*mathematics*

MDPI

*Article*

# Instance Segmentation of Sparse Point Clouds with Spatio-Temporal Coding for Autonomous Robot

**Na Liu [1], Ye Yuan [1], Sai Zhang [2], Guodong Wu [2], Jie Leng [1,2] and Lihong Wan [2,*]**

[1] Institute of Machine Intelligence, University of Shanghai for Science and Technology, Shanghai 200093, China; liuna@usst.edu.cn (N.L.); yuanye_usst@usst.edu.cn (Y.Y.); 191550059@st.usst.edu.cn (J.L.)
[2] Origin Dynamics Intelligent Robot Co., Ltd., Zhengzhou 450000, China; 1910242006@email.szu.edu.cn (S.Z.); wuguodong@odrobots.com (G.W.)
* Correspondence: wanlihong@odrobots.com

**Abstract:** In the study of Simultaneous Localization and Mapping (SLAM), the existence of dynamic obstacles will have a great impact on it, and when there are many dynamic obstacles, it will lead to great challenges in mapping. Therefore, segmenting dynamic objects in the environment is particularly important. The common data format in the field of autonomous robots is point clouds. How to use point clouds to segment dynamic objects is the focus of this study. The existing point clouds instance segmentation methods are mostly based on dense point clouds. In our application scenario, we use 16-line LiDAR (sparse point clouds) and propose a sparse point clouds instance segmentation method based on spatio-temporal encoding and decoding for autonomous robots in dynamic environments. Compared with other point clouds instance segmentation methods, the proposed algorithm has significantly improved average percision and average recall on instance segmentation of our point clouds dataset. In addition, the annotation of point clouds is time-consuming and laborious, and the existing dataset for point clouds instance segmentation is also very limited. Thus, we propose an autonomous point clouds annotation algorithm that integrates object tracking, segmentation, and point clouds to 2D mapping methods, the resulting data can then be used for training robust model.

**Keywords:** point clouds; dynamic targets; instance segmentation; spatio-temporal; autonomous robot

**MSC:** 93-10



**Citation:** Liu, N.; Yuan, Y.; Zhang, S.; Wu, G.; Leng, J.; Wan, L. Instance Segmentation of Sparse Point Clouds with Spatio-Temporal Coding for Autonomous Robot. *Mathematics* **2024**, *12*, 1200. https://doi.org/10.3390/math12081200

Academic Editor: António Lopes

Received: 8 March 2024
Revised: 3 April 2024
Accepted: 10 April 2024
Published: 17 April 2024



## 1. Introduction

SLAM [1] is an important module of autonomous robots [2]. The tasks of robots include mapping, localization, and path planning. Building an environment map is the foundation of robot tasks, and the map can be used for subsequent tasks. The construction of map often relies on point clouds, and the dynamic targets in point clouds pose challenges to map construction [3]. A point clouds map containing dynamic objects is shown in the following Figure 1. The current solution to address challenges is to start with point clouds. The processing of point clouds includes semantic segmentation and instance segmentation [4]. Instance segmentation not only needs to distinguish which class each point belongs to, but also needs to distinguish different individuals in the same class [5].

There are several ways to deal with object extraction in different stages. At the registration stage [6,7], for objects with rapid changes in motion state, traditional or neural networks can usually be used to filter them out. At the stage of mapping [3,8–10], high-dynamic objects are filtered synchronously during the SLAM process, in order to use the information of all frames. Post-processing is performed on the map after the SLAM process is completed to filter out objects with slow changes in motion status. This method is more effective for temporarily stationary objects. In addition, the construction level includes lifelong processing [11,12] for dynamic object filtering and semi static object updates. The

post-processing method can combine more information to more accurately filter out the target object, which is a better way.

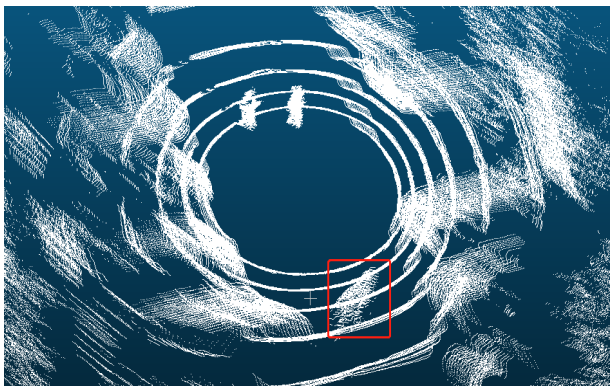


**Figure 1.** Point cloud map containing dynamic targets. The red rectangle in the map shows a moving person. Due to this person's movement, the point cloud in the red rectangle looks like a "ghost shadow".

The final foothold of all methods is in the processing of point clouds. The environmental perception of autonomous driving also needs to process the data from LiDAR, perceive the specific target in the point clouds scanned by the LiDAR, and provide corresponding strategies. Directly processing the dynamic objects in the point clouds can avoid the influence of the dynamic environment on the construction of the map. Currently, for the acquisition of data to study specific targets in autonomous robots, most applications use 64-line [13] or above LiDAR. According to the number of lines, LiDAR can be divided into single-line, 4-line, 16-line, 64-line, 128-line, etc. As the number of lines increases, the number of points in the point clouds obtained by the LiDAR continues to increase. The difficulty in the instance segmentation of the point clouds decreases, but the cost also increases. The price of 64-line LiDAR is about three times than that of 16-line LiDAR. Autonomous robots typically use 16-line LiDAR for development and research. The number of points obtained by 16-line LiDAR is one fourth of 64-line LiDAR. A comparison between sparse point clouds and dense point clouds is shown in Figure 2. For data with sparse point clouds, there are also fewer features of the target in Figure 2. Currently, there is relatively little research on instance segmentation for sparse point clouds. In reality, the use of 16-line LiDAR is much more common than that of 64 and 128-line LiDAR.

Thus, we propose a solution for instance segmentation of sparse point clouds. In general, sparser point clouds have fewer features and it is difficult to recognize target objects with the naked eye, making manual annotation more difficult. To address this issue, we propose a scheme for the instance segmentation and annotation of sparse point clouds using integrated spatio-temporal information. Overall, the contributions of this paper are as follows. First, a new point clouds annotation method is proposed to provide a large amount of data for point clouds instance segmentation model training. Secondly, we propose a novel spatio-temporal encoding and decoding, and incorporate spatio-temporal semantic loss into the instance segmentation model. The segmentation results have significantly improved compared to when they were not introduced. Finally, we propose spatio-temporal information splitting to generate instance segmentation results for sparse point clouds.

The remainder of this paper is organized as follows. After presenting the related work, we present methods for the creation of datasets and object-specific instance segmentation in the point clouds of dynamic environments, followed by the experimental description, discussion and conclusions in this section.

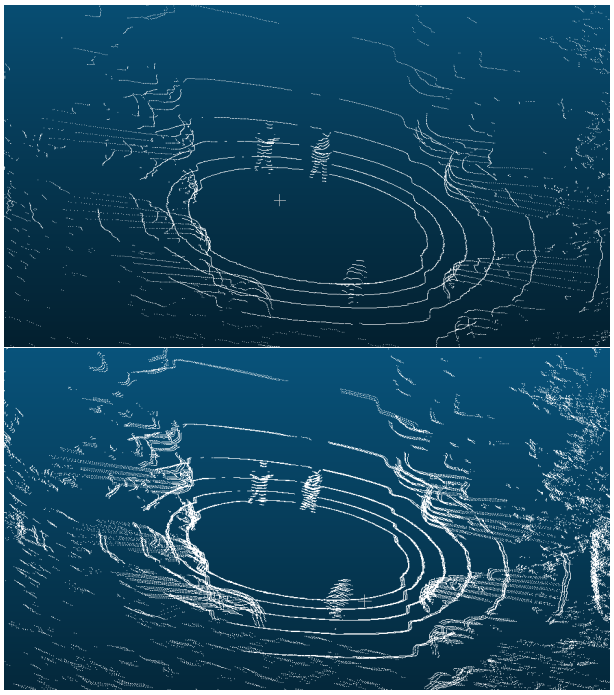


**Figure 2.** Sparse (top) and dense (bottom) point clouds. The top is sparse point clouds and the bottom is dense point clouds.

## 2. Related Work

### *2.1. Point Clouds Instance Segmentation*

The top-down proposal based approach is done based on zones, and then the objects are segmented within each zone. Because the point clouds has the characteristic of data irregularity, Yi et al. [14] propose a top-down method, the resulting proposals are highly characteristic, and the overall network is based on PointNet. In addition to this, some studies also consider RGB information. Hou et al. [15] present 3D semantic instance segmentation network (3DSIS). The network fuses the RGB information of the view, and works with the geometry information to predict the bounding box and predict the instance. Yang et al. [16] propose a new network which breaks away from the traditional anchor points, the method does not use non-maximum suppression, and a classifier to classify each point is selected to achieve object segmentation. Liu et al. [17] present a network which can extract the approximate instance center of each object, and then sample the results to get the desired instance. Proposal based methods process each target proposal independently without interference from other instances. However, proposal based methods struggle to generate high-quality proposals, because the acquired point exists on the surface of the object.

### *2.2. Autonomous Robot Dynamic Environment Target Filtering*

The existence of dynamic problems has attracted widespread attention and there are many studies either from camera data [18,19] or from laser scans [20–22]. To extract dynamic objects from camera data, Chabot et al. [19] as well as Reddy et al. [23] use neural networks to process images as input, while outputting classification and motion status. Similarly, Vertens et al. [24] propose fuse detection of vehicle status, the neural network takes into account the camera's image flow and optical flow information as inputs to the network. Chen et al. [25] select image information from different 3D views to predict bounding boxes of different categories. The task of processing and detecting objects by Xu et al. [26] combines the information of images and 3D scans, and assigns 3D scans to each detection. Li et al. [22] use neural networks to detect objects, prior to which distance images were obtained through 3D scanning. Engelcke et al. [21] achieved object detection in 3D point clouds by utilizing feature centered voting schemes. Wang et al. [27] were

able to directly detect target objects in 3D scanning. They select a fast network based on sliding windows for directly detecting objects in 3D scanning. Dewan et al. [20] detect moving points in 3D scanning by calculating the motion information between two frames of scanning. Hahnel et al. [28] propose a probability based method that can estimate the beam reflected by moving objects throughout 3D scanning, and establish a mapping of stationary objects. Meyer-Delius et al. [29] propose a grid occupying method using a hidden Markov model, which can detect potential changes in each element.

Most of the current methods are based on traditional machine learning methods and tracking of specific objects in point clouds of dynamic environments. In order to perform environment perception and dynamic target confirmation more accurately, we use the instance segmentation of the dynamic environment point clouds to complete. Sparse point cloudssets are few and difficult to label, while deep learning methods for 2D images are relatively mature. We combine the tracking and segmentation methods of 2D images to complete the labeling of sparse point clouds.

## 3. Methods

### *3.1. Automatic Data Annotation*

Most methods for point clouds annotation are manual annotation, mainly by using existing annotation software to manually identify point clouds instances and perform annotation. Therefore, manual annotation is time-consuming and laborious, and due to the sparse point clouds obtained by 16-line LiDAR used in our study, the target objects in the point clouds are not obvious, making manual annotation more difficult. Therefore, we also studied an annotation scheme for sparse point clouds. While using the LiDAR of autonomous robots to collect point clouds, the Intel RealSense Depth Camera D435 (Intel-D435 camera) is also used to collect image data, Intel-D435 camera is produced by Intel corporation in the United States. Intel is headquartered in the United States, specifically in Santa Clara, California. We know that based on external parameters of LiDAR and camera, we can project point clouds to 2D mapping. Compared with instance segmentation of point clouds, there are more studies on instance segmentation of 2D images, and the segmentation model is also relatively mature. And in order to preserve the spatio-temporal information of adjacent frames. We integrate target tracking of images with instance segmentation methods to autonomously annotate point clouds. In this study, we independently annotated the person and car in the point clouds.

The process of autonomous annotation of point clouds is shown in the Figure 3. The leftmost column in the Figure 3 is the original data obtained by the LiDAR and camera. Firstly, the image data obtains the mask of person and car through the instance segmentation network and target tracking process of the 2D image. Secondly, based on the coordinate system relationship between the camera and the LiDAR, the point clouds is mapped to 2D, and the annotation results of the corresponding point clouds instances are obtained based on the results of the mask, namely the rightmost columns Figure 3. We choose yolov5 as the model for image instance segmentation, and Fastmot as the target tracking model. The details of these two models will no longer be described. The following is a detailed description of how to use image segmentation results to generate point clouds annotations.

The key to this process lies in the coordinate transformation from the LiDAR to the camera. The 3D coordinates in space are $(X_w, Y_w, Z_w)^T$, The homogeneous coordinates is expressed as $(X_w, Y_w, Z_w, 1)^T$. The coordinates of the projection point are $(u_c, v_c)^T$. The homogeneous coordinates is expressed as $(u_c, v_c, 1)^T$. The internal parameter matrix of the camera is $K$. The perspective projection model of $R$ and $t$ is specifically described as Equations (1) and (2) :

$$z_c \begin{bmatrix} u_c \\ v_c \\ 1 \end{bmatrix} = \begin{bmatrix} f_{11} & f_{12} & f_{13} & f_{14} \\ f_{21} & f_{22} & f_{23} & f_{24} \\ f_{31} & f_{32} & f_{33} & f_{34} \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \tag{1}$$

where

$$\begin{bmatrix} f_{11} & f_{12} & f_{13} & f_{14} \\ f_{21} & f_{22} & f_{23} & f_{24} \\ f_{31} & f_{32} & f_{33} & f_{34} \end{bmatrix} = K \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix} \tag{2}$$

write Equation (1) in the form of a system of equations and eliminate $z_c$ to obtain Equation (3):

$$\begin{aligned} &f_{11}X_w + f_{12}Y_w + f_{13}Z_w + f_{14} - f_{31}X_w u_c - f_{32}Y_w u_c - \\ &f_{33}Z_w u_c - f_{34}u_c + f_{21}X_w + f_{22}Y_w + f_{23}Z_w + f_{24} - \\ &f_{31}X_w v_c - f_{32}Y_w v_c - f_{33}Z_w v_c - f_{34}v_c = 0 \end{aligned} \tag{3}$$

Each set of 3D-2D matching points corresponds to two equations, with a total of 12 unknowns, requiring at least 6 sets of matching points. The above Equation (3) is written in matrix form, and the values of $f_{11}$-$f_{34}$ of system of linear equations are solved. Therefore, the rotation matrix and translation matrix can be obtained as Equations (4) and (5) :

$$R = K^{-1} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \tag{4}$$

$$t = K^{-1} \begin{bmatrix} f_{14} \\ f_{24} \\ f_{34} \end{bmatrix} \tag{5}$$

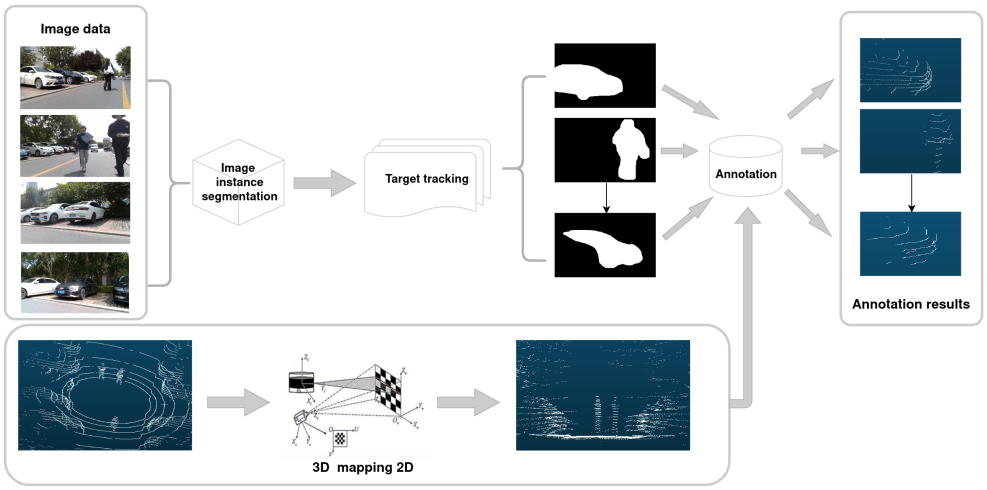



**Figure 3.** The flowchart of the proposed annotation for point clouds.

After obtaining the external parameter matrix of the camera and LiDAR, we map the point clouds onto a 2D mapping. Based on the results obtained from image segmentation, we extract the corresponding 3D point clouds and save it as a point clouds instance. Complete the autonomous annotation of the point clouds.

### *3.2. Proposed Instance Segmentation*

The overall architecture of the proposed model is depicted in Figure 4, the overall model consists of two stages. The first stage is the semantic segmentation, the input point clouds generates point-level semantic labels and offset vectors, and the second stage generates instance proposals for these output groupings. Using proposal method, utilize a backbone network to extract features from the data that can be used for classification, generation of instance masks, and scoring of generated masks. During the movement of the autonomous robot, the point clouds information obtained by the LiDAR has continuity in time and space. Therefore, we add spatio-temporal coding to make full use of spatio-temporal information.

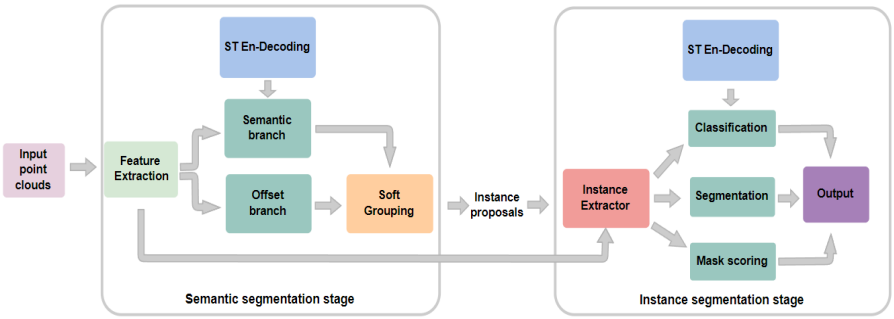



**Figure 4.** The framework of the proposed instance segmentation for point clouds.

Using a point by point prediction method, the input of the prediction network is a set of $N$ points that each point contains coordinate and color information, and then the point clouds is voxelated into an ordered voxel grid. These voxel grids are used as inputs for U-Net-style backbone [30] to obtain features. The backbone of U-Net-style is shown in the Figure 5. The term 'cat' in the network refers to the connection of feature vectors and the term 'identity' in the network refers to the feature. where the structures of 'conv' and 'deconv' are shown in the following Figure 6. The Spconv (Spatially Sparse Convolution) in the figure is a spatially sparse convolutional library used in this study to replace conventional convolutions. The conv and deconv operations are represented by Equations (6) and (7), where $\mu$ is the mean of x, $\sigma$ is the variance of x, $\epsilon$ is a very small positive number (used for numerical stability), $\gamma$ and $\beta$ are learnable scaling factor and offset parameters. The ReLU function turns each negative value in the input vector to zero, mathematically represented as max of 0 and $f$. Our 3D point clouds feature extraction is achieved using Submanifold Sparse Convolution [31], and the model outputs features through two branches to obtain pointwise semantic scores and offset vectors.

$$F_{conv} = spconv.SparseConv3d\left(max\left(0, \frac{F-\mu}{\sqrt{\sigma^2-\epsilon}}\gamma+\beta\right)\right) \tag{6}$$

$$F_{deconv} = spconv.SparseInverseConv3d\left(max\left(0, \frac{F-\mu}{\sqrt{\sigma^2-\epsilon}}\gamma+\beta\right)\right) \tag{7}$$

Cross-entropy loss ($CE$) is used in the semantic training branch, and $l_1$ regression loss is used in the offset branch. The semantic loss and offset loss are as follow Equations (8) and (9):

$$L_{semantic} = \frac{1}{N}\sum_{i=1}^{N} CE(s_i, s_i^*) \tag{8}$$

$$L_{offset} = \frac{1}{\sum_{i=1}^{N}\mathbb{I}_{\{p_i\}}}\sum_{i=1}^{N}\mathbb{I}_{\{p_i\}}\|o_i - o_i^*\|_1 \tag{9}$$

where the semantic score of the output is represented by $s$, the output offset vectors is $o$, $s^*$ is the semantic label, $o^*$ is offset label representing the vector from a point to the geometric center of the instance that the point belongs to (analogous to [32–34]), $N$ is the number of points, and $\mathbb{I}_{\{p_i\}}$ is the indicator function indicating whether the point $p_i$ belongs to any instance. In addition, in order to preserve the spatio-temporal information, we add spatio-temporal encoding and decoding in the training process, the loss between point clouds in the loss function, and extract the results of $N-1$ frames that are exactly the same from two adjacent point clouds, and solve the cross entropy loss function. The semantic loss is shown in Equation (10).

$$L_{\text{semantic local}} = \frac{1}{N}\sum_{i=1}^{N} CE(sl1_i, sl2_i) \tag{10}$$

wherein, if the frame of two overlapping point clouds is *i-j*, then the *sl1* is the semantic score of frame *i-j* of the first point clouds, *sl2* is the semantic score of frame *i-j* of the second point clouds.

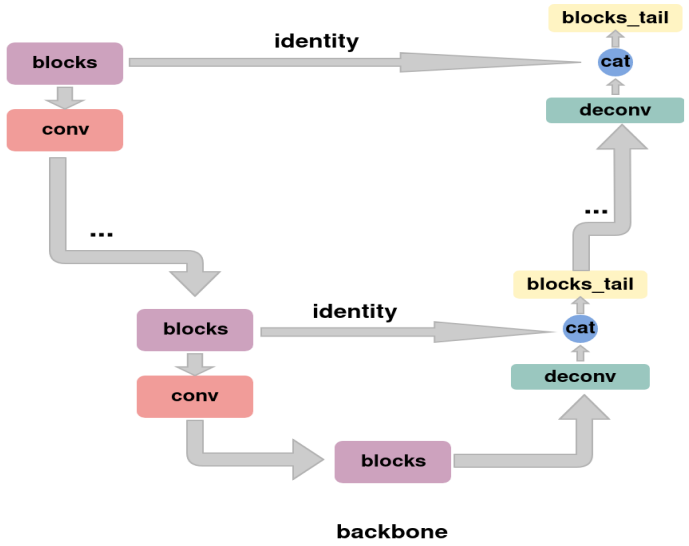



**Figure 5.** The U-Net-style backbone and the detailed structure descriptions. The structures of the blocks and blocks_tail are shown in the following Figures 7 and 8.

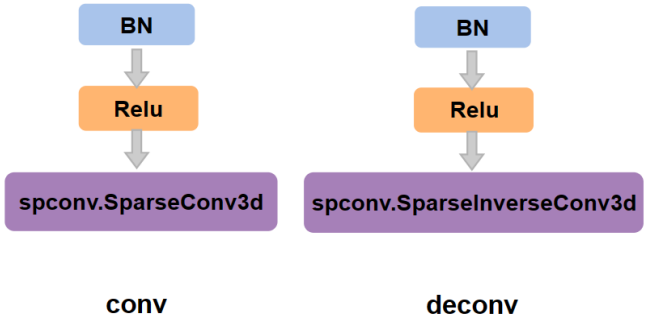



**Figure 6.** The conv and deconv modules.

For the generated instances, it is recommended to refine them from top to bottom, obtain classification and refinement results, extract features from each proposal through a feature extractor, and then input the features into a U-Net network with fewer layers. The tiny U-Net network is shown in Figure 9. The structural details in the network, such as 'conv','deconv','blocks'..., are consistent with the previous ones. The training loss [35,36] of these branches is the combination of cross-entropy, binary cross-entropy (*BCE*), and $l_2$ regression losses. The losses of class, mask, and mask score are Equation (11), Equation (12), and Equation (13), respectively.

$$L_{class} = \frac{1}{K}\sum_{k=1}^{K} CE(c_k, c_k^*) \tag{11}$$

$$L_{mask} = \frac{1}{\sum_{k=1}^{K} \mathbb{I}_{\{m_k\}}} \sum_{k=1}^{K} \mathbb{I}_{\{m_k\}} BCE(\boldsymbol{m}_k, \boldsymbol{m}_k^*) \tag{12}$$

$$L_{\text{mask score}} = \frac{1}{\sum_{k=1}^{K} \mathbb{I}_{\{r_k\}}} \sum_{k=1}^{K} \mathbb{I}_{\{r_k\}} \|\boldsymbol{r}_k - \boldsymbol{r}_k^*\|_2 \tag{13}$$

where $c^*$, $m^*$, $r^*$ are the classification, segmentation, and mask scoring targets, respectively. $K$ is the total number of proposals and $\mathbb{I}_{\{.\}}$ indicates whether the proposal is a positive sample.

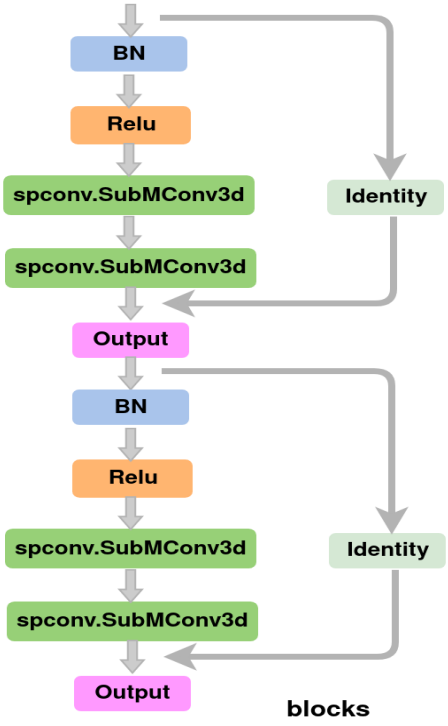



**Figure 7.** The structure of the blocks.

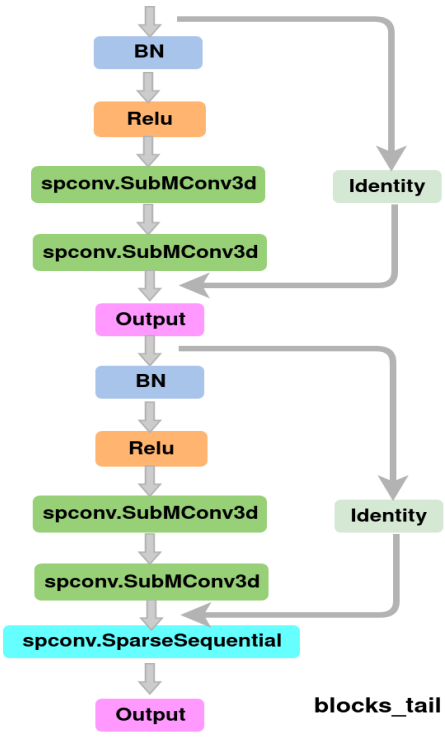



**Figure 8.** The structure of blocks_tail.

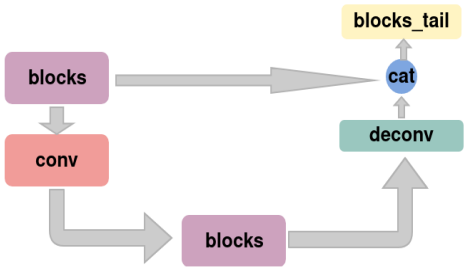



**Figure 9.** The tiny U-Net network.

Similarly, in the classification stage, we also add the idea of spatio-temporal encoding and decoding to solve the loss function for the overlapping parts of the two point clouds. The classification loss is represented by Equation (14). The *cl1* is the semantic score of frame *i-j* of the first point clouds, *cl2* is the semantic score of frame *i-j* of the second point clouds.

$$L_{\text{class local}} = \frac{1}{K}\sum_{k=1}^{K} CE(cl1_k, cl2_k) \tag{14}$$

### *3.3. Spatio-Temporal Encoding and Decoding*

Due to the continuity of point clouds obtained by LiDAR in space and time, in order to preserve spatio-temporal information, we overlay adjacent frame point clouds, starting from the first frame, and overlay adjacent point clouds as network inputs. The first point clouds is the superposition of frames *1* to *N*, the second point clouds is the superposition of frames *2* to *N+1*, and so on. *N-1* frame point clouds are the same between adjacent point clouds. Therefore, during model training, the segmentation results of *N-1* frames between adjacent point clouds should be the same. After obtaining the segmentation results, there are still *N-1* frames with similar results before the point clouds segmentation results. We perform intersection processing on the results of the same frames to obtain more accurate point clouds segmentation results.

## 4. Experiments

### *4.1. Autonomous Robot Hardware Settings*

We build a hardware platform for point clouds and image data collection. The hardware platform includes two-wheel differential chassis, a LiDAR for point clouds collection, four Intel-D435 cameras for image data collection,Jetson AGX Xavier (AGX) for computing. AGX is manufactured by NVIDIA company which is located in Santa Clara, CA, USA. The hardware platform is shown in Figure 10. There is a 16-line LiDAR on the top of the car, and four Intel-D435 cameras are distributed below the LiDAR (front, rear, left and right).

### *4.2. Dataset*

We started the 16-line LiDAR and four Intel-D435 cameras to collect the data required for the experiment in nine scenes, with a total of 8321 point clouds and 33284 images. In addition to the two specific targets of person and car required for our experimental scenes, there are also trees and buildings on both sides of the road. In these point clouds, the point clouds of the target object is very sparse, and thus it is a huge challenge for labeling and instance segmentation.

### *4.3. Experiments and Discussions*

The point clouds obtained by the 16-line LiDAR is sparse, and the frequency of obtaining point clouds is relatively high, with 15 frames of point clouds obtained in one second. The difference between the point clouds of adjacent frames is relatively small, and the problem of sparse point clouds can be solved by overlaying the point clouds of

adjacent frames, while also obtaining more features. We will set the experimental settings to 5, 9, 13, and 15 frames to stack the point clouds to find the optimal number of stacked frames, respectively.

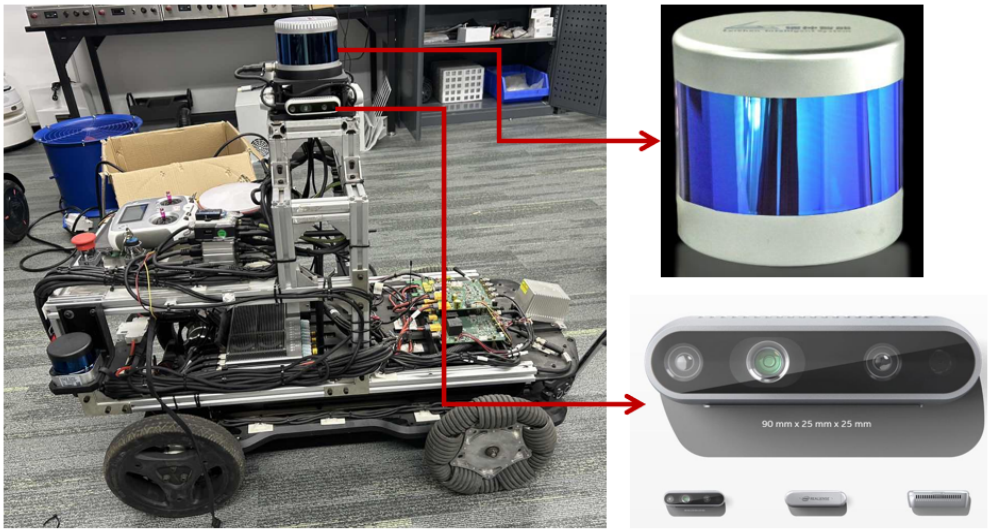



**Figure 10.** The hardware platform used in this study.

The evaluation indicators are standard average precision (AP) and average recall rate (AR). Here, AP_50, AP_25, RC_50, and RC_25 represent scores with IoU (Intersection over Union) thresholds of 0.5 and 0.25, respectively. Similarly, AP and AR represent an average score with an IoU threshold of 0.5 to 0.95, with a step size of 0.05.

We trained and tested these five sets of data separately. The model was implemented using the PyTorch v1.11 (https://pytorch.org/get-started/previous-versions/) deep learning framework and trained using the Adam optimizer. This batch size is set to 2. The learning rate is initialized to 0.001 and scheduled through cosine annealing. The voxel size grouping bandwidth is set to 0.02m and 0.04m, respectively. The score threshold for soft grouping is set to 0.2.

The results of four sets of data training and testing are shown in Tables 1–3. From these tables, as the number of adjacent frames increases, the point clouds gradually becomes dense, and the trained model can gradually segment specific targets in the point clouds. It can be seen that when the number of point clouds reaches 15, the AP of the specific target segmentation result is three times higher than when the number of point clouds is 5. We divided the AP and AR of person and car in the five sets of data instances and drew a Figure 11.

**Table 1.** Comparisons of segmentation results for cars with different framerate.

| **Framerate** | **5** | **9** | **13** | **15** |
|---|---|---|---|---|
| type | car | car | car | car |
| AP_25 | 0.608 | 0.701 | 0.733 | 0.783 |
| AP_50 | 0.477 | 0.487 | 0.656 | 0.704 |
| AP | 0.178 | 0.381 | 0.492 | 0.602 |
| RC_25 | 0.644 | 0.783 | 0.849 | 0.879 |
| RC_50 | 0.525 | 0.655 | 0.797 | 0.807 |
| AR | 0.247 | 0.512 | 0.566 | 0.658 |

**Table 2.** Comparisons of segmentation results for person with different framerate.

| **Framerate** | **5** | **9** | **13** | **15** |
|---|---|---|---|---|
| type | person | person | person | person |
| AP_25 | 0.577 | 0.814 | 0.821 | 0.905 |
| AP_50 | 0.528 | 0.759 | 0.816 | 0.871 |
| AP | 0.365 | 0.757 | 0.773 | 0.828 |
| RC_25 | 0.635 | 0.898 | 0.898 | 0.910 |
| RC_50 | 0.584 | 0.810 | 0.876 | 0.881 |
| AR | 0.339 | 0.809 | 0.805 | 0.842 |

**Table 3.** Comparisons of segmentation results for cars and person with different framerate.

| **Framerate** | **5** | **9** | **13** | **15** |
|---|---|---|---|---|
| type | car + person | car + person | car + person | car + person |
| AP_25 | 0.592 | 0.757 | 0.777 | 0.844 |
| AP_50 | 0.502 | 0.623 | 0.736 | 0.787 |
| AP | 0.272 | 0.569 | 0.633 | 0.715 |
| RC_25 | 0.639 | 0.840 | 0.873 | 0.894 |
| RC_50 | 0.554 | 0.732 | 0.837 | 0.844 |
| AR | 0.339 | 0.660 | 0.685 | 0.750 |

According to the Tables 1–3, and Figure 11, we can observe that the idea of overlaying point clouds of adjacent frames to obtain dense information has a significant impact on the instance segmentation of person. When the point clouds frames are 5, the model can already segment the person in the point clouds. However, due to the fact that the characteristics of the car are not obvious, it is only when the point clouds frames are stacked to 15 that the model achieves good results in car segmentation. Therefore, we adopted the point clouds overlay of 15 adjacent frames as the input of the model. Moreover, the frequency of the 16-line LiDAR we used is exactly 15, and the data we used is the point clouds information obtained by the LiDAR within one second.

According to the control experiment, we choose to stack adjacent 15 frames as the final parameter of the experiment. Then, experiments are set up to verify the effectiveness of the spatio-temporal coding. For the same point clouds, we set up a control experiment, a set of original point clouds instance segmentation models, and a second set of spatio-temporal coding for training. We will quantitatively compare the segmentation performance of our model and softgroup, and compare the segmentation results of car and person in the point clouds. The comparison results are shown in the Tables 4–6. The table compares the results of the point clouds instance segmentation of softgroup and our method, which is a model based on the softgroup network structure with spatio-temporal encoding and decoding. Both models are based on the Pytorch framework, and the learning rate, threshold, and other parameter configurations and the training point clouds were the same. From Table 4, it can be seen that after adding the spatio-temporal encoding and decoding part, the model achieved the better segmentation of car, with varying degrees of improvement in the AP and AR. Due to the fact that the car does not have many feature points compared to person, the addition of some new information is helpful for instance segmentation. From Table 5, correspondingly, for the person in the point clouds, the segmentation performance of the model is already relatively good without the addition of spatio-temporal encoding and decoding. However, when spatio-temporal encoding and decoding are added, the segmentation performance on person is slightly improved. Overall, adding spatio-temporal information has a certain promoting effect on the instance segmentation of the model.

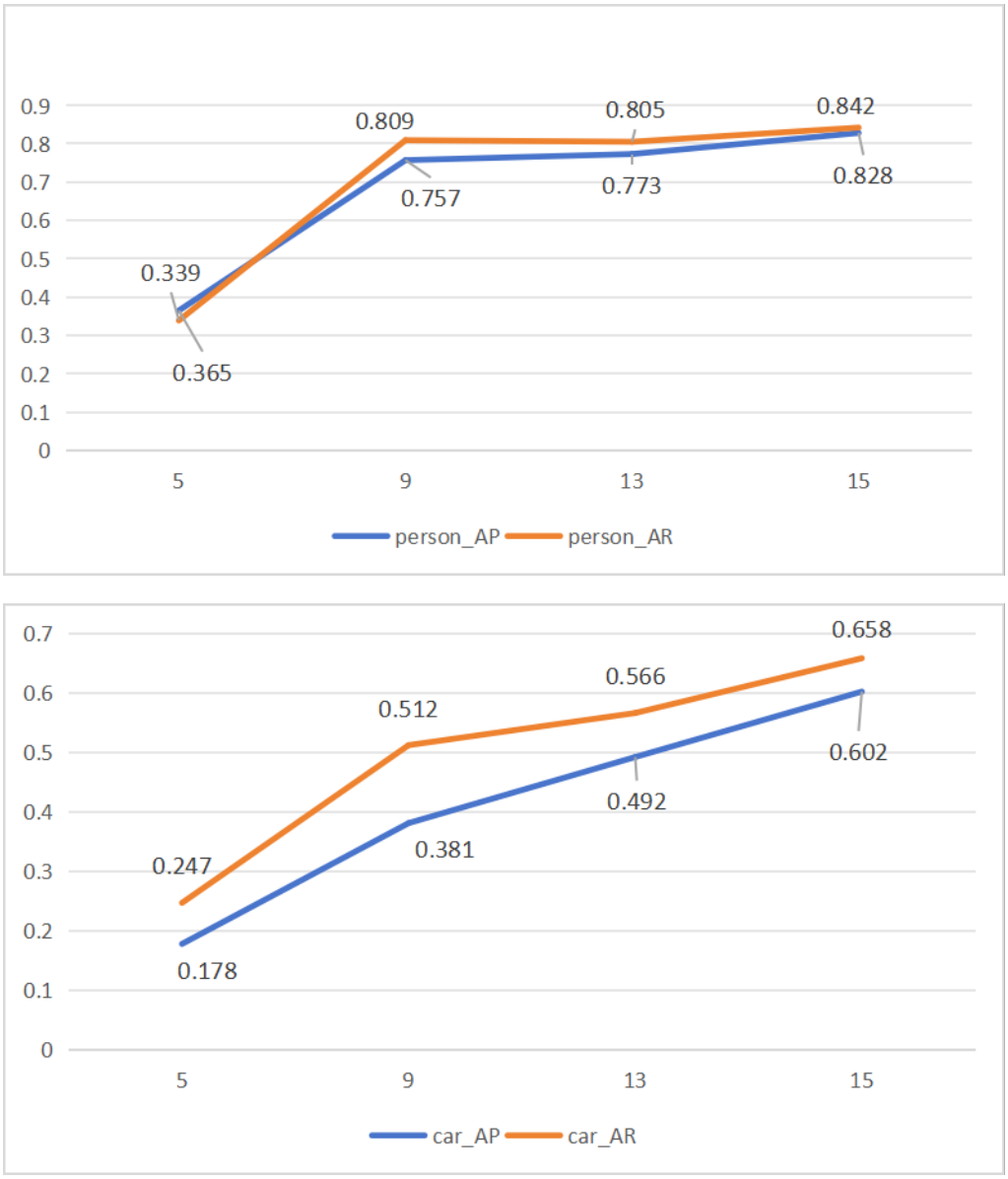



**Figure 11.** Comparisons of person (top) and cars (bottom) for instance segmentation of point clouds with different framerate.

**Table 4.** Comparisons of instance segmentation for cars.

| Method | Softgroup | Ours |
|---|---|---|
| type | car | car |
| AP_25 | 0.783 | **0.810** |
| AP_50 | 0.704 | **0.767** |
| AP | 0.602 | **0.765** |
| RC_25 | **0.879** | 0.832 |
| RC_50 | **0.807** | 0.802 |
| AR | 0.658 | **0.796** |

The test results of the model include the point clouds results of fifteen adjacent frames. We extract the point clouds segmentation results of a single frame and perform post-processing. The same point clouds frames are intersected to obtain more accurate segmentation results. The schematic diagram of the point clouds from a single frame, overlaying adjacent fifteen frames, and the model outputting the results of adjacent fifteen frames, as

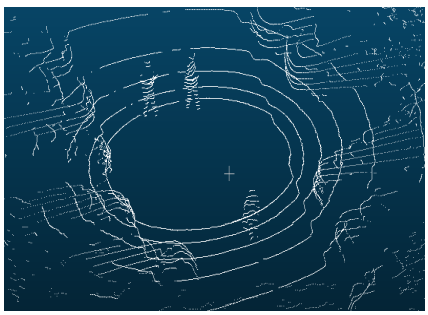
well as the segmentation results of the final split single frame, is shown in the following Figure 12. The different colors in the figure represent different instances, and the background is displayed in black. We compared the three images (c), (d), and (e), the specific comparison of segmentation results is shown in the Figure 13. It can be seen from the figure that our method can segment all instances as much as possible, and the segmentation results without incorporating spatio-temporal encoding and decoding not only have unrecognized car instances, but also have cases of misidentification. Compared with visual and quantitative results, the proposed method for sparse point clouds instance segmentation is feasible.

**Table 5.** Comparisons of instance segmentation for person.

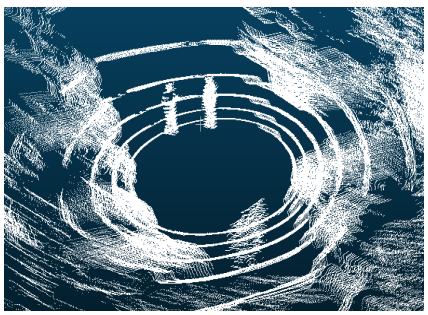
| **Method** | **Softgroup** | **Ours** |
|---|---|---|
| type | person | person |
| AP_25 | 0.905 | **0.924** |
| AP_50 | 0.871 | **0.897** |
| AP | 0.828 | **0.887** |
| RC_25 | 0.910 | **0.933** |
| RC_50 | 0.881 | **0.913** |
| AR | 0.842 | **0.907** |

**Table 6.** Comparisons of instance segmentation for cars and person.

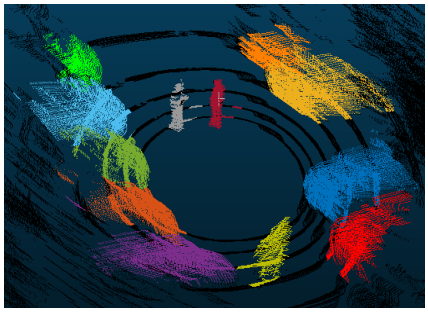
| **Method** | **Softgroup** | **Ours** |
|---|---|---|
| type | car + person | car + person |
| AP_25 | 0.844 | **0.867** |
| AP_50 | 0.787 | **0.832** |
| AP | 0.715 | **0.826** |
| RC_25 | **0.894** | 0.883 |
| RC_50 | 0.844 | **0.857** |
| AR | 0.750 | **0.851** |



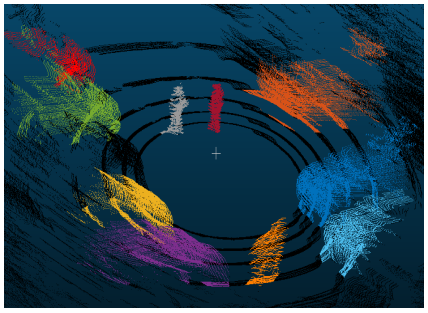
(**a**) Single point clouds



(**b**) Fifteen frames of point clouds

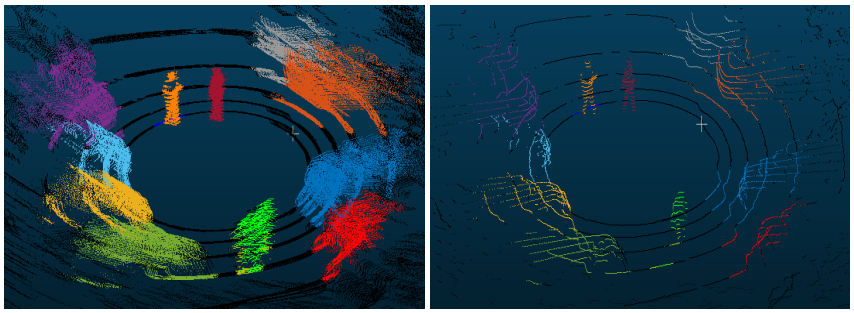


(**c**) Label of point clouds



(**d**) Segmentation of softgroup

**Figure 12.** *Cont.*



(**e**) Proposed Segmentation (**f**) Single frame

**Figure 12.** Point cloud processing and comparison. (**a**) Original single frame of point clouds. (**b**) Point clouds obtained by overlaying fifteen adjacent frames. (**c**) The annotation of the point clouds. (**d**) The instance segmentation of point clouds obtained by the softgroup model. (**e**) The instance segmentation of point clouds obtained by the proposed model. (**f**) Extract a single frame point clouds from (**e**).

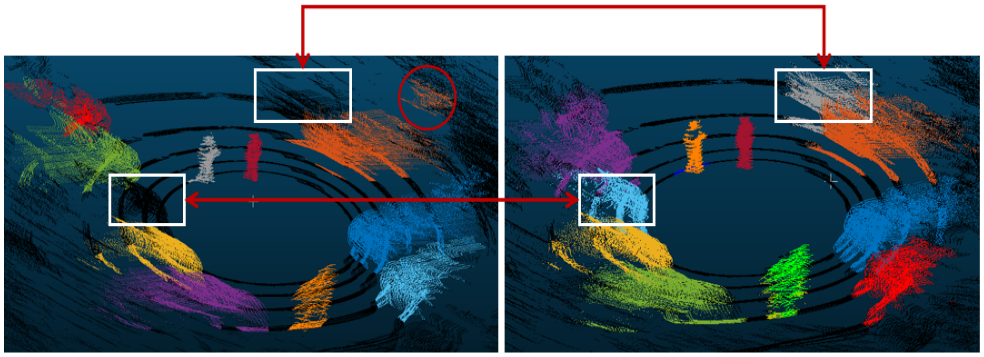


**Figure 13.** Comparisons of instance segmentation between softgroup and the proposed method on point clouds.

## 5. Conclusions

This study mainly focuses on instance segmentation of sparse point clouds. Firstly, in practical applications, most of them are sparse point clouds, but datasets related to sparse point clouds are relatively rare. We built a hardware platform, selected different scenarios, and collected sparse point clouds.

Secondly, due to the sparsity of point clouds, the characteristics of specific targets in point clouds are not obvious, making annotation of sparse point clouds relatively difficult. Therefore, we propose an autonomous annotation scheme for sparse point clouds, utilizing target tracking and segmentation methods of 2D images combined with the relationship between 3D point clouds and 2D mappings. Moreover, we perform autonomous annotation on point clouds. Then, because in practical applications, the point clouds collected by LiDAR has continuity in both space and time, we incorporate spatio-temporal encoding and decoding into the model for point clouds instance segmentation. In order to solve the problem of sparse point clouds, we also overlay adjacent frame point clouds to generate training data and propose a point clouds instance segmentation model that integrates spatio-temporal information.

Finally, we extract the segmentation results of a single frame instance from the model output and process them to obtain the segmentation results of a single frame point clouds. The entire process we propose can be applied to the segmentation, extraction, and filtering of specific targets in dynamic environments, which will help autonomous robots construct map in dynamic environments and avoid the impact of dynamic targets on the construction of map. Because this study introduces spatio-temporal encoding and decoding, it is more effective for segmenting point clouds instances with temporal information, but there is not much improvement in segmenting point clouds without spatio-temporal information. In

the future, we will strive to use the point clouds instance segmentation model to perceive specific objects in the environment for autonomous driving, which will help generate strategies during the autonomous driving process.

**Author Contributions:** Conceptualization, G.W.; Methodology, Y.Y., G.W. and J.L.; Software, N.L.; Validation, Y.Y.; Formal analysis, N.L., S.Z. and J.L.; Investigation, J.L.; Resources, S.Z.; Writing—original draft, N.L., Y.Y., G.W. and L.W.; Writing – review & editing, N.L., S.Z. and L.W.; Project administration, L.W.; Funding acquisition, L.W. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was supported by the National key Research & Development plan of Ministry of Science and Technology of China (Grant No. 2023YFC3605800, 2023YFC3605803).

**Data Availability Statement:** The data presented in this study are available on request from the corresponding author. The data are not publicly available due to privacy.

**Conflicts of Interest:** Authors Sai Zhang, Guodong Wu, Jie Leng and Lihong Wan were employed by the company Origin Dynamics Intelligent Robot Co., Ltd. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

## References

1. Tixiao, S.; Brendan, E.; Drew, M.; Wei, W.; Carlo, R.; Daniela, R. LIO-SAM: Tightly-coupled Lidar Inertial Odometry via Smoothing and Mapping. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, LA, USA, 25–29 October 2020. IEEE: Piscataway, NJ, USA, 2020; pp. 5135–5142.
2. Cadena, C.; Carlone, L.; Carrillo, H.; Latif, Y.; Scaramuzza, D.; Neira, J.; Reid, I.D.; Leonard, J.J. Simultaneous localization and mapping: Present, future, and the robust-perception age. *arXiv* **2016**, arXiv:1606.05830.
3. Kim, G.; Kim, A. Remove, then Revert: Static point clouds Map Construction using Multiresolution Range Images. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, LA, USA, 25–29 October 2020. IEEE: Piscataway, NJ, USA, 2020; pp. 10758–10765. . [CrossRef]
4. Guo, Y.; Wang, H.; Hu, Q.; Liu, H.; Liu, L.; Bennamoun, M. Deep learning for 3d point clouds: A survey. In Proceedings of the IEEE Transactions on Pattern Analysis and Machine Intelligence, Dubai, United Arab Emirates, 29 June 2020; Volume 43, pp. 4338–4364.
5. Lahoud, J.; Ghanem, B.; Pollefeys, M.; Oswald, M.R. 3d instance segmentation via multi-task metric learning. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 9256–9266.
6. Qian, C.; Xiang, Z.; Wu, Z.; Sun, H. RF-LIO: Removal-First Tightly-coupled Lidar Inertial Odometry in High Dynamic Environments. In Proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Prague, Czech Republic, 27 September–1 October 2021; pp. 4421–4428. . [CrossRef]
7. Pfreundschuh, P.; Hendrikx, H.F.; Reijgwart, V.; Dubé, R.; Siegwart, R.; Cramariuc, A. Dynamic object aware lidar slam based on automatic generation of training data. In 2021 IEEE International Conference on Robotics and Automation (ICRA). In Proceedings of the IEEE International Conference on Robotics and Automation, Xi'an, China, 30 May–5 June 2021; pp. 11641–11647.
8. Yoon, D.; Tang, T.; Barfoot, T. Mapless online detection of dynamic objects in 3d lidar. In Proceedings of the 16th Conference on Computer and Robot Vision, Kingston, QC, Canada, 29–31 May 2019.
9. Schauer, J.; Nüchter, A. The Peopleremover—Removing Dynamic Objects From 3-D point clouds by Traversing a Voxel Occupancy Grid. *IEEE Robot. Autom. Lett.* **2018**, *3*, 1679–1686. . [CrossRef]
10. Lim, H.; Hwang, S.; Myung, H. ERASOR: Egocentric Ratio of Pseudo Occupancy-Based Dynamic Object Removal for Static 3D point clouds Map Building. *IEEE Robot. Autom. Lett.* **2021**, *6*, 2272–2279. . [CrossRef]
11. Kim, G.; Kim, A. LT-mapper: A Modular Framework for LiDAR-based Lifelong Mapping. In Proceedings of the 2022 International Conference on Robotics and Automation (ICRA), Philadelphia, PA, USA, 23–27 May 2022; pp. 7995–8002.
12. Pomerleau, F.; Krüsi, P.; Colas, F.; Furgale, P.; Siegwart, R. Long-term 3D map maintenance in dynamic environments. In Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, 31 May–7 June 2014; pp. 3712–3719. . [CrossRef]
13. Milioto, A.; Vizzo, I.; Behley, J.; Stachniss, C. Rangenet++: Fast and accurate lidar semantic segmentation. In 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Kyoto, Japan, 23–27 October 2022; IEEE: Piscataway, NJ, USA, 2019; pp. 4213–4220.
14. Yi, L.; Zhao, W.; Wang, H.; Sung, M.; Guibas, L. GSPN: Generative Shape Proposal Network for 3D Instance Segmentation in point clouds. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; pp. 3942–3951. . [CrossRef]

15. Hou, J.; Dai, A.; Nießner, M. 3D-SIS: 3D Semantic Instance Segmentation of RGB-D Scans. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; pp. 4421–4430. . [CrossRef]
16. Yang, B.; Wang, J.; Clark, R.; Hu, Q.; Wang, S.; Markham, A.; Trigoni, N. Learning Object Bounding Boxes for 3D Instance Segmentation on point clouds. In Proceedings of the 2019 Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019.
17. Liu, S.H.; Yu, S.Y.; Wu, S.C.; Chen, H.T.; Liu, T.L. Learning gaussian instance segmentation in point clouds. *arXiv* **2020**, arXiv:2007.09860.
18. Valada, A.; Vertens, J.; Dhall, A.; Burgard, W. Adapnet: Adaptive semantic segmentation in adverse environmental conditions. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation, Singapore, 29 May–3 June 2017; pp. 4644–4651.
19. Chabot, F.; Chaouch, M.; Rabarisoa, J.; Teuliere, C.; Chateau, T. Deep manta: A coarse-to-fine many-task network for joint 2d and 3d vehicle analysis from monocular image. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2040–2049). *arXiv* **2017**, arXiv:1703.07570.
20. Dewan, A.; Oliveira, G.L.; Burgard, W. Deep semantic classification for 3d lidar data. In 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 3544–3549). *arXiv* **2017**, arXiv:1706.08355.
21. Engelcke, M.; Rao, D.; Wang, D.Z.; Tong, C.H.; Posner, I. Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 1355–1361.
22. Li, B.; Zhang, T.; Xia, T. Vehicle detection from 3d lidar using fully convolutional network. *arXiv* **2016**, arXiv:1608.07916.
23. Reddy, N.D.; Singhal, P.; Krishna, K.M. Semantic motion segmentation using dense crf formulation. In Proceedings of the 2014 Indian Conference on Computer Vision Graphics and Image Processing, ACM, Bangalore, India, 14–18 December 2014; p. 56.
24. Vertens, J.; Valada, A.; Burgard, W. Smsnet: Semantic motion segmentation using deep convolutional neural networks. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, Vancouver, BC, Canada, 24–28 September 2017.
25. Chen, X.; Ma, H.; Wan, J.; Li, B.; Xia, T. Multi-view 3d object detection network for autonomous driving. In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (pp. 1907-1915). *arXiv* **2016**, arXiv:1611.07759.
26. Xu, J.; Kim, K.; Zhang, Z.; Chen, H.W.; Owechko, Y. 2d/3d sensor exploitation and fusion for enhanced object detection. In Proceedings of the Computer Vision and Pattern Recognition Workshops (CVPRW, pp. 764-770), Columbus, OH, USA, 23–28 June 2014; pp. 778–784.
27. Wang, D.Z.; Posner, I. Voting for voting in online point clouds object detection. *Robot. Sci. Syst.* **2015**, *1*, 10–15.
28. Hahnel, D.; Triebel, R.; Burgard, W.; Thrun, S.September). Map building with mobile robots in dynamic environments. In Proceedings of the 2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422), Taipei, Taiwan, 12–17 May 2003; Volume 2, pp. 1557–1563.
29. Meyer-Delius, D.; Beinhofer, M.; Burgard, W. Occupancy grid models for robot mapping in changing environments. In Proceedings of the AAAI Conference on Artificial Intelligence, Atlanta, GA, USA, 11–15 July 2012; Volume 26, pp. 2024-2030.
30. Ronneberger, O.; Fischer, P.; Brox, T. Unet: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention, Proceedings of the MICCAI 2015: 18th International Conference, Munich, Germany, 5–9 October 2015, Proceedings, part III 18*; Springer International Publishing: Berlin/Heidelberg, Germany, 2015; pp. 234–241.
31. Graham, B.; Engelcke, M.; Maaten, L.V. 3d semantic segmentation with submanifold sparse convolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 9224–9232.
32. Chen, S.; Fang, J.; Zhang, Q.; Liu, W.; Wang, X. Hierarchical aggregation for 3d instance segmentation. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021; pp. 15467–15476.
33. Jiang, L.; Zhao, H.; Shi, S.; Liu, S.; Fu, C.W.; Jia, J. Pointgroup: Dual-set point grouping for 3d instance segmentation. In Proceedings of the IEEE/CVF conference on computer vision and Pattern recognition, Seattle, WA, USA, 13–19 June 2020; pp. 4867–4876.
34. Liang, Z.; Li, Z.; Xu, S.; Tan, M.; Jia, K. Instance segmentation in 3d scenes using semantic superpoint tree networks. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021; pp. 2783–2792.
35. He, K.; Gkioxari, G.; Dollár, P.; Girshick, R. Mask r-cnn. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2961–2969.
36. Huang, Z.; Huang, L.; Gong, Y.; Huang, C.; Wang, X. Mask scoring r-cnn. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 6409–6418.