國立臺灣大學電機資訊學院資訊工程學系

碩士論文

Department of Computer Science and Information Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master's Thesis

基於多幀點雲對齊的三維物體偵測增強

3D Object Detection Enhancement Based on Multi-frame Point Cloud Alignment

王浚睿

Jun-Rui Wang

指導教授: 施吉昇 博士

Advisor: Chi-Sheng Shih, Ph.D.

中華民國 113 年 8 月

August, 2024

doi:10.6342/NTU202403763

誌謝

感謝指導教授施吉昇教授在論文研究和寫作上的悉心指導，也感謝實驗室的林祥瑞學長提供的寶貴建議，以及實驗室同學在資料收集方面的辛勤努力。最後，感謝家人及朋友在碩士期間給予的支持。

台北，夏，2024

王浚睿

doi:10.6342/NTU202403763

摘要

三維物件偵測是自動駕駛系統中不可或缺的組件，負責定位和分類由感測器收集的點雲或深度影像中的物件。這項任務使得自動駕駛車輛和路邊單元能夠有效地感知其周圍環境。此外，後續的決策任務大量依賴於三維物件偵測的結果，因此其準確性直接影響自動駕駛系統的性能和安全性。

近年來，大多數關於三維物件偵測的研究都利用深度神經網絡，需要標註的數據集來訓練模型。然而，在點雲中標註物件邊界框是一項耗時且具有挑戰性的任務。光達受到遮擋影響，只能提供環境的部分點雲。人工標註者發現很難在沒有其他感測器輔助的情況下為部分點雲標註完整的邊界框。

在這份工作中，我們提出了一個點雲對齊流程，可以對齊稀疏的車輛點雲而無需任何標註數據，並從聚合的點雲中生成邊界框。我們的流程利用車輛的輪廓進行對齊，解決了基於特徵的配準方法難以解決的稀疏點雲對齊挑戰。該流程包括一個邊界框估算器，用於生成粗略的邊界框，基於這些粗略邊界框進行初始對齊，並結合點對點和面對面方法進行點雲配準。

實驗結果顯示，我們的方法改善了邊界框的品質。在 IoU 閾值為 0.7 時，召回率提高了 10%，而且在平移誤差和旋轉誤差方面也勝過基於特徵的配準方法。

關鍵字：三維物件偵測、點雲配準、多幀點雲、自駕車、無監督學習

doi:10.6342/NTU202403763

Abstract

3D object detection is an essential component of autonomous driving systems, responsible for localizing and classifying the objects within the point clouds or depth images collected by sensors. This task enables self-driving vehicles and roadside units to effectively perceive their environment. Moreover, the subsequent tasks such as decision-making heavily relying on the results of 3D object detection. Therefore, the accuracy of 3D object detection directly influences the performance and safety of autonomous driving systems.

Recently, most works on 3D object detection leverage deep neural networks, requiring annotated datasets to train models. However, annotating object bounding boxes in point clouds is a time-consuming and challenging task. LiDAR is affected by occlusions and can only provide partial views of the environment. Human annotators find it difficult to label complete bounding boxes for partial point clouds without the assistance of other sensors.

In this work, we propose a point cloud alignment pipeline that can align sparse vehicle point clouds without requiring any annotated data and generate bounding boxes from aggregated point cloud. Our pipeline uses the vehicle's contour for alignment, addressing the sparse point cloud alignment challenge that feature-based registration methods struggle to solve. The pipeline comprises a bounding box estimator for generating rough bounding boxes, initial alignment based on these rough bounding boxes, and point cloud registration combining point-to-point and plane-to-plane methods.

The experimental results show that our method improves the quality of bounding

doi:10.6342/NTU202403763

boxes. It achieves a 10% increase in recall at an IoU threshold of 0.7, and outperforms feature-based registration methods in terms of translation error and rotation error as well.

Keywords: 3D Object Detection, Point Cloud Registraion, Multi-frame Point Clouds, Autonomous Vehicles, Unsupervised Learning

doi:10.6342/NTU202403763

Contents

doi:10.6342/NTU202403763

doi:10.6342/NTU202403763

doi:10.6342/NTU202403763

List of Figures

doi:10.6342/NTU202403763

List of Tables

doi:10.6342/NTU202403763

Chapter 1

Introduction

1.1 Motivation

3D object detection is an essential component of autonomous driving systems, responsible for localizing and classifying the objects within the point clouds or depth images collected by sensors. This task enables self-driving vehicles and roadside units to effectively perceive their environment. Moreover, the subsequent tasks such as decision-making heavily relying on the results of 3D object detection. Therefore, the accuracy of 3D object detection directly influences the performance and safety of autonomous driving systems.

Recently, most works on 3D object detection leverage deep neural networks, requiring annotated datasets to train models. However, annotating object bounding boxes in point clouds is a time-consuming and challenging task. LiDAR is affected by occlusions and can only provide partial views of the environment. Human annotators find it difficult to label complete bounding boxes for partial point clouds without the assistance of other sensors. As shown in Figure 1.1, the point cloud contains only the front-left portion of the

doi:10.6342/NTU202403763

object. Due to the absence of the rear portion, it is impossible to ascertain the true size of the object, making determination of the object's bounding box even more challenging.

The sequence of point clouds may capture different viewpoints of the same object. By aggregating point clouds from multiple frames, we can achieve a more comprehensive understanding of the object's shape and generate more reliable bounding box labels, or can assists annotators in accurately determining the object's size. As shown in Figure 1.2, the aggregated point cloud offers richer geometric information about the object. Our work is grounded in this concept, where we introduce a framework to aggregate point clouds from various time frames, using this aggregated data to generate bounding box labels. These generated bounding box labels can use for training deep learning models or providing detailed shape information to support human annotators.

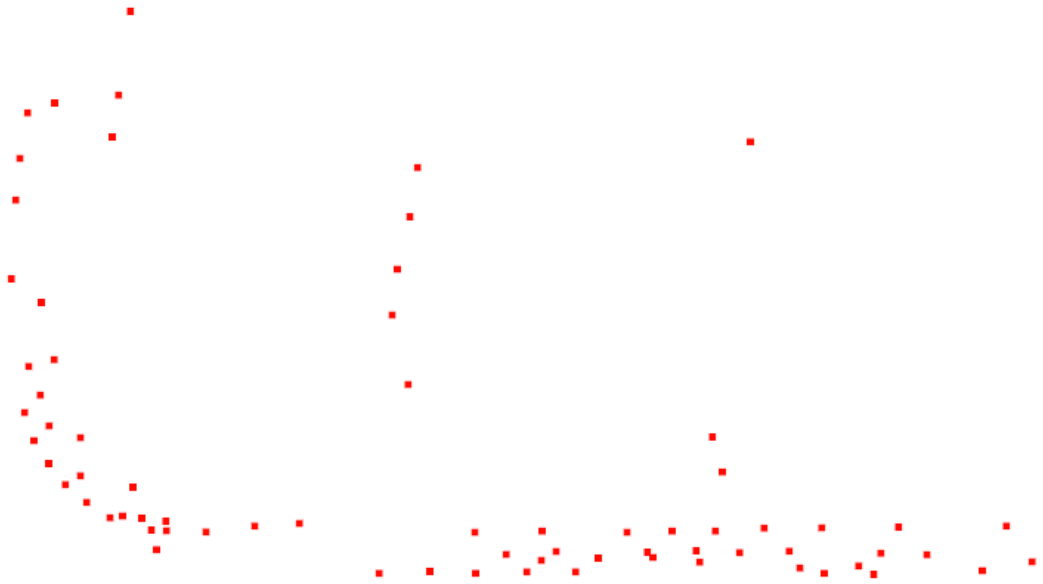


Figure 1.1: Point Cloud of an Object from Single Frame.

doi:10.6342/NTU202403763

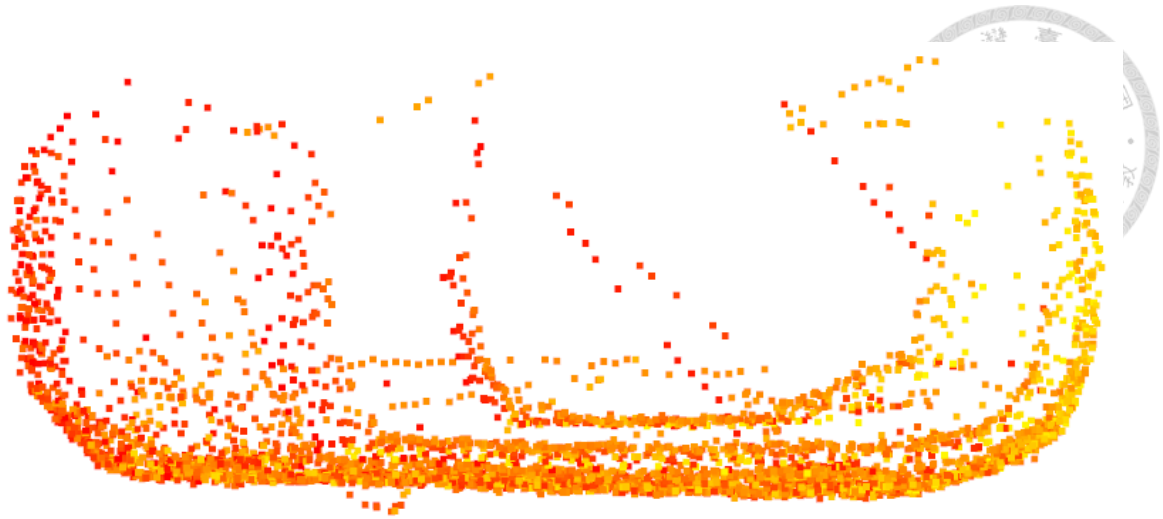


Figure 1.2: Point Cloud of an Object from Multiple Frame.

1.2 Contribution

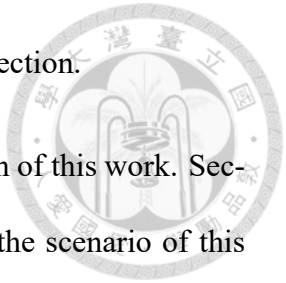
This work proposes a pipeline for aligning object point clouds across multiple frames and generating bounding boxes based on the aggregated point cloud. This pipeline can be used on the data collected by roadside LiDAR. The contribution of this work lies in designing a point cloud alignment framework capable of aligning sparse vehicle point clouds without training data and automatically generating bounding box annotations. We address the alignment of sparse and unevenly distributed point clouds, which traditional point cloud registration methods struggle to handle, by utilizing geometric features of the point clouds. The experimental results demonstrate that our method can handle sparse point clouds which current methods struggle to process effectively.

1.3 Thesis Organization

The remainder of this thesis is organized as follows.

Chapter 2 presents the concept of 3D object detection and point cloud registration. It

doi:10.6342/NTU202403763



also presents the previous related works about Multi-frame object detection.

Chapter 3 presents the system architecture and problem definition of this work. Section 3.1 presents system architecture, including the assumption and the scenario of this work. Section 3.1 presents the formally formulates of the problem.

Chapter 4 presents the design and implementation details of this work. Chapter 4.1 presents the workflow of this work. Chapter 4.2 to Chapter 4.5 presents the implementation details of each components in the workflow.

Chapter 5 presents the evaluation results of our experiments. Chapter 5.1 presents the evaluation metrics used in the experiments. Chapter 5.2 presents the quantitative results and their analysis. Chapter 5.3 presents the visualized examples.

Chapter 6 presents the conclusion of this work.

doi:10.6342/NTU202403763

Chapter 2

Background and Related Works

2.1 Background

This section presents the basic concept of 3D object detection using point cloud data and point cloud registration.

3D object detection is the task to localizing and identifying the objects in the point cloud and output the objects' bounding boxes, each bounding box contain center, size, and the heading. Recent works on 3D object detection focus on utilizing deep neural networks. These methods can be classified into three categories based on how they process point clouds: voxel-based [1, 2, 3], point-based [4, 5, 6], and voxel-point-based [7, 8]. Voxel-based methods divide the point cloud into voxels and then extract features from these voxels by using 2D or 3D Convolutional Neural Networks. Point-based methods extract features directly from the points by using special network architecture such as PointNet [9]. Voxel-point-based extract features from both voxels and points.

Point cloud registration is a problem of finding a transformation to align two point sets. ICP is the widely-used method for soloving point cloud registration problem. The

doi:10.6342/NTU202403763

input to the ICP is the source poinr cloud and the reference point cloud, the output of ICP is a transformation matrix that align source point cloud to reference point cloud. The following is the steps of the ICP algorithm:

1. For each point in the source point cloud, find and match the closest point in the reference point cloud.
2. Estimate the transformation matrix that minimize the RMS point to point distance of the corresponding point pairs.
3. Apply the transformation matrix on the source point cloud.
4. Repeat 1. $\sim$ 3. until the termination condition is satisfied.

2.2 Related Works

This section presents related works relevant to our work, including multi-frame 3D object detection, point cloud registrationm, and unsupervised 3D object detection.

2.2.1 Multi-frame 3D Object Detection

Recent work on multi-frame 3D object detection can be classified into two categories: frame-based and object-based. Frame-based methods extend the input of single-frame object detectors to accept multi-frame inputs. Some methods concatenate multiple point clouds into a single point cloud and then input this aggregated point cloud into a single-frame object detector. Object-based methods, on the other hand, process each object's point cloud across different frames rather than considering the entire scene. Compared to frame-based methods, object-based methods can handle longer sequences of point clouds.

doi:10.6342/NTU202403763

3D-MAN: 3DMulti-frame Attention Network for Object Detection

3D-MAN [10] uses a single-frame object detector to generate bounding box proposals. These proposals along with their features are then stored in a memory bank. Finally, an attention-based module is employed to extract and aggregate temporal features from the memory bank. However, the attention-based module computes relationships not only between the same objects across different frames but also between different objects across different frames. This increases the computational complexity of the model and limits its capability to handle longer sequences of point clouds.

Offboard 3D Object Detection from Point Cloud Sequences

Offboard 3D Object Detection from Point Cloud Sequences [11] is a work aims to utilize the point cloud sequence to generate more accurate bounding box. This work first using single-frame object detection and object tracking to obtain the track of an object. Then using a deep neural network to process the track of the object and output more accurate bounding box. This work is a supervised deep learning methods and therefore requires annotated data for training. Waymo Open Dataset is used for training and evaluating.

2.2.2 Point Cloud Registraion

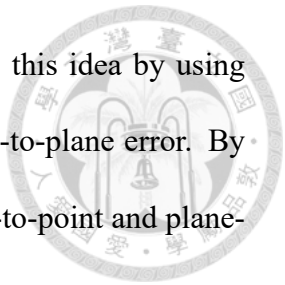
This section presents methods for solving the point cloud registration problem, including ICP variants, global registration methods, and deep learning-based approaches.

ICP Variants

ICP variants improve the steps in ICP. Chen and Medioni [12] used the plane-to-point error instead of the point-to-point error to take advantage of the planes near the points,

doi:10.6342/NTU202403763

rather than considering the points alone. GICP [13] further extends this idea by using the covariance matrix of the planes near the points to compute plane-to-plane error. By adjusting the covariance matrix, GICP can also switch between point-to-point and plane-to-point error computations.



Global Registraion

ICP and its variants rely heavily on the quality of the initial transformation because they use coordinate distances to find the nearest point pairs. To obtain this initial transformation, global registration methods are typically used to estimate a rough alignment. Unlike ICP and its variants, which match points directly based on their coordinates, global registration methods use point features for matching. A commonly used method for computing these features is Fast Point Feature Histograms (FPFH) [14]. FPFH calculates features that describe the geometric information in the neighborhood of each point using statistical methods. After computing the features for each point, point pairs can be matched based on the distance between these features. Point pairs obtained through feature matching often include many outliers. Global registration methods generally address this issue; for instance, Fast Global Registration [15] reduces outliers by filtering feature-matched point pairs, while TEASER++ [16] changes the method for solving the transformation matrix to make it less sensitive to outliers. Although FPFH can effectively capture point features, it requires dense point clouds to generate meaningful features.

Deep Learning-based Point Cloud Registraion

Recently, there have been several efforts to use deep neural networks to solve the point cloud registration problem. These methods generally rely on neural networks to handle feature extraction and point pair matching in global registration. One such method

doi:10.6342/NTU202403763

is Deep Closest Point (DCP) [17]. First, it uses PointNet [9] and DGCNN [18] to extract features from the point clouds. Subsequently, an attention-based module [19] is used to predict a soft matching between the source and reference point clouds. Finally, the transformation matrix is predicted based on the soft matching results. Although these methods perform well on dense point clouds, they struggle to achieve good results on sparse point clouds due to the difficulty in obtaining meaningful point cloud features.

2.2.3 Unsupervised 3D Object Detection

L-shape fitting [20] is a method for determining the bounding box of a point cloud without requiring a dataset for training. It is based on the idea that the point cloud of a vehicle typically approximates an L-shaped form. The bounding box is determined by finding the one that best fits this L-like shape. Since optimization methods to find this bounding box can be computationally intensive, L-shape fitting exhaustively explores all possible angles to generate all potential bounding boxes and then selects the best one based on certain criteria.

doi:10.6342/NTU202403763

Chapter 3

System Architecture and Problem Definition

This chapter presents the system architecture and problem definition details. Section 3.1 presents the system architecture. Section 3.2 presents the problem definition of this work. Section 3.3 presents the challenges of this work.

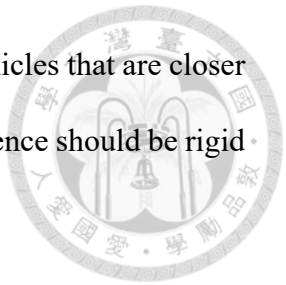
3.1 System Architecture

This work aims to align a sequence of point clouds belonging to the same object and generate corresponding bounding boxes for each point cloud using the aggregated point cloud after alignment. We align the point clouds by solving point cloud registration problem.

The input point cloud must be collected from stationary roadside LiDAR, and points in the point cloud should contain coordinate information, while other additional information, such as intensity, is not required. The height of the LiDAR should be the same as that of regular vehicles, approximately 1.6 meters. The reason for setting this height is

doi:10.6342/NTU202403763

that, at this level, distant vehicles are less likely to be occluded by vehicles that are closer to the LiDAR. The object corresponding to the input point cloud sequence should be rigid object. The output bounding boxes contain center, size, and heading.



3.2 Problem Definition

The problem of this work can be formulated as follow:

The input to this work is a sequence of point cloud belonging to the same object collected by LiDAR, denoted as $\boldsymbol{P}$ and is defined as:

$$\boldsymbol{P} = \{P_1, P_2, ..., P_N\} \tag{3.1}$$

$$P_i = \{(x_1, y_1, z_1), (x_2, y_2, z_2), ..., (x_M, y_M, z_M)\} \tag{3.2}$$

where N is the length of the point cloud sequence $\boldsymbol{P}$ and M is the number of the points in point cloud P_i.

The output of this work is the bounding boxes corresponding to each point cloud in the input point cloud sequence $\boldsymbol{P}$. The bounding boxes denoted as $\boldsymbol{B}$ and is defined as:

$$\boldsymbol{B} = \{B_1, B_2, ..., B_N\} \tag{3.3}$$

$$B_i = \{x, y, z, s_x, s_y, s_z, \theta\} \tag{3.4}$$

where x, y, and z is the center of the bounding box B_i; s_x, s_y, and s_z is the size of the bounding box B_i; θ is the heading of the bounding box B_i.

The output bounding boxes $\boldsymbol{B}$ has to minimize the IoU between $\boldsymbol{B}$ and the ground

doi:10.6342/NTU202403763

truth $\boldsymbol{B^{gt}}$, ground truth $\boldsymbol{B^{gt}}$ and IoU is defined as:

$$\boldsymbol{B^{gt}} = \left\{B_1^{gt}, B_2^{gt}, ..., B_N^{gt}\right\} \tag{3.5}$$

$$IoU_i = \frac{B_i \cap B_i^{gt}}{B_i \cup B_i^{gt}} \tag{3.6}$$

3.3 Challenges

Although there has been extensive research on point cloud registration, the problem that this work aims to solve involves several challenges. The two point clouds to be registered are somewhat separated, making it impossible to directly use ICP and its variants without an initial alignment. Furthermore, using feature-based point cloud registration methods is challenging because the vehicle point clouds differ from those typically used in general point cloud registration studies. They are sparser and have uneven density distribution, making it difficult to extract useful information for point cloud registration from the points and their neighbors. This results in poor performance of feature-based point cloud registration methods. Therefore, alternative methods are needed to obtain the initial alignment.

doi:10.6342/NTU202403763

Chapter 4

Design and Implementation

This chapter presents the workflow of our work and the design and Implementation details of each component in the workflow. Section 4.1 presents the workflow of our method. Section 4.2 presents the implementation details of the bounding box estimator. Section 4.3 presents the implementation details of the initial alignment. Section 4.4 presents the implementation details of the point cloud registration. Section 4.5 presents the implementation details of final bounding box generation.

4.1 Workflow

This section presents the workflow of our method. For the sequence of point clouds $\boldsymbol{P} = \{P_1, P_2, ..., P_N\}$ input to our method, we sequentially register each point cloud onto the previous point clouds in the sequence. For instance, in iteration i, P_i is aligned to $\hat{P}_{1,i-1}$, where $\hat{P}_{1,i-1}$ is the aggregated point cloud that contains point clouds from P_1 to P_{i-1}. This results in an aggregated point cloud formed by aligning all point clouds. Subsequently, bounding boxes are generated based on this aggregated point cloud. Figure 4.1 illustrates the overall workflow.

doi:10.6342/NTU202403763

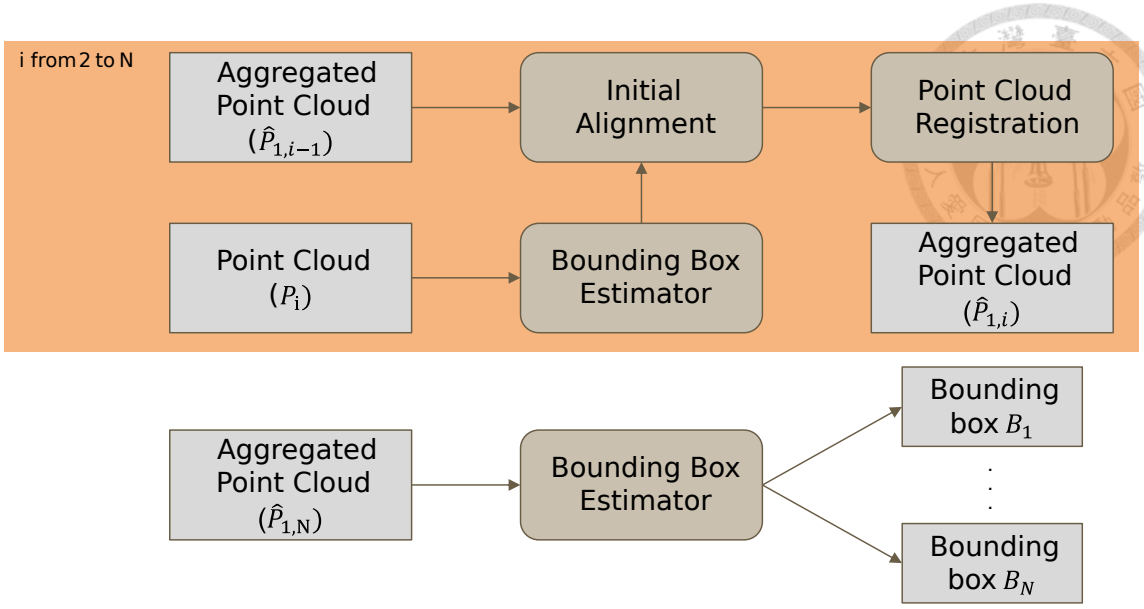


Figure 4.1: Workflow of our work.

4.2 Bounding Box Estimator

This section presents the implementation details of the bounding box estimator in the workflow. Bounding box estimator is responsible for generating rough bounding box $\hat{B}_i$ for the input point cloud P_i. The center, size, and heading of the rough bounding box are crucial for the subsequent point cloud alignment. L-shape fitting, proposed by [20], is adopted as the foundational algorithm for the bounding box estimator. L-shape fitting is a search-based method that estimates the bounding box of the vehicle's point cloud by leveraging the geometric contour of the vehicle. The algorithm first projects the point cloud onto the x-y plane to obtain 2D points. Then, it computes the minimal bounding boxes for all possible headings. Specifically, the algorithm computes the bouning boxes with heading θ ranging from 0° to 90° with a step size δ, where δ is set as 1° in our implementation. Once the bounding box is obtained, the algorithm will use certain metrics to select the best bounding box. The metric used in this work is variance of the bounding box. In order to compute the variance of the bounding box, the 2D points within the bounding

doi:10.6342/NTU202403763

box will be divided into two group according to whether the closest edge in the edges of the bounding box is parallel to the orientation or not. The algorithm then calculates the variance for each of these two groups and sums them to obtain the overall variance of the bounding box. Algorithm 1 and Algorithm 2 are the details of L-fitting algorithm. Because the bounding box's variation along the angle is irregular, search methods such as binary search cannot be used; instead, an exhaustive search of all angles is required. We also attempted to narrow the search range, but the reduced range did not guarantee an optimal solution, leading to incorrect bounding boxes.

While L-shape fitting can obtain the best bounding box of the vehicle's point cloud, it can be affected by points near the edges, such as side mirrors. Figure 4.2 illustrates an example of an incorrect bounding box affected by the side mirror. To address this issue, we utilize the RANSAC (RANdom SAmple Consensus) algorithm to correct the heading of the bounding box. Points are divided into four group according to the closest edge, and the RANSAC algorithm is applied to the group with the highest number of points to fit a line. The 2D bounding box is then computed using the slope of this line as the heading.

Algorithm 1: Search-Based BBox Fitting

Input: Points $P \in R^{n\times 3}$
Output: 2D Bounding box

$Q \leftarrow \emptyset$;
$\hat{P} \leftarrow$ ProjectOntoXYPlane(P);
for $\theta \leftarrow 0$ **to** $\pi/2$ **by** δ **do**
  $e_1 \leftarrow (\cos\theta, \sin\theta)$;
  $e_2 \leftarrow (-\sin\theta, \cos\theta)$;
  $C_1 \leftarrow \hat{P} \cdot {e_1}^T$;
  $C_2 \leftarrow \hat{P} \cdot {e_2}^T$;
  $q \leftarrow$ CalculateVariance(C_1, C_2);
  insert (b, q) into Q;
end
select b from Q with the smallest q;

doi:10.6342/NTU202403763

Algorithm 2: CalculateVariance

Input: C_1, C_2

Output: variance q

$c_1^{max} \leftarrow \max(C_1), c_1^{min} \leftarrow \min(C_1)$;

$c_2^{max} \leftarrow \max(C_2), c_2^{min} \leftarrow \min(C_2)$;

$D_1 \leftarrow \arg\min_{v \in \{c_1^{max} - C_1, C_1 - c_1^{min}\}} \|v\|$;

$D_2 \leftarrow \arg\min_{v \in \{c_2^{max} - C_2, C_2 - c_2^{min}\}} \|v\|$;

$E_1 \leftarrow \{D_{1(i)} | D_{1(i)} < D_{2(i)}\}$;

$E_2 \leftarrow \{D_{2(i)} | D_{2(i)} < D_{1(i)}\}$;

$q \leftarrow \text{variance}(E_1) + \text{variance}(E_2)$;

return q;

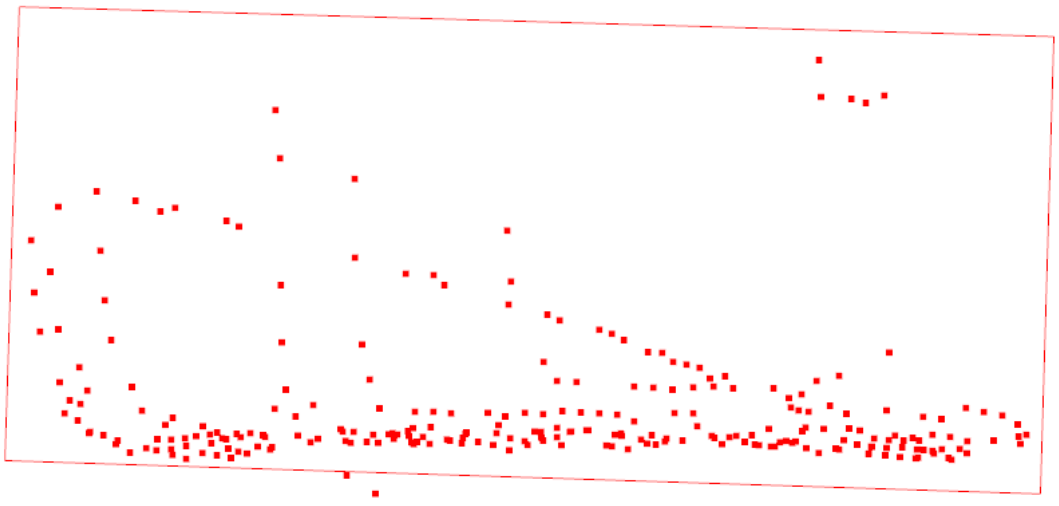


Figure 4.2: Incorrect Bounding Box Generated by L-shape Fitting.

doi:10.6342/NTU202403763

4.3 Initial Alignment

This section presents the implementation details of the initial alignment in the workflow. Before performing point cloud registration, we need to obtain an initial transformation matrix that roughly aligns the point cloud pair. Traditional methods using the feature of the point cloud to compute the initial transformation matrix. For instance, Fast Global Registration [15] using Fast Point Feature Histogram (FPFH) descriptors [14] to compute the matrix. However, these methods are generally designed for denser point clouds, whereas point clouds from vehicles are not consistently dense. After experiments, we found that traditional methods did not perform well on our data. Therefore, we have designed a new method to obtain the initial transformation matrix. This method leverages the rough bounding box generated from the bounding box estimator to compute the initial transformation matrix. A trivial approach would be to align using the center of the bounding box, but the center derived from partial point cloud may be offset, leading to alignment errors. Therefore, we choose to align using the corners of the bounding box. Compared to the center, the corners provide more accurate positioning, resulting in a more accurate initial transformation matrix. Figure 4.3 and Figure 4.4 illustrate the difference between center alignment and corner alignment. It can be seen that corner alignment yields noticeably better results.

In order to locate the corners used for alignment, we divide points into four groups based on their nearest corner, selecting the group with the most points as the alignment reference. Subsequently, based on the orientation of the bounding box, we determine the position of this corner within the bounding box, then identify the corresponding corner from the previous bounding box and use these two corners to compute the initial alignment

doi:10.6342/NTU202403763

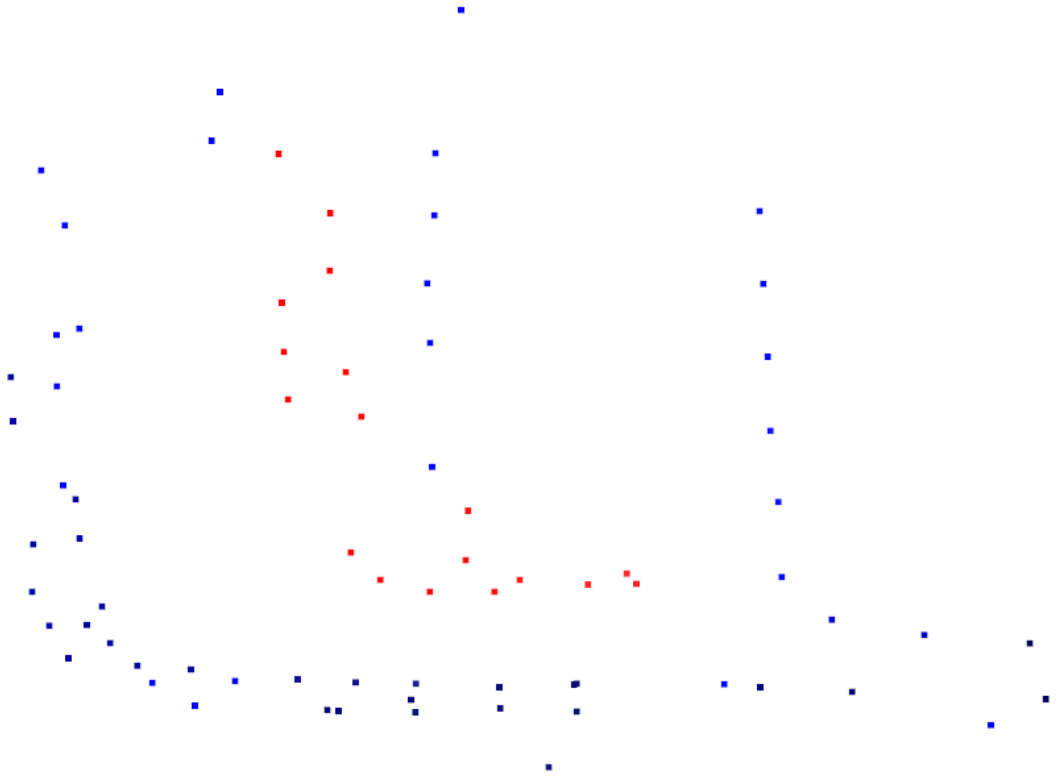


Figure 4.3: Center Alignment.

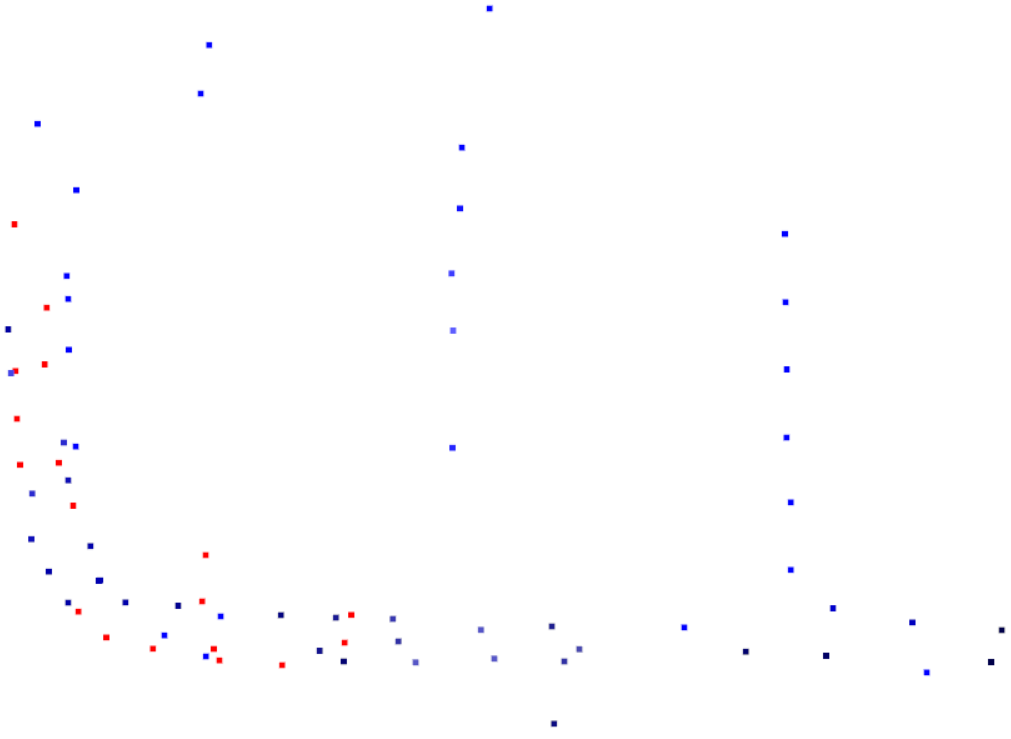


Figure 4.4: Corner Alignment.

doi:10.6342/NTU202403763

matrix.

4.4 Point Cloud Registration

This section presents the implementation details of the point cloud registration in the workflow. After initial alignment, point cloud registration is applied to gain more accurate alignment. Traditional methods like ICP (Iterative Closest Point) assume that the point pairs used for alignment are located at same positions on the object. However, due to the sparsity of LiDAR-collected point clouds, it is challenging to scan the exact same positions of an object at different times. Using a point-to-point calculation method can lead to some offset in the results.

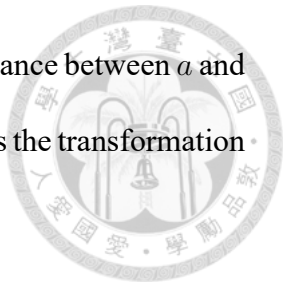
Although LiDAR cannot scan the exact same position in different time, it can scan the same plane. Therefore, we decided to use a plane-to-plane registration method. Plane-to-plane registration primarily utilizes planes constructed from nearby points for alignment. However, if points are located on corners or curved surfaces where surrounding points cannot form a plane, a point-to-point registration method becomes necessary. To achieve this goal, we designed a hybrid registration approach that combines point-to-point and plane-to-plane registration methods.

To switch between point-to-point and plane-to-plane alignment, we adopted GICP [13] as the framework for point cloud registration. GICP computes the error using the covariance matrix of points instead of point-to-point distance. The following equation represents how GICP calculates the error:

$$d^T(C_b + \mathbf{T}C_a\mathbf{T}^T)^{-1}d \tag{4.1}$$

doi:10.6342/NTU202403763

where a, b is the point pair for which the error is calculated. d is the distance between a and b, and C_a and C_b are the covariance matrices of a and b respectively, **T** is the transformation matrix that align a to b.



By using different covariance matrices, we can switch between point-to-point and plane-to-plane alignment methods. Here is how we compute the covariance matrix: first, we calculate the covariance matrix from a point and its nearest n points within a radius r. In this work, r is set to 0.4 meters and n is set to 300 points. Next, we perform the eigen decomposition on the covariance matrix. If the explained variance of the smallest eigenvalue exceeds 0.3, the output covariance matrix is **I**. If the explained variance of the smallest eigenvalue is less than 0.3, the output covariance matrix is $\mathbf{U}\mathbf{D}\mathbf{U}^\top$, where **U** consists of eigenvectors sorted in ascending order, and **D** is a diagonal matrix $\text{diag}(0.001, 1, 1)$.

4.5 Final Bounding Box Generation

This section presents the implementation details of final bounding box generation. When we obtain the aggregated point cloud $\hat{P}_{1,N}$. Bounding box estimator is used to generate the 3D bounding box. We use the same algorithm as in Section 4.2 to generate the 2D bounding box, and then use the maximum and minimum values along the z-axis of the points to generate the 3D bounding box.

doi:10.6342/NTU202403763

Chapter 5

Experiment Evaluation

This chapter presents the evaluation results of our work. In Section 5.1 introduces the evaluation metrics. Section 5.2 presents the quantitative results of the experiment. Section 5.3 presents the visualized examples of the results.

5.1 Evaluation Dataset

To evaluate the alignment performance of our work, the center and heading of vehicles' ground truth is required to compute the ground truth transformation matrix. However, existing datasets lack these information. Therefore, we decide to use synthetic dataset generated from Carla for evaluation. Carla is a simulation framework that can simulate real world traffic and generate LiDAR's point clouds. The settings of the LiDAR is shown in the Table 5.1. We use blueprints provided by Carla to generate vehicles and use autopilot feature to simulate real-world traffic. Figure 5.1 illustrates the point cloud generated from Carla.

The dataset generated from Carla contains 25 tracks of objects. The length of these tracks ranges approximately from 30 to 40 frames. The detailed distribution of track

doi:10.6342/NTU202403763

lengths is listed in the Table 5.2.

Frequency	10Hz
Channels	64
Points per Second	1200000
LiDAR Range	30 meters
Noise	N(0, 0.01) (meters)

Table 5.1: Settings of the LiDAR.

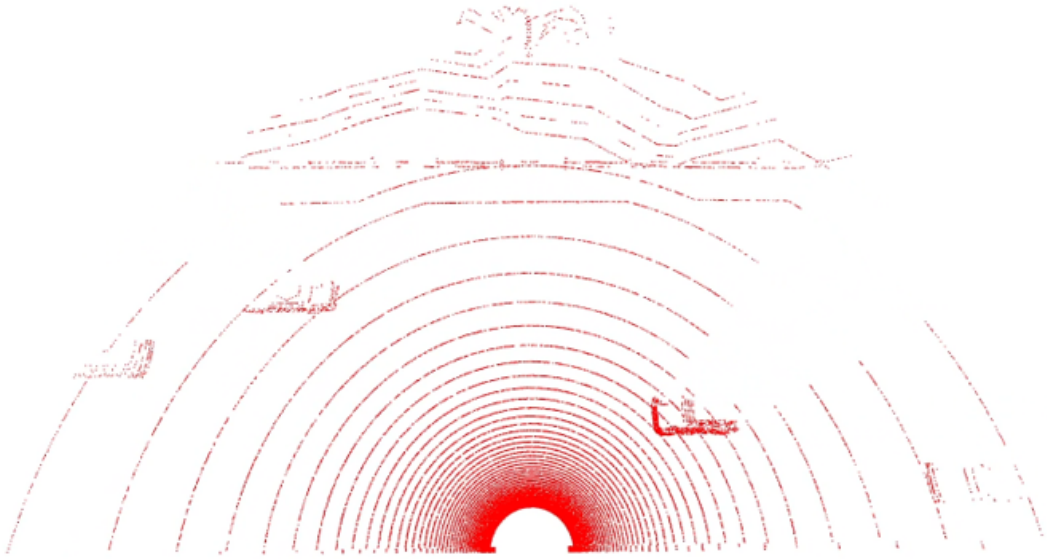


Figure 5.1: Point Cloud Generated from Carla.

Addtionly, we also use a WAYSIDE dataset that collect by the real-world roadside unit to evaluate our method's performance on real-world data. Due to the lack of the center and heading of vehicles' ground truth, we only evaluate the quality of the bounding box generated by our work on the WAYSIDE dataset.

The WAYSIDE dataset is collected by three roadside units named RSU_1, RSU_2, and RSU_3. We use the point cloud collected by RSU_1. The roadside unit consists of a VLP-32C LiDAR, a GPS, and three cameras. The height of the LiDAR is around 1.6 meters, same as the setting in Carla dataset. Figure 5.2 illustrates the roadside unit. Only the point cloud data collected by the LiDAR is used in our work. The data were collected at the

doi:10.6342/NTU202403763

Length of track (frames)	Number of tracks
$10 \sim 15$	1
$15 \sim 20$	1
$20 \sim 25$	0
$25 \sim 30$	3
$30 \sim 35$	7
$35 \sim 40$	7
$40 \sim 45$	2
$45 \sim 50$	1
$50 \sim 55$	1
$55 \sim 60$	0
$60 \sim 65$	0
$65 \sim 70$	0
$70 \sim 75$	0
$75 \sim 80$	1
$80 \sim 85$	0
$85 \sim 90$	0
$90 \sim 95$	0
$95 \sim 100$	1

Table 5.2: Distribution of Track Lengths.

intersection of JianGuo Road and QingJing Road in Pingtung county, Taiwan. Figure 5.3 illustrates the position of the roadside units. The data were recorded at 07:00, 10:00, 13:00, and 16:00 on March 8th, 2022, and March 9th, 2022, with each recording session lasting 2 hours.

The difference between the point clouds of the Carla dataset and the WAYSIDE dataset is that the point cloud from the Carla dataset is smoother than the point cloud from the WAYSIDE dataset. Specifically, in the case of windows, the WAYSIDE dataset exhibits unevenness due to light refraction, with some points passing through the window and entering the interior of the vehicle. In contrast, the windows in the Carla dataset are very smooth. Figure 5.4 and Figure 5.5 illustrates the difference between the Carla dataset and the WAYSIDE dataset.

doi:10.6342/NTU202403763

Figure 5.2: Roadside Unit that Collected WAYSIDE Dataset.

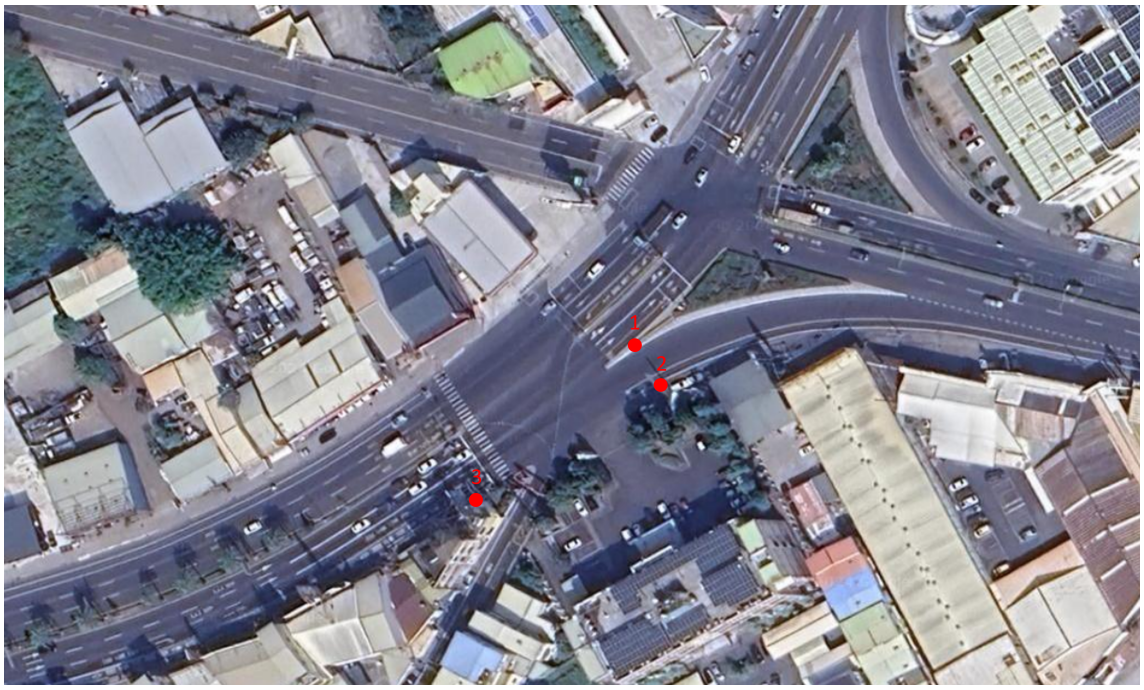


Figure 5.3: Position of Roadside Units.

doi:10.6342/NTU202403763

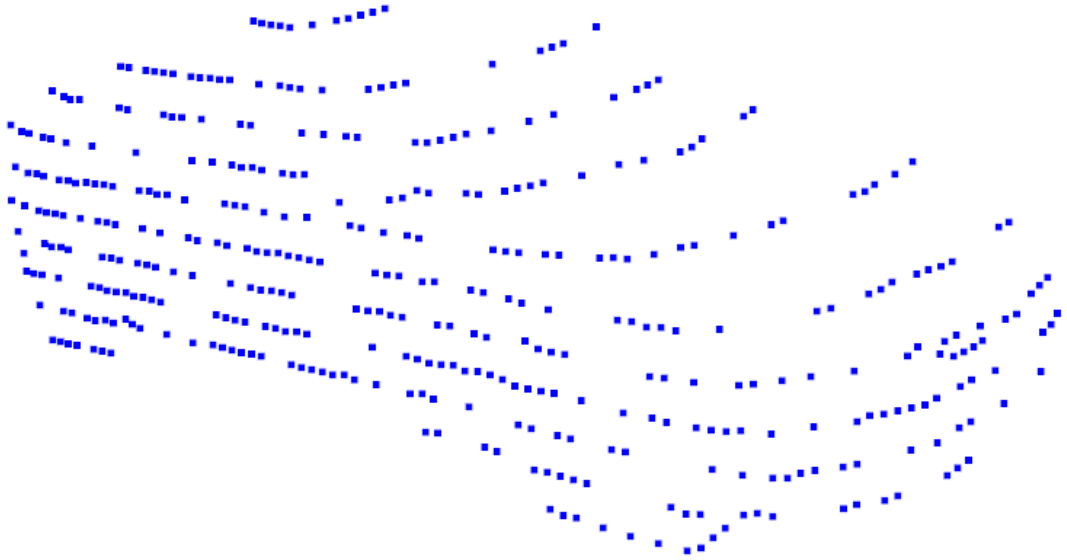


Figure 5.4: Vehicle's Point Cloud of Carla Dataset.

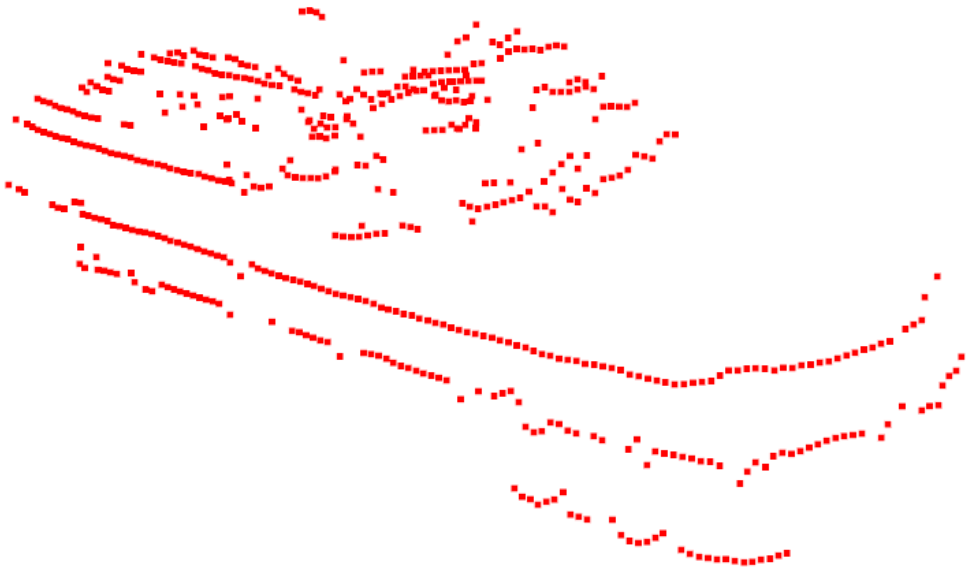


Figure 5.5: Vehicle's Point Cloud of WAYSIDE Dataset.

doi:10.6342/NTU202403763

5.2 Evaluation Metrics and Methodology

The goals of experiment is to evaluate the performance of point cloud alignment and the quality of the generated bounding boxes. To evaluate the performance of point cloud alignment, we use the translation error and rotation error, which are commonly use metrics in point cloud registration works. Translation error is the l2-norm of the difference between the estimate translation vector and the ground truth translation vector. The following is the equation of the translation error:

$$\left\|\hat{\mathbf{t}} - \mathbf{t}_{gt}\right\|_2 \tag{5.1}$$

where $\hat{\mathbf{t}}$ is the estimate translation vector and $\mathbf{t}_{gt}$ is the ground truth translation vector.

Rotation error is defined as the absolute difference between the estimated Euler angle of rotation matrix and the ground truth Euler angle of rotation matrix. The following is the equation of the rotation error:

$$\arccos\left(\frac{\text{trace}(\hat{R}R_{gt}^{-1}) - 1}{2}\right) \tag{5.2}$$

where $\hat{R}$ is the estimate rotation matrix and R_{gt} is the ground truth rotation matrix.

To evaluate the quality of the generated bounding boxes, we use the IoU between the generated bounding boxes and the ground truth bounding boxes. The following equation is the definition of the IoU:

$$IoU = \frac{\hat{B} \cap B^{gt}}{\hat{B} \cup B^{gt}} \tag{5.3}$$

where $\hat{B}$ is the generated bounding box and B_{gt} is the ground truth bounding box. We also use recall to evaluate the quality of the generated bounding boxes. If the IoU exceeds a

doi:10.6342/NTU202403763

certain threshold, the bounding box is considered as a true positive. We set the thresholds to 0.3, 0.5, and 0.7. The following equation is the definition of the recall:

$$Recall = \frac{\text{Number of True Positive Bounding Box}}{\text{Number of Bounding Box}} \tag{5.4}$$

5.3 Quantitative Results

This section presents the evaluation result of the translation error, rotation error, IoU, and the recall on the Carla dataset.

5.3.1 Translation Error and Rotation Error between Consecutive Frames

This section presents the translation error and rotation error between consecutive frames. We compare different point cloud registration methods by varying the registration methods within our multi-frame alignment framework. Our comparisons include point-to-point registration, plane-to-plane registration, DCP [17] and TEASER++ [16]. DCP is a deep learning method based on the features extract from neural networks and TEASER++ is a non deep learning method that utilize point features to alignment. Table 5.3 shows the result. We observed that feature-based methods (DCP and TEASER++) exhibit poor performance on our dataset. This is because these methods are designed to solve point cloud registration tasks on dense point clouds, whereas the point clouds from vehicles are too sparse to effectively extract features. The results also show that our hybrid method performs better than point-to-point registration (ICP) and plane-to-plane registration methods.

doi:10.6342/NTU202403763

Point Cloud Registraion Method	Translation Error (m), ↓	Rotation Error (°), ↓
ICP	0.10824	4.22297
Plane-to-Plane	0.16168	3.66189
DCP	1.46532	119.77314
TEASER++	2.05850	155.47393
Ours	0.08480	3.39233

Table 5.3: Translation Error and Rotation Error between Consecutive Frames.

5.3.2 Translation Error between Non-consecutive Frames

This section presents the translation error between non-consecutive frames. Because the point cloud alignment is performed frame by frame, there is a possibility of cumulative translation errors. We conducted experiments at intervals of 1, 4, and 9 frames respectively to investigate the error accumulation. Table 5.4 presents the experimental results, showing that the ICP method accumulates errors very quickly. At a 9-frame interval, the accumulation error is four times higher compared to consecutive frames. In contrast, our method shows a slower rate of increase in errors over intervals.

Frame intervals	0 Frame	1 Frame	4 Frames	9 Frames
ICP	0.10824	0.16392	0.28439	0.41801
Ours	0.08480	0.13064	0.20122	0.27776

Table 5.4: Translation Error between Non-consecutive Frames.

5.3.3 Cover Rate Versus Translation Error

TThis section primarily discusses the impact of cover rate on translation error. The cover rate is defined as the proportion of the ground truth point cloud that occupies the volume of the entire ground truth bounding box. Since point cloud registration requires the point cloud contains parts of the vehicle shape, we aim to investigate how the completeness

doi:10.6342/NTU202403763

of the point cloud affects registration.

Figure illustrates the relationship between cover rate and translation error. It shows that after a cover rate of 0.3, the translation error starts to decrease. Beyond a cover rate of 0.4, the translation error remains consistently below 0.1 meters.

5.3.4 Average IoU and Recall on Carla Dataset

This section presents the experimental results for average IoU and Recall on the Carla dataset. The experiment evaluates the performance of unsupervised object detection after multi-frame alignment. To assess the enhancement of point cloud alignment, we compare with the bounding boxes generated from single frame using L-shape fittng. Table 5.5 and Table 5.6 present the experimental results, indicating that the performance of alignment using ICP is worse than bounding boxes generated from single frame. This discrepancy arises because inaccurate alignment with ICP results in an aggregated point cloud shape that does not accurately represent the actual object, leading to poor IoU performance.

In contrast, our method demonstrates higher IoU compared to bounding boxes generated from single frame.

Registraion Method	Frames	Average IoU, ↑
Single Frame	1	0.629
ICP	Full Sequence	0.640
Ours	Full Sequence	0.629

Table 5.5: Average IoU on Carla Dataset.

doi:10.6342/NTU202403763

Registraion Method	Recall@IoU0.7, ↑	Recall@IoU0.5, ↑	Recall@IoU0.3, ↑
Single Frame	0.522	0.793	0.862
ICP	0.457	0.778	0.951
Ours	0.627	0.844	0.955

Table 5.6: Recall at Different Thresholds on Carla Dataset.

5.3.5 Average IoU and Recall on WAYSIDE Dataset

This section presents the experiment result of average IoU and Recall on WAYSIDE dataset. In this experiment, we also compare with the bounding boxes generated from single frame using L-shape fittng. Table 5.7 and Table 5.8 present the experimental results. The experimental reveals that our method significantly improved the Recall at an IoU of 0.5, increasing it by approximately 20%.

Registraion Method	Frames	Average IoU, ↑
Single Frame	1	0.479
Ours	Full Sequence	0.553

Table 5.7: Average IoU on WAYSIDE Dataset.

Registraion Method	Recall@IoU0.7, ↑	Recall@IoU0.5, ↑	Recall@IoU0.3, ↑
Single Frame	0.043	0.518	0.863
Ours	0.179	0.704	0.899

Table 5.8: Recall at Different Thresholds on WAYSIDE Dataset.

5.4 Time Complexity and Execution Time Analysis

This section analyzes the time complexity and execution time of each component in our workflow. The execution time analysis was performed on a machine with an AMD Ryzen 5 5600X CPU, using Python for implementation, with all matrix operations per-

doi:10.6342/NTU202403763

formed using the Numpy library.

5.4.1 Bounding Box Estimator

In this section, we will discuss the time complexity of Bounding Box Estimator with respect to the number of points n. The Bounding Box Estimator includes L-shape fitting and RANSAC. L-shape fitting calculates 90 bounding boxes, with each bounding box having a time complexity of $O(n)$. Therefore, the time complexity of L-shape fitting is $O(n)$. RANSAC has a time complexity of $O(n)$ per iteration, and since the number of iterations is fixed and does not depend on the number of points, the time complexity of RANSAC is also $O(n)$. Thus, the overall time complexity of the Bounding Box Estimator is $O(n)$. Figure 5.6 illustrates a graph of execution time of Bounding Box Estimator versus the number of points. The observed fluctuations in the graph are due to RANSAC initially dividing the points into four groups based on their distance to the bounding box, and then performing operations on the largest group. The size of this largest group may not be proportional to n, leading to greater variations in execution time.

5.4.2 Initial Alignment

In this section, we will discuss the time complexity of Initial Alignment with respect to the number of points n. Initial alignment requires finding the nearest bounding box corners for all points, so the time complexity is $O(n)$. In practice, we use the Python library NumPy to implement this, and NumPy completes the initial alignment for different numbers of points in less than 0.002 seconds.

doi:10.6342/NTU202403763

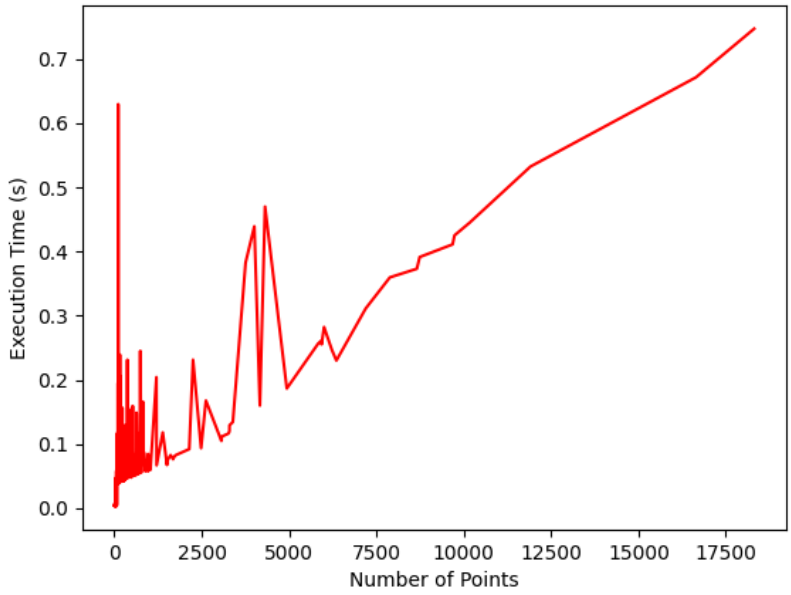


Figure 5.6: Execution Time of Bounding Box Estimator Versus the Number of Points.

5.4.3 Point Cloud Registraion

In this section, we will discuss the time complexity of Point Cloud Registration with respect to the number of points n. Point Cloud Registration includes covariance estimation and GICP (Generalized Iterative Closest Point). Covariance estimation involves finding the neighborhood of each point, which is done using a k-d tree. The time complexity of searching in a k-d tree is $O(\log n)$. Therefore, the time complexity of covariance estimation is $O(n\log n)$. In GICP, the point-related operation is finding the nearest point pairs between two point clouds in each iteration, which is also accomplished using a k-d tree. As a result, the time complexity of GICP is $O(n\log n)$. Thus, the overall time complexity of Point Cloud Registration is $O(n\log n)$. Figure 5.7 and Figure 5.8 illustrate the graphs of execution time versus the number of points for covariance estimation and GICP, respectively. For GICP, the number of points is the sum of the points in the two input point clouds. GICP execution stops when the error falls below a certain threshold. Some of

doi:10.6342/NTU202403763

the higher execution times in the graph are due to cases where the error did not decrease below the threshold, leading to a larger number of iterations.

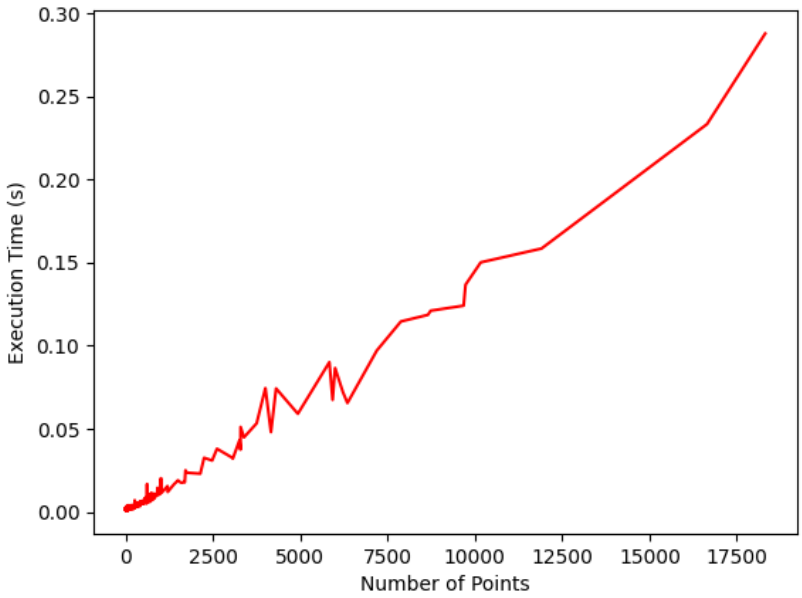


Figure 5.7: Execution Time of Covariance Estimate Versus the Number of Points.

5.5 Quantitative Results

This section visualized the results of the point cloud alignment and the generated bounding box. Figure 5.9 and Figure 5.10 show the aggregated point cloud aligned by our work. The redder points represent the point cloud in the earlier frames and yellower points represent the point cloud in the later frames. Figure 5.11 show the generated bounding box, where the blue points is the multi-frame alignment point cloud, red points is the point cloud from single frame, green bounding box is the ground truth bounding box, blue bounding box is the bounding box generated from our work, and red bounding box is the bounding box generated from single frame.

doi:10.6342/NTU202403763

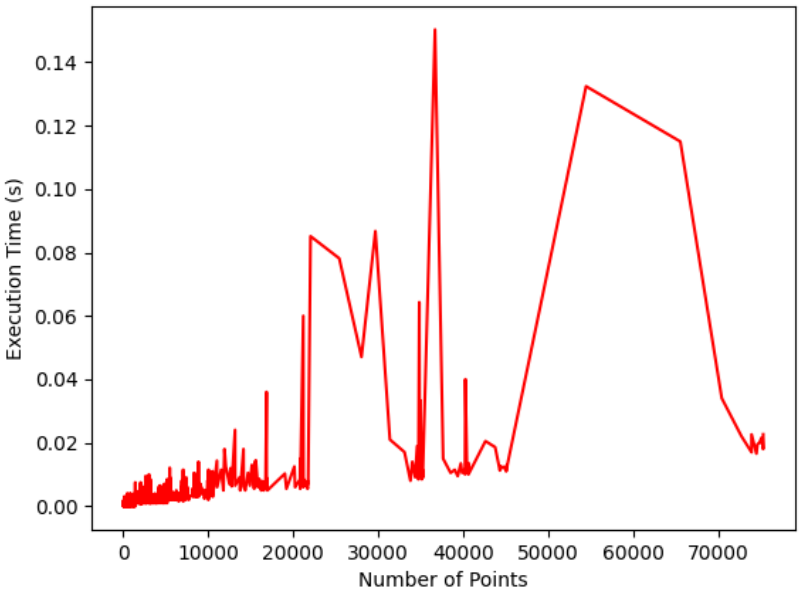


Figure 5.8: Execution Time of GICP Versus the Number of Points.

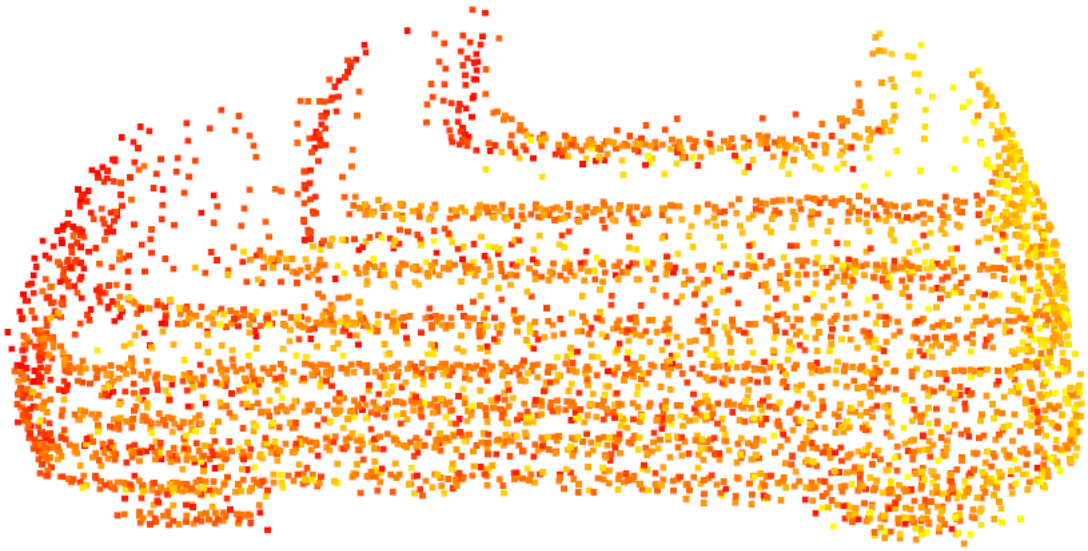


Figure 5.9: Aggregated Point Cloud.

doi:10.6342/NTU202403763

Figure 5.10: Aggregated Point Cloud.

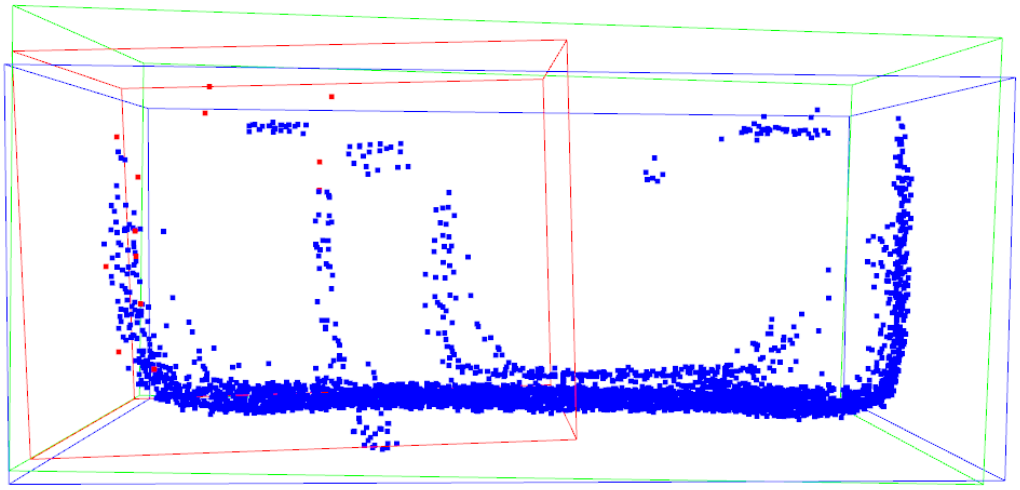


Figure 5.11: Bounding Box Generated from Our Work.

doi:10.6342/NTU202403763

Chapter 6

Conclusion

In this work, we propose a point cloud alignment pipeline that can align sparse vehicle point clouds without requiring any annotated data and generate bounding boxes from aggregated point cloud. Our pipeline uses the vehicle's contour for alignment, addressing the sparse point cloud alignment challenge that feature-based registration methods struggle to solve. The pipeline comprises a bounding box estimator for generating rough bounding boxes, initial alignment based on these rough bounding boxes, and point cloud registration combining point-to-point and plane-to-plane methods.

The experimental results show that our method improves the quality of bounding boxes. It achieves a 10% increase in recall at an IoU threshold of 0.7, and outperforms feature-based registration methods in terms of translation error and rotation error as well.

doi:10.6342/NTU202403763

References

[1] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 12 697–12 705.

[2] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4490–4499.

[3] Y. Yan, Y. Mao, and B. Li, "Second: Sparsely embedded convolutional detection," *Sensors*, vol. 18, no. 10, p. 3337, 2018.

[4] S. Shi, X. Wang, and H. Li, "Pointrcnn: 3d object proposal generation and detection from point cloud," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 770–779.

[5] W. Shi and R. Rajkumar, "Point-gnn: Graph neural network for 3d object detection in a point cloud," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 1711–1719.

[6] Z. Yang, Y. Sun, S. Liu, and J. Jia, "3dssd: Point-based 3d single stage object detector," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 040–11 048.

[7] F. Ma and S. Karaman, "Sparse-to-dense: Depth prediction from sparse depth samples and a single image," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 4796–4803.

[8] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, and H. Li, "Pv-rcnn: Point-voxel feature set abstraction for 3d object detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 10 529–10 538.

[9] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

doi:10.6342/NTU202403763

[10] Z. Yang, Y. Zhou, Z. Chen, and J. Ngiam, "3d-man: 3d multi-frame attention network for object detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 1863–1872.

[11] C. R. Qi, Y. Zhou, M. Najibi, P. Sun, K. Vo, B. Deng, and D. Anguelov, "Offboard 3d object detection from point cloud sequences," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 6134–6144.

[12] Y. Chen and G. Medioni, "Object modelling by registration of multiple range images," *Image and vision computing*, vol. 10, no. 3, pp. 145–155, 1992.

[13] A. Segal, D. Haehnel, and S. Thrun, "Generalized-icp." in *Robotics: science and systems*, vol. 2, no. 4. Seattle, WA, 2009, p. 435.

[14] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (fpfh) for 3d registration," in *2009 IEEE international conference on robotics and automation*. IEEE, 2009, pp. 3212–3217.

[15] Q.-Y. Zhou, J. Park, and V. Koltun, "Fast global registration," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*. Springer, 2016, pp. 766–782.

[16] H. Yang, J. Shi, and L. Carlone, "Teaser: Fast and certifiable point cloud registration," *IEEE Transactions on Robotics*, vol. 37, no. 2, pp. 314–333, 2020.

[17] Y. Wang and J. M. Solomon, "Deep closest point: Learning representations for point cloud registration," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 3523–3532.

[18] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," *ACM Transactions on Graphics (tog)*, vol. 38, no. 5, pp. 1–12, 2019.

[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[20] X. Zhang, W. Xu, C. Dong, and J. M. Dolan, "Efficient l-shape fitting for vehicle detection using laser scanners," in *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 54–59.

doi:10.6342/NTU202403763