Upsampling Indoor LiDAR Point Clouds for Object Detection

by

Yikai Yao

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Science
in
Geography

Waterloo, Ontario, Canada, 2023

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

As an emerging technology, LiDAR point cloud has been applied in a wide range of fields. With the ability to recognize and localize the objects in a scene, point cloud object detection has numerous applications. However, low-density LiDAR point clouds would degrade the object detection results. Complete, dense, clean, and uniform LiDAR point clouds can only be captured by high-precision sensors which need high budgets. Therefore, point cloud upsampling is necessary to derive a dense, complete, and uniform point cloud from a noisy, sparse, and non-uniform one.

To address this challenge, we proposed a methodology of utilizing point cloud upsampling methods to enhance the object detection results of low-density point clouds in this thesis. Specifically, we conduct three point cloud upsampling methods, including PU-Net, 3PU, and PU-GCN, on two datasets, which are a dataset we collected on our own in an underground parking lot located at Highland Square, Kitchener, Canada, and SUN-RGBD. We adopt VoteNet as the object detection network. We subsampled the datasets to get a low-density dataset to stimulate the point cloud captured by the low-budget sensors. We evaluated the proposed methodology on two datasets, which are SUN RGB-D and the collected underground parking lot dataset. PU-Net, 3PU, and PU-GCN increase the mean Average Precision (under the threshold of 0.25) by 18.8%,18.0%, and 18.7% on the underground parking lot dataset and 9.8%, 7.2%, and 9.7% on SUN RGB-D.

Acknowledgements

First of all, I am deeply grateful to Prof. Dr. Jonathan Li from the Department of Geography and Environmental Management, University of Waterloo for his direction as my supervisor during my master's study.

I would also like to express my sincere thanks to Dr. Linlin Xu from the Department of System Design Engineering, University of Waterloo, and Prof. Michael A. Chapman from the Department of Civil Engineering, Ryerson University for being thesis defense members.

In addition, I would like to extend my heartfelt appreciation to those who give me help and support during my graduate study. I am thankful to Dr. Yiping Chen from the School of Geospatial Engineering and Science, Sun Yat-sen University for her instruction and guidance in making this thesis and exploring this field. I am thankful to Weikai Tan and Dedong Zhang from the Department of System Design Engineering, University of Waterloo, and Hongjie He from the Department of Geography and Environmental Management, University of Waterloo for their help and advice in experiments and data capturing. I am thankful to Sarah Fatholahi from the Department of Geography and Environmental Management, University of Waterloo for her cooperation with me in a previous conference paper during my graduate study. I am thankful to Jirui Hu and Jingtian Tan from the Department of Geography and Environmental Management, University of Waterloo for their equipment support for my presentation in thesis defense. And I am also thankful to all the members of the Geospatial Intelligence and Mapping Lab for their support during my graduate study.

At last, I would like to convey my gratitude to my parents, all my friends, and my instructors. Without their guidance, company, and support, I could not have completed my graduate study.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AP	Average Precision	40
AR	Average Recall	40
AUC	Area Under Curve	40
CAD	Computer Aided Design	22
CD	Chamfer Distance	25
EAR	Edge-Aware Resampling	9
EMD	Earth Mover's Distance	26
FN	False Negative	40
FP	False Positive	40
GAN	Generative Adversarial Networks	18
GCN	Graph Convolutional Network	20
GNSS	Global Navigation Satellite System	1
GTV	Graph Total Variation	11
HD	Hausdorff Distance	25
IoU	Intersection over Union	40
KNN	K-Nearest-Neighborhood	12
LiDAR	Light Detection and Ranging	1
LOP	Locally Optimal Projection	8

MLP	Multi-Layer Perceptions	17
MLS	Moving Least Square	6
NUC	Normalized Uniformity Coefficient	26
TP	True Positive	40

Chapter 1

Introduction

1.1 Motivation

3D geospatial information is essential in recognizing objects and phenomena in the real world. Accurate 3D spatial information has important applications in a large range of fields, including cultural heritage documentation(Soler et al., 2017), mobile robotics (Guerry et al., 2017), autonomous driving (Pham et al., 2020), and navigation (McCrae et al., 2009). As a traditional way to obtain 3D spatial information, Global Navigation Satellite System (GNSS) and other satellite-based positioning technologies lose precision or become invalid inside buildings, underground, and in other locations where there is a large number of obstacles to their transmission signal (Chelly & Samama, 2009). Therefore, Light Detection and Ranging (LiDAR), a new technology that can obtain 3D spatial information in the form of point clouds from the GNSS-denied environment, has been attached more attention from scholars, investigators, and those who may concern (Chen et al., 2022).

As a datatype that contains rich 3D information, the 3D LiDAR point cloud has a variety of applications in real-time intelligent systems such as autonomous driving and augmented reality (Hu et al., 2020). Point cloud object detection is a significant application of LiDAR point cloud. The object detection method would identify objects within a point cloud and their corresponding classes (Mao et al., 2022). With the ability to intelligently predict the positions, size, and categories of critical objects around autonomous vehicles and intelligent robots, point cloud object detection is essential to autonomous driving and intelligent robotics.

However, in some circumstances, data quality becomes an important challenge that affects the performance of point cloud object detection methods. Researchers have indicated

that noisy, sparse, and non-uniform data with low quality can result in a bad performance in many point cloud-based subjects including object detection (Choi et al., 2021). However, low-budget LiDAR sensors usually produce noisy, sparse, and non-uniform point cloud data, and the cost of high-precision devices that produce high quality is also high.

In order to solve the data quality challenge resulting from device limitation, point cloud upsampling aims at generating a dense, complete, and uniform point cloud from the sparse, noisy, and non-uniform input (Zhang et al., 2022). In recent years, different deep learning-based and non-deep learning-based point cloud upsampling methods have been proposed. However, they are tested on a limited range of benchmarks. Otherwise, the effectiveness of these methods in other practical applications is rarely evaluated.

Considering the need of enhancing the object detection results on low-density point clouds and the lack of exploration of point cloud upsampling on practical applications, this thesis utilizes point cloud upsampling methods to enhance the performance of object detection on low-density point clouds.

1.2 Objectives of Thesis

The thesis aims to address the challenges of the degradation of object detection results stemming from low point cloud density by point cloud upsampling methods. Specifically, these objectives can be summarized as follows:

1. To enhance the object detection results of low-density point clouds via point cloud upsampling.

2. To conduct an accuracy assessment to evaluate the effectiveness of point cloud upsampling in enhancing object detection results of the low-density point cloud and to compare the performance of different point cloud upsampling methods.

3. To collect and create an indoor point cloud dataset of an underground parking lot located at Highland Square, Kitchener, Canada for object detection tasks.

1.3 Structure of the Thesis

The thesis consists of five chapters.

Chapter 1 illustrates the thesis motivation, objectives, and structure.

Chapter 2 illustrates the related works of this thesis. It first provides the definition of related terminology, then reviews the related point cloud object detection methods, point cloud upsampling methods, and data benchmarks for point cloud upsampling.

Chapter 3 describes the proposed methodology of this thesis, including workflow, dataset, point cloud object detection methods, point cloud upsampling methods, and evaluation metrics.

Chapter 4 shows the results of the experiments, including experimental setups, results on the underground parking lot dataset, results on the SUN RGB-D dataset, and some discussion of the results.

Chapter 5 concludes the contribution of the thesis, discusses its limitations, and plans for future work.

Chapter 2

Background and Related Work

The content of this chapter is a literature review of the related methods and benchmarks. Section 2.1 defines the terms that would be used in this thesis. Section 2.2 shows the related point cloud upsampling methods, including deep learning-based methods, non-deep learning-based methods, and benchmarks used in these methods. Section 2.3 talks about the benchmarks adopted in the related point cloud upsampling methods. Section 2.4 discusses related point cloud object detection methods. And section 2.5 summarizes this chapter.

2.1 Terminology Definition

LiDAR: LiDAR (Light Detection and Ranging) is a range measurement method. LiDAR measures distance by first targeting an object with a pulsed laser and then recording the time that the reflectance returns to the receiver (Williams Jr, 2017). 3D LiDAR sensors have strong environmental adaptability as they not only allow detection of all kinds of obstacles (Moosmann et al., 2009) but are also robust to a large range of conditions including day or night, with or without glare and shadows (Wu et al., 2018).

Point Cloud: The point cloud is a set of discrete data points in space with their cartesian coordinates recorded, and sometimes with some other attributes such as RGB or reflectance rate (Griffiths & Boehm, 2019). The LiDAR point cloud, which refers to the point cloud captured by LiDAR sensors, is an important type of point cloud.

Point Cloud Density: Point cloud density is used to describe the intensity of the dense degree of the point cloud. There are many different definitions of point cloud density. As

in this thesis, points within point clouds are distributed on the surface of the object we choose the definition used in CloudCompare (GPL software, 2022), which is the number of points per unit area on the surface.

Point Cloud Upsampling: Point cloud upsampling refers to the technique that generates high-density point clouds with uniform point distribution from the sparse input without changing the shape of the objects (Yu et al., 2018).

Point Cloud Subsampling: Point cloud subsampling refers to getting a portion of the point cloud data to reduce its size as well as preserve the original data structure (Lang et al., 2020).

Point Cloud Object Detection: Point cloud object detection refers to the technique that recognizes the sizes, categories, and positions of the object inside a 3D space. Given an input point cloud scene, point cloud object detection would output 3D bounding boxes of the predicted objects within, their predicted classes, and a confidential score of the prediction (Mao et al., 2022).

Low-density Point Cloud: Low-density point cloud refers to the point cloud dataset that the point distribution is sparse, which would decrease the performance of some of the applications such as object detection. The threshold of the low-density point cloud varies with different scenarios. A point cloud density of 1000 pts/m_2 is used in the evaluation of some point cloud object detection methods (Ding et al., 2019). Therefore, in this thesis, the point cloud with a density lower than this value is regarded as a low-density point cloud.

Deep learning: Deep learning is a special kind of machine learning method. It utilizes multiple processing layers to learn multiple levels of data representations (Goodfellow et al., 2016). In recent years, deep learning has made a significant improvement in numerous fields of artificial intelligence, which includes speech recognition, semantic segmentation, and big data analysis.

2.2 Point Cloud Upsampling Methods

Point cloud upsampling methods can be divided into two types: non-deep learning-based methods and deep learning-based methods. Non-deep learning-based methods are proposed in the early years before the development of deep learning and deep learning-based methods utilized deep learning networks.

2.2.1 Non-Deep Learning-Based Methods

2.2.1.1 Moving Least Square-Based Surface Interpolation

Early work by Alexa et al. (2003) proposed the first point cloud upsampling algorithm by inserting points into the original point cloud. This method first conducts the Moving Least Square (MLS) projection to generate a smooth manifold surface as the approximation of the original point cloud called the MLS surface. Then it interpolates points on that surface to increase the density of the point cloud.

MLS Surface Approximation: MLS reconstructs a continuous surface from a set of dispersed points at a local region that minimized the sum of weighted least squares measure biased of these points. In the situation of generating the local approximation surface near point r, as is shown in Figure 2.1 below, MLS first finds a local reference domain near the point and then conducts local mapping.

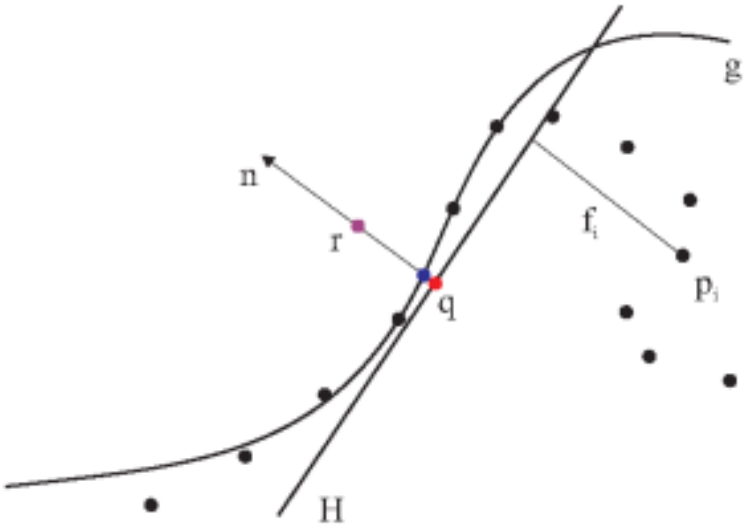


Figure 2.1 Schematic diagram of Moving Least Square projection
(Source: Alexa et al., 2003)

The reference domain is a local plane for reference. It can be described as the following equation:

$$H = \left\{ x \mid \langle n, x \rangle - D = 0, x \in \mathbb{R}^3 \right\}, n \in \mathbb{R}^3, \|n\| = 1 \tag{2.1}$$

where n is the normal vector of the plane. And H needs to minimize a weighted sum of squared projection distances of the points p_i to the plane, which can be described as :

$$\sum_{i=1}^{N} \left(\langle n, p_i \rangle - D\right)^2 \theta\left(\|p_i - q\|\right) \tag{2.2}$$

where $\|\|$ stands for L2 normalization, and θ is a smooth, positive, monotone decreasing function.

Based on the reference domain, a polynomial approximation g is computed to map the local point to the approximation surface. This polynomial approximation minimized the weighted least squares error:

$$\sum_{i=1}^{N} \left(g\left(x_i, y_i\right) - f_i\right)^2 \theta\left(\|p_i - q\|\right) \tag{2.3}$$

where (xi, yi) is the representation of the projection of p_i on H in a local coordinate system, and $f_i = n(p_i - q)$, which is the projection distance from p_i to H.

Notice that the function θ gives less weight to the point which is further from r in two steps, so these two steps can only consider local points, which saves computational resources when the number of points is big.

Points interpolation: After generating the MLS surface for approximation, points are inserted into the surface for increasing the density. The principle of inserting points is computing the Voronoi diagrams on the MLS surface and adding points at the vertices of the diagrams. However, generating the Voronoi diagrams on the whole surface is computationally excessive, and local approximations are used. In each iteration, one existing point is randomly selected, and a local MLS approximation surface is generated. The nearby points are projected to the surface and a Voronoi diagram on this surface is computed. The vertex which is the furthest to its nearest point is added as the new point. The process keeps iterating until the furthest distance from the vertices to the point is smaller than a threshold.

This method is based on the approximation surface of the point cloud, so it does not work well to point clouds that have sharp structures. And it cannot control the upsampling ratio.

2.2.1.2 Locally Optimal Projection operator

Lipman et al. (2007) introduced a Locally Optimal Projection (LOP) operator for point cloud surface approximation and applied it to point cloud upsampling. LOP projects an arbitrary number of points to a given point cloud to make it fit the distribution of that point cloud. Point cloud upsampling can be done with multiple iterations of this distribution. Given an initial point cloud P and a projected point cloud Q is projected such that it minimized the sum of the weighted distance to points in P, with respect to radial weights centered at Q. And points in Q should have a uniform distribution, and the point inside should not have a too short distance from each other. Q should follow the equation below:

$$Q = \text{argmin}_{X=\{x_i\}_{i\in I}} \{E_1(X, P, C) + E_2(X, C)\} \tag{2.4}$$

$$E_1(X, P, C) = \sum_{i\in I} \sum_{j\in J} \|x_i - p_j\| \, \theta\left(\|c_i - p_j\|\right) \tag{2.5}$$

$$E_2(X, C) = \sum_{i'\in I} \lambda_{i'} \sum_{i\in I\setminus\{i'\}} \eta\left(\|x_{i'} - c_i\|\right) \theta\left(\|c_{i'} - c_i\|\right) \tag{2.6}$$

where C is the original point cloud of Q before projection, θ, and η are two fast-decreasing smooth weight functions and lambda are balancing terms. The term E_1 leads the projection points moving toward the local distribution center, and the term E_2 limits the distance between the point of Q and keeps its distribution uniform.

LOP is a parameterization-free method and it does not rely on local normal estimating, local plane fitting, and other forms of local parametric representation. Therefore, it can deal with data and shape that have an ambiguous orientation or complex structures. However, LOP cannot produce a uniformly distributed point cloud on some occasions when the given point cloud is highly non-uniform, and Huang et al. (2009) proposed the weighted locally optimal projection (WLOP) to improve LOP using a weighting option for noise and outlier removal, which could produce more uniformly distributed data for better shape reconstruction.

Preiner et al. (2014) applied the WLOP operator to a continuous representation of a point set and proposed a novel surface reconstruction technique called Continuous LOP (CLOP). CLOP describes the point cloud's density in a geometry-preserving manner by a Gaussian mixture model, which makes it more compact for representation. CLOP runs several times faster than WLOP. And it also provides better sampling regularity with no constrain on the number of sampling points, which makes it more capable of upsampling.

2.2.1.3 Edge-Aware Resampling

As previous works all assumed that point cloud surfaces are smooth and thus do not perform well in processing sharp structures, Huang et al. (2013) proposed an edge-aware point cloud resampling method called Edge-Aware Resampling (EAR) which is able to process noisy and outlier-ridden point clouds in an edge-aware manner. EAR can be applied in upsampling, and is capable of producing point sets with normal that are free from noise and also preserve sharp features.

EAR is conducted by first resampling the points away from the edges and calculating normals, and progressively inserting points to approach the edge singularities. Specifically, EAR can be divided into two steps: Resampling away from edges and edge-preserving upsampling.

Resampling away from edges: In this step, given an unorganized and noisy point set a resampled point set associated with normals that can better represent the underlying smooth surface way from the edges is output. To create an initial input, the normals of points in the point cloud are estimated by traditional normal estimating methods like LOP. These normals are not accurate when the underlying surface is not smooth, so the following steps are needed. Taking the initial input, an iteration between separating and smoothing normals and resampling the points away from the edges is conducted.

In the separating and smoothing normals process, normals are estimated based on an anisotropic neighborhood. The normals are calculated based on the goal of minimizing the sum of differences between the assigned normals and other normals in their neighborhood. Specifically, given a point $s_i = (p_i, \mathbf{n}_i)$ from the point cloud where p_i stands for the coordinates and $\mathbf{n}_i$ stands for the normal, the normal difference between this point and other points in its neighborhood is measured by the equation below:

$$f\left(p_i, \mathbf{n}_i\right) = \sum_{s_{i'} \in \mathcal{N}_{s_i}} \left\|\mathbf{n}_i - \mathbf{n}_{i'}\right\|^2 \theta\left(\left\|p_i - p_{i'}\right\|\right) \psi\left(\mathbf{n}_i, \mathbf{n}_{i'}\right) \tag{2.7}$$

where $\|\|$ stands for the l2 normalization and $\mathcal{N}_{s_i} = \{s_{i'} \mid s_{i'} \in S \wedge \|p_i - p_{i'}\| < \sigma_p\}$ defines the neighborhood of s_i with the given neighborhood size σ_p .θ and ψ are two weighted functions, which follow the equations:

$$\theta(r) = e^{-r^2/\sigma_p^2} \tag{2.8}$$

$$\psi\left(\mathbf{n}_i, \mathbf{n}_{i'}\right) = e^{-\left(\frac{1-\mathbf{n}_i^\top \mathbf{n}_{i'}}{1-\cos(\sigma_n)}\right)^2} \tag{2.9}$$

where σ_n is the angle parameter that scales the similarity of normals in the neighborhood and is set to 15° by default. The goal of this process is to get an oriented point set $S = \{s_i\}_{i \in I} = \{(p_i, \mathbf{n}_i)\}_{i \in I} \subset \mathbb{R}^6$ with normal which minimized the sum of the normal difference between points and their neighborhood points, which can be described as the equation below:

$$S = \text{argmin}_{\{s_i\}_{i \in I} = \{(p_i, \mathbf{n}_i)\}_{i \in I}} \sum_{i \in I} f\left(p_i, \mathbf{n}_i\right) \tag{2.10}$$

To achieve this, the normal $\mathbf{n}_i$ for each point s_i is iteratively updated as:

$$\frac{\sum_{s_{i'} \in \mathcal{N}_{s_i}} \theta\left(\|p_i - p_{i'}\|\right) \psi\left(\mathbf{n}_i, \mathbf{n}_{i'}\right) \mathbf{n}_{i'}}{\sum_{s_{i'} \in \mathcal{N}_{s_i}} \theta\left(\|p_i - p_{i'}\|\right) \psi\left(\mathbf{n}_i, \mathbf{n}_{i'}\right)} \rightarrow \mathbf{n}_i \tag{2.11}$$

To remove the noise and outlier points, the resampling process is conducted to resample points away from the edges. In the resampling process, an altered anisotropic LOP operator is used. This LOP is edge-aware and added a normal-dependent weight function.

Edge-preserving upsampling: To fill in the gap along edges which appears in the last process, an edge-preserving upsampling is conducted to insert points in these gap areas. There are three requirements for the insertion operations: (1) the inserted point $s_k = (p_k, \mathbf{n}_k)$ should lie on the underlying surface, (2) the normal $\mathbf{n}_k$ should be vertical to the surface at p_k and (3) the distribution of the points in the local neighborhood should be uniform. EAR designs a novel projector that divides the insertion operation into three steps: (1) finding an insertion base location b_k in a spare area, (2) optimizing the projection distance d_k to the underlying surface and (3)determining the normal direction $\mathbf{n}_k$ and project points from the base location to the underlying surface. The final location p_k can be described as the following:

$$p_k = b_k + d_k \mathbf{n}_k \tag{2.12}$$

To make the final distribution uniform, the base location should be selected in the sparse area of the local neighborhood. Therefore, given an existing point s_i and its neighborhood $\mathcal{N}_{s_i}$ where the base would be located, the location of the base b should follow:

$$C(b) = \min_{s_{i'} \in \mathcal{N}_{s_i}} D\left(b, s_{i'}\right) \tag{2.13}$$

It has the advantage that the same location would not be secondly chosen as the base. To decrease computation, an approximation base location is calculated by limiting the candidate locations on the midpoint between s_i and its neighbor points in $\mathcal{N}_{s_i}$. The following

base should be placed in the low-density area or along sharp boundaries. So, a priority score is defined to decide which neighborhood the next point should be inserted to. For a given point s_i, its priority follows:

$$P\left(s_i\right) = \max_{s_{i'} \in \mathcal{N}_{s_i}} \left(2 - \mathbf{n}_i^\top \mathbf{n}_{i'}\right)^\rho C\left(\frac{p_i + p_{i'}}{2}\right) \tag{2.14}$$

where ρ is an edge-sensitivity parameter and is set to 5 by default.

The projection distance is determined by minimizing the sum of the weighted total projection distance between p and the other points in the neighborhood, which follows:

$$\sum_{s_i \in \mathcal{N}_{b_k}} \left(\mathbf{n}^\top \left(p - p_i\right)\right)^2 \theta\left(\|p - p_i\|\right) \psi\left(\mathbf{n}, \mathbf{n}_i\right) \tag{2.15}$$

Therefore, the distance d_k is obtained as:

$$d_k\left(b_k, \mathbf{n}\right) = \frac{\sum_{s_i \in \mathcal{N}_{b_k}} \left(\mathbf{n}^\top \left(b_k - p_i\right)\right) \theta\left(\|b_k - p_i\|\right) \psi\left(\mathbf{n}, \mathbf{n}_i\right)}{\sum_{s_i \in \mathcal{N}_{b_k}} \theta\left(\|b_k - p_i\|\right) \psi\left(\mathbf{n}, \mathbf{n}_i\right)} \tag{2.16}$$

Normal is determined by two criteria: the projection distance should be small, and the normal of the inserted point should fit the normal distribution of the local neighborhood of the base point. Therefore, it is first decided whose neighborhood of the two neighbors of the base point to use based on the projection distance:

$$l = \underset{l \in \{i,j\}}{\operatorname{argmin}} d_k\left(b_k, \mathbf{n}_l\right) \tag{2.17}$$

Then, $\mathbf{n}_k$ is computed by minimizing $f(b_k, \mathbf{n})$, and fixing $\mathbf{n} = \mathbf{n}_l$ when calculating the directional weight $\psi(\mathbf{n}, .)$.

As b_k, d_k, and $\mathbf{n}_k$ are calculated, the inserting point can be determined by Eq.(2.12).

2.2.1.4 Graph Total Variation Based Method

Leveraging on the progress in graph signal processing, Dinesh et al. (2019) proposed a local 3D point cloud upsampling algorithm via Graph Total Variation (GTV). This method achieves upsampling by inserting points in the origin point cloud and then adjusting the points' position to promote piecewise smooth surfaces, which means the surface normals between neighboring points differ minimally over the 2D surface except at the boundaries.

The input of this method should be a low-density point cloud that the normal is estimated in prior. First, this method constructs a triangular mesh of the origin point cloud via Delaunay triangulation and initializes the new points at the centroids of the local triangles. To make the underlying surface piecewise smooth, the inserted points should be fine-tuned, and the GTV is defined for judging. A K-Nearest-Neighborhood (KNN) graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is constructed to connect points in the point cloud. The nodes in V are points in the point cloud, and the edges in $\mathcal{E}$ describe the connection among points. And a weight function $w_{i,j} \in \mathbb{R}^+$ is designed to describe the similarity between nodes, which follows:

$$w_{i,j} = \exp\left\{-\frac{\|\mathbf{p}_i - \mathbf{p}_j\|_2^2}{\sigma_p^2}\right\} \cos^2\theta_{i,j} \tag{2.18}$$

where p_i and p_j are the positions of points i and j, $\theta_{i,j}$ are the angle between the surface normal of i and j, and σ_p is a parameter. A higher value of $w_{i,j}$ means that the position of points i and j are close, and their normal are similar. Therefore, GTV is defined as follows:

$$\|\mathbf{n}\|_{\mathrm{GTV}} = \sum_{i,j\in\mathcal{E}} w_{i,j} \|\mathbf{n}_i - \mathbf{n}_j\|_1 \tag{2.19}$$

And the point-adjusting function can be formulated to the minimization of GTV value.

As the normal of points cannot be described as a linear function of its coordinate and its neighbors, the point cloud is separated into two disjoint parts (say green and yellow), and when adjusting the point in the yellow part, only neighboring points in the green part are employed as reference. Therefore, the normal $\mathbf{n}_i$ for a point i in the yellow part can be written as:

$$\mathbf{n}_i = \mathbf{A}_i\mathbf{p}_i + \mathbf{b}_i \tag{2.20}$$

where $\mathbf{A}_i \in \mathbb{R}^{3\times3}$ and $\mathbf{b}_i \in \mathbb{R}^3$ can be computed by the neighbors of i in the green part. For each part, a new KNN graph $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1)$ is constructed that only connects neighboring points in the opposite part. Therefore, based on Eq.(2.20), the difference between normal in two connected points in $\mathcal{G}_\infty$, donate by $\mathbf{m}$, can be defined as:

$$\mathbf{m} = \mathbf{B}\mathbf{p} + \mathbf{v} \tag{2.21}$$

And the object of minimizing GTV can be written as follows:

$$S = \mathrm{argmin}_{\{s_i=(p_i,\mathbf{n}_i)\}\in S} \sum_{i,j} w_{i,j} \|\mathbf{m}_{i,j}\|_1 \tag{2.22}$$

While subject to linear constraints in Eq.(2.21). Therefore, the optimization framework can start with solving Eq.(2.22) for the yellow part while fixing the green parts. Then, construct another k-NN for the green part and solve Eq.(2.22) with the newly solved yellow part. This alternate optimization is continuously conducted between these two parts until convergence.

In general, non-deep learning-based methods can achieve the upsampling task to some extent. However, they have many limitations. Some of these methods rely on priors, such as normal estimation. And some also have an assumption that the surface is smooth, which is not a common case in practice. And most of these methods are not data-driven and cannot precisely control the density after upsampling.

2.2.2 Deep learning-based methods

As point cloud is a format that does not have specific spatial order and regular grid structure, the early-stage research converted point cloud into other representation structures including 3D volumetric grids and geometric graphs, which increase unnecessary calculation and redundancy. In recent years, however, some networks have been proposed to directly process point clouds, including PointNet (Qi et al., 2017a), PointNet++ (Qi et al., 2017b), and DGCNN (Phan et al., 2018). Therefore, point cloud upsampling with deep learning has become a popular topic and many deep learning-based point cloud upsampling methods have been proposed.

Most of the deep leaning-based point cloud upsampling pipelines consist of three components: the feature extraction component, the upsampling component, and the point set generation component, which is shown in Figure 2.2 below.

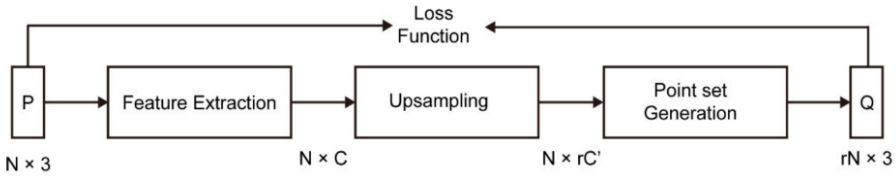


Figure 2.2 General structure of deep learning-based point cloud upsampling methods (Source: Zhang et al., 2022)

As the first component after an $N \times 3$ point cloud is input, the feature extraction component would extract an $N \times C$ point cloud feature from the input. Here N is the

number of points in the point cloud and C is the dimension of the feature. Then, the upsampling component would expand the point feature to $N \times rC'$, and r is the upsampling ratio. Finally, the point set generation component would reconstruct the point feature to point clouds with 3D coordinates. The density would increase to r times of the origin. Among all these three components, the upsampling component is the one that has the largest influence on the upsampling performance.

Before the main structure of most of the deeplearning-basedd point cloud upsampling methods, patch extraction is conducted. The point clouds generated from the complete object are different in size and point number, which can also be too large. So, Patch extraction aims at avoiding a too-large input size for computation saving and keeping the size of input patches consistent. The deep learning-based point cloud upsampling methods have similar patch extraction processes, and some of them are shown in Figure 2.3 below.

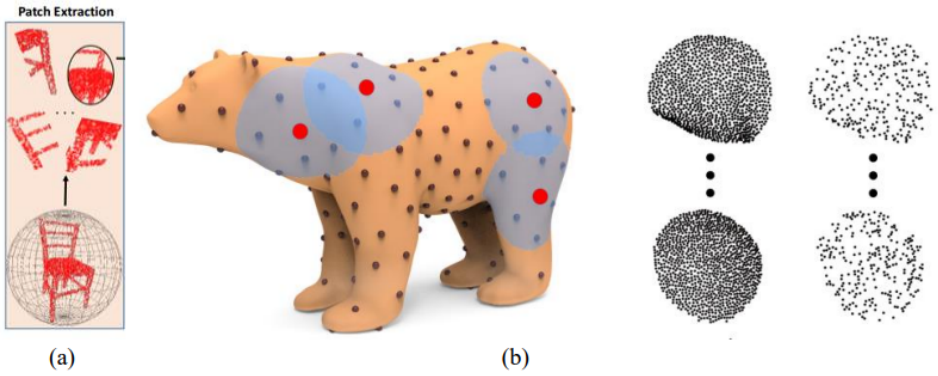


Figure 2.3 Patch extraction of (a) Pu-Net and (b) PU-GAN.
(Source: Yu et al., 2018, Li et al., 2019b)

On the surface of the points cloud of these objects, a certain number of points are randomly selected. For each of the selected points, the nearby points within a certain geodesic distance d are selected to grow surface patches. Then these patches are sampled with Poisson disk sampling and only a certain number of points are reserved to generate the final patch to be input to the upsampling networks. To preserve features of different sizes and densities, the geodesic distance d is set with varying sizes.

2.2.2.1 PU-Net

Yu et al. (2018) proposed represented Point Cloud Upsampling Network (PU-Net) to generate a denser and uniform point cloud from the low-density input. The architecture of PU-Net is shown in Figure 2.4 below.

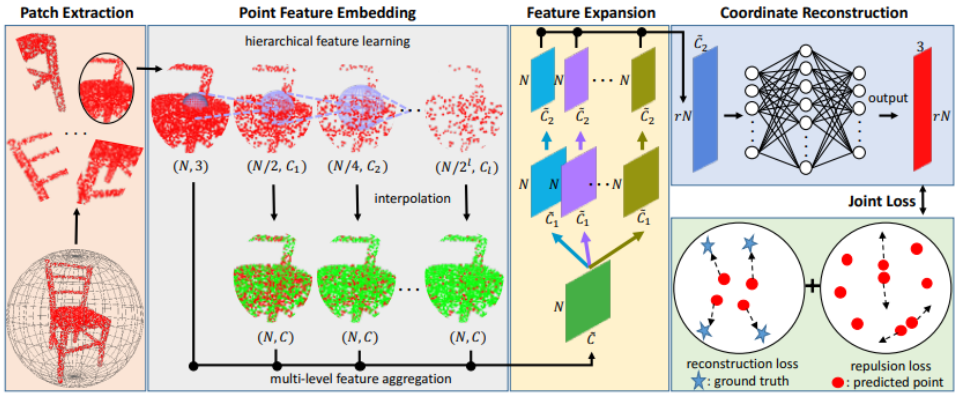


Figure 2.4 Architecture of PU-Net
(Source: Yu et al., 2018)

Like most of the deep learning-based point cloud upsampling methods, the architecture of PU-Net consists of three components: the feature extraction component, the upsampling component, and the point set generation component. In its feature extraction component, PU-Net adopts the hierarchical feature learning mechanism and Multi-level feature aggregation mechanism proposed by PointNet++. To extract both local and global features, PU-Net used hierarchical feature learning to capture features under different scales. Then, these features are concatenated together as embedded point features by Multi-level feature aggregation. In its Upsampling component, PU-Net duplicates the embedded feature r times before they are put into different convolutional layers with the same sizes. Then the results are concatenated together and reshaped. Finally, in its Point set Generation component, features are reconstructed to 3D coordinates through a series of fully connected layers.

PU-Net is the first deep learning-based point cloud upsampling model, and it overcomes many limitations of the non-deep learning-based methods including the reliance on prior and the assumption of the smooth surface. However, it also has some limitations as it cannot fill the holes and it would miss part or meaningful details when processing tiny

structures. And its upsamling component structure would make the generated point cloud clustered around the original point position.

2.2.2.2 3PU

As PU-Net fixes the level of detail included in the input patches, it ignored both high-level and low-level geometric structures. To solve this limitation, Wang et al. (2019) proposed a patch-based progressive 3D point cloud upsampling network called 3PU. 3PU proposed a multi-step upsampling method, which separates the whole upsampling process into a series of sub-steps. The architecture of 3PU, which is shown in Figure 2.5 below, consists of several Upsampling network units and each expands the feature size two times. According to the detailed structure shown in Figure 2.6, each of the upsampling network units contains a feature extraction component and an upsampling component. These units with the same structure are employed to different levels of detail.

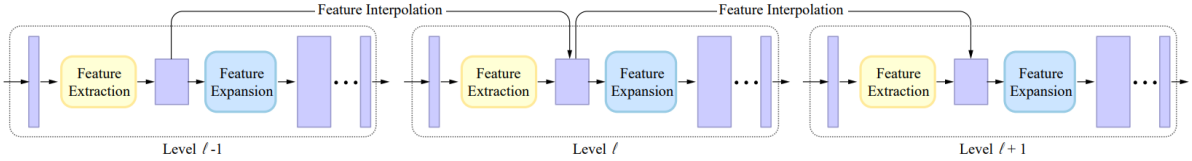


Figure 2.5 Architecture of 3PU
(Source: Wang et al., 2019)

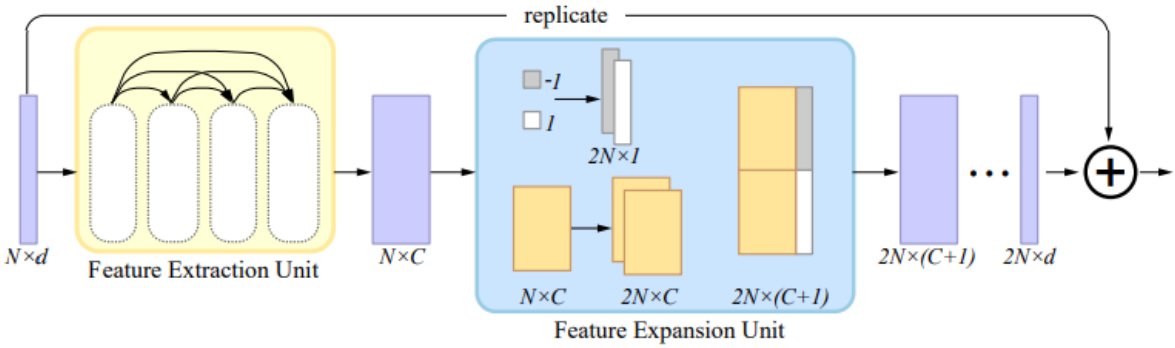


Figure 2.6 Architecture of the upsampling network unit
(Source: Wang et al., 2019)

The feature extraction component would extract an N*C feature from an $N \times d$ (d is the dimension) point set which is the origin input point cloud or the output of the previous upsampling network unit. As shown in Figure 2.7, each feature extraction component also contains several dense blocks. Each dense block would first compress the input, which could be a point set or the output feature from the last dense block to a fixed number of features. Then features are grouped by feature-based KNN, passed through a series of densely connected Multi-Layer Perceptions (MLP), and finally max-pooled to get order-invariant. The dense connection means concatenating feature output from two different layers together to form a new feature. The dense connection reuses the information, which increases the reconstruction accuracy as well as reduces the number of parameters in the model. 3PU also introduces the dense connection between dense blocks and features produced by dense blocks are fed into the following blocks.

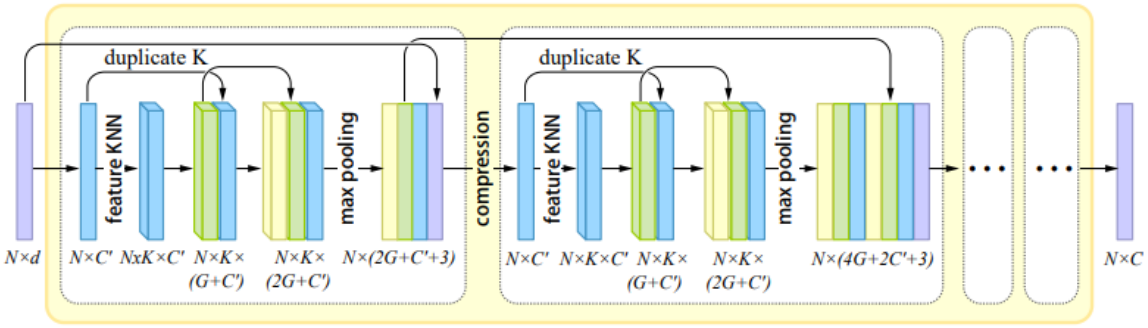


Figure 2.7 Architecture of the feature extraction unit
(Source: Wang et al., 2019)

The Upsamling component, which is specifically called the feature expansion unit in 3PU, would expand the feature with the size $N \times C$ to coordinate with size 2N*d. In the feature expansion unit, 3PU added position variations to features. Features are duplicated 2 times, and each of them is assigned a 1D code with the value -1 or 1 before they were concatenated together to form a feature with the size $2N \times (C + 1)$. And a set of MLPs compress the feature to the coordinate size $2N \times d$ as a simple point set generation component.

To enhance the communication between units, 3PU also introduces inter-level skip-connections. Features extracted from the previous level are passed to the current level and interpolated to the feature generated there.

3PU superiors the previous methods in point cloud upsampling. However, its multi-step structure makes it computationally expensive.

2.2.2.3 PU-GAN

Li et al. (2019b) proposed a point cloud upsampling adversarial network called PU-GAN. Like other Generative Adversarial Networks (GAN), the structure of PU-GAN contains a generator and a discriminator. The generator tries to generate 'fake' data, which is the high-density point clouds in this case, and the discriminator tries to tell the real data to force the generator to create data closer to the real ones.

As is shown in Figure 2.8, the structure of the PU-GAN generator also contains those three components: the feature extraction component, the upsampling component, and the point set generation component. PU-GAN adopted the feature extraction component from 3PU and its dense connection mechanism, where an $N \times d$ point cloud is input and an $N \times C$ point-wise feature is output.

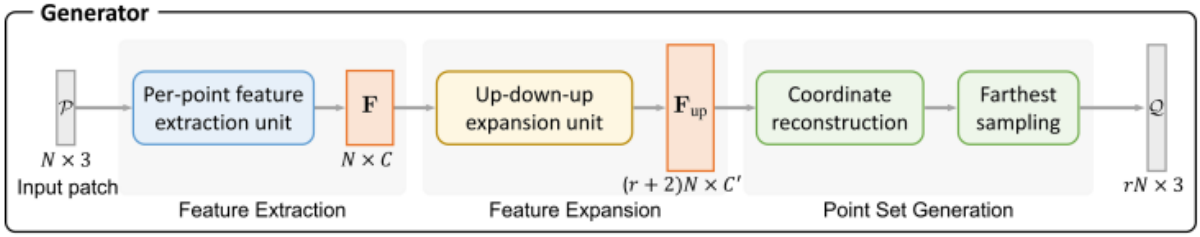


Figure 2.8 Architecture of the generator of PU-GAN
(Source: Li et al., 2019b)

In the Upsampling component, PU-GAN proposed an Up-down-up feature expansion unit Figure 2.9. The input feature F is first processed by MLPs to form feature F_1. Then F1 is upsampled by an up-feature operator to form $F_{up}\prime$, then downsampled by a down-feature operator to create F_2. Then the difference between F_1 and F_2, donate as F_Δ, is calculated. Then F_Δ is upsampled again by another up-feature operator, and $F_{up}\prime$ is added to it for self-correction to create the final F_{up}.

In the up-feature operator, the input feature with the size $N \times C$ is first duplicated r times. And the 2D grid mechanism in FoldingNet(Yang et al., 2018) is used to generate a unique 2D vector from each feature copy. These vectors are appended to the corresponding copy to create a feature with size $N \times (rC + 2)$. Then it is passed through a self-attention unit and a set of MLPs to produce the output upsampled features with the size $rN \times C$.

The self-attention unit is proposed for feature integration after concatenation. Three matrices G, H, and K are extracted from the input feature by three separated MLPs.

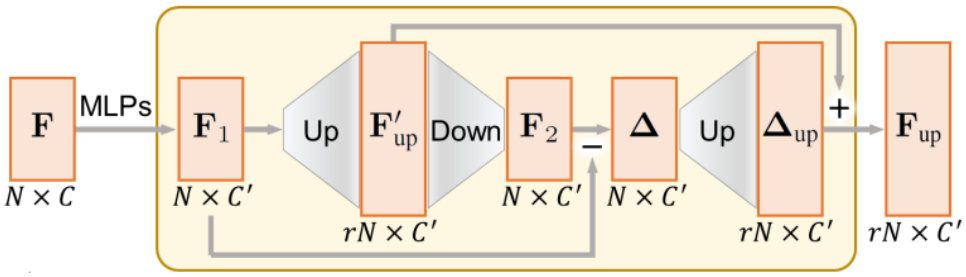


Figure 2.9 Architecture of up-down-up feature expasion unit
(Source: Li et al., 2019b)

Then the attention weight W is calculated by the equation $\mathbf{W} = f_{\text{softmax}}\left(\mathbf{G}^T\mathbf{H}\right)$ means the softmax function. Then, the feature weight is calculated by $\mathbf{W}^T\mathbf{K}$, and the input feature is added to it to create the output feature.

In the down-feature operator, the $rN \times C$ input is first reshaped to an $N \times rC$ format and then compressed by MLPs to create the origin shape $N \times C$.

In the point set generation component, a set of MLPs is used to regress 3D coordinates from F_{up}. As the points in the latent space are still close to the input, the farthest sampling is adopted to keep the generated point not too close to each other. Therefore, the value of r when creating F_{up} is a certain value (specifically, 2) larger than the point cloud upsampling ratio.

The discriminator would generate a confidential value about whether its input is a real point cloud or it is created by the generator. The architecture of the discriminator is shown in Figure 2.10, which utilizes the basic network architecture in Point Completion Network (PCN) (Yuan et al., 2018) to extract global features. Then, a self-attention unit is adopted after the features are concatenated to enhance feature integration and improve extraction capability. Then, features are input into a set of MLPs, a max pooling layer, and a set of Fully Connected layers to generate a confidence value. If the confidence value is close to 1, the discriminator predicts with much confidence that the input should come from the real world and otherwise be generated by the generator.

PU-GAN makes great progress in a more uniform distribution of output point clouds, as well as the reduction of tiny holes and detail structures. However, PU-GAN is specially designed for filling tiny holes, so it does have a good performance to fill in large gaps and holes in point clouds.

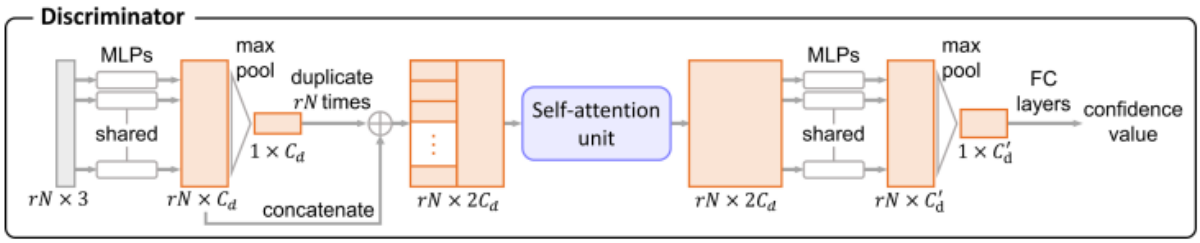


Figure 2.10 Architecture of the discriminator of PU-GAN.
(Source: Li et al., 2019b)

2.2.2.4 PU-GCN

Qian et al. (2021) proposed a novel Graph Convolutional Network (GCN) based point cloud upsampling model called PU-GCN. As the author believes that the Upsampling component of the previous methods either operates points only based on themselves and ignores the relationship to their neighborhood or generates patches too similar to the input, PU-GCN adopts GCN structure into point cloud upsampling to aggregate the neighborhood information among points. Like most of the other deep learning-based point cloud upsampling networks, PU-GCN the architecture of PU-GCN, which is shown in Figure 2.11, also has those three components: an interception feature extractor for feature extraction, an upsampler for upsampling, and a coordinate reconstructor for point cloud generation.

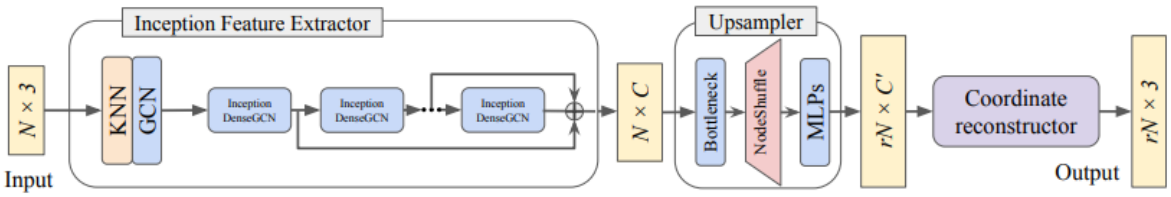


Figure 2.11 Architecture of PU-GCN
(Source: Qian et al., 2021)

In the feature extraction component, PU-GCN uses an inception feature extractor. A KNN layer is used at the beginning of this component to build the neighborhood graph of the input points. In this graph, the nodes represent the points, and the edges represent the connections between each point and its K nearest neighbors. This neighborhood graph is needed for the following GCN processing. Then a GCN layer is used to extract features in latent space. Then the features are passed through several (two by default) Inception

DenseGCN blocks. The outputs are combined with residual connections before they are input into the upsampler component. The default GCN layer is EdgeConv proposed in DGCNN.

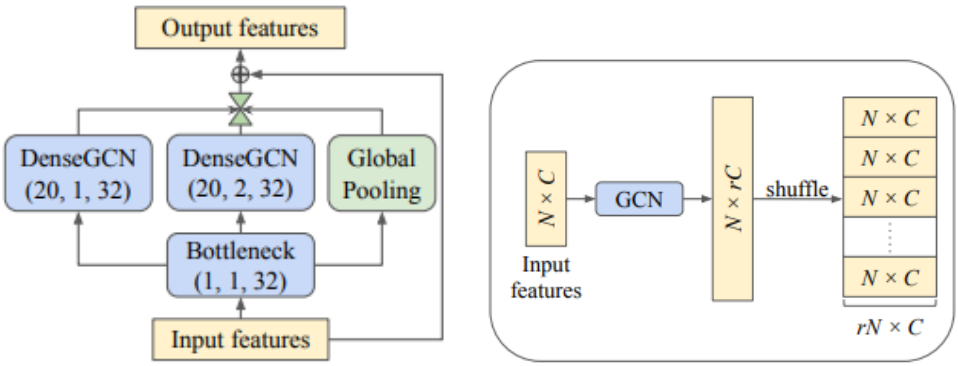
The Inception DenseGCN block is a new multi-scale point feature extractor proposed by PU-GCN. A single-layer MLP, which is named a bottleneck layer by PU-GCN, is adopted for feature compression to reduce the calculation. Then the features are input into two parallel DenseGCN blocks. As the architecture shown in Figure 2.12a, each of the DenseGCN blocks consists of three dilated graph convolutions (Li et al., 2019a) layers, which are densely connected. These DenseGCN blocks have different dilation rates, which ensure the ability to gain different respective fields without increasing the number of nodes in the neighborhood. The DenseGCN blocks also share the same graph structure to reduce calculation. Then global contextual information is extracted from the compressed feature by a global pooling layer. These layers have different receptive fields, and therefore multi-scale information is extracted. The output of these layers and the input features are concatenated together to form the final output.

In the Upsampling component, PU-GCN proposed NodeShuffle, and its architecture is shown in Figure 2.12b. A bottleneck layer first compressed the input size to $N \times C$ to reduce the calculation. Then feature is expanded to a wider size of $rN \times C$ by the NodeShuffle. In the NodeShuffle, a GCN layer is adopted to expand the input feature to shape $N \times rC$ and rearranged the shape of the feature of $rN \times C$. PU-GCN is the first Method that introduces Graph Convolutions into Point Cloud Upsampling. The GCN structure enables NodeShuffle to encode local neighborhood spatial information and learn new points from latent space.

In the coordinate reconstruction component, PU-GCN adopted two sets of MLPs to reconstruct point feature to 3D coordinates. The output is the desired denser point cloud with the size of $rN \times 3$.

PU-GCN is able to generate point clouds with intricate structures and details. PU-GCN outperformed other state-of-the-art Point Cloud upsampling methods in many public datasets. It can generate point clouds with fewer outliers. Furthermore, it requires fewer parameters than other methods.

Although many of the deep learning-based point cloud upsampling methods have been proposed, there are few pieces of research utilizing them on other real-world point cloud applications such as object detection and semantic segmentation. Furthermore, although many of the more recent methods claimed that they have some improvement to the previous methods, the actual influence of them on the practical application is not clarified.



(a) (b)

Figure 2.12 Architecture of (a) Inception DenseGCN and (b) nodeshuffle (Source: Qian et al., 2021)

2.3 Benchmark

2.3.1 Datasets

Many of the datasets have been employed in the evaluation of point cloud upsampling methods and the training of deep learning-based methods, and some of them are public. Many of the research teams of deep learning-based point cloud upsampling methods also construct their own dataset as a supplement to the public dataset. These datasets cover a wide range of object types, and they are also different in density, quality, generating methods, and sample number. The basic information of some of the public datasets are listed in Table 2.1.

2.3.1.1 Synthetic Datasets

ModelNet (Wu et al., 2015)(which have two visions, ModelNet10, and Model40), ShapeNet (Chang et al., 2015), SHREC15 (Pickup et al., 2015), and PU1K (Qian et al., 2021) are some of the typical synthetic datasets adopted by point cloud upsampling methods. Some of the upsampling methods select and combine some subsets of these datasets for training and testing. Most of these datasets contain 3D Computer Aided Design (CAD) models of objects that belong to some of the most common categories in the real world, such as animals, airplanes, toilets, chairs, and beds, to name a few. These CAD models are

Table 2.1 Basic information of the public datasets adopted by upsampling research

Name	Sample	Training	Testing	Type
ModelNet10(Wu et al., 2015)	4899	3991	605	Synthetic
ModelNet40(Wu et al., 2015)	12311	9843	2468	Synthetic
ShapeNet(Chang et al., 2015)	51190	-	-	Synthetic
PU1K(Qian et al., 2021)	1147	1020	127	Synthetic
SHREC15(Pickup et al., 2015)	1200	-	-	Synthetic
FASUST(Bogo et al., 2014)	300	100	200	Real-Scan
ScanObjectNN(Uy et al., 2019)	2902	2321	581	Real-Scan
KITTI(Geiger et al., 2013)	14999	7481	7518	Real-Scan

collected from many public resources, including Princeton Shape Benchmark (Shilane et al., 2004). These CAD models are converted to point clouds by depth images and other methods such as furthest point sampling and Principal component analysis. As these methods are synthetic, the objects within are well-classified, complete, and do not have noise (Uy et al., 2019)., which are not similar to real-world situation.

2.3.1.2 Human Pose Real-Scan Datasets

The synthetic dataset does not have noise and missing data, which made them unrealistic. Therefore, Bogo et al. (2014) construct a real scanned high-resolution dataset called Fine Alignment Using Scan Texture (FAUST). FAUST contains scans of 10 different people making a variety of poses. Bogo et al., use a full-body 3D stereo capture system for data capturing. It consists of 22 scanning units and each of them is composed of a single 5MP RGB camera, several speckle projectors, and a pair of stereo cameras. The appearance of FAUST is a supplement to the lack of public real-scanned point cloud datasets, and it is also complementary to synthetic datasets. However, as the dataset is derived from human poses, it does not have wide usage and is not very representative. And it cannot be used in the comparison with synthetic datasets as the type difference.

2.3.1.3 Indoor Object Real-Scan Datasets

ScanObjectNN (Uy et al., 2019) is another real-scan dataset aiming at providing more realistic point cloud information than those complete, well-segmented, and noiseless synthetic datasets. ScanObjectNN is constructed by extracting objects from the annotated scenes

within two real-world scene mesh datasets: SceneNN (Hua et al., 2016) and ScanNet (Dai et al., 2017). The presence of background noise, non-uniform density, and holes due to incomplete scans makes it different from the CAD-based synthetic datasets. The objects in ScanObjectNN belong to 15 categories, which are common indoor objects in the real world, such as desks, chairs, and sofas. Therefore, ScanObjectNN is a representative dataset and can be used in the comparison with synthetic datasets with similar object categories. But one of its limitations is that it does not provide the information in a whole scene.

2.3.1.4 Large-Scene Real-Scan Datasets

KITTI (Geiger et al., 2013) (Karlsruhe Institute of Technology and Toyota Technological Institute) is one of the most popular public datasets for research in autonomous driving, robotics, and computer vision, and is also employed in point cloud upsampling[PU-GAN]. It is a calibrated, synchronized, and rectified dataset on traffic scenarios, which contains point cloud information. The dataset is captured by a system with different sensor modalities, including two grayscale cameras, two color cameras, four optics lenses for 2D image capturing, one GPS navigation system for location recording, and one 3D laser scanner for point cloud capturing. The traffic scenarios represented by KITTI are useful in a wide range of applications, which ensure its versatilities and representativeness. However, as it utilized GPS in data capturing, it cannot represent the GNSS-denied scenarios, such as some indoor and underground places.

2.3.2 Evaluation Metrics

To evaluate the performance of a point cloud upsampling method and train a deep learning-based model with gradient descent, a ground truth high-density point cloud is necessary for each input low-density point cloud to make a comparison with the generated point cloud. And due to the adoption of patch extraction, each patch should have a ground truth patch with a density a fixed number of times larger than it. However, for each of the objects in public datasets, there isn't a corresponding high-density ground truth. And this cannot be achieved by rescanning the same object with devices that can capture denser point clouds, as it is difficult to find corresponding patches. Therefore, during experiments, the researchers regard the point clouds in the current datasets as ground truth, and each extracted patch is downsampled by Poisson disk sampling, random downsampling, and some of the other downsampling methods to get a low-density patch. Then the ground truth is compared with the generated patch from the low-density one for evaluation or training.

The performance of the point cloud upsampling method is evaluated by the generated point cloud. There are two requirements for the generated point clouds: they should keep the original shape and feature, and the points within should not cluster together. Therefore, there are some metrics that evaluate the performance of the upsampling methods in the corresponding two aspects: the similarity between the generated point cloud and the ground truth, and the uniformity of the generated point cloud.

2.3.2.1 Similarity Evaluation

Many metrics for measuring the similarity of sets and distributions are adopted by point cloud upsampling methods to measure the similarity of point clouds. Some typical metrics are introduced below:

Chamfer Distance (CD): For two point cloud, Chamfer Distance would calculate the sum of the mean value of the minimum square distance from all points in each point cloud to the other. It follows the equation below:

$$d_{CD}(X,Y) = \frac{1}{|X|}\sum_{x\in X}\min_{y\in Y}||x-y||_2^2 + \frac{1}{|Y|}\sum_{y\in Y}\min_{x\in X}||x-y||_2^2 \tag{2.23}$$

where X and Y are two point clouds, $|X|$ and$|Y|$ are the number of points within X and Y, and x and y are points within them. Eq. (2.23) represents the sum of the average distance from any of the points x in X to its closest point in Y, and the average distance from any of the points y in Y to its closest point in X.

Hausdorff Distance (HD): Hausdorff Distance measures the distance of two sets in metrics space. When measuring the distance between point clouds, HD can be regarded as the maximum value of the distance from a point in one point cloud to its closest neighbor point in another. It follows the equation below:

$$d_{HD}(X,Y) = \max\left\{\sup_{x\in X} d(x,Y), \sup_{y\in Y} d(y,X), \right\} \tag{2.24}$$

$$d(a,B) = \inf_{b\in B}||a-b||_2 \tag{2.25}$$

where X and Y are two point clouds, sup means supremum and inf means infimum. Eq. (2.24) calculates the minimum distance from a point in one point cloud to all the points in another. And Eq. (2.25) calculates the maximum one of this minimum distance.

Earth Mover's Distance (EMD): Earth Mover's Distance (EMD) is a metric that measures the similarity in transportation processes. It measures the minimum cost of converting one distribution into another. If it only takes a small cost to change one distribution into another, it means these two distributions are similar. When measuring the similarity of two point clouds, EMD calculates the minimum of the sum distance of moving points in one point cloud to change it to another point cloud. It follows the equation below:

$$d_{EMD}(S_1, S_2) = \min_{\phi: S_1 \to S_2} \sum_{x \in S_1} ||x - \phi(x)||_2 \tag{2.26}$$

where $S1$ and $S2$ are two point clouds, and $\phi : S_1 \to S_2$ is a bijection mapping.

2.3.2.2 Uniformity Evaluation

Another important evaluation indicator for generated point clouds is the uniformity of the points distribution. The distribution of the points in generated point cloud should be uniform. There are also some metrics measuring the uniformity of point clouds.

Normalized Uniformity Coefficient (NUC): Normalized Uniformity Coefficient is first proposed in PU-Net to measure the uniformity of the point cloud distribution. To calculate NUC, several equal size disks are randomly put on the surfaces of objects of the generated point cloud datasets. And NUC is the standard deviation of the number of points in disks. It follows the equation below:

$$avg = \frac{1}{K \times D} \sum_{K=1}^{K} \sum_{D=1}^{D} \frac{n_i^k}{n_k \times p} \tag{2.27}$$

$$NUC = \frac{1}{K \times D} \sum_{K=1}^{K} \sum_{D=1}^{D} (\frac{n_i^k}{n_k \times p} - avg)^2 \tag{2.28}$$

where n_i^k is the number of the points in the i-th disk of the k-th object, n_k is the total points number within the k_th object and p represents the percentage of the disk areas over the sum of the surface area.

Uniformity Metric in PU-GAN: As NUC does not consider the point cloud distribution in disks, it neglects the local points clutter within disks. Therefore, PU-GAN proposes another uniformity measuring metric to improve this limitation. This metric measures the uniformity of the point distribution in patches after patch extraction, so the

uniformity of objects and datasets can be measured by the average number of all the patch uniformity values within.

For the j-th patch S_j, M seed points are randomly selected on the patch surface by the farthest sampling. From each seed point, a disk is grown by a ball query with radius r_d. Following the chi-squared model, the distribution of points among disks is defined as:

$$U_{\text{imbalance}}\ (S_j) = \frac{(|S_j| - \hat{n})^2}{\hat{n}} \tag{2.29}$$

$$\hat{n} = rNp \tag{2.30}$$

$$p = \frac{\pi r_d^2}{\pi 1^2} = r_d^2 \tag{2.31}$$

where $\hat{n}$ is the expected point number within a disk, r is the upsampling ratio and rN is the points number in the upsampled patch. P is the proportion of the disk area to the whole patch area. What needs to be noticed is that PU-GAN would normalize patches into a unit sphere in patch extraction. So, p is calculated in Eq. (2.31).

To measure the distribution within disks, the distance of each point to its closest neighbor is measured and donated as $d_{j,k}$, where j is the index of disks and k is the index of point. In an ideally uniformly distributed patch, the distance between points and their closest neighbor $\hat{d}$, is the same, and its value can be calculated using Eq. (2.33). So, the deviation of $d_{j,k}$ from $\hat{d}$ can be measured following chi-squared model as:

$$U_{\text{clutter}}\ (S_j) = \sum_{k=1}^{|S_j|} \frac{\left(d_{j,k} - \hat{d}\right)^2}{\hat{d}} \tag{2.32}$$

$$\hat{d} = \sqrt{\frac{2\pi r_d^2}{|S_j|\sqrt{3}}} \tag{2.33}$$

As U_{clutter} considers local uniformity, and $U_{\text{imbalance}}$ considers nonlocal uniformity, the final uniformity metric combines them together to get:

$$\mathcal{L}_{\text{uni}} = \sum_{j=1}^{M} U_{\text{imbalance}}\ (S_j) \cdot U_{\text{clutter}}\ (S_j) \tag{2.34}$$

2.4 Object Detection Method

Object detection is one of the important applications of the LiDAR point cloud. Point cloud object detection can recognize and localize objects of certain categories from 3D point clouds. In recent years, numerous state-of-the-art point cloud object detection methods have been proposed by researchers. Most of the state-of-the-art point cloud object detection methods are deep learning-based, and they can also be divided into the Grid and Voxel-based method, and the point-based method.

2.4.1 Grid and Voxel-based method

Unlike 2D images which have regular grid structures, the irregular and unstructured characteristics of the 3D point cloud make the point cloud object detection challenging. Some early-stage researchers try to build regular structures for point clouds by converting points into regular 3D voxels or projecting 3D point clouds on a 2D surface, which can be directly applied by convolutional networks.

A group of researchers (Beltrán et al., 2018) represents a point cloud by its bird's-eye view (BEV) and utilize 2D convolutional networks for feature learning and bounding box generation. And some of these methods also utilize multi-sensor and also utilize the information in the 2D image of the corresponding point cloud (Ku et al., 2018; Liang et al., 2018). These methods are often adopted in outdoor scenarios such as car detection in autonomous driving where there aren't many obstacles in the horizontal direction. However, these methods ignore information from other directions. Therefore, these methods do not have satisfactory performance in indoor environments where the objects to be detected are blocked by other objects in the bird's-eye view. And the need for 2D images also increases the cost. There are also other methods that project point clouds on the front view(Zhou et al., 2019), and some methods combine the bird's-eye view, front view, and 2D image together(Chen et al., 2017). However, they all have similar limitations.

Zhou and Tuzel (2018) proposed an End-to-End Voxel-based point cloud object detection network called VoxelNet. VoxelNet conduct a series of Voxel-based process to convert the irregular point cloud into a regular structure. The 3D space is subdivided into equal size voxels and points in the input point cloud are grouped into the corresponding voxel. Then, random sampling is conducted to reduce computational costs and the imbalance of points number among voxels. These processes use voxels to build a regular spatial structure for the input point cloud, which can be later processed by 3D convolutional networks.

Compare to the methods projecting point cloud to 2D surfaces, VoxelNet better utilizes features in all direction and have fewer scenario limitation. However, its voxel-based structure cost large memories and computation resources. And information is lost in the building of the voxel structure. Yan et al. (2018) proposed Sparsely Embedded Convolutional Detection (SECOND) and tries to improve the computation efficiency by sparse convolution operation, but it also has the second limitation.

2.4.2 Point-based method

Attribute to PointNet(Qi et al., 2017a) and PointNet++(Qi et al., 2017b), an increasing number of point cloud object detection methods have been proposed to directly process the input point cloud and extract features from it. The common way of point-based object detection method is first assigning a group of points to the candidates of each object, and then computing features from these point groups(Liu et al., 2021).

2.4.2.1 VoteNet

Ding et al. (2019) proposed a point-based object detection method called VoteNet. Inspired by the Hough Voting strategies, VoteNet locates the centroids of objects by point voting and aggregate votes to generate object proposals with high quality. The architecture of VoteNet is shown in Figure 2.13. The whole architecture can be divided into two parts: feature learning and point cloud voting, and object proposal generation and classification from votes.

Learning the features, including geometric contexts and spatial relationships within the input point cloud is essential in the subsequent vote generation process. VoteNet leverages the PointNet++ backbone for the feature-learning. The PointNet++ backbone is significant, not only it ensures the ability to directly process the irregular input point cloud, but it can also extract features at different levels from the input point cloud because of its multi-scale and skip connection structure. The feature extraction structure would output a point set subsampled from the input point with not only their coordinates but also an enriched feature vector.

Hough Transformation is a technique for detecting object instances within a certain class of shapes from a scene, even with noise (Milletari, 2018). The traditional Hough Transform method finds the centroid of the object to represent the object's position by voting through key points within the object. Key points would vote for candidate positions based on their own feature. The process of mapping the voted position and point feature

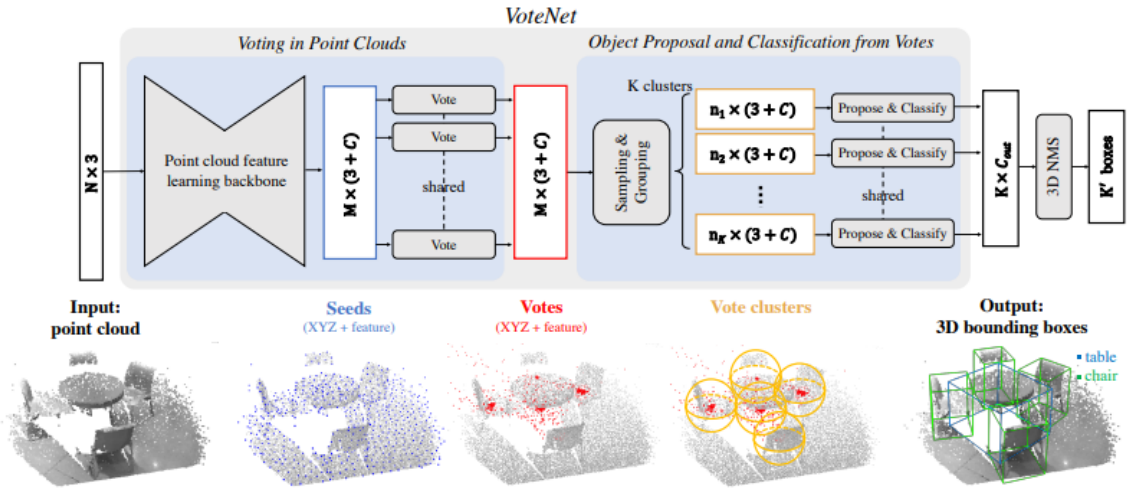


Figure 2.13 Architecture of VoteNet
(Source: Ding et al., 2019)

is achieved based on indexing the pre-build code book, which costs high computational resources. In VoteNet these processes are completed by the voting module, which consists of a fully connected MLP network, with ReLU and batch normalization. Taking the seed feature, the voting module would output votes with renewed positions and features. Unlike the seeds, the votes are no longer located at the surface of the objects. However, they are more likely to locate inside the objects. And votes are clustered together, which is easier for the cue combination from different parts of the object.

After votes are generated, VoteNet needs to generate the object proposal from votes and classify the objects. The votes are first clustered together by sampling and grouping. In order to integrate a clustering technique into the end-to-end network, VoteNet first conducts a farthest point sampling. A subset with K votes is sampled from the original set of votes. Then clusters are formed from these sampled votes by position. A cluster would be grown from each of the sampled votes, and other votes within a certain distance would be grouped into these clusters. Each of these clusters is a candidate object. The number K should be a number much larger than the object within the scene, and only the candidate that has a confident number larger than a certain value would be reserved, which would be included in the final object's proposal.

The traditional Hough Transformation method determines the boundary of objects by a back-tracking process. VoteNet aggregates votes and generates proposals in a more efficient way by leveraging a point-set learning network. A shared PointNet is chosen in

VoteNet for proposal generation. The vote clusters are first normalized to fully use the local vote geometry. Then, the cluster is processed by a module with a structure similar to PointNet. The votes clusters are first processed by an MLP independently before pooling by a channel-wised max pooling layer. Then, a second MLP is conducted to further combine information from different votes to generate the final proposal. The final proposal contains information on the assigned categories of the objects, the confidential scores, and some information that can reconstruct the bounding boxes.

VoteNet is the fundament of many state-of-the-art point cloud object detection methods. The following methods optimize VoteNet in different ways, including speed, voting strategies, and the localization of the bounding boxes (Yang et al., 2020; Zhang et al., 2020).

Point-based object detection methods have a great advantage over grid and voxel-based methods. They fully utilize the information in the point cloud, and they need less memory and computational resource.

2.5 Chapter Summary

According to the literature review, the more recent deep learning-based point cloud upsampling methods had more advantages than non-deep learning-based methods. Therefore, the thesis is more focused on the deep learning-based point cloud upsampling methods. The state-of-the-art point cloud upsampling methods were tested on a wide range of benchmarks. However, these benchmarks failed to include full indoor scenes. The evaluation metrics used to evaluate the state-of-the-art point cloud upsampling methods failed to illustrate the performance of point cloud upsampling methods on concrete practical applications. Therefore, this thesis would explore the application of point cloud upsampling in some more specific practical applications such as point cloud object detection.

Chapter 3

Proposed Methodology

This chapter introduces the specific methodology proposed in this thesis. Section 3.1 introduces the workflow of the proposed methodology. Section 3.2 introduces the datasets used in this thesis, including. Sections 3.3 and 3.4 introduce the point cloud upsampling and object detection methods, the process of training models, and the information of pre-trained models, respectively. Section 3.5 is the chapter summary.

3.1 Workflow

In this thesis, a methodology is proposed for enhancing the object detection result of the low-density point cloud via point cloud upsampling. Figure 3.1 shows the workflow of the proposed methodology, which consists of four parts: dataset preprocessing, object detection model training, point cloud upsampling, and point cloud object detection. In part one, the raw datasets are preprocessed into a training set and a low-density testing set. In part two, an object detection model is trained by the training set. In part three, the low-density testing set is upsampled by the pre-trained point cloud upsampling models. And in part four, object detection is conducted on the upsampled testing set by the trained object detection model.

3.2 Dataset

In order to fill up the lack of indoor benchmarks on point cloud upsampling research, this thesis adopts indoor LiDAR point cloud datasets for supplements. SUN RGB-D (Song et

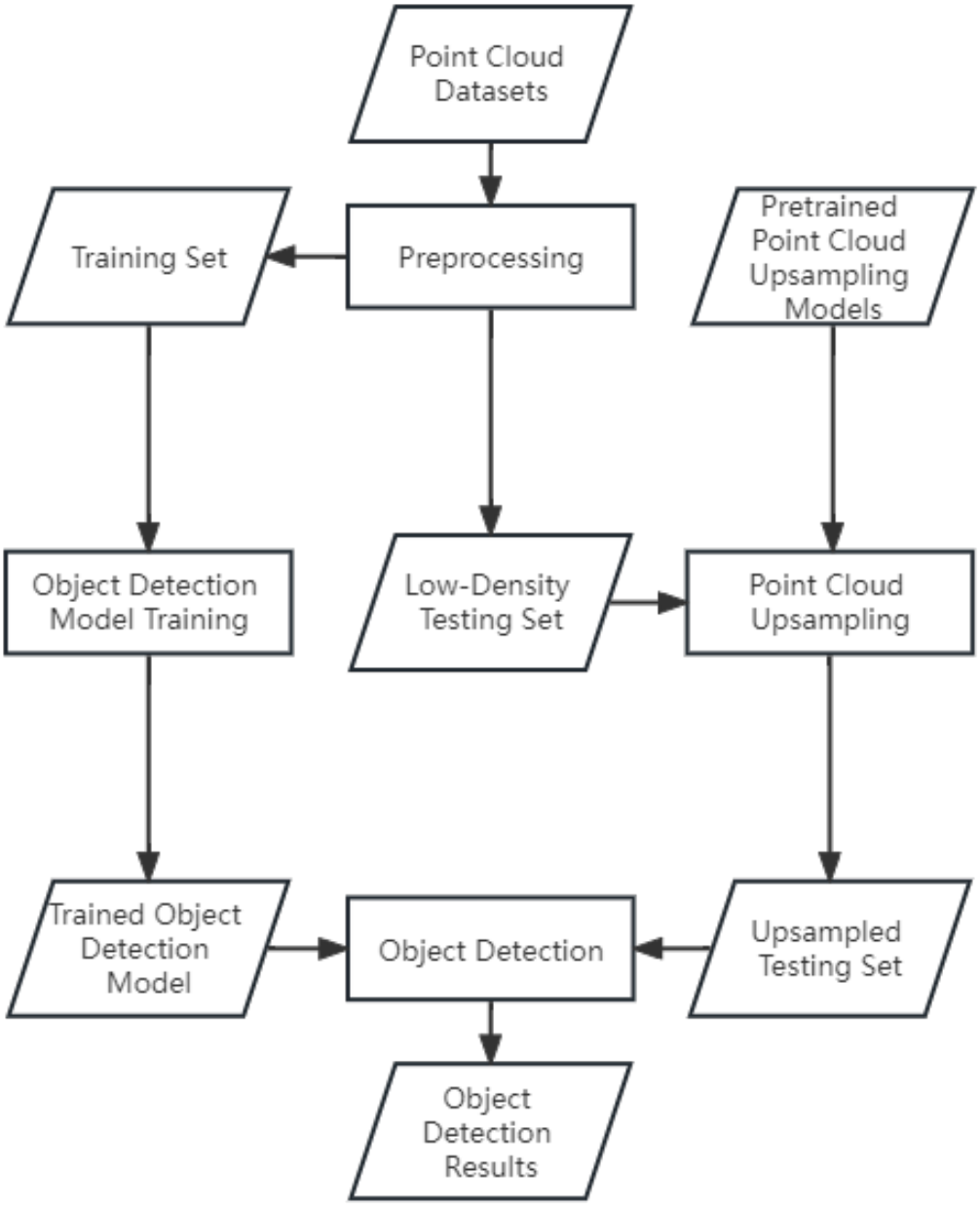


Figure 3.1 Workflow of the proposed method

al., 2015) is a famous public real-scanned indoor scene dataset. It contains scenes of rooms in the buildings. And the underground parking lot is a special kind of indoor environment with special characteristics. Compared to outdoor driving environments, underground parking lots are more challenging as their dim lighting, reflections on the road, the presence of pillars, and poor GNSS signals (Suhr & Jung, 2016), thus they require higher precision for the navigation systems. It can also be regarded as the working environment for indoor intelligent robots like other indoor environments. Therefore, both SUN RGB-D dataset and an underground parking lot dataset are adopted in this thesis. As there is a lack of public point cloud datasets for the underground parking lot, an indoor LiDAR point cloud dataset collected from the underground parking lot located at Highland Square, Kitchen, Waterloo is collected. The underground parking lot has a large scene with sparse object distribution and the room scenes represented by SUN RGB-D have dense object distribution, which makes them complementary

3.2.1 Dataset Collection and Specifications

3.2.1.1 Specifications of SUN RGB-D

SUN RGB-D is collected by Intel RealSense 3D Camera, Asus Xtion LIVE PRO, and Kinect V1 and V2. It contains 10335 RGB-D images, each of which represents a scene of a room. The objects within the rooms are common objects in daily life and belong to 800 different categories. Their 2D and 3D dense annotations are also included in the dataset. Table 3.1 shows the specification of SUN RGB-D.

Table 3.1 Specifications of SUN RGB-D

Parameter	Specifications
Sensor	Intel RealSense, Asus Xtion, and Kinect v1 and v2
Images Nuimber	10335
2D Annoations Number	146617
3D Annoations Number	64959
Objects Categories Number	Over 800

3.2.1.2 Collection and Specification of the Underground Parking Lot Dataset

Our own dataset on the underground parking lot is collected by a single Livox Horizon laser scanner used for data collection. The scanner has an effective field of view (FOV) of

the scanner is 81.7°(Horizontal) × 25.1°(Vertical), with a distance random error of fewer than 2 centimeters and an angular random error of less than 0.05°. The scanner's strongest return point rate is 480,000 pts/s and the dual return point rate is 240,000 pts/s. The scanner is mounted on a baby carrier to compose a Mobile Laser Scanning system. The pictures of the devices are shown in Figure 3.2 below. In the data collection process, the Mobile Laser Scanning system is pushed across the parking lot through the drive path, forming a closed loop.

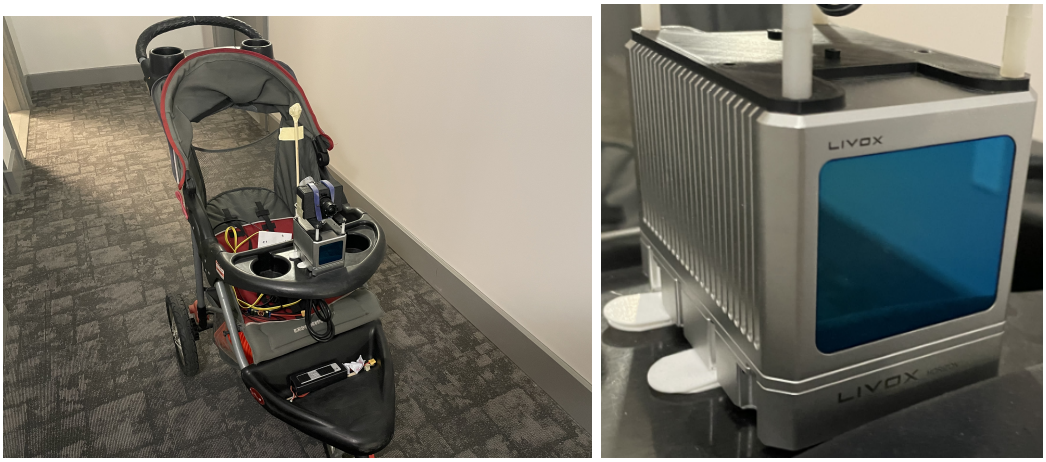


Figure 3.2 Mobile Laser Scanning system (left) and the Livox Horizon laser scanner(right)

Our dataset is collected from an underground parking lot located at Highland Square, Kitchener, Waterloo. The main object categories within the parking lot are pillars, columns, ceilings, floors, and vehicles. The whole parking lot covers an area of 8000 m^2, one-fifth of which is used for testing and the rest is used for training. The dataset we collected from the underground parking lot contains 170 million points, as shown in Figure 3.3. For each point, its 3D coordinates and RGB reflectance are recorded by the sensor. Table 3.2 shows the specification of the underground parking lot dataset.

3.2.2 Data Preprocessing

To generate a training set for efficient training, and a testing set simulating the point cloud captured by low-budget sensors, both SUN RGB-D dataset and the underground parking lot dataset are preprocessed.

For the SUN RGB-D dataset, the preprocessing process can be easily completed by the toolbox provided with the public dataset. The toolbox converts the whole 10335 RGB-D

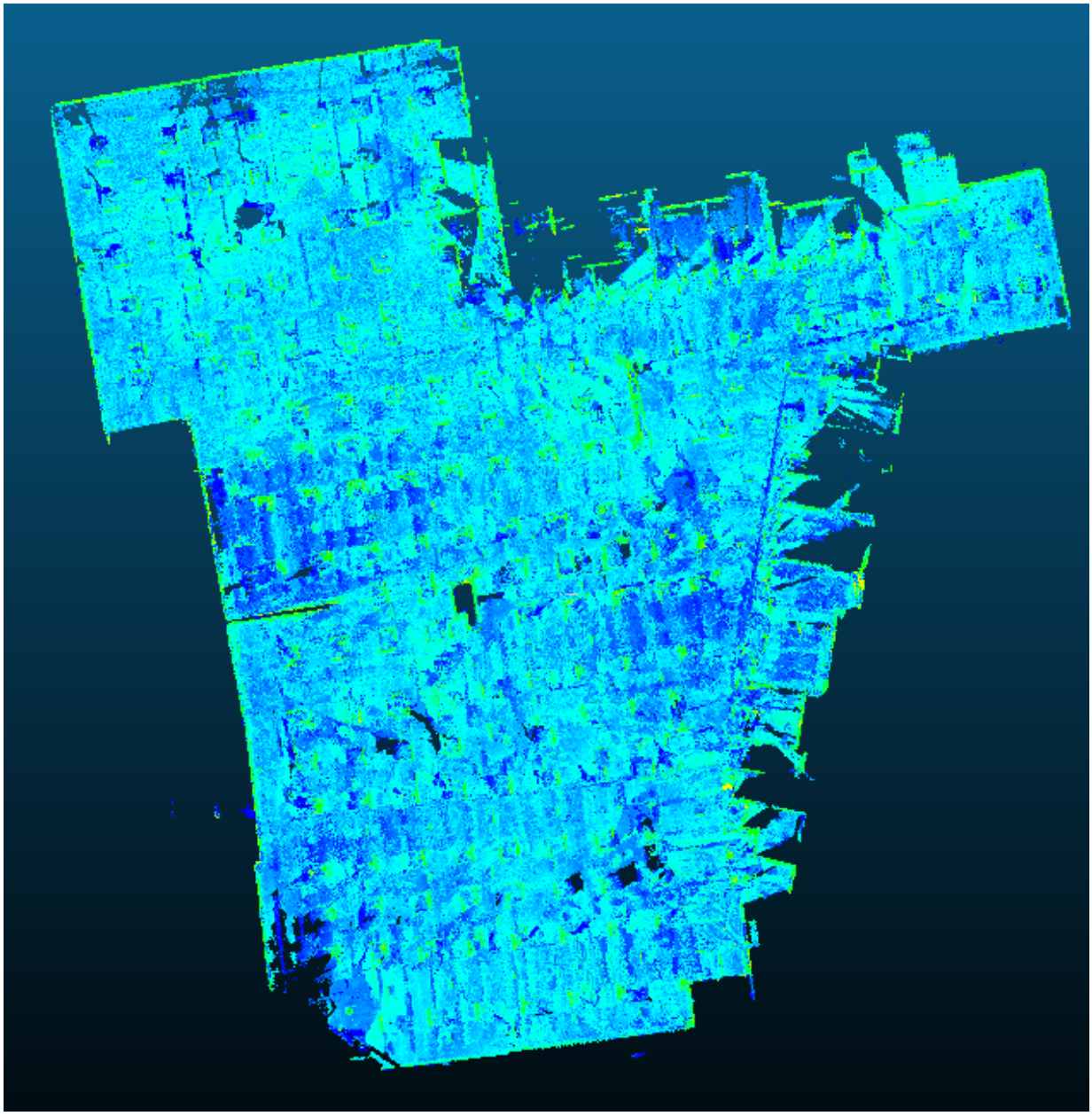


Figure 3.3 Bird's eye view of the collected dataset

Table 3.2 Specifications of the underground parking lot data

Parameter	Specifications
Sensor	Livox Horizon laser scanner
Area	8000 m^2
Number of Points	150 million
Density	About 17,000 pts/m^2
Main Structure	pillars, columns, ceilings, floors, vehicles
Type of Information	3D coordinates, RGB reflectance
Collecting Speed	0.5 m/s

images to point cloud data, each of which contains 50000 points and annotations of the objects inside. The whole dataset is split into around half for training and half for testing. As the sample number of SUN RGB-D is too large, we get a subset of 150 patches from the testing set to reduce the time cost.

For our own dataset collected from the underground parking lot, the dataset is first merged to generate the whole scene by Simultaneous Localization And Mapping (SLAM) algorithm provided by the scanner manufacturer (Livox, 2022). Then the dataset is manually labeled, and 3D bounding boxes are computed with the labels. We manually remove some obvious noises that have a large impact on the training processing, and also reserve some not obvious noise points to keep the characteristic of real-scanned data. Then the dataset is divided into cells with around $5m^2$ to reduce computation. Each cell contains at least one pillar, and the pillars inside should not be split by the cell boundaries. Then, the point cloud of each cell is subsampled into patches with 50000 points, to keep consistency with SUN RGB-D data. The whole dataset is split into the training set with 1023 patches and the testing set with 170 patches based on the training and testing area.

After the previous process, the point cloud density of both datasets is 1000 pts/m_2. Therefore, to simulate the sparse and non-uniform point cloud captured by the low-budget sensors, the testing set of both SUN RGB-D and our own collected dataset are subsampled to generate low-density datasets. The random subsample is used to stimulate the sparse and non-uniform characteristic of the point cloud collected by the low-budget sensors.

3.3 Point Cloud Upsampling

To get a more satisfying object detection result, point cloud upsampling methods are adopted to upsample the sparse and non-uniform low-density testing set to generate an upsampled testing set with higher density and uniformity. Three deep learning-based point cloud upsampling methods are chosen in this thesis, including PU-Net, 3PU, and PU-GCN. All the methods are adopted to upsample the low-density testing set, and their impacts on the final object detection results are compared.

PU-Net is a deep learning-based point cloud upsampling method. First, features of multi-level are extracted from the input point with the PointNet++ backbone. Then the features are upsampled in a multi-branch upsampling module, which input features into multiple parallel MLPs and concatenate the output to get the expanded feature. Then, the coordinate is reconstructed by a series of MLPs to generate the upsampled point cloud.

3PU is a multi-step point cloud upsampling method. It consists of a series of upsampling units, each of which can expand the size of the input point cloud twice. In each upsampling unit, 3PU first extracts features from the input point via intra-level dense connections, then it expands feature size via code assignment. Finally, features are reconstructed into 3D point clouds by MLPs. 3PU also adopts an inter-level skip connection to enhance the communication between upsampling units.

PU-GCN is a GCN-based point cloud upsampling method. PU-GCN first uses a KNN layer to build the neighborhood graph of the input point cloud. Then the feature is extracted from the neighborhood graph through a GCN layer and a series of Inception DenseGCN. The outputs of each DenseGCN layer are not only passed through the next DenseGCN but also concatenated with the final output. Then the extracted feature is input into the upsampling module with NodeShuffle layer, where the feature is first input into a GCN layer to expand feature size, and then shuffled to expand the feature number. Then features are input into the coordinate reconstructor, which consists of two MLPs to generate the final point cloud.

In this thesis, we adopt the public pre-trained models provided by the PU-GCN research team Qian et al. (2021). These models can enlarge the density of the input point cloud 4 times. These models were trained on PU1K for 100 epochs and a batch size of 64. The Adam optimizer was used with a learning rate of 0.001 and a beta of 0.9. The models were trained on the same computer with an NVIDIA TITAN 2080Ti GPU and an Intel Xeon E5-2680 CPU. These models are trained in the same condition, which is considered to be rigorous.

3.4 Point Cloud Object Detection

Object Detection is conducted on the upsampled testing set by the model trained with the training set or pre-trained. Then the object detection result is evaluated by the evaluation metric. Object detection with the same model is also conducted on the original and low-density testing set for comparison.

3.4.1 Object Detection Method

We adopt VoteNet as the object detection method. Although some subsequent methods optimize VoteNet in different aspects, we employ VoteNet to emphasize the effects of point cloud upsampling methods.

VoteNet is a point-based deep learning point cloud object detection method. VoteNet first extracts multi-level features from the input point cloud using PointNet++ backbone. Then, votes are generated from the multi-level features by the voting module, which consists of a series of fully connected MLPs. In the next step, votes are sampled and grouped into clusters. And in the final process, the final object proposals are generated from the vote clusters through a shared PointNet.

The final proposal is a multidimensional vector, and it contains an objectness score, which estimates whether the proposal represents a true object, a classification confidential number, an assigned class index, which estimates the possibility of the object belongs to the assigned class, and information of bounding boxes, including its center, size, and orientation.

3.4.2 Model Training

Different object detection models should be trained for different datasets. For SUN RGB-D, we used the pre-trained model provided by the VoteNet research team (Qi et al., 2019). The model was trained for 180 epochs and a batch size of 8. The Adam optimizer was used with a learning rate of 0.001 at the beginning, decreasing by 10 times at the 80^{th} and 120^{th} epochs. The models were trained on Volta Quadro GP100 GPU. This model is capable of the object detection of 10 common indoor objects, including chairs and tables.

And we also trained a VoteNet model on our own collected dataset. We focus on the object detection of Pillars and Vehicles, as they are obstacles in autonomous driving that the autonomous shouldn't bump into, and pillars are also the working object for

cleaning robots. Our model was also trained for 180 epochs and a batch size of 8. The Adam optimizer was used with a learning rate of 0.001 at the beginning, and the learning rate also decreased by 10 times at the 80^{th} and 120^{th} epochs. Our model was trained on one NVIDIA GeForce GTX 1080 8GB GPU and one eight-core Intel® CPU i7-9700k @ 3.60GHZ.

3.4.3 Evaluation Metric

The evaluation metric adopted in this thesis is Average Precision (AP) and Average Recall (AR) proposed by Song et al. (2015). After the predicted bounding boxes are generated, the 3D Intersection over Union (IoU) number between ground truth bounding boxes and predicted bounding boxes are calculated, and the pairs with the IOU number larger than a threshold are regarded as matched. In this thesis, the threshold is chosen to be 0.25 and 0.5. The matched bounding boxes are regarded as True Positive (TP). And the unmatched predicted and ground truth bounding boxes are regarded as False Negative (FN) and False Positive (FP). Let M be the number of matched pairs and G be the number of ground truth bounding boxes, the AR follows:

$$AR = \frac{TP}{TP + FN} = \frac{M}{G} \tag{3.1}$$

Each of the matched predicted bounding boxes have a confidential score. The confidential score can be used as another criterion and only matches with a confidential number higher than a threshold number are obtained. When changing this threshold, the precision and recall of the output subsequently change. Therefore, a Precision-Recall curve can be plotted, and AP of the output can be calculated by the Area Under Curve (AUC) of the Precision-Recall curve, which follows:

$$AP = \int_0^1 prec(rec)d(rec) \tag{3.2}$$

where *prec* is precision and *rec* indicates recall.

3.5 Chapter Summary

In this chapter, we discussed the workflow and details of our proposed methods. The workflow consisted of four steps: data preprocessing, point cloud upsampling, object detection

model training, and object detection. In data preprocessing, a training test for model training and a low-density testing set was generated. Then, the low-density testing set was upsampled by three pre-trained point cloud upsampling models, including PU-Net, 3PU, and PU-GCN. Parallelly, a VoteNet model for object detection was trained on the training set. Finally, object detection was conducted on the upsampled testing set by trained or pre-trained VoteNet models. We also illustrated the metrics for evaluation.

Chapter 4

Results and Discussion

This chapter introduces the result of this thesis and makes further discussion on it. Specifically, Section 4.1 discusses the setup details for this thesis. Section 4.2 illustrates the results, which include bout quantitative results and qualitative results. Section 4.3 summarizes this result and makes further discussion.

4.1 Experimental Setups

In the experiment process, we find that the decrease in point cloud density does not have an intense influence on the performance of the object detection models. The speed at which we push the Mobile Laser Scanning when collecting data is 0.5 m/s which is tens of times lower than the speed of the vehicles driving in the underground parking lot. And considering the fact that pillars and vehicles are repeatedly scanned from different angles, and some low-budget sensors have a point rate many times lower than our LiDAR sensors, the subsample ratio is chosen to be 32 to simulate the point cloud capturing by the low-budget sensors. To verify whether a 32 times subsample would result in an evident decrease in point cloud object detection performance, we plot the change of AP with the subsample ratio increase, and the result is shown in Figure 4.1. The result shows that the AP decrease at the subsample ratio of 32 is significant, so the selection of 32 times subsampling is confirmed.

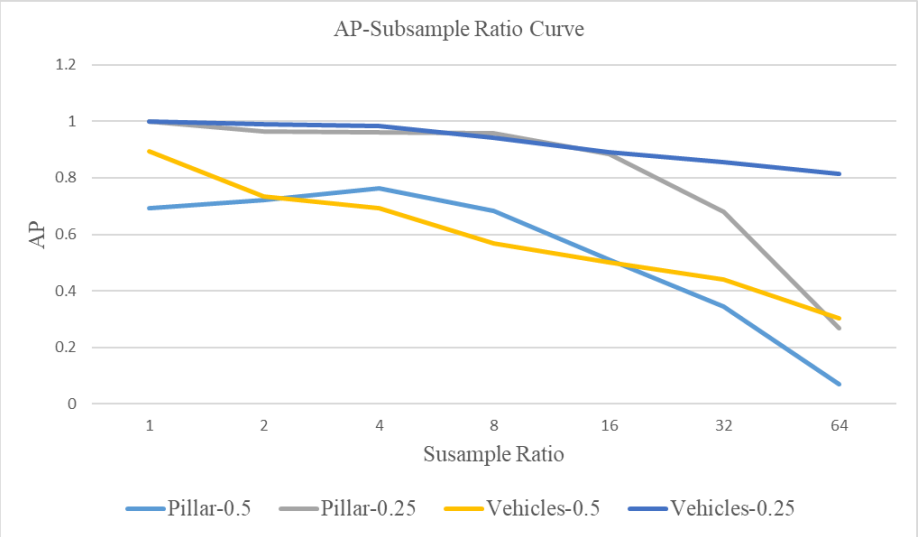


Figure 4.1 Relationship between subsampling ratio and AP

4.2 Experimental Results

4.2.1 Quantitative Results of the Underground Parking Lot Dataset

The quantitative results of our own collected dataset are shown in the table 4.1 below. According to the table, the VoteNet model we trained has high accuracy on the pillars and vehicle detection on our own collected dataset, with the AP and AR close to 1 under the threshold of 0.25. On the threshold of 0.5, its AP and AR are still higher than the rest of the data. And the 32 times decrease in density cause an evident decrease in object detection performance, and the AP and AR are the lowest. And after conducting point cloud upsampling, the performance of the point cloud object detection is improved, with the mean AP increasing from 0.77 and 0.39 to around 0.95 and 0.65 under the threshold of 0.25 and 0.5. According to Figure 4.1, these are similar to the mean AP when the original point cloud is subsampled 8 times, which has the same point cloud density as the generated point cloud. In practical application, this can result in a three times faster data collecting speed, which leads to a three-fourth time-saving. It can also decrease the budget of the sensor. And the mean AR at the threshold of 0.25 and 0.5 also increase from 0.79 and 0.48 to around 0.73 and 0.96, and the mean AR increases from around 0.5 to around 0.75.

The performance of the three point cloud upsampling methods is similar. Although their metric values are close, PU-GCN has the best effect on pillars, and PU-Net has the

best effect on vehicles. PU-Net also has the best average performance when the threshold is low, and 3PU has the best average performance when the threshold is high.

Table 4.1 AP(%) and AR(%) for the object detection result of low-density and upsampled data

	Origin	Low-Density	PU-Net	3PU	PUGCN
Pillar-AP@0.25	99.7	68.1	93.6	93.9	94.1
Pillar-AP@0.5	69.3	34.3	66.8	66.8	67.0
Pillar-AR@0.25	100.0	68.5	93.7	94.1	94.4
Pillar-AR@0.5	80.1	43.0	75.9	76.9	74.5
Vehicle-AP@0.25	100.0	85.5	97.6	95.8	96.9
Vehicle-AP@0.5	89.4	44.1	63.1	64.0	63.6
Vehicle-AR@0.25	100.0	88.5	98.5	96.3	97.8
Vehicle-AR@0.5	90.3	53.5	70.6	69.9	70.3
mAP@0.25	99.8	76.8	95.6	94.8	95.5
mAP@0.5	79.3	39.2	65.0	65.4	65.3
mAR@0.25	100.0	78.5	96.1	95.2	96.1
mAR@0.5	85.2	48.3	73.3	73.4	72.4

4.2.2 Qualitative Results of the Underground Parking Lot Dataset

Figures 4.2-4.6 show the visualized object detection results. We extract one patch from each of the cells and the object detection results of these patches are merged together. A subsampled point cloud of the whole testing area is also presented with bound boxes for reference. The bounding boxes represent the tested (or ground truth) vehicles or pillars, while purple stands for True Positive vehicles, blue stands for False Positive vehicles, light green stands for True Positive pillars, and dark green stands for False Positive pillars.

Figure 4.2 shows 23 pillars and 24 vehicles in the testing area. However, only 15 pillars and 19 vehicles are tested from the low-density dataset and 4 vehicles are False Positive. Figures 4.4-4.6 show that point cloud upsampling makes an improvement to the low-density data. After the upsampling of PU-Net, 21 True Positive pillars with 21 True Positive vehicles are detected from the scene with only 2 False Positive vehicles. 3PU has a better performance with 22 True Positive pillars and 23 True Positive vehicles detected and with no false detection. The performance of PU-GCN is the worst in this visualization. However, it still outperforms the low-density data, with 20 pillars and 20 vehicles of True Positive detection, and one pillar and three vehicles of False Positive detection.

Figure 4.2 Reference point cloud and ground truth bounding boxes

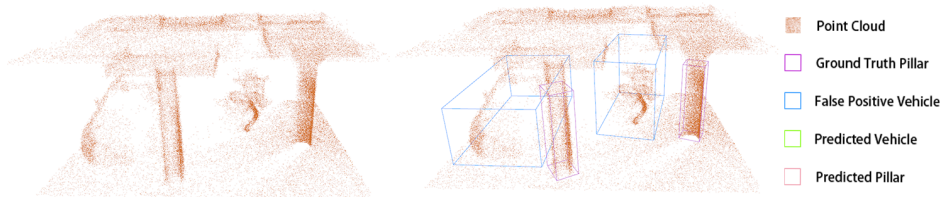
Figure 4.3 Object Detection Result of low-density point cloud

Figure 4.4 Object Detection Result of upsampled point cloud by PU-Net

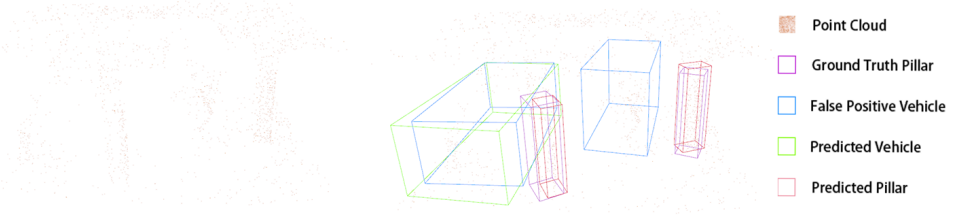
Figure 4.5 Object Detection Result of upsampled point cloud by 3PU

Figure 4.6 Object Detection Result of upsampled point cloud by PU-GCN

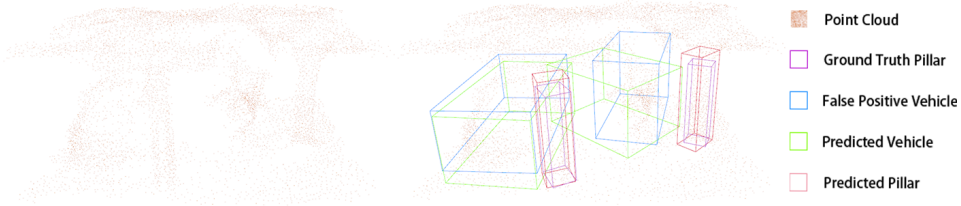
Figure 4.7 provides a detailed view of the point clouds in a cell, the ground truth bounding boxes, and object detection results. There are two vehicles and two pillars in this cell. However, only one vehicle and two pillars can be detected when the density is low. After the upsampled by all the three point cloud upsampling methods, two vehicles and two pillars can be detected. Overall, object detection on pillars and the car, which belongs to the vehicle category, would generate bounding boxes closer to the ground truth, and the predicted bounding boxes of the motorcycle, which also belongs to the vehicle category, have an orientation difference to the ground truth. This is because of the fact that the number of cars in this underground parking lot is more than motorcycles, and the trained VoteNet model learns more features from cars. Among all three point cloud upsampling methods, the point cloud generated by PU-GCN has the best object detection results in the motorcycle as its predicted bounding box has the biggest overlay with the ground truth. Under the same criteria, the performance of 3PU is better than PU-Net. This indicates that from PU-Net to PU-GCN, the feature capturing and feature reconstruction ability improves. For the object detection result on the other three objects, 3PU has the best results. And the predicted bounding boxes generated by point cloud upsampled by PU-GCN would cross the border of the ground truth bounding boxes. This can result from the overfitting of the noise points. As the most recent method, PU-GCN optimizes architecture from PU-Net and 3PU, which has better feature extracting and reconstruction ability and can be more likely to overfit noise points. This could explain the reason that the performance of PU-GCN is not as satisfying as the other two methods on some occasions.



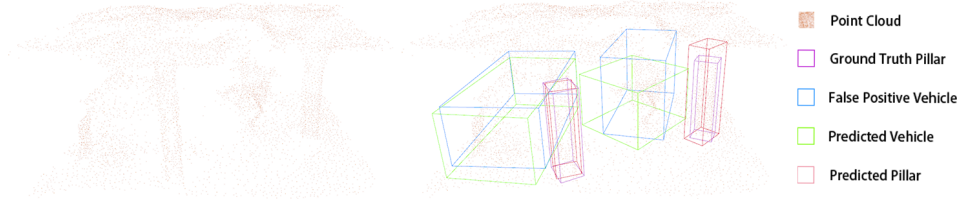
(a)



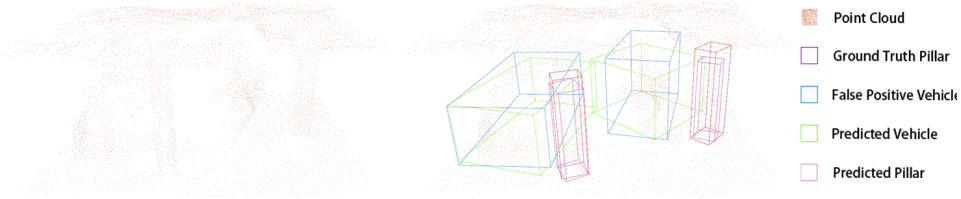
(b)



(c)



(d)



(e)

Figure 4.7 From top to down is: (a) original point cloud and ground truth bounding box, (b) low-density point cloud and the object detection result, and (c)-(e) upsampled point cloud by PU-Net, 3PU and PU-GCN and corresponding Object Detection Results

4.2.3 Quantitative Results of SUN RGB-D Dataset

Tables 4.2-4.5 show the quantitative results of the SUN RGB-D dataset. The sense of SUN RGB-D has more objects and more categories, even though point cloud upsampling methods still show their capability to enhance object detection results. According to the tables, the effect of point cloud upsampling is more evident when the threshold is 0.25. And in the threshold of 0.5, the amount of AP and AR increase is smaller.

The objects in SUN RGB-D often have more complicated structures than the underground parking lot dataset and the overall performance enhancement due is point cloud upsampling is lower than that of the underground parking lot dataset. It can indicate that the point cloud upsampling would have a better effect when the structures are not complicated. Under the threshold of 0.25, point cloud upsampling can enhance the object detection results from most of the categories. However, under the threshold of o.5, these methods do not enhance the object detection results of some categories, and some are less satisfying. Under the threshold of 0.5, these three point cloud upsampling methods cannot enhance the detection of chairs and bathtubs, the that of the bathtubs even decreases. This may be because that chairs and bathtubs have too many hollow or empty structures, which may be easily affected by the noise points. Among the rest of the objects, PU-GCN does not perform well on desks and dressers. The object that belongs to these categories are similar, and the overfitting of PU-GCN makes the generated point cloud even more similar, so the objects can be misclassified to the other category. 3PU does not perform well on tables and bookshelves. It has extremely bad performance on the bookshelves, the AP decreases even under the threshold of 0.25, and under 0.5, the decrease amount is huge. Point cloud upsampling methods cannot perfectly preserve the original shape and the multi-step upsampling structure of the 3PU increase the amount of the shape change. Therefore, the performance decrease of these two categories may be because the plain structure of the tables and the multi-layer structure of bookshelves are too sensitive to shape change.

Table 4.2 AP@0.25(%)) of the object detection result on SUN RGB-D

AP@0.25					
	Ground Truth	Low-density	PU-Net	3PU	PUGCN
bed	99.4	91.0	98.8	97.5	98.9
table	53.2	34.1	37.6	44.3	39.1
sofa	74.0	52.8	70.9	71.0	70.3
chair	91.3	68.9	72.9	78.2	75.1
toilet	98.2	80.1	99.1	74.0	91.6
desk	49.8	26.4	25.4	36.7	32.0
dresser	34.0	19.0	27.9	18.8	26.7
night stand	89.9	36.7	62.2	62.6	58.8
bookshelf	46.0	7.3	18.9	4.6	21.2
bathtub	100.0	100.0	100.0	100.0	100.0
mean	73.6	51.6	61.4	58.8	61.3

Table 4.3 AP@0.5(%) of the object detection result on SUN RGB-D

AP@0.5					
	Ground Truth	Low-density	PU-Net	3PU	PUGCN
bed	82.2	72.2	79.4	80.1	82.2
table	18.2	7.8	9.5	7.5	10.4
sofa	47.2	40.0	52.9	43.6	45.8
chair	77.3	48.8	43.7	46.5	48.4
toilet	74.8	43.7	46.5	64.9	63.0
desk	10.0	6.3	12.7	25.8	2.5
dresser	19.3	3.6	4.0	6.2	2.3
night stand	61.2	20.5	32.9	34.6	19.8
bookshelf	12.6	0.9	11.1	0.5	11.1
bathtub	83.3	66.7	33.3	50.0	52.8
mean	48.6	31.0	32.6	36.0	33.8

Table 4.4 AR@0.25(%) of the object detection result on SUN RGB-D

AR@0.25					
	Ground Truth	Low-density	PU-Net	3PU	PUGCN
bed	100.0	93.0	100.0	98.2	100.0
table	86.2	72.4	65.5	75.9	75.9
sofa	91.2	70.6	85.3	88.2	91.2
chair	99.2	85.2	94.3	93.4	89.3
toilet	100.0	100.0	100.0	90.0	100.0
desk	93.8	81.3	87.5	93.8	81.3
dresser	88.5	57.7	65.4	65.4	65.4
night stand	96.2	67.3	75.0	82.7	78.8
bookshelf	88.9	22.2	55.6	33.3	33.3
bathtub	100.0	100.0	100.0	100.0	100.0
mean	94.4	75.0	82.9	82.1	81.5

Table 4.5 AR@0.5(%) of the object detection result on SUN RGB-D

AR@0.5					
	Ground Truth	Low-density	PU-Net	3PU	PUGCN
bed	87.7	82.5	89.5	89.5	89.5
table	31.0	20.7	17.2	13.8	13.8
sofa	58.8	50.0	61.8	55.9	47.1
chair	84.4	61.5	55.7	61.5	63.1
toilet	80.0	80.0	70.0	80.0	70.0
desk	37.5	31.3	31.3	31.3	18.8
dresser	50.0	15.4	15.4	30.8	19.2
night stand	73.1	36.5	40.4	48.1	38.5
bookshelf	44.4	11.1	11.1	11.1	11.1
bathtub	83.3	66.7	33.3	50.0	66.7
mean	63.0	45.6	42.6	47.2	43.8

4.3 Chapter Summary

According to the experimental results, the effectiveness of point cloud upsampling methods on object detection can be proved. Point cloud upsampling methods have more effectiveness in scenes that do not have many objects and the detection accuracy requirements are not too high. Although many of the more recent point cloud upsampling methods can better reconstruct object detail structures, this optimization does not take a significant part in object detection. As an early-stage point cloud upsampling method, the performance of PU-Net is not always exceeded by the other two more advanced methods. Due to its feature extracting and reconstruction ability, PU-GCN is sensitive to noise, and its performance would decrease in noise-rich senses or on noise-sensitive objects. And the multi-step architecture of 3PU makes it not perform well on shape-change-sensitive objects. And these three methods do not perform well on objects having large hollow or empty structures. In a general view, the difference between these three methods in the application is not significant, so the selection of the upsampling methods in practical applications can be based on some more criteria including computation and time consumption.

Chapter 5

Conclusions and Recommendations

5.1 Conclusions

In this thesis, we enhance the object detection results of low-density datasets via point cloud upsampling methods. Three point cloud upsampling methods, PU-Net, 3PU, and PU-GCN are adopted in this thesis. And VoteNet is selected as the object detection method. These methods are tested on a dataset collected in the thesis from an underground parking lot located at Highland Square, Kitchener, Canada, and the public dataset SUN RGB-D, and their advantage and disadvantage in different circumstances are evaluated.

In this thesis, the object detection results were increased by 18.8%, 18.0%, and 18.7% on the underground parking lot dataset, and 9.8%, 7.2%, and 9.7% on SUN RGB-D in AP@0.25 with the conduct of the point cloud upsampling methods PU-Net, 3PU, and PU-GCN. Other evaluation metrics and visualization results also show that point cloud upsampling methods make a satisfying improvement in object detection results.

5.2 Contributions

The contributions of the thesis are listed as follows:

1. This thesis proposes a methodology for enhancing the object detection result of the low-density point cloud through point cloud upsampling. This can result in a decrease in the sensors budget or about three-fourths of time-saving.

2. Collecting an indoor point cloud dataset from an underground parking lot located at Highland Square, Kitchener, Waterloo, which filled in the blank to the lack of public underground parking lot point cloud datasets.

3. Evaluating the performance of three state-of-the-art point cloud upsampling methods on the ability on enhancing the point cloud object detection of the low-density point cloud across different environments, and analyzing their advantages and disadvantages.

5.3 Recommendations for Future Research

In this thesis, we face many difficulties and challenges and have many recommendations for future research. In this thesis, some attributes in the underground parking lot dataset are not utilized, such as intensity. This is for keeping consistent with the SUN RGB-D dataset, which does not have intensity attributes. However, as the laser has different reflectance rates on different kinds of surfaces, the intensity can be utilized to identify the type of objects and avoid manual labeling. Therefore, in future works which do not have this limitation, intensity is recommended to use. Because of the noise of the underground parking lot dataset, we have to remove some of the noise points manually to improve the training effectiveness. And the object detection methods are sensitive to noise points, and some noise points can lead to object deformation after upsampling. Therefore, future work can aim at making point cloud upsampling methods not sensitive to noise. And due to the noise points in our dataset in the underground parking lot, we cannot train the point cloud upsampling methods on that. Therefore, although most of the real-scan point cloud datasets reserve the noise points within to preserve the characteristics of the real-scan point cloud, we suggest that a clean real-scan indoor point cloud data set is also important for letting point cloud upsampling networks learning the feature of objects from the real world including internal building structures. The model trained in this way may have a better performance. And the current point cloud upsampling methods are not specific to certain applications. Therefore, a point cloud upsampling method specially designed for certain applications, such as object detection, should be meaningful. And at the early stage of the thesis, we also evaluate the performance of point cloud upsampling on semantic segmentation. However, the result is not good. And this can be another meaningful topic in the future.

The methodology proposed in this thesis can also be expanded. It can be tested in a wider range of datasets, and more point cloud upsampling methods can be tested on that. Our proposed method has a wide range of applications. One of the most evident applications is that intelligent systems with only low-budget sensors do not need more model

training steps, and can just utilize the model trained with higher-density datasets, and upsample their low-density datasets through point cloud upsampling. This is meaningful in autonomous driving and robotics. And the source data can also be collected at a faster speed, which is time-saving. We believe that point cloud upsampling would have more applications in the future.

References

Alexa, M., Behr, J., Cohen-Or, D., Fleishman, S., Levin, D., & Silva, C. T. (2003). Computing and rendering point set surfaces. *IEEE Transactions on Visualization and Computer Graphics*, *9*(1), 3–15.

Beltrán, J., Guindel, C., Moreno, F. M., Cruzado, D., Garcia, F., & De La Escalera, A. (2018). Birdnet: A 3d object detection framework from lidar information. *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 3517–3523.

Bogo, F., Romero, J., Loper, M., & Black, M. J. (2014). Faust: Dataset and evaluation for 3d mesh registration. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3794–3801.

Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al. (2015). Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*.

Chelly, M., & Samama, N. (2009). New techniques for indoor positioning, combining deterministic and estimation methods. *ENC-GNSS 2009: European Navigation Conference-Global Navigation Satellite Systems*, 1–12.

Chen, X., Ma, H., Wan, J., Li, B., & Xia, T. (2017). Multi-view 3d object detection network for autonomous driving. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1907–1915.

Chen, Z., Xu, A., Sui, X., Wang, C., Wang, S., Gao, J., & Shi, Z. (2022). Improved-uwb/lidar-slam tightly coupled positioning system with nlos identification using a lidar point cloud in gnss-denied environments. *Remote Sensing*, *14*(6), 1380.

Choi, J., Song, Y., & Kwak, N. (2021). Part-aware data augmentation for 3d object detection in point cloud. *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 3391–3397.

Dai, A., Chang, A. X., Savva, M., Halber, M., Funkhouser, T., & Nießner, M. (2017). Scannet: Richly-annotated 3d reconstructions of indoor scenes. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5828–5839.

Dinesh, C., Cheung, G., & Bajić, I. V. (2019). 3d point cloud super-resolution via graph total variation on surface normals. *2019 IEEE International Conference on Image Processing (ICIP)*, 4390–4394.

Ding, Z., Han, X., & Niethammer, M. (2019). Votenet: A deep learning label fusion method for multi-atlas segmentation. *Medical Image Computing and Computer Assisted Intervention–MICCAI 2019: 22nd International Conference, Shenzhen, China, October 13–17, 2019, Proceedings, Part III 22*, 202–210.

Geiger, A., Lenz, P., Stiller, C., & Urtasun, R. (2013). Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, *32*(11), 1231–1237.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

GPL software. (2022, March 30). *Cloudcompare* (Version 2.12.0). https://www.cloudcompare.org

Griffiths, D., & Boehm, J. (2019). A review on deep learning techniques for 3d sensed data classification. *Remote Sensing*, *11*(12), 1499.

Guerry, J., Boulch, A., Le Saux, B., Moras, J., Plyer, A., & Filliat, D. (2017). Snapnet-r: Consistent 3d multi-view semantic labeling for robotics. *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 669–678.

Hu, Q., Yang, B., Xie, L., Rosa, S., Guo, Y., Wang, Z., Trigoni, N., & Markham, A. (2020). Randla-net: Efficient semantic segmentation of large-scale point clouds. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11108–11117.

Hua, B.-S., Pham, Q.-H., Nguyen, D. T., Tran, M.-K., Yu, L.-F., & Yeung, S.-K. (2016). Scenenn: A scene meshes dataset with annotations. *2016 fourth International Conference on 3D vision (3DV)*, 92–101.

Huang, H., Li, D., Zhang, H., Ascher, U., & Cohen-Or, D. (2009). Consolidation of unorganized point clouds for surface reconstruction. *ACM Transactions on Graphics (TOG)*, *28*(5), 1–7.

Huang, H., Wu, S., Gong, M., Cohen-Or, D., Ascher, U., & Zhang, H. (2013). Edge-aware point set resampling. *ACM Transactions on Graphics (TOG)*, *32*(1), 1–12.

Ku, J., Mozifian, M., Lee, J., Harakeh, A., & Waslander, S. L. (2018). Joint 3d proposal generation and object detection from view aggregation. *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1–8.

Lang, I., Manor, A., & Avidan, S. (2020). Samplenet: Differentiable point cloud sampling. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7578–7588.

Li, G., Muller, M., Thabet, A., & Ghanem, B. (2019a). Deepgcns: Can gcns go as deep as cnns? *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 9267–9276.

Li, R., Li, X., Fu, C.-W., Cohen-Or, D., & Heng, P.-A. (2019b). Pu-gan: A point cloud upsampling adversarial network. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 7203–7212.

Liang, M., Yang, B., Wang, S., & Urtasun, R. (2018). Deep continuous fusion for multi-sensor 3d object detection. *Proceedings of the European Conference on Computer Vision (ECCV)*, 641–656.

Lipman, Y., Cohen-Or, D., Levin, D., & Tal-Ezer, H. (2007). Parameterization-free projection for geometry reconstruction. *ACM Transactions on Graphics (TOG)*, *26*(3), 22–es.

Liu, Z., Zhang, Z., Cao, Y., Hu, H., & Tong, X. (2021). Group-free 3d object detection via transformers. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2949–2958.

Livox. (2022). Livox-sdk/lio-livox: A robust lidar-inertial odometry for livox lidar. https://github.com/Livox-SDK/LIO-Livox

Mao, J., Shi, S., Wang, X., & Li, H. (2022). 3d object detection for autonomous driving: A review and new outlooks. *arXiv preprint arXiv:2206.09474*.

McCrae, J., Mordatch, I., Glueck, M., & Khan, A. (2009). Multiscale 3d navigation. *Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games*, 7–14.

Milletari, F. (2018). *Hough voting strategies for segmentation, detection and tracking* (Doctoral dissertation). Technische Universität München.

Moosmann, F., Pink, O., & Stiller, C. (2009). Segmentation of 3d lidar data in non-flat urban environments using a local convexity criterion. *2009 IEEE Intelligent Vehicles Symposium*, 215–220.

Pham, Q.-H., Sevestre, P., Pahwa, R. S., Zhan, H., Pang, C. H., Chen, Y., Mustafa, A., Chandrasekhar, V., & Lin, J. (2020). A 3d dataset: Towards autonomous driving in challenging environments. *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2267–2273.

Phan, A. V., Le Nguyen, M., Nguyen, Y. L. H., & Bui, L. T. (2018). Dgcnn: A convolutional neural network over large-scale labeled graphs. *Neural Networks*, *108*, 533–543.

Pickup, D., Sun, X., Rosin, P. L., Martin, R. R., Cheng, Z., Nie, S., & Jin, L. (2015). Shrec'15 track: Canonical forms for non-rigid 3d shape retrieval.

Preiner, R., Mattausch, O., Arikan, M., Pajarola, R., & Wimmer, M. (2014). Continuous projection for fast l1 reconstruction. *ACM Transactions on Graphics (TOG)*, *33*(4), 47–1.

Qi, C. R., Litany, O., He, K., & Guibas, L. J. (2019). Deep hough voting for 3d object detection in point clouds. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 9277–9286.

Qi, C. R., Su, H., Mo, K., & Guibas, L. J. (2017a). Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 652–660.

Qi, C. R., Yi, L., Su, H., & Guibas, L. J. (2017b). Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing systems*, *30*.

Qian, G., Abualshour, A., Li, G., Thabet, A., & Ghanem, B. (2021). Pu-gcn: Point cloud upsampling using graph convolutional networks. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11683–11692.

Shilane, P., Min, P., Kazhdan, M., & Funkhouser, T. (2004). The princeton shape benchmark. *Proceedings Shape Modeling Applications, 2004.*, 167–178.

Soler, F., Melero, F. J., & Luzón, M. V. (2017). A complete 3d information system for cultural heritage documentation. *Journal of Cultural Heritage*, *23*, 49–57.

Song, S., Lichtenberg, S. P., & Xiao, J. (2015). Sun rgb-d: A rgb-d scene understanding benchmark suite. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 567–576.

Suhr, J. K., & Jung, H. G. (2016). Automatic parking space detection and tracking for underground and indoor environments. *IEEE Transactions on Industrial Electronics*, *63*(9), 5687–5698.

Uy, M. A., Pham, Q.-H., Hua, B.-S., Nguyen, T., & Yeung, S.-K. (2019). Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 1588–1597.

Wang, Y., Wu, S., Huang, H., Cohen-Or, D., & Sorkine-Hornung, O. (2019). Patch-based progressive 3d point set upsampling. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5958–5967.

Williams Jr, G. M. (2017). Optimization of eyesafe avalanche photodiode lidar for automobile safety and autonomous navigation systems. *Optical Engineering*, *56*(3), 031224–031224.

Wu, B., Wan, A., Yue, X., & Keutzer, K. (2018). Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 1887–1893.

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., & Xiao, J. (2015). 3d shapenets: A deep representation for volumetric shapes. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1912–1920.

Yan, Y., Mao, Y., & Li, B. (2018). Second: Sparsely embedded convolutional detection. *Sensors*, *18*(10), 3337.

Yang, Y., Feng, C., Shen, Y., & Tian, D. (2018). Foldingnet: Point cloud auto-encoder via deep grid deformation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 206–215.

Yang, Z., Sun, Y., Liu, S., & Jia, J. (2020). 3dssd: Point-based 3d single stage object detector. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11040–11048.

Yu, L., Li, X., Fu, C.-W., Cohen-Or, D., & Heng, P.-A. (2018). Pu-net: Point cloud upsampling network. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2790–2799.

Yuan, W., Khot, T., Held, D., Mertz, C., & Hebert, M. (2018). Pcn: Point completion network. *2018 International Conference on 3D vision (3DV)*, 728–737.

Zhang, Y., Zhao, W., Sun, B., Zhang, Y., & Wen, W. (2022). Point cloud upsampling algorithm: A systematic review. *Algorithms*, *15*(4), 124.

Zhang, Z., Sun, B., Yang, H., & Huang, Q. (2020). H3dnet: 3d object detection using hybrid geometric primitives. *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XII 16*, 311–329.

Zhou, J., Tan, X., Shao, Z., & Ma, L. (2019). Fvnet: 3d front-view proposal generation for real-time object detection from point clouds. *2019 12th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, 1–8.

Zhou, Y., & Tuzel, O. (2018). Voxelnet: End-to-end learning for point cloud based 3d object detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4490–4499.