Object Segmentation and Reconstruction Using Infrastructure Sensor Nodes for Autonomous Mobility

by

Yifeng Cao

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Mechanical and Mechatronics Engineering

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

This thesis focuses on the Lidar point cloud processing for the infrastructure sensor node that serves as the perception system for autonomous robots with general mobility in indoor applications. Compared with typical schemes mounting sensors on the robots, the method acquires data from infrastructure sensor nodes, providing a more comprehensive view of the environment, which benefits the robot's navigation. The number of sensors would not need to be increased even for multiple robots, significantly reducing costs. In addition, with a central perception system using the infrastructure sensor nodes navigating every robot, a more comprehensive understanding of the current environment and all the robots' locations can be obtained for the control and operation of the autonomous robots.

For a robot in the detection range of the sensor node, the sensor node can detect and segment obstacles in its driveable area and reconstruct the incomplete, sparse point cloud of objects upon their movement. The complete shape by the reconstruction benefits the localization and path planning which follows the perception part of the robot's system.

Considering the sparse Lidar data and the variety of object categories in the environment, a model-free scheme is selected for object segmentation. Point segmentation starts with background filtering. Considering the complexity of the indoor environment, a depth-matching-based background removal approach is first proposed. However, later tests imply that the method is adequate but not time-efficient. Therefore, based on the depth matching-based method, a process that only focuses on the drive-able area of the robot is proposed, and the computational complexity is significantly reduced. With optimization, the computation time for processing one frame of data can be greatly increased, from 0.2 second by the first approach to 0.01 second by the second approach. After background filtering, the remaining points for occurring objects are segmented as separate clusters using an object clustering algorithm.

With independent clusters of objects, an object tracking algorithm is followed to allocate the point clusters with IDs and arrange the clusters in a time sequence. With a stream of clusters for a specific object in a time sequence, point registration is deployed to aggregate the clusters into a complete shape. And as noticed during the experiment, one of the differences between indoor and outdoor environments is that contact between objects in the indoor environment is much more common. The objects in contact are likely to be segmented as a single cluster by the model-free clustering algorithm, which needs to be avoided in the reconstruction process. Therefore an improvement is made in the tracking algorithm when contact happens. The algorithms in this thesis have been experimentally evaluated and presented.

Acknowledgements

For those who instructed me, I would like to thank professor Khajepour, who gave me valuable instructions and advises during my project, and encourage me when problems occurred. I was having little progress in my first term for the project, but with professor's suggestions, I was able to have a better understanding for the project and my specific task. I would also like to thank professor Hashemi, who also gave instructions during the meeting, with his suggestions, many unnecessary literature review and attempts was avoided.

For those who work in the same project, I would also like to thank Neel, Naren and Soham, who collaborate with me to finish the whole perception system of the infrastructure sensor node, many problems for the whole project were solved during our discussions and collaborations, Naren is the one who came up with the idea of the first approach, though in later experiment I found it very slow. I would like to thank Minghao, who helped me over the ROS operations, which I was not familiar at first, and also helped a lot when conducting the experiment.

I would also like to thank Frank, Ruizeng, minghao, Jiaming, Yang and Keqi, who are also in the lab, when chatting with them about my project some unexpected good advises were heard.

Table of Contents

List of Figures

Chapter 1

Introduction

With the recent advancement in artificial intelligence and its wide application in robotics and autonomous vehicles, self-driving service robots have played various roles in human life, such as assistance, cleaning and delivering[3]. In the field of medical healthcare, robots can be applied to aid in clinical tasks, [37]support injuries or disabled[62][12].In the industrial field, transportation robots can be applied to transport spare parts or equipment between warehouses[61]. Several products of service robots have been developed; ABB developed Yumi with dual arms to work alongside staff by sorting test tubes, preparing medicines and handling liquids[36]. TUG is developed to help deliver supplies, medications, tests and wastes[35].

This project introduces a point cloud processing system using an infrastructure sensor node for self-driving robots. Different from schemes with mounted sensors, the infrastructure sensors can detect the environment with a much wider field of view. The sensors mounted on the robot can detect the environment with the same perspective as the robot, which can navigate the robot more intuitively. And infrastructure sensor will view the environment and the robot in a third-person perspective and navigate the robot remotely. The global understanding of the environment, which detects obstacles and predicts their motion in advance, could benefit the global and local path planning for the robot. Also, when multiple robots are deployed, the infrastructure sensor node could serve as the perception system for every robot entering the sensor node's detection field. The infrastructure perception module can reduce the number of sensors needed as the number of deployed robots increases, thus significantly reducing the cost of the robot system, as the camera, especially Lidar, has a relatively high price. Also, the perception using infrastructure sensor nodes for multiple robots could quickly locate and calculate the relative position between each

robot, hence optimizing the dispatching of various robots, which is a typical scene in the indoor environment.

However, there are also challenges for schemes with infrastructure sensor nodes. The mounted sensors on robots would always detect the surrounding environment needed for the robot's current navigation. In contrast, the infrastructure sensor would detect a relatively wider, global scene, with more necessary and unnecessary information about the environment included. Also, due to the characteristic of laser scanning, sensors mounted on the robot would detect nearby obstacles with denser point cloud data. The infrastructure sensor-node are relatively remote to the obstacles, compared to the mounted sensor scheme. As the characteristic of scanning laser, the scanning would scatter to a wider area as the distance grows, and the acquired data would be much sparser for roadside Lidar, which could be difficult to extract enough features for the objects' detection and classification. Also, though the roadside sensor has a global view of the environment, the case that objects occlude with each other may also happen. For the mounted sensor, though the occlusion case is much more common due to the sensor's field of view, the data surrounding the vehicle, which is needed to avoid the collision, can always be acquired. Although occlusion is a rare case for the infrastructure sensor for this scheme, the potential occlusion may omit the data nearby the vehicle needed for its navigation.

Multiple sensors are attached to the sensor node to address the challenges and fulfill the environment perception to navigate the robots. The Lidar would easily detect distance and locate the objects, while the camera would easily acquire the RGB data, which could provide rich features for objects' classification. The Lidar would detect the objects and segment them as clusters for the perception pipeline. The clusters of the same objects would be aggregated to complete the objects' shape. The camera would be applied to detect objects as bounding boxes and track them. The Lidar and camera detection results would be fused at the result level to complete the perception pipeline.

This thesis proposes the point cloud processing module of the perception system using an infrastructure sensor node for robots' navigation. The module would be able to detect moving objects in a stationary indoor environment. And by following an object tracking algorithm, the sparse point cloud of a certain would be aggregated, completing its shape as it moves. The thesis is composed of following parts:

The first chapter is an introduction to the project, which involves building a perception pipeline utilizing a camera and Lidar to navigate indoor robots from fixed infrastructure nodes. The Lidar system will use a model-free structure to detect and segment objects in 3D space as object clusters and can reconstruct the sparse point cloud of a specific object into a complete shape as it moves by point registration. The camera system would use

Yolo[58] to detect objects in 2D space as 2D bounding boxes and track their motion using Deep-SORT[8]. The Lidar and camera module's data would be fused at the result level.

The second chapter presents the background knowledge relevant to the project. The background review starts with object detection, the most fundamental task for robot and autonomous vehicle perception. Then the object tracking approaches for both image and point cloud are introduced. The camera fulfills the tracking task in the project's perception module, which is more robust and efficient than tracking by Lidar. However, the reconstruction needs a continuous frame of point clusters for a certain object. Therefore, point cloud tracking is also necessary for the project. Then the point cloud reconstruction methods are introduced, which the project needs the complete the shape from extremely sparse point cloud data. Also, such shape completion could benefit the localization of objects and also benefit the path tracking following perception. At the end of the chapter, Lidar camera fusion approaches are mentioned.

The third chapter presents the experiment setup process, which would introduce the hardware, the composition of the infrastructure sensor node, and the working of Lidar and the camera. Then the installation and working environment of the sensor node is introduced, followed by data collection for the perception system, including both point cloud data and image data, which would be used for point cloud segmentation, reconstruction, and image object detection.

The fourth chapter presents the point cloud segmentation section, and this part can be viewed as the main detection part for the Lidar processing, for every moving object could be detected and segmented as clusters of points. And this section could also be viewed as the pre-processing part for point cloud reconstruction, in which separate clusters for target objects are needed. The point cloud segmentation section starts with the background removal step, in which the points belonging to the static background are removed, and points of obstacles are left. The background removal step tried two approaches; the first approach is fulfilled by matching with a depth map and searching for obstacles that appear irrelevant to the objects' motion. It is able to obliterate background points but needs more time efficiency. The second approach focuses on the driveable area by Region Of Interest(ROI), which is effective and also time efficient. With optimization, the computation time for processing one frame of data can be greatly increased, from 0.2 second by the first approach to 0.01 second by the second approach. With ground points removed, the remaining object points are segmented as several independent clusters representing independent objects by the object clustering algorithm.

The fifth chapter presents the point cloud reconstruction section. Firstly the case for the project and the data will be discussed to prove the necessity of point cloud reconstruc-

tion. Secondly, the point cloud tracking method for the project is presented, global nearest neighbour(GNN) is applied for multi-object tracking, and continuous frames for particular objects can be obtained for their shape reconstruction. Then the point cloud reconstruction is applied to complete the objects' shape by aggregating frames of points. Considering the sparse character of the data, the reconstruction method should be multi-modal, combining point and plane information. Therefore generalized ICP(G-ICP) is applied as the reconstruction algorithm. Also, a particular condition in the project may affect reconstruction is studied. In the indoor environment, the objects' relative distances are closer, and objects' contact happens frequently. Objects are segmented as clusters for the model-free method based on their relative distance neglecting the shape feature. Therefore multiple objects' contact would result in a joint cluster, affecting the reconstruction. The solution is done by modifying the point cloud tracking algorithm and resetting an object's reconstruction as it comes into contact with other obstacles. An experiment is conducted to verify the optimized reconstruction process.

The sixth chapter presents the conclusion of the thesis's work and analyses future works which could optimize the current structure of the project.

Chapter 2

Background

2.1 Object Detection

Object detection can be the most important task of computer vision, which involves detecting and locating objects in an environment and estimating their corresponding bounding boxes Fig.2.1. Object detection has a wide application in autonomous driving, robotics, security and customer service, and also serves as the basis for higher-level computer vision problems such as object tracking, motion tracking, scene understanding, and semantic segmentation[44].

2.1.1 Image Object Detection

The most typical approaches for object detection are based on images from cameras that try to detect objects by extracting features among pixels.

Traditional methods use model-free structures, which try to segment objects or find region proposals using hand-crafted features that detect edges, corners or key points, then use a none-learning based approach to classify the detected objects by allocating a label among a given set of predefined categories such as support vector machine(SVM) or decision tree. Several typical algorithms of image processing are developed or applied in those model-free approaches. Based on the intuition of defining edges of changing intensity among pixels, Dalal proposed Histogram of Oriented Gradients(HOG)[21] for image feature description, in which each local region is described over a histogram counting the occurrence of gradient orientation. Mean Shift proposed by Comaniciu et al., is able to search and

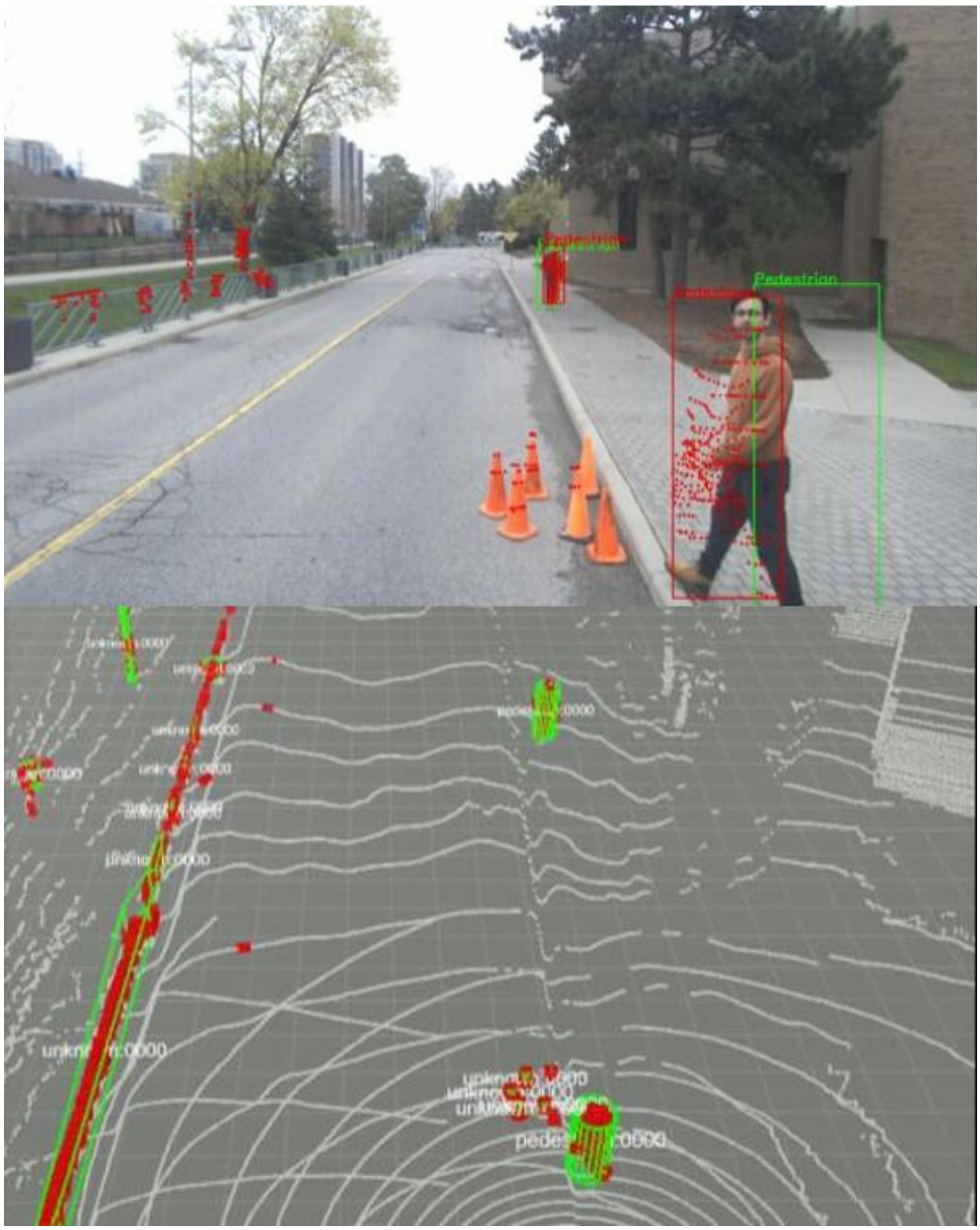

Figure 2.1: An illustration of object detection, including 2D, image-based object detection and 3D, point cloud-based object detection.

segment pixels or sparse data like points based on defined density[18], which is a common method for binary segmentation of objects in images in early approaches.

Wang et al. used a combined feature composed of Local Binary Pattern(LBP) and Histogram of Oriented Gradients(HOG) for human detection[75]. An occlusion likelihood map is built by features extracted by HOG to tackle the occlusion problem by identifying occluded regions. The method applied a two-stage structure, which is composed of a sliding scanning window global detector acquiring region proposal and a local detector classifying objects as human or miscellaneous by support vector machine(SVM) over the unoccluded regions. Lampert et al., also proposed a similar structure for detecting and localizing objects, which has a sliding window global detector and a local detector. However, different from former approaches, which are only able to perform binary classification, multi-class classification such as kernel SVM can be applied in their structure while preserving computational efficiency. In the methodology of Tuzel et al., further improvement of object classification was made by priority information of geometry space was incorporated for classifying human pixels based on Riemannian manifold[70].

With the development of deep learning, deep neural network structure have been applied to image object detection with more precise and complex feature extraction and description, which lead to improvement of time efficiency, accuracy and preciseness together with the vast improvement of the computational platform. One of the most efficient basic structures of deep learning in the image is the convolution neural network(CNN) which is shift-invariant and able to extract hierarchical features and patterns among neighbouring pixels, thus is more effective and efficient than simple MLP. He et al. added input of the former layer as a reference to the latter rather than simply a hierarchical connection among layers of a convolutional neural network, which is ResNet[31]. ResNet enables the increase of accuracy and complexity of learned features from deep neural networks with considerable depth increase, therefore is used as one of the most popular backbones of later CNN-based structures. Deep learning-based approaches to image detection can be categorized into two categories: sequential-based and end-to-end. The sequential-based model involves finding region proposals, and performing specific detection and classification of proposals. One of the most typical works with sequential-based structure is Fast-RCNN[28] which projects the original image and the extracted multiple generating a feature map, then convolutional layers and fully connected layers are followed to finish optimal classification and bounding box regression. As an enhancement of Fast-RCNN, Faster-RCNN proposed a region proposal network(RPN) which can simultaneously generate object scores and regress bounding boxes, which makes it possible for real-time detection using RCNN[60]. End-to-end methods tend to generate region proposals and detect objects in a single stage. One of the most typical works with end-to-end structure is YOLO[58]. Rather than taking a

step of generating region proposal like RCNN, YOLO starts with a number of generated bounding boxes scattered in the 2D image space and would regress to detected objects while generating detection scores in one time. YOLO reaches a balance between accuracy and real-time efficiency, thus is one of the most widely-used approaches, therefore serves as the basis of plenty of later works, which try to further increase the performance[65][59]. Similar anchor-based end-to-end structures can also be seen in algorithms for detection of specific target, LaneATT[69] use line-like anchor fitting detected lines in traffic lane detection.

2.1.2 Point Cloud Object Detection

The difference between image and point cloud object detection is from the data acquired by different sensors. Image-based methods receive data from RGB information among pixels acquired by cameras; point cloud data offers 3D geometric information that can provide valuable insights into objects' location, shape, and actual size. Point cloud data are normally acquired by Light Detection and Ranging Lidar, which emit laser pulses and measure the time the reflected signals return, thereby creating a 3D representation of the surrounding environment. Like image-based object detection, object detection approaches on point cloud can be classified as two categories: sequential-based methods and end-to-end methods[19][26].

Sequential-based methods of object detection mainly show a two-stage characteristic, which is generating proposals followed by a bounding box regression stage[7]. Early methods tend to use a model-free approach to generating object proposals, which is to remove the background and segment the remaining points as object clusters. Zermas et al. proposed a pipeline for point segmentation for outdoor autonomous vehicles, which includes ground point removal and object clustering[88]. The intuition of ground plane fitting(GPF) is as follows: 1) ground tends to have simple geometrical features as planes and surface normal vector vertical to x-y-plane; 2) ground points tend to have low height values. Similar to RANSAC, the mechanism to choose an initial set of points based on height. Besides ground filtering, they also propose an improved method of object clustering based on the mechanism of laser scanning; cylindrical coordinates represent the points, and the points are clustered based on their distance from neighbouring points and also their distance and direction to the Lidar. Wu et al. proposed a simple method for removing background for infrastructure Lidar, which is based on the intuition that while aggregating a sequence of frames, the density of static environment voxels will increase, and dynamic objects will not be influenced due to the position change[79]. Cui et al. proposed a detection pipeline based on the same consequence[20], which is first removing background using 3-D density

statistic filtering(3-D-DSF)[79], followed by DBSCAN to get object clusters, the classification of object cluster is done using decision tree-based method RF[81]. Borcs et al. select objects over a background based on height information[11], the selected object clusters are projected into multi directions as images based on their principle component, and the object classification is done over projected images. The sequential-based methods are usually less time efficient than end-to-end procedures, and the performance of each individual stage can limit the overall result.

There are also deep-learning-based sequential models, which use deep-learning models to generate region proposals and estimate bounding boxes over detected objects. Similar to R-CNN, Point R-CNN first generates region proposal, then features are developed over selected regions using 3D convolution and perform classification[42]. As an improvement of Point R-CNN, Fast Point R-CNN fuses coordinate features of points in initial predictions with extracted features from PointNet to refine the detection result[17].

Unlike sequential-based methods, end-to-end methods don't have typical two-stage characteristics, and tend to estimate 3d bounding boxes directly over extracted features using deep learning model[98]. Just like the 2D convolutional neural network(CNN) is one of the most important models for image object detection, early approaches of deep learning for point cloud processing take the intuition of directly using 3D convolution block for 3d point cloud, which has seen applications in detection and segmentation tasks[4][13][16]. However, 3D-CNN struggles in time efficiency due to the point cloud's sparsity, and the cubical computational complexity increase as the data is changed from 2D to 3D. Several optimizations have been done over 3D-CNN. Maturana et al. reduces the 3D-CNN task by integrating the occupancy grid over the x-y-plane[49]. Wang et al. optimize the convolution process by proposing novel octo-tree data structure[74]. Liu et al. utilize the sparse characteristic of point cloud and applied sparse convolution mechanism for 3D-CNN by reducing the redundancy of sparse space, the computational efficiency is significantly increased while preserving accuracy[43], such mechanism is also applied by Engelcke et al. to their detection pipeline[25].

Original deep-learning models for point clouds are also developed. PointNet is one of the pioneering deep learning models applied directly for point cloud, which uses shared MLP together with T-Net module to extract point and point-wise features while ensuring rotation and scale invariant[56]. VoteNet[54] uses feature learning backbones to generate seed points and deep hough voting to concatenate seed points as objects and cluster points for each object among the concatenated seed points. Point Pillars is a classic model for point cloud object detection[39]. The model converts point cloud in to a pseudo image by scattering points into a x-y-grid as a set of pillars, and learned features from the pillar set are encoded as a pseudo image which scatter features back to the locations of each

pillar. Then a backbone with a top-down network followed by an up-sampling network will concatenate features from multiple strides. Objects are detected with an SSD(single shot multiple detector)[45] detection head which 3d bounding boxes will also be fitted. Zhou et al. uses voxel to separate points with a voxel grid. The voxel space would be concatenated into a 4D tensor characterizing shape information. Then a following SSD as RPN(region proposal network) will perform object detection[97].

Besides 3D-based detection models, approaches are also trying to represent point clouds as 2D pseudo images. HVNet extract feature from multi-scale voxels to pseudo feature image, then perform object detection using detection head after feature encoding[87]. Like Point Pillars, BirdNet project point cloud into BEV and perform 2D object detection using Faster-RCNN; the 3D bounding box is generated over post-processing structure[34]. Yolo3D[5] tries to project the point cloud into two pseudo images, a depth map, and a BEV image. The model shows that the single-stage detector yolov2[59] can be modified into a 3d detection environment while ensuring its original real-time efficiency, but the precision of location may not be satisfactory.

2.2 Object Tracking

While object detection's task is detecting objects and estimating corresponding bounding boxes over each frame, object tracking aims to evaluate the position of selected objects by calibrating detection results of consecutive frames of images or point clouds. Therefore object tracking can be seen as a higher-level problem of object detection. Based on the specific goal, which is to track and follow a single, selected object during its movement or track multiple candidates of categories chosen in the environment, the task of object tracking can be categorized as single-object tracking(SOT) and multi-object tracking(MOT) Fig.2.2.

2.2.1 Image Object Tracking

Image-based tracking methods utilize the image detection result, which is usually represented in 2D space, which can be more chellenging to acquire the targets' position in the real-3D world. However, due to the rich information embedded in RGB images and the efficiency of image detection, the image-based object tracking method is still the most popular approach among tracking schemes.

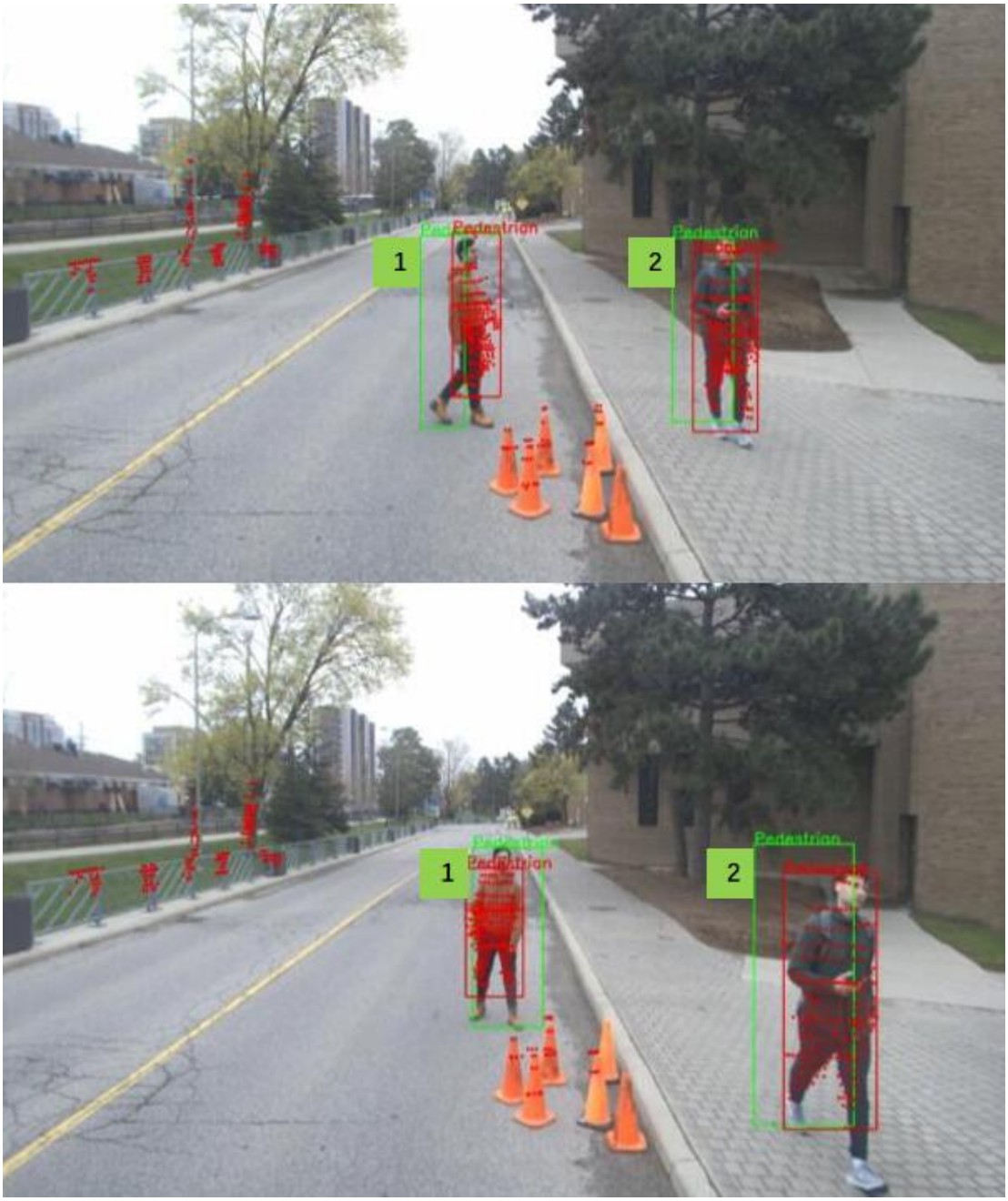

Figure 2.2: An illustration of multi-object tracking.

Model-free tracking algorithms tend to follow and track targets by estimating the motion information or size of the detection result. Noticing that the tracking effectiveness is greatly affected by the result of detection, SORT took the scale and ratio of the 2D bounding box as a part of the state vector for Kalman filter while estimating the motion of the 2D bounding box[8]. Though the intuition of SORT is simple, the performance is significantly improved compared to the standard Kalman filter, which only takes center point and velocity as state vectors. Zhou et al. try to simplify the task by facilitating the detection result[96], which only detects the objects as center points using CornerNet[40] while the scaling variance of bounding boxes can then be neglected. Noticing that low matching score due to occlusion result may be neglected, thus causing loss of tracking, ByteTrack[90] uses a two-stage structure which categorizes the tracking result as low-score and high-score tracklets. After the high-score target is matched, the low-score tracklets are then selected and sorted, which is efficient in tackling the case of occlusion.

Rather than directly matching with scaling or motion information, the feature-based approach try to use a combined score tracking targets by adding learned or hand-crafted features of detected targets as one of the targets matching elements. Ma et al. use a combined feature of HOG and IBP to perform detection tracking tasks both[47]. Dias et al. try to associate targets by extracted edges of objects and match the detected edges among feature space using RANSC[22], which the testing shows is efficient for detecting robots in complex, high-speed environments. Deep-SORT tries to enhance the association metric of SORT by adding deep appearance features extracted by a lightweight CNN-based model over the detected tracklets[77]. Aharon et al. further improved SORT by optimizing the Kalman filter state vector, adding camera motion compensation and combining motion information with extracted appearance information[2].

2.2.2 Point Cloud Object Tracking

Due to the disorder and sparsity of the point cloud, the task of object tracking on the point cloud differs from image object tracking, where many image-based methods cannot be applied to the point cloud directly[57]. The features used for tracking in the point cloud can be the motion information, bounding boxes or directly the point features.

For motion-based methods, Wu et al. proposed a new motion predictor via a constant acceleration model. The predicted position is also applied to enhance object detection and provide guidance for data association. Register pairwise object with a multi-model cost function including motion, appearance and geometric information[78].

Most approaches utilize the geometric feature of points or bounding boxes, Qi etc.

proposed P2B(Point to Box), which uses a template and search area of points to generate seed points. Project seed points with augmented features to potential targets via Hough Voting, and verify tracking target via potential clusters[57]. Notice that different instances of the same template may have an obvious difference in bounding box size but may have highly similar shape features. Hence based on P2B, box-aware features are added to enhance object tracking in the work of Zheng et al.[93]. Giancola Project selected object clusters into a K-dimensional vector and extracted features with a 1D and 2D CNN-based network. Besides tracking, process registered tracklet with shape completion to regularize training template. Shan et al. encodes th selected template and input point cloud and apply a transformer-based structure for point cloud object-tracking[66]. The PTT network comprises feature embedding, position encoding and a self-attention mechanism. Zhou et al. [95] also utilize a transformer to fulfill multi-object tracking on the point cloud. The self and co-relation-based attention mechanism will help with feature matching in finding the potential targets in the point cloud and pairwise matching in the object-tracking task. Pang et al. tried to match point features based on the idea of point registration[51] by registering shape information over continuous frame help for tracking tracklets(single object tracking). The work combines shape registration and motion estimation for single-object tracking. Based on WOD[68] data set, the work builds a new benchmark LIDAR-SOT as a by-product.

2.3 Point Cloud Object Registration

Point cloud registration is also one of the main tasks of computer vision. The scanned point cloud is usually sparse or incomplete to represent the whole object due to the scanning range and aspect. The task of point cloud registration is calculating the transformation relationship between point clouds[24], which can then be applied to combine two sets of points into a complete figure. Point cloud registration has seen applications in civil engineering, medical healthcare and autonomous driving, which can be used to reconstruct architectural structures, human organs or road maps and can also be applied for localizing autonomous vehicles.

For a point registration algorithm, the input should be two point clouds, and the output should be the transformation matrix and matching information. The point cloud registration methods can be classified into two categories: optimization-based method and learning-based[32].

The optimization-based method will view the problem as an optimization method, which will first search matching points between point clouds and use the optimization

method to estimate the best transformation relationship during iterations. One of the most famous and widely-used methods is ICP(Iterative Closest Point) and its variants. [89][23]. RANSAC(Random sample consensus) can also be applied as a point cloud registration method. Though RANSAC is a fitting algorithm, given select feature points and described features, point registration can also be viewed as a fitting problem: fitting the transformation matrix over the feature space. However, with selected feature points, RANSAC is time efficient but usually struggles in accuracy. This is usually used for global registration. With the coarse result by global registration as the input, ICP or other methods can be applied for local registration to get the optimal result. (NDT)Normal Distribution Transformation is also a popular method[10], NDT represents a point cloud as a piece-wise normal distribution. The transformation is calculated by maximizing the likelihood of two point clouds' distribution.

The optimization-based algorithms can guarantee their convergence with data of unknown categories by giving proper initial values but usually struggle with time efficiency. Also additional method is required to process noise and the initial solution.

Learning based method will use deep learning to either estimate the correspondence feature or directly evaluate the final solution of the transformation matrix[24]. Based on the structure, the point registration method based on deep learning can be classified into two categories, sequential-based method and end-to-end method.

The sequential-based methods will use a neural network to find specific feature points and encoded features to represent the point cloud in a more efficient way. Then the transformation matrix can be calculated over the feature points in the feature space.Gojcic etc. proposed a rotation invariant feature descriptor using a fully convolutional network which describes the point cloud with a density-based, voxelized representation[29]. Li et al. proposed USIP, which is an unsupervised feature point detector that can detect highly accurate and repeatable key points[41].

The End-to-end methods tend to view registration as a regression problem, which uses an end-to-end structure that directly solves the matching matrix or the transformation matrix between two point clouds. As an early attempt to solve point registration with an end-to-end form, Aoki accommodates PointNet and Lucas and Kanade (LK) algorithm in their proposed architecture[6]. Wang et al. proposed Deep Closest Points, which use a three-stage structure that generates matching information in a point cloud embedding, attention-based module and pointer structure sequence[76]. Yang et al. use a graph-based network that solves the problem of scaling, transformation and rotation in cascade. The truncated least square cost also enables robustness to the model[85].

2.4 Point Cloud-Image Fusion

Lidar and camera have different characteristics, in which Lidar can easily acquire depth information, while the camera can easily obtain rich information contained in RGB-images[19]. Thus camera-Lidar fusion algorithm, which utilizes the complementary characteristic of two types of sensors, can help understand the real-time 3d information of the surrounding environment for autonomous driving vehicles.

Depending on the stage of fusing point cloud and image data, the camera-Lidar fusion method can be divided into three categories: result-level fusion, feature-level fusion and multi-level fusion[52]. Result-level fusion combines the results of the separate camera and Lidar data processing pipelines to generate a final output. Feature-level fusion involves extracting specific features from camera and Lidar data and then fusing them. Multi-level fusion combines data at multiple abstraction levels, such as at pixel or object levels, to produce a more comprehensive and accurate output.

The intuition of result-level fusion is to combine the detection result of two detectors or use one of the detectors to guide the detection of another.

Frustum-PointNet is an early approach of combing a 2d detector with 3d detection[55], which generates a frustum over the detected 2d bounding box, and uses PointNet to detect selected objects in the segmented frustum space. However, the detection result is limited by the 2d detection result, which could lead to the omission of 3d detection when the 2d detector fails to detect a particular object. As a similar approach to Frustum PointNet, RoarNet also uses 2d proposal to guide 3d detection[67]. Compared to Frustum-PointNet, RoarNet can generate a more precise column-like region for 3d detection. Instead of PointNet, Paigwar et al. use PointPillars to find 3d proposal[50]. Noticing that the 2d bounding-box proposal could face occlusion problems in a crowded environment, Yang et al. use 2d semantic segmentation to achieve precise region proposal[86].

Rather than directly combining the multi-model detection result, feature-level fusion utilizes the extracted from the raw point cloud and image. Most feature-level approaches combine point cloud and image into image-like format data for further feature extraction. An early system would project a point cloud onto an image and detect objects in 2d space[30]. Valda et al. utilize features from rich representations in RGB images for semantic segmentation or point cloud by fusing image feature and depth feature of points[71]. 4D-Net also has similar intuition[53], which processes the point cloud into a pseudo image. The extracted features of the pseudo image and the real image are combined with multi-layer connections to ensure optimum detection results.

Multi-level fusion is a combination of feature-level fusion and result-level fusion. One

standard structure of such methods is using 2d detection to guide 3d detection, just like result-level fusion, while feature-level fusion is applied for 3d proposal generation. The intuition of such approaches is to use the detection result of a single model to accelerate or optimize the outcome of feature-level fusion. Point-Fusion[84] and SIFR-Net[92] all have such structures, with the selected frustum generated by 2d detector, PointNet[56] or PointSIFT[33] is applied in later feature-level fusion for generating the final 3d bounding box.

Chapter 3

Infrastructure Sensor Node

This chapter explains the structure of the infrastructure sensor node, and details of the Lidar and camera sensors used for perception. Then the setup of sensor node is introduced, with established sensor node, data was collected for Lidar processing and transfer learning for image perception.

3.1 Sensor Structure

Unlike typical schemes of robots' perception systems, the sensor is placed in a fixed place as an infrastructure sensor node rather than mounted directly on the robot. Lidar scans objects and acquires data as point clouds containing spatial information, which is convenient for detecting and locating objects in 3D space. The camera obtains images which contain rich RGB information that is easy to identify target objects in 2D space. The perception system would use a multi-sensor scheme: a sensor node comprises a Lidar and a wide-angle camera.

The Lidar is placed above the ground and scan the entire environment top-down. Therefore a wide scanning range is necessary when selecting the Lidar. Thus the Lidar used in the project is RS-Bpearl from Robosense, which has a Hemispherical detection field and a narrow blind spot designed for short-range detection, which is suitable for the narrow indoor environment in the project. The Lidar has a 360° horizontal and a 90° vertical field of view, with 32 lines of laser scanning at $10Hz$ and a maximum range of $100m$. The Lidar sensor is shown in Fig.3.1.

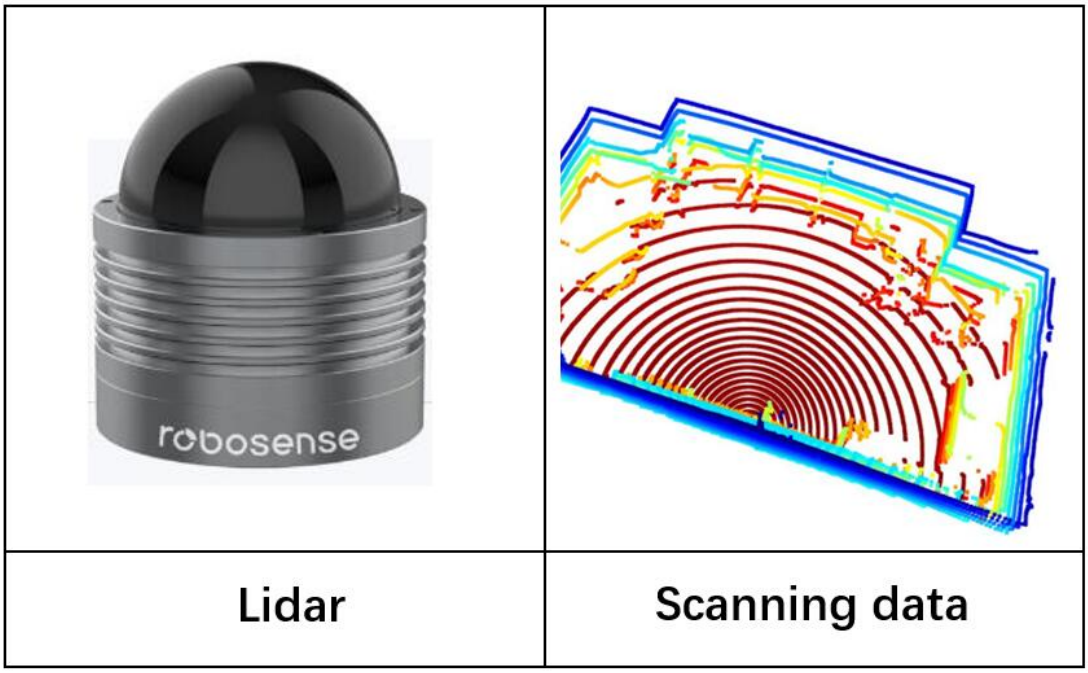

Figure 3.1: The Lidar used in the project and the collected point cloud data.

The camera used in the project is Basler daA1920-160uc camera with a compact and wide field view Fig.3.2. The camera and Lidar are attached closely to the sensor node to acquire data from similar aspects for fusing data from two separate sensors. The sensor node has a shelf-like structure which is easy to attach and adjust Fig.3.3.

3.2 Data Collection

The sensor is placed over the ceiling rather than the side walls for a wider detection field of view. A higher position means a wider range for laser scanning and camera detection, and it is possible to be placed in the middle of a room. Data is acquired after the sensor is fixed. Firstly the raw data of the scanned environment is obtained Fig.3.4. The sensor was attached to two separate rooms to get two groups of data. The raw point cloud data would be used to test the point cloud processing, and streams of object clusters can also be obtained from the raw data for later testing the effect of point cloud reconstruction. A chair

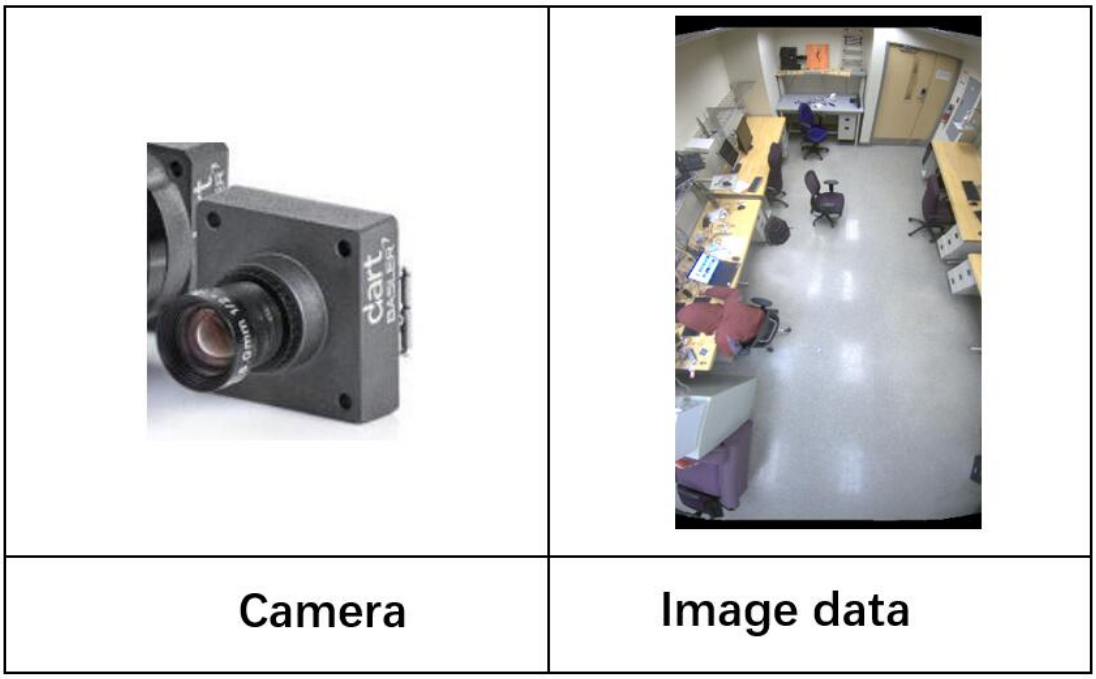

Figure 3.2: The camera used in the project and the collected image data.

is pushed across Lidar's field of view and recorded as the data for object reconstruction. The motion of pedestrians during the data collection is also recorded.

Raw image data can be applied to test the object detection result of YOLOV4 and the object tracking result of Deep-SORT. The original training data used to train YOLO had a horizontal aspect rather than the top-down aspect of the project. Therefore, the difference in aspect may lead to different features extracted and further lead to the omission of detection. To further train the existing model, data of humans and chairs with different angles is collected with the attached infrastructure sensor node for the transfer learning of YOLO.

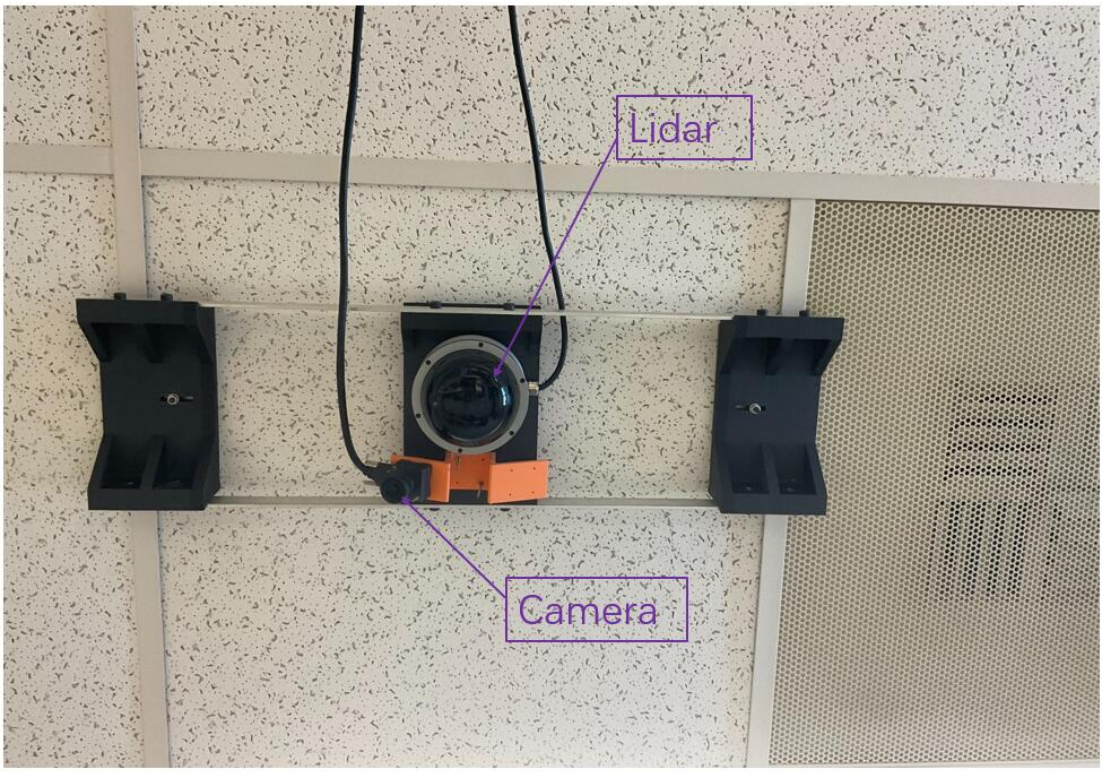

Figure 3.3: The attached infrastructure sensor node contains the Lidar and the camera.

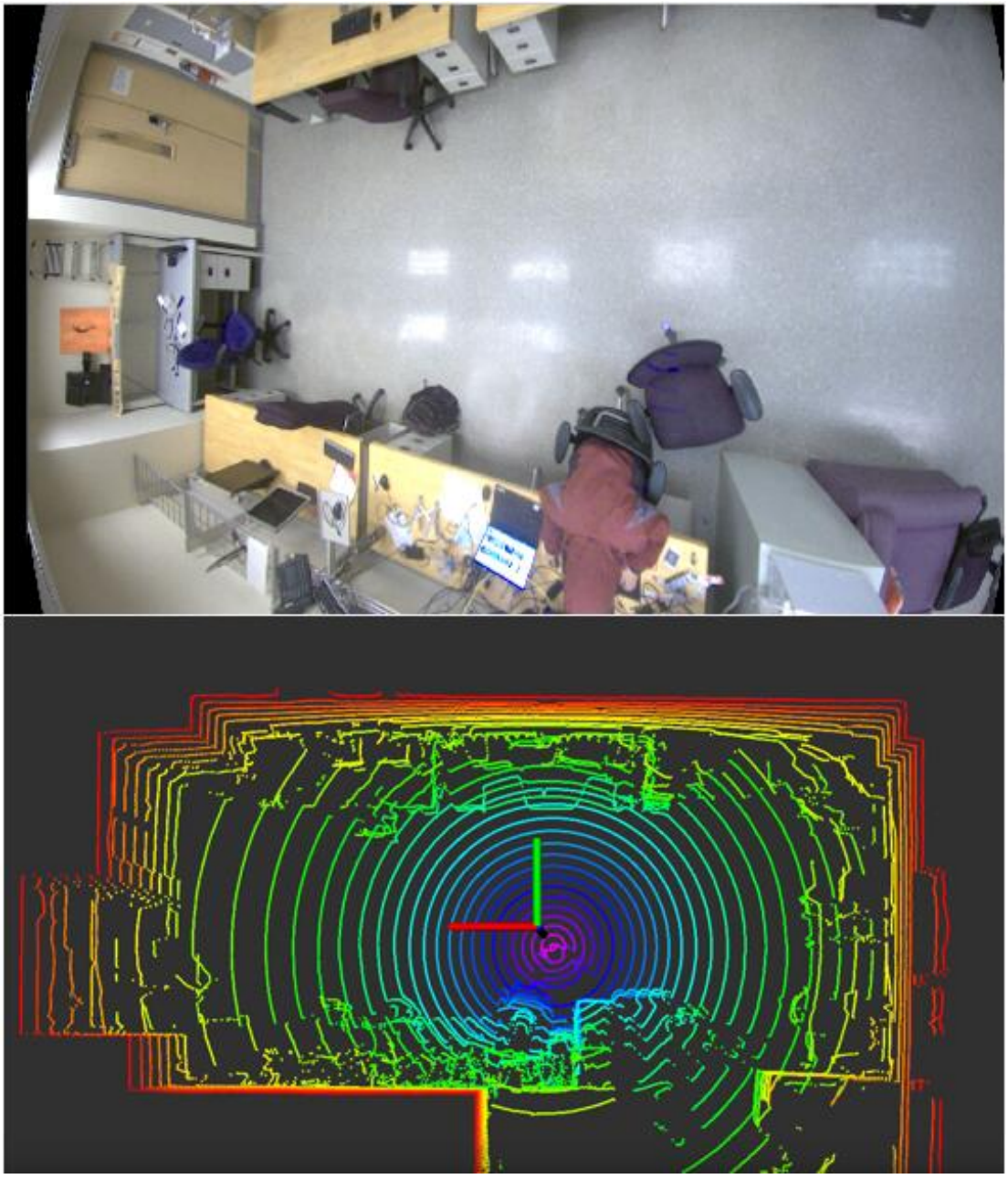

Figure 3.4: The raw sensor data of point cloud and image viewed on RViz.

Chapter 4

Point Cloud Object Segmentation

4.1 Motivation

This chapter presents the point cloud segmentation part of the infrastructure sensor node, in which the input is the raw point cloud and the output is the segmented object clusters. Considering the complex background in the indoor environment, the point segmentation will be established over a model-free structure with a background removal and object clustering sequence.

4.2 Background Removal

The background removal method should filter out irrelevant points and leave points of road users' remains. Raw point cloud data acquired from the project's environment include floor, walls, doors, furniture as desks, and shelf with objects on them and moving objects over the driveable area. Hence the indoor environment is much more complex than the outdoor environment, which has mainly ground points. RANSAC can be used to fit ground plane, which is helpful for background removal in outdoor environment[73][83][1], but due to there being walls, desks which have multiple planes and also non-plane objects in the background, such approach is not suitable for the project. Wu et al. used the frame aggregation method to filter the static environment and remain moving objects[80]. Still, the method must fulfill the projects' real-time performance or simultaneously detect dynamic and stationary objects.

4.2.1 Depth Matching-Based Background Removal

Two approaches have been proposed and tested over background removal for the project.

The first approach is based on the information from the depth map. The intuition of this approach is simple: given an empty frame of point cloud with the environment, if an object occurs, then the maximum height of points in its corresponding position will be increased.

The detail of the first approach is as follows:

(1)Given an empty frame of the point cloud with only the background as the reference frame and the current frame of the point cloud as the target frame, normalize the point clouds.

(2)Voxelize the point clouds, and project them into the x-y-plane in two depths images. The value of each pixel is the maximum height of points in the voxelized pillar.

(3)Match the target depth image with the reference image for a given pixel. If the distance of depth value is more significant than a threshold, then the corresponding position in this pixel occurs as an object. Matching two depth images could acquire pixel indices of occurred objects.

(4)The segmented points belonging to the objects can be obtained using the matching indices. For a given pixel with an object occurring, the object points should be found, as demonstrated in Figure Fig.4.1:

Therefore by this method, an empty frame of the environment is needed as the reference frame, as shown in Figure Fig.4.2

Given a reference frame and target frame, the background removal result is as Fig.4.3:

The result shows that the method can successfully remove the background with the reference frame. However, it is also noticed during the experiment the technique needs to improve its time efficiency. The Lidar scanning is at a frequency of 10Hz, which is 0.1s for each new frame. However, the background removal method takes 0.1 to 0.2s to process a single frame of a point cloud, which makes the whole time unable to catch up with the task of real-time perception. One of the reasons is that it is computationally expensive to convert the point clouds to depth images and recover the object points using matching results; also, the method process the non-driveable area of the environment, which could be more efficient.

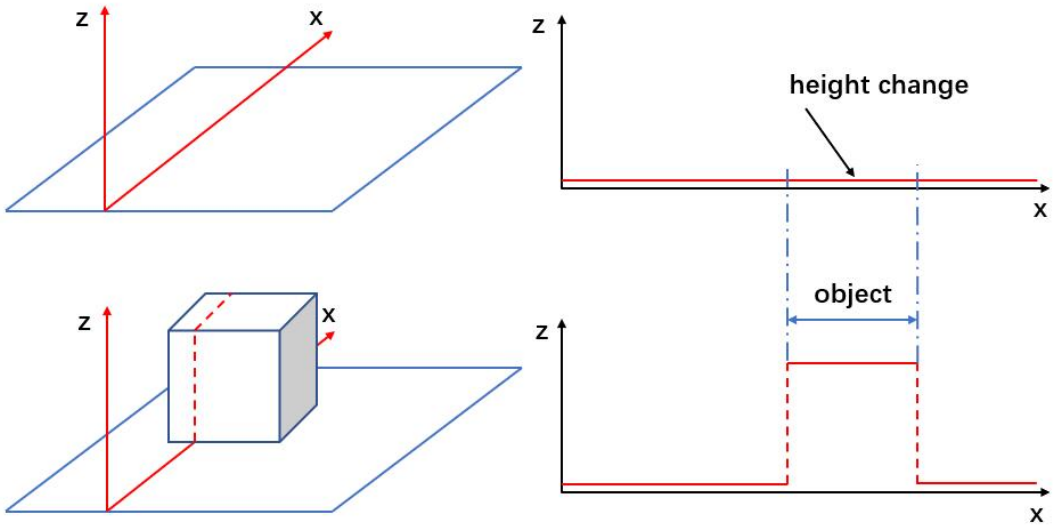

Figure 4.1: Given an empty frame of the environment as the reference, when an object occurs, it can be detected by a height change by comparing it with the reference frame.

4.2.2 Region Selection-Based Background Removal

Due to the depth-matching-based background removal approach is not able to fulfil real-time perception for the project, the second approach is proposed and tested. It was noticed that for the depth-matching-based method, which matches the two point clouds as two pseudo images over the x-y-plane, the most matching area belongs to static obstacles like desks or shelves, which belongs to the non-driveable area. However, the objective of removing the background is to leave objects in the driveable site remaining. Hence, the second method starts with a region of interest, focusing only on the driveable area.

The intuition of the region selection-based background filtering is simple, the objects that need to be detected are only in the drive-able area for the indoor robot, and the background in the drive-able area is only ground floor. And rather than trying to fit the ground plane using RANSAC, the ground point can be seen as points with heights lower than a certain threshold. Hence the details of the second approach are as follows:

(1) Draw a measured polygon representing the driveable area and segment points in the polygon region.

(2) With segmented point cloud from step (1), filter out points with z value lower than threshold τ_2

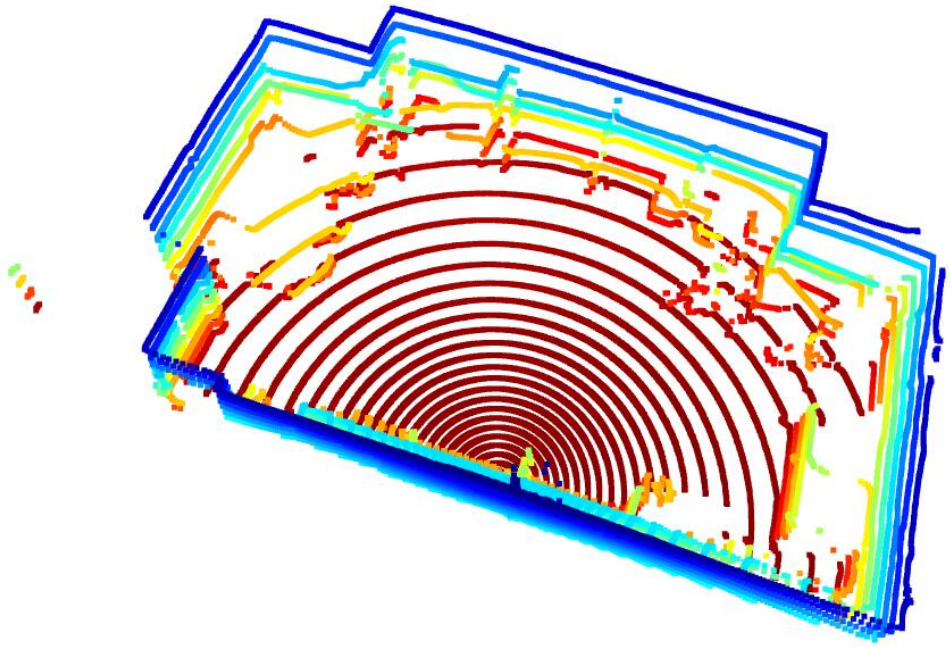

Figure 4.2: An empty frame of the environment with no moving obstacles is to be detected and used as the reference frame.

Given a frame of raw point cloud, the filtered points in the drive-able area segmented by ROI polygon is as Fig.4.4:

With points in the driveable area, object points are segmented by removing ground points which can be taken as points with a height lower than a threshold.

The background removal shows that the background can be successfully removed with only the objects remaining. At the same time, the time consumed for processing one frame takes 0.0011-0.0013 seconds, which means the time efficiency for the second approach is enough for the system's real-time performance.

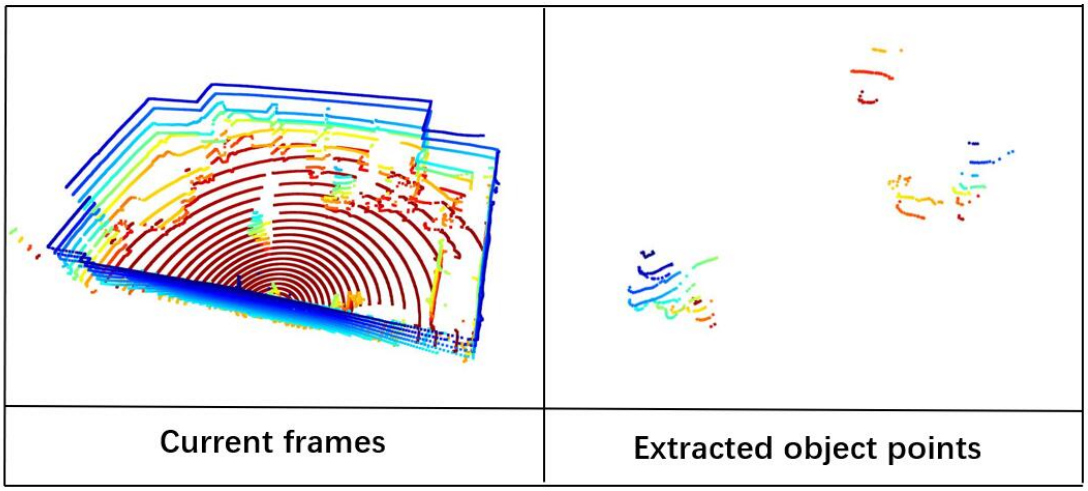

Figure 4.3: The background removal result uses depth-matching, with only object points remaining.

4.3 Point Clustering

After removing the background, the remaining points would belong to the occurred objects. The next step is to segment the remaining points based on the existing object targets. Due to the sparsity and incomplete shape of the data over the scanned objects, and also the uncertainty of occurring object categories and scales in the future, the actual environment for the project, the model-based approaches may struggle to obtain valuable features to detect and segment objects, a model-free method of segmentation would be applied to separate the remaining points into individual point clusters. Unlike outdoor road environments, where the objects are mainly pedestrians and vehicles, the object types indoors may be much more complicated. Still, model-based approaches may struggle to segment unfamiliar objects. Hence a model-free method is applied in this section.

The clustering algorithm is applied to segment the remaining points into object candidates. As the number of objects and their shape is unknown, we use DBSCAN(Density-Based Spatial Clustering of Applications with Noise) as the clustering method, which is also a standard method in existing autonomous driving schemes[9][48][14].

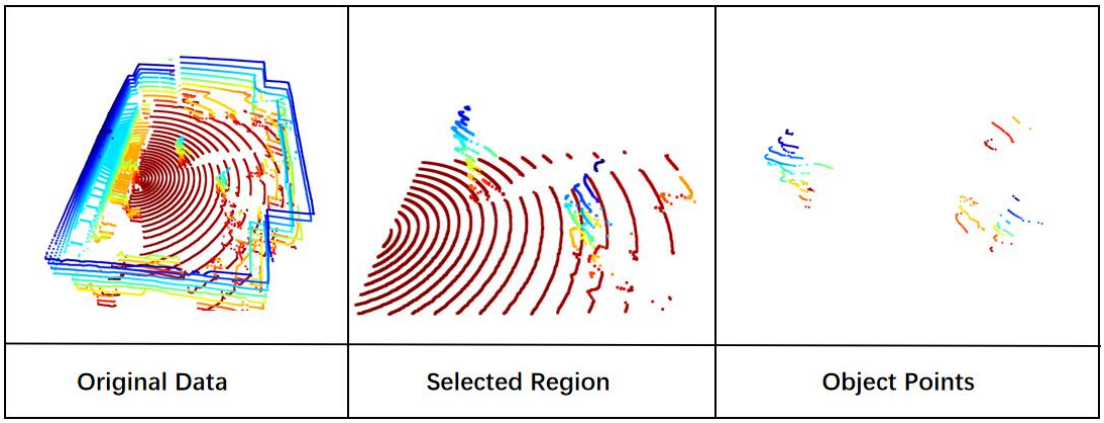

Figure 4.4: The drawn polygon can extract the points in the driveable area. The figure shown is an empty frame with no obstacles.

Given a selected point in a point cloud, the segmentation of DBSCAN is done over the density distribution of the point and its neighbouring points. The density can be described by relative distance between points. The process of DBSCAN generating a cluster in the point cloud is as follows:

(1) Randomly picking a point in the point cloud p_i.

(2) Given a searching radius ϵ and minimum number threshold m, if the number of neighbour points of p_i in radius ϵ is larger than m, then the selected point p_i and its neighbour points would be considered as a cluster.

(3) Randomly select a new point in the cluster, and expand the cluster by repeating step(2).

The input of this step would be the remaining points belonging to the object, and the output would be the clusters for each potential object. Given a specific point cloud frame with only extracted object points, the clustering result is shown as Fig.4.5. To demonstrate that the objects are segmented into several clusters, 3D bounding boxes are generated over the separate clusters. The result shows objects that appeared can be successfully segmented after completely removing the background.

In similar works, it has been observed that with the distance of objects from the infrastructure sensor node[72][15], the sparsity of acquired objects points increase significantly that DBSCAN with a fixed distance threshold would possibly ignore an object when it is far enough from the Lidar and become sparse enough. Thus, they defined the distance threshold as a function over the object's distance[91]. During the experiment, such a case

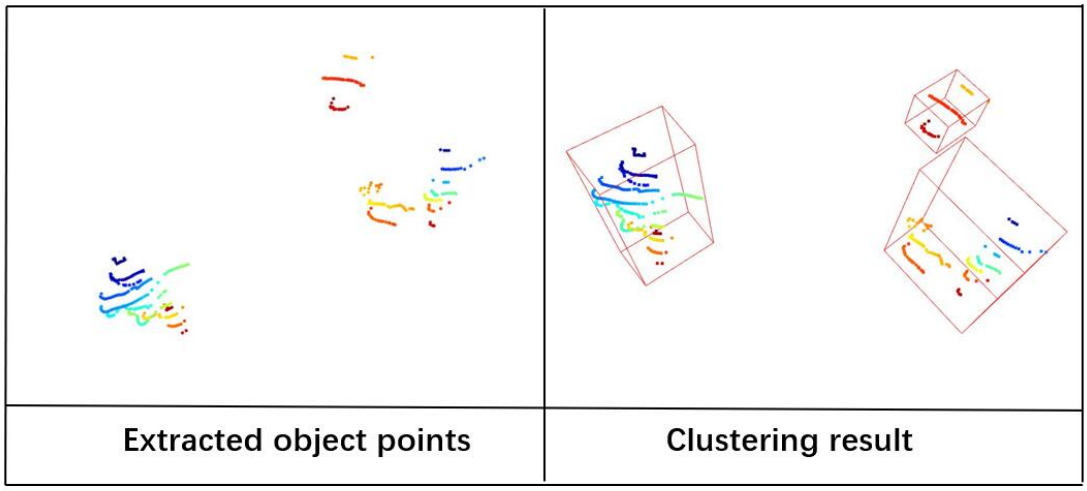

Figure 4.5: The result of DBSCAN clustering that segments points into separate clusters. For demonstration, 3D bounding boxes are generated over each cluster

that a sparse object is ignored at a certain distance is not observed. One possible reason is that in the outdoor environment, the Lidar is mounted on a high pole position to ensure the maximum field of view for the narrow indoor environment, where the Lidar can only mount on the ceiling with a low height. Also, the sensor node is composed of a Lidar and a wide-angle camera. Thus, the real detection range of a sensor node should be the field that both sensors can detect on, while the camera has a significantly smaller detection range compared to the Lidar, for the camera is at the same position as the Lidar while detecting the lower space over a frustum field. Therefore the detection range of the sensor node in this project is limited to a relatively narrow rectangle space. So the sparsity change in the project would be much smaller than their work.

During the experiment, it is also noticed that there happens is a significant density increase of points right under the Lidar due to its hemispherical scanning structure. Therefore, a significant lag of segmentation is observed when the object is approaching areas under the sensor node, so it is necessary to average the distribution of the point clusters by voxel-down sampling. With down-sampling added, the raw point cloud would be sparser

but more evenly distributed, and the computation time of background removal and object clustering is also accelerated. Therefore, the lag of time is not observed. The sparsity problem of the data is that only some lines of scanning are scanned upon objects. Thus, the sparsity problem mainly occurs over the $z$axis. The down-sampling size of the voxel is smaller than the gap distance of the scanning lines. Therefore, the voxel-down sampling would not decrease the existing features from the raw, sparse point cloud.

Chapter 5

Point Cloud Object Reconstruction

5.1 Motivation

When the Lidar is mounted on the robot, the sensor moves with the robot and provides a real-time view of the robot's immediate surroundings. The robot's field of view is limited by the location of the Lidar sensor, which can lead to blind spots and incomplete maps of the environment. However, it also allows the robot to adapt to changing environments and detect nearby obstacles.

In contrast, the sensors on an infrastructure node could provide a more comprehensive view of the environment, covering a larger area and allowing for more accurate mapping of the surroundings, which can be particularly helpful for autonomous vehicles. Therefore the method with an infrastructure sensor node is used in the project.

Due to the characteristic of laser scanning, the point cloud of a certain object is incomplete, with the occluded part from Lidar missing. Also, the scanned result is usually sparse, which is particularly obvious in this project, in which the sensor node has a 32-line Lidar, with $360° * 90°$ field of view, mounted on the ceiling of the room, only 5-6 lines will be scanned on tall objects like standing pedestrians, and only 3-4 lines will be observed on objects with low height like chairs. The scanned result of objects is so sparse that their shape and occupied space can hardly be determined, as demonstrated in Fig.5.1.

There are also other differences between the mounted sensors and the infrastructure sensors. For most perception schemes for autonomous driving, sensors are mounted on the autonomous robot or vehicle. In these schemes, the detected obstacles that can potentially collide with the vehicle can always be detected, for the visual field of the sensor is the same

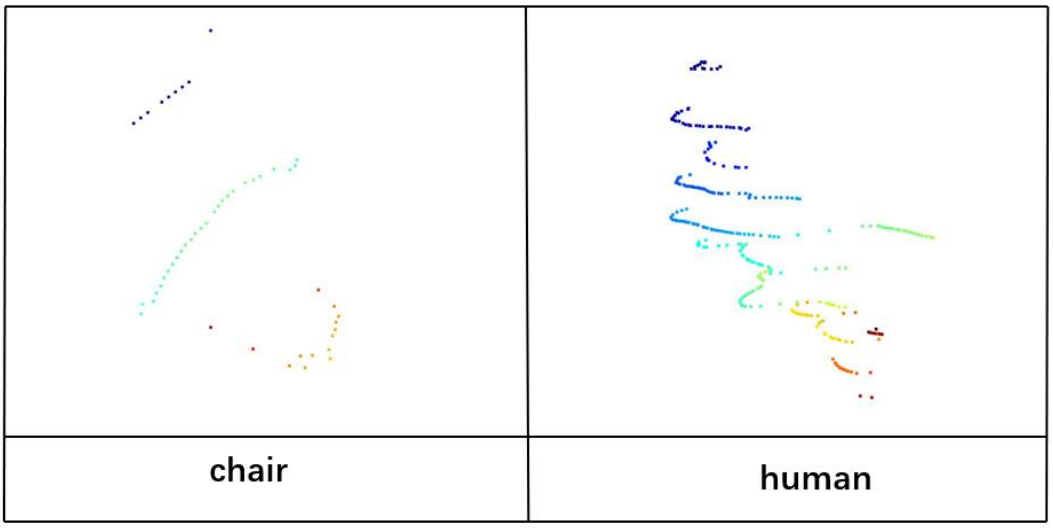

Figure 5.1: The extracted raw point cloud of each object.

as the robot, and the detected environment is the robot's surrounding environment. Due to the difference in the visual field, the surrounding information needed for the robot's navigation may need to be improved for infrastructure sensor nodes.

The complete shape of detected objects is essential for the robot's understanding of the surrounding environment and will benefit the robot's navigation. Due to the sparsity and incomplete of the detected point cloud and the difference in the field of view of the infrastructure sensor node, shape completion is essential for the perception pipeline. For infrastructure sensors, an object will be detected with different aspects, as its position and pose constantly change during its movement. The occluded part information can be completed with its information in unoccluded frames. Hence, objects' shape completion is performed during objects' moving in the perception system's field of view, as demonstrated in Fig.5.2.

5.2 Point Cloud Reconstruction

5.2.1 Point Cloud Object Tracking

The reconstruction of detected objects needs continuous frames of point clouds. Therefore, it is necessary to locate and select point clouds of the target object in a consecutive frame

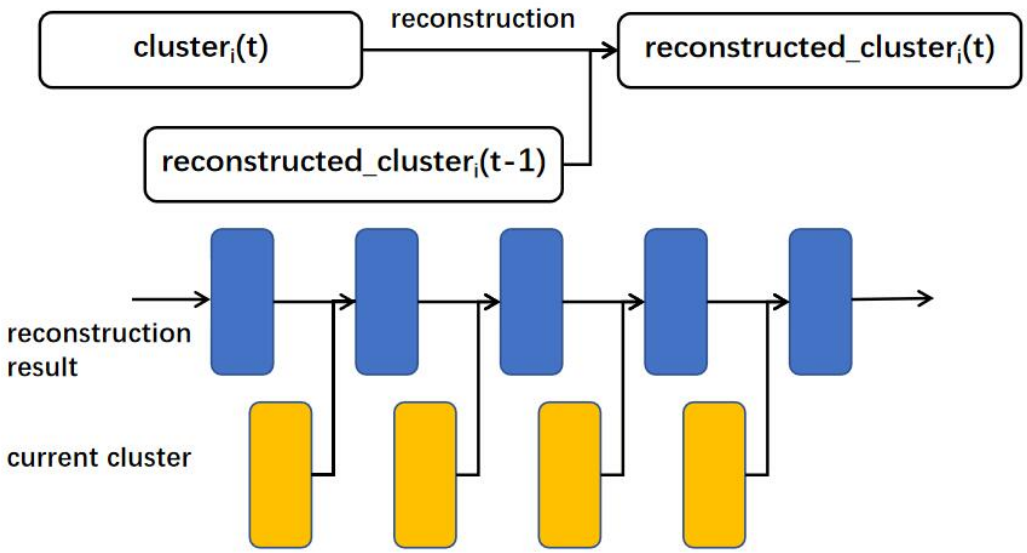

Figure 5.2: An illustration of aggregating points of a particular object to complete its shape when it's moving.

and combine the set of point clouds into a complete figure. Therefore, besides segmenting objects' points, an object tracking module is also needed as the pre-possessing part for the objects' reconstruction.

Unlike outdoor environments, where vehicles have a high velocity, usually more than $10m/s$, indoor robots and other objects, as pedestrians, move at relatively low speeds, with about $1 - 2m/s$. Also, due to the detected data, the scanned result usually contains only the part that faces the sensor node. Hence the calculated position, which is the x and y coordinate of the center point of the detected object, has an error with the object's actual position. Due to the characteristic of the detected point cloud, the center point will not only be changed by the object's changing position but will also have a slight change by the object's rotation even at the same position.

Point cloud tracking is usually done by Kalman filter to predict the position and motion of the objects. However, due to the low speed and irregular motion characteristic of indoor things and the imprecise position of the point cluster caused by the incomplete data, The established Kalman filter needs to show better performance when predicting the objects' motion. Considering the low movement speed, all the detected objects will have a relatively small position change between consecutive frames. Therefore global nearest

neighbour(GNN) is applied to track the objects in the point cloud. The concept of global nearest is simple: when all objects are moving at a low speed in the environment, the same object in two consecutive frames should be two detected objects that have the smallest relative distance. The pseudo-code of global nearest neighbour(GNN) for the object tracking is shown as algorithm 1:

Algorithm 1 Global Nearest Neighbour(GNN)

Input: clusters in time t: $C_t = c_{t_i}(i = 1, 2, ..., n)$
clusters in time $t+1$: $C_{t+1} = c_{t+1_i}(i = 1, 2, ..., m)$
point-tracking IDs in time t: $ID_t = id_{t_i}(i = 1, 2, ..., n)$
Output: point-tracking IDs in time $t+1$: $ID_{t+1} = id_{t+1_i}(i = 1, 2, ..., m)$

for i in range m **do**
    calculate $center_{t+1_i}$ from c_{t+1_i}
    $dist_{lowest} = threshold_{dist}$
    for j in range n: **do**
        calculate $center_{t_j}$ from c_{t_j}
        calculate distance $dist_{i_j}$ from $center_{t+1_i}$ and $center_{t_j}$
        get $lowestdist_i$ from all $dist_{i_j}$
        label cluster with $lowestdist_i$ as k
        if $dist_{i_j} \geq threshold_{dist}$ **then**
            allocate cluster c_{t+1_i} with a new id
        else
            $id_{t+1_i} = id_{t_k}$
        end if
    end for
end for

By global nearest neighbour, the clusters between consecutive frames are registered as several objects and tracked by given IDs.

5.2.2 Point Cloud Registration

A simple point cloud multi-object tracking module can obtain consecutive frames of point cloud for a specific object. With these successive clusters, it can reconstruct the point clouds and complete the shape of the entities.

w Iterative closest point(ICP) is one of the most widely used methods for objects' point cloud reconstruction. The input of the ICP algorithm is a target point cloud, a source point cloud and an initial transformation matrix. The algorithm's goal is to match the two point clouds as two parts of a complete figure. The concept of the ICP algorithm is simple: The matched pair of points from two point clouds should have the closest distance after transformation. Given two point clouds and an initial guess of transformation, find the matched points by the smallest global distance and calculate the conversion, transform the source point cloud with the transformation and match the point clouds iteratively. When the algorithm reaches the stopping criteria, a matching matrix and the optimal transformation result would be the output.

Given two point sets: $P_1 = a_i(i = 1, 2, ..., n)$, $P_2 = b_j(j = 1, 2, ..., m)$ and a initial transformationT_0 the details of the ICP is demonstrated as algorithm 2:

Algorithm 2 Iterative Closest Point(ICP)

Input: point cloud: $P_1 = a_i(i = 1, 2, ..., n)$
Point cloud $P_2 = b_j(j = 1, 2, ..., m)$
initial transformation matrixT_0
Output: optimal transformation matrix$T$

$T \leftarrow T_0$
while not converged **do**
  for i in range n **do**
    Transform pointsb_i in point cloudP_2 by $T \cdot b_j$
    find closest point in point cloudP_1 of $T \cdot b_j$ as m_i
    if $||T \cdot b_j - m_i|| \leq dist_{max}$ **then**
      $w_j \leftarrow 1$(the two points is matched)
    else
      $w_j \leftarrow 0$(the two points is not matched)
    end if
  end for
  $T \leftarrow \arg\min_T \sum_i w_j ||T \cdot b_j - m_i||^2$
end while

The Iterative closest point(ICP) algorithm has its limitations. Firstly the ICP is vulnerable to its initial guess of transformation. A bad initial value could lead to local optimum or take more steps of iteration. Hence optimization of the initial value for the ICP algorithm is tested. One of the standard methods for acquiring a more optimal initial guess is

to use another simple registration to get a coarse transformation matrix: global registration. Then with the global registration result as the initial input, the ICP algorithm will be applied to acquire the optimal result. The typical approach is to use RANSAC to fit the transformation matrix over detected key points by ISS[94] in corresponding described features by Fast Point Feature Histogram(FPFH)[63].

The principle of ISS is simple: the key points should bu points that have large variations in their neighbourhood[94]. Given a selected point p_j and a certain radius r, a weight is calculated showing the sparsity of the point's neighbour as Eq.5.1:

$$w_j = \frac{1}{|p_k : ||p_k - p_j||_2 < r|} \tag{5.1}$$

The point's weighted covariance matrix can be calculated as Eq.5.2:

$$w_i = \frac{\sum_{||p_i - p_j||_2 < r} w_i (p_i - p_j)(p_i - p_j)^{\mathrm{T}}}{\sum_{||p_k - p_i||_2 < r} w_j} \tag{5.2}$$

The key points can be calculated by performing principle component analysis over its weighted covariance matrix, the eigenvalues of the weight matrix can be calculated and shown in a decreased order:$\lambda_i^1, \lambda_i^2, \lambda_i^3$ The key point should have a distinct variation over its neighbour, for example, a corner like feature, also a point on a flat surface or a line should not be a key point. Thus given threshold γ_{21}, γ_{32}, a key point can be selected by the Eq.5.3 and Eq.5.4:

$$\frac{\lambda_i^2}{\lambda_i^1} < \gamma_{21} \tag{5.3}$$

$$\frac{\lambda_i^3}{\lambda_i^2} < \gamma_{32} \tag{5.4}$$

After detecting key points using ISS, Feature description is done over the key points using FPFH[63]. Fast Point Feature Histogram is a point cloud feature descriptor developed from Simple Point Feature Histogram(SPFH) and Point Feature Histogram(PFH). Given a certain point p_m and its neighbouring points p_k in a radius r, the point's FPFH feature can be described by their SPFH feature $SPFH(p_i)$ as Eq.5.5:

$$FPFH(p_m) = SPFH(p_m) + \frac{1}{k} \sum_{i=1}^{k} \frac{1}{||p_m - p_k||_2} \cdot SPFH(p_k) \tag{5.5}$$

Due to the problem that the point cloud is obviously sparse and has a typical lined feature, which is caused by the mechanism of laser scanning, and is not the intrinsic feature

of the detected objects, hence the ISS struggles to find effective key points over the sparse point cloud. Therefore optimizing the registration by applying a modified algorithm of the original ICP is tested.

For the original ICP, Given two point sets: $P_1 = a_i$, $P_2 = b_i$, its optimization goal in each step is to get the closest overall distance from each matched point pairs as Eq.5.6:

$$T_{opt} = \arg\min_{T} \sum_{i} ||T \cdot b_i - T \cdot a_i||^2 \tag{5.6}$$

For optimized algorithms of the original ICP, Plane to Plane ICP sets the optimization goal as matching the surfaces by surface normal. The Point to Plane ICP changes the optimization goal to projecting the point from the source point cloud to the surface plane of the target point cloud in the project. The data is sparse and has only a few lines of points scanned on the objects' surface. Hence in order to reconstruct objects with sparse scanned data, a good solution is to combine the point-wise information, the surface plane-wise information and the information between the point and surface plane. Hence Generalized-ICP[64] is applied to the project, which can be seen as a combination of Point to Point ICP(the original approach), Plane to Plane ICP and Point to Plane ICP. The generalized ICP uses a covariance matrix to represent points on their corresponding local surfaces. Hence the optimization goal can be viewed as a probabilistic problem.

The distance of a certain point in target point and the source point cloud can be represented by probabilistic distribution as Eq.5.7:

$$d_i^{(T_i)} = N(0, (T^*) \cdot C_i^A \cdot (T^*)^T) \tag{5.7}$$

And the objective function of G-ICP's optimization step can be described as:

$$T_{opt} = \arg\min_{T} \sum_{i} (d_i^{(T_i)})^{\mathrm{T}} \cdot (C_i^B + (T^*) \cdot C_i^A \cdot (T^*)^T) \cdot d_i^{(T_i)} \tag{5.8}$$

With two consecutive frames as the input, set the initial guess as a simple matrix considering the rotation and translation of the object is small over the scanning time of one frame, which is only $0.1s$. And the experiment shows that when only reconstructing the sparse object point clouds, the generalized ICP takes about 0.01 seconds to process one frame by voxelized the point cloud to make the data more evenly distributed and can also further accelerate the computation.

The result of generalized ICP to reconstruct a chair when its moving is as Fig.5.3: The reconstruction result over a consecutive set of frames is as Fig.5.4. It should be noted that

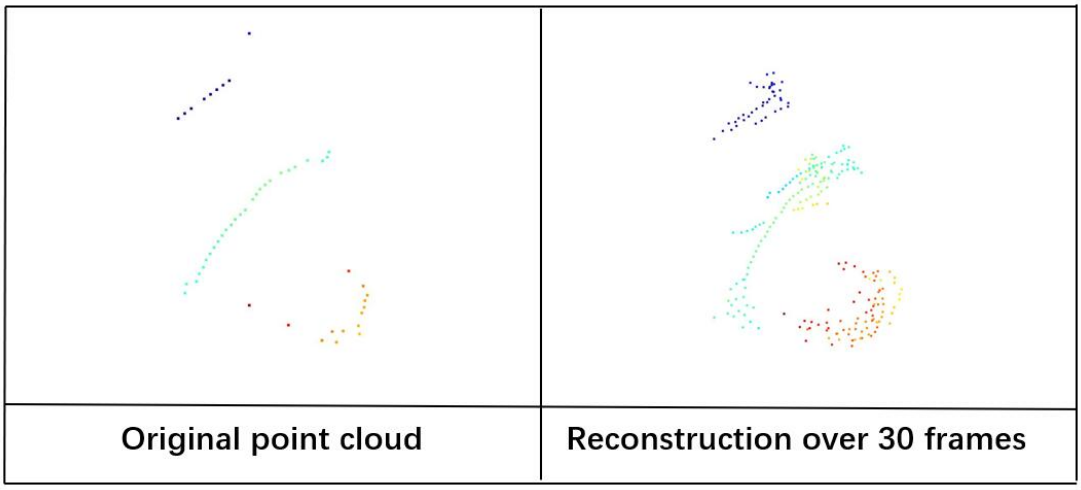

Figure 5.3: The original Lidar data and the reconstruction result of a chair

a human body is not a solid body, with arms, legs and waist moving by human activity. Hence the reconstruction is drawing space for the human body's field of movement. But by reconstruction, the data shows a much more obvious human feature than the original raw point cloud.

The result shows that the original point cloud is very sparse, and reconstruction can complete the objects' shape by registering the scanned sparse point clouds, which will benefit the path and motion planning system for the project in avoiding collision.

5.3 The Influence of Objects' Contact

During the experiment, it is noticed that there is also a difference in the environment between the normal outdoors and this project. In the outdoor environment, the most common scene for autonomous driving, there is usually a certain distance between every two objects, pedestrians will avoid getting too close to the vehicles, and the vehicles will keep a certain distance from obstacles to prevent the collision. Due to the narrow character of the indoor environment, the controlled distance between objects is usually tiny. In some cases, the objects would even have contact. For example, the pedestrian would grab or sit on the chair, or the workers would push the medical health care robot. In this case, the model-free approach to segment the object clusters would estimate the points of the two contacted objects as a single cluster.

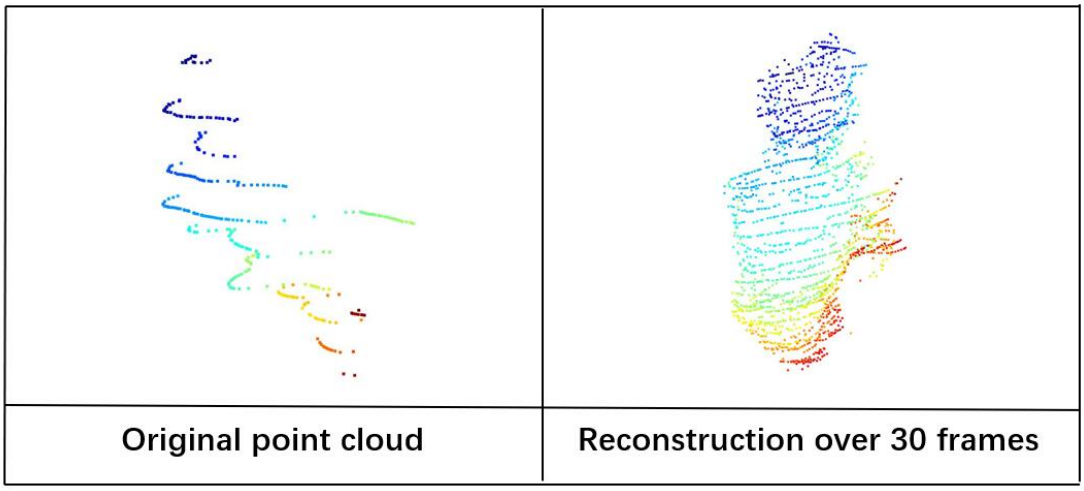

Figure 5.4: The original Lidar data and the reconstruction result of a human

5.3.1 Optimization Over Existing Structure

There are works that use the model-free method to segment points of objects and background, starting with the background removal method followed by point clustering algorithms that segment the remaining points into several existing object clusters. However, the works are all focused on the outdoor environment, where contact between objects seldom happens. Therefore such a situation is not mentioned in those works. In the indoor environment of the project, two objects are contacted or overlapped when viewed in the point cloud is much more common. Hence this chapter will discuss the impact of objects' contact and will propose a possible solution to its influence on the perception system.

When two or more objects are contacted and segmented as a single cluster, for the current frame, such a situation will not affect the path and motion planning followed by the perception, for in the current frame, the whole cluster can be viewed as one obstacle which needs to avoid collision with. However, the point cloud reconstruction process will be affected. When two objects move closer, the point reconstruction can be performed normally. When two objects are contacted, the two former reconstructed clusters will be registered with the current single cluster. However, the problem happens when the contract is over, for the combined cluster of two objects may be considered a history frame for reconstruction. In that case, there will be a distortion for the shape completion by point reconstruction, affecting our pipeline's environmental perception. Hence the contacted frame needs additional operation when the point reconstruction module is added to our

pipeline.

The global nearest neighbour is the point cloud tracking method to get the consecutive frames of a certain object. The distance between frames compares the clusters, and the clusters which have the shortest distance would be considered the same object. However, there is also a distance threshold. If the closest distance is larger than the threshold, the tracking algorithm would consider the cluster a new cluster and allocate it with a new id. Noticing that when two objects are contacted and segmented as one cluster, the center point would also shift to the center of the two objects Fig.5.5. Therefore by tuning the distance threshold, it is possible that when two objects with low-velocity contact with each other, the shift of the cluster center would cause the distances between the current and two former cluster centers both larger than the distance threshold. And when the contract is over, and the two objects depart from each other, the center points would shift from the center of two objects to the centers of the two separate clusters. In this case, the distance caused by the center shift would also be larger than the distance threshold. Hence in this process, the joint cluster of the two clusters and the two departed would both be considered as a new object and be allocated with a new point-tracking id. In this case, when the contact of objects is over, the joint cluster caused by contact would be departed, and the reconstruction would be restarted. Given two objects interacting in the environment, the explanation of the process is shown below:

(1) When the two objects are moving separately and start approaching, the tracking algorithm, which is GNN, can function normally and allocate the two objects as two point-tracking-IDs i_1 and i_2. Based on these two tracking ids, the objects' clusters are aggregated, and the reconstructed is preserved in a dictionary corresponding to the relevant id.

(2) When the two objects begin to contact, the relative distance would be small enough that DBSCAN would consider the points of the two objects as a single cluster. Then the centers of the two object clusters would shift to the single center of the merged cluster. Based on the assumption that the indoor objects are moving at a low velocity, the center shift would be larger than the tuned threshold of GNN. Thus the tracking algorithm would then consider the merged cluster as a new object and allocate with a new tracking-ID i_3. Therefore, the merged point cloud data would not be added to the reconstruction process of any of the two objects in this stage.

(3) When the two objects begin to depart from each other and the contact stage is over, the center of the merged cluster would again shift to the two separate centers of the two object clusters. Thus in this process, the shift of the center would again exceed the set distance threshold; therefore, GNN would consider the two objects different from the merged cluster. Therefore the two objects would be allocated with new tracking IDs i_4 and

i_5. Therefore the reconstruction process of the two objects would be reset to the beginning and begin to aggregate their point during their continuing motion again.

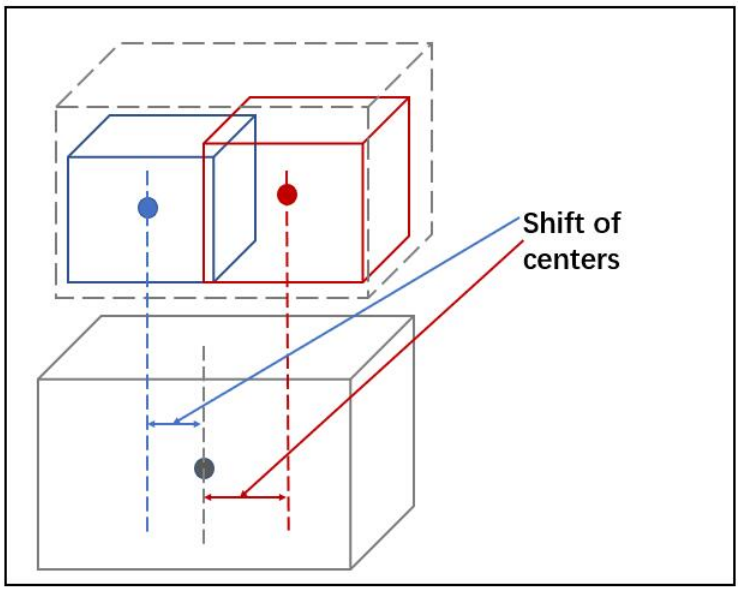

Figure 5.5: The centers of the two objects may shift to the center of the entire cluster in the process of the two object's contact.

5.3.2 Experiment

An experiment is conducted to test the tracking and reconstruction result with contacted objects using the modified GNN algorithm. The room is cleared with only a candidate and a chair left in the perception field of the infrastructure sensor node, which would serve as two candidates for interaction. The reconstruction result is published as the ROS topic and visualized in RViz. To further clarify the result, the relative detected image from the sensor node's camera of the environment is also visualized in RViz.

The interactive process is composed of three stages:

(1)The first stage would serve as an example of normal indoor circumstances. The candidate keeps a distance from the chair and begins walking to the chair.

(2)The second stage would illustrate the cases that are being studied, that multiple objects have contact, and a model-free approach would segment them as one cluster. In

this case, the candidate approaches the chair and begins to contact it by putting his arm on it.

(3) The last stage would simulate the ending stage of multi objects contact. The candidate leaves his arm from the chair and departs from it.

The reconstruction result would be observed to be two separate clusters before their contact and a single merged cluster when the contact happens. And if the function of the tuned distance of GNN is working as expected over its affection to the reconstruction process, it would be observed that the reconstruction result would again separate into two independent clusters when the contact process is ceased.

The result is shown as follows:

(1) In the first stage, the candidate is segmented into separate clusters, and the candidate is reconstructed during his approach to the chair. The chair is static to the Lidar; therefore, no new points are added in the reconstruction process. The reconstructed result is published as red clusters in RViz Fig.5.6.

(2) In the second stage, the candidate approaches the chair and puts his chair on the arm. By his contact, DBSCAN would consider segmenting the candidate and the chair as a single cluster, which is published as one merged cluster Fig.5.7.

(3) In the third stage, the candidate departs from the chair, and in this process, two separate clusters are shown as the reconstruction result in RViz, and it can be seen that the data of the big, merged cluster is not added to the reconstruction result of any one of the objects Fig.5.8. This means that, as expected, as the distance threshold of GNN is tuned, the shift of centers during the departure of contacted objects would cause the tracking algorithm to allocate the two objects with two new tracking IDs. With new tracking IDs, the reconstruction process of the two objects is reset when contact happens.

As the experiment shows, by tuning the distance threshold of the applied object tracking algorithm, which is GNN, the tracking and reconstruction process would be reset as contact happens. Therefore the merged point cloud caused by multi-objects' contact would not be added to the reconstruction process of the target obstacles.

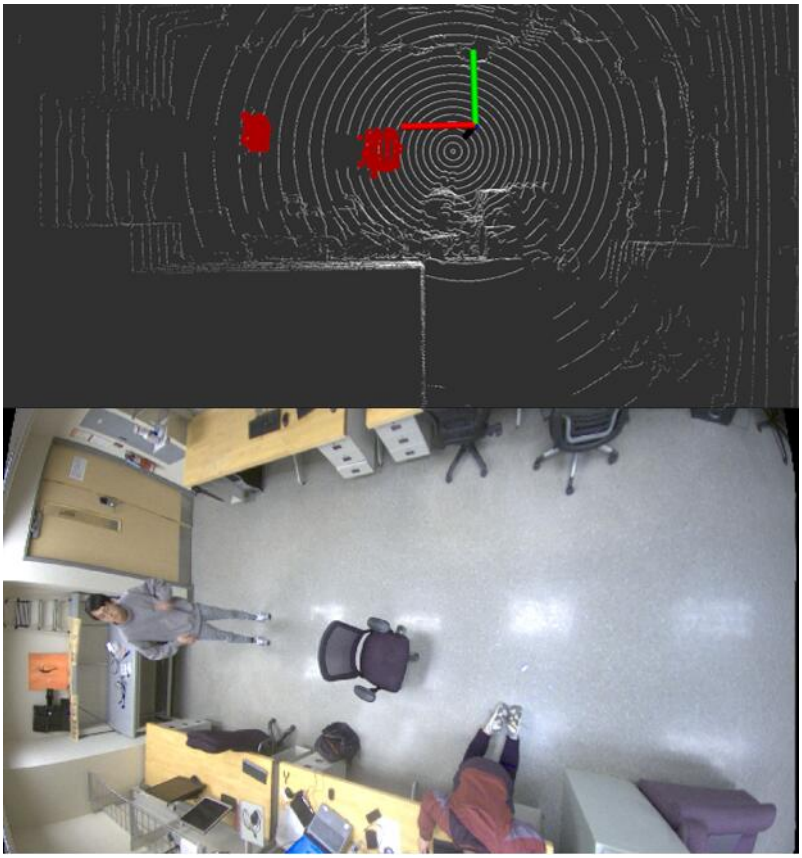

Figure 5.6: In stage 1, the objects are detected as separate clusters and the tracking and reconstruction are done normally.

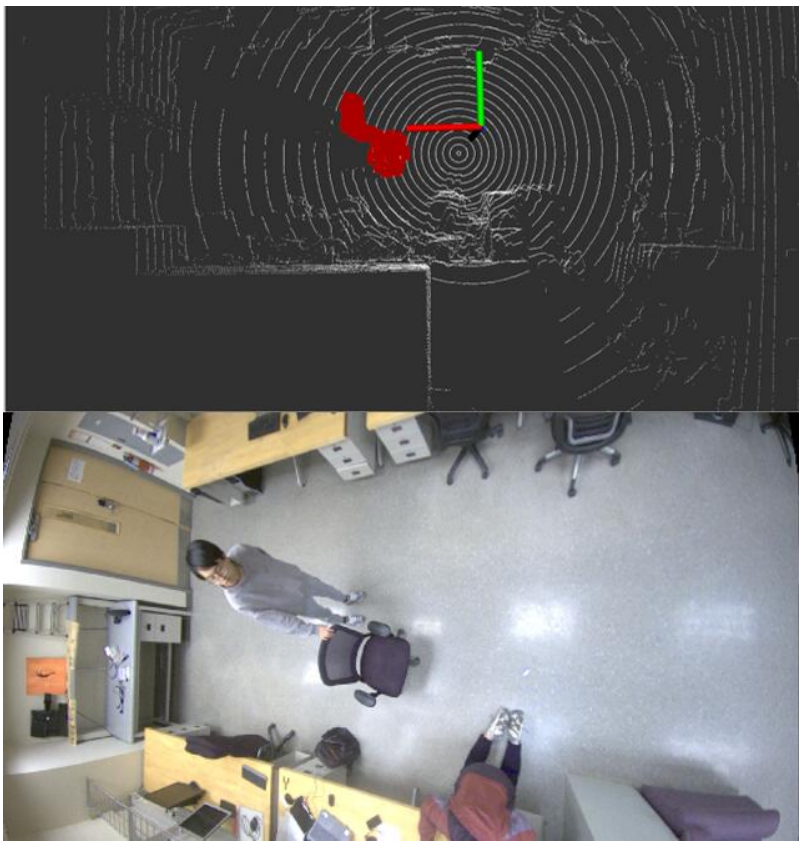

Figure 5.7: In stage 2, the objects are contacted and segmented as one single, combined cluster, the reconstruction for the two objects is stopped.

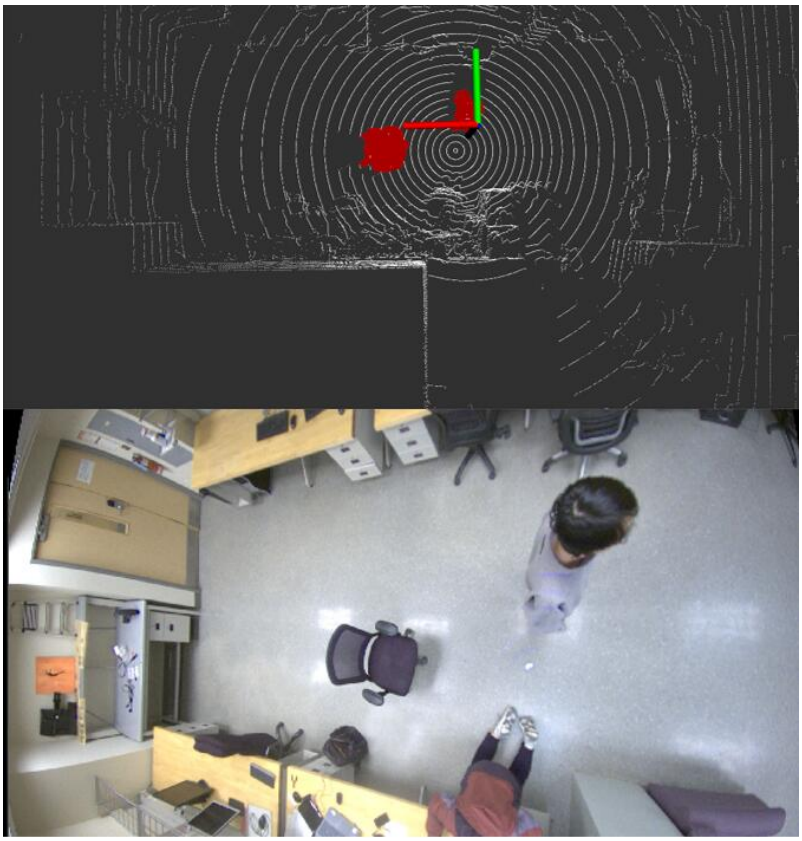

Figure 5.8: In stage 3, the objects are detected as separate clusters and the tracking and reconstruction are reset. Therefore the merged data in stage 2 is not added to the reconstruction process for each object.

Chapter 6

Conclusion

The work presents a pipeline with point cloud object segmentation and reconstruction using infrastructure sensor nodes for autonomous robots with general mobility. The proposed method is able to detect objects in the environment and separate them as independent clusters, and is also able to track and reconstruct these objects during their movement. Fig6.1.

Considering the variety of objects in the indoor environment and the sparsity of the acquired point cloud data, model-free approaches are more robust than the project's deep learning structure. To address the problems of the point cloud of detected objects, point cloud reconstruction was also applied.

The architecture starts with background removal with two methods. The first approach tries to process a point cloud into a depth map-like image by voxelization and compressing to x-y-plane. The method needs a clear frame of the environment as a reference. Given a new frame of the point cloud as a target point cloud, the depth where an obstacle occurs would be increased when matched with the depth image of the reference point cloud. Thus the voxels with occurring obstacles can be extracted. Though this approach can efficiently remove background points in a complex indoor environment, in further experiments, however, an obvious time lag between detection results and real-time data was noticed. For the first approach, the matching works well in removing complex surfaces, but the ROI that should be focused on is simply the ground floor. Thus the second approach uses a polygon to extract points in the drive-able area of robots and remove ground points by setting a height threshold, thus is much more efficient. With the background removed, objects are separated into independent object clusters using DBSCAN. Therefore, background removal and object clustering can detect and segment potential obstacles in 3D space.

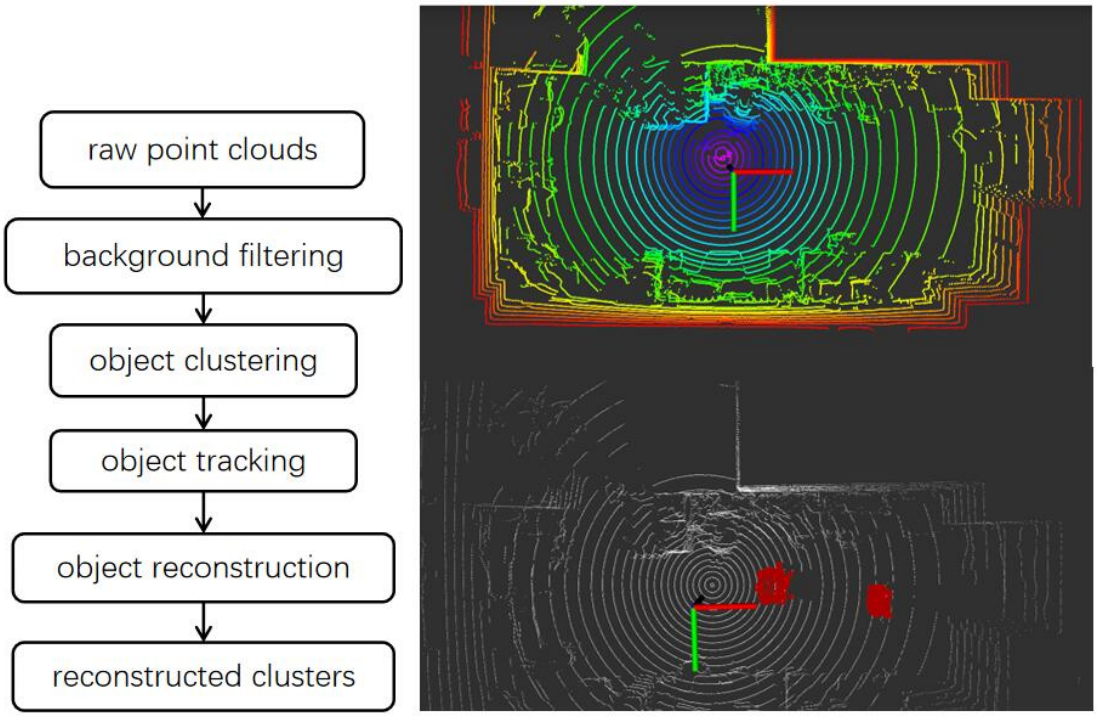

Figure 6.1: The overall structure has a sequence of background filtering, object clustering, object tracking and object reconstruction, the input would be raw pint cloud detected by the Lidar, and the output would be the segmented and reconstructed object clusters.

With objects detected and extracted from the environment as independent clusters, global nearest neighbour(GNN) is applied to track the occurred objects during their motion and preserve its detected point cloud as a consecutive stream. With continuous frames of point sets for a certain object, a point cloud registration algorithm is applied to combine and reconstruct these point clouds to complete the object's shape. Due to the characteristic of the point cloud data, which is sparse and has a typical line feature due to the scanning mechanism of Lidar rather than the object, generalized ICP is applied as the registration algorithm. Besides reconstructing objects by aggregating their point cloud, a special case is studied for the experiment environment. Different from road environments where vehicles and pedestrians tend to keep a distance from each other, contact between objects is more common in the low-speed, indoor environment. Two contacted objects will likely be segmented as a joint cluster for the model-free clustering algorithm, which should be separated from the point cloud reconstruction stream. Noticing that the process of

merging two clusters would also cause the shift of the center point, by tuning the distance threshold of GNN, which is used for tracking, the merged point cloud would be recognized as a new object and allocated with a new ID. Therefore rather than adding the joint cluster into any one of the two objects' reconstruction, the reconstruction process would restart once the contact happens. An experiment is also designed and conducted to verify this process.

6.1 Future Works

6.1.1 Further Optimization by Camera-Lidar Fusion

The sensor node is composed of a Lidar together with a camera. Combining the Lidar and camera's perception results for 3D detection is also an indispensable part of the perception pipeline. The camera module can not only be utilized to benefit the 3D detection but can also help optimize the reconstruction part.

The camera system uses Yolov4 to detect 2D objects on images and generate corresponding 2D bounding boxes and labels of classification. Then with the detection result by Yolov4, Deep-SORT is applied to track detected objects in 2D space and allocate them with their corresponding image-tracking id. The final goal of the perception pipeline is to enable real-time 3D detection and object tracking with Lidar and a camera established on the sensor node. Therefore, the detected point clusters must be labelled with object labels and IDs acquired from the camera module. The fusion algorithm has mainly three types: result level, feature level, and combined level. For the Lidar and camera modules to be done individually, it is difficult and unnecessary to go back to the unprocessed features of the two separate modules. Hence result-level fusion is suitable for the project. When establishing the sensor node, a point cloud-image projection matrix is acquired by Lidar-camera calibration. A point cloud can be projected on the image space with the projection matrix.

It is able to fuse the point cloud by locating the projected object clusters on the image. Image detection results in whether the projected point cluster is in the exact location with a particular bounding box. The intuition of determining whether a cluster represents the same object with a 2D bounding box is by calculating the overlap. The first considered idea is to fit another bounding box over the detected cluster and calculate the ratio of overlapped size with the existing bounding box:

By fusing the segmented result of the point cloud and the image, It can describe the situation of multi objects contact in a more precise way. When two objects are merged as

a single cluster under the clustering method, they would still be detected separately under the image detection module. For each frame, the fusion is done by searching existing clusters and finding their corresponding 2D bounding box in the image. If more than one 2D bounding box is allocated to a single cluster, then it means that multiple objects having contact and be merged as a single cluster; if only one 2D bounding box is allocated with the certain cluster, then the corresponding object label and tracking ID is allocated to the 3d cluster.

By allocating 3D object clusters with classified labels and tracking ID, 3D object detection and tracking with fusing Lidar and camera could be fulfilled. When considering the full sensor node, including the Lidar and the camera, there are two tracking modules: Global Nearest Neighbour(GNN) for point cloud tracking and Deep-Sort for image object tracking. The point cloud tracking module by GNN is used only for extracting the consecutive frames of a certain object for reconstruction, and the image tracking module by Deep-SORT is the main approach for tracking objects for the pipeline. Learning-based image detection and tracking would not face the problem of a model-free point cloud module that two or more objects merging into one cluster and is more robust to situations such as occlusion.

For the current point reconstruction, which is done by only the point cloud, the reconstruction of each object would restart each time when one object comes into contact with another object, and the history information of reconstruction would be lost. Point cloud-tracking would be interfered with by the contact situation. However, the image object tracking module would not interfere with such a situation. Each object in the camera's detection field would be tracked normally. Therefore, besides fusing the image information and point cloud information only with the detection and tracking result, the image detection and tracking module is also able to optimize the objects' point cloud reconstruction.

By fusing the image information, it is able to restore the former reconstruction result before contact by image-tracking IDs. Also, the mechanism of distance threshold of GNN that can separate objects is based on the assumption that the objects are all moving with a low velocity in a certain range, but the tracking reconstruction module with only the point cloud cannot precisely recognize the contact situation. Therefore, by detecting objects' contact, the fusion of the point cloud and image would give the point cloud reconstruction additional robustness.

6.1.2 Association of Multiple Sensor Nodes

Currently, only one infrastructure node is built and attached. A single sensor node can fulfill the perception task for robots entering its detection field. Multiple infrastructure sensor nodes would cover the entire area to navigate a robot through a complete indoor environment. When numerous sensors are added to the task, two new problems would appear sensor placement and data association.

The first task for multiple sensors is sensor placement. The detection range of the sensor nodes should cover the complete environment while minimizing the number of sensor nodes. Given the defined environment and the detection range for each sensor node, the problem can be acquiring the coordinates for sensors placed in the environment. Minimizing the number of sensor nodes can be converted to minimizing the overlap between the detection field and each sensor. Therefore the sensor placement task can be converted into an optimization problem.

As each sensor node is placed, it is necessary to associate data between sensor nodes. When an object is moving from one sensor's detection range to another, it is essential to transfer detection and reconstruction data between sensors. One of the possible solutions is to calibrate multiple sensor nodes and convert the point cloud data into a global detection coordinate. With the global coordinate, detecting and coarsely locating each obstacle in the environment is possible. When a specific object is moving in the overlapped location for multiple sensors, the point data for the objects can be transferred and completed by registration with the existing object reconstruction result and the detected data from each sensor node.

References

[1] Sanskar Agrawal, Indu Kant Deo, Siddhant Haldar, G Rahul Kranti Kiran, Vaibhav Lodhi, and Debashish Chakravarty. Off-road lane detection using superpixel clustering and ransac curve fitting. In *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 1942–1946. IEEE, 2018.

[2] Nir Aharon, Roy Orfaig, and Ben-Zion Bobrovsky. Bot-sort: Robust associations multi-pedestrian tracking. *arXiv preprint arXiv:2206.14651*, 2022.

[3] Ho Seok Ahn, Min Ho Lee, and Bruce A. MacDonald. Healthcare robot systems for a hospital environment: Carebot and receptionbot. In *2015 24th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 571–576, 2015.

[4] Wafaa Alakwaa, Mohammad Nassef, and Amr Badr. Lung cancer detection and classification with 3d convolutional neural network (3d-cnn). *International Journal of Advanced Computer Science and Applications*, 8(8), 2017.

[5] Waleed Ali, Sherif Abdelkarim, Mahmoud Zidan, Mohamed Zahran, and Ahmad El Sallab. Yolo3d: End-to-end real-time 3d oriented object bounding box detection from lidar point cloud. In *Proceedings of the European conference on computer vision (ECCV) workshops*, pages 0–0, 2018.

[6] Yasuhiro Aoki, Hunter Goforth, Rangaprasad Arun Srivatsan, and Simon Lucey. Pointnetlk: Robust & efficient point cloud registration using pointnet. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7163–7172, 2019.

[7] Eduardo Arnold, Omar Y Al-Jarrah, Mehrdad Dianati, Saber Fallah, David Oxtoby, and Alex Mouzakitis. A survey on 3d object detection methods for autonomous driving

applications. *IEEE Transactions on Intelligent Transportation Systems*, 20(10):3782–3795, 2019.

[8] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *2016 IEEE international conference on image processing (ICIP)*, pages 3464–3468. IEEE, 2016.

[9] Keerti Chand Bhupathi and Hasan Ferdowsi. Sharp curve detection of autonomous vehicles using dbscan and augmented sliding window techniques. *International Journal of Intelligent Transportation Systems Research*, 20(3):651–671, 2022.

[10] Peter Biber and Wolfgang Straßer. The normal distributions transform: A new approach to laser scan matching. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, volume 3, pages 2743–2748. IEEE, 2003.

[11] Attila Börcs, Balázs Nagy, and Csaba Benedek. Instant object detection in lidar point clouds. *IEEE Geoscience and Remote Sensing Letters*, 14(7):992–996, 2017.

[12] Elizabeth Broadbent, Rebecca Stafford, and Bruce MacDonald. Acceptance of healthcare robots for the older population: Review and future directions. *International journal of social robotics*, 1:319–330, 2009.

[13] Young-Hyen Byeon and Keun-Chang Kwak. Facial expression recognition using 3d convolutional neural network. *International journal of advanced computer science and applications*, 5(12), 2014.

[14] Mingcong Cao and Junmin Wang. Obstacle detection for autonomous driving vehicles with multi-lidar sensor fusion. *Journal of Dynamic Systems, Measurement, and Control*, 142(2):021007, 2020.

[15] Ferhat Ozgur Catak, Tao Yue, and Shaukat Ali. Prediction surface uncertainty quantification in object detection models for autonomous driving. In *2021 IEEE International Conference on Artificial Intelligence Testing (AITest)*, pages 93–100. IEEE, 2021.

[16] Hao Chen, Qi Dou, Lequan Yu, and Pheng-Ann Heng. Voxresnet: Deep voxelwise residual networks for volumetric brain segmentation. *arXiv preprint arXiv:1608.05895*, 2016.

[17] Yilun Chen, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Fast point r-cnn. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9775–9784, 2019.

[18] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 24(5):603–619, 2002.

[19] Yaodong Cui, Ren Chen, Wenbo Chu, Long Chen, Daxin Tian, Ying Li, and Dongpu Cao. Deep learning for image and point cloud fusion in autonomous driving: A review. *IEEE Transactions on Intelligent Transportation Systems*, 23(2):722–739, 2021.

[20] Yuepeng Cui, Hao Xu, Jianqing Wu, Yuan Sun, and Junxuan Zhao. Automatic vehicle tracking with roadside lidar data for the connected-vehicles system. *IEEE Intelligent Systems*, 34(3):44–51, 2019.

[21] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee, 2005.

[22] Ricardo Dias, Bernardo Cunha, Eduardo Sousa, José Luís Azevedo, João Silva, Filipe Amaral, and Nuno Lau. Real-time multi-object tracking on highly dynamic environments. In *2017 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 178–183. IEEE, 2017.

[23] Jianmin Dong, Yaxin Peng, Shihui Ying, and Zhiyu Hu. Lietricp: An improvement of trimmed iterative closest point algorithm. *Neurocomputing*, 140:67–76, 2014.

[24] Zhen Dong, Fuxun Liang, Bisheng Yang, Yusheng Xu, Yufu Zang, Jianping Li, Yuan Wang, Wenxia Dai, Hongchao Fan, Juha Hyyppä, et al. Registration of large-scale terrestrial laser scanner point clouds: A review and benchmark. *ISPRS Journal of Photogrammetry and Remote Sensing*, 163:327–342, 2020.

[25] Martin Engelcke, Dushyant Rao, Dominic Zeng Wang, Chi Hay Tong, and Ingmar Posner. Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1355–1361, 2017.

[26] Duarte Fernandes, António Silva, Rafael Névoa, Cláudia Simões, Dibet Gonzalez, Miguel Guevara, Paulo Novais, João Monteiro, and Pedro Melo-Pinto. Point-cloud based 3d object detection and classification methods for self-driving applications: A survey and taxonomy. *Information Fusion*, 68:161–191, 2021.

[27] Silvio Giancola, Jesus Zarzar, and Bernard Ghanem. Leveraging shape completion for 3d siamese tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1359–1368, 2019.

[28] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[29] Zan Gojcic, Caifa Zhou, Jan D Wegner, and Andreas Wieser. The perfect match: 3d point cloud matching with smoothed densities. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5545–5554, 2019.

[30] Saurabh Gupta, Ross Girshick, Pablo Arbeláez, and Jitendra Malik. Learning rich features from rgb-d images for object detection and segmentation. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part VII 13*, pages 345–360. Springer, 2014.

[31] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[32] Xiaoshui Huang, Guofeng Mei, Jian Zhang, and Rana Abbas. A comprehensive survey on point cloud registration. *arXiv preprint arXiv:2103.02690*, 2021.

[33] Mingyang Jiang, Yiran Wu, Tianqi Zhao, Zelin Zhao, and Cewu Lu. Pointsift: A sift-like network module for 3d point cloud semantic segmentation. *arXiv preprint arXiv:1807.00652*, 2018.

[34] Stefan Kahl, Connor M Wood, Maximilian Eibl, and Holger Klinck. Birdnet: A deep learning solution for avian diversity monitoring. *Ecological Informatics*, 61:101236, 2021.

[35] Zeashan Hameed Khan, Afifa Siddique, and Chang Won Lee. Robotics utilization for healthcare digitization in global covid-19 management. *International journal of environmental research and public health*, 17(11):3819, 2020.

[36] David Kirschner, Rosemarie Velik, Saeed Yahyanejad, Mathias Brandstötter, and Michael Hofbaur. Yumi, come and play with me! a collaborative robot for piecing together a tangram puzzle. In *Interactive Collaborative Robotics: First International Conference, ICR 2016, Budapest, Hungary, August 24-26, 2016, Proceedings 1*, pages 243–251. Springer, 2016.

[37] Maria Kyrarini, Fotios Lygerakis, Akilesh Rajavenkatanarayanan, Christos Sevastopoulos, Harish Ram Nambiappan, Kodur Krishna Chaitanya, Ashwin Ramesh Babu, Joanne Mathew, and Fillia Makedon. A survey of robots in healthcare. *Technologies*, 9(1):8, 2021.

[38] Christoph H Lampert, Matthew B Blaschko, and Thomas Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. In *2008 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE, 2008.

[39] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12697–12705, 2019.

[40] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European conference on computer vision (ECCV)*, pages 734–750, 2018.

[41] Jiaxin Li and Gim Hee Lee. Usip: Unsupervised stable interest point detection from 3d point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 361–370, 2019.

[42] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. *Advances in neural information processing systems*, 31, 2018.

[43] Baoyuan Liu, Min Wang, Hassan Foroosh, Marshall Tappen, and Marianna Pensky. Sparse convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 806–814, 2015.

[44] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. Deep learning for generic object detection: A survey. *International journal of computer vision*, 128:261–318, 2020.

[45] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pages 21–37. Springer, 2016.

[46] Weiping Liu, Jia Sun, Wanyi Li, Ting Hu, and Peng Wang. Deep learning on point clouds and its application: A survey. *Sensors*, 19(19):4188, 2019.

[47] Yingdong Ma, Xiankai Chen, and George Chen. Pedestrian detection and tracking using hog and oriented-lbp features. In *Network and Parallel Computing: 8th IFIP International Conference, NPC 2011, Changsha, China, October 21-23, 2011. Proceedings 8*, pages 176–184. Springer, 2011.

[48] Ankith Manjunath, Ying Liu, Bernardo Henriques, and Armin Engstle. Radar based object detection and tracking for autonomous driving. In *2018 IEEE MTT-S International Conference on Microwaves for Intelligent Mobility (ICMIM)*, pages 1–4. IEEE, 2018.

[49] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928, 2015.

[50] Anshul Paigwar, David Sierra-Gonzalez, Özgür Erkent, and Christian Laugier. Frustum-pointpillars: A multi-stage approach for 3d object detection using rgb camera and lidar. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2926–2933, 2021.

[51] Ziqi Pang, Zhichao Li, and Naiyan Wang. Model-free vehicle tracking and state estimation in point cloud sequences. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8075–8082. IEEE, 2021.

[52] Ying Peng, Yechen Qin, Xiaolin Tang, Zhiqiang Zhang, and Lei Deng. Survey on image and point-cloud fusion-based object detection in autonomous vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 23(12):22772–22789, 2022.

[53] AJ Piergiovanni, Vincent Casser, Michael S Ryoo, and Anelia Angelova. 4d-net for learned multi-modal alignment. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15435–15445, 2021.

[54] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep hough voting for 3d object detection in point clouds. In *proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9277–9286, 2019.

[55] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 918–927, 2018.

[56] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas 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*, pages 652–660, 2017.

[57] Haozhe Qi, Chen Feng, Zhiguo Cao, Feng Zhao, and Yang Xiao. P2b: Point-to-box network for 3d object tracking in point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6329–6338, 2020.

[58] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[59] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[60] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[61] Tiago Ribeiro, Inês Garcia, Dylan Pereira, João Ribeiro, Gil Lopes, and A.Fernando Ribeiro. Development of a prototype robot for transportation within industrial environments. In *2017 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 192–197, 2017.

[62] Laurel D Riek. Healthcare robotics. *Communications of the ACM*, 60(11):68–78, 2017.

[63] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *2009 IEEE international conference on robotics and automation*, pages 3212–3217. IEEE, 2009.

[64] Aleksandr Segal, Dirk Haehnel, and Sebastian Thrun. Generalized-icp. In *Robotics: science and systems*, volume 2, page 435. Seattle, WA, 2009.

[65] Mohammad Javad Shafiee, Brendan Chywl, Francis Li, and Alexander Wong. Fast yolo: A fast you only look once system for real-time embedded object detection in video. *arXiv preprint arXiv:1709.05943*, 2017.

[66] Jiayao Shan, Sifan Zhou, Zheng Fang, and Yubo Cui. Ptt: Point-track-transformer module for 3d single object tracking in point clouds. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1310–1316. IEEE, 2021.

[67] Kiwoo Shin, Youngwook Paul Kwon, and Masayoshi Tomizuka. Roarnet: A robust 3d object detection based on region approximation refinement. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 2510–2515, 2019.

[68] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2446–2454, 2020.

[69] Lucas Tabelini, Rodrigo Berriel, Thiago M Paixao, Claudine Badue, Alberto F De Souza, and Thiago Oliveira-Santos. Keep your eyes on the lane: Real-time attention-guided lane detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 294–302, 2021.

[70] Oncel Tuzel, Fatih Porikli, and Peter Meer. Human detection via classification on riemannian manifolds. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[71] Abhinav Valada, Rohit Mohan, and Wolfram Burgard. Self-supervised model adaptation for multimodal semantic segmentation. *International Journal of Computer Vision*, 128(5):1239–1285, 2020.

[72] Micaela Verucchi, Luca Bartoli, Fabio Bagni, Francesco Gatti, Paolo Burgio, and Marko Bertogna. Real-time clustering and lidar-camera fusion on embedded platforms for self-driving cars. In *2020 Fourth IEEE International Conference on Robotic Computing (IRC)*, pages 398–405. IEEE, 2020.

[73] Bingxu Wang, Jinhui Lan, and Jiangjiang Gao. Lidar filtering in 3d object detection based on improved ransac. *Remote Sensing*, 14(9):2110, 2022.

[74] Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. O-cnn: Octree-based convolutional neural networks for 3d shape analysis. *ACM Transactions On Graphics (TOG)*, 36(4):1–11, 2017.

[75] Xiaoyu Wang, Tony X. Han, and Shuicheng Yan. An hog-lbp human detector with partial occlusion handling. In *2009 IEEE 12th International Conference on Computer Vision*, pages 32–39, 2009.

[76] Yue Wang and Justin M Solomon. Deep closest point: Learning representations for point cloud registration. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3523–3532, 2019.

[77] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE international conference on image processing (ICIP)*, pages 3645–3649. IEEE, 2017.

[78] Hai Wu, Wenkai Han, Chenglu Wen, Xin Li, and Cheng Wang. 3d multi-object tracking in point clouds based on prediction confidence-guided data association. *IEEE Transactions on Intelligent Transportation Systems*, 23(6):5668–5677, 2021.

[79] Jianqing Wu, Hao Xu, Yuan Sun, Jianying Zheng, and Rui Yue. Automatic background filtering method for roadside lidar data. *Transportation Research Record*, 2672(45):106–114, 2018.

[80] Jianqing Wu, Hao Xu, and Jianying Zheng. Automatic background filtering and lane identification with roadside lidar data. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6. IEEE, 2017.

[81] Jianqing Wu, Hao Xu, Yichen Zheng, Yongsheng Zhang, Bin Lv, and Zong Tian. Automatic vehicle classification using roadside lidar data. *Transportation Research Record*, 2673(6):153–164, 2019.

[82] Yutian Wu, Yueyu Wang, Shuwei Zhang, and Harutoshi Ogai. Deep 3d object detection networks using lidar data: A review. *IEEE Sensors Journal*, 21(2):1152–1171, 2020.

[83] Zhuoyue Wu, Guirong Zhuo, and Feng Xue. Self-supervised monocular depth estimation scale recovery using ransac outlier removal. In *2020 4th CAA International Conference on Vehicular Control and Intelligence (CVCI)*, pages 97–102. IEEE, 2020.

[84] Danfei Xu, Dragomir Anguelov, and Ashesh Jain. Pointfusion: Deep sensor fusion for 3d bounding box estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 244–253, 2018.

[85] Heng Yang, Jingnan Shi, and Luca Carlone. Teaser: Fast and certifiable point cloud registration. *IEEE Transactions on Robotics*, 37(2):314–333, 2020.

[86] Zetong Yang, Yanan Sun, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Ipod: Intensive point-based object detector for point cloud. *arXiv preprint arXiv:1812.05276*, 2018.

[87] Maosheng Ye, Shuangjie Xu, and Tongyi Cao. Hvnet: Hybrid voxel network for lidar based 3d object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1631–1640, 2020.

[88] Dimitris Zermas, Izzat Izzat, and Nikolaos Papanikolopoulos. Fast segmentation of 3d point clouds: A paradigm on lidar data for autonomous vehicle applications. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5067–5073, 2017.

[89] Juyong Zhang, Yuxin Yao, and Bailin Deng. Fast and robust iterative closest point. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[90] Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Fucheng Weng, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. Bytetrack: Multi-object tracking by associating every detection box. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXII*, pages 1–21. Springer, 2022.

[91] Junxuan Zhao, Hao Xu, Hongchao Liu, Jianqing Wu, Yichen Zheng, and Dayong Wu. Detection and tracking of pedestrians and vehicles using roadside lidar sensors. *Transportation research part C: emerging technologies*, 100:68–87, 2019.

[92] Xin Zhao, Zhe Liu, Ruolan Hu, and Kaiqi Huang. 3d object detection using scale invariant and feature reweighting networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9267–9274, 2019.

[93] Chaoda Zheng, Xu Yan, Jiantao Gao, Weibing Zhao, Wei Zhang, Zhen Li, and Shuguang Cui. Box-aware feature enhancement for single object tracking on point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13199–13208, 2021.

[94] Yu Zhong. Intrinsic shape signatures: A shape descriptor for 3d object recognition. In *2009 IEEE 12th international conference on computer vision workshops, ICCV Workshops*, pages 689–696. IEEE, 2009.

[95] Changqing Zhou, Zhipeng Luo, Yueru Luo, Tianrui Liu, Liang Pan, Zhongang Cai, Haiyu Zhao, and Shijian Lu. Pttr: Relational 3d point cloud object tracking with transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8531–8540, 2022.

[96] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Tracking objects as points. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IV*, pages 474–490. Springer, 2020.

[97] Yin Zhou and Oncel 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*, pages 4490–4499, 2018.

[98] Walter Zimmer, Emec Ercelik, Xingcheng Zhou, Xavier Jair Diaz Ortiz, and Alois Knoll. A survey of robust 3d object detection methods in point clouds. *arXiv preprint arXiv:2204.00106*, 2022.