UCLouvain

École polytechnique de Louvain

Segmentation of pipelines and power cables from point clouds using a deep learning approach

Author: **Jules LESUISSE**
Supervisor: **Pierre SCHAUS**
Readers: **Shahriar AGAAJANI, Achille MORENVILLE**
Academic year 2022–2023
Master [120] in Computer Science

Abstract

This master's thesis aims to develop a novel deep learning approach for the accurate segmentation of pipelines and power cables from point clouds, addressing the challenges of infrastructure monitoring and enhancing construction processes. The research is conducted in collaboration with Space Time, an innovative company specializing in 4D scanning services. The model utilizes a shifting local view mechanism to accurately track the path of pipelines and extract relevant features over a custom dataset. This dataset was meticulously developed to suit the specific requirements of the pipeline and power cable segmentation task. The integration of PointNet, a renowned module for point cloud segmentation, initially showed suboptimal performance. Consequently, a thorough theoretical analysis was conducted, providing extensive evaluations that clearly demonstrate the effectiveness of the model in accurately segmenting pipelines and power cables. The findings contribute to the field of point cloud processing and showcase the lessons learnt in the application of deep learning techniques for complex pattern recognition tasks.

Acknowledgements

I would like to express my sincere gratitude to the individuals who contributed to the realization of this work.

Firstly, I communicate my appreciation to Shahriar Agaajani, the CEO of Space Time, for proposing the thesis in collaboration with his company. I am also grateful to Professor Pierre Schaus for accepting to support this research.

On a practical level, I would like to thank Professor Schaus for his valuable mentorship. His availability, advice, and the time he dedicated to guiding me through this project were instrumental to its success. I would also like to acknowledge the reliable assistance provided by Achille Morenville, whose constant support, availability, guidance, feedback, and numerous meetings greatly aided in the construction and formalization of this master's thesis.

Furthermore, I am grateful to the employees of Space Time for their warm welcome and support throughout this work. In particular, I would like to thank Rohith Belur for his continuous assistance, supervision, and availability.

Finally, but certainly not least, I would like to extend a heartfelt hug to all the individuals who were by my side during the development of this work. They are well aware of the challenges faced throughout the months dedicated to crafting this thesis and how much I may have tested their patience. Nevertheless, their positive energy was incredibly valuable. To my mom, dad, sister, Simon, Laura, Martin, Valentin, Jérémie, Nina, the kot's neighbors, my extended family, and all my other friends, this acknowledgement is dedicated to you. I want to express my sincerest gratitude for your protective support, warmth, and caring words. My deepest thank you!

Contents

1 Introduction

Segmentation of pipelines and power cables from point clouds is a critical task in various industries, including infrastructure maintenance and planning. Accurate and efficient segmentation techniques are essential for effective decision-making and management of these vital assets. This thesis aims to address the challenges associated with pipeline and power cable segmentation by proposing a novel deep learning approach that leverages a dynamic local view mechanism.

With its capacity to learn from data and automatically extract relevant features, deep learning has proven to be a powerful approach in various computer vision domains, including object recognition, anomaly detection, and image segmentation. Its success in these areas attests to its potential in tackling the challenges of complex pattern recognition in diverse applications. The application of deep learning in complex data, such as point clouds, is of significant interest.

In collaboration with Space Time, an innovative company at the forefront of research and development in the field of point cloud processing, this research aims to contribute to the advancement of segmentation techniques. SpaceTime is dedicated to pushing the boundaries of technology and enhancing the capabilities of 4D scanning services. Their relentless focus on research and development provides valuable insights and sets the stage for exploring new approaches in pipeline and power cable segmentation.

The thesis begins with a comprehensive overview of the challenges and requirements associated with accurate segmentation from point clouds. It provides a foundation for understanding the intricacies of the problem.

The Literature Review examines existing methods for point cloud segmentation, encompassing non-learning, unsupervised learning, and supervised learning approaches. A with special attention given to PointNet, a module renowned for its effectiveness in segmenting objects from point clouds.

Moving forward, the third section, the Methodology, provides an in-depth explanation of the proposed model. The dynamic local view mechanism takes advantage of a specific 3D coordinate within the point cloud, representing the starting point of the pipeline. This approach allows the model to adjust its local view along the pipeline, facilitating precise segmentation and extraction. Furthermore, this section covers the development of a custom dataset and the incorporation of an interpolation process for reconstructing the pipeline.

Finally, the Results section evaluates the performance and quality of the developed model. Despite limitations observed in the incorporation of the PointNet module in the main tool, the overall model demonstrates promising results. A theoretical model is employed to forecast various

precision levels for point labelling, showcasing the efficacy of the proposed approach. Comparative evaluations with the RANSAC segmentation method highlight the superiority of the proposed model in terms of recall, emphasizing the importance of accurately identifying pipeline regions crucial for the subsequent interpolation process. The obtained results are discussed, highlighting key findings and insights.

This thesis contributes to the advancement of pipeline and power cable segmentation from point clouds by introducing a novel deep learning approach. By leveraging a dynamic local view mechanism, it aims to overcome the challenges associated with traditional segmentation methods and goes further by introducing an interpolation mechanism of the extracted conduits. The findings have significant implications and open the door to further development and research.

The subsequent sections of this thesis delve into the problem statement, a literature review of the existing techniques for point cloud segmentation, the methodology behind this innovative shifting local view model, and the results, providing a comprehensive exploration of the proposed approach and its takeaways to the field of pipeline and power cable segmentation from point clouds.

2 Problem statement

Our research focus is based on a segmentation problem. As defined by [1], *"3D point cloud segmentation is the process of classifying point clouds into multiple homogeneous regions, the points in the same region will have the same properties"*. It is challenging because of the rawness of point clouds: redundancy, uneven sampling density, lack of explicit structure of point cloud data. The idea can be summarised as object recognition, as it aims to differentiate the shapes within the data. The problem has interested many people over the last years, who have approached it from different angles.

Segmentation can be done in different ways: object recognition, semantic segmentation, instance segmentation are some of them. Object detection consists of defining the bounding boxes of objects in the dataset, while semantic segmentation applies the detection directly to the points themselves. The difference between semantic and instance segmentation lies in the fact that semantic groups all corresponding objects in the same categories, whereas instance segmentation will differentiate each instance into a different object.

A visual representation of the principle is shown in Figure 1. The aim of the semantic segmentation in this room is to recognize the different objects in the cloud. Any point belonging to a chair is tagged and colorized in this category color. The same goes for the tables, storage cupboard, floor, door and lights in this scene.

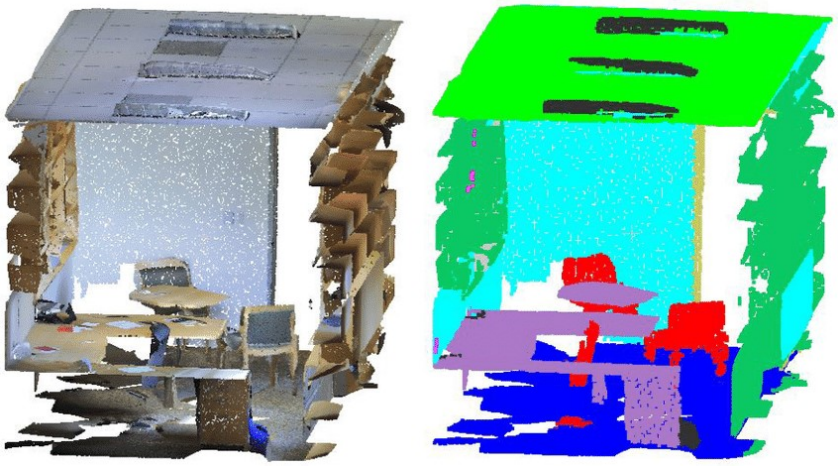


Figure 1: Example of point cloud segmentation [2]. On the left the point cloud with real colors, on the right the segmented point cloud with a color per object type: chair, desk, wall, floor, ceiling, cupboard.

The aim of this research is the development of a segmentation and extraction tool capable of accurately identifying and locating pipes and power cables within 3D point clouds of buildings.

The tool employs machine learning techniques to learn and recognize the specific features of these components within their surrounding environment. The output of the tool is a complete path of the target conduit, extracted from its environment with precision as well as an interpolation of that path.

3 Literature review

3.1 Context

This thesis is specifically concerned with construction site tracking. The construction of a building comprises several stages, including the installation of essential systems, such as electrical, ventilation, plumbing, and heating systems. These systems are, at the stage studied, the most conspicuous elements of the building site, which primarily consists of walls, floors, ceilings, and these structures. The systems can be visualized as pipes and power cables, as they facilitate the transmission of electricity, air, water, and heat throughout the building.

The complexity of construction sites, which comprise a vast number of pipes and cables that traverse intricate pathways throughout a building, raises pertinent questions. As depicted in Figure 2, locating a particular pipe or cable without becoming disoriented, misdirected, or making errors presents significant challenges. Moreover, identifying the connection points between cables and pipes and tracing their routes to determine their ultimate destination is a daunting task. Considering that these systems will be concealed behind walls and floors in subsequent construction stages, managing their organization is equally challenging.

Figure 2: Construction sites with wires installed

The stakes are numerous and significant. Visualizing intermediate steps allows for saving information that would later be inaccessible with significant precision. A 3D scanning of the environment as being performed in Figure 3, allows to output such an intermediate visual representation of the space. Point clouds also enable precise millimeter-scale measurement studies. Specifically in the field of electrical pipes and cables, they can help accurately locate pipes and cables, thereby avoiding

potentially costly location and drilling errors. They can also quickly identify problem areas, such as damaged pipes or cables or overloaded areas, allowing for quick and effective intervention. Scans can also help optimize the design of the electrical system by providing a clear view of how pipes and cables are arranged and interconnected and by providing accurate information for future updates and modifications.

Figure 3: Spot robot performing a 3D scan of the surrounding space [3].

The implementation of a tool that can determine the precise location and path of pipes and cables in construction sites would offer multiple benefits, including enhanced accuracy, efficiency, and time-saving measures for all parties involved. This is where 3D scanning and point clouds can provide valuable solutions. With an accurate scan of the construction site described earlier, and the use of a tool capable of accurately recognizing and extracting the pipes and cables to recreate their complete paths, the potential for substantial improvements in construction management is significant.

3.2 Point clouds

Point clouds are sets of points in 3D space that represent the surface of an object or environment. Each point is defined by an (x, y, z) coordinate. They are most commonly generated by scanners and LiDARs (Light Detection and Ranging), as well as by computer graphics tools. They have a wide range of applications in fields such as architecture, engineering and construction, but also in autonomous driving or robotics. Depending on the type of scanner, settings and accuracy used, they can carry a richer value with more attributes per point such as colour, light intensity and reflectivity.

3.2.1 Generation

Many different tools can generate them, as long as they can detect points in space: LiDAR, photogrammetry and depth cameras are the most commonly used.

LiDAR is a remote sensing technology that utilizes laser beams to precisely measure distances within its surrounding environment. The LiDAR system emits laser beams towards objects in space and measures the duration it takes for the beam to reflect back to the sensor. By calculating the time elapsed between transmission and reception, the system can estimate the distance to the object or surface. In order to generate a comprehensive 3D representation of the vicinity, LiDARs can emit millions of signals per second, with the angle of projection ranging from narrow to up to 360 degrees. Each reflected signal detected by the scanner is assigned a position in 3D space, thereby constructing an accurate representation of the surrounding environment [4]. The process is illustrated in Figure 4.

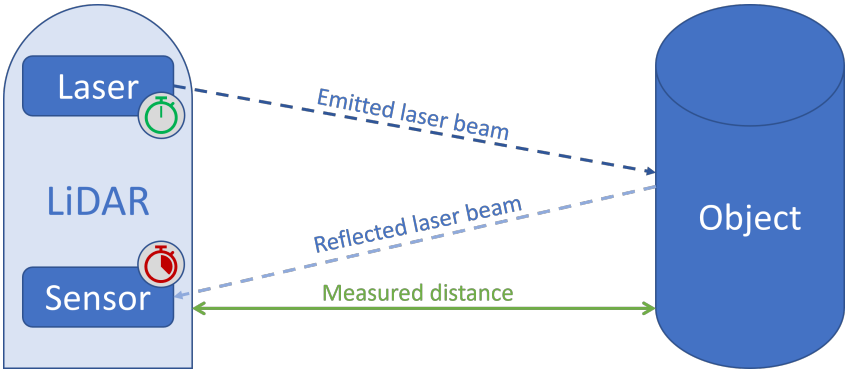


Figure 4: Schematic of LiDAR operation. Emission of a laser beam to reflect light off objects, the sensor detects the light that bounces back, measuring the elapsed time for the light to return to calculate distance and create a detailed image.

Photogrammetry is another method that can create point clouds by taking a collection of images of an object or environment and overlaying them. The 3D coordinates of the points are then calculated from the different angles of the images. The process is repeated for many points in each image, resulting in a dense cloud of points from which we can extract a detailed point cloud [5].

LiDAR technology is expensive, but produces the most detailed, precise and accurate point clouds available to represent a real space in a three-dimensional digital space.

The final result can be stored in a variety of file formats. The different formats can be written in either ASCII or binary, some allow both encodings. They are quite simple as they usually store the data as a list of each point and its associated information. A point cloud is an unordered data type, which means that the points don't have to be stored in any particular order and no assumptions can

be made about the order in which they are read.

3.2.2 Visualisation

Point cloud technology allows many problems to be approached in a new way. Although we use its written version during processing and the fact that it has a lot of applications, its first aim is to represent any 3D environment. Therefore, the most useful application is the use of visualisation.

There are several tools for visualising point clouds. Some famous Python libraries like Open3D, Trimesh, Vedo(V3do), Pyrender, PlotOptiX, Polyscope, PyVista or Simple-3dviz [6] are easy to use options. Note that they often come with some simple processing methods to paint, scale or rotate point clouds. The most commonly used during the research for this thesis were Open3D and Trimesh. They can also be visualised online through browsers using tools such as WebGL [7].

These different methods allow visualisation with easy and efficient navigation through the cloud. An example of a point cloud can be seen in Figure 5.

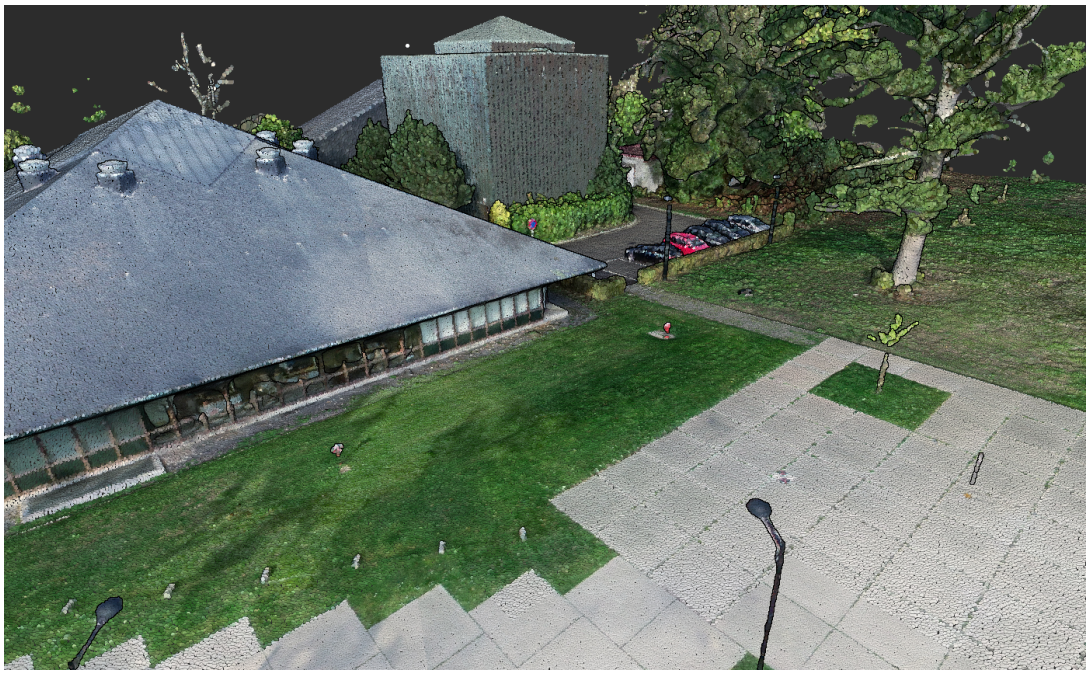


Figure 5: 3D point cloud from a DJI Phantom 4 flight followed by a Photogrammetry reconstruction process [8].

3.3 Segmentation methods

Point cloud segmentation can be performed using a variety of techniques, which can broadly be classified into three categories: non-learning, unsupervised, and supervised. Non-learning methods

are based on simple geometric or threshold-based rules and do not require any training data. Unsupervised methods use clustering algorithms to group together similar points in the data, while supervised methods rely on machine learning algorithms that are trained on labelled datasets. In the following sections, we will describe each of these methods in more detail, and discuss their strengths and weaknesses in the context of different applications with various models.

3.3.1 Non-learning based

Several methods have been presented for segmenting 3D scans. In the field of traditional algorithms without learning, Anh Nguyen and Bac Le wrote "3D Point Cloud Segmentation: A survey"[1] and divided them into five categories:

- **Edge based**: The intuition is to find edges of shapes as they can be considered as boundaries of regions. An edge is considered to be a point with a rapid change in intensity. The intensity is the ratio between the received energy of the reflected signal and the transmitted laser power from the LiDAR used to scan [9]. Although this technique provides fast results, it is highly sensitive to noise, as most decisions are based on these critical points, leading to accuracy problems.

- **Region based**: uses neighbourhood information to find similar properties of a group of nearby points and isolate them from other points. Because they are similar and close together, they can be considered to represent the same object. It also uses the dissimilarity between different regions to separate them. The accuracy is better than edge methods, but there are problems with finding accurate sharp boundaries between regions.

- **Attribute based**: uses characteristics or features of points to group them into different clusters. It is divided into two steps, which distinguishes it from the region-based methods. The first aims to calculate the different attributes, while the second builds clusters based on these attributes. The parameter that attracts attention is the quality of the derived attributes. As they are the basis for the classification, the accuracy of the calculation will determine the degree of correctness of the separation between the different classes.

- **Graph based**: considers the point cloud as a graph, i.e. each point is a vertex that we can connect with edges. From this point of view, the problems can be treated in the same way as many other graph problems. For example, Kruskal's algorithm and K-Nearest Neighbors (KNN) are widely used. The techniques usually don't run in real time and require some preprocessing. The segmentation is done in a way related to finding components in the graph. Calculating the connections in the graph is the part that requires preprocessing. It most often builds edges between points (vertices in the lexical language of the graph) based on distance,

intensity or texture. They perform better on data with noise or uneven density.

- **Model based**: groups the points using primitive geometric shapes such as sphere, cone, plane or cylinder. All points on the surface or inside the 3D shape are considered to belong to the same representation of a region. The most important contribution to these model-based methods is the well-known Random Sample Consensus (RANSAC) algorithm [10] introduced by Fischler. It can be used in the point cloud domain to retrieve 3D shapes [11]. The first step is to define the 3D shape to be fitted into the point cloud. Then comes the iterative process, which includes two loops: the external and the internal. The internal one aims to fit a given number of random instances of the given shape, the Figure 6 shows visually how many planes can be fitted in a scene.

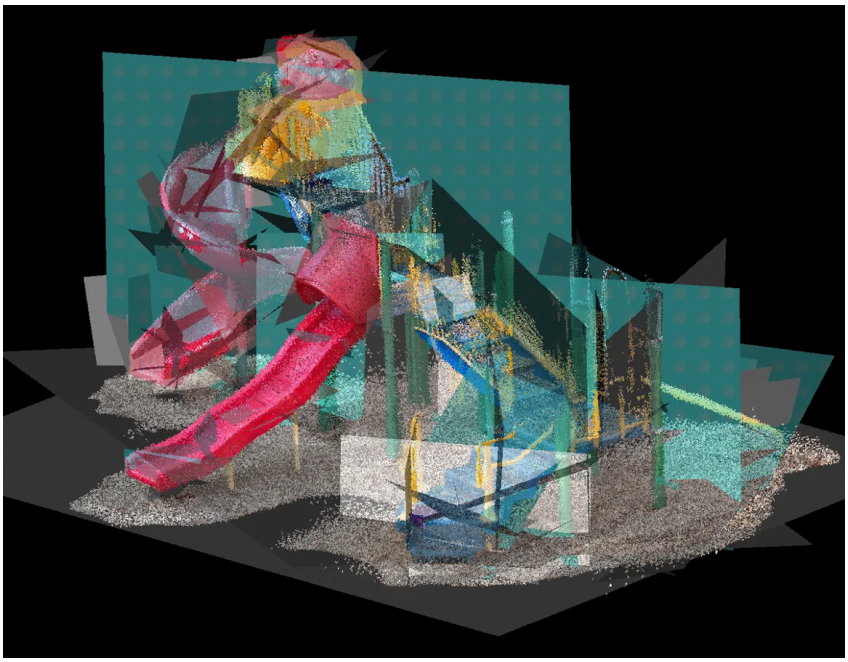


Figure 6: Representation of several planes selected by the execution of the RANSAC algorithm to find the plane including the largest number of points. Each of the planes is defined by 3 random points picked in the cloud.

In the case of a plane, it takes a triplet of coordinates. At each iteration, the parameters that result in the inclusion of the greatest number of points inside the shape are retained as the best parameters met. The result of this internal loop is returned to the external loop. The output applied to the 3D scan of a desk can be visualised in the Figure 7.

The external loop then takes the points belonging to the pattern of these most influential parameters into a subset and excludes them from the next internal iterations. The external loop continues until all points have been consumed.

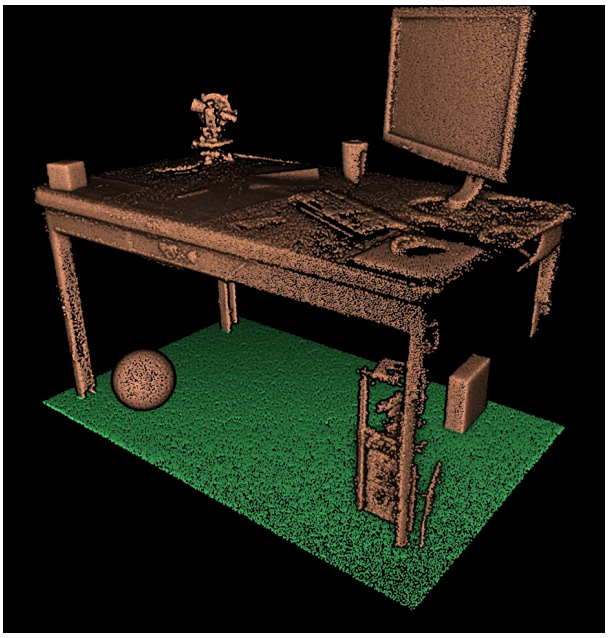


Figure 7: Result of the first iteration of the main loop of the RANSAC algorithm. The points included by this plane are represented in green. The plane is covering the ground of the point cloud as it is the plane defined by 3 points that includes the largest number of points in the point cloud.

This approach is very interesting in the context of this thesis. In fact, it can be used to fit planes and remove all the points that belong to walls and then find the remaining planes that represent the pipes and cables inside.

3.3.2 Unsupervised learning

Working with labelled datasets is often hampered by the need for costly human annotation. This limits the wide applicability of point clouds, especially when dealing with large datasets. To improve this at a lower cost, unsupervised learning aims to learn the feature representation of unlabelled point clouds [12].

The main idea that we can exploit to segment a point cloud without having any information about the content is to group the points with similar characteristics. This can be done using clustering techniques or deep learning algorithms that identify the regions and objects represented in the cloud through patterns, shapes and global structure. Many papers take different approaches to unsupervised deep learning for segmentation.

A survey [13] has been made providing a comprehensive review of unsupervised learning of point cloud representations using Deep Neural Networks (DNNs), covering the general pipelines, terminologies, relevant background, existing methods and future challenges, with quantitative bench-

marking on several widely used point cloud datasets.

The architectures of unsupervised deep learning models designed for segmentation are divided into five different main categories:

1. **Point-based**: consumes raw point cloud data without any preprocessing. Stacks of Multi-Layer Perceptrons (MLPs) are typically used to learn independent point features, which are later aggregated into global features using symmetric aggregation functions. Such architectures are widely used for point cloud tasks due to their ease of use and adaptability to many tasks.

2. **Graph-based**: as in the non-learning methods, points are used as vertices and point relations are represented by edges. Graph convolution is used to capture the strong relationships between adjacent points. This approach dynamically generates filter weights for each input to share the weights and capture local features that define the dependencies of nearby points.

3. **Sparse voxel-based**: voxelisation preprocessing is used to transform the point clouds into 3D grids. The volumetric representation is then processed with 3D Convolutional Neural Networks (CNNs). The main drawback is that the sparseness of the data implies a lot of redundancy in the computation or a loss of accuracy in the representation.

4. **Spatial CNN-based**: designed to improve the performance of regular-grid CNNs on irregularly spaced point clouds. They come in continuous and discrete convolutional networks. Their improvements fall into two categories, depending on the convolutional kernels. The continuous ones determine the weights of the neighbouring points by their spatial distribution around the centre point in a continuous space. The discrete ones work on regular grids and the kernel is defined in a discrete space with a fixed offset from the centre.

5. **Transformer-based**: popular in Natural Language Processing (NLP), they have also been applied to point clouds. They consist of a stack of Transformer blocks, each of which consists of a multi-head self-attention layer and a feed-forward network. With some fine tuning, better performance in object classification and semantic segmentation can be achieved. The main concept of the Transformers is that they are able to learn the main components of the point clouds and extract the information that defines them.

To give more insight of a novel framework introduced in unsupervised semantic segmentation PointDC [14] presents a novel approach. The framework consists of two steps, Cross-Modal Distillation (CMD) and Super-Voxel Clustering (SVC), which address the challenges of clustering ambiguity and irregularity caused by limited data and unbalanced class distribution, and the irregular sparsity of point clouds, respectively. CMD back-projects multi-view visual features into 3D space to distill

point representation training, while SVC aggregates point features into super-voxels and feeds them into an iterative clustering process to extract semantic classes. PointDC outperforms the previous state-of-the-art unsupervised methods on both the ScanNet-v2 and S3DIS semantic segmentation benchmarks with significant improvements.

3.3.3 Supervised learning

The review [15] aims at a large coverage of the segmentation techniques in general. It divides them into two types of techniques: the indirect and the direct segmentation. The indirect ones involve some preprocessing to transform the 3D point cloud into another representation, such as voxel grids or multiple views. These formats can then be processed by well-known image segmentation methods. Direct methods, on the other hand, use raw point cloud data to predict a label for each point in the cloud. Figure 8 provides an overview of the methods of supervised learning.

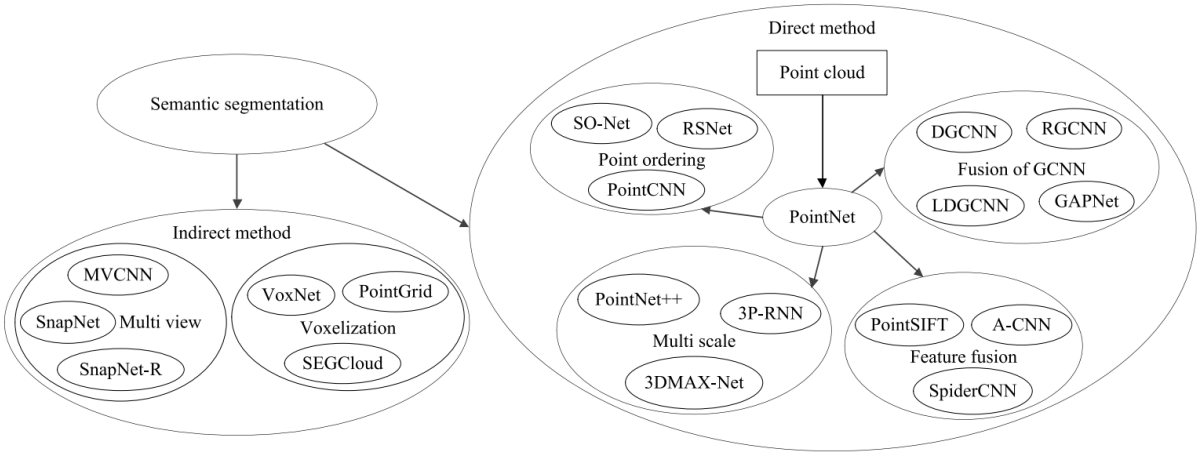


Figure 8: Visual representation of the semantic segmentation methods for point clouds discussed in [15]. It is divided in two main techniques of segmentation: indirect and direct ones. The direct methods are all based on the PointNet model fed by point clouds.

Indirect point cloud semantic segmentation methods

1. **Multi-view based**: deep learning studies on 2D data are very popular and many techniques have been developed. However, these 2D networks cannot be directly applied to point clouds due to their high irregularity. A simple way to continue using these algorithms with 3D data is to transform this data into 2D views and then apply the knowledge to them to extract the features and find the segmentation.

   Three different point cloud segmentation methods are discussed in this paper. The first method, Multi-view Convolutional Neural Networks (MVCNN) [16], uses multiple 2D images generated from different perspectives to extract features using a CNN for classification and

segmentation. The second method, SnapNet [17], addresses the problem of information loss by selecting snapshots of the point cloud to generate RGB and depth images and using fully convolutional networks to label each pair of 2D images pixel by pixel. Finally, the model projects the tagged points into 3D space to complete the task. The third method, SnapNet-R [18], improves on SnapNet by directly processing multiple views to obtain dense 3D point markers for improved segmentation results. However, all three methods are affected by the loss of geometric spatial information and projection angle, which can affect segmentation accuracy.

2. **Voxelisation**: as seen in the unsupervised methods section, voxelisation is the process of transforming unstructured point clouds into a regular three-dimensional grid, where each unit in the grid is called a 'voxel'. This transformation allows standard convolutional neural network operations to be applied to point cloud data, enabling semantic segmentation of the point cloud. It solves the problem of point cloud unstructure, but has drawbacks such as high memory consumption and loss of information. Nevertheless, it is a widely used method in point cloud analysis and has paved the way for various voxel-based point cloud processing techniques.

   Several methods have been developed to address the sparsity and resolution issues of voxel grids. VoxNet [19] uses a 3D CNN to predict class labels directly from the occupancy grid. SEGCloud [20] segregates the point cloud into voxel grids, interpolates class labels to 3D points, and uses Conditional Random Fields (CRF) for post-processing. PointGrid [21] is a mixed model that uses 3D CNN to learn grid cells with fixed points, avoiding information loss and using less memory.

However, issues such as quantitative artefacts, computational cost and loss of information remain to be addressed. Indirect methods using regular views and voxel grids have also been successful, but have their own drawbacks.

Direct methods for semantic segmentation of point clouds

In the field of point cloud processing with deep learning, PointNet [22] has brought and developed an innovative, simple and revolutionary approach. It has become the pillar of point cloud processing and has inspired many other point cloud processing architectures, especially for classification and segmentation models. A comprehensive review of the model is presented in a dedicated Section 3.3.5, as it forms the basis of the following extensions.

1. **Point Ordering**: neural network models designed to address the challenge of irregularity and disorder in point cloud data for semantic segmentation tasks. They aim to transform the input

point cloud data into a regular order to improve the utilisation of convolution kernels and to enhance the ability of convolution operations for feature extraction on unordered data. These point-ordering models can improve the efficiency of semantic segmentation of point clouds, but they may also have limitations in capturing fine-grained structures. PointCNN [23] uses the X-Conv operator, which applies convolution to X-transformed features to improve the utilisation of convolution kernels and enhance the ability of convolution operations for feature extraction on unordered data. RSNet [24] uses slice pooling, Recurrent Neural Network (RNN) layer, and slice de-pooling to project irregular point features into regular-order feature vectors. The RNN can simulate the relevance between feature vectors. SO-Net [25] constructs a self-organising mapping to fix the position of points and realise the efficient segmentation of the point cloud. It also proposes a point cloud auto-encoder as pre-training to improve the network performance in various tasks.

2. **Multi-scale**: many applications have used convolutional neural networks to extract features for object segmentation. But when using them, size is crucial. And a small receptive field captures only local features, while a large one contains invalid information. To overcome this problem, multi-scale model architectures have been developed to extract features at different scales. PointNet++ [26] is an enhanced version of PointNet for point cloud segmentation that selects local regions using Farthest Point Sampling (FPS) and recursively applies PointNet to extract local features. FPS is a technique that selects a subset of points by always picking the one that is farthest away from the ones already in the set. 3DMAX-Net [27] uses multi-scale feature learning and local-global feature aggregation to improve segmentation accuracy. 3P-RNN [28] uses a pointwise pyramid pooling module and bi-directional hierarchical RNNs to learn spatial context information.

3. **Feature Fusion**: it is a crucial technique for improving the performance of semantic segmentation of point clouds by combining global and local features. As PointNet only extracts global features and does not consider local regions, researchers have made many improvements to address this shortcoming. PointSIFT [29] is a module that encodes the information of the main directions into a unit, and stacks multiple units to obtain different features. It is embedded in the network to achieve the semantic segmentation of the point cloud, so it can introduce the knowledge of 2D image processing into 3D point cloud to obtain the local feature of the scene. A-CNN [30] applies annular convolution in a hierarchical neural network to extract the geometric features of the local neighbourhood around each point, while using feature fusion to combine the global features and local features. SpiderCNN [31] is a point cloud semantic segmentation model that uses a unit called SpiderConv to extend convolution operations to irregular sets of points, allowing effective extraction of geometric features from point clouds in the scene.

4. **GCNNs**: Graph Convolutional Neural Network (GCNN) is widely used in computer vision, which operates on the graph structure to capture dependencies by transferring information between nodes. Dynamic Graph CNN (DGCNN) [32] was the first to apply GCNN to point clouds in combination with PointNet for segmentation. The construction of the graph is dynamic and is updated after each layer. Linked Dynamic Graph CNN (LDGCNN) [33] is a modified version of DGCNN whose basic idea is to link the hierarchical features extracted from different dynamic graphs and replace the transformation network with MLP. Regularized Graph CNN (RGCNN) [34] is composed of three regular graph convolution layers, all of which consist of graph construction, graph convolution and feature filtering. It allows to capture the dynamic graph structure by updating the graph Laplacian matrix. GAPNet [35] uses a graph attention mechanism to learn local geometric representations. It introduces a GAPLayer to learn attention features for each point, a multi-head mechanism to aggregate different features, and an attention pooling layer to capture the local signature.

3.3.4 Pipe segmentation in point clouds

Taking a closer look at the task of pipe recognition in industrial zones and their 3D point cloud representation, the task has interested many researches [36]. These sites contain many primitive shapes such as cylinders, curves, planes representing the pipes, bends or walls.

The detection of cylinders in large scale point cloud interested [37]. The main steps are shown in Figure 9. The authors use structural features to develop their model and propose a method for detecting and decomposing hierarchical structures. This method simplifies the complex task of reconstructing pipeline systems in 3D space into a series of simple circle detection problems in 2D space.

Other simpler problems of cylinder detection have been studied. The paper [38] describes a fitting technique used to locate cylinders in area data. The technique involves two key steps: fitting ellipses to partial data and fitting lines to sets of three-dimensional points. The technique is designed to filter out large errors before applying a smoothing procedure to compute an accurate model. The paper provides examples to illustrate the application of this technique.

Another approach by [39] presents a novel method for extracting cylinders from a disordered set of 3D points. The method consists of two distinct steps: in the first step, a constrained plane is extracted from the Gaussian image, providing a subset of 3D points and a direction. In the second step, cylinders with a known direction are extracted from the corresponding subset of points, using a random sampling method for robustness. The paper includes experimental results demonstrating the extraction of pipes in digitised industrial environments.

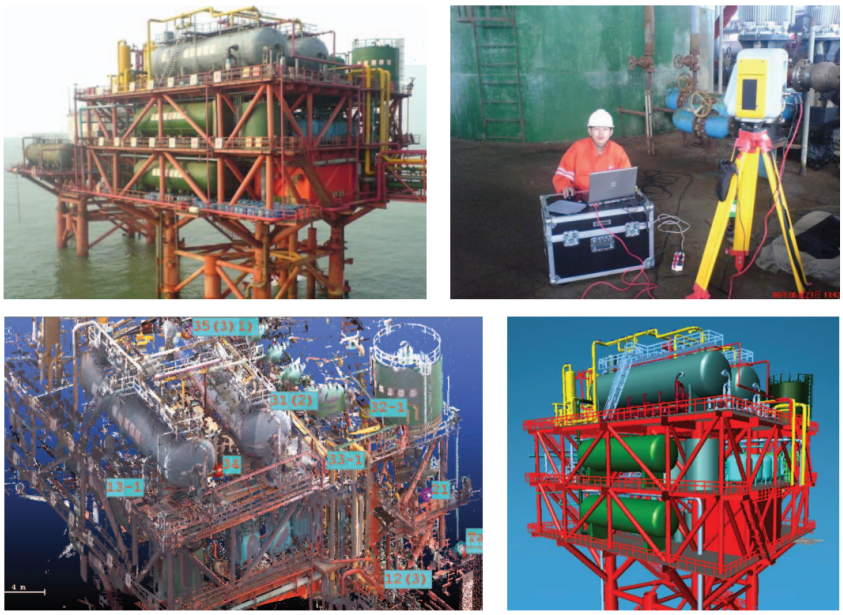


Figure 9: From left to right, top to bottom: the physical plant, the laser scanning phase, the scanned point cloud and the final reconstructed model [37].

More recently, [40] discusses the use of laser scanners to acquire point cloud data for 3D modelling of industrial plants. The paper presents two methods for registering point clouds of industrial scenes with different coordinate definitions, using corresponding object models to determine the translation and rotation parameters. The first method is a two-step approach where object fitting and registration are performed separately, while the second method simultaneously determines the shape and pose parameters of the objects as well as the registration parameters. Both methods use non-linear least squares and provide quality measures in the form of a covariance matrix of the estimated parameters, which can be used to decide whether more scans are needed and where they should be taken. Specifically for pipes, they say they are often represented as cylinders, which are difficult to model because of fuzzy start and end points. Point spacing and occlusions also complicate the task. For this reason, they use an infinite cylinder parameterisation. The paper includes results from actual industrial sites where registration was done without the use of artificial targets.

Moreover, a new technique for automatic 3D modelling of industrial plants from point cloud data with a focus on cylinder feature extraction is proposed in [41]. The traditional 5D Hough transform algorithm is computationally expensive, so a more efficient approach using a 2D and 3D Hough transform has been proposed. To further reduce the runtime and improve robustness, a coarse-to-fine approach and a clustering technique are implemented. The coarse-to-fine approach generates an initial estimate of the cylinder features and iteratively refines the estimate to reduce runtime and improve accuracy. The clustering technique groups the cells in the accumulator to

improve the robustness and accuracy of the algorithm in diverse point cloud data with multiple cylinders. The proposed methods have shown improved accuracy and reduced runtime in cylinder feature extraction, making them promising for pipe segmentation in point clouds.

To provide further explanation, the Hough transform (HT) is first used as an efficient method for line detection in binary images. Spatially extended patterns are transformed to produce compact features in a parameter space. In this way, HT transforms a global detection problem in image space into an easier local peak detection problem in parameter space [42].

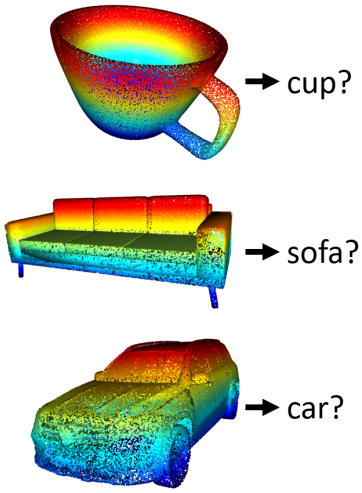
3.3.5 PointNet

As we have seen, first introduced in December 2016, the PointNet [22] is an innovative and effective method for processing point cloud data. It had a significant impact in the scientific community and has demonstrated versatility and efficacy in a multitude of applications. It has been largely reused and adapted to suit the needs of many other tasks. Its notable characteristics of adaptability and simplicity have contributed to its widespread adoption and popularity. Based on these advantages, the proposed model in this research will adopt the PointNet as its foundation. The PointNet architecture comprises several modules, and its basic design is focused on point cloud classification. In this context, classification refers to the model's capability to receive a complete set of point cloud data as input and classify them into predefined categories learned during its training phase.

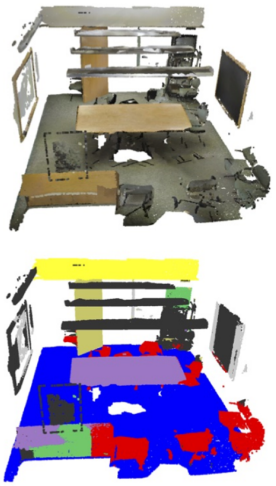
Let us note the difference between classification and segmentation of a point cloud. Both have been defined before, the main difference is the output. The goal of cloud classification is to output a single value that is the category of the input. Whereas segmentation aims to output a label per point in the cloud, where the label is a given category. This means that where classification tries to recognise the whole point cloud as a given object, segmentation finds the different objects within the data and their boundaries to accurately recognise different objects within them. An illustrative example is shown in Figure 10.

What interests us in this work is the segmentation part, but the classification module of PointNet has a big impact on the architecture. An explanation of the main challenges is given first, before more details about the architecture and the extension to allow segmentation of the points are given.

As emphasised by the authors, the architecture has to deal with properties of point clouds such as permutation invariance, transformation invariance and point interactions. This means that no assumptions can be made about the order of the input points. The model should also be robust enough to handle transformations such as rotation and translation of the content. Finally, neighbouring points often carry valuable information about their meaning. The algorithm should be able to understand local interactions between points.



(a) Example of point cloud classification. The aim is to find out what the object represented by the data is.



(b) Example of point cloud segmentation. The aim is to label each point of the point cloud with its exact object type.

Figure 10: Visual representation of the difference between point cloud classification and segmentation.

Three key modules meet these requirements: the Max-Pooling Layer, a Local and Global Information Combination Structure, and Two Joint Alignment Networks.

1. **The Symmetry Function for Unordered Input**: there are different strategies to deal with the permutation problems. PointNet uses a symmetry function. This function allows to aggregate the information from each point by treating each of them equally, without privileging any order. It takes n vectors as input and outputs a single vector invariant to the input order. The function used in the paper is max pooling, which takes the maximum value in each dimension of the feature vectors.

2. **Local and Global Information Aggregation**: this part takes as input the result of the symmetry function step. The resulting feature vectors are fed into a shared Multi-Layer Perceptron network. This MLP learns the higher level features of the point cloud. The output is a global feature vector that captures the overall shape of the input point cloud. However, the local information is missing. To incorporate it, the global feature vector is concatenated with each of the individual feature vectors of the input points. The resulting concatenated vectors are then passed through another MLP network to compute a new set of per-point features. These take into account both the local and global context. The learned per-point features can be used with a fully connected network to perform classification or segmentation tasks.

3. **Joint Alignment Network**: its purpose is to ensure that the learned representation of the point cloud is invariant to geometric transformations, such as a rigid transformation. The network achieves this by computing a transformation matrix using a mini-network called a T-net. This matrix is applied to the coordinates of the input points to align them in a canonical space. The T-net consists of basic modules of point-independent feature extraction, max-pooling and fully connected layers, similar to the large network. The same idea can be extended to feature space alignment by inserting another alignment mesh on point features and predicting a feature transformation matrix. However, since the feature space transformation matrix has a higher dimension than the spatial transformation matrix, a regularisation term is added to the softmax training loss. This is to force the feature transformation matrix to be close to an orthogonal matrix.

All this together gives the architecture in Figure 11. The input transformation is first applied to obtain robustness against transformation. Note that the shared MLP is used periodically to change the dimensions. These MLPs are a single MLP shared by all n points. To obtain the global feature, a maximum pooling is applied on a higher dimensional embedding space. In order to focus more precisely on the segmentation version that is of interest in this work, the model should use both the local and global features. Therefore, they are concatenated to obtain a per-point vector. The same shared MLPs are used to change the dimensions [43].

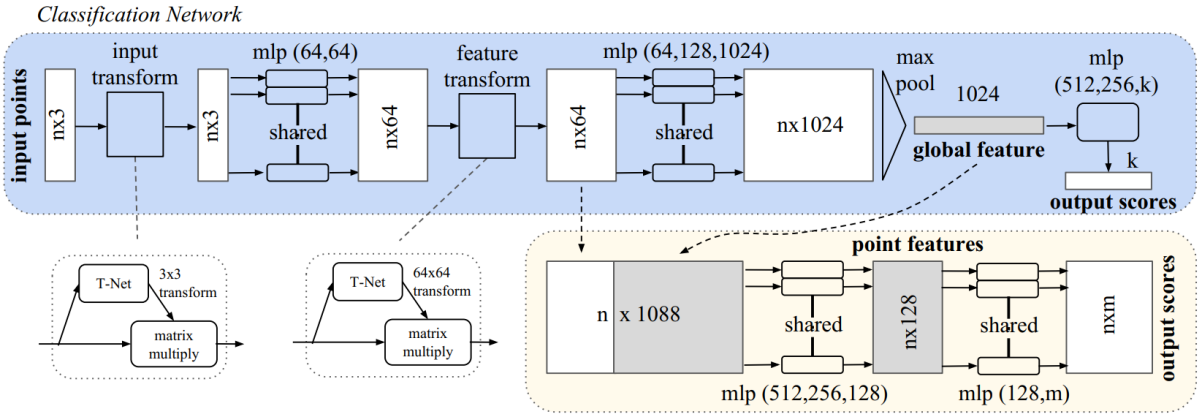


Figure 11: PointNet architecture [22].

More tuning is actually applied to the final segmentation model as shown in Figure 12. In particular, we can see that the sizes of the MLP are adjusted. Another major difference is the concatenation of the layers. Many local feature vectors are added to the global feature during concatenation to get the full model. The model also uses a combination of T, FC and MLP. The T are alignment or transformation networks. The FC are fully connected MLP layers. The MLP is a MLP applied to each point.

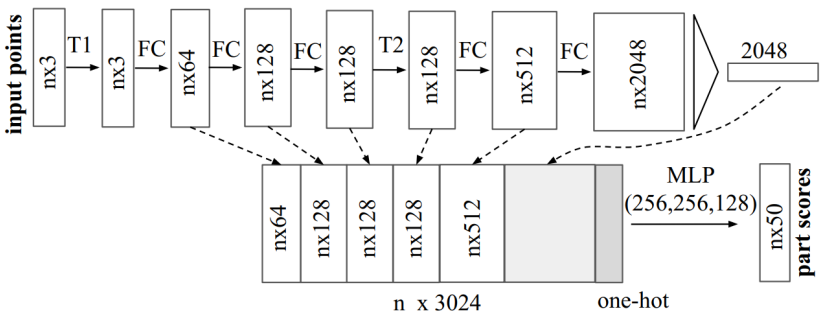


Figure 12: PointNet architecture [22] for segmentation.

On the innovative side, let us note the input format. Most of the researchers were preprocessing and postprocessing the point clouds with voxelization, block partitioning, image flattening or graphs construction. But all of these steps are actually very demanding on both temporal and spatial resources. PointNet takes the raw point cloud as input, it means list of point coordinates (x, y, z).

On a more practical note, the original paper used the ModelNet dataset for classification, it learned to classify into 40 categories of man-made objects, divided into 9843 for training and 2468 for testing. Examples of categories are: bag, cap, car, guitar, engine or laptop. For the segmentation training, the authors used the Stanford 3D Semantic Parsing dataset. It contains 3D scans taken with Matterport scanners for a total of 271 rooms. Each point in the scan is annotated with one of the semantic labels from the 13 categories found in the scenes, i.e. chair, table, floor, wall, etc.

4 Methodology

4.1 Model overview

The purpose of this master thesis is to develop a tool capable of recognizing various types of conduits, including pipes and power cables, in a construction site point cloud that contains a large number of such materials with different shapes, curves, and installations. To achieve this, a dataset of 8 rooms with various architectures, floors, walls, and pipes has been created, which serves as input for a deep learning model that focuses on segmentation and extraction of these materials. The model is based on the idea that the full path of a cable can be retrieved by finding its starting point and following it until the other end is reached. Using this method allows for a local view of the point cloud at all times, meaning that we don't have to look at the whole scene, but that we can focus on a subset of it where we are currently following the conduit. Point cloud data are very large and this methodology allows to focus on the interesting parts to better localize the pipe and reducing the complexity of the segmentation. This local view moves along the focused pipe and follows its direction, which has been translated into a concept called as *box of search*. A box of search is a subset of the general point cloud that consists of a cube with a given dimension centered on a specific point.

More precisely, we develop a deep learning model that takes as input unordered point clouds and the coordinates of the beginning of a pipe. Each of the point clouds is represented by a set of points $\{P_i | i = 1, ..., n\}$ with a number n of points per cloud. A point P_i is defined by its coordinates in the 3 dimensions (x, y, z). The starting coordinate is defined the same way as a point, i.e. as a 3D coordinate (x, y, z). The model aims to retrieve the full path of the target pipe, modelled as a sequence of ordered points $(O_0, O_1, ..., O_n)$ representing the path. They are centered in the conduit and interpolate its entire route through the input point cloud.

4.2 Dataset

The segmentation of labelled point cloud datasets remains an underdeveloped field, resulting in a scarcity of datasets, particularly within the realm of construction site analysis. Therefore, the primary objective of this research was to create a comprehensive and meticulously labelled dataset specifically focusing on spaces containing pipes. To achieve this, three main processes were employed: model building, box cutting, and data augmentation. A dataset of substantial richness was required to effectively train the model. The constructed rooms exhibited diverse characteristics, including varying shapes, multiple walls, floors, and staircases, as well as diverse pipe paths.

Subsequently, a large collection of search boxes was extracted from the created rooms. These search boxes were centered on individual pipe points and randomly selected within the rooms.

Lastly, data augmentation techniques were applied to expand the dataset's size and diversity.

The dataset employed in this study consists of a carefully chosen selection of 898 boxes distributed across eight distinct rooms.

4.2.1 Mesh creation

The meshes utilized in this research were generated using the software Blender [44]. These meshes represent the rooms and are comprised of planes that depict the walls, floor, and ceiling. Each room possesses distinct sizes and shapes, as illustrated in Figure 13.

The pipes within the dataset are represented by cylinders of fixed dimensions, as they adhere to standard measurements in the realm of building materials. The pipes exhibit various patterns, including straight paths, frequent changes in direction, and instances where multiple pipes converge before diverging. Furthermore, the pipes traverse along walls and ceilings, adapting to changes in elevation by following slopes.

The generated meshes are subsequently transformed into point clouds using the Open3D library [45]. To ensure a detailed and realistic representation of the space, the point allocation density remains consistent throughout the entire mesh. This approach enables the creation of a point cloud that accurately reflects the object's surface, mirroring the behavior of a realistic 3D scan.

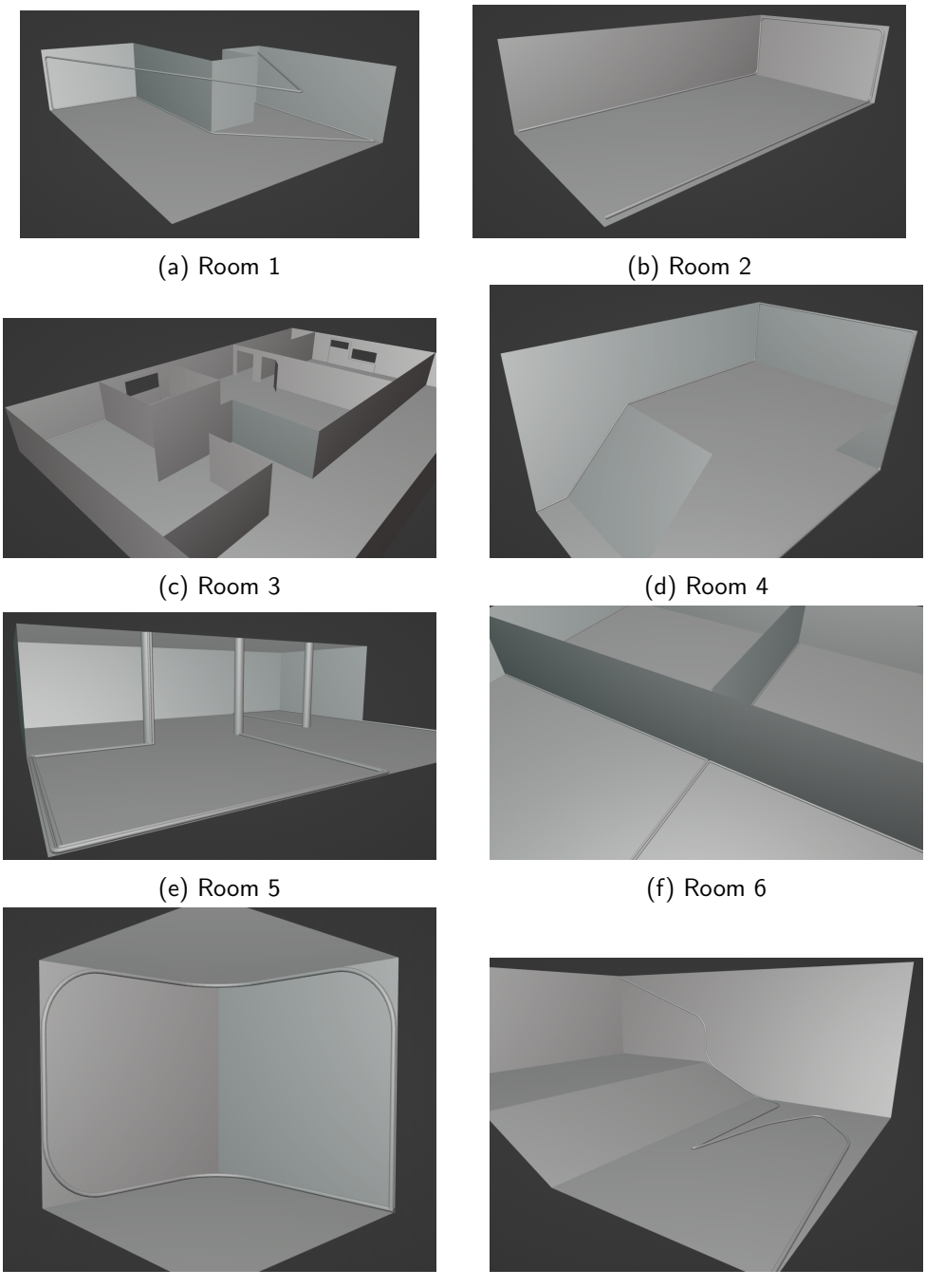
The points within the point cloud are distributed randomly across the surface, rather than adopting a uniform distribution. This random distribution aligns with the characteristics of a 3D scan obtained through LiDAR technology. LiDAR scans involve the emission of signals throughout the space, resulting in a non-uniform distribution of sampled points.

Figure 14 displays some of the resulting point clouds within the dataset, showcasing the overall structure of the spaces. For a more detailed examination of the pipes and their specific characteristics, a closer view is presented in Figure 15.

4.2.2 Boxes creation

Since the modelling process works as a sequence of local views, the dataset is generated from the handmade spaces by modelling random search boxes within them. The boxes are selected with some constraint being:

- The boxes are centered on a point belonging to a pipe. This helps to fill the dataset with realistic examples. In fact, since it is trying to follow a pipe, there should always be pipe points inside the search box.



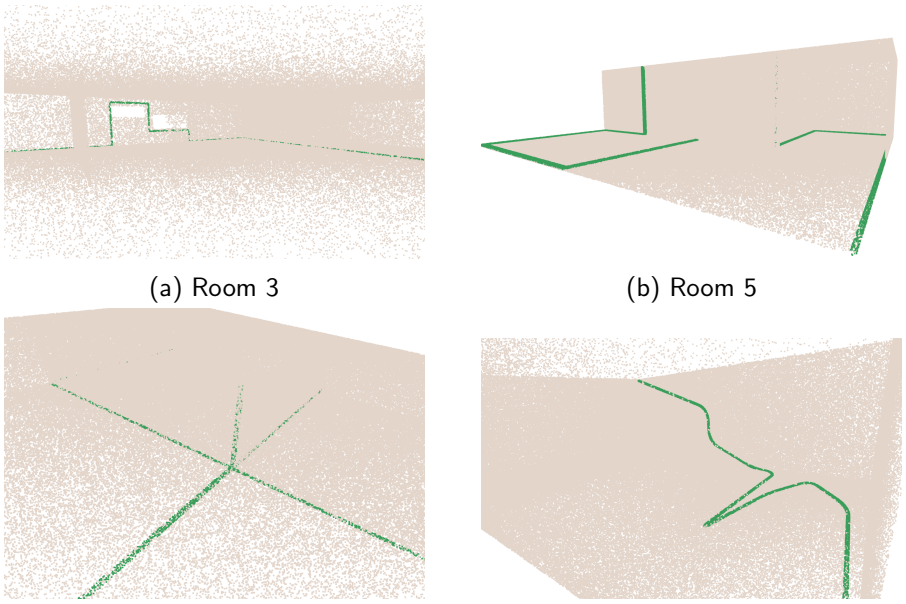
(a) Room 1 (b) Room 2

(c) Room 3 (d) Room 4

(e) Room 5 (f) Room 6

(g) Room 7 (h) Room 8

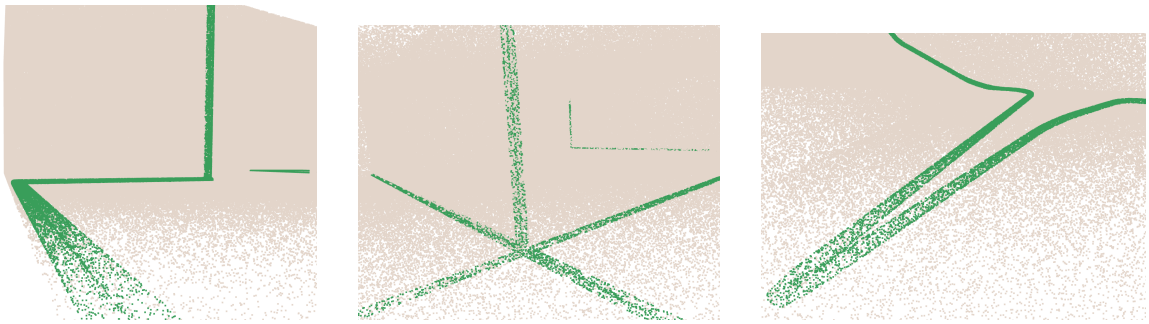
Figure 13: Rooms meshes generated using Blender for dataset construction in the context of this research. The rooms contain diverse representations of pipes, encompassing different shapes and paths typically found on construction sites. It is important to note that these images depict an internal view of the rooms, with certain walls or ceilings hidden for enhanced visibility.



(a) Room 3 (b) Room 5

(c) Room 6 (d) Room 8

Figure 14: Point clouds generated using the Open3D library from the meshes depicted in Figure 13. The point clouds represent several rooms, with points corresponding to pipes colored in green, while other components are represented in light brown.



(a) Room 5 (b) Room 6 (c) Room 8

Figure 15: Detailed view of pipes in select rooms, showcasing their shape, connections, splits, and curves. This close-up perspective provides a comprehensive understanding of the intricate characteristics and configurations exhibited by the pipes within the dataset.

- The boxes are filtered by a ratio of pipe points to total points. If there are less than 10% of pipe points in the scene, the box is discarded from the dataset. This again allows the model to be trained with an accurate dataset that most closely matches the reality of the input that will be provided when it is used. As the process follows the pipe, a considerable portion of the points in the boxes of search will be pipe ones.

An illustration of the box is shown in Figure 16.

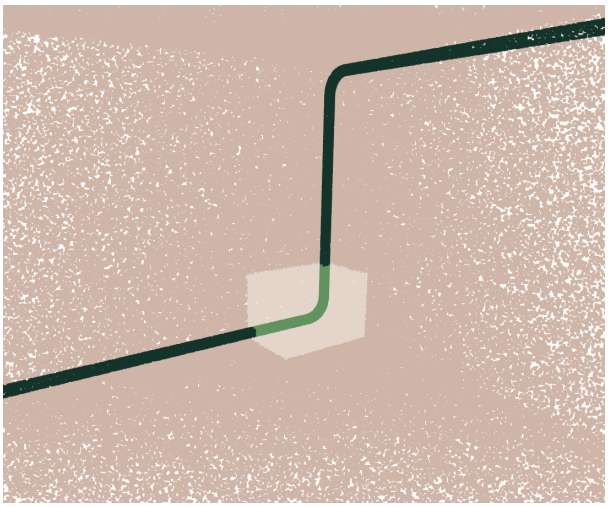


Figure 16: Illustration of a search box within a full point cloud. The points inside the box are coloured a lighter brown, with pipes shown in green and all other components shown in brown.

4.2.3 Data augmentation

The process of creating realistic boxes is often limited and time-consuming. To effectively expand the dataset, data augmentation techniques were employed. Various augmentations were applied to the boxes, including:

- **Point drop-out**: the generated rooms possessed an exceptionally high density of points, surpassing even the richest point clouds. This facilitated the multiplication of boxes by randomly selecting points to represent spaces in diverse ways. As a result, the same room could be represented by multiple distinct point clouds.

- **Noise**: the current point clouds exhibited sharpness, with their surfaces being entirely flat and uniform. However, such digitization processes do introduce noise, even when conducted with high-precision materials and LiDAR technology. The addition of noise enabled the generation of different representations of the data from a common base, enhancing the dataset's realism and aligning it closer to real-world environments.

- **Rotation**: the boxes underwent random rotations in all three directions. These rotations enabled the model to capture the fundamental characteristics of a pipe, relying less on its specific orientation and emphasizing its overall shape. Given the varied and undefined directions and curves of the pipes, this augmentation facilitated the development of a more robust model.

4.3 Implementation

4.3.1 Global behavior

The model initiates the pipeline by utilizing the input coordinate and point cloud to establish the starting point for the target pipe. The surrounding local area is then extracted from the overall point cloud to construct the initial search box. This box is subsequently provided to the model, which generates predictions regarding the likelihood of each point within the box being part of the pipe. Following the extraction of the pipe shape from the box, an analysis is performed to determine the next point of interest along the pipe. This new point serves as the updated current position within the model, initiating a loop in the process. This iterative approach enables the model to progressively shift its focus to different points along the pipe, facilitating a thorough examination of its entire length.

4.3.2 Preprocessing

During the model training phase, no preprocessing techniques are applied to the data. The dataset has been meticulously tuned to align with real-world scenarios, and no additional methods or transformations are employed on the boxes before feeding the data into the model. As a result, when utilizing the trained model, there is no need for preprocessing the input data. The raw point cloud is directly consumed as the input, without any preprocessing steps.

4.3.3 Architecture

The global-view architecture of the model is depicted in Figure 17.

The model comprises five key steps, namely: The Box of Search Transform, The Label Prediction with PointNet, The Pipe Extraction Based on the Prediction, and The Computation Module.

The Box of Search Transform. The Box of Search Transform is a crucial step in the model's pipeline. It involves filtering the points of the point cloud by centering the filtering process on a given coordinate. During the first iteration, the starting coordinate of the pipe is utilized to extract a subset of the point cloud. This subset is considered to be the local view from which the pipe extraction process commences. The filtering operation is implemented using a bounding box provided by the Open3D module. The bounding box is defined by its eight vertices, which are point cloud coordinates. These vertices are computed based on the center coordinates (x, y, z) and the box size. By default, the box is set to a cube with an edge size of 1 meter.

The Label Prediction with PointNet. Before utilizing the PointNet model, the box of search undergoes preprocessing steps. The PointNet model requires input sets of points with a fixed size of 4096. To conform to this input size, a random selection process is employed to reduce the number

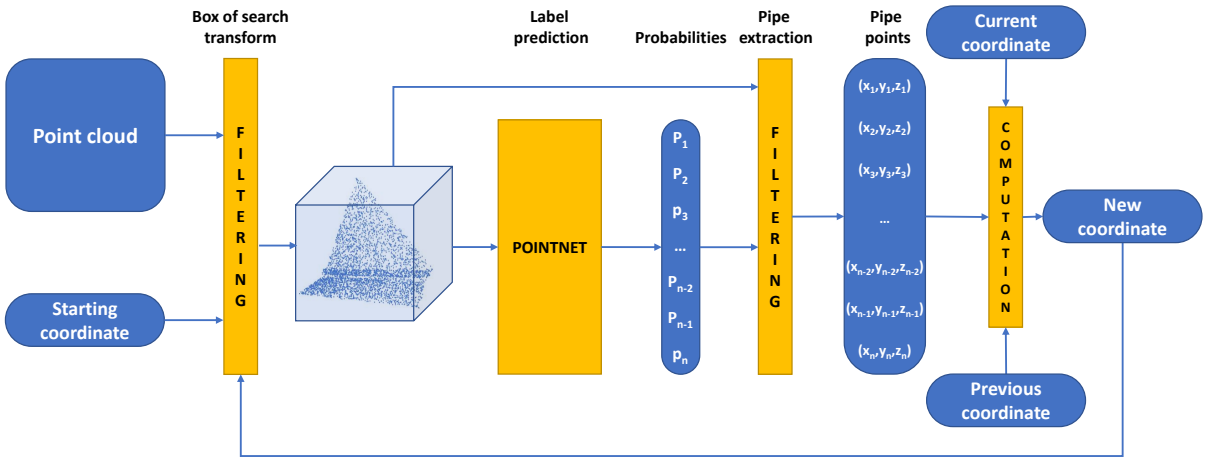


Figure 17: **Global Architecture of the Model.** The model takes a point cloud and the starting coordinate of a targeted pipe as inputs. A filtering process is applied to generate a box of search centered around the input coordinate. Subsequently, a PointNet model is utilized to predict the per-point probabilities of being classified as a pipe. Based on these predictions, the points within the box are filtered, resulting in a set of points that represent the pipe within the box. From this set, the next point of interest is identified based on the recent ones. The sequence of coordinates, along with all the recognized pipe points, constitute the final output of the model, effectively segmenting and extracting the pipe from the input point cloud.

of points in the box. The random selection ensures that the density of the data is uniformly reduced without compromising the information it encapsulates.

The resulting list of points, denoted as P, is then fed into the PointNet model. The model utilizes this set of points to estimate the probability of each point belonging to the pipe class. By leveraging PointNet's capabilities, the model can effectively analyze and classify the points within the box, providing probabilities for pipe point membership.

The Pipe Extraction Based on the Prediction. After obtaining the per-point probability array p, generated by the model, where the order corresponds to the points in P, the next step is to segment the box of search into two classes: pipe points and non-pipe points. This segmentation is achieved by applying a threshold to the probability values. By default, a threshold of 0.5 is used. Points with probabilities above this threshold are labelled as belonging to the pipe class, while points below the threshold are labelled as non-pipe points. As a result, the pipe extraction process yields a new list of points, denoted as X. This list contains all the coordinates from the local view of the box of search that have probabilities above the threshold, representing the extracted pipe points.

The Computation Module. This module is made of two main computations: the pipe inter-

polation and the next coordinate computation. Only the latter is explained in this paragraph, see the dedicated Section 4.3.4 for detailed explanation of the pipe interpolation.

In order to continue the process, the last step involves calculating the new coordinate for the center of the subsequent box. This calculation takes into account a list (L) of meaningful coordinates made of the last 3 centers and the current center to determine the direction followed by the model. The new coordinate is based on the distance between these four points and the pipe points, which allows to keep on translating the search in the forward direction.

To calculate the distance, we use the Euclidean distance metric, which measures the linear distance between two points in three-dimensional space. In this computation, we maintain a global distance array d_L of size equal to the number of points labelled as pipe points, and initialised at 0 for each index. We iterate over the four coordinates in L while calculating the distance from the coordinate to all other points labelled as pipe points. The resulting array contains the sum of the distances from the last coordinates to each pipe point. That is, the value at index i in d_L corresponds to the sum of the Euclidean distance from each coordinate in L to the point at index i in the list of pipe points X. The code is found in the Algorithm 1.

```
import numpy as np

L = box_centers[-3:] + [current_center]
d_L = np.zeros((len(pipe_points)))
for l in L:
    d = np.linalg.norm(pipe_points - l, axis=1)
    d_L = d_L + d
max_distance_index = np.argmax(d_L)
new_center = pipe_points[max_distance_index]
```

Listing 1: Computation of the next coodinate to shift the box focus on. The algorithm uses the last coordinates and the current one to understand the direction the model is moving to. It enables the election of the furthest pipe-labelled coordinate as next box center.

Finally, we identify the index (j) in the combined distance array that corresponds to the maximum value. This index represents the pipe point in the search box that is farthest from both the previous and current centers. We select this point as the new current center for the subsequent iteration, effectively guiding the model along the path of the pipe.

Finally, note that the coordinates are being updated. When the iteration is finished, the current coordinate is retained as in the last coordinates, while the status of the current coordinate is assigned to the newly computed center.

Result Construction. The result of the model consists of three main components: the pipe extraction list (Y) and the sequence of centers (s).

The pipe extraction list, denoted as Y, contains all the points that have been labelled as pipe points throughout the iteration process. These points are aggregated to form the final segmented and extracted pipe points from the input point cloud.

The sequence of centers, represented as s, records the coordinates of the centers used in each iteration. This sequence provides information about the path followed by the model during the extraction process.

Together, the pipe extraction list Y and the sequence of centers s constitute the output of the model, representing the segmented and extracted pipe points from the input point cloud.

4.3.4 Pipe interpolation

From the extracted pipe points, we perform an interpolation on the currently analyzed section of the conduit. The interpolated path is represented by a sequence of points that depict the conduit. This interpolation process is illustrated in Figure 18.

The modeling of the pipe occurs within the primary iterative process of the overall model. It takes place within the computation module depicted in Figure 17. The point cloud resulting from the extracted pipe points is initially voxelized into multiple blocks. This voxelization process is applied along the local path of the pipe to accommodate curves and irregularities. An example of voxel grid is shown in Figure 19. Each voxel generated undergoes an individual processing step, where the center point is computed based on the points it encompasses. In essence, the voxel enables a localized filtering of the pipe points, and the center point is computed accordingly. The sequence of voxels is thereby transformed into a sequence of points centered at the midpoint of the represented pipe. This sequence of points serves as an interpolation of the extracted section of the pipe.

During each iteration of the interpolation process, the computed interpolation points are systematically appended to a list that accumulates all points calculated since the initialization of the process. Once the process reaches its termination, the final interpolation is computed.

The computation of the final interpolation revolves around determining the appropriate order for the points involved in the interpolation. This is achieved through the utilization of a simple custom path-finding, the Algorithm 2, which has been developed to discern the optimal path among the given set of points. The underlying concept behind this algorithm is that the overall path can be constructed by connecting each point to its closest neighbor.

To facilitate this path determination, the algorithm employs a distance matrix, allowing for

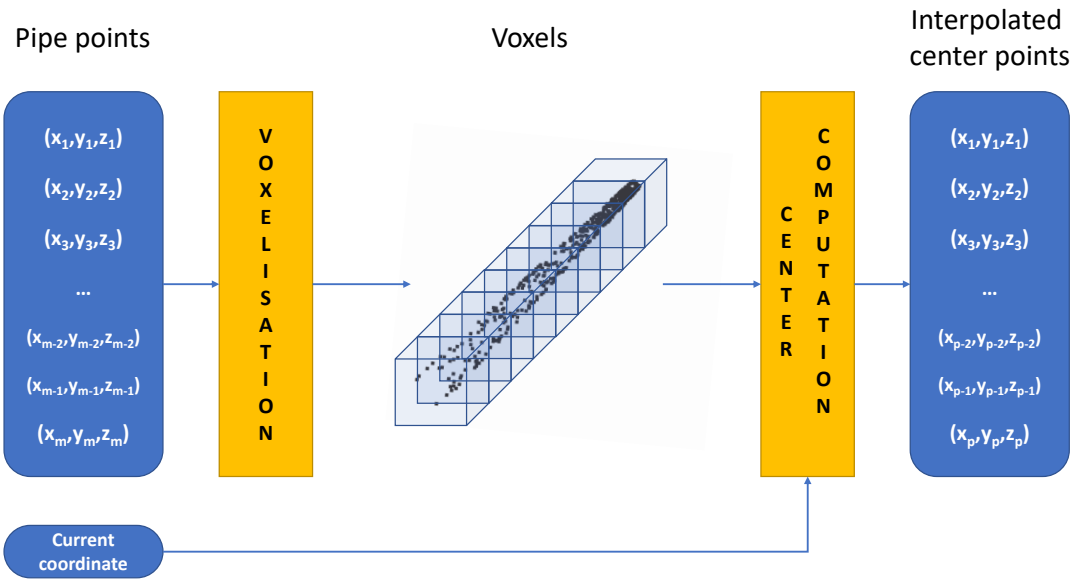


Figure 18: **Architecture of the Pipe Interpolation.** The interpolation process involves utilizing the identified pipe points and the current coordinate of the box being interpolated. The pipe points are first voxelized into several blocks. Subsequently, each voxel undergoes individual processing to determine its center point based on the enclosed points. The resulting list of interpolated center points provides an accurate representation of the extracted pipe points from the box, thereby facilitating the interpolation process.

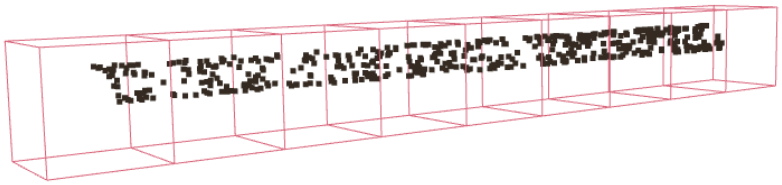


Figure 19: **Voxel grid applied to labelled pipe points.** This figure showcases the application of a voxel grid along a path of points labelled as a pipe. The voxelization process follows the point cloud, ensuring that all points are encompassed. By dividing the pipe path into small voxel units, we obtain localized views that enable the local computation of point centers. These computed centers assist in the interpolation of the pipe, enhancing our understanding of its trajectory and shape.

the calculation of pairwise distances between every pair of points in the dataset. The algorithm commences from the provided starting point, which serves as the initial input to the program. Through iterative iterations, the algorithm progressively identifies the nearest unvisited point and incorporates it into the path. This iterative process continues until all points have been visited,

guaranteeing comprehensive coverage of the interpolation.

The primary objective of the algorithm is to optimize proximity, aiming to minimize the total distance traveled along the resulting path. Its custom design stems from its specific development tailored to address the precise requirements of this interpolation task. As a result, it furnishes an efficient and effective solution for determining the optimal path among the set of collected points.

```python
import numpy as np

def get_centerline_order(centerline, current_coord):
    # Compute the distance matrix between all points
    distances = distance_matrix(centerline, centerline)

    # Find the index of the starting point
    start_index = np.argmin(np.linalg.norm(centerline - current_coord, axis=1))

    # Initialize the visited list and the path list
    visited = [start_index]
    path = [start_index]

    # Iterate until all points are visited
    while len(visited) < len(centerline):
        # Find the nearest unvisited point
        current_point = path[-1]
        nearest_point = np.argsort(distances[current_point])
        for point in nearest_point:
            if point not in visited:
                visited.append(point)
                path.append(point)
                break

    return path
```

Listing 2: **Point Ordering Algorithm for Interpolation.** This algorithm is employed to determine the order of the interpolated points. The `centerline` variable is an array containing all points of the interpolation. Starting from the `current_coord` the algorithm consistently selects the closest unvisited point. It ultimately returns the index order that minimizes the distance between point visits.

4.3.5 PointNet

In the construction of the model, the performance of the PointNet module is of crucial importance as it plays a key role in labelling the points and performing local segmentation of the point cloud.

The PointNet architecture used in this work is similar to the segmentation version described in the original paper [22]. However, some modifications have been made to adapt it to the specific task of binary classification for point segmentation.

In our task, we are interested in distinguishing between pipe and non-pipe points, which requires binary classification. Therefore, the output layer of the PointNet model has been adjusted accordingly. Instead of outputting probabilities for multiple classes, the model produces a single probability value between 0 and 1. A value closer to 1 indicates a higher likelihood of the point being classified as a pipe, while a value closer to 0 indicates a lower likelihood.

To facilitate the binary classification and optimize the model's performance, the loss function has also been adapted. The module employs a binary cross-entropy loss function, as depicted in Equation 1. The loss (L) is calculated based on the true label (y) and the predicted probability ($\hat{y}$) assigned to the positive class (i.e., the pipe class).

The choice of binary cross-entropy over categorical cross-entropy is motivated by several reasons. It simplifies the model, making it more efficient and requiring fewer computational resources. Additionally, the binary representation of the classification result enhances the interpretability of the model's output, allowing for clear and understandable results.

$$L(y, \hat{y}) = -\left(y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})\right) \tag{1}$$

5 Results

In this section, the objective is to formally evaluate the performance of a novel approach for pipeline and power cable segmentation and interpolation from point cloud data.

Firstly, an explanation of the metrics employed to evaluate the results is provided. The results are then presented in three distinct parts: (1) a comparison of the results obtained with other existing and theoretical tools, (2) an analysis of the performance of the PointNet module, and (3) a detailed qualitative examination of the overall model. Furthermore, a discussion regarding the results is conducted, along with an addressing of the limitations of the models used in this study. Finally, the key takeaways from this work are described.

5.1 Evaluation metrics

The metrics used to assess the performance of our segmentation tool are accuracy, precision, recall and F1 score.

- **Accuracy** measures the overall correctness of the binary point cloud segmentation model by calculating the ratio of correctly classified points to the total number of points. It provides a general indication of the model's performance and is useful for assessing its overall effectiveness. However, accuracy can be misleading if the dataset is imbalanced, meaning one class dominates the other in terms of the number of points. In our search boxes, it is observed that the majority of them only contains between 12% and 20% of points that belong to the pipe class. In such cases, accuracy alone may not provide a complete picture of the model's performance.

- **Precision** is a metric that assesses the model's ability to correctly classify positive points (points belonging to the pipe class) out of all points identified as positive. In the context of binary point cloud segmentation, precision measures the accuracy of identifying points belonging to the target class, which is the pipe class in our case. For our scenario, a low rate of false positives indicates that when the model predicts a point as belonging to the pipe class, it is highly likely to be correct. This is particularly important in applications where misclassifications can have significant consequences. By achieving a high precision score, the model demonstrates its capability to accurately identify and differentiate the points belonging to the pipe class while minimizing the occurrence of false positives.

- **Recall**, also known as sensitivity or true positive rate, is a metric that quantifies the proportion of correctly classified positive points (points belonging to the pipe class) out of all points that truly belong to the positive class. In the context of binary point cloud segmentation, recall

evaluates the model's effectiveness in accurately detecting and capturing all points belonging to the target class (i.e., the pipe class). In our specific case, a high recall score indicates that the model has a low rate of missing positive points. This means that the model can effectively identify and include the majority of points belonging to the pipe class in its predictions, minimizing the occurrence of false negatives. Minimizing false negatives is crucial when it is important to capture all positive instances accurately. Therefore, achieving a high recall score signifies that the model demonstrates a strong ability to correctly detect and include points belonging to the pipe class, reducing the likelihood of missing any relevant positive points.

- **F1-score** is a metric that offers a balanced evaluation of the model's performance by taking into account both precision and recall. It is calculated as the harmonic mean of precision and recall, providing a single score that reflects the model's ability to balance between minimizing false positives and false negatives. In binary point cloud segmentation, the F1-score is particularly useful when dealing with imbalanced datasets, where one class (e.g., the not pipe class) may dominate in terms of the number of points. It ensures that both types of errors, false positives and false negatives, are considered equally important in the evaluation process. By optimizing the F1-score, a model aims to strike a balance between precision and recall. This means finding the best trade-off between accurately identifying positive points (minimizing false positives) and capturing as many positive points as possible (minimizing false negatives). The F1-score provides a comprehensive metric that helps assess the overall performance of the model, taking into account both precision and recall simultaneously.

Binary point cloud segmentation tasks are often focused on detecting specific objects or regions of interest, where both false positives and false negatives can have significant implications. Evaluating the performance of such models based solely on accuracy may not provide a comprehensive assessment. To address this, additional metrics such as precision, recall, and F1-score are employed, as they offer a more detailed analysis of the model's ability to correctly classify positive points (e.g., pipeline points) while minimizing false positives and false negatives. These metrics are particularly suitable for the evaluation of binary point cloud segmentation tasks like the one at hand.

In addition to these essential quantitative metrics, point cloud segmentation presents unique challenges due to the complex boundaries between objects and shapes. The inherent complexity of point cloud data, characterized by the absence of explicit boundary representations and potential noise or sparsity, makes accurate segmentation a demanding task. To gain deeper insights and obtain a comprehensive evaluation of the model's performance, visualization plays a crucial role.

Visualization of example point clouds allows for a subjective evaluation that complements the quantitative metrics. It provides a human-readable perspective, enabling observers to visually inspect the quality of the segmentation output. Through visualization, one can assess how well the algorithm

captures intricate details, accurately identifies object boundaries, and correctly classifies points within the desired objects.

By examining visual examples, it becomes possible to intuitively understand the segmentation performance and evaluate the perceptual quality of the results. Visualization serves as a valuable tool for identifying specific challenges or limitations of the segmentation algorithm. It can reveal cases where the algorithm struggles to distinguish between object classes, handle occlusions, or cope with complex geometries. These visual insights guide further improvements or refinements in the segmentation approach, leading to enhanced performance and addressing specific shortcomings.

Finally, to assess the pipe interpolation results, we use the RMSE (Root Mean Square Error) over distance metric.

Let P be a sequence of points, and let L be a list of points. We want to compute the RMSE of each point in P to the line formed by the two closest points from L. For each point p_i in P, we find the two closest points l_1 and l_2 in L to p_i. Then, we compute the distance between the line formed by l_1 and l_2 and the point p_i. Finally, we calculate the RMSE of all these distances to obtain the overall RMSE. The computation can be represented by the following equation:

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n} \left(\text{distance}(p_i, \text{line}(l_1, l_2))\right)^2}{n}}$$

where:

- n is the number of points in P,
- p_i is the i-th point in P,
- l_1 and l_2 are the two closest points in L to p_i,
- $\text{line}(l_1, l_2)$ represents the line formed by l_1 and l_2,
- $\text{distance}(p, \text{line})$ is the distance between a point p and a line.

The distance is calculated as the perpendicular distance from the point (x_0, y_0, z_0) to the line segment defined by (x_1, y_1, z_1) and (x_2, y_2, z_2) in 3D space in the following formula:

$$\text{distance}\left((x_0, y_0, z_0), P_1, P_2\right) =$$
$$\frac{|(x_2 - x_1)(y_1 - y_0) - (x_1 - x_0)(y_2 - y_1) + (x_2 - x_1)(z_1 - z_0) - (x_1 - x_0)(z_2 - z_1)|}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}}$$

where:

- (x_0, y_0, z_0) is the 3D point for which we want to compute the distance,
- P_1 and P_2 are the two 3D points defining the line segment,
- (x_1, y_1, z_1) and (x_2, y_2, z_2) are the coordinates of the two points defining the line segment.

Given two sequences of points, this distance metric is employed to quantitatively evaluate the dissimilarity between the two sequences. In our specific scenario, we have a target sequence of points representing a reference or desired outcome, and an evaluated sequence of points representing the actual or observed outcome. The metric is computed by averaging the root mean square error for each evaluated point and the corresponding point in the target sequence. To achieve this, for each evaluated point, we identify the two closest points from the target sequence. By considering the line defined by these two closest points, we determine the distance between the evaluated point and this line. This process allows us to project the evaluated point onto the goal line and evaluate the discrepancy or error. The resulting mean distance provides a quantitative measure indicating the overall similarity or dissimilarity between the two sequences.

5.2 Comparative analysis

In this study, a comparison is conducted between the model developed in this work and other approaches:

1. **A theoretical model of given accuracy**: The comparative analysis involves a model based on the same architecture as our model, but with the PointNet model replaced by a theoretical model. This theoretical model is designed to achieve a given accuracy (a), which is provided as an input to the model. The accuracy represents the average accuracy achieved by the model in point labelling. In this comparison, we consider the theoretical model to possess a good understanding of pipe positions, whereby errors primarily arise from selecting points in close proximity to the pipe rather than random points within the bounding box. The purpose of this model is to approximate the performance level of the overall system by utilizing a module that can variably recognize pipe points. If a model with $a = 1.0$ is employed, it serves as a proof of concept for the global model, demonstrating the optimal results.
2. **A RANSAC model**: The RANSAC-based model, as discussed in Section 3.3.1, is employed to address the segmentation problem. This approach offers a different perspective by utilizing a non-learning-based technique. Unlike our model, which focuses on extracting specific pipes, the RANSAC-based model does not prioritize the precise delineation of individual pipes during segmentation. Additionally, it requires more human supervision.

In these tests, the RANSAC method utilized is a basic one, involving the fitting of planes into the point cloud. The algorithm proceeds by iteratively selecting points that lie on a plane and removing them for subsequent iterations until either all points have been consumed or the maximum number of planes is reached. To evaluate the results of this technique, we simplify the human role by employing a threshold: any plane with more than 50% of its points labelled as "pipe" assigns all the points on that plane to the "pipe" label. Conversely, any plane with less than 50% labelled as "pipe" assigns all its points to the "not pipe" label. It is important to note that a plane includes all points within a maximum distance of 2 centimeters from it.

5.2.1 Quantitative comparison to a RANSAC approach

In this section, it is important to interpret the accuracy metric with caution when comparing the RANSAC method to our model. The RANSAC method extracts all pipes simultaneously and provides an accuracy measure over the classification of all points, whereas our model focuses on targeted pipes and does not cover the entire point cloud. Therefore, the accuracy metric is not based on the same segmentation processes and does not encompass the same points for segmentation. However, precision, recall, and F1-score, which evaluate pipe classification, hold significant meaning in this context.

The results obtained for models with theoretical accuracies ranging from 70% to 100% and the RANSAC method are presented in Table 1. This table facilitates a comparative analysis of performance metrics, including accuracy, precision, recall, and F1-score, for different rooms and techniques. Each row of the table corresponds to a specific room, while each column represents a particular metric or technique.

Our model consistently outperforms the RANSAC model in terms of pipe recognition, as indicated by the higher recall metric. However, there are specific room conditions, such as Room 2, where the RANSAC method achieves a higher recall value. These rooms are characterized by pipes that are not closely situated near walls, which may not accurately represent real construction sites. Overall, our model demonstrates a balanced performance between precision and recall. While the precision value is generally lower than that of the RANSAC method, our model excels in locating pipe points accurately and uniformly, as reflected in its high recall value. In other words, although RANSAC achieves high accuracy in labelling points as pipes, it tends to miss more pipe points compared to our model, which effectively identifies pipe locations.

Furthermore, our model exhibits lower deviation and offers a more versatile solution, yielding a stable and higher average F1 score across various input data types. When comparing F1 scores, we observe that segmentation models with a module accuracy of over 85% can compete with the RANSAC method, while models with an accuracy above 90% perform even better on average. It

Room	Metric	Model						
		70%	80%	85%	90%	95%	100%	RANSAC
Room1	accuracy	0.847	0.899	0.915	0.942	0.971	1	0.993
	precision	0.593	0.679	0.718	0.780	0.886	1	0.998
	recall	0.695	0.795	0.836	0.891	0.952	1	0.782
	f1score	0.595	0.703	0.748	0.815	0.910	1	0.877
Room2	accuracy	0.741	0.815	0.864	0.901	0.957	1	0.999
	precision	0.277	0.382	0.482	0.584	0.780	1	0.999
	recall	0.704	0.811	0.841	0.903	0.948	1	1
	f1score	0.397	0.518	0.607	0.702	0.848	1	0.999
Room3	accuracy	0.864	0.894	0.909	0.938	0.956	1	0.996
	precision	0.319	0.390	0.427	0.516	0.618	1	0.999
	recall	0.702	0.801	0.853	0.893	0.942	1	0.404
	f1score	0.437	0.522	0.567	0.647	0.735	1	0.575
Room4	accuracy	0.778	0.829	0.859	0.911	0.948	1	0.987
	precision	0.278	0.362	0.436	0.571	0.727	1	0.997
	recall	0.693	0.796	0.850	0.891	0.940	1	0.313
	f1score	0.396	0.497	0.572	0.689	0.808	1	0.476
Room5	accuracy	0.749	0.821	0.851	0.895	0.940	1	0.973
	precision	0.426	0.543	0.598	0.684	0.795	1	0.977
	recall	0.709	0.804	0.847	0.889	0.947	1	0.415
	f1score	0.527	0.642	0.696	0.764	0.857	1	0.582
Room6	accuracy	-	-	-	-	-	-	0.989
	precision	-	-	-	-	-	-	0.998
	recall	-	-	-	-	-	-	0.327
	f1score	-	-	-	-	-	-	0.493
Room7	accuracy	0.774	0.824	0.852	0.899	0.958	1	0.966
	precision	0.306	0.410	0.458	0.602	0.810	1	0.964
	recall	0.700	0.799	0.845	0.882	0.946	1	0.371
	f1score	0.425	0.540	0.590	0.707	0.864	1	0.536
Room8	accuracy	0.764	0.834	0.855	0.909	0.952	1	0.993
	precision	0.347	0.476	0.516	0.655	0.801	1	0.999
	recall	0.697	0.799	0.846	0.891	0.949	1	0.680
	f1score	0.462	0.594	0.637	0.745	0.859	1	0.809

Table 1: Comparative analysis of accuracy, precision, recall, and F1 score for different rooms using various models.

is important to note that the average balance of labels in the search dataset is around 85%, and these results are expected to be related to that value. This suggests that data with a more balanced distribution may allow the segmentation module to perform well with a slightly lower accuracy than 85%.

The missing values in Room 6 are attributed to the model's inability to accurately track the targeted pipe when it is surrounded by multiple pipes. This room consists of four pipes closely meeting at a central point, causing the search box to occasionally switch the target to the wrong pipe during critical moments. A more detailed and visual explanation is provided in Section 5.4 on qualitative analysis.

It is worth noting that some theoretical models achieve higher accuracies than their input accuracy values. This discrepancy arises from point clouds with lower density, which hinders the model from performing at the expected accuracy. In such cases, the selected error points are often located in the areas surrounding the pipe. Additionally, certain parts of the point cloud may contain isolated pipes that are not closely surrounded by many points, especially if the pipe is not adjacent to a wall or situated at the edge of a surface. These cases are identifiable as the recall value, representing the percentage of recognized pipe points, remains close to the theoretical accuracy, but the counterbalance of non-pipe selection cannot lower its level to the desired value.

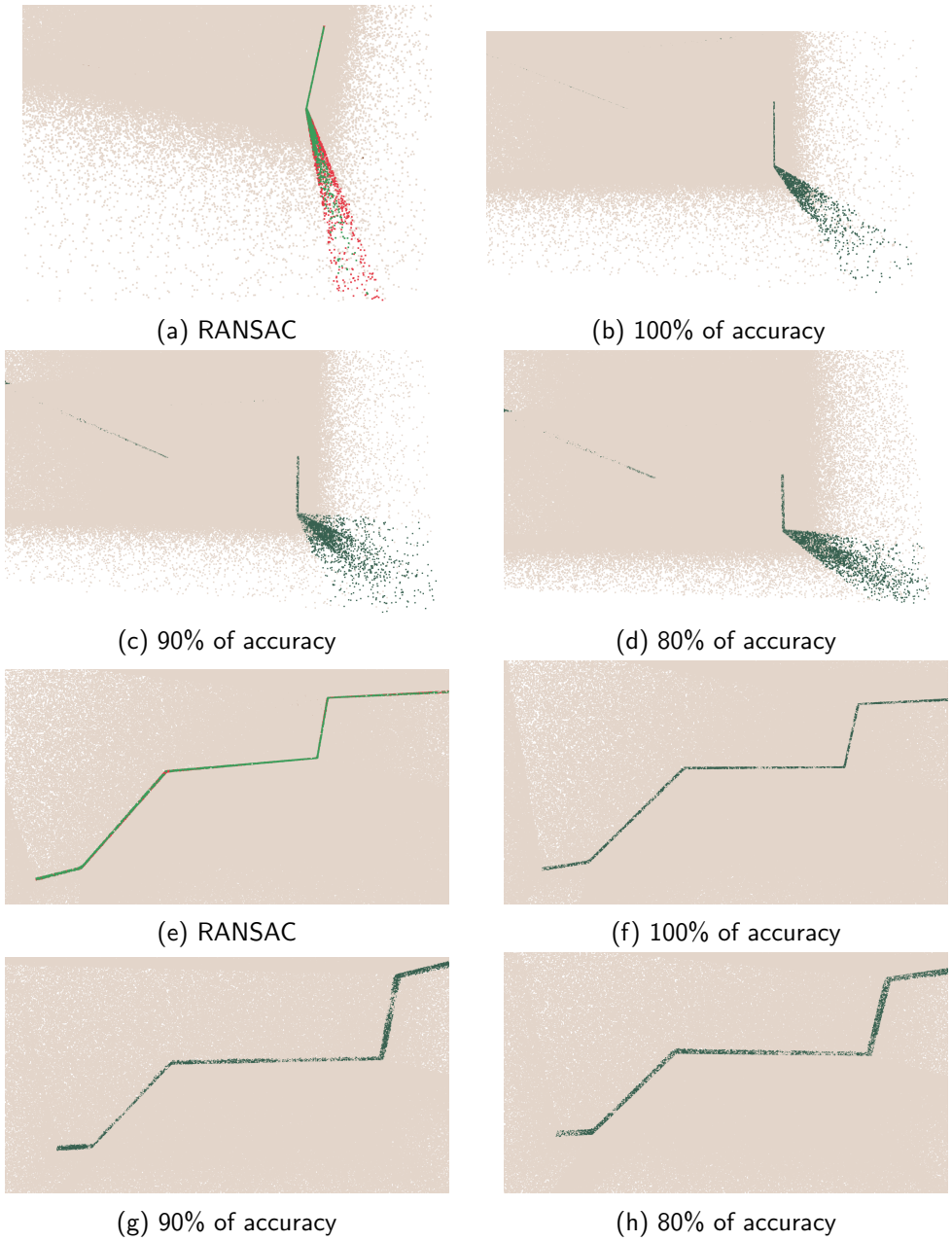
5.2.2 Qualitative comparison to a RANSAC approach

From a more qualitative perspective, Figure 20 provides insights into the pipe segmentation results obtained for Room 4. These results help us gain a better understanding of the metrics discussed earlier.

One noticeable observation is that the RANSAC approach tends to make a significant number of errors by incorrectly classifying points close to the wall as non-pipe points. On the other hand, the theoretical models, which aim to include more points, tend to classify a larger portion of the surrounding area as pipe points. This finding is particularly interesting for segmentation because it indicates a preference for including more points and achieving a higher recall, rather than risking the loss of important pipe information by missing some of its points.

Both methods successfully extract the pipe from the point cloud, even when it involves curved sections. However, upon closer examination, the second view of the results reveals that the RANSAC model misclassifies a portion of the pipe when it descends, as the points are included in the horizontal plane. In contrast, our model does not encounter such issues and consistently follows the pipe along its entire path without interruption.

These qualitative observations further support the effectiveness of our model in accurately segmenting the pipe from the point cloud data, demonstrating its ability to maintain continuity and accurately classify pipe points even in complex scenarios.



(a) RANSAC (b) 100% of accuracy

(c) 90% of accuracy (d) 80% of accuracy

(e) RANSAC (f) 100% of accuracy

(g) 90% of accuracy (h) 80% of accuracy

Figure 20: The figure displays the segmentation outcomes of the point cloud analysis for the 4th room, utilizing two views and four models. It consists of eight images illustrating the results. Light-brown points represent the "not-pipe" label. In the RANSAC results, accurately identified pipe points are shown in green, while red points indicate pipe points mislabelled as "not-pipe" by the method. In the theoretical model results, all green points correspond to points attributed to the "pipe" label by the model.

5.2.3 Comparison of the theoretical models

Focusing on the comparison of the theoretical models, Figure 21 visually illustrates how the metrics increase with the accuracy of the model. The figure clearly demonstrates that as the accuracy of the model improves, the metrics, particularly recall, also increase. However, it is important to note that while the model exhibits a high recall, its precision is consistently lower, which in turn affects the F1 score. This observation reinforces the fact that the theoretical model is designed based on the assumption that it can accurately identify pipe areas, but it may make errors in labelling the surrounding points. Consequently, the high recall value indicates a high percentage of correctly recognized pipe points, but the lower precision value reflects the classification errors in terms of false positives.

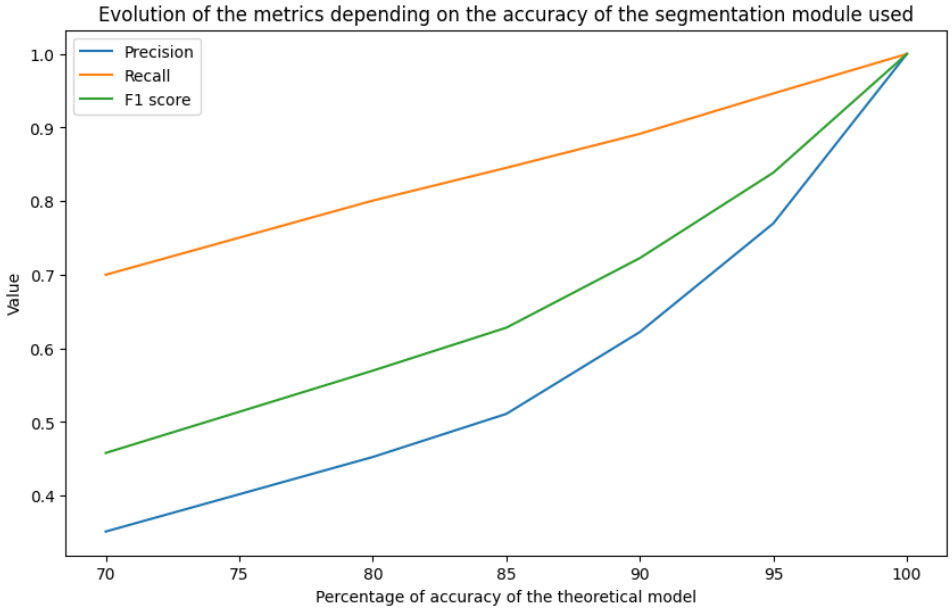


Figure 21: The figure illustrates the evolution of precision, recall, and F1 score metrics based on the accuracy of the segmentation module used. The x-axis represents the percentage of accuracy of the theoretical model, while the y-axis represents the corresponding values of the metrics. The precision metric is depicted by the blue line, the recall metric by the orange line, and the F1 score metric by the green line. As the accuracy of the segmentation module increases, there is a clear upward trend in all three metrics, indicating improved performance.

A comprehensive quantitative analysis is conducted to evaluate the interpolation performance of different models on the pipe dataset. The analysis focuses on measuring the average RMSE as a metric to understand the gap distance between the 100% accuracy model and models with 90%, 80%, and 70% accuracy across three reference rooms. The results are presented in Table 2. As anticipated, the interpolation accuracy improves as the model's point labelling precision increases.

It is worth noting that rooms with a higher density of surrounding points exhibit a greater deviation from the target interpolation. For instance, in Room 2, where certain sections of the pipe do not come into contact with the walls, the interpolation deviates to a lesser extent compared to Room 9, where the pipe is in close proximity to both the wall and floor.

We observe that rooms with a lower density of surrounding points exhibit a relatively small decrease in the average RMSE between the first two models. This suggests that the first percents of accuracy lost in these models have a limited impact on the overall interpolation.

Furthermore, we observe that Room 9 faces greater challenges in maintaining close proximity to the target interpolation as the model accuracy decreases. This suggests that the complexity of the pipe's surroundings influences the interpolation accuracy, with more challenging environments leading to higher deviations from the target interpolation.

% of accuracy of the model	90%	80%	70%
Room 2	0.0377	0.0397	0.0417
Room 5	0.0230	0.0233	0.0309
Room 9	0.0268	0.0407	0.0460

Table 2: Accuracy of the model for different rooms

5.3 PointNet performance results

This section focuses on presenting the results of the PointNet segmentation module of the developed tool. Although the module did not perform optimally when integrated into the main tool, it still produced interesting results on the dataset. The training was conducted using the Google Colab service with the following configuration:

- Runtime type: Python 3
- Hardware accelerator: GPU
- GPU type: T4

Various model setups were experimented with to analyze the module's results. The adapted parameters are explained in detail below.

The number of nodes in the layers was adjusted to 100%, 50%, and 25% of the original paper's size. The results are shown in Table 3. For all three models, the final model after the specified number of epochs was used. It is observed that reducing the number of nodes slightly improves the performance compared to keeping the original full size. This can be attributed to the simplicity of the dataset, which is less complex than the shapes originally used in the PointNet model dataset.

Having too many nodes in our case tends to lead to overfitting and decreases the performance. On the other hand, excessively simplifying the model degrades its ability to learn the characteristics specific to the application field, as evident from the results of the 25% node model.

PointNet Node %	Epochs	Precision	Recall	F1-Score
100%	100	0.368	0.607	0.436
50%	50	0.380	0.642	0.454
25%	50	0.329	0.567	0.396

Table 3: Performance metrics of PointNet on different subsets of the dataset.

Other parameter trials were conducted in an attempt to improve the performance, but none yielded better results. The number of concatenated layers for the segmentation part of the PointNet module, as depicted in Figure 12 of Section 3.3.5, was also varied from the original architecture. Lowering the number of joined layers did not improve the model's performance and resulted in a loss of knowledge as well.

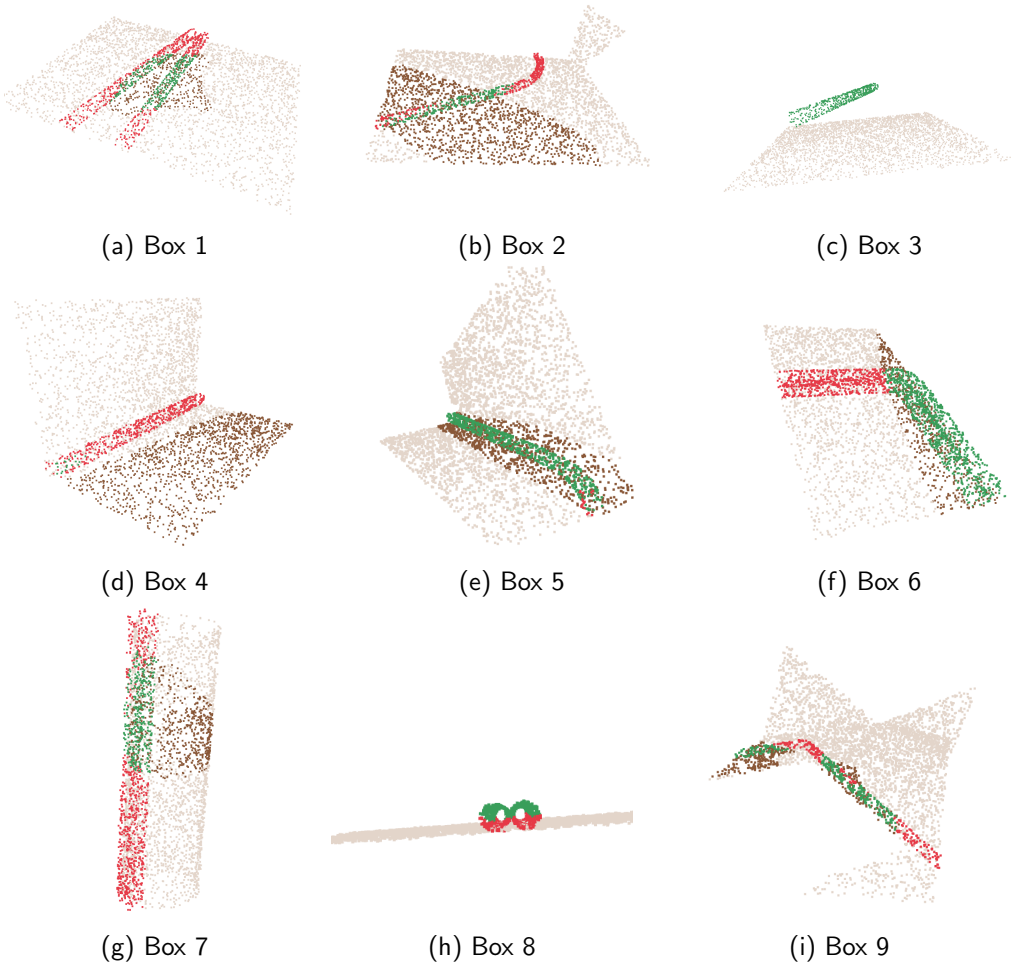
From a visual perspective, Figure 22 showcases the results of the model with the most promising metrics from Table 3, which is the model with 50% of the initial nodes, trained for 50 epochs, and achieving an F1 score of 0.454 on the entire test dataset. The figure presents the segmentation on nine boxes from the test dataset, summarizing the overall behavior of the model.

The first observation is that in most cases, the model struggles to identify the entire path of the pipe. It generally finds the center but fails to locate the two extreme points of the pipe portion present in the box. Another notable observation is that for nearly all cases, the model can identify some points of the pipe, but occasionally makes significant errors, as shown in Box 4.

Taking a closer look, we can observe that when the pipe is separated from the walls, the model performs a very accurate segmentation (Box 3). The same applies when the pipe is touching the walls, where the model either includes the surrounding points as part of the pipe (Box 5) or recognizes only the portion that does not touch the planes (Box 8).

When dealing with multiple linked pipes, the model performs reasonably well but struggles to identify sharp direction changes (Box 6). This failure to recognize curves can also be generalized to single pipe boxes (Box 2), although the model demonstrates better understanding in some cases (Box 9). However, the model correctly handles slight changes in direction (Box 5).

Finally, even though the model was trained on fixed sizes of pipes, it encounters difficulties in distinguishing between two cylindrical shapes in edge cases involving pipes on columns (Box 7).



(a) Box 1 (b) Box 2 (c) Box 3

(d) Box 4 (e) Box 5 (f) Box 6

(g) Box 7 (h) Box 8 (i) Box 9

Figure 22: Visualization of Segmented Search Boxes: Correctly attributed pipe labels are highlighted in green, while mistakes are marked in red. Light brown indicates not-pipe correctly identified, and dark brown represents misclassified points. The figure showcases some of the most interesting instances of the test dataset, utilizing the highest-scoring classification model.

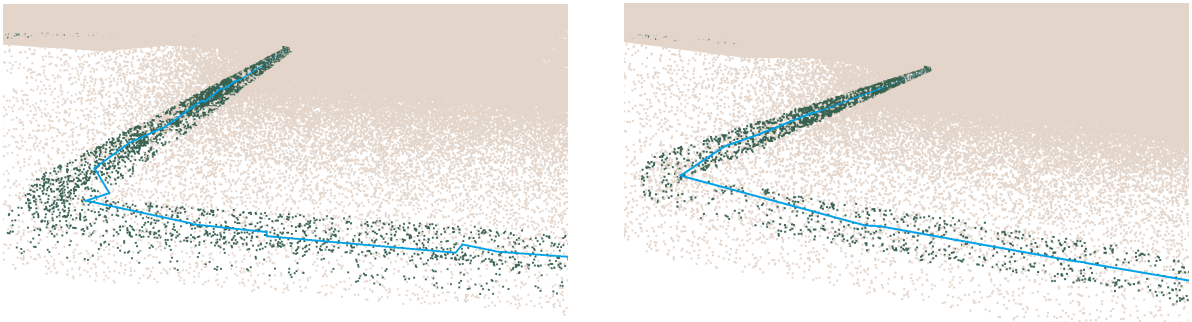
5.4 Qualitative analysis

The qualitative results of this work focus on the two main innovations of this study.

The first innovation is the segmentation of a targeted pipe in the point cloud with a shifting view. This approach allows for accurate identification and isolation of the desired pipe within the point cloud data.

The second innovation is the interpolation of the segmented pipe using a voxelization technique, which generates a sequence of points that represents the interpolated pipe.

The smoothness and centering of the interpolation depend on the accuracy of the segmentation. Figure 23 provides a visual representation of this relationship. It demonstrates that as the segmentation becomes more precise, the resulting interpolation line becomes more centered and smooth. This indicates that a higher level of accuracy in the segmentation process leads to improved interpolation results.



(a) 90% (b) 100%

Figure 23: Visualization of point cloud interpolations of a pipe, demonstrating how different model accuracies affect the central alignment of the interpolation shown in blue.

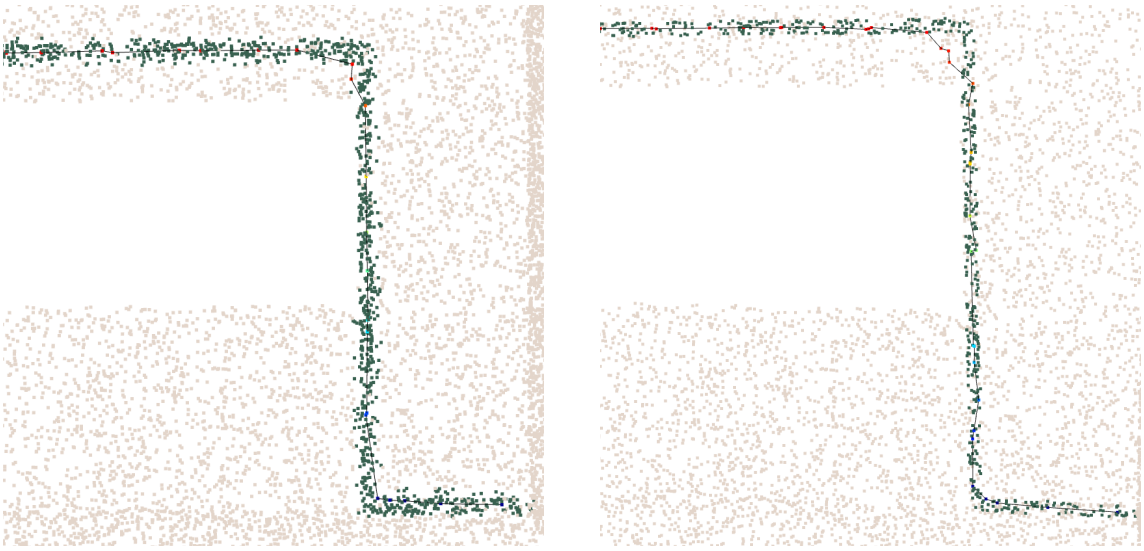
In cases where the point cloud has a lower density, it has been observed that the interpolation results are less precise. Interestingly, when the model performs at a slightly lower accuracy and aggregates points around the pipe during segmentation, the interpolation actually improves.

This phenomenon occurs because the noise added to the pipe during segmentation pulls the model towards a broader angle, resulting in a better voxelization of the surrounding space. As a result, the interpolation algorithm can generate a smoother and more accurate representation of the pipe. Figure 24 visually illustrates this behavior.

One of the challenges highlighted earlier is the difficulty of the model in properly following a targeted pipe when encountering multiple pipes in a confined area. This issue is exemplified in Figure 25 using an extreme case where four pipes closely meet (see picture (a)).

In some instances, the model is still able to perform correctly and find the right direction in the critical box, as depicted in (b). In this case, it means that the focus is shifted along the blue line to continue the iterations and eventually achieve the segmentation shown in sub-figure (a).

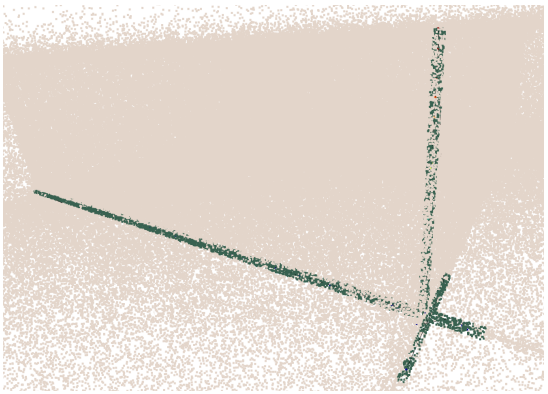
However, in other cases, the outcome depends on the points included in the critical box. The shift might be incorrectly done onto a wrong pipe, causing the model to follow the red direction instead. It is important to note that the model determines the next box's focus direction based on the furthest point labelled as a pipe in the current box, taking into account the previous positions it has focused on.



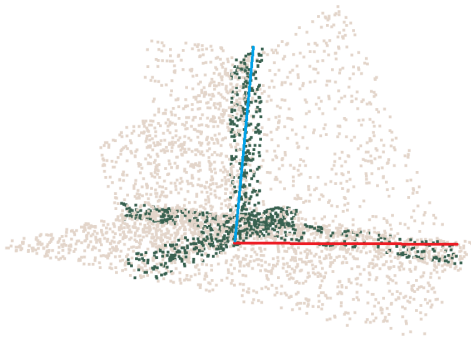
(a) 90%

(b) 100%

Figure 24: Visualization of pipe segmentation and interpolation in a lower density point cloud, highlighting the impact of model accuracy on the interpolation results.



(a) Correct path



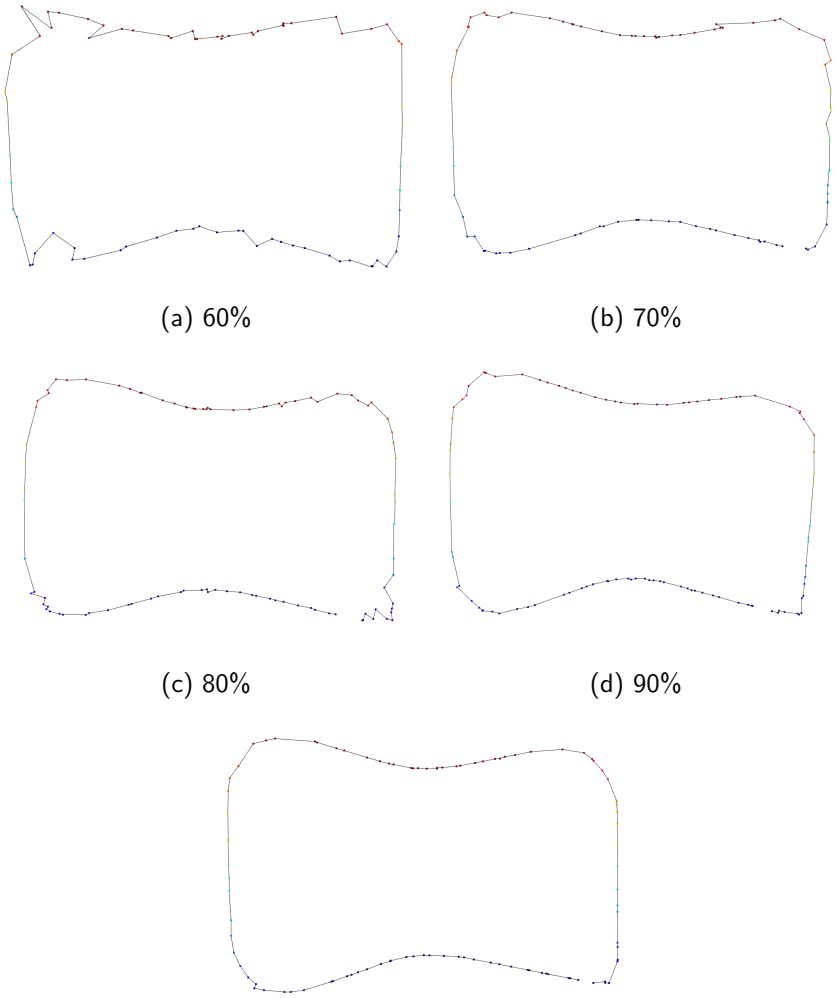
(b) Shift of the focus in the critical box where four pipes get close.

Figure 25: Challenges with multiple pipes in the same box: (a) illustrates a box of search where four pipes closely meet, while (b) demonstrates two possibilities of path selection, following the incorrect direction (red) instead of the intended direction (blue).

As mentioned previously, the quality of the segmentation, which directly impacts the accuracy of the model, significantly influences the interpolation process. This is clearly demonstrated in Figure 26, where we showcase the extracted interpolation of a circuit pipe from its point cloud.

It is evident that as the accuracy of the model increases, the resulting interpolation becomes smoother. A higher accuracy enables the model to accurately capture the shape and trajectory of

the pipe, leading to a more precise and refined interpolation. When the accuracy of the model falls below 70%, the interpolated points are noticeably impacted by an excessive amount of noise. As a consequence, the resulting shape of the interpolated pipe becomes rough and lacks precision, failing to accurately capture the true shape and trajectory of the pipe.



(a) 60%

(b) 70%

(c) 80%

(d) 90%

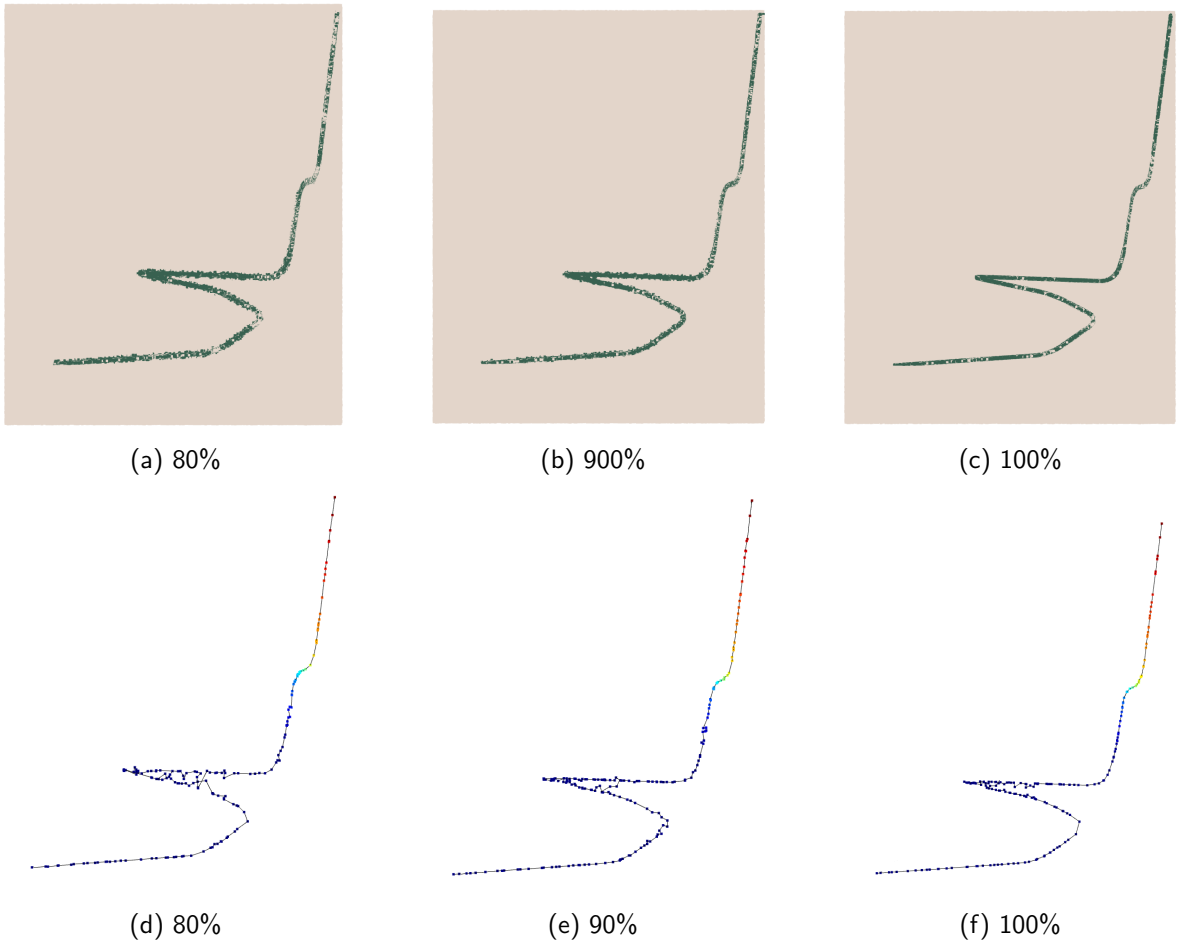
(e) 100%

Figure 26: Interpolation of a circuit-shaped pipe demonstrating the impact of model accuracy on the smoothness of the interpolation.

Hairpin turns pose an interesting challenge for the model, particularly when boxing such environments. The box surrounding a hairpin turn typically contains both the current segment of the targeted pipe as well as the other segment that would be visited before or after the turn. This can lead to noisy boxes that struggle to accurately follow the targeted pipe.

In the case of hairpin turns, the model's behavior is illustrated in Figure 27, which depicts the top view of the room. Despite lower accuracy from the segmentation module, the model manages to perform properly by identifying the entire path of the pipe, including the hairpin turn. However, the interpolation step tends to exhibit poorer performance with lower accuracy.

When the pipe undergoes a hairpin turn, the presence of surrounding noisy points in the model's input data affects the voxelization step. The voxelization process generates voxels that connect both parts of the pipe, creating a bridge between them. This results in a noisy interpolation, as observed in sub-figure (d) of Figure 27.



(a) 80% (b) 900% (c) 100%

(d) 80% (e) 90% (f) 100%

Figure 27: Segmentation of a hairpin curve-shaped pipe in a point cloud, demonstrating the model's ability to identify the correct direction and trace the full path of the pipe. Although lower accuracy affects the interpolation, the model still showcases promising results.

5.5 Discussion

The practical results obtained through this new approach have provided valuable insights. However some results deviate from initial theoretical expectations. One notable aspect is that the PointNet module is identified as the weak point of the model, which exhibits moderate performance during training and testing but presents difficulties in knowledge transfer when integrated into the global model. With further improvement from the segmentation module of the tool, we can expect the method to perform a precise and valuable pipe segmentation and interpolation. From the theoretical simulation and results, a model reaching an accuracy of at least 80% offers a high precision of pipe point area recognition.

With a focus on evaluating the method's performance in relation to the hypothesis of employing deep learning for pipe segmentation in point clouds using a shifting local view, the innovative approach demonstrates promising potential. However, as described, further improvements are required, particularly in the segmentation module and dataset. In fact, the PointNet model was expected to perform a better segmentation than the observed results. This model is a pioneer in the field of point cloud segmentation, however it appears that applied with the aim of segmenting a very local view point cloud, the results are not promising and incorporating properly in our model.

The findings from this study reveal several key aspects regarding the segmentation of pipes from point clouds, presenting a valuable learning experience. The utilization of a shifting view model exhibits promising potential. The module demonstrates the ability to generalize and adapt to diverse scenarios, particularly when applied to boxes selected from the global model, thereby opening up exciting opportunities for further enhancement. Notably, the theoretical models yield interesting results, even when segmentation tools exhibit moderate accuracy, as long as they effectively recognize pipe point areas. These models outperform the F1 scores achieved by the RANSAC method. Furthermore, these models offer the advantage of interpolating the intended conduit path using a sequence of points accurately centered within the pipe, thus augmenting the overall understanding and precision of the representation.

Consequently, this study raises additional research questions concerning the use of deep learning tools for the segmentation of highly localized views. These inquiries pertain to various aspects, including dataset development, model architectures, conduit interpolation techniques, and pipe-path tracking. With dedicated efforts and meticulous fine-tuning, the module has the potential to surpass its existing limitations and attain more precise and consistent pipe detection outcomes.

In summary, although the practical results may not align completely with the initial theoretical expectations, they provide invaluable insights and avenues for improvement. By addressing the limitations in the PointNet model as the primary segmentation module, refining the segmentation

process using a comprehensive dataset, and leveraging theoretical models, this methodology can enhance its impact by achieving heightened accuracy and more reliable pipe segmentation and interpolation in point clouds.

5.6 Limitations

One of the primary limitations encountered in this work was the dataset. Since no suitable dataset existed for this specific task, a significant challenge was to develop a realistic dataset that could sufficiently represent real-life examples. The final dataset comprises eight manually created rooms, containing only walls and pipes. While this basic environment helps to limit the noise introduced into the model, it does not capture the full complexity of real construction sites, which typically consist of various objects and shapes beyond isolated conduits.

Another aspect to consider is that the pipes in the dataset are represented as cylindrical shapes, resembling those encountered in worksites. However, in real-life scenarios, 3D scans may not accurately capture the connection between the cable part of the conduit and the floor. Consequently, while the 3D scans might recognize conduits as semicircles, the dataset considered them as full circles.

Furthermore, it is important to note that the dataset does not differentiate between instances of pipes. In other words, while it can identify what constitutes a pipe, it lacks the ability to distinguish between different pipes.

These limitations in the dataset highlight the need for further development and refinement to create more comprehensive and diverse datasets that accurately represent real-life construction sites. In fact, the limitations prevent the model from being directly transferred to real-life applications. By incorporating a wider range of objects, shapes, and pipe instances, future datasets can provide a more realistic and nuanced representation, enabling the model to better handle the complexities and variations encountered in actual construction environments.

A desired aspect of the model is that it relies on the known conduit diameter, which is obtained from the dataset where all pipes have the same size. However, this limited information restricts the model's ability to handle variations in pipe sizes encountered in real-world scenarios. To overcome this limitation, a potential enhancement would involve incorporating the targeted pipe diameter as an input to the model. By allowing the model to take the pipe diameter as an input parameter, it can adapt to different pipe sizes and improve its accuracy in practical applications.

A final notable limitation linked to the model's inability to differentiate between multiple instances of pipes in the point cloud is found in scenarios where multiple pipes are close by. Because of the fact that the model has been trained on a dataset that does not distinguish between different

pipe instances, it can only follow one pipe at a time. In fact, the model's current approach using boxes to segment the point cloud poses additional limitations on instance recognition. It may unintentionally switch from the targeted pipe to another pipe in the vicinity when a box encounters multiple pipes within its boundaries, as the model tends to move towards the furthest point.

To address this limitation, it is crucial to enhance the segmentation model's ability to focus on a specific pipe and prevent it from switching to another pipe when multiple pipes are present. By improving the model's instance recognition capabilities, it can accurately track and segment the targeted pipe throughout the point cloud, leading to more reliable and precise results.

5.7 Takeaways

The developed model in this work demonstrates an impactful application in segmenting and interpolating a full pipe path based on a given starting point. The qualitative results indicate that any model with an accuracy higher than 80% can successfully perform pipe segmentation and interpolation. However, it is important to note that the errors should primarily occur in the vicinity of the pipe rather than in unrelated areas. Thus, the ideal model would achieve a very high recall and a good precision, accurately identifying the regions where pipes are present, which is crucial for the model's performance.

The PointNet module, although exhibiting varying performance in box tests, does not align well with the main tool. Exploring alternative segmentation tools that excel in learning the locality of the data would lead to improvements in the segmentation process at the heart of the tool. Models having this feature, such as PointNet++, would be worth a future research question.

In addition to exploring different models, incorporating additional input features can enhance the performance of the model. The inclusion of color information from the point cloud can provide more detailed insights into the characteristics of pipes. While pipes may not always have consistent colors, the variation in color between the pipe and its surroundings, combined with its shape, can aid the model in improving its understanding. Furthermore, integrating the diameter of the pipe as an input feature can help filter out points that have been erroneously identified as pipes. One approach could involve flattening the box in two dimensions and fitting a circle to the data based on the pipe's diameter. However, the dataset used may introduce challenges due to data augmentation and rotation, which may misalign the point cloud. In such cases, considering the direction of the pipe could assist in properly flattening the box and aligning the data accurately. By incorporating additional input features, such as color and diameter, into the model and the dataset, it has the potential to improve the accuracy and robustness of the segmentation and interpolation processes, leading to more reliable and precise results.

Indeed, the limitations observed in the model's performance could be attributed, in part, to the dataset used. To address this, a more diverse and comprehensive dataset comprising various rooms with different objects and details could provide the segmentation module with a broader understanding of the characteristics of pipes in different contexts. This richer dataset would enable the model to learn and generalize better, leading to improved segmentation results.

Moreover, the efficacy of data augmentation raises concerns that warrant further investigation. Given that the input point clouds are typically well-aligned along the axes, it may prove advantageous to generate additional boxes within the point clouds rather than relying extensively on data augmentation techniques that introduce rotations. This alternative approach has the potential to enhance overall segmentation performance and contribute to more accurate and reliable results. On a broader scale, improvements in the dataset are crucial for achieving a more realistic representation. It is advisable to incorporate other types of objects and include pipes with semi-circle shapes that resemble those found on construction sites. This expansion of the dataset is necessary to ensure that the model can effectively transfer its knowledge to practical situations.

Additionally, adapting the boxes used for segmentation is a viable approach. Firstly, by introducing a distinguishing label for the different pipe encountered in the rooms. This would allow a better understanding of how to segment the pipe points in the box depending on the targeted one. Then, employing rectangular parallelepipeds that align with the direction of the pipes instead of cube-shaped boxes could potentially improve the distribution of points between the two classes (pipe and non-pipe). This balanced distribution of points can assist the model in discerning the key features of conduits more effectively, enhancing its segmentation capabilities.

While considering alternative segmentation methods, using RANSAC is an intriguing option. RANSAC has the advantage of avoiding attracting undesired points, particularly when dealing with local plane selection. However, employing RANSAC would require significant human intervention and manual control, potentially reducing the reliance on the deep learning aspect of the model. This deviation from deep learning-based development does not align with the original intention and goals of the model.

6 Conclusion

In light of the research conducted, this master's thesis has addressed the problem of pipeline and power cable segmentation from point clouds using a novel deep learning approach. The primary objective of this study was to develop an accurate and efficient method for extracting and segmenting pipelines and power cables from complex 3D data. The model developed in this study incorporates a dynamic local view mechanism to effectively track the trajectory of the pipeline.

The results obtained using the PointNet module for segmentation yielded suboptimal outcomes. However, the overall method demonstrated promising results, with a theoretical model simulating various segmentation accuracies. One of the strengths of the model is its ability to extract the pipe without the need to process the entire point cloud, resulting in a more precise and focused segmentation. It outperforms traditional approaches like RANSAC by identifying more conduit points, leading to improved pipe segmentation.

The findings indicate that PointNet is not suitable for very localized point cloud segmentation. Replacing this module with another that has a higher recall metric would enhance the tool's performance. The global model developed in this work introduces an innovative approach to pipe segmentation, incorporating locality to achieve accurate recognition of pipe areas. This feature is crucial for the interpolation of the conduit.

However, it is important to acknowledge certain limitations. The handmade dataset used in this study represents a theoretical world, which may not fully capture the complexities of real-life scenarios where such a tool would be applied. The model performs well for single pipe segmentation but may encounter challenges when multiple pipes are present in the search area during execution.

This work paves the way for numerous interesting improvements, research avenues, and adaptations. Two main areas of enhancement are a richer dataset and a more sophisticated segmentation module. Enriching the dataset with a diverse range of pipes, complex shapes, instance labelling, color features, and meeting points would significantly enhance the model's knowledge. Regarding the segmentation module, exploring models such as PointNet++, RSNet, or PointCNN that possess locality understanding would provide valuable assistance in segmenting boxes from point clouds.

Through this experience, my skills have been enriched in various ways. I have gained a deep understanding of point cloud technology, including its processing and analysis. The literature review has broadened my knowledge of the field and sharpened my critical and analytical thinking. Additionally, it has familiarized me with academic conventions. The methodology developed in this research has honed my skills in artificial intelligence development, particularly in problem identification, understanding, and solution adaptation. The result assessment has enabled me to justify

and establish connections between the data, the model's behavior, and its output. Furthermore, this work has taught me valuable lessons in project organization, dedication, and the importance of listening to multiple viewpoints and striving to satisfy diverse stakeholders.

To draw a final conclusion, this work explores an innovative approach to point cloud segmentation, specifically for pipeline and power cable extraction and interpolation from 3D worksite scans. The approach shows promise and opens avenues for further improvement and advancement.

7 References

[1] Anh Nguyen and Bac Le. "3D point cloud segmentation: A survey". In: *2013 6th IEEE Conference on Robotics, Automation and Mechatronics (RAM)*. 2013, pp. 225–230. DOI: 10.1109/RAM.2013.6758588.

[2] Yuliang Sun et al. "PGCNet: Patch Graph Convolutional Network for point cloud segmentation of indoor scenes". In: *The Visual Computer* 36 (Oct. 2020). DOI: 10.1007/s00371-020-01892-8.

[3] SpaceTime S.A. URL: https://www.spacetime.lu/.

[4] Lynn Puzzo. *What is LiDAR, how does it work, and what is it used for?* Jan. 2021. URL: https://www.mosaic51.com/technology/what-is-lidar-how-does-it-work-and-what-is-it-used-for/ (visited on 04/26/2023).

[5] Emmanuel P. Baltsavias. "A comparison between photogrammetry and laser scanning". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 54.2 (1999), pp. 83–94. ISSN: 0924-2716. DOI: https://doi.org/10.1016/S0924-2716(99)00014-3. URL: https://www.sciencedirect.com/science/article/pii/S0924271699000143.

[6] Ivan Nikolov. *Python Libraries for Mesh, Point Cloud, and Data Visualization (Part 1)*. URL: https://towardsdatascience.com/python-libraries-for-mesh-and-point-cloud-visualization-part-1-daa2af36de30. (accessed: 10.04.2023).

[7] WebGL Public Wiki. *Main Page — WebGL Public Wiki*. [Online; accessed 28-April-2023]. 2022. URL: http://www.khronos.org/webgl/wiki_1_15/index.php?title=Main_Page&oldid=2632.

[8] Florent Poux. *Guide to real-time visualisation of massive 3D point clouds in Python*. URL: https://medium.com/towards-data-science/guide-to-real-time-visualisation-of-massive-3d-point-clouds-in-python-ea6f00241ee0. (accessed: 28.04.2023).

[9] Xiangyu Zheng et al. "3D Point Cloud Mapping Based on Intensity Feature". In: *Artificial Intelligence in China*. Ed. by Qilian Liang et al. Singapore: Springer Singapore, 2022, pp. 514–521. ISBN: 978-981-16-9423-3.

[10] Martin A. Fischler and Robert C. Bolles. "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography". In: *Commun. ACM* 24.6 (June 1981), pp. 381–395. ISSN: 0001-0782. DOI: 10.1145/358669.358692. URL: https://doi.org/10.1145/358669.358692.

[11] Florent Poux. *3D Model Fitting for Point Clouds with RANSAC and Python*. URL: https://towardsdatascience.com/3d-model-fitting-for-point-clouds-with-ransac-and-python-2ab87d5fd363. (accessed: 28.04.2023).

[12] Haolan Chen et al. "Unsupervised learning of geometric sampling invariant representations for 3D point clouds". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 893–903.

[13] Aoran Xiao et al. "Unsupervised Point Cloud Representation Learning with Deep Neural Networks: A Survey". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2023), pp. 1–20. DOI: 10.1109/tpami.2023.3262786. URL: https://doi.org/10.1109%5C%2Ftpami.2023.3262786.

[14] Zisheng Chen and Hongbin Xu. *Unsupervised Semantic Segmentation of 3D Point Clouds via Cross-modal Distillation and Super-Voxel Clustering*. 2023. arXiv: 2304.08965 [cs.CV].

[15] Jiaying Zhang et al. "A Review of Deep Learning-Based Semantic Segmentation for Point Cloud". In: *IEEE Access* 7 (2019), pp. 179118–179133. DOI: 10.1109/ACCESS.2019.2958671.

[16] Hang Su et al. "Multi-view Convolutional Neural Networks for 3D Shape Recognition". In: *CoRR* abs/1505.00880 (2015). arXiv: 1505.00880. URL: http://arxiv.org/abs/1505.00880.

[17] Alexandre Boulch et al. "SnapNet: 3D point cloud semantic labeling with 2D deep segmentation networks". In: *Comput. Graph.* 71 (2017), pp. 189–198.

[18] Joris Guerry et al. "Snapnet-r: Consistent 3d multi-view semantic labeling for robotics". In: *Proceedings of the IEEE international conference on computer vision workshops*. 2017, pp. 669–678.

[19] Daniel Maturana and Sebastian Scherer. "VoxNet: A 3D Convolutional Neural Network for real-time object recognition". In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2015, pp. 922–928. DOI: 10.1109/IROS.2015.7353481.

[20] Lyne Tchapmi et al. "SEGCloud: Semantic Segmentation of 3D Point Clouds". In: *2017 International Conference on 3D Vision (3DV)*. 2017, pp. 537–547. DOI: 10.1109/3DV.2017.00067.

[21] Truc Le and Ye Duan. "PointGrid: A Deep Network for 3D Shape Understanding". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018.

[22] Charles R. Qi et al. *PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation*. 2017. arXiv: 1612.00593 [cs.CV].

[23] Yangyan Li et al. *PointCNN: Convolution On $\mathcal{X}$-Transformed Points*. 2018. arXiv: 1801.07791 [cs.CV].

[24] Qiangui Huang, Weiyue Wang, and Ulrich Neumann. *Recurrent Slice Networks for 3D Segmentation of Point Clouds*. 2018. arXiv: 1802.04402 [cs.CV].

[25] Jiaxin Li, Ben M. Chen, and Gim Hee Lee. *SO-Net: Self-Organizing Network for Point Cloud Analysis*. 2018. arXiv: 1803.04249 [cs.CV].

[26] Charles R. Qi et al. *PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space*. 2017. arXiv: 1706.02413 [cs.CV].

[27] Yanxin Ma et al. "3DMAX-Net: A Multi-Scale Spatial Contextual Network for 3D Point Cloud Semantic Segmentation". In: *2018 24th International Conference on Pattern Recognition (ICPR)*. 2018, pp. 1560–1566. DOI: 10.1109/ICPR.2018.8546281.

[28] Xiaoqing Ye et al. "3D Recurrent Neural Networks with Context Fusion for Point Cloud Semantic Segmentation". In: *Computer Vision – ECCV 2018: 15th European Conference, Munich, Germany, September 8–14, 2018, Proceedings, Part VII*. Munich, Germany: Springer-Verlag, 2018, pp. 415–430. ISBN: 978-3-030-01233-5. DOI: 10.1007/978-3-030-01234-2_25. URL: https://doi.org/10.1007/978-3-030-01234-2_25.

[29] Mingyang Jiang et al. *PointSIFT: A SIFT-like Network Module for 3D Point Cloud Semantic Segmentation*. 2018. arXiv: 1807.00652 [cs.CV].

[30] Artem Komarichev, Zichun Zhong, and Jing Hua. *A-CNN: Annularly Convolutional Neural Networks on Point Clouds*. 2019. arXiv: 1904.08017 [cs.CV].

[31] Yifan Xu et al. *SpiderCNN: Deep Learning on Point Sets with Parameterized Convolutional Filters*. 2018. arXiv: 1803.11527 [cs.CV].

[32] Yue Wang et al. *Dynamic Graph CNN for Learning on Point Clouds*. 2019. arXiv: 1801.07829 [cs.CV].

[33] Kuangen Zhang et al. *Linked Dynamic Graph CNN: Learning on Point Cloud via Linking Hierarchical Features*. 2019. arXiv: 1904.10014 [cs.CV].

[34] Gusi Te et al. *RGCNN: Regularized Graph CNN for Point Cloud Segmentation*. 2018. arXiv: 1806.02952 [cs.CV].

[35] Can Chen, Luca Zanotti Fragonara, and Antonios Tsourdos. *GAPNet: Graph Attention based Point Neural Network for Exploiting Local Feature of Point Cloud*. 2019. arXiv: 1905.08705 [cs.CV].

[36] Youngdoo Kim, Cong Hong Phong Nguyen, and Young Choi. "Automatic pipe and elbow recognition from three-dimensional point cloud model of industrial plant piping system using convolutional neural network-based primitive classification". In: *Automation in Construction* 116 (2020), p. 103236. ISSN: 0926-5805. DOI: https://doi.org/10.1016/j.autcon.2020.103236. URL: https://www.sciencedirect.com/science/article/pii/S092658051931547X.

[37] Yong-Jin Liu et al. "Cylinder Detection in Large-Scale Point Cloud of Pipeline Plant". In: *IEEE Transactions on Visualization and Computer Graphics* 19.10 (2013), pp. 1700–1707. DOI: 10.1109/TVCG.2013.74.

[38] Robert C. Bolles and Martin A. Fischler. "A RANSAC-Based Approach to Model Fitting and Its Application to Finding Cylinders in Range Data". In: *International Joint Conference on Artificial Intelligence*. 1981.

[39] Thomas Chaperon and François Goulette. "Extracting Cylinders in Full 3D Data Using a Random Sampling Method and the Gaussian Image." In: vol. 2001. Jan. 2001, pp. 35–42.

[40] Tahir Rabbani et al. "An integrated approach for modelling and global registration of point clouds". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 61.6 (2007), pp. 355–370. ISSN: 0924-2716. DOI: https://doi.org/10.1016/j.isprsjprs.2006.09.006. URL: https://www.sciencedirect.com/science/article/pii/S0924271606001213.

[41] Yun-Ting Su and James Bethel. "DETECTION AND ROBUST ESTIMATION OF CYLINDER FEATURES IN POINT CLOUDS". In: 2010.

[42] Allam Shehata Hassanein et al. *A Survey on Hough Transform, Theory, Techniques and Applications*. 2015. arXiv: 1502.02160 [cs.CV].

[43] Luis Gonzales. *An In-Depth Look at PointNet*. URL: https://medium.com/@luis_gonzales/an-in-depth-look-at-pointnet-111d7efdaa1a. (accessed: 01.05.2023).

[44] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation. Stichting Blender Foundation, Amsterdam, 2018. URL: http://www.blender.org.

[45] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. "Open3D: A Modern Library for 3D Data Processing". In: *arXiv:1801.09847* (2018).

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**