has 1069 points with origin [-6.81233795 30.0515285   6.21562471]
5: Internal node at depth 5 has 6 children and 4469 points ([-6.8281192 30.00418475 6.23140596])
1: Leaf node at depth 6 has 684 points with origin [-6.81233795 30.00418475 6.23140596]
2: Leaf node at depth 6 has 324 points with origin [-6.8281192 30.019966   6.23140596]
3: Leaf node at depth 6 has 1107 points with origin [-6.81233795 30.019966   6.23140596]
5: Leaf node at depth 6 has 722 points with origin [-6.81233795 30.00418475 6.24718721]
6: Leaf node at depth 6 has 467 points with origin [-6.8281192 30.019966   6.24718721]
7: Leaf node at depth 6 has 1165 points with origin [-6.81233795 30.019966   6.24718721]
7: Internal node at depth 5 has 8 children and 2973 points ([-6.8281192 30.03574725 6.23140596])
0: Leaf node at depth 6 has 253 points with origin [-6.8281192 30.03574725 6.23140596]
1: Leaf node at depth 6 has 501 points with origin [-6.81233795 30.03574725 6.23140596]
2: Leaf node at depth 6 has 231 points with origin [-6.8281192 30.0515285   6.23140596]
3: Leaf node at depth 6 has 869 points with origin [-6.81233795 30.0515285   6.23140596]
4: Leaf node at depth 6 has 93 points with origin [-6.8281192 30.03574725 6.24718721]
5: Leaf node at depth 6 has 268 points with origin [-6.81233795 30.03574725 6.24718721]
6: Leaf node at depth 6 has 266 points with origin [-6.8281192 30.0515285   6.24718721]
7: Leaf node at depth 6 has 492 points with origin [-6.81233795 30.0515285   6.24718721]
3: Internal node at depth 4 has 3 children and 104 points ([-6.7965567 30.00418475 6.19984346])
3: Internal node at depth 5 has 2 children and 73 points ([-6.7649942 30.03574725 6.19984346])
4: Leaf node at depth 6 has 7 points with origin [-6.7649942 30.03574725 6.21562471]
6: Leaf node at depth 6 has 66 points with origin [-6.7649942 30.0515285   6.21562471]
6: Internal node at depth 5 has 1 children and 2 points ([-6.7965567 30.03574725 6.23140596])
7: Internal node at depth 5 has 3 children and 29 points ([-6.7649942 30.03574725 6.23140596])
4: Internal node at depth 4 has 4 children and 7370 points ([-6.8596817 29.94105975 6.26296846])
1: Internal node at depth 5 has 2 children and 614 points ([-6.8281192 29.94105975 6.26296846])
1: Leaf node at depth 6 has 286 points with origin [-6.81233795 29.94105975 6.26296846]
5: Leaf node at depth 6 has 328 points with origin [-6.81233795 29.94105975 6.27874971]
3: Internal node at depth 5 has 4 children and 3187 points ([-6.8281192 29.97262225 6.26296846])
1: Leaf node at depth 6 has 832 points with origin [-6.81233795 29.97262225 6.26296846]
3: Leaf node at depth 6 has 735 points with origin [-6.81233795 29.9884035   6.26296846]
5: Leaf node at depth 6 has 828 points with origin [-6.81233795 29.97262225 6.27874971]
7: Leaf node at depth 6 has 792 points with origin [-6.81233795 29.9884035   6.27874971]
5: Internal node at depth 5 has 2 children and 615 points ([-6.8281192 29.94105975 6.29453096])
1: Leaf node at depth 6 has 298 points with origin [-6.81233795 29.94105975 6.29453096]
5: Leaf node at depth 6 has 317 points with origin [-6.81233795 29.94105975 6.31031221]
7: Internal node at depth 5 has 4 children and 2954 points ([-6.8281192 29.97262225 6.29453096])
1: Leaf node at depth 6 has 779 points with origin [-6.81233795 29.97262225 6.29453096]
3: Leaf node at depth 6 has 706 points with origin [-6.81233795 29.9884035   6.29453096]
5: Leaf node at depth 6 has 791 points with origin [-6.81233795 29.97262225 6.31031221]
7: Leaf node at depth 6 has 678 points with origin [-6.81233795 29.9884035   6.31031221]
5: Internal node at depth 4 has 8 children and 24717 points ([-6.7965567 29.94105975 6.26296846])
0: Internal node at depth 5 has 4 children and 2780 points ([-6.7965567 29.94105975 6.26296846])
0: Leaf node at depth 6 has 701 points with origin [-6.7965567 29.94105975 6.26296846]
1: Leaf node at depth 6 has 704 points with origin [-6.78077545 29.94105975 6.26296846]
4: Leaf node at depth 6 has 687 points with origin [-6.7965567 29.94105975 6.27874971]
5: Leaf node at depth 6 has 688 points with origin [-6.78077545 29.94105975 6.27874971]

1: Internal node at depth 5 has 4 children and 3133 points ([-6.7649942 29.94105975 6.26296846])
0: Leaf node at depth 6 has 672 points with origin [-6.7649942 29.94105975 6.26296846]
1: Leaf node at depth 6 has 817 points with origin [-6.74921295 29.94105975 6.26296846]
4: Leaf node at depth 6 has 754 points with origin [-6.7649942 29.94105975 6.27874971]
5: Leaf node at depth 6 has 890 points with origin [-6.74921295 29.94105975 6.27874971]
2: Internal node at depth 5 has 5 children and 3318 points ([-6.7965567 29.97262225 6.26296846])
0: Leaf node at depth 6 has 734 points with origin [-6.7965567 29.97262225 6.26296846]
1: Leaf node at depth 6 has 816 points with origin [-6.78077545 29.97262225 6.26296846]
4: Leaf node at depth 6 has 769 points with origin [-6.7965567 29.97262225 6.27874971]
5: Leaf node at depth 6 has 946 points with origin [-6.78077545 29.97262225 6.27874971]
6: Leaf node at depth 6 has 53 points with origin [-6.7965567 29.9884035   6.27874971]
3: Internal node at depth 5 has 4 children and 3222 points ([-6.7649942 29.97262225 6.26296846])
0: Leaf node at depth 6 has 701 points with origin [-6.7649942 29.97262225 6.26296846]
1: Leaf node at depth 6 has 722 points with origin [-6.74921295 29.97262225 6.26296846]
4: Leaf node at depth 6 has 921 points with origin [-6.7649942 29.97262225 6.27874971]
5: Leaf node at depth 6 has 878 points with origin [-6.74921295 29.97262225 6.27874971]
4: Internal node at depth 5 has 4 children and 2658 points ([-6.7965567 29.94105975 6.29453096])
0: Leaf node at depth 6 has 654 points with origin [-6.7965567 29.94105975 6.29453096]
1: Leaf node at depth 6 has 671 points with origin [-6.78077545 29.94105975 6.29453096]
4: Leaf node at depth 6 has 692 points with origin [-6.7965567 29.94105975 6.31031221]
5: Leaf node at depth 6 has 641 points with origin [-6.78077545 29.94105975 6.31031221]
5: Internal node at depth 5 has 4 children and 2947 points ([-6.7649942 29.94105975 6.29453096])
0: Leaf node at depth 6 has 719 points with origin [-6.7649942 29.94105975 6.29453096]
1: Leaf node at depth 6 has 717 points with origin [-6.74921295 29.94105975 6.29453096]
4: Leaf node at depth 6 has 749 points with origin [-6.7649942 29.94105975 6.31031221]
5: Leaf node at depth 6 has 762 points with origin [-6.74921295 29.94105975 6.31031221]
6: Internal node at depth 5 has 4 children and 3551 points ([-6.7965567 29.97262225 6.29453096])
0: Leaf node at depth 6 has 823 points with origin [-6.7965567 29.97262225 6.29453096]
1: Leaf node at depth 6 has 1135 points with origin [-6.78077545 29.97262225 6.29453096]
4: Leaf node at depth 6 has 790 points with origin [-6.7965567 29.97262225 6.31031221]
5: Leaf node at depth 6 has 803 points with origin [-6.78077545 29.97262225 6.31031221]
7: Internal node at depth 5 has 4 children and 3108 points ([-6.7649942 29.97262225 6.29453096])
0: Leaf node at depth 6 has 892 points with origin [-6.7649942 29.97262225 6.29453096]
1: Leaf node at depth 6 has 741 points with origin [-6.74921295 29.97262225 6.29453096]
4: Leaf node at depth 6 has 760 points with origin [-6.7649942 29.97262225 6.31031221]
5: Leaf node at depth 6 has 715 points with origin [-6.74921295 29.97262225 6.31031221]
6: Internal node at depth 4 has 4 children and 14374 points ([-6.8596817 30.00418475 6.26296846])
1: Internal node at depth 5 has 6 children and 4581 points ([-6.8281192 30.00418475 6.26296846])
1: Leaf node at depth 6 has 673 points with origin [-6.81233795 30.00418475 6.26296846]
2: Leaf node at depth 6 has 460 points with origin [-6.8281192 30.019966   6.26296846]
3: Leaf node at depth 6 has 1251 points with origin [-6.81233795 30.019966   6.26296846]
5: Leaf node at depth 6 has 888 points with origin [-6.81233795 30.00418475 6.27874971]
6: Leaf node at depth 6 has 437 points with origin [-6.8281192 30.019966   6.27874971]
7: Leaf node at depth 6 has 872 points with origin [-6.81233795 30.019966   6.27874971]
3: Internal node at depth 5 has 8 children and 4837 points ([-6.8281192 30.03574725 6.26296846])
0: Leaf node at depth 6 has 212 points with origin [-6.8281192 30.03574725 6.26296846]
1: Leaf node at depth 6 has 668 points with origin [-6.81233795 30.03574725 6.26296846]
2: Leaf node at depth 6 has 248 points with origin [-6.8281192 30.0515285   6.26296846]
3: Leaf node at depth 6 has 802 points with origin [-6.81233795 30.0515285   6.26296846]
4: Leaf node at depth 6 has 477 points with origin [-6.8281192 30.03574725 6.27874971]
5: Leaf node at depth 6 has 938 points with origin [-6.81233795 30.03574725 6.27874971]
6: Leaf node at depth 6 has 546 points with origin [-6.8281192 30.0515285   6.27874971]
7: Leaf node at depth 6 has 946 points with origin [-6.81233795 30.0515285   6.27874971]
5: Internal node at depth 5 has 5 children and 2769 points ([-6.8281192 30.00418475 6.29453096])
1: Leaf node at depth 6 has 751 points with origin [-6.81233795 30.00418475 6.29453096]
2: Leaf node at depth 6 has 1 points with origin [-6.8281192 30.019966   6.29453096]

3: Leaf node at depth 6 has 683 points with origin [-6.81233795 30.019966 6.29453096]
5: Leaf node at depth 6 has 642 points with origin [-6.81233795 30.00418475 6.31031221]
7: Leaf node at depth 6 has 692 points with origin [-6.81233795 30.019966 6.31031221]
7: Internal node at depth 5 has 6 children and 2187 points ([-6.8281192 30.03574725 6.29453096])
0: Leaf node at depth 6 has 1 points with origin [-6.8281192 30.03574725 6.29453096]
1: Leaf node at depth 6 has 673 points with origin [-6.81233795 30.03574725 6.29453096]
2: Leaf node at depth 6 has 1 points with origin [-6.8281192 30.0515285 6.29453096]
3: Leaf node at depth 6 has 448 points with origin [-6.81233795 30.0515285 6.29453096]
5: Leaf node at depth 6 has 673 points with origin [-6.81233795 30.03574725 6.31031221]
7: Leaf node at depth 6 has 391 points with origin [-6.81233795 30.0515285 6.31031221]
7: Internal node at depth 4 has 4 children and 727 points ([-6.7965567 30.00418475 6.26296846])
0: Internal node at depth 5 has 2 children and 213 points ([-6.7965567 30.00418475 6.26296846])
4: Leaf node at depth 6 has 120 points with origin [-6.7965567 30.00418475 6.27874971]
6: Leaf node at depth 6 has 93 points with origin [-6.7965567 30.019966 6.27874971]
2: Internal node at depth 5 has 2 children and 277 points ([-6.7965567 30.03574725 6.26296846])
4: Leaf node at depth 6 has 105 points with origin [-6.7965567 30.03574725 6.27874971]
6: Leaf node at depth 6 has 172 points with origin [-6.7965567 30.0515285 6.27874971]
6: Internal node at depth 5 has 2 children and 226 points ([-6.7965567 30.03574725 6.29453096])
2: Leaf node at depth 6 has 132 points with origin [-6.7965567 30.0515285 6.29453096]
6: Leaf node at depth 6 has 94 points with origin [-6.7965567 30.0515285 6.31031221]
7: Internal node at depth 5 has 1 children and 11 points ([-6.7649942 30.03574725 6.29453096])
7: Internal node at depth 3 has 8 children and 111896 points ([-6.8596817 30.06730975 6.19984346])
0: Internal node at depth 4 has 4 children and 24358 points ([-6.8596817 30.06730975 6.19984346])
1: Internal node at depth 5 has 8 children and 7184 points ([-6.8281192 30.06730975 6.19984346])
0: Leaf node at depth 6 has 24 points with origin [-6.8281192 30.06730975 6.19984346]
1: Leaf node at depth 6 has 1404 points with origin [-6.81233795 30.06730975 6.19984346]
2: Leaf node at depth 6 has 546 points with origin [-6.8281192 30.083091 6.19984346]
3: Leaf node at depth 6 has 1388 points with origin [-6.81233795 30.083091 6.19984346]
4: Leaf node at depth 6 has 267 points with origin [-6.8281192 30.06730975 6.21562471]
5: Leaf node at depth 6 has 1488 points with origin [-6.81233795 30.06730975 6.21562471]
6: Leaf node at depth 6 has 619 points with origin [-6.8281192 30.083091 6.21562471]
7: Leaf node at depth 6 has 1448 points with origin [-6.81233795 30.083091 6.21562471]
3: Internal node at depth 5 has 8 children and 7305 points ([-6.8281192 30.09887225 6.19984346])
0: Leaf node at depth 6 has 583 points with origin [-6.8281192 30.09887225 6.19984346]
1: Leaf node at depth 6 has 1568 points with origin [-6.81233795 30.09887225 6.19984346]
2: Leaf node at depth 6 has 419 points with origin [-6.8281192 30.1146535 6.19984346]
3: Leaf node at depth 6 has 900 points with origin [-6.81233795 30.1146535 6.19984346]
4: Leaf node at depth 6 has 646 points with origin [-6.8281192 30.09887225 6.21562471]
5: Leaf node at depth 6 has 1711 points with origin [-6.81233795 30.09887225 6.21562471]
6: Leaf node at depth 6 has 909 points with origin [-6.8281192 30.1146535 6.21562471]
7: Leaf node at depth 6 has 569 points with origin [-6.81233795 30.1146535 6.21562471]
5: Internal node at depth 5 has 8 children and 6089 points ([-6.8281192 30.06730975 6.23140596])
0: Leaf node at depth 6 has 341 points with origin [-6.8281192 30.06730975 6.23140596]
1: Leaf node at depth 6 has 1110 points with origin [-6.81233795 30.06730975 6.23140596]
2: Leaf node at depth 6 has 754 points with origin [-6.8281192 30.083091 6.23140596]
3: Leaf node at depth 6 has 1193 points with origin [-6.81233795 30.083091 6.23140596]
4: Leaf node at depth 6 has 241 points with origin [-6.8281192 30.06730975 6.24718721]
5: Leaf node at depth 6 has 612 points with origin [-6.81233795 30.06730975 6.24718721]
6: Leaf node at depth 6 has 822 points with origin [-6.8281192 30.083091 6.24718721]
7: Leaf node at depth 6 has 1016 points with origin [-6.81233795 30.083091 6.24718721]
7: Internal node at depth 5 has 8 children and 3780 points ([-6.8281192 30.09887225 6.23140596])
0: Leaf node at depth 6 has 336 points with origin [-6.8281192 30.09887225 6.23140596]
1: Leaf node at depth 6 has 889 points with origin [-6.81233795 30.09887225 6.23140596]
2: Leaf node at depth 6 has 397 points with origin [-6.8281192 30.1146535 6.23140596]
3: Leaf node at depth 6 has 722 points with origin [-6.81233795 30.1146535 6.23140596]
4: Leaf node at depth 6 has 210 points with origin [-6.8281192 30.09887225 6.24718721]

5: Leaf node at depth 6 has 599 points with origin [-6.81233795 30.09887225 6.24718721]
6: Leaf node at depth 6 has 104 points with origin [-6.8281192 30.1146535 6.24718721]
7: Leaf node at depth 6 has 523 points with origin [-6.81233795 30.1146535 6.24718721]
1: Internal node at depth 4 has 8 children and 9374 points ([-6.7965567 30.06730975 6.19984346])
0: Internal node at depth 5 has 5 children and 1478 points ([-6.7965567 30.06730975 6.19984346])
0: Leaf node at depth 6 has 67 points with origin [-6.7965567 30.06730975 6.19984346]
1: Leaf node at depth 6 has 32 points with origin [-6.78077545 30.06730975 6.19984346]
2: Leaf node at depth 6 has 489 points with origin [-6.7965567 30.083091 6.19984346]
3: Leaf node at depth 6 has 872 points with origin [-6.78077545 30.083091 6.19984346]
5: Leaf node at depth 6 has 18 points with origin [-6.78077545 30.06730975 6.21562471]
1: Internal node at depth 5 has 4 children and 458 points ([-6.7649942 30.06730975 6.19984346])
2: Leaf node at depth 6 has 380 points with origin [-6.7649942 30.083091 6.19984346]
4: Leaf node at depth 6 has 55 points with origin [-6.7649942 30.06730975 6.21562471]
5: Leaf node at depth 6 has 1 points with origin [-6.74921295 30.06730975 6.21562471]
6: Leaf node at depth 6 has 22 points with origin [-6.7649942 30.083091 6.21562471]
2: Internal node at depth 5 has 7 children and 1722 points ([-6.7965567 30.09887225 6.19984346])
0: Leaf node at depth 6 has 489 points with origin [-6.7965567 30.09887225 6.19984346]
1: Leaf node at depth 6 has 861 points with origin [-6.78077545 30.09887225 6.19984346]
2: Leaf node at depth 6 has 99 points with origin [-6.7965567 30.1146535 6.19984346]
3: Leaf node at depth 6 has 183 points with origin [-6.78077545 30.1146535 6.19984346]
4: Leaf node at depth 6 has 6 points with origin [-6.7965567 30.09887225 6.21562471]
6: Leaf node at depth 6 has 67 points with origin [-6.7965567 30.1146535 6.21562471]
7: Leaf node at depth 6 has 17 points with origin [-6.78077545 30.1146535 6.21562471]
3: Internal node at depth 5 has 6 children and 3105 points ([-6.7649942 30.09887225 6.19984346])
0: Leaf node at depth 6 has 663 points with origin [-6.7649942 30.09887225 6.19984346]
2: Leaf node at depth 6 has 448 points with origin [-6.7649942 30.1146535 6.19984346]
3: Leaf node at depth 6 has 492 points with origin [-6.74921295 30.1146535 6.19984346]
4: Leaf node at depth 6 has 15 points with origin [-6.7649942 30.09887225 6.21562471]
6: Leaf node at depth 6 has 771 points with origin [-6.7649942 30.1146535 6.21562471]
7: Leaf node at depth 6 has 716 points with origin [-6.74921295 30.1146535 6.21562471]
4: Internal node at depth 5 has 1 children and 17 points ([-6.7965567 30.06730975 6.23140596])
5: Internal node at depth 5 has 4 children and 375 points ([-6.7649942 30.06730975 6.23140596])
0: Leaf node at depth 6 has 34 points with origin [-6.7649942 30.06730975 6.23140596]
2: Leaf node at depth 6 has 80 points with origin [-6.7649942 30.083091 6.23140596]
4: Leaf node at depth 6 has 8 points with origin [-6.7649942 30.06730975 6.24718721]
6: Leaf node at depth 6 has 253 points with origin [-6.7649942 30.083091 6.24718721]
6: Internal node at depth 5 has 5 children and 1423 points ([-6.7965567 30.09887225 6.23140596])
2: Leaf node at depth 6 has 222 points with origin [-6.7965567 30.1146535 6.23140596]
3: Leaf node at depth 6 has 828 points with origin [-6.78077545 30.1146535 6.23140596]
5: Leaf node at depth 6 has 87 points with origin [-6.78077545 30.09887225 6.24718721]
6: Leaf node at depth 6 has 205 points with origin [-6.7965567 30.1146535 6.24718721]
7: Leaf node at depth 6 has 81 points with origin [-6.78077545 30.1146535 6.24718721]
7: Internal node at depth 5 has 4 children and 796 points ([-6.7649942 30.09887225 6.23140596])
0: Leaf node at depth 6 has 50 points with origin [-6.7649942 30.09887225 6.23140596]
2: Leaf node at depth 6 has 366 points with origin [-6.7649942 30.1146535 6.23140596]
4: Leaf node at depth 6 has 314 points with origin [-6.7649942 30.09887225 6.24718721]
6: Leaf node at depth 6 has 66 points with origin [-6.7649942 30.1146535 6.24718721]
2: Internal node at depth 4 has 4 children and 11062 points ([-6.8596817 30.13043475 6.19984346])
1: Internal node at depth 5 has 8 children and 3116 points ([-6.8281192 30.13043475 6.19984346])
0: Leaf node at depth 6 has 250 points with origin [-6.8281192 30.13043475 6.19984346]
1: Leaf node at depth 6 has 722 points with origin [-6.81233795 30.13043475 6.19984346]
2: Leaf node at depth 6 has 556 points with origin [-6.8281192 30.146216 6.19984346]
3: Leaf node at depth 6 has 10 points with origin [-6.81233795 30.146216 6.19984346]
4: Leaf node at depth 6 has 146 points with origin [-6.8281192 30.13043475 6.21562471]
5: Leaf node at depth 6 has 887 points with origin [-6.81233795 30.13043475 6.21562471]
6: Leaf node at depth 6 has 528 points with origin [-6.8281192 30.146216 6.21562471]

7: Leaf node at depth 6 has 17 points with origin [-6.81233795 30.146216 6.21562471]
3: Internal node at depth 5 has 1 children and 6 points ([-6.8281192 30.16199725 6.19984346])
5: Internal node at depth 5 has 8 children and 5019 points ([-6.8281192 30.13043475 6.23140596])
0: Leaf node at depth 6 has 312 points with origin [-6.8281192 30.13043475 6.23140596]
1: Leaf node at depth 6 has 910 points with origin [-6.81233795 30.13043475 6.23140596]
2: Leaf node at depth 6 has 536 points with origin [-6.8281192 30.146216 6.23140596]
3: Leaf node at depth 6 has 429 points with origin [-6.81233795 30.146216 6.23140596]
4: Leaf node at depth 6 has 595 points with origin [-6.8281192 30.13043475 6.24718721]
5: Leaf node at depth 6 has 684 points with origin [-6.81233795 30.13043475 6.24718721]
6: Leaf node at depth 6 has 547 points with origin [-6.8281192 30.146216 6.24718721]
7: Leaf node at depth 6 has 1006 points with origin [-6.81233795 30.146216 6.24718721]
7: Internal node at depth 5 has 8 children and 2921 points ([-6.8281192 30.16199725 6.23140596])
0: Leaf node at depth 6 has 6 points with origin [-6.8281192 30.16199725 6.23140596]
1: Leaf node at depth 6 has 729 points with origin [-6.81233795 30.16199725 6.23140596]
2: Leaf node at depth 6 has 1 points with origin [-6.8281192 30.1777785 6.23140596]
3: Leaf node at depth 6 has 539 points with origin [-6.81233795 30.1777785 6.23140596]
4: Leaf node at depth 6 has 3 points with origin [-6.8281192 30.16199725 6.24718721]
5: Leaf node at depth 6 has 884 points with origin [-6.81233795 30.16199725 6.24718721]
6: Leaf node at depth 6 has 1 points with origin [-6.8281192 30.1777785 6.24718721]
7: Leaf node at depth 6 has 758 points with origin [-6.81233795 30.1777785 6.24718721]
3: Internal node at depth 4 has 7 children and 18822 points ([-6.7965567 30.13043475 6.19984346])
0: Internal node at depth 5 has 2 children and 665 points ([-6.7965567 30.13043475 6.19984346])
5: Leaf node at depth 6 has 492 points with origin [-6.78077545 30.13043475 6.21562471]
7: Leaf node at depth 6 has 173 points with origin [-6.78077545 30.146216 6.21562471]
1: Internal node at depth 5 has 8 children and 6497 points ([-6.7649942 30.13043475 6.19984346])
0: Leaf node at depth 6 has 89 points with origin [-6.7649942 30.13043475 6.19984346]
1: Leaf node at depth 6 has 1549 points with origin [-6.74921295 30.13043475 6.19984346]
2: Leaf node at depth 6 has 8 points with origin [-6.7649942 30.146216 6.19984346]
3: Leaf node at depth 6 has 1310 points with origin [-6.74921295 30.146216 6.19984346]
4: Leaf node at depth 6 has 1266 points with origin [-6.7649942 30.13043475 6.21562471]
5: Leaf node at depth 6 has 341 points with origin [-6.74921295 30.13043475 6.21562471]
6: Leaf node at depth 6 has 370 points with origin [-6.7649942 30.146216 6.21562471]
7: Leaf node at depth 6 has 1564 points with origin [-6.74921295 30.146216 6.21562471]
3: Internal node at depth 5 has 7 children and 4834 points ([-6.7649942 30.16199725 6.19984346])
1: Leaf node at depth 6 has 788 points with origin [-6.74921295 30.16199725 6.19984346]
2: Leaf node at depth 6 has 39 points with origin [-6.7649942 30.1777785 6.19984346]
3: Leaf node at depth 6 has 734 points with origin [-6.74921295 30.1777785 6.19984346]
4: Leaf node at depth 6 has 66 points with origin [-6.7649942 30.16199725 6.21562471]
5: Leaf node at depth 6 has 1678 points with origin [-6.74921295 30.16199725 6.21562471]
6: Leaf node at depth 6 has 208 points with origin [-6.7649942 30.1777785 6.21562471]
7: Leaf node at depth 6 has 1321 points with origin [-6.74921295 30.1777785 6.21562471]
4: Internal node at depth 5 has 7 children and 2715 points ([-6.7965567 30.13043475 6.23140596])
0: Leaf node at depth 6 has 395 points with origin [-6.7965567 30.13043475 6.23140596]
1: Leaf node at depth 6 has 923 points with origin [-6.78077545 30.13043475 6.23140596]
2: Leaf node at depth 6 has 317 points with origin [-6.7965567 30.146216 6.23140596]
3: Leaf node at depth 6 has 650 points with origin [-6.78077545 30.146216 6.23140596]
4: Leaf node at depth 6 has 43 points with origin [-6.7965567 30.13043475 6.24718721]
5: Leaf node at depth 6 has 308 points with origin [-6.78077545 30.13043475 6.24718721]
7: Leaf node at depth 6 has 79 points with origin [-6.78077545 30.146216 6.24718721]
5: Internal node at depth 5 has 5 children and 710 points ([-6.7649942 30.13043475 6.23140596])
0: Leaf node at depth 6 has 10 points with origin [-6.7649942 30.13043475 6.23140596]
2: Leaf node at depth 6 has 577 points with origin [-6.7649942 30.146216 6.23140596]
3: Leaf node at depth 6 has 4 points with origin [-6.74921295 30.146216 6.23140596]
4: Leaf node at depth 6 has 95 points with origin [-6.7649942 30.13043475 6.24718721]
6: Leaf node at depth 6 has 24 points with origin [-6.7649942 30.146216 6.24718721]
6: Internal node at depth 5 has 6 children and 1352 points ([-6.7965567 30.16199725 6.23140596])

```
    0: Leaf node at depth 6 has 120 points with origin [-6.7965567  30.16199725  6.23140596]
    1: Leaf node at depth 6 has 580 points with origin [-6.78077545 30.16199725  6.23140596]
    2: Leaf node at depth 6 has 9 points with origin [-6.7965567  30.1777785   6.23140596]
    3: Leaf node at depth 6 has 629 points with origin [-6.78077545 30.1777785   6.23140596]
    5: Leaf node at depth 6 has 1 points with origin [-6.78077545 30.16199725  6.24718721]
    6: Leaf node at depth 6 has 13 points with origin [-6.7965567  30.1777785   6.24718721]
  7: Internal node at depth 5 has 4 children and 2049 points ([-6.7649942  30.16199725  6.23140596])
    0: Leaf node at depth 6 has 792 points with origin [-6.7649942  30.16199725  6.23140596]
    1: Leaf node at depth 6 has 369 points with origin [-6.74921295 30.16199725  6.23140596]
    2: Leaf node at depth 6 has 721 points with origin [-6.7649942  30.1777785   6.23140596]
    3: Leaf node at depth 6 has 167 points with origin [-6.74921295 30.1777785   6.23140596]
4: Internal node at depth 4 has 4 children and 12419 points ([-6.8596817  30.06730975  6.26296846])
  1: Internal node at depth 5 has 8 children and 6597 points ([-6.8281192  30.06730975  6.26296846])
    0: Leaf node at depth 6 has 265 points with origin [-6.8281192  30.06730975  6.26296846]
    1: Leaf node at depth 6 has 1196 points with origin [-6.81233795 30.06730975  6.26296846]
    2: Leaf node at depth 6 has 1072 points with origin [-6.8281192  30.083091    6.26296846]
    3: Leaf node at depth 6 has 1074 points with origin [-6.81233795 30.083091    6.26296846]
    4: Leaf node at depth 6 has 581 points with origin [-6.8281192  30.06730975  6.27874971]
    5: Leaf node at depth 6 has 897 points with origin [-6.81233795 30.06730975  6.27874971]
    6: Leaf node at depth 6 has 554 points with origin [-6.8281192  30.083091    6.27874971]
    7: Leaf node at depth 6 has 958 points with origin [-6.81233795 30.083091    6.27874971]
  3: Internal node at depth 5 has 8 children and 5561 points ([-6.8281192  30.09887225  6.26296846])
    0: Leaf node at depth 6 has 544 points with origin [-6.8281192  30.09887225  6.26296846]
    1: Leaf node at depth 6 has 774 points with origin [-6.81233795 30.09887225  6.26296846]
    2: Leaf node at depth 6 has 444 points with origin [-6.8281192  30.1146535   6.26296846]
    3: Leaf node at depth 6 has 696 points with origin [-6.81233795 30.1146535   6.26296846]
    4: Leaf node at depth 6 has 578 points with origin [-6.8281192  30.09887225  6.27874971]
    5: Leaf node at depth 6 has 1008 points with origin [-6.81233795 30.09887225  6.27874971]
    6: Leaf node at depth 6 has 560 points with origin [-6.8281192  30.1146535   6.27874971]
    7: Leaf node at depth 6 has 957 points with origin [-6.81233795 30.1146535   6.27874971]
  5: Internal node at depth 5 has 4 children and 131 points ([-6.8281192  30.06730975  6.29453096])
    0: Leaf node at depth 6 has 2 points with origin [-6.8281192  30.06730975  6.29453096]
    2: Leaf node at depth 6 has 3 points with origin [-6.8281192  30.083091    6.29453096]
    5: Leaf node at depth 6 has 37 points with origin [-6.81233795 30.06730975  6.31031221]
    7: Leaf node at depth 6 has 89 points with origin [-6.81233795 30.083091    6.31031221]
  7: Internal node at depth 5 has 4 children and 130 points ([-6.8281192  30.09887225  6.29453096])
    0: Leaf node at depth 6 has 3 points with origin [-6.8281192  30.09887225  6.29453096]
    2: Leaf node at depth 6 has 4 points with origin [-6.8281192  30.1146535   6.29453096]
    5: Leaf node at depth 6 has 60 points with origin [-6.81233795 30.09887225  6.31031221]
    7: Leaf node at depth 6 has 63 points with origin [-6.81233795 30.1146535   6.31031221]
5: Internal node at depth 4 has 6 children and 8746 points ([-6.7965567  30.06730975  6.26296846])
  0: Internal node at depth 5 has 4 children and 995 points ([-6.7965567  30.06730975  6.26296846])
    1: Leaf node at depth 6 has 1 points with origin [-6.78077545 30.06730975  6.26296846]
    3: Leaf node at depth 6 has 17 points with origin [-6.78077545 30.083091    6.26296846]
    4: Leaf node at depth 6 has 503 points with origin [-6.7965567  30.06730975  6.27874971]
    6: Leaf node at depth 6 has 474 points with origin [-6.7965567  30.083091    6.27874971]
  1: Internal node at depth 5 has 6 children and 391 points ([-6.7649942  30.06730975  6.26296846])
    0: Leaf node at depth 6 has 275 points with origin [-6.7649942  30.06730975  6.26296846]
    2: Leaf node at depth 6 has 53 points with origin [-6.7649942  30.083091    6.26296846]
    3: Leaf node at depth 6 has 3 points with origin [-6.74921295 30.083091    6.26296846]
    5: Leaf node at depth 6 has 3 points with origin [-6.74921295 30.06730975  6.27874971]
    6: Leaf node at depth 6 has 51 points with origin [-6.7649942  30.083091    6.27874971]
    7: Leaf node at depth 6 has 6 points with origin [-6.74921295 30.083091    6.27874971]
  2: Internal node at depth 5 has 4 children and 1287 points ([-6.7965567  30.09887225  6.26296846])
    1: Leaf node at depth 6 has 152 points with origin [-6.78077545 30.09887225  6.26296846]
    3: Leaf node at depth 6 has 127 points with origin [-6.78077545 30.1146535   6.26296846]
```

4: Leaf node at depth 6 has 497 points with origin [-6.7965567 30.09887225 6.27874971]
6: Leaf node at depth 6 has 511 points with origin [-6.7965567 30.1146535 6.27874971]
3: Internal node at depth 5 has 4 children and 495 points ([-6.7649942 30.09887225 6.26296846])
0: Leaf node at depth 6 has 56 points with origin [-6.7649942 30.09887225 6.26296846]
2: Leaf node at depth 6 has 8 points with origin [-6.7649942 30.1146535 6.26296846]
4: Leaf node at depth 6 has 189 points with origin [-6.7649942 30.09887225 6.27874971]
6: Leaf node at depth 6 has 242 points with origin [-6.7649942 30.1146535 6.27874971]
4: Internal node at depth 5 has 4 children and 2902 points ([-6.7965567 30.06730975 6.29453096])
0: Leaf node at depth 6 has 898 points with origin [-6.7965567 30.06730975 6.29453096]
2: Leaf node at depth 6 has 893 points with origin [-6.7965567 30.083091 6.29453096]
4: Leaf node at depth 6 has 586 points with origin [-6.7965567 30.06730975 6.31031221]
6: Leaf node at depth 6 has 525 points with origin [-6.7965567 30.083091 6.31031221]
6: Internal node at depth 5 has 4 children and 2676 points ([-6.7965567 30.09887225 6.29453096])
0: Leaf node at depth 6 has 903 points with origin [-6.7965567 30.09887225 6.29453096]
2: Leaf node at depth 6 has 773 points with origin [-6.7965567 30.1146535 6.29453096]
4: Leaf node at depth 6 has 486 points with origin [-6.7965567 30.09887225 6.31031221]
6: Leaf node at depth 6 has 514 points with origin [-6.7965567 30.1146535 6.31031221]
6: Internal node at depth 4 has 4 children and 12237 points ([-6.8596817 30.13043475 6.26296846])
1: Internal node at depth 5 has 8 children and 6327 points ([-6.8281192 30.13043475 6.26296846])
0: Leaf node at depth 6 has 586 points with origin [-6.8281192 30.13043475 6.26296846]
1: Leaf node at depth 6 has 830 points with origin [-6.81233795 30.13043475 6.26296846]
2: Leaf node at depth 6 has 681 points with origin [-6.8281192 30.146216 6.26296846]
3: Leaf node at depth 6 has 942 points with origin [-6.81233795 30.146216 6.26296846]
4: Leaf node at depth 6 has 535 points with origin [-6.8281192 30.13043475 6.27874971]
5: Leaf node at depth 6 has 1179 points with origin [-6.81233795 30.13043475 6.27874971]
6: Leaf node at depth 6 has 346 points with origin [-6.8281192 30.146216 6.27874971]
7: Leaf node at depth 6 has 1228 points with origin [-6.81233795 30.146216 6.27874971]
3: Internal node at depth 5 has 4 children and 3509 points ([-6.8281192 30.16199725 6.26296846])
1: Leaf node at depth 6 has 812 points with origin [-6.81233795 30.16199725 6.26296846]
3: Leaf node at depth 6 has 832 points with origin [-6.81233795 30.1777785 6.26296846]
5: Leaf node at depth 6 has 1007 points with origin [-6.81233795 30.16199725 6.27874971]
7: Leaf node at depth 6 has 858 points with origin [-6.81233795 30.1777785 6.27874971]
5: Internal node at depth 5 has 5 children and 1424 points ([-6.8281192 30.13043475 6.29453096])
0: Leaf node at depth 6 has 7 points with origin [-6.8281192 30.13043475 6.29453096]
1: Leaf node at depth 6 has 661 points with origin [-6.81233795 30.13043475 6.29453096]
2: Leaf node at depth 6 has 3 points with origin [-6.8281192 30.146216 6.29453096]
3: Leaf node at depth 6 has 399 points with origin [-6.81233795 30.146216 6.29453096]
5: Leaf node at depth 6 has 354 points with origin [-6.81233795 30.13043475 6.31031221]
7: Internal node at depth 5 has 4 children and 977 points ([-6.8281192 30.16199725 6.29453096])
0: Leaf node at depth 6 has 5 points with origin [-6.8281192 30.16199725 6.29453096]
1: Leaf node at depth 6 has 406 points with origin [-6.81233795 30.16199725 6.29453096]
3: Leaf node at depth 6 has 565 points with origin [-6.81233795 30.1777785 6.29453096]
7: Leaf node at depth 6 has 1 points with origin [-6.81233795 30.1777785 6.31031221]
7: Internal node at depth 4 has 7 children and 14878 points ([-6.7965567 30.13043475 6.26296846])
0: Internal node at depth 5 has 6 children and 572 points ([-6.7965567 30.13043475 6.26296846])
1: Leaf node at depth 6 has 184 points with origin [-6.78077545 30.13043475 6.26296846]
3: Leaf node at depth 6 has 142 points with origin [-6.78077545 30.146216 6.26296846]
4: Leaf node at depth 6 has 126 points with origin [-6.7965567 30.13043475 6.27874971]
5: Leaf node at depth 6 has 3 points with origin [-6.78077545 30.13043475 6.27874971]
6: Leaf node at depth 6 has 115 points with origin [-6.7965567 30.146216 6.27874971]
7: Leaf node at depth 6 has 2 points with origin [-6.78077545 30.146216 6.27874971]
1: Internal node at depth 5 has 4 children and 328 points ([-6.7649942 30.13043475 6.26296846])
0: Leaf node at depth 6 has 5 points with origin [-6.7649942 30.13043475 6.26296846]
2: Leaf node at depth 6 has 8 points with origin [-6.7649942 30.146216 6.26296846]
4: Leaf node at depth 6 has 233 points with origin [-6.7649942 30.13043475 6.27874971]
6: Leaf node at depth 6 has 82 points with origin [-6.7649942 30.146216 6.27874971]

2: Internal node at depth 5 has 4 children and 1340 points ([-6.7965567 30.16199725 6.26296846])
2: Leaf node at depth 6 has 23 points with origin [-6.7965567 30.1777785 6.26296846]
4: Leaf node at depth 6 has 228 points with origin [-6.7965567 30.16199725 6.27874971]
6: Leaf node at depth 6 has 631 points with origin [-6.7965567 30.1777785 6.27874971]
7: Leaf node at depth 6 has 458 points with origin [-6.78077545 30.1777785 6.27874971]
3: Internal node at depth 5 has 2 children and 1089 points ([-6.7649942 30.16199725 6.26296846])
6: Leaf node at depth 6 has 544 points with origin [-6.7649942 30.1777785 6.27874971]
7: Leaf node at depth 6 has 545 points with origin [-6.74921295 30.1777785 6.27874971]
4: Internal node at depth 5 has 4 children and 3744 points ([-6.7965567 30.13043475 6.29453096])
0: Leaf node at depth 6 has 762 points with origin [-6.7965567 30.13043475 6.29453096]
2: Leaf node at depth 6 has 905 points with origin [-6.7965567 30.146216 6.29453096]
4: Leaf node at depth 6 has 830 points with origin [-6.7965567 30.13043475 6.31031221]
6: Leaf node at depth 6 has 1247 points with origin [-6.7965567 30.146216 6.31031221]
6: Internal node at depth 5 has 6 children and 5171 points ([-6.7965567 30.16199725 6.29453096])
0: Leaf node at depth 6 has 926 points with origin [-6.7965567 30.16199725 6.29453096]
2: Leaf node at depth 6 has 1036 points with origin [-6.7965567 30.1777785 6.29453096]
3: Leaf node at depth 6 has 659 points with origin [-6.78077545 30.1777785 6.29453096]
4: Leaf node at depth 6 has 856 points with origin [-6.7965567 30.16199725 6.31031221]
6: Leaf node at depth 6 has 985 points with origin [-6.7965567 30.1777785 6.31031221]
7: Leaf node at depth 6 has 709 points with origin [-6.78077545 30.1777785 6.31031221]
7: Internal node at depth 5 has 4 children and 2634 points ([-6.7649942 30.16199725 6.29453096])
2: Leaf node at depth 6 has 645 points with origin [-6.7649942 30.1777785 6.29453096]
3: Leaf node at depth 6 has 680 points with origin [-6.74921295 30.1777785 6.29453096]
6: Leaf node at depth 6 has 635 points with origin [-6.7649942 30.1777785 6.31031221]
7: Leaf node at depth 6 has 674 points with origin [-6.74921295 30.1777785 6.31031221]
1: Internal node at depth 2 has 8 children and 673077 points ([-6.7334317 29.94105975 6.07359346])
0: Internal node at depth 3 has 8 children and 105122 points ([-6.7334317 29.94105975 6.07359346])
0: Internal node at depth 4 has 2 children and 2113 points ([-6.7334317 29.94105975 6.07359346])
6: Internal node at depth 5 has 4 children and 253 points ([-6.7334317 29.97262225 6.10515596])
4: Leaf node at depth 6 has 113 points with origin [-6.7334317 29.97262225 6.12093721]
5: Leaf node at depth 6 has 2 points with origin [-6.71765045 29.97262225 6.12093721]
6: Leaf node at depth 6 has 121 points with origin [-6.7334317 29.9884035 6.12093721]
7: Leaf node at depth 6 has 17 points with origin [-6.71765045 29.9884035 6.12093721]
7: Internal node at depth 5 has 4 children and 1860 points ([-6.7018692 29.97262225 6.10515596])
4: Leaf node at depth 6 has 130 points with origin [-6.7018692 29.97262225 6.12093721]
5: Leaf node at depth 6 has 203 points with origin [-6.68608795 29.97262225 6.12093721]
6: Leaf node at depth 6 has 836 points with origin [-6.7018692 29.9884035 6.12093721]
7: Leaf node at depth 6 has 691 points with origin [-6.68608795 29.9884035 6.12093721]
1: Internal node at depth 4 has 2 children and 3287 points ([-6.6703067 29.94105975 6.07359346])
6: Internal node at depth 5 has 4 children and 1686 points ([-6.6703067 29.97262225 6.10515596])
4: Leaf node at depth 6 has 139 points with origin [-6.6703067 29.97262225 6.12093721]
5: Leaf node at depth 6 has 139 points with origin [-6.65452545 29.97262225 6.12093721]
6: Leaf node at depth 6 has 740 points with origin [-6.6703067 29.9884035 6.12093721]
7: Leaf node at depth 6 has 668 points with origin [-6.65452545 29.9884035 6.12093721]
7: Internal node at depth 5 has 4 children and 1601 points ([-6.6387442 29.97262225 6.10515596])
4: Leaf node at depth 6 has 138 points with origin [-6.6387442 29.97262225 6.12093721]
5: Leaf node at depth 6 has 127 points with origin [-6.62296295 29.97262225 6.12093721]
6: Leaf node at depth 6 has 730 points with origin [-6.6387442 29.9884035 6.12093721]
7: Leaf node at depth 6 has 606 points with origin [-6.62296295 29.9884035 6.12093721]
2: Internal node at depth 4 has 4 children and 13819 points ([-6.7334317 30.00418475 6.07359346])
4: Internal node at depth 5 has 3 children and 729 points ([-6.7334317 30.00418475 6.10515596])
4: Leaf node at depth 6 has 2 points with origin [-6.7334317 30.00418475 6.12093721]
5: Leaf node at depth 6 has 367 points with origin [-6.71765045 30.00418475 6.12093721]
7: Leaf node at depth 6 has 360 points with origin [-6.71765045 30.019966 6.12093721]
5: Internal node at depth 5 has 4 children and 3730 points ([-6.7018692 30.00418475 6.10515596])
4: Leaf node at depth 6 has 1612 points with origin [-6.7018692 30.00418475 6.12093721]

5: Leaf node at depth 6 has 892 points with origin [-6.68608795 30.00418475 6.12093721]
6: Leaf node at depth 6 has 457 points with origin [-6.7018692 30.019966 6.12093721]
7: Leaf node at depth 6 has 769 points with origin [-6.68608795 30.019966 6.12093721]
6: Internal node at depth 5 has 4 children and 4989 points ([-6.7334317 30.03574725 6.10515596])
4: Leaf node at depth 6 has 1878 points with origin [-6.7334317 30.03574725 6.12093721]
5: Leaf node at depth 6 has 1286 points with origin [-6.71765045 30.03574725 6.12093721]
6: Leaf node at depth 6 has 1016 points with origin [-6.7334317 30.0515285 6.12093721]
7: Leaf node at depth 6 has 809 points with origin [-6.71765045 30.0515285 6.12093721]
7: Internal node at depth 5 has 4 children and 4371 points ([-6.7018692 30.03574725 6.10515596])
4: Leaf node at depth 6 has 1311 points with origin [-6.7018692 30.03574725 6.12093721]
5: Leaf node at depth 6 has 1224 points with origin [-6.68608795 30.03574725 6.12093721]
6: Leaf node at depth 6 has 1088 points with origin [-6.7018692 30.0515285 6.12093721]
7: Leaf node at depth 6 has 748 points with origin [-6.68608795 30.0515285 6.12093721]
3: Internal node at depth 4 has 4 children and 12785 points ([-6.6703067 30.00418475 6.07359346])
4: Internal node at depth 5 has 4 children and 3350 points ([-6.6703067 30.00418475 6.10515596])
4: Leaf node at depth 6 has 879 points with origin [-6.6703067 30.00418475 6.12093721]
5: Leaf node at depth 6 has 807 points with origin [-6.65452545 30.00418475 6.12093721]
6: Leaf node at depth 6 has 863 points with origin [-6.6703067 30.019966 6.12093721]
7: Leaf node at depth 6 has 801 points with origin [-6.65452545 30.019966 6.12093721]
5: Internal node at depth 5 has 4 children and 3087 points ([-6.6387442 30.00418475 6.10515596])
4: Leaf node at depth 6 has 790 points with origin [-6.6387442 30.00418475 6.12093721]
5: Leaf node at depth 6 has 768 points with origin [-6.62296295 30.00418475 6.12093721]
6: Leaf node at depth 6 has 825 points with origin [-6.6387442 30.019966 6.12093721]
7: Leaf node at depth 6 has 704 points with origin [-6.62296295 30.019966 6.12093721]
6: Internal node at depth 5 has 4 children and 3257 points ([-6.6703067 30.03574725 6.10515596])
4: Leaf node at depth 6 has 801 points with origin [-6.6703067 30.03574725 6.12093721]
5: Leaf node at depth 6 has 798 points with origin [-6.65452545 30.03574725 6.12093721]
6: Leaf node at depth 6 has 816 points with origin [-6.6703067 30.0515285 6.12093721]
7: Leaf node at depth 6 has 842 points with origin [-6.65452545 30.0515285 6.12093721]
7: Internal node at depth 5 has 4 children and 3091 points ([-6.6387442 30.03574725 6.10515596])
4: Leaf node at depth 6 has 818 points with origin [-6.6387442 30.03574725 6.12093721]
5: Leaf node at depth 6 has 722 points with origin [-6.62296295 30.03574725 6.12093721]
6: Leaf node at depth 6 has 815 points with origin [-6.6387442 30.0515285 6.12093721]
7: Leaf node at depth 6 has 736 points with origin [-6.62296295 30.0515285 6.12093721]
4: Internal node at depth 4 has 8 children and 25461 points ([-6.7334317 29.94105975 6.13671846])
0: Internal node at depth 5 has 4 children and 155 points ([-6.7334317 29.94105975 6.13671846])
4: Leaf node at depth 6 has 31 points with origin [-6.7334317 29.94105975 6.15249971]
5: Leaf node at depth 6 has 14 points with origin [-6.71765045 29.94105975 6.15249971]
6: Leaf node at depth 6 has 30 points with origin [-6.7334317 29.956841 6.15249971]
7: Leaf node at depth 6 has 80 points with origin [-6.71765045 29.956841 6.15249971]
1: Internal node at depth 5 has 3 children and 21 points ([-6.7018692 29.94105975 6.13671846])
2: Internal node at depth 5 has 7 children and 3787 points ([-6.7334317 29.97262225 6.13671846])
0: Leaf node at depth 6 has 691 points with origin [-6.7334317 29.97262225 6.13671846]
1: Leaf node at depth 6 has 372 points with origin [-6.71765045 29.97262225 6.13671846]
2: Leaf node at depth 6 has 232 points with origin [-6.7334317 29.9884035 6.13671846]
3: Leaf node at depth 6 has 568 points with origin [-6.71765045 29.9884035 6.13671846]
4: Leaf node at depth 6 has 792 points with origin [-6.7334317 29.97262225 6.15249971]
5: Leaf node at depth 6 has 860 points with origin [-6.71765045 29.97262225 6.15249971]
7: Leaf node at depth 6 has 272 points with origin [-6.71765045 29.9884035 6.15249971]
3: Internal node at depth 5 has 8 children and 5567 points ([-6.7018692 29.97262225 6.13671846])
0: Leaf node at depth 6 has 748 points with origin [-6.7018692 29.97262225 6.13671846]
1: Leaf node at depth 6 has 839 points with origin [-6.68608795 29.97262225 6.13671846]
2: Leaf node at depth 6 has 932 points with origin [-6.7018692 29.9884035 6.13671846]
3: Leaf node at depth 6 has 1015 points with origin [-6.68608795 29.9884035 6.13671846]
4: Leaf node at depth 6 has 887 points with origin [-6.7018692 29.97262225 6.15249971]
5: Leaf node at depth 6 has 620 points with origin [-6.68608795 29.97262225 6.15249971]

```
    6: Leaf node at depth 6 has 20 points with origin [-6.7018692  29.9884035   6.15249971]
    7: Leaf node at depth 6 has 506 points with origin [-6.68608795 29.9884035   6.15249971]
  4: Internal node at depth 5 has 7 children and 3566 points ([-6.7334317  29.94105975  6.16828096])
    0: Leaf node at depth 6 has 892 points with origin [-6.7334317  29.94105975  6.16828096]
    1: Leaf node at depth 6 has 835 points with origin [-6.71765045 29.94105975  6.16828096]
    2: Leaf node at depth 6 has 5 points with origin [-6.7334317  29.956841    6.16828096]
    3: Leaf node at depth 6 has 44 points with origin [-6.71765045 29.956841    6.16828096]
    4: Leaf node at depth 6 has 868 points with origin [-6.7334317  29.94105975  6.18406221]
    5: Leaf node at depth 6 has 921 points with origin [-6.71765045 29.94105975  6.18406221]
    7: Leaf node at depth 6 has 1 points with origin [-6.71765045 29.956841    6.18406221]
  5: Internal node at depth 5 has 8 children and 2795 points ([-6.7018692  29.94105975  6.16828096])
    0: Leaf node at depth 6 has 675 points with origin [-6.7018692  29.94105975  6.16828096]
    1: Leaf node at depth 6 has 580 points with origin [-6.68608795 29.94105975  6.16828096]
    2: Leaf node at depth 6 has 9 points with origin [-6.7018692  29.956841    6.16828096]
    3: Leaf node at depth 6 has 35 points with origin [-6.68608795 29.956841    6.16828096]
    4: Leaf node at depth 6 has 715 points with origin [-6.7018692  29.94105975  6.18406221]
    5: Leaf node at depth 6 has 761 points with origin [-6.68608795 29.94105975  6.18406221]
    6: Leaf node at depth 6 has 2 points with origin [-6.7018692  29.956841    6.18406221]
    7: Leaf node at depth 6 has 18 points with origin [-6.68608795 29.956841    6.18406221]
  6: Internal node at depth 5 has 4 children and 4062 points ([-6.7334317  29.97262225  6.16828096])
    0: Leaf node at depth 6 has 970 points with origin [-6.7334317  29.97262225  6.16828096]
    1: Leaf node at depth 6 has 1011 points with origin [-6.71765045 29.97262225  6.16828096]
    4: Leaf node at depth 6 has 1027 points with origin [-6.7334317  29.97262225  6.18406221]
    5: Leaf node at depth 6 has 1054 points with origin [-6.71765045 29.97262225  6.18406221]
  7: Internal node at depth 5 has 6 children and 5508 points ([-6.7018692  29.97262225  6.16828096])
    0: Leaf node at depth 6 has 999 points with origin [-6.7018692  29.97262225  6.16828096]
    1: Leaf node at depth 6 has 1115 points with origin [-6.68608795 29.97262225  6.16828096]
    3: Leaf node at depth 6 has 598 points with origin [-6.68608795 29.9884035   6.16828096]
    4: Leaf node at depth 6 has 1057 points with origin [-6.7018692  29.97262225  6.18406221]
    5: Leaf node at depth 6 has 1076 points with origin [-6.68608795 29.97262225  6.18406221]
    7: Leaf node at depth 6 has 663 points with origin [-6.68608795 29.9884035   6.18406221]
5: Internal node at depth 4 has 6 children and 26436 points ([-6.6703067  29.94105975  6.13671846])
  2: Internal node at depth 5 has 6 children and 4647 points ([-6.6703067  29.97262225  6.13671846])
    0: Leaf node at depth 6 has 945 points with origin [-6.6703067  29.97262225  6.13671846]
    1: Leaf node at depth 6 has 932 points with origin [-6.65452545 29.97262225  6.13671846]
    2: Leaf node at depth 6 has 964 points with origin [-6.6703067  29.9884035   6.13671846]
    3: Leaf node at depth 6 has 885 points with origin [-6.65452545 29.9884035   6.13671846]
    4: Leaf node at depth 6 has 454 points with origin [-6.6703067  29.97262225  6.15249971]
    5: Leaf node at depth 6 has 467 points with origin [-6.65452545 29.97262225  6.15249971]
  3: Internal node at depth 5 has 6 children and 1057 points ([-6.6387442  29.97262225  6.13671846])
    0: Leaf node at depth 6 has 216 points with origin [-6.6387442  29.97262225  6.13671846]
    2: Leaf node at depth 6 has 378 points with origin [-6.6387442  29.9884035   6.13671846]
    3: Leaf node at depth 6 has 156 points with origin [-6.62296295 29.9884035   6.13671846]
    4: Leaf node at depth 6 has 71 points with origin [-6.6387442  29.97262225  6.15249971]
    6: Leaf node at depth 6 has 141 points with origin [-6.6387442  29.9884035   6.15249971]
    7: Leaf node at depth 6 has 95 points with origin [-6.62296295 29.9884035   6.15249971]
  4: Internal node at depth 5 has 6 children and 2506 points ([-6.6703067  29.94105975  6.16828096])
    0: Leaf node at depth 6 has 574 points with origin [-6.6703067  29.94105975  6.16828096]
    1: Leaf node at depth 6 has 420 points with origin [-6.65452545 29.94105975  6.16828096]
    2: Leaf node at depth 6 has 1 points with origin [-6.6703067  29.956841    6.16828096]
    4: Leaf node at depth 6 has 743 points with origin [-6.6703067  29.94105975  6.18406221]
    5: Leaf node at depth 6 has 764 points with origin [-6.65452545 29.94105975  6.18406221]
    6: Leaf node at depth 6 has 4 points with origin [-6.6703067  29.956841    6.18406221]
  5: Internal node at depth 5 has 4 children and 1658 points ([-6.6387442  29.94105975  6.16828096])
    0: Leaf node at depth 6 has 239 points with origin [-6.6387442  29.94105975  6.16828096]
    1: Leaf node at depth 6 has 16 points with origin [-6.62296295 29.94105975  6.16828096]
```

```
    4: Leaf node at depth 6 has 777 points with origin [-6.6387442  29.94105975  6.18406221]
    5: Leaf node at depth 6 has 626 points with origin [-6.62296295 29.94105975  6.18406221]
  6: Internal node at depth 5 has 8 children and 8131 points ([-6.6703067  29.97262225  6.16828096])
    0: Leaf node at depth 6 has 1326 points with origin [-6.6703067  29.97262225  6.16828096]
    1: Leaf node at depth 6 has 1269 points with origin [-6.65452545 29.97262225  6.16828096]
    2: Leaf node at depth 6 has 907 points with origin [-6.6703067  29.9884035   6.16828096]
    3: Leaf node at depth 6 has 1001 points with origin [-6.65452545 29.9884035   6.16828096]
    4: Leaf node at depth 6 has 1267 points with origin [-6.6703067  29.97262225  6.18406221]
    5: Leaf node at depth 6 has 1015 points with origin [-6.65452545 29.97262225  6.18406221]
    6: Leaf node at depth 6 has 678 points with origin [-6.6703067  29.9884035   6.18406221]
    7: Leaf node at depth 6 has 668 points with origin [-6.65452545 29.9884035   6.18406221]
  7: Internal node at depth 5 has 8 children and 8437 points ([-6.6387442  29.97262225  6.16828096])
    0: Leaf node at depth 6 has 1101 points with origin [-6.6387442  29.97262225  6.16828096]
    1: Leaf node at depth 6 has 1038 points with origin [-6.62296295 29.97262225  6.16828096]
    2: Leaf node at depth 6 has 1235 points with origin [-6.6387442  29.9884035   6.16828096]
    3: Leaf node at depth 6 has 1118 points with origin [-6.62296295 29.9884035   6.16828096]
    4: Leaf node at depth 6 has 1080 points with origin [-6.6387442  29.97262225  6.18406221]
    5: Leaf node at depth 6 has 1282 points with origin [-6.62296295 29.97262225  6.18406221]
    6: Leaf node at depth 6 has 879 points with origin [-6.6387442  29.9884035   6.18406221]
    7: Leaf node at depth 6 has 704 points with origin [-6.62296295 29.9884035   6.18406221]
6: Internal node at depth 4 has 5 children and 15489 points ([-6.7334317  30.00418475  6.13671846])
  0: Internal node at depth 5 has 7 children and 5782 points ([-6.7334317  30.00418475  6.13671846])
    0: Leaf node at depth 6 has 578 points with origin [-6.7334317  30.00418475  6.13671846]
    1: Leaf node at depth 6 has 1782 points with origin [-6.71765045 30.00418475  6.13671846]
    2: Leaf node at depth 6 has 715 points with origin [-6.7334317  30.019966    6.13671846]
    3: Leaf node at depth 6 has 1295 points with origin [-6.71765045 30.019966    6.13671846]
    5: Leaf node at depth 6 has 599 points with origin [-6.71765045 30.00418475  6.15249971]
    6: Leaf node at depth 6 has 798 points with origin [-6.7334317  30.019966    6.15249971]
    7: Leaf node at depth 6 has 15 points with origin [-6.71765045 30.019966    6.15249971]
  1: Internal node at depth 5 has 5 children and 1429 points ([-6.7018692  30.00418475  6.13671846])
    0: Leaf node at depth 6 has 755 points with origin [-6.7018692  30.00418475  6.13671846]
    1: Leaf node at depth 6 has 325 points with origin [-6.68608795 30.00418475  6.13671846]
    2: Leaf node at depth 6 has 67 points with origin [-6.7018692  30.019966    6.13671846]
    4: Leaf node at depth 6 has 168 points with origin [-6.7018692  30.00418475  6.15249971]
    5: Leaf node at depth 6 has 114 points with origin [-6.68608795 30.00418475  6.15249971]
  2: Internal node at depth 5 has 4 children and 4268 points ([-6.7334317  30.03574725  6.13671846])
    0: Leaf node at depth 6 has 2100 points with origin [-6.7334317  30.03574725  6.13671846]
    1: Leaf node at depth 6 has 1286 points with origin [-6.71765045 30.03574725  6.13671846]
    4: Leaf node at depth 6 has 753 points with origin [-6.7334317  30.03574725  6.15249971]
    5: Leaf node at depth 6 has 129 points with origin [-6.71765045 30.03574725  6.15249971]
  3: Internal node at depth 5 has 6 children and 3413 points ([-6.7018692  30.03574725  6.13671846])
    0: Leaf node at depth 6 has 1928 points with origin [-6.7018692  30.03574725  6.13671846]
    1: Leaf node at depth 6 has 838 points with origin [-6.68608795 30.03574725  6.13671846]
    2: Leaf node at depth 6 has 208 points with origin [-6.7018692  30.0515285   6.13671846]
    3: Leaf node at depth 6 has 9 points with origin [-6.68608795 30.0515285   6.13671846]
    4: Leaf node at depth 6 has 404 points with origin [-6.7018692  30.03574725  6.15249971]
    6: Leaf node at depth 6 has 26 points with origin [-6.7018692  30.0515285   6.15249971]
  5: Internal node at depth 5 has 2 children and 597 points ([-6.7018692  30.00418475  6.16828096])
    1: Leaf node at depth 6 has 363 points with origin [-6.68608795 30.00418475  6.16828096]
    5: Leaf node at depth 6 has 234 points with origin [-6.68608795 30.00418475  6.18406221]
7: Internal node at depth 4 has 4 children and 5732 points ([-6.6703067  30.00418475  6.13671846])
  0: Internal node at depth 5 has 2 children and 799 points ([-6.6703067  30.00418475  6.13671846])
    0: Leaf node at depth 6 has 411 points with origin [-6.6703067  30.00418475  6.13671846]
    1: Leaf node at depth 6 has 388 points with origin [-6.65452545 30.00418475  6.13671846]
  1: Internal node at depth 5 has 4 children and 3200 points ([-6.6387442  30.00418475  6.13671846])
    0: Leaf node at depth 6 has 710 points with origin [-6.6387442  30.00418475  6.13671846]
```

```
        1: Leaf node at depth 6 has 779 points with origin [-6.62296295 30.00418475  6.13671846]
        4: Leaf node at depth 6 has 759 points with origin [-6.6387442  30.00418475  6.15249971]
        5: Leaf node at depth 6 has 952 points with origin [-6.62296295 30.00418475  6.15249971]
    4: Internal node at depth 5 has 2 children and 753 points ([-6.6703067  30.00418475  6.16828096])
        0: Leaf node at depth 6 has 368 points with origin [-6.6703067  30.00418475  6.16828096]
        1: Leaf node at depth 6 has 385 points with origin [-6.65452545 30.00418475  6.16828096]
    5: Internal node at depth 5 has 2 children and 980 points ([-6.6387442  30.00418475  6.16828096])
        0: Leaf node at depth 6 has 502 points with origin [-6.6387442  30.00418475  6.16828096]
        1: Leaf node at depth 6 has 478 points with origin [-6.62296295 30.00418475  6.16828096]
1: Internal node at depth 3 has 8 children and 93935 points ([-6.6071817  29.94105975  6.07359346])
  0: Internal node at depth 4 has 2 children and 791 points ([-6.6071817  29.94105975  6.07359346])
    6: Internal node at depth 5 has 4 children and 785 points ([-6.6071817  29.97262225  6.10515596])
        4: Leaf node at depth 6 has 113 points with origin [-6.6071817  29.97262225  6.12093721]
        5: Leaf node at depth 6 has 7 points with origin [-6.59140045 29.97262225  6.12093721]
        6: Leaf node at depth 6 has 627 points with origin [-6.6071817  29.9884035   6.12093721]
        7: Leaf node at depth 6 has 38 points with origin [-6.59140045 29.9884035   6.12093721]
    7: Internal node at depth 5 has 2 children and 6 points ([-6.5756192  29.97262225  6.10515596])
  1: Internal node at depth 4 has 2 children and 1175 points ([-6.5440567  29.94105975  6.07359346])
    6: Internal node at depth 5 has 3 children and 77 points ([-6.5440567  29.97262225  6.10515596])
        4: Leaf node at depth 6 has 32 points with origin [-6.5440567  29.97262225  6.12093721]
        5: Leaf node at depth 6 has 8 points with origin [-6.52827545 29.97262225  6.12093721]
        7: Leaf node at depth 6 has 37 points with origin [-6.52827545 29.9884035   6.12093721]
    7: Internal node at depth 5 has 2 children and 1098 points ([-6.5124942  29.97262225  6.10515596])
        5: Leaf node at depth 6 has 479 points with origin [-6.49671295 29.97262225  6.12093721]
        7: Leaf node at depth 6 has 619 points with origin [-6.49671295 29.9884035   6.12093721]
  2: Internal node at depth 4 has 4 children and 13577 points ([-6.6071817  30.00418475  6.07359346])
    4: Internal node at depth 5 has 4 children and 3275 points ([-6.6071817  30.00418475  6.10515596])
        4: Leaf node at depth 6 has 1280 points with origin [-6.6071817  30.00418475  6.12093721]
        5: Leaf node at depth 6 has 190 points with origin [-6.59140045 30.00418475  6.12093721]
        6: Leaf node at depth 6 has 1296 points with origin [-6.6071817  30.019966    6.12093721]
        7: Leaf node at depth 6 has 509 points with origin [-6.59140045 30.019966    6.12093721]
    5: Internal node at depth 5 has 5 children and 2121 points ([-6.5756192  30.00418475  6.10515596])
        3: Leaf node at depth 6 has 1 points with origin [-6.55983795 30.019966    6.10515596]
        4: Leaf node at depth 6 has 282 points with origin [-6.5756192  30.00418475  6.12093721]
        5: Leaf node at depth 6 has 182 points with origin [-6.55983795 30.00418475  6.12093721]
        6: Leaf node at depth 6 has 695 points with origin [-6.5756192  30.019966    6.12093721]
        7: Leaf node at depth 6 has 961 points with origin [-6.55983795 30.019966    6.12093721]
    6: Internal node at depth 5 has 4 children and 3712 points ([-6.6071817  30.03574725  6.10515596])
        4: Leaf node at depth 6 has 1095 points with origin [-6.6071817  30.03574725  6.12093721]
        5: Leaf node at depth 6 has 1051 points with origin [-6.59140045 30.03574725  6.12093721]
        6: Leaf node at depth 6 has 722 points with origin [-6.6071817  30.0515285   6.12093721]
        7: Leaf node at depth 6 has 844 points with origin [-6.59140045 30.0515285   6.12093721]
    7: Internal node at depth 5 has 4 children and 4469 points ([-6.5756192  30.03574725  6.10515596])
        4: Leaf node at depth 6 has 1204 points with origin [-6.5756192  30.03574725  6.12093721]
        5: Leaf node at depth 6 has 1589 points with origin [-6.55983795 30.03574725  6.12093721]
        6: Leaf node at depth 6 has 787 points with origin [-6.5756192  30.0515285   6.12093721]
        7: Leaf node at depth 6 has 889 points with origin [-6.55983795 30.0515285   6.12093721]
  3: Internal node at depth 4 has 4 children and 10701 points ([-6.5440567  30.00418475  6.07359346])
    4: Internal node at depth 5 has 8 children and 2119 points ([-6.5440567  30.00418475  6.10515596])
        0: Leaf node at depth 6 has 25 points with origin [-6.5440567  30.00418475  6.10515596]
        1: Leaf node at depth 6 has 7 points with origin [-6.52827545 30.00418475  6.10515596]
        2: Leaf node at depth 6 has 50 points with origin [-6.5440567  30.019966    6.10515596]
        3: Leaf node at depth 6 has 5 points with origin [-6.52827545 30.019966    6.10515596]
        4: Leaf node at depth 6 has 441 points with origin [-6.5440567  30.00418475  6.12093721]
        5: Leaf node at depth 6 has 638 points with origin [-6.52827545 30.00418475  6.12093721]
        6: Leaf node at depth 6 has 522 points with origin [-6.5440567  30.019966    6.12093721]
```

7: Leaf node at depth 6 has 431 points with origin [-6.52827545 30.019966 6.12093721]
5: Internal node at depth 5 has 4 children and 1983 points ([-6.5124942 30.00418475 6.10515596])
4: Leaf node at depth 6 has 569 points with origin [-6.5124942 30.00418475 6.12093721]
5: Leaf node at depth 6 has 574 points with origin [-6.49671295 30.00418475 6.12093721]
6: Leaf node at depth 6 has 296 points with origin [-6.5124942 30.019966 6.12093721]
7: Leaf node at depth 6 has 544 points with origin [-6.49671295 30.019966 6.12093721]
6: Internal node at depth 5 has 6 children and 3089 points ([-6.5440567 30.03574725 6.10515596])
1: Leaf node at depth 6 has 2 points with origin [-6.52827545 30.03574725 6.10515596]
3: Leaf node at depth 6 has 1 points with origin [-6.52827545 30.0515285 6.10515596]
4: Leaf node at depth 6 has 791 points with origin [-6.5440567 30.03574725 6.12093721]
5: Leaf node at depth 6 has 644 points with origin [-6.52827545 30.03574725 6.12093721]
6: Leaf node at depth 6 has 801 points with origin [-6.5440567 30.0515285 6.12093721]
7: Leaf node at depth 6 has 850 points with origin [-6.52827545 30.0515285 6.12093721]
7: Internal node at depth 5 has 4 children and 3510 points ([-6.5124942 30.03574725 6.10515596])
4: Leaf node at depth 6 has 319 points with origin [-6.5124942 30.03574725 6.12093721]
5: Leaf node at depth 6 has 499 points with origin [-6.49671295 30.03574725 6.12093721]
6: Leaf node at depth 6 has 1314 points with origin [-6.5124942 30.0515285 6.12093721]
7: Leaf node at depth 6 has 1378 points with origin [-6.49671295 30.0515285 6.12093721]
4: Internal node at depth 4 has 6 children and 13258 points ([-6.6071817 29.94105975 6.13671846])
2: Internal node at depth 5 has 3 children and 166 points ([-6.6071817 29.97262225 6.13671846])
2: Leaf node at depth 6 has 94 points with origin [-6.6071817 29.9884035 6.13671846]
3: Leaf node at depth 6 has 2 points with origin [-6.59140045 29.9884035 6.13671846]
6: Leaf node at depth 6 has 70 points with origin [-6.6071817 29.9884035 6.15249971]
3: Internal node at depth 5 has 4 children and 39 points ([-6.5756192 29.97262225 6.13671846])
2: Leaf node at depth 6 has 2 points with origin [-6.5756192 29.9884035 6.13671846]
3: Leaf node at depth 6 has 5 points with origin [-6.55983795 29.9884035 6.13671846]
5: Leaf node at depth 6 has 9 points with origin [-6.55983795 29.97262225 6.15249971]
7: Leaf node at depth 6 has 23 points with origin [-6.55983795 29.9884035 6.15249971]
4: Internal node at depth 5 has 3 children and 866 points ([-6.6071817 29.94105975 6.16828096])
0: Leaf node at depth 6 has 15 points with origin [-6.6071817 29.94105975 6.16828096]
4: Leaf node at depth 6 has 523 points with origin [-6.6071817 29.94105975 6.18406221]
5: Leaf node at depth 6 has 328 points with origin [-6.59140045 29.94105975 6.18406221]
5: Internal node at depth 5 has 1 children and 17 points ([-6.5756192 29.94105975 6.16828096])
6: Internal node at depth 5 has 8 children and 6843 points ([-6.6071817 29.97262225 6.16828096])
0: Leaf node at depth 6 has 994 points with origin [-6.6071817 29.97262225 6.16828096]
1: Leaf node at depth 6 has 780 points with origin [-6.59140045 29.97262225 6.16828096]
2: Leaf node at depth 6 has 1290 points with origin [-6.6071817 29.9884035 6.16828096]
3: Leaf node at depth 6 has 831 points with origin [-6.59140045 29.9884035 6.16828096]
4: Leaf node at depth 6 has 1008 points with origin [-6.6071817 29.97262225 6.18406221]
5: Leaf node at depth 6 has 725 points with origin [-6.59140045 29.97262225 6.18406221]
6: Leaf node at depth 6 has 997 points with origin [-6.6071817 29.9884035 6.18406221]
7: Leaf node at depth 6 has 218 points with origin [-6.59140045 29.9884035 6.18406221]
7: Internal node at depth 5 has 7 children and 5327 points ([-6.5756192 29.97262225 6.16828096])
0: Leaf node at depth 6 has 994 points with origin [-6.5756192 29.97262225 6.16828096]
1: Leaf node at depth 6 has 417 points with origin [-6.55983795 29.97262225 6.16828096]
2: Leaf node at depth 6 has 760 points with origin [-6.5756192 29.9884035 6.16828096]
3: Leaf node at depth 6 has 722 points with origin [-6.55983795 29.9884035 6.16828096]
4: Leaf node at depth 6 has 711 points with origin [-6.5756192 29.97262225 6.18406221]
5: Leaf node at depth 6 has 633 points with origin [-6.55983795 29.97262225 6.18406221]
7: Leaf node at depth 6 has 1090 points with origin [-6.55983795 29.9884035 6.18406221]
5: Internal node at depth 4 has 6 children and 2626 points ([-6.5440567 29.94105975 6.13671846])
0: Internal node at depth 5 has 1 children and 2 points ([-6.5440567 29.94105975 6.13671846])
1: Internal node at depth 5 has 1 children and 1 points ([-6.5124942 29.94105975 6.13671846])
2: Internal node at depth 5 has 7 children and 341 points ([-6.5440567 29.97262225 6.13671846])
0: Leaf node at depth 6 has 15 points with origin [-6.5440567 29.97262225 6.13671846]
1: Leaf node at depth 6 has 33 points with origin [-6.52827545 29.97262225 6.13671846]

```
    2: Leaf node at depth 6 has 5 points with origin [-6.5440567  29.9884035   6.13671846]
    3: Leaf node at depth 6 has 99 points with origin [-6.52827545 29.9884035   6.13671846]
    4: Leaf node at depth 6 has 49 points with origin [-6.5440567  29.97262225  6.15249971]
    5: Leaf node at depth 6 has 76 points with origin [-6.52827545 29.97262225  6.15249971]
    7: Leaf node at depth 6 has 64 points with origin [-6.52827545 29.9884035   6.15249971]
  3: Internal node at depth 5 has 7 children and 1420 points ([-6.5124942  29.97262225  6.13671846])
    1: Leaf node at depth 6 has 345 points with origin [-6.49671295 29.97262225  6.13671846]
    2: Leaf node at depth 6 has 22 points with origin [-6.5124942  29.9884035   6.13671846]
    3: Leaf node at depth 6 has 636 points with origin [-6.49671295 29.9884035   6.13671846]
    4: Leaf node at depth 6 has 3 points with origin [-6.5124942  29.97262225  6.15249971]
    5: Leaf node at depth 6 has 186 points with origin [-6.49671295 29.97262225  6.15249971]
    6: Leaf node at depth 6 has 39 points with origin [-6.5124942  29.9884035   6.15249971]
    7: Leaf node at depth 6 has 189 points with origin [-6.49671295 29.9884035   6.15249971]
  6: Internal node at depth 5 has 7 children and 486 points ([-6.5440567  29.97262225  6.16828096])
    0: Leaf node at depth 6 has 88 points with origin [-6.5440567  29.97262225  6.16828096]
    1: Leaf node at depth 6 has 102 points with origin [-6.52827545 29.97262225  6.16828096]
    3: Leaf node at depth 6 has 5 points with origin [-6.52827545 29.9884035   6.16828096]
    4: Leaf node at depth 6 has 104 points with origin [-6.5440567  29.97262225  6.18406221]
    5: Leaf node at depth 6 has 83 points with origin [-6.52827545 29.97262225  6.18406221]
    6: Leaf node at depth 6 has 2 points with origin [-6.5440567  29.9884035   6.18406221]
    7: Leaf node at depth 6 has 102 points with origin [-6.52827545 29.9884035   6.18406221]
  7: Internal node at depth 5 has 7 children and 376 points ([-6.5124942  29.97262225  6.16828096])
    1: Leaf node at depth 6 has 95 points with origin [-6.49671295 29.97262225  6.16828096]
    2: Leaf node at depth 6 has 34 points with origin [-6.5124942  29.9884035   6.16828096]
    3: Leaf node at depth 6 has 69 points with origin [-6.49671295 29.9884035   6.16828096]
    4: Leaf node at depth 6 has 1 points with origin [-6.5124942  29.97262225  6.18406221]
    5: Leaf node at depth 6 has 13 points with origin [-6.49671295 29.97262225  6.18406221]
    6: Leaf node at depth 6 has 133 points with origin [-6.5124942  29.9884035   6.18406221]
    7: Leaf node at depth 6 has 31 points with origin [-6.49671295 29.9884035   6.18406221]
6: Internal node at depth 4 has 7 children and 21399 points ([-6.6071817  30.00418475  6.13671846])
  0: Internal node at depth 5 has 8 children and 6367 points ([-6.6071817  30.00418475  6.13671846])
    0: Leaf node at depth 6 has 1390 points with origin [-6.6071817  30.00418475  6.13671846]
    1: Leaf node at depth 6 has 179 points with origin [-6.59140045 30.00418475  6.13671846]
    2: Leaf node at depth 6 has 801 points with origin [-6.6071817  30.019966    6.13671846]
    3: Leaf node at depth 6 has 375 points with origin [-6.59140045 30.019966    6.13671846]
    4: Leaf node at depth 6 has 1368 points with origin [-6.6071817  30.00418475  6.15249971]
    5: Leaf node at depth 6 has 710 points with origin [-6.59140045 30.00418475  6.15249971]
    6: Leaf node at depth 6 has 716 points with origin [-6.6071817  30.019966    6.15249971]
    7: Leaf node at depth 6 has 828 points with origin [-6.59140045 30.019966    6.15249971]
  1: Internal node at depth 5 has 8 children and 2962 points ([-6.5756192  30.00418475  6.13671846])
    0: Leaf node at depth 6 has 236 points with origin [-6.5756192  30.00418475  6.13671846]
    1: Leaf node at depth 6 has 133 points with origin [-6.55983795 30.00418475  6.13671846]
    2: Leaf node at depth 6 has 370 points with origin [-6.5756192  30.019966    6.13671846]
    3: Leaf node at depth 6 has 581 points with origin [-6.55983795 30.019966    6.13671846]
    4: Leaf node at depth 6 has 231 points with origin [-6.5756192  30.00418475  6.15249971]
    5: Leaf node at depth 6 has 334 points with origin [-6.55983795 30.00418475  6.15249971]
    6: Leaf node at depth 6 has 480 points with origin [-6.5756192  30.019966    6.15249971]
    7: Leaf node at depth 6 has 597 points with origin [-6.55983795 30.019966    6.15249971]
  2: Internal node at depth 5 has 4 children and 2142 points ([-6.6071817  30.03574725  6.13671846])
    0: Leaf node at depth 6 has 480 points with origin [-6.6071817  30.03574725  6.13671846]
    1: Leaf node at depth 6 has 429 points with origin [-6.59140045 30.03574725  6.13671846]
    4: Leaf node at depth 6 has 525 points with origin [-6.6071817  30.03574725  6.15249971]
    5: Leaf node at depth 6 has 708 points with origin [-6.59140045 30.03574725  6.15249971]
  3: Internal node at depth 5 has 6 children and 3206 points ([-6.5756192  30.03574725  6.13671846])
    0: Leaf node at depth 6 has 360 points with origin [-6.5756192  30.03574725  6.13671846]
    1: Leaf node at depth 6 has 595 points with origin [-6.55983795 30.03574725  6.13671846]
```

```
    3: Leaf node at depth 6 has 108 points with origin [-6.55983795 30.0515285   6.13671846]
    4: Leaf node at depth 6 has 812 points with origin [-6.5756192  30.03574725  6.15249971]
    5: Leaf node at depth 6 has 675 points with origin [-6.55983795 30.03574725  6.15249971]
    7: Leaf node at depth 6 has 656 points with origin [-6.55983795 30.0515285   6.15249971]
  4: Internal node at depth 5 has 3 children and 2560 points ([-6.6071817  30.00418475  6.16828096])
    0: Leaf node at depth 6 has 528 points with origin [-6.6071817  30.00418475  6.16828096]
    1: Leaf node at depth 6 has 947 points with origin [-6.59140045 30.00418475  6.16828096]
    3: Leaf node at depth 6 has 1085 points with origin [-6.59140045 30.019966    6.16828096]
  5: Internal node at depth 5 has 4 children and 4161 points ([-6.5756192  30.00418475  6.16828096])
    0: Leaf node at depth 6 has 981 points with origin [-6.5756192  30.00418475  6.16828096]
    1: Leaf node at depth 6 has 1047 points with origin [-6.55983795 30.00418475  6.16828096]
    2: Leaf node at depth 6 has 1133 points with origin [-6.5756192  30.019966    6.16828096]
    5: Leaf node at depth 6 has 1000 points with origin [-6.55983795 30.00418475  6.18406221]
  7: Internal node at depth 5 has 1 children and 1 points ([-6.5756192  30.03574725  6.16828096])
7: Internal node at depth 4 has 8 children and 30408 points ([-6.5440567  30.00418475  6.13671846])
  0: Internal node at depth 5 has 4 children and 2992 points ([-6.5440567  30.00418475  6.13671846])
    0: Leaf node at depth 6 has 382 points with origin [-6.5440567  30.00418475  6.13671846]
    1: Leaf node at depth 6 has 718 points with origin [-6.52827545 30.00418475  6.13671846]
    4: Leaf node at depth 6 has 807 points with origin [-6.5440567  30.00418475  6.15249971]
    5: Leaf node at depth 6 has 1085 points with origin [-6.52827545 30.00418475  6.15249971]
  1: Internal node at depth 5 has 8 children and 5075 points ([-6.5124942  30.00418475  6.13671846])
    0: Leaf node at depth 6 has 600 points with origin [-6.5124942  30.00418475  6.13671846]
    1: Leaf node at depth 6 has 667 points with origin [-6.49671295 30.00418475  6.13671846]
    2: Leaf node at depth 6 has 595 points with origin [-6.5124942  30.019966    6.13671846]
    3: Leaf node at depth 6 has 617 points with origin [-6.49671295 30.019966    6.13671846]
    4: Leaf node at depth 6 has 691 points with origin [-6.5124942  30.00418475  6.15249971]
    5: Leaf node at depth 6 has 553 points with origin [-6.49671295 30.00418475  6.15249971]
    6: Leaf node at depth 6 has 662 points with origin [-6.5124942  30.019966    6.15249971]
    7: Leaf node at depth 6 has 690 points with origin [-6.49671295 30.019966    6.15249971]
  2: Internal node at depth 5 has 7 children and 3071 points ([-6.5440567  30.03574725  6.13671846])
    0: Leaf node at depth 6 has 10 points with origin [-6.5440567  30.03574725  6.13671846]
    1: Leaf node at depth 6 has 134 points with origin [-6.52827545 30.03574725  6.13671846]
    3: Leaf node at depth 6 has 3 points with origin [-6.52827545 30.0515285   6.13671846]
    4: Leaf node at depth 6 has 307 points with origin [-6.5440567  30.03574725  6.15249971]
    5: Leaf node at depth 6 has 503 points with origin [-6.52827545 30.03574725  6.15249971]
    6: Leaf node at depth 6 has 1131 points with origin [-6.5440567  30.0515285   6.15249971]
    7: Leaf node at depth 6 has 983 points with origin [-6.52827545 30.0515285   6.15249971]
  3: Internal node at depth 5 has 8 children and 5206 points ([-6.5124942  30.03574725  6.13671846])
    0: Leaf node at depth 6 has 288 points with origin [-6.5124942  30.03574725  6.13671846]
    1: Leaf node at depth 6 has 601 points with origin [-6.49671295 30.03574725  6.13671846]
    2: Leaf node at depth 6 has 735 points with origin [-6.5124942  30.0515285   6.13671846]
    3: Leaf node at depth 6 has 944 points with origin [-6.49671295 30.0515285   6.13671846]
    4: Leaf node at depth 6 has 548 points with origin [-6.5124942  30.03574725  6.15249971]
    5: Leaf node at depth 6 has 580 points with origin [-6.49671295 30.03574725  6.15249971]
    6: Leaf node at depth 6 has 733 points with origin [-6.5124942  30.0515285   6.15249971]
    7: Leaf node at depth 6 has 777 points with origin [-6.49671295 30.0515285   6.15249971]
  4: Internal node at depth 5 has 4 children and 3548 points ([-6.5440567  30.00418475  6.16828096])
    0: Leaf node at depth 6 has 857 points with origin [-6.5440567  30.00418475  6.16828096]
    1: Leaf node at depth 6 has 901 points with origin [-6.52827545 30.00418475  6.16828096]
    4: Leaf node at depth 6 has 831 points with origin [-6.5440567  30.00418475  6.18406221]
    5: Leaf node at depth 6 has 959 points with origin [-6.52827545 30.00418475  6.18406221]
  5: Internal node at depth 5 has 8 children and 3288 points ([-6.5124942  30.00418475  6.16828096])
    0: Leaf node at depth 6 has 586 points with origin [-6.5124942  30.00418475  6.16828096]
    1: Leaf node at depth 6 has 106 points with origin [-6.49671295 30.00418475  6.16828096]
    2: Leaf node at depth 6 has 636 points with origin [-6.5124942  30.019966    6.16828096]
    3: Leaf node at depth 6 has 387 points with origin [-6.49671295 30.019966    6.16828096]
```

```
        4: Leaf node at depth 6 has 602 points with origin [-6.5124942  30.00418475  6.18406221]
        5: Leaf node at depth 6 has 74 points with origin [-6.49671295 30.00418475  6.18406221]
        6: Leaf node at depth 6 has 694 points with origin [-6.5124942  30.019966    6.18406221]
        7: Leaf node at depth 6 has 203 points with origin [-6.49671295 30.019966    6.18406221]
    6: Internal node at depth 5 has 5 children and 1742 points ([-6.5440567  30.03574725  6.16828096])
        1: Leaf node at depth 6 has 251 points with origin [-6.52827545 30.03574725  6.16828096]
        2: Leaf node at depth 6 has 14 points with origin [-6.5440567  30.0515285   6.16828096]
        3: Leaf node at depth 6 has 748 points with origin [-6.52827545 30.0515285   6.16828096]
        5: Leaf node at depth 6 has 143 points with origin [-6.52827545 30.03574725  6.18406221]
        7: Leaf node at depth 6 has 586 points with origin [-6.52827545 30.0515285   6.18406221]
    7: Internal node at depth 5 has 8 children and 5486 points ([-6.5124942  30.03574725  6.16828096])
        0: Leaf node at depth 6 has 561 points with origin [-6.5124942  30.03574725  6.16828096]
        1: Leaf node at depth 6 has 577 points with origin [-6.49671295 30.03574725  6.16828096]
        2: Leaf node at depth 6 has 818 points with origin [-6.5124942  30.0515285   6.16828096]
        3: Leaf node at depth 6 has 737 points with origin [-6.49671295 30.0515285   6.16828096]
        4: Leaf node at depth 6 has 657 points with origin [-6.5124942  30.03574725  6.18406221]
        5: Leaf node at depth 6 has 539 points with origin [-6.49671295 30.03574725  6.18406221]
        6: Leaf node at depth 6 has 872 points with origin [-6.5124942  30.0515285   6.18406221]
        7: Leaf node at depth 6 has 725 points with origin [-6.49671295 30.0515285   6.18406221]
2: Internal node at depth 3 has 6 children and 51975 points ([-6.7334317  30.06730975  6.07359346])
  0: Internal node at depth 4 has 4 children and 12497 points ([-6.7334317  30.06730975  6.07359346])
    4: Internal node at depth 5 has 4 children and 3505 points ([-6.7334317  30.06730975  6.10515596])
        4: Leaf node at depth 6 has 888 points with origin [-6.7334317  30.06730975  6.12093721]
        5: Leaf node at depth 6 has 952 points with origin [-6.71765045 30.06730975  6.12093721]
        6: Leaf node at depth 6 has 815 points with origin [-6.7334317  30.083091    6.12093721]
        7: Leaf node at depth 6 has 850 points with origin [-6.71765045 30.083091    6.12093721]
    5: Internal node at depth 5 has 4 children and 2839 points ([-6.7018692  30.06730975  6.10515596])
        4: Leaf node at depth 6 has 759 points with origin [-6.7018692  30.06730975  6.12093721]
        5: Leaf node at depth 6 has 637 points with origin [-6.68608795 30.06730975  6.12093721]
        6: Leaf node at depth 6 has 711 points with origin [-6.7018692  30.083091    6.12093721]
        7: Leaf node at depth 6 has 732 points with origin [-6.68608795 30.083091    6.12093721]
    6: Internal node at depth 5 has 4 children and 3179 points ([-6.7334317  30.09887225  6.10515596])
        4: Leaf node at depth 6 has 757 points with origin [-6.7334317  30.09887225  6.12093721]
        5: Leaf node at depth 6 has 762 points with origin [-6.71765045 30.09887225  6.12093721]
        6: Leaf node at depth 6 has 853 points with origin [-6.7334317  30.1146535   6.12093721]
        7: Leaf node at depth 6 has 807 points with origin [-6.71765045 30.1146535   6.12093721]
    7: Internal node at depth 5 has 4 children and 2974 points ([-6.7018692  30.09887225  6.10515596])
        4: Leaf node at depth 6 has 683 points with origin [-6.7018692  30.09887225  6.12093721]
        5: Leaf node at depth 6 has 823 points with origin [-6.68608795 30.09887225  6.12093721]
        6: Leaf node at depth 6 has 710 points with origin [-6.7018692  30.1146535   6.12093721]
        7: Leaf node at depth 6 has 758 points with origin [-6.68608795 30.1146535   6.12093721]
  1: Internal node at depth 4 has 4 children and 11129 points ([-6.6703067  30.06730975  6.07359346])
    4: Internal node at depth 5 has 4 children and 2632 points ([-6.6703067  30.06730975  6.10515596])
        4: Leaf node at depth 6 has 665 points with origin [-6.6703067  30.06730975  6.12093721]
        5: Leaf node at depth 6 has 713 points with origin [-6.65452545 30.06730975  6.12093721]
        6: Leaf node at depth 6 has 636 points with origin [-6.6703067  30.083091    6.12093721]
        7: Leaf node at depth 6 has 618 points with origin [-6.65452545 30.083091    6.12093721]
    5: Internal node at depth 5 has 4 children and 2713 points ([-6.6387442  30.06730975  6.10515596])
        4: Leaf node at depth 6 has 684 points with origin [-6.6387442  30.06730975  6.12093721]
        5: Leaf node at depth 6 has 738 points with origin [-6.62296295 30.06730975  6.12093721]
        6: Leaf node at depth 6 has 648 points with origin [-6.6387442  30.083091    6.12093721]
        7: Leaf node at depth 6 has 643 points with origin [-6.62296295 30.083091    6.12093721]
    6: Internal node at depth 5 has 4 children and 2952 points ([-6.6703067  30.09887225  6.10515596])
        4: Leaf node at depth 6 has 740 points with origin [-6.6703067  30.09887225  6.12093721]
        5: Leaf node at depth 6 has 763 points with origin [-6.65452545 30.09887225  6.12093721]
        6: Leaf node at depth 6 has 738 points with origin [-6.6703067  30.1146535   6.12093721]
```

7: Leaf node at depth 6 has 711 points with origin [-6.65452545 30.1146535 6.12093721]
7: Internal node at depth 5 has 4 children and 2832 points ([-6.6387442 30.09887225 6.10515596])
4: Leaf node at depth 6 has 718 points with origin [-6.6387442 30.09887225 6.12093721]
5: Leaf node at depth 6 has 647 points with origin [-6.62296295 30.09887225 6.12093721]
6: Leaf node at depth 6 has 724 points with origin [-6.6387442 30.1146535 6.12093721]
7: Leaf node at depth 6 has 743 points with origin [-6.62296295 30.1146535 6.12093721]
2: Internal node at depth 4 has 4 children and 14728 points ([-6.7334317 30.13043475 6.07359346])
4: Internal node at depth 5 has 4 children and 3944 points ([-6.7334317 30.13043475 6.10515596])
4: Leaf node at depth 6 has 1388 points with origin [-6.7334317 30.13043475 6.12093721]
5: Leaf node at depth 6 has 927 points with origin [-6.71765045 30.13043475 6.12093721]
6: Leaf node at depth 6 has 837 points with origin [-6.7334317 30.146216 6.12093721]
7: Leaf node at depth 6 has 792 points with origin [-6.71765045 30.146216 6.12093721]
5: Internal node at depth 5 has 4 children and 3445 points ([-6.7018692 30.13043475 6.10515596])
4: Leaf node at depth 6 has 800 points with origin [-6.7018692 30.13043475 6.12093721]
5: Leaf node at depth 6 has 919 points with origin [-6.68608795 30.13043475 6.12093721]
6: Leaf node at depth 6 has 955 points with origin [-6.7018692 30.146216 6.12093721]
7: Leaf node at depth 6 has 771 points with origin [-6.68608795 30.146216 6.12093721]
6: Internal node at depth 5 has 4 children and 3679 points ([-6.7334317 30.16199725 6.10515596])
4: Leaf node at depth 6 has 846 points with origin [-6.7334317 30.16199725 6.12093721]
5: Leaf node at depth 6 has 857 points with origin [-6.71765045 30.16199725 6.12093721]
6: Leaf node at depth 6 has 1034 points with origin [-6.7334317 30.1777785 6.12093721]
7: Leaf node at depth 6 has 942 points with origin [-6.71765045 30.1777785 6.12093721]
7: Internal node at depth 5 has 4 children and 3660 points ([-6.7018692 30.16199725 6.10515596])
4: Leaf node at depth 6 has 1140 points with origin [-6.7018692 30.16199725 6.12093721]
5: Leaf node at depth 6 has 783 points with origin [-6.68608795 30.16199725 6.12093721]
6: Leaf node at depth 6 has 923 points with origin [-6.7018692 30.1777785 6.12093721]
7: Leaf node at depth 6 has 814 points with origin [-6.68608795 30.1777785 6.12093721]
3: Internal node at depth 4 has 4 children and 12710 points ([-6.6703067 30.13043475 6.07359346])
4: Internal node at depth 5 has 4 children and 3060 points ([-6.6703067 30.13043475 6.10515596])
4: Leaf node at depth 6 has 775 points with origin [-6.6703067 30.13043475 6.12093721]
5: Leaf node at depth 6 has 708 points with origin [-6.65452545 30.13043475 6.12093721]
6: Leaf node at depth 6 has 746 points with origin [-6.6703067 30.146216 6.12093721]
7: Leaf node at depth 6 has 831 points with origin [-6.65452545 30.146216 6.12093721]
5: Internal node at depth 5 has 4 children and 3180 points ([-6.6387442 30.13043475 6.10515596])
4: Leaf node at depth 6 has 759 points with origin [-6.6387442 30.13043475 6.12093721]
5: Leaf node at depth 6 has 860 points with origin [-6.62296295 30.13043475 6.12093721]
6: Leaf node at depth 6 has 794 points with origin [-6.6387442 30.146216 6.12093721]
7: Leaf node at depth 6 has 767 points with origin [-6.62296295 30.146216 6.12093721]
6: Internal node at depth 5 has 4 children and 3468 points ([-6.6703067 30.16199725 6.10515596])
4: Leaf node at depth 6 has 762 points with origin [-6.6703067 30.16199725 6.12093721]
5: Leaf node at depth 6 has 1115 points with origin [-6.65452545 30.16199725 6.12093721]
6: Leaf node at depth 6 has 827 points with origin [-6.6703067 30.1777785 6.12093721]
7: Leaf node at depth 6 has 764 points with origin [-6.65452545 30.1777785 6.12093721]
7: Internal node at depth 5 has 4 children and 3002 points ([-6.6387442 30.16199725 6.10515596])
4: Leaf node at depth 6 has 745 points with origin [-6.6387442 30.16199725 6.12093721]
5: Leaf node at depth 6 has 720 points with origin [-6.62296295 30.16199725 6.12093721]
6: Leaf node at depth 6 has 755 points with origin [-6.6387442 30.1777785 6.12093721]
7: Leaf node at depth 6 has 782 points with origin [-6.62296295 30.1777785 6.12093721]
6: Internal node at depth 4 has 3 children and 851 points ([-6.7334317 30.13043475 6.13671846])
0: Internal node at depth 5 has 2 children and 773 points ([-6.7334317 30.13043475 6.13671846])
0: Leaf node at depth 6 has 382 points with origin [-6.7334317 30.13043475 6.13671846]
4: Leaf node at depth 6 has 391 points with origin [-6.7334317 30.13043475 6.15249971]
4: Internal node at depth 5 has 1 children and 62 points ([-6.7334317 30.13043475 6.16828096])
0: Leaf node at depth 6 has 62 points with origin [-6.7334317 30.13043475 6.16828096]
6: Internal node at depth 5 has 1 children and 16 points ([-6.7334317 30.16199725 6.16828096])
7: Internal node at depth 4 has 1 children and 60 points ([-6.6703067 30.13043475 6.13671846])

3: Internal node at depth 5 has 2 children and 60 points ([-6.6387442 30.16199725 6.13671846])
2: Leaf node at depth 6 has 12 points with origin [-6.6387442 30.1777785   6.13671846]
3: Leaf node at depth 6 has 48 points with origin [-6.62296295 30.1777785   6.13671846]
3: Internal node at depth 3 has 8 children and 71609 points ([-6.6071817 30.06730975 6.07359346])
0: Internal node at depth 4 has 4 children and 13612 points ([-6.6071817 30.06730975 6.07359346])
4: Internal node at depth 5 has 4 children and 3182 points ([-6.6071817 30.06730975 6.10515596])
4: Leaf node at depth 6 has 715 points with origin [-6.6071817 30.06730975 6.12093721]
5: Leaf node at depth 6 has 760 points with origin [-6.59140045 30.06730975 6.12093721]
6: Leaf node at depth 6 has 767 points with origin [-6.6071817 30.083091   6.12093721]
7: Leaf node at depth 6 has 940 points with origin [-6.59140045 30.083091   6.12093721]
5: Internal node at depth 5 has 4 children and 3829 points ([-6.5756192 30.06730975 6.10515596])
4: Leaf node at depth 6 has 761 points with origin [-6.5756192 30.06730975 6.12093721]
5: Leaf node at depth 6 has 1164 points with origin [-6.55983795 30.06730975 6.12093721]
6: Leaf node at depth 6 has 1075 points with origin [-6.5756192 30.083091   6.12093721]
7: Leaf node at depth 6 has 829 points with origin [-6.55983795 30.083091   6.12093721]
6: Internal node at depth 5 has 4 children and 3345 points ([-6.6071817 30.09887225 6.10515596])
4: Leaf node at depth 6 has 804 points with origin [-6.6071817 30.09887225 6.12093721]
5: Leaf node at depth 6 has 834 points with origin [-6.59140045 30.09887225 6.12093721]
6: Leaf node at depth 6 has 905 points with origin [-6.6071817 30.1146535   6.12093721]
7: Leaf node at depth 6 has 802 points with origin [-6.59140045 30.1146535   6.12093721]
7: Internal node at depth 5 has 4 children and 3256 points ([-6.5756192 30.09887225 6.10515596])
4: Leaf node at depth 6 has 921 points with origin [-6.5756192 30.09887225 6.12093721]
5: Leaf node at depth 6 has 845 points with origin [-6.55983795 30.09887225 6.12093721]
6: Leaf node at depth 6 has 745 points with origin [-6.5756192 30.1146535   6.12093721]
7: Leaf node at depth 6 has 745 points with origin [-6.55983795 30.1146535   6.12093721]
1: Internal node at depth 4 has 4 children and 13239 points ([-6.5440567 30.06730975 6.07359346])
4: Internal node at depth 5 has 4 children and 3130 points ([-6.5440567 30.06730975 6.10515596])
4: Leaf node at depth 6 has 732 points with origin [-6.5440567 30.06730975 6.12093721]
5: Leaf node at depth 6 has 921 points with origin [-6.52827545 30.06730975 6.12093721]
6: Leaf node at depth 6 has 701 points with origin [-6.5440567 30.083091   6.12093721]
7: Leaf node at depth 6 has 776 points with origin [-6.52827545 30.083091   6.12093721]
5: Internal node at depth 5 has 4 children and 3600 points ([-6.5124942 30.06730975 6.10515596])
4: Leaf node at depth 6 has 1048 points with origin [-6.5124942 30.06730975 6.12093721]
5: Leaf node at depth 6 has 849 points with origin [-6.49671295 30.06730975 6.12093721]
6: Leaf node at depth 6 has 877 points with origin [-6.5124942 30.083091   6.12093721]
7: Leaf node at depth 6 has 826 points with origin [-6.49671295 30.083091   6.12093721]
6: Internal node at depth 5 has 4 children and 3115 points ([-6.5440567 30.09887225 6.10515596])
4: Leaf node at depth 6 has 786 points with origin [-6.5440567 30.09887225 6.12093721]
5: Leaf node at depth 6 has 789 points with origin [-6.52827545 30.09887225 6.12093721]
6: Leaf node at depth 6 has 760 points with origin [-6.5440567 30.1146535   6.12093721]
7: Leaf node at depth 6 has 780 points with origin [-6.52827545 30.1146535   6.12093721]
7: Internal node at depth 5 has 4 children and 3394 points ([-6.5124942 30.09887225 6.10515596])
4: Leaf node at depth 6 has 882 points with origin [-6.5124942 30.09887225 6.12093721]
5: Leaf node at depth 6 has 777 points with origin [-6.49671295 30.09887225 6.12093721]
6: Leaf node at depth 6 has 925 points with origin [-6.5124942 30.1146535   6.12093721]
7: Leaf node at depth 6 has 810 points with origin [-6.49671295 30.1146535   6.12093721]
2: Internal node at depth 4 has 4 children and 12935 points ([-6.6071817 30.13043475 6.07359346])
4: Internal node at depth 5 has 4 children and 3206 points ([-6.6071817 30.13043475 6.10515596])
4: Leaf node at depth 6 has 914 points with origin [-6.6071817 30.13043475 6.12093721]
5: Leaf node at depth 6 has 716 points with origin [-6.59140045 30.13043475 6.12093721]
6: Leaf node at depth 6 has 827 points with origin [-6.6071817 30.146216   6.12093721]
7: Leaf node at depth 6 has 749 points with origin [-6.59140045 30.146216   6.12093721]
5: Internal node at depth 5 has 4 children and 3116 points ([-6.5756192 30.13043475 6.10515596])
4: Leaf node at depth 6 has 787 points with origin [-6.5756192 30.13043475 6.12093721]
5: Leaf node at depth 6 has 706 points with origin [-6.55983795 30.13043475 6.12093721]
6: Leaf node at depth 6 has 884 points with origin [-6.5756192 30.146216   6.12093721]

7: Leaf node at depth 6 has 739 points with origin [-6.55983795 30.146216 6.12093721]
6: Internal node at depth 5 has 4 children and 3473 points ([-6.6071817 30.16199725 6.10515596])
4: Leaf node at depth 6 has 822 points with origin [-6.6071817 30.16199725 6.12093721]
5: Leaf node at depth 6 has 983 points with origin [-6.59140045 30.16199725 6.12093721]
6: Leaf node at depth 6 has 858 points with origin [-6.6071817 30.1777785 6.12093721]
7: Leaf node at depth 6 has 810 points with origin [-6.59140045 30.1777785 6.12093721]
7: Internal node at depth 5 has 4 children and 3140 points ([-6.5756192 30.16199725 6.10515596])
4: Leaf node at depth 6 has 779 points with origin [-6.5756192 30.16199725 6.12093721]
5: Leaf node at depth 6 has 921 points with origin [-6.55983795 30.16199725 6.12093721]
6: Leaf node at depth 6 has 785 points with origin [-6.5756192 30.1777785 6.12093721]
7: Leaf node at depth 6 has 655 points with origin [-6.55983795 30.1777785 6.12093721]
3: Internal node at depth 4 has 4 children and 12910 points ([-6.5440567 30.13043475 6.07359346])
4: Internal node at depth 5 has 4 children and 3522 points ([-6.5440567 30.13043475 6.10515596])
4: Leaf node at depth 6 has 748 points with origin [-6.5440567 30.13043475 6.12093721]
5: Leaf node at depth 6 has 817 points with origin [-6.52827545 30.13043475 6.12093721]
6: Leaf node at depth 6 has 980 points with origin [-6.5440567 30.146216 6.12093721]
7: Leaf node at depth 6 has 977 points with origin [-6.52827545 30.146216 6.12093721]
5: Internal node at depth 5 has 4 children and 3531 points ([-6.5124942 30.13043475 6.10515596])
4: Leaf node at depth 6 has 915 points with origin [-6.5124942 30.13043475 6.12093721]
5: Leaf node at depth 6 has 882 points with origin [-6.49671295 30.13043475 6.12093721]
6: Leaf node at depth 6 has 880 points with origin [-6.5124942 30.146216 6.12093721]
7: Leaf node at depth 6 has 854 points with origin [-6.49671295 30.146216 6.12093721]
6: Internal node at depth 5 has 4 children and 2579 points ([-6.5440567 30.16199725 6.10515596])
4: Leaf node at depth 6 has 1071 points with origin [-6.5440567 30.16199725 6.12093721]
5: Leaf node at depth 6 has 1083 points with origin [-6.52827545 30.16199725 6.12093721]
6: Leaf node at depth 6 has 80 points with origin [-6.5440567 30.1777785 6.12093721]
7: Leaf node at depth 6 has 345 points with origin [-6.52827545 30.1777785 6.12093721]
7: Internal node at depth 5 has 4 children and 3278 points ([-6.5124942 30.16199725 6.10515596])
4: Leaf node at depth 6 has 831 points with origin [-6.5124942 30.16199725 6.12093721]
5: Leaf node at depth 6 has 837 points with origin [-6.49671295 30.16199725 6.12093721]
6: Leaf node at depth 6 has 877 points with origin [-6.5124942 30.1777785 6.12093721]
7: Leaf node at depth 6 has 733 points with origin [-6.49671295 30.1777785 6.12093721]
4: Internal node at depth 4 has 1 children and 1212 points ([-6.6071817 30.06730975 6.13671846])
1: Internal node at depth 5 has 2 children and 1212 points ([-6.5756192 30.06730975 6.13671846])
1: Leaf node at depth 6 has 413 points with origin [-6.55983795 30.06730975 6.13671846]
5: Leaf node at depth 6 has 799 points with origin [-6.55983795 30.06730975 6.15249971]
5: Internal node at depth 4 has 6 children and 4129 points ([-6.5440567 30.06730975 6.13671846])
0: Internal node at depth 5 has 4 children and 2610 points ([-6.5440567 30.06730975 6.13671846])
0: Leaf node at depth 6 has 98 points with origin [-6.5440567 30.06730975 6.13671846]
1: Leaf node at depth 6 has 368 points with origin [-6.52827545 30.06730975 6.13671846]
4: Leaf node at depth 6 has 1074 points with origin [-6.5440567 30.06730975 6.15249971]
5: Leaf node at depth 6 has 1070 points with origin [-6.52827545 30.06730975 6.15249971]
1: Internal node at depth 5 has 2 children and 217 points ([-6.5124942 30.06730975 6.13671846])
0: Leaf node at depth 6 has 129 points with origin [-6.5124942 30.06730975 6.13671846]
4: Leaf node at depth 6 has 88 points with origin [-6.5124942 30.06730975 6.15249971]
3: Internal node at depth 5 has 3 children and 18 points ([-6.5124942 30.09887225 6.13671846])
4: Internal node at depth 5 has 2 children and 1253 points ([-6.5440567 30.06730975 6.16828096])
1: Leaf node at depth 6 has 691 points with origin [-6.52827545 30.06730975 6.16828096]
5: Leaf node at depth 6 has 562 points with origin [-6.52827545 30.06730975 6.18406221]
5: Internal node at depth 5 has 2 children and 25 points ([-6.5124942 30.06730975 6.16828096])
7: Internal node at depth 5 has 1 children and 6 points ([-6.5124942 30.09887225 6.16828096])
6: Internal node at depth 4 has 3 children and 3721 points ([-6.6071817 30.13043475 6.13671846])
2: Internal node at depth 5 has 3 children and 55 points ([-6.6071817 30.16199725 6.13671846])
2: Leaf node at depth 6 has 35 points with origin [-6.6071817 30.1777785 6.13671846]
6: Leaf node at depth 6 has 17 points with origin [-6.6071817 30.1777785 6.15249971]
7: Leaf node at depth 6 has 3 points with origin [-6.59140045 30.1777785 6.15249971]

3: Internal node at depth 5 has 4 children and 2188 points ([-6.5756192 30.16199725 6.13671846])
1: Leaf node at depth 6 has 319 points with origin [-6.55983795 30.16199725 6.13671846]
3: Leaf node at depth 6 has 735 points with origin [-6.55983795 30.1777785   6.13671846]
5: Leaf node at depth 6 has 357 points with origin [-6.55983795 30.16199725 6.15249971]
7: Leaf node at depth 6 has 777 points with origin [-6.55983795 30.1777785   6.15249971]
7: Internal node at depth 5 has 4 children and 1478 points ([-6.5756192 30.16199725 6.16828096])
1: Leaf node at depth 6 has 421 points with origin [-6.55983795 30.16199725 6.16828096]
3: Leaf node at depth 6 has 982 points with origin [-6.55983795 30.1777785   6.16828096]
5: Leaf node at depth 6 has 13 points with origin [-6.55983795 30.16199725 6.18406221]
7: Leaf node at depth 6 has 62 points with origin [-6.55983795 30.1777785   6.18406221]
7: Internal node at depth 4 has 5 children and 9851 points ([-6.5440567 30.13043475 6.13671846])
1: Internal node at depth 5 has 1 children and 8 points ([-6.5124942 30.13043475 6.13671846])
2: Internal node at depth 5 has 7 children and 4232 points ([-6.5440567 30.16199725 6.13671846])
0: Leaf node at depth 6 has 877 points with origin [-6.5440567 30.16199725 6.13671846]
1: Leaf node at depth 6 has 873 points with origin [-6.52827545 30.16199725 6.13671846]
2: Leaf node at depth 6 has 11 points with origin [-6.5440567 30.1777785   6.13671846]
3: Leaf node at depth 6 has 682 points with origin [-6.52827545 30.1777785   6.13671846]
4: Leaf node at depth 6 has 864 points with origin [-6.5440567 30.16199725 6.15249971]
5: Leaf node at depth 6 has 781 points with origin [-6.52827545 30.16199725 6.15249971]
7: Leaf node at depth 6 has 144 points with origin [-6.52827545 30.1777785   6.15249971]
3: Internal node at depth 5 has 2 children and 639 points ([-6.5124942 30.16199725 6.13671846])
4: Leaf node at depth 6 has 119 points with origin [-6.5124942 30.16199725 6.15249971]
6: Leaf node at depth 6 has 520 points with origin [-6.5124942 30.1777785   6.15249971]
6: Internal node at depth 5 has 6 children and 4766 points ([-6.5440567 30.16199725 6.16828096])
0: Leaf node at depth 6 has 910 points with origin [-6.5440567 30.16199725 6.16828096]
1: Leaf node at depth 6 has 831 points with origin [-6.52827545 30.16199725 6.16828096]
2: Leaf node at depth 6 has 1277 points with origin [-6.5440567 30.1777785   6.16828096]
3: Leaf node at depth 6 has 830 points with origin [-6.52827545 30.1777785   6.16828096]
4: Leaf node at depth 6 has 486 points with origin [-6.5440567 30.16199725 6.18406221]
6: Leaf node at depth 6 has 432 points with origin [-6.5440567 30.1777785   6.18406221]
7: Internal node at depth 5 has 2 children and 206 points ([-6.5124942 30.16199725 6.16828096])
0: Leaf node at depth 6 has 48 points with origin [-6.5124942 30.16199725 6.16828096]
2: Leaf node at depth 6 has 158 points with origin [-6.5124942 30.1777785   6.16828096]
4: Internal node at depth 3 has 8 children and 115411 points ([-6.7334317 29.94105975 6.19984346])
0: Internal node at depth 4 has 8 children and 27393 points ([-6.7334317 29.94105975 6.19984346])
0: Internal node at depth 5 has 5 children and 3087 points ([-6.7334317 29.94105975 6.19984346])
0: Leaf node at depth 6 has 710 points with origin [-6.7334317 29.94105975 6.19984346]
1: Leaf node at depth 6 has 813 points with origin [-6.71765045 29.94105975 6.19984346]
4: Leaf node at depth 6 has 794 points with origin [-6.7334317 29.94105975 6.21562471]
5: Leaf node at depth 6 has 767 points with origin [-6.71765045 29.94105975 6.21562471]
7: Leaf node at depth 6 has 3 points with origin [-6.71765045 29.956841   6.21562471]
1: Internal node at depth 5 has 6 children and 3572 points ([-6.7018692 29.94105975 6.19984346])
0: Leaf node at depth 6 has 885 points with origin [-6.7018692 29.94105975 6.19984346]
1: Leaf node at depth 6 has 1033 points with origin [-6.68608795 29.94105975 6.19984346]
4: Leaf node at depth 6 has 713 points with origin [-6.7018692 29.94105975 6.21562471]
5: Leaf node at depth 6 has 930 points with origin [-6.68608795 29.94105975 6.21562471]
6: Leaf node at depth 6 has 9 points with origin [-6.7018692 29.956841   6.21562471]
7: Leaf node at depth 6 has 2 points with origin [-6.68608795 29.956841   6.21562471]
2: Internal node at depth 5 has 4 children and 2917 points ([-6.7334317 29.97262225 6.19984346])
0: Leaf node at depth 6 has 730 points with origin [-6.7334317 29.97262225 6.19984346]
1: Leaf node at depth 6 has 729 points with origin [-6.71765045 29.97262225 6.19984346]
4: Leaf node at depth 6 has 729 points with origin [-6.7334317 29.97262225 6.21562471]
5: Leaf node at depth 6 has 729 points with origin [-6.71765045 29.97262225 6.21562471]
3: Internal node at depth 5 has 6 children and 4393 points ([-6.7018692 29.97262225 6.19984346])
0: Leaf node at depth 6 has 762 points with origin [-6.7018692 29.97262225 6.19984346]
1: Leaf node at depth 6 has 774 points with origin [-6.68608795 29.97262225 6.19984346]

```
    3: Leaf node at depth 6 has 668 points with origin [-6.68608795 29.9884035   6.19984346]
    4: Leaf node at depth 6 has 819 points with origin [-6.7018692  29.97262225  6.21562471]
    5: Leaf node at depth 6 has 1002 points with origin [-6.68608795 29.97262225  6.21562471]
    7: Leaf node at depth 6 has 368 points with origin [-6.68608795 29.9884035   6.21562471]
  4: Internal node at depth 5 has 5 children and 3262 points ([-6.7334317  29.94105975  6.23140596])
    0: Leaf node at depth 6 has 877 points with origin [-6.7334317  29.94105975  6.23140596]
    1: Leaf node at depth 6 has 864 points with origin [-6.71765045 29.94105975  6.23140596]
    3: Leaf node at depth 6 has 2 points with origin [-6.71765045 29.956841    6.23140596]
    4: Leaf node at depth 6 has 719 points with origin [-6.7334317  29.94105975  6.24718721]
    5: Leaf node at depth 6 has 800 points with origin [-6.71765045 29.94105975  6.24718721]
  5: Internal node at depth 5 has 4 children and 3202 points ([-6.7018692  29.94105975  6.23140596])
    0: Leaf node at depth 6 has 764 points with origin [-6.7018692  29.94105975  6.23140596]
    1: Leaf node at depth 6 has 739 points with origin [-6.68608795 29.94105975  6.23140596]
    4: Leaf node at depth 6 has 851 points with origin [-6.7018692  29.94105975  6.24718721]
    5: Leaf node at depth 6 has 848 points with origin [-6.68608795 29.94105975  6.24718721]
  6: Internal node at depth 5 has 4 children and 3348 points ([-6.7334317  29.97262225  6.23140596])
    0: Leaf node at depth 6 has 707 points with origin [-6.7334317  29.97262225  6.23140596]
    1: Leaf node at depth 6 has 942 points with origin [-6.71765045 29.97262225  6.23140596]
    4: Leaf node at depth 6 has 866 points with origin [-6.7334317  29.97262225  6.24718721]
    5: Leaf node at depth 6 has 833 points with origin [-6.71765045 29.97262225  6.24718721]
  7: Internal node at depth 5 has 4 children and 3612 points ([-6.7018692  29.97262225  6.23140596])
    0: Leaf node at depth 6 has 983 points with origin [-6.7018692  29.97262225  6.23140596]
    1: Leaf node at depth 6 has 999 points with origin [-6.68608795 29.97262225  6.23140596]
    4: Leaf node at depth 6 has 821 points with origin [-6.7018692  29.97262225  6.24718721]
    5: Leaf node at depth 6 has 809 points with origin [-6.68608795 29.97262225  6.24718721]
1: Internal node at depth 4 has 8 children and 32096 points ([-6.6703067  29.94105975  6.19984346])
  0: Internal node at depth 5 has 5 children and 3240 points ([-6.6703067  29.94105975  6.19984346])
    0: Leaf node at depth 6 has 895 points with origin [-6.6703067  29.94105975  6.19984346]
    1: Leaf node at depth 6 has 927 points with origin [-6.65452545 29.94105975  6.19984346]
    2: Leaf node at depth 6 has 1 points with origin [-6.6703067  29.956841    6.19984346]
    4: Leaf node at depth 6 has 744 points with origin [-6.6703067  29.94105975  6.21562471]
    5: Leaf node at depth 6 has 673 points with origin [-6.65452545 29.94105975  6.21562471]
  1: Internal node at depth 5 has 4 children and 3228 points ([-6.6387442  29.94105975  6.19984346])
    0: Leaf node at depth 6 has 862 points with origin [-6.6387442  29.94105975  6.19984346]
    1: Leaf node at depth 6 has 897 points with origin [-6.62296295 29.94105975  6.19984346]
    4: Leaf node at depth 6 has 739 points with origin [-6.6387442  29.94105975  6.21562471]
    5: Leaf node at depth 6 has 730 points with origin [-6.62296295 29.94105975  6.21562471]
  2: Internal node at depth 5 has 8 children and 6868 points ([-6.6703067  29.97262225  6.19984346])
    0: Leaf node at depth 6 has 994 points with origin [-6.6703067  29.97262225  6.19984346]
    1: Leaf node at depth 6 has 959 points with origin [-6.65452545 29.97262225  6.19984346]
    2: Leaf node at depth 6 has 541 points with origin [-6.6703067  29.9884035   6.19984346]
    3: Leaf node at depth 6 has 780 points with origin [-6.65452545 29.9884035   6.19984346]
    4: Leaf node at depth 6 has 1244 points with origin [-6.6703067  29.97262225  6.21562471]
    5: Leaf node at depth 6 has 1133 points with origin [-6.65452545 29.97262225  6.21562471]
    6: Leaf node at depth 6 has 598 points with origin [-6.6703067  29.9884035   6.21562471]
    7: Leaf node at depth 6 has 619 points with origin [-6.65452545 29.9884035   6.21562471]
  3: Internal node at depth 5 has 8 children and 6143 points ([-6.6387442  29.97262225  6.19984346])
    0: Leaf node at depth 6 has 1017 points with origin [-6.6387442  29.97262225  6.19984346]
    1: Leaf node at depth 6 has 816 points with origin [-6.62296295 29.97262225  6.19984346]
    2: Leaf node at depth 6 has 591 points with origin [-6.6387442  29.9884035   6.19984346]
    3: Leaf node at depth 6 has 1 points with origin [-6.62296295 29.9884035   6.19984346]
    4: Leaf node at depth 6 has 1212 points with origin [-6.6387442  29.97262225  6.21562471]
    5: Leaf node at depth 6 has 1271 points with origin [-6.62296295 29.97262225  6.21562471]
    6: Leaf node at depth 6 has 598 points with origin [-6.6387442  29.9884035   6.21562471]
    7: Leaf node at depth 6 has 637 points with origin [-6.62296295 29.9884035   6.21562471]
  4: Internal node at depth 5 has 4 children and 3078 points ([-6.6703067  29.94105975  6.23140596])
```

```
    0: Leaf node at depth 6 has 703 points with origin [-6.6703067  29.94105975  6.23140596]
    1: Leaf node at depth 6 has 695 points with origin [-6.65452545 29.94105975  6.23140596]
    4: Leaf node at depth 6 has 832 points with origin [-6.6703067  29.94105975  6.24718721]
    5: Leaf node at depth 6 has 848 points with origin [-6.65452545 29.94105975  6.24718721]
  5: Internal node at depth 5 has 4 children and 3161 points ([-6.6387442  29.94105975  6.23140596])
    0: Leaf node at depth 6 has 693 points with origin [-6.6387442  29.94105975  6.23140596]
    1: Leaf node at depth 6 has 698 points with origin [-6.62296295 29.94105975  6.23140596]
    4: Leaf node at depth 6 has 838 points with origin [-6.6387442  29.94105975  6.24718721]
    5: Leaf node at depth 6 has 932 points with origin [-6.62296295 29.94105975  6.24718721]
  6: Internal node at depth 5 has 4 children and 3143 points ([-6.6703067  29.97262225  6.23140596])
    0: Leaf node at depth 6 has 845 points with origin [-6.6703067  29.97262225  6.23140596]
    1: Leaf node at depth 6 has 741 points with origin [-6.65452545 29.97262225  6.23140596]
    4: Leaf node at depth 6 has 747 points with origin [-6.6703067  29.97262225  6.24718721]
    5: Leaf node at depth 6 has 810 points with origin [-6.65452545 29.97262225  6.24718721]
  7: Internal node at depth 5 has 4 children and 3235 points ([-6.6387442  29.97262225  6.23140596])
    0: Leaf node at depth 6 has 863 points with origin [-6.6387442  29.97262225  6.23140596]
    1: Leaf node at depth 6 has 799 points with origin [-6.62296295 29.97262225  6.23140596]
    4: Leaf node at depth 6 has 857 points with origin [-6.6387442  29.97262225  6.24718721]
    5: Leaf node at depth 6 has 716 points with origin [-6.62296295 29.97262225  6.24718721]
2: Internal node at depth 4 has 2 children and 539 points ([-6.7334317  30.00418475  6.19984346])
  1: Internal node at depth 5 has 2 children and 535 points ([-6.7018692  30.00418475  6.19984346])
    1: Leaf node at depth 6 has 241 points with origin [-6.68608795 30.00418475  6.19984346]
    5: Leaf node at depth 6 has 294 points with origin [-6.68608795 30.00418475  6.21562471]
  7: Internal node at depth 5 has 1 children and 4 points ([-6.7018692  30.03574725  6.23140596])
3: Internal node at depth 4 has 4 children and 2173 points ([-6.6703067  30.00418475  6.19984346])
  0: Internal node at depth 5 has 4 children and 1110 points ([-6.6703067  30.00418475  6.19984346])
    1: Leaf node at depth 6 has 2 points with origin [-6.65452545 30.00418475  6.19984346]
    4: Leaf node at depth 6 has 534 points with origin [-6.6703067  30.00418475  6.21562471]
    5: Leaf node at depth 6 has 569 points with origin [-6.65452545 30.00418475  6.21562471]
    7: Leaf node at depth 6 has 5 points with origin [-6.65452545 30.019966    6.21562471]
  1: Internal node at depth 5 has 3 children and 1036 points ([-6.6387442  30.00418475  6.19984346])
    0: Leaf node at depth 6 has 1 points with origin [-6.6387442  30.00418475  6.19984346]
    4: Leaf node at depth 6 has 514 points with origin [-6.6387442  30.00418475  6.21562471]
    5: Leaf node at depth 6 has 521 points with origin [-6.62296295 30.00418475  6.21562471]
  4: Internal node at depth 5 has 1 children and 8 points ([-6.6703067  30.00418475  6.23140596])
  7: Internal node at depth 5 has 3 children and 19 points ([-6.6387442  30.03574725  6.23140596])
4: Internal node at depth 4 has 8 children and 27008 points ([-6.7334317  29.94105975  6.26296846])
  0: Internal node at depth 5 has 5 children and 3848 points ([-6.7334317  29.94105975  6.26296846])
    0: Leaf node at depth 6 has 905 points with origin [-6.7334317  29.94105975  6.26296846]
    1: Leaf node at depth 6 has 931 points with origin [-6.71765045 29.94105975  6.26296846]
    3: Leaf node at depth 6 has 1 points with origin [-6.71765045 29.956841    6.26296846]
    4: Leaf node at depth 6 has 1079 points with origin [-6.7334317  29.94105975  6.27874971]
    5: Leaf node at depth 6 has 932 points with origin [-6.71765045 29.94105975  6.27874971]
  1: Internal node at depth 5 has 4 children and 3387 points ([-6.7018692  29.94105975  6.26296846])
    0: Leaf node at depth 6 has 904 points with origin [-6.7018692  29.94105975  6.26296846]
    1: Leaf node at depth 6 has 714 points with origin [-6.68608795 29.94105975  6.26296846]
    4: Leaf node at depth 6 has 881 points with origin [-6.7018692  29.94105975  6.27874971]
    5: Leaf node at depth 6 has 888 points with origin [-6.68608795 29.94105975  6.27874971]
  2: Internal node at depth 5 has 4 children and 3156 points ([-6.7334317  29.97262225  6.26296846])
    0: Leaf node at depth 6 has 784 points with origin [-6.7334317  29.97262225  6.26296846]
    1: Leaf node at depth 6 has 826 points with origin [-6.71765045 29.97262225  6.26296846]
    4: Leaf node at depth 6 has 780 points with origin [-6.7334317  29.97262225  6.27874971]
    5: Leaf node at depth 6 has 766 points with origin [-6.71765045 29.97262225  6.27874971]
  3: Internal node at depth 5 has 4 children and 3468 points ([-6.7018692  29.97262225  6.26296846])
    0: Leaf node at depth 6 has 839 points with origin [-6.7018692  29.97262225  6.26296846]
    1: Leaf node at depth 6 has 816 points with origin [-6.68608795 29.97262225  6.26296846]
```

4: Leaf node at depth 6 has 796 points with origin [-6.7018692 29.97262225 6.27874971]
5: Leaf node at depth 6 has 1017 points with origin [-6.68608795 29.97262225 6.27874971]
4: Internal node at depth 5 has 4 children and 3854 points ([-6.7334317 29.94105975 6.29453096])
0: Leaf node at depth 6 has 998 points with origin [-6.7334317 29.94105975 6.29453096]
1: Leaf node at depth 6 has 1189 points with origin [-6.71765045 29.94105975 6.29453096]
4: Leaf node at depth 6 has 749 points with origin [-6.7334317 29.94105975 6.31031221]
5: Leaf node at depth 6 has 918 points with origin [-6.71765045 29.94105975 6.31031221]
5: Internal node at depth 5 has 4 children and 3436 points ([-6.7018692 29.94105975 6.29453096])
0: Leaf node at depth 6 has 938 points with origin [-6.7018692 29.94105975 6.29453096]
1: Leaf node at depth 6 has 904 points with origin [-6.68608795 29.94105975 6.29453096]
4: Leaf node at depth 6 has 878 points with origin [-6.7018692 29.94105975 6.31031221]
5: Leaf node at depth 6 has 716 points with origin [-6.68608795 29.94105975 6.31031221]
6: Internal node at depth 5 has 4 children and 2828 points ([-6.7334317 29.97262225 6.29453096])
0: Leaf node at depth 6 has 728 points with origin [-6.7334317 29.97262225 6.29453096]
1: Leaf node at depth 6 has 672 points with origin [-6.71765045 29.97262225 6.29453096]
4: Leaf node at depth 6 has 709 points with origin [-6.7334317 29.97262225 6.31031221]
5: Leaf node at depth 6 has 719 points with origin [-6.71765045 29.97262225 6.31031221]
7: Internal node at depth 5 has 4 children and 3031 points ([-6.7018692 29.97262225 6.29453096])
0: Leaf node at depth 6 has 713 points with origin [-6.7018692 29.97262225 6.29453096]
1: Leaf node at depth 6 has 853 points with origin [-6.68608795 29.97262225 6.29453096]
4: Leaf node at depth 6 has 727 points with origin [-6.7018692 29.97262225 6.31031221]
5: Leaf node at depth 6 has 738 points with origin [-6.68608795 29.97262225 6.31031221]
5: Internal node at depth 4 has 8 children and 25723 points ([-6.6703067 29.94105975 6.26296846])
0: Internal node at depth 5 has 4 children and 3274 points ([-6.6703067 29.94105975 6.26296846])
0: Leaf node at depth 6 has 730 points with origin [-6.6703067 29.94105975 6.26296846]
1: Leaf node at depth 6 has 754 points with origin [-6.65452545 29.94105975 6.26296846]
4: Leaf node at depth 6 has 987 points with origin [-6.6703067 29.94105975 6.27874971]
5: Leaf node at depth 6 has 803 points with origin [-6.65452545 29.94105975 6.27874971]
1: Internal node at depth 5 has 4 children and 2952 points ([-6.6387442 29.94105975 6.26296846])
0: Leaf node at depth 6 has 752 points with origin [-6.6387442 29.94105975 6.26296846]
1: Leaf node at depth 6 has 706 points with origin [-6.62296295 29.94105975 6.26296846]
4: Leaf node at depth 6 has 790 points with origin [-6.6387442 29.94105975 6.27874971]
5: Leaf node at depth 6 has 704 points with origin [-6.62296295 29.94105975 6.27874971]
2: Internal node at depth 5 has 4 children and 3578 points ([-6.6703067 29.97262225 6.26296846])
0: Leaf node at depth 6 has 871 points with origin [-6.6703067 29.97262225 6.26296846]
1: Leaf node at depth 6 has 771 points with origin [-6.65452545 29.97262225 6.26296846]
4: Leaf node at depth 6 has 935 points with origin [-6.6703067 29.97262225 6.27874971]
5: Leaf node at depth 6 has 1001 points with origin [-6.65452545 29.97262225 6.27874971]
3: Internal node at depth 5 has 4 children and 3524 points ([-6.6387442 29.97262225 6.26296846])
0: Leaf node at depth 6 has 754 points with origin [-6.6387442 29.97262225 6.26296846]
1: Leaf node at depth 6 has 742 points with origin [-6.62296295 29.97262225 6.26296846]
4: Leaf node at depth 6 has 1156 points with origin [-6.6387442 29.97262225 6.27874971]
5: Leaf node at depth 6 has 872 points with origin [-6.62296295 29.97262225 6.27874971]
4: Internal node at depth 5 has 4 children and 3272 points ([-6.6703067 29.94105975 6.29453096])
0: Leaf node at depth 6 has 904 points with origin [-6.6703067 29.94105975 6.29453096]
1: Leaf node at depth 6 has 809 points with origin [-6.65452545 29.94105975 6.29453096]
4: Leaf node at depth 6 has 734 points with origin [-6.6703067 29.94105975 6.31031221]
5: Leaf node at depth 6 has 825 points with origin [-6.65452545 29.94105975 6.31031221]
5: Internal node at depth 5 has 5 children and 2874 points ([-6.6387442 29.94105975 6.29453096])
0: Leaf node at depth 6 has 731 points with origin [-6.6387442 29.94105975 6.29453096]
1: Leaf node at depth 6 has 713 points with origin [-6.62296295 29.94105975 6.29453096]
2: Leaf node at depth 6 has 1 points with origin [-6.6387442 29.956841   6.29453096]
4: Leaf node at depth 6 has 727 points with origin [-6.6387442 29.94105975 6.31031221]
5: Leaf node at depth 6 has 702 points with origin [-6.62296295 29.94105975 6.31031221]
6: Internal node at depth 5 has 4 children and 3279 points ([-6.6703067 29.97262225 6.29453096])
0: Leaf node at depth 6 has 925 points with origin [-6.6703067 29.97262225 6.29453096]

1: Leaf node at depth 6 has 901 points with origin [-6.65452545 29.97262225 6.29453096]
4: Leaf node at depth 6 has 729 points with origin [-6.6703067 29.97262225 6.31031221]
5: Leaf node at depth 6 has 724 points with origin [-6.65452545 29.97262225 6.31031221]
7: Internal node at depth 5 has 4 children and 2970 points ([-6.6387442 29.97262225 6.29453096])
0: Leaf node at depth 6 has 814 points with origin [-6.6387442 29.97262225 6.29453096]
1: Leaf node at depth 6 has 704 points with origin [-6.62296295 29.97262225 6.29453096]
4: Leaf node at depth 6 has 716 points with origin [-6.6387442 29.97262225 6.31031221]
5: Leaf node at depth 6 has 736 points with origin [-6.62296295 29.97262225 6.31031221]
6: Internal node at depth 4 has 2 children and 24 points ([-6.7334317 30.00418475 6.26296846])
7: Internal node at depth 4 has 4 children and 455 points ([-6.6703067 30.00418475 6.26296846])
2: Internal node at depth 5 has 2 children and 39 points ([-6.6703067 30.03574725 6.26296846])
6: Leaf node at depth 6 has 26 points with origin [-6.6703067 30.0515285   6.27874971]
7: Leaf node at depth 6 has 13 points with origin [-6.65452545 30.0515285   6.27874971]
4: Internal node at depth 5 has 4 children and 136 points ([-6.6703067 30.00418475 6.29453096])
0: Leaf node at depth 6 has 21 points with origin [-6.6703067 30.00418475 6.29453096]
1: Leaf node at depth 6 has 28 points with origin [-6.65452545 30.00418475 6.29453096]
2: Leaf node at depth 6 has 52 points with origin [-6.6703067 30.019966   6.29453096]
3: Leaf node at depth 6 has 35 points with origin [-6.65452545 30.019966   6.29453096]
5: Internal node at depth 5 has 1 children and 31 points ([-6.6387442 30.00418475 6.29453096])
2: Leaf node at depth 6 has 31 points with origin [-6.6387442 30.019966   6.29453096]
6: Internal node at depth 5 has 2 children and 249 points ([-6.6703067 30.03574725 6.29453096])
2: Leaf node at depth 6 has 155 points with origin [-6.6703067 30.0515285   6.29453096]
3: Leaf node at depth 6 has 94 points with origin [-6.65452545 30.0515285   6.29453096]
5: Internal node at depth 3 has 8 children and 149566 points ([-6.6071817 29.94105975 6.19984346])
0: Internal node at depth 4 has 8 children and 28696 points ([-6.6071817 29.94105975 6.19984346])
0: Internal node at depth 5 has 4 children and 3174 points ([-6.6071817 29.94105975 6.19984346])
0: Leaf node at depth 6 has 845 points with origin [-6.6071817 29.94105975 6.19984346]
1: Leaf node at depth 6 has 815 points with origin [-6.59140045 29.94105975 6.19984346]
4: Leaf node at depth 6 has 759 points with origin [-6.6071817 29.94105975 6.21562471]
5: Leaf node at depth 6 has 755 points with origin [-6.59140045 29.94105975 6.21562471]
1: Internal node at depth 5 has 4 children and 2662 points ([-6.5756192 29.94105975 6.19984346])
0: Leaf node at depth 6 has 548 points with origin [-6.5756192 29.94105975 6.19984346]
1: Leaf node at depth 6 has 434 points with origin [-6.55983795 29.94105975 6.19984346]
4: Leaf node at depth 6 has 936 points with origin [-6.5756192 29.94105975 6.21562471]
5: Leaf node at depth 6 has 744 points with origin [-6.55983795 29.94105975 6.21562471]
2: Internal node at depth 5 has 8 children and 5715 points ([-6.6071817 29.97262225 6.19984346])
0: Leaf node at depth 6 has 747 points with origin [-6.6071817 29.97262225 6.19984346]
1: Leaf node at depth 6 has 723 points with origin [-6.59140045 29.97262225 6.19984346]
2: Leaf node at depth 6 has 655 points with origin [-6.6071817 29.9884035   6.19984346]
3: Leaf node at depth 6 has 464 points with origin [-6.59140045 29.9884035   6.19984346]
4: Leaf node at depth 6 has 1137 points with origin [-6.6071817 29.97262225 6.21562471]
5: Leaf node at depth 6 has 1070 points with origin [-6.59140045 29.97262225 6.21562471]
6: Leaf node at depth 6 has 536 points with origin [-6.6071817 29.9884035   6.21562471]
7: Leaf node at depth 6 has 383 points with origin [-6.59140045 29.9884035   6.21562471]
3: Internal node at depth 5 has 8 children and 4819 points ([-6.5756192 29.97262225 6.19984346])
0: Leaf node at depth 6 has 841 points with origin [-6.5756192 29.97262225 6.19984346]
1: Leaf node at depth 6 has 620 points with origin [-6.55983795 29.97262225 6.19984346]
2: Leaf node at depth 6 has 459 points with origin [-6.5756192 29.9884035   6.19984346]
3: Leaf node at depth 6 has 782 points with origin [-6.55983795 29.9884035   6.19984346]
4: Leaf node at depth 6 has 988 points with origin [-6.5756192 29.97262225 6.21562471]
5: Leaf node at depth 6 has 985 points with origin [-6.55983795 29.97262225 6.21562471]
6: Leaf node at depth 6 has 100 points with origin [-6.5756192 29.9884035   6.21562471]
7: Leaf node at depth 6 has 44 points with origin [-6.55983795 29.9884035   6.21562471]
4: Internal node at depth 5 has 5 children and 3141 points ([-6.6071817 29.94105975 6.23140596])
0: Leaf node at depth 6 has 726 points with origin [-6.6071817 29.94105975 6.23140596]
1: Leaf node at depth 6 has 746 points with origin [-6.59140045 29.94105975 6.23140596]

3: Leaf node at depth 6 has 1 points with origin [-6.59140045 29.956841 6.23140596]
4: Leaf node at depth 6 has 865 points with origin [-6.6071817 29.94105975 6.24718721]
5: Leaf node at depth 6 has 803 points with origin [-6.59140045 29.94105975 6.24718721]
5: Internal node at depth 5 has 4 children and 3006 points ([-6.5756192 29.94105975 6.23140596])
0: Leaf node at depth 6 has 736 points with origin [-6.5756192 29.94105975 6.23140596]
1: Leaf node at depth 6 has 777 points with origin [-6.55983795 29.94105975 6.23140596]
4: Leaf node at depth 6 has 758 points with origin [-6.5756192 29.94105975 6.24718721]
5: Leaf node at depth 6 has 735 points with origin [-6.55983795 29.94105975 6.24718721]
6: Internal node at depth 5 has 4 children and 2953 points ([-6.6071817 29.97262225 6.23140596])
0: Leaf node at depth 6 has 765 points with origin [-6.6071817 29.97262225 6.23140596]
1: Leaf node at depth 6 has 703 points with origin [-6.59140045 29.97262225 6.23140596]
4: Leaf node at depth 6 has 755 points with origin [-6.6071817 29.97262225 6.24718721]
5: Leaf node at depth 6 has 730 points with origin [-6.59140045 29.97262225 6.24718721]
7: Internal node at depth 5 has 4 children and 3226 points ([-6.5756192 29.97262225 6.23140596])
0: Leaf node at depth 6 has 718 points with origin [-6.5756192 29.97262225 6.23140596]
1: Leaf node at depth 6 has 790 points with origin [-6.55983795 29.97262225 6.23140596]
4: Leaf node at depth 6 has 768 points with origin [-6.5756192 29.97262225 6.24718721]
5: Leaf node at depth 6 has 950 points with origin [-6.55983795 29.97262225 6.24718721]
1: Internal node at depth 4 has 8 children and 20010 points ([-6.5440567 29.94105975 6.19984346])
0: Internal node at depth 5 has 5 children and 748 points ([-6.5440567 29.94105975 6.19984346])
0: Leaf node at depth 6 has 53 points with origin [-6.5440567 29.94105975 6.19984346]
2: Leaf node at depth 6 has 21 points with origin [-6.5440567 29.956841 6.19984346]
4: Leaf node at depth 6 has 581 points with origin [-6.5440567 29.94105975 6.21562471]
5: Leaf node at depth 6 has 90 points with origin [-6.52827545 29.94105975 6.21562471]
6: Leaf node at depth 6 has 3 points with origin [-6.5440567 29.956841 6.21562471]
1: Internal node at depth 5 has 4 children and 308 points ([-6.5124942 29.94105975 6.19984346])
3: Leaf node at depth 6 has 108 points with origin [-6.49671295 29.956841 6.19984346]
4: Leaf node at depth 6 has 8 points with origin [-6.5124942 29.94105975 6.21562471]
5: Leaf node at depth 6 has 74 points with origin [-6.49671295 29.94105975 6.21562471]
7: Leaf node at depth 6 has 118 points with origin [-6.49671295 29.956841 6.21562471]
2: Internal node at depth 5 has 6 children and 5114 points ([-6.5440567 29.97262225 6.19984346])
0: Leaf node at depth 6 has 906 points with origin [-6.5440567 29.97262225 6.19984346]
1: Leaf node at depth 6 has 1233 points with origin [-6.52827545 29.97262225 6.19984346]
2: Leaf node at depth 6 has 555 points with origin [-6.5440567 29.9884035 6.19984346]
3: Leaf node at depth 6 has 657 points with origin [-6.52827545 29.9884035 6.19984346]
4: Leaf node at depth 6 has 978 points with origin [-6.5440567 29.97262225 6.21562471]
5: Leaf node at depth 6 has 785 points with origin [-6.52827545 29.97262225 6.21562471]
3: Internal node at depth 5 has 5 children and 3327 points ([-6.5124942 29.97262225 6.19984346])
0: Leaf node at depth 6 has 815 points with origin [-6.5124942 29.97262225 6.19984346]
2: Leaf node at depth 6 has 922 points with origin [-6.5124942 29.9884035 6.19984346]
3: Leaf node at depth 6 has 3 points with origin [-6.49671295 29.9884035 6.19984346]
4: Leaf node at depth 6 has 800 points with origin [-6.5124942 29.97262225 6.21562471]
6: Leaf node at depth 6 has 787 points with origin [-6.5124942 29.9884035 6.21562471]
4: Internal node at depth 5 has 5 children and 3076 points ([-6.5440567 29.94105975 6.23140596])
0: Leaf node at depth 6 has 817 points with origin [-6.5440567 29.94105975 6.23140596]
1: Leaf node at depth 6 has 685 points with origin [-6.52827545 29.94105975 6.23140596]
2: Leaf node at depth 6 has 5 points with origin [-6.5440567 29.956841 6.23140596]
4: Leaf node at depth 6 has 825 points with origin [-6.5440567 29.94105975 6.24718721]
5: Leaf node at depth 6 has 744 points with origin [-6.52827545 29.94105975 6.24718721]
5: Internal node at depth 5 has 4 children and 945 points ([-6.5124942 29.94105975 6.23140596])
0: Leaf node at depth 6 has 235 points with origin [-6.5124942 29.94105975 6.23140596]
1: Leaf node at depth 6 has 135 points with origin [-6.49671295 29.94105975 6.23140596]
4: Leaf node at depth 6 has 414 points with origin [-6.5124942 29.94105975 6.24718721]
5: Leaf node at depth 6 has 161 points with origin [-6.49671295 29.94105975 6.24718721]
6: Internal node at depth 5 has 4 children and 3107 points ([-6.5440567 29.97262225 6.23140596])
0: Leaf node at depth 6 has 841 points with origin [-6.5440567 29.97262225 6.23140596]

```
    1: Leaf node at depth 6 has 754 points with origin [-6.52827545 29.97262225  6.23140596]
    4: Leaf node at depth 6 has 757 points with origin [-6.5440567  29.97262225  6.24718721]
    5: Leaf node at depth 6 has 755 points with origin [-6.52827545 29.97262225  6.24718721]
  7: Internal node at depth 5 has 6 children and 3385 points ([-6.5124942  29.97262225  6.23140596])
    0: Leaf node at depth 6 has 817 points with origin [-6.5124942  29.97262225  6.23140596]
    2: Leaf node at depth 6 has 791 points with origin [-6.5124942  29.9884035   6.23140596]
    4: Leaf node at depth 6 has 871 points with origin [-6.5124942  29.97262225  6.24718721]
    5: Leaf node at depth 6 has 69 points with origin [-6.49671295 29.97262225  6.24718721]
    6: Leaf node at depth 6 has 765 points with origin [-6.5124942  29.9884035   6.24718721]
    7: Leaf node at depth 6 has 72 points with origin [-6.49671295 29.9884035   6.24718721]
2: Internal node at depth 4 has 4 children and 2943 points ([-6.6071817  30.00418475  6.19984346])
  0: Internal node at depth 5 has 2 children and 1156 points ([-6.6071817  30.00418475  6.19984346])
    4: Leaf node at depth 6 has 599 points with origin [-6.6071817  30.00418475  6.21562471]
    5: Leaf node at depth 6 has 557 points with origin [-6.59140045 30.00418475  6.21562471]
  1: Internal node at depth 5 has 4 children and 1458 points ([-6.5756192  30.00418475  6.19984346])
    0: Leaf node at depth 6 has 9 points with origin [-6.5756192  30.00418475  6.19984346]
    1: Leaf node at depth 6 has 624 points with origin [-6.55983795 30.00418475  6.19984346]
    4: Leaf node at depth 6 has 613 points with origin [-6.5756192  30.00418475  6.21562471]
    5: Leaf node at depth 6 has 212 points with origin [-6.55983795 30.00418475  6.21562471]
  5: Internal node at depth 5 has 3 children and 76 points ([-6.5756192  30.00418475  6.23140596])
    0: Leaf node at depth 6 has 7 points with origin [-6.5756192  30.00418475  6.23140596]
    2: Leaf node at depth 6 has 67 points with origin [-6.5756192  30.019966    6.23140596]
    6: Leaf node at depth 6 has 2 points with origin [-6.5756192  30.019966    6.24718721]
  7: Internal node at depth 5 has 4 children and 253 points ([-6.5756192  30.03574725  6.23140596])
    0: Leaf node at depth 6 has 81 points with origin [-6.5756192  30.03574725  6.23140596]
    2: Leaf node at depth 6 has 38 points with origin [-6.5756192  30.0515285   6.23140596]
    4: Leaf node at depth 6 has 63 points with origin [-6.5756192  30.03574725  6.24718721]
    6: Leaf node at depth 6 has 71 points with origin [-6.5756192  30.0515285   6.24718721]
3: Internal node at depth 4 has 5 children and 21618 points ([-6.5440567  30.00418475  6.19984346])
  0: Internal node at depth 5 has 3 children and 1044 points ([-6.5440567  30.00418475  6.19984346])
    0: Leaf node at depth 6 has 505 points with origin [-6.5440567  30.00418475  6.19984346]
    1: Leaf node at depth 6 has 536 points with origin [-6.52827545 30.00418475  6.19984346]
    3: Leaf node at depth 6 has 3 points with origin [-6.52827545 30.019966    6.19984346]
  1: Internal node at depth 5 has 7 children and 3830 points ([-6.5124942  30.00418475  6.19984346])
    0: Leaf node at depth 6 has 968 points with origin [-6.5124942  30.00418475  6.19984346]
    1: Leaf node at depth 6 has 49 points with origin [-6.49671295 30.00418475  6.19984346]
    2: Leaf node at depth 6 has 689 points with origin [-6.5124942  30.019966    6.19984346]
    3: Leaf node at depth 6 has 329 points with origin [-6.49671295 30.019966    6.19984346]
    4: Leaf node at depth 6 has 820 points with origin [-6.5124942  30.00418475  6.21562471]
    6: Leaf node at depth 6 has 710 points with origin [-6.5124942  30.019966    6.21562471]
    7: Leaf node at depth 6 has 265 points with origin [-6.49671295 30.019966    6.21562471]
  3: Internal node at depth 5 has 8 children and 5745 points ([-6.5124942  30.03574725  6.19984346])
    0: Leaf node at depth 6 has 745 points with origin [-6.5124942  30.03574725  6.19984346]
    1: Leaf node at depth 6 has 553 points with origin [-6.49671295 30.03574725  6.19984346]
    2: Leaf node at depth 6 has 775 points with origin [-6.5124942  30.0515285   6.19984346]
    3: Leaf node at depth 6 has 758 points with origin [-6.49671295 30.0515285   6.19984346]
    4: Leaf node at depth 6 has 726 points with origin [-6.5124942  30.03574725  6.21562471]
    5: Leaf node at depth 6 has 642 points with origin [-6.49671295 30.03574725  6.21562471]
    6: Leaf node at depth 6 has 795 points with origin [-6.5124942  30.0515285   6.21562471]
    7: Leaf node at depth 6 has 751 points with origin [-6.49671295 30.0515285   6.21562471]
  5: Internal node at depth 5 has 7 children and 4351 points ([-6.5124942  30.00418475  6.23140596])
    0: Leaf node at depth 6 has 858 points with origin [-6.5124942  30.00418475  6.23140596]
    2: Leaf node at depth 6 has 813 points with origin [-6.5124942  30.019966    6.23140596]
    3: Leaf node at depth 6 has 296 points with origin [-6.49671295 30.019966    6.23140596]
    4: Leaf node at depth 6 has 957 points with origin [-6.5124942  30.00418475  6.24718721]
    5: Leaf node at depth 6 has 79 points with origin [-6.49671295 30.00418475  6.24718721]
```

```
    6: Leaf node at depth 6 has 1133 points with origin [-6.5124942  30.019966    6.24718721]
    7: Leaf node at depth 6 has 215 points with origin [-6.49671295 30.019966    6.24718721]
  7: Internal node at depth 5 has 8 children and 6648 points ([-6.5124942  30.03574725  6.23140596])
    0: Leaf node at depth 6 has 894 points with origin [-6.5124942  30.03574725  6.23140596]
    1: Leaf node at depth 6 has 674 points with origin [-6.49671295 30.03574725  6.23140596]
    2: Leaf node at depth 6 has 941 points with origin [-6.5124942  30.0515285   6.23140596]
    3: Leaf node at depth 6 has 740 points with origin [-6.49671295 30.0515285   6.23140596]
    4: Leaf node at depth 6 has 941 points with origin [-6.5124942  30.03574725  6.24718721]
    5: Leaf node at depth 6 has 593 points with origin [-6.49671295 30.03574725  6.24718721]
    6: Leaf node at depth 6 has 1122 points with origin [-6.5124942  30.0515285   6.24718721]
    7: Leaf node at depth 6 has 743 points with origin [-6.49671295 30.0515285   6.24718721]
4: Internal node at depth 4 has 8 children and 25539 points ([-6.6071817  29.94105975  6.26296846])
  0: Internal node at depth 5 has 4 children and 3015 points ([-6.6071817  29.94105975  6.26296846])
    0: Leaf node at depth 6 has 735 points with origin [-6.6071817  29.94105975  6.26296846]
    1: Leaf node at depth 6 has 743 points with origin [-6.59140045 29.94105975  6.26296846]
    4: Leaf node at depth 6 has 768 points with origin [-6.6071817  29.94105975  6.27874971]
    5: Leaf node at depth 6 has 769 points with origin [-6.59140045 29.94105975  6.27874971]
  1: Internal node at depth 5 has 4 children and 3172 points ([-6.5756192  29.94105975  6.26296846])
    0: Leaf node at depth 6 has 855 points with origin [-6.5756192  29.94105975  6.26296846]
    1: Leaf node at depth 6 has 713 points with origin [-6.55983795 29.94105975  6.26296846]
    4: Leaf node at depth 6 has 827 points with origin [-6.5756192  29.94105975  6.27874971]
    5: Leaf node at depth 6 has 777 points with origin [-6.55983795 29.94105975  6.27874971]
  2: Internal node at depth 5 has 4 children and 3298 points ([-6.6071817  29.97262225  6.26296846])
    0: Leaf node at depth 6 has 918 points with origin [-6.6071817  29.97262225  6.26296846]
    1: Leaf node at depth 6 has 854 points with origin [-6.59140045 29.97262225  6.26296846]
    4: Leaf node at depth 6 has 794 points with origin [-6.6071817  29.97262225  6.27874971]
    5: Leaf node at depth 6 has 732 points with origin [-6.59140045 29.97262225  6.27874971]
  3: Internal node at depth 5 has 4 children and 3352 points ([-6.5756192  29.97262225  6.26296846])
    0: Leaf node at depth 6 has 856 points with origin [-6.5756192  29.97262225  6.26296846]
    1: Leaf node at depth 6 has 735 points with origin [-6.55983795 29.97262225  6.26296846]
    4: Leaf node at depth 6 has 777 points with origin [-6.5756192  29.97262225  6.27874971]
    5: Leaf node at depth 6 has 984 points with origin [-6.55983795 29.97262225  6.27874971]
  4: Internal node at depth 5 has 4 children and 3318 points ([-6.6071817  29.94105975  6.29453096])
    0: Leaf node at depth 6 has 744 points with origin [-6.6071817  29.94105975  6.29453096]
    1: Leaf node at depth 6 has 908 points with origin [-6.59140045 29.94105975  6.29453096]
    4: Leaf node at depth 6 has 735 points with origin [-6.6071817  29.94105975  6.31031221]
    5: Leaf node at depth 6 has 931 points with origin [-6.59140045 29.94105975  6.31031221]
  5: Internal node at depth 5 has 4 children and 3383 points ([-6.5756192  29.94105975  6.29453096])
    0: Leaf node at depth 6 has 764 points with origin [-6.5756192  29.94105975  6.29453096]
    1: Leaf node at depth 6 has 1006 points with origin [-6.55983795 29.94105975  6.29453096]
    4: Leaf node at depth 6 has 839 points with origin [-6.5756192  29.94105975  6.31031221]
    5: Leaf node at depth 6 has 774 points with origin [-6.55983795 29.94105975  6.31031221]
  6: Internal node at depth 5 has 4 children and 2885 points ([-6.6071817  29.97262225  6.29453096])
    0: Leaf node at depth 6 has 749 points with origin [-6.6071817  29.97262225  6.29453096]
    1: Leaf node at depth 6 has 712 points with origin [-6.59140045 29.97262225  6.29453096]
    4: Leaf node at depth 6 has 728 points with origin [-6.6071817  29.97262225  6.31031221]
    5: Leaf node at depth 6 has 696 points with origin [-6.59140045 29.97262225  6.31031221]
  7: Internal node at depth 5 has 5 children and 3116 points ([-6.5756192  29.97262225  6.29453096])
    0: Leaf node at depth 6 has 762 points with origin [-6.5756192  29.97262225  6.29453096]
    1: Leaf node at depth 6 has 861 points with origin [-6.55983795 29.97262225  6.29453096]
    3: Leaf node at depth 6 has 2 points with origin [-6.55983795 29.9884035   6.29453096]
    4: Leaf node at depth 6 has 694 points with origin [-6.5756192  29.97262225  6.31031221]
    5: Leaf node at depth 6 has 797 points with origin [-6.55983795 29.97262225  6.31031221]
```