Circular Accessible Depth: A Robust Traversability Representation for UGV Navigation

Shikuan Xie, Ran Song, *Senior Member, IEEE*, Yuenan Zhao, Xueqin Huang, Yibin Li, *Member, IEEE*, and Wei Zhang, *Senior Member, IEEE*

***Abstract*—In this article, we present the circular accessible depth (CAD), a robust traversability representation for an unmanned ground vehicle (UGV) to learn traversability in various scenarios containing irregular obstacles. To predict CAD, we propose a neural network, namely CADNet, with an attention-based multiframe point cloud fusion module, stability-attention module (SAM), to encode the spatial features from point clouds captured by LiDAR. CAD is designed based on the polar coordinate system and focuses on predicting the border of traversable area. Since it encodes the spatial information of the surrounding environment, which enables a semisupervised learning for the CADNet, and thus, desirably avoids annotating a large amount of data. Extensive experiments demonstrate that CAD outperforms baselines in terms of robustness and precision. We also implement our method on a real UGV and show that it performs well in real-world scenarios.**

***Index Terms*—Semisupervised learning, traversability representation, unmanned ground vehicle (UGV) navigation.**

NOMENCLATURE

CADNet

N_l, N_u	Numbers of labeled samples and unlabeled samples.
N	Dataset size, satisfying $N = N_l + N_u$.
N_r, N_φ	Number of partition along r-axis and φ-axis.
r_r, r_φ	Resolutions partitioned along r-axis and φ-axis.
f	Number of historical frames.
i, j, k	Indices for data samples, directions of CAD, and frames.
S_{ik}	Set of points.
N_{ik}	Number of points in S_{ik}.
p_{ik}	Pose of LiDAR.
L_i	Label of the ith sample.
l_{ij}	Index formed depth annotation of the jth direction in L_i.
$\mathbf{F}_p$, F_h, F_a	Polar features, historical feature, and fused feature.
C_p, C_a	Number of channels of $\mathbf{F}_p$ and F_a.

Semisupervised Training

Ψ_i^l, Ψ_i^u	Predicted probability for labeled/unlabeled samples.
Y_i	One-hot coding form of the label L_i.
t	Training epoch.
d	Index of partition along r-axis.
y_{jd}, ψ_{jd}^l, ψ_{jd}^u	dth probability value of the jth direction in $Y/\Psi^l/\Psi^u$.
α, β, λ	Balance factors in $\mathcal{L}_l$, $\mathcal{L}_u$, and $\mathcal{L}$.
g	Distance weight.
b, m	Number/Index of adjacent partition used in $\mathcal{L}_{\text{reg}}$.

Manuscript received 5 June 2023; accepted 28 July 2023. Date of publication 7 September 2023; date of current version 6 December 2023. This paper was recommended for publication by Associate Editor Ayoung Kim and Editor Nancy Amato upon evaluation of the reviewers' comments. This work was supported in part by the National Key Research and Development Program of China under Grant 2021ZD0112002, in part by the National Natural Science Foundation of China under Grant U1913204, Grant 61991411, and Grant U22A2057, in part by the Natural Science Foundation of Shandong Province for Distinguished Young Scholars under Grant ZR2020JQ29, and in part by Project for Self-Developed Innovation Team of Jinan City under Grant 2021GXRC038. *(Corresponding author: Wei Zhang.)*

The authors are with the School of Control Science and Engineering, Shandong University, Jinan 250061, China (e-mail: brucexie1998@gmail.com; ransong@sdu.edu.cn; zhaoyuenan0927@126.com; 201914389@mail.sdu.edu.cn; liyb@sdu.edu.cn; davidzhang@sdu.edu.cn).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TRO.2023.3308780.

Digital Object Identifier 10.1109/TRO.2023.3308780

I. INTRODUCTION

MOST existing work [1], [2] on autonomous navigation focuses on urban self-driving, where only pre-defined and/or structured objects, such as vehicles and pedestrians, are recognized via 3-D detection techniques [3], [4], [5], [6], [7]. Such perception approaches may cause serious accidents due to the lack of a robust traversability prediction in the case that some objects are not present in the training dataset or have irregular shapes that cannot be represented by regular 3-D boxes. In particular, this problem becomes more pronounced for unmanned ground vehicles (UGV), which inevitably confront irregular obstacles such as trunks, lamp posts, barriers, and even downward stairs when performing searching, inspection, and delivery tasks in environments significantly different from a typical highway for autonomous driving.

Bird's-eye view (BEV) semantic maps have recently received increasing attention [8], [9], [10], [11], [12], [13] as traversability representations in autonomous navigation. This kind of representation treats traversability prediction as a semantic segmentation of the surrounding environment. On the one hand, it provides richer and more diverse information on road traversability than most 3-D detection techniques, which improves the security of autonomous navigation. On the other hand, it describes the traversability uniformly in 2-D BEV, which facilitates the subsequent planning and control for UGV. However, BEV semantic

map has two major issues, which significantly affect the robustness of the traversability prediction.

First, the BEV semantic map is not robust in predicting the curb. The root cause of this issue is that a BEV semantic map has no spatial constraint on the border of traversable area. The roads and sidewalks on both sides of the curb are flat and the height difference is not obvious, and thus, their point clouds exhibit similar spatial features. And the BEV semantic map is essentially a pixel-wise classification, which focuses only on the semantic features of different locations but not on the spatial distribution of the border of the traversable area. The key to predicting objects like curbs lies in learning the locations where spatial features change significantly. However, the cross-entropy loss commonly used in BEV methods does not impose any spatial constraint on the border of the traversable area, which leads to difficulty for the network to focus on the locations of feature variations, resulting in imprecise predictions of the border locations.

Second, the BEV semantic map is not robust in predicting negative obstacles, especially downward stairs and cliffs, which lack ground support on one side. This issue arises from the ground truth generation process for the BEV semantic map, which requires concatenating multiframe semantically annotated point clouds [8], [10], [11]. However, the lack of ground support on the other side of the negative obstacles results in the inability of the LiDAR to generate effective laser echoes at these locations. As a result, these vacant locations are represented as void in the BEV labels, and do not contribute to the loss during network training. Fundamentally, the training of BEV for negative obstacle prediction lacks sufficient negative samples, which eventually leads to the inaccurate prediction of the edge locations of negative obstacles.

Aiming at the aforementioned two issues, we propose a robust LiDAR-based traversability representation for UGV navigation, called circular accessible depth (CAD). Same as BEV semantic map, CAD can uniformly represent different kinds of obstacles for predicting the traversability of the surrounding environment. However, instead of predicting the semantic category for each pixel-wise location, CAD predicts the maximum accessible depth in all circular directions centered at the UGV. For the first issue, unlike BEV semantic map based on the Cartesian coordinate system, CAD is based on the polar coordinate system and, thus, directly expresses the distance to the border of the traversable area, which facilitates the introduction of a spatial constraint. With the representation based on the polar coordinate system and the spatial constraint, CAD predicts the border of the traversable area more robustly. For the second issue, CAD focuses only on predicting the location of the edges of negative obstacles rather than the semantic class of each location. Since the locations of negative obstacle edges are accurately labeled in the CAD ground truth, CAD does not suffer from the lack of negative samples. With more reliable annotations, CAD provides more robust predictions for negative obstacles.

We present a neural network, namely CADNet, based on the CAD representation to analyze road traversability for UGV navigation by learning the features with regard to the spatial distribution of LiDAR point clouds. Different from BEV methods that predict the independent probabilities of semantic categories, CADNet predicts the probability distribution of the maximum accessible depth along the radial direction in the spatial dimension. Although the training of CADNet is supervised, benefiting from the inherent property of CAD that represents the maximum accessible depth as a probability distribution, we are able to introduce a semisupervised training to restrict both the entropy and the variance of the probability distribution to make it more concentrated in the spatial dimension. As such, we only need to annotate a small number of labels to inform CADNet what kinds of points potentially represent obstacles or border of traversable area for UGV navigation. Also, the semisupervision enables the learning of a wider sample distribution from a large amount of unlabeled data. Since the ground truth annotations for CAD are easier to be generated than those for BEV semantic map, its applicability to semisupervised learning ensures a high efficiency in real-world applications.

The traversability prediction faces the problems of large LiDAR blind spots and the sparsity of point clouds. Single-frame point cloud lacks sufficient information for the prediction of the traversability, especially facing low and negative obstacles, and thus, it is important to fuse historical information to assist in the prediction. However, dynamic objects in the history frames, such as moving vehicles and pedestrians, may leave trailing shadows in the fused point cloud, which seriously interferes with the traversability prediction. Since the BEV semantic map contains semantic information of dynamic objects, Shaban et al. [10] proposed to eliminate the dynamic features remaining in the memory via an recurrent neural network (RNN) module. In contrast, CAD does not include semantic information about dynamic objects, making it inapplicable to the RNN approach. To cope with this problem, we start from the perspective of feature stability and propose an attention-based multiframe point clouds fusion module, namely stability-attention module (SAM). SAM calculates the correlation coefficients between the features of each frame and the features of the whole sequence as the attention weights. The features of dynamic objects are unstable and have less correlation with the overall features of the whole sequence and, thus, will be assigned smaller attention weights and eliminated in the final fused features. This significantly reduces the interference of historical points of dynamic objects in CAD prediction.

We also develop an easy-to-use CAD annotation tool to facilitate the application of our method in the real world. We demonstrate the feasibility of the proposed CAD in practical UGV navigation experiments by combining some commonly used path planning methods. Extensive experiments show that CAD outperforms baselines in terms of the robustness of the traversability prediction.

Overall, the contributions of this article are threefold.

1) We propose CAD, a robust traversability representation for UGV navigation.
2) We present the CADNet, trained in a semisupervised manner, to extract the features with regard to the spatial distribution of point clouds and predict CAD.
3) In the CADNet, we design a SAM to tackle the interference of dynamic objects in the fusion of point clouds from multiple frames.

II. Related Work

This section first reviews different types of traversability representation for UGV navigation. Then, it briefly reviews some vision-based methods and relevant LiDAR point processing methods including multiframe point clouds fusion methods necessary for traversability prediction.

A. Traversability Representation for UGV Navigation

Usually, UGV navigation is not supported with high-definition maps and confronts diverse obstacles. Thus, designing a flexible and efficient traversability representation subject to the surrounding environment is crucial. Some existing methods [14], [15], [16] attempted to model the surrounding environment as a 3-D occupancy map according to the LiDAR point clouds to calculate the traversability. However, such methods lead to high memory consumption and computational costs, especially in vast outdoor scenarios, making it difficult for the UGV to operate rapidly and flexibly. Recently, the BEV semantic map representation has received more and more attention [8], [9], [10], [11], [12], [13] due to the rich traversability information it provides. The authors in [8] and [9] considered it as a scene completion task to complete sparse semantic LiDAR points into semantic map via inpainting [17]. The authors in [10], [11], [12], and [13] proposed to predict the BEV semantic map directly from LiDAR points or camera images. However, such semantic segmentation approaches consider only the semantic classification of each location, but lack the constraints in the spatial dimension of the edges of the traversable area.

Occupancy grid map (OGM) is also a widely used representation in the field of autonomous navigation, which represents the traversability as occupancy probabilities. This type of representation is similar to the BEV semantic map that represents the space around the ego as 2-D BEV grid map. However, OGM only represents spatial occupancy information without semantics of categories, enabling self-supervised learning from point cloud sequences with no need of large-scale semantic annotations. Ondrúška et al. [18] proposed an RNN-based network that takes laser data sequences as input and trains the network to predict the OGM in future states with self-supervised learning, to solve the problem of object occlusion when using a laser sensor. Mohajerin et al. [19] proposed a self-supervised method to predict the motion trend of dynamic objects in the OGM. Hoermann et al. [20] focused on predicting the long-term motion of dynamic objects via CNN based on the dynamic occupancy grid map (DOGMa) [21]. It divided the occupancy map into static and dynamic regions based on the time series to achieve automatic labeling for self-supervised training of the network. Based on this, Engel et al. [20] proposed using an RNN-based method combined with DOGMa-based detection technology [22] to track objects on DOGMa. Hu et al. [23] calculated occupancy based on segmented nonground LiDAR points through raycasting [24] and trained the network in a self-supervised manner by predicting occupancy in future states. While these OGM-based methods are well-suited for training through self-supervised learning, this representation cannot fully address the problems encountered in UGV navigation, particularly negative obstacles, as these areas are not occupied by any objects. In contrast, the proposed CAD can more uniformly represent various untraversable situations, including occupation and negative obstacles, which makes it more suitable for UGV navigation. There also exists some traditional approaches [25], [26], [27] to calculate the obstacle distance by the geometry of point clouds. For example, Bovbel et al. [25] converted points within a height range to a laser scan, which can only handle obstacles of a certain height. Shang et al. [26] and Larson et al. [27] focused on detecting negative obstacles while had strict requirements for LiDAR setup, limiting their applications.

Recently, 3-D occupancy representation has received more attention [28], [29], [30]. 3-D occupancy can represent the semantics of occupied voxels, and thus, more accurately captures the fine-grained details of obstacles. However, on the one hand, the annotation cost required to build a large-scale 3-D occupancy dataset is high. On the other hand, it is still necessary to compress 3-D occupancy grids into 2-D BEV semantics map to reduce computational complexity and calculate edge constraints to represent the traversability of the scene when applying it for motion planning. Instead, we propose CAD to represent the traversability and present the CADNet to learn the spatial distribution of LiDAR points for a wider range of applications.

B. Vision-Based Methods for UGV Navigation

Vision-based navigation is also an important research field in UGV navigation. Zhu et al. [31] proposed to predict intermediate traversability information from images by manually labeling bottom pixels, and then guiding the motion of UGVs using imitation learning. Huang et al. [32] proposed to train UGV navigation using online reinforcement learning in a simulator, avoiding the need for manual demonstration required by imitation learning. However, these methods require manual labeling of image data. In contrast, Kahn et al. [33] designed a self-supervised offline reinforcement learning-based mobile robot navigation system called BADGR, allowing the network to learn traversability information from the robot's motion events. To address the potential damage caused to the robot during data collection, Kahn et al. [34] proposed an offline reinforcement learning method supplemented with human guidance. This method allowed the robot to learn dangerous information with human participation and avoided the lack of exploration, which often exists in traditional imitation learning. These end-to-end navigation methods based on vision implicitly learn traversability information from images, especially in scenarios, such as off-road scenes with vegetation cover, whose traversability is difficult to represent by any explicit metrics. However, they also face the problem of poor interpretability, and the end-to-end training usually struggles with individual failure cases.

There are also some vision-based navigation methods [35], [36], [37], [38] that can be implemented separately from planning and control. However, vision-based traversability estimation typically needs to project 2-D semantics to a local map, which may lead to a loss of accuracy compared to the LiDAR-based methods, which directly establish the spatial relationship of obstacles around the UGV. Differing from existing

approaches, our method represents traversability explicitly in the form of CAD, enabling targeted supplementation of data and fine-tuning in actual deployment, which increases the reliability.

C. LiDAR Points Processing for UGV Navigation

A key role of the perception module in LiDAR-based UGV navigation is to extract spatial features from the point clouds. Early methods [39], [40] for 3-D object detection often used voxel grid instead of points to represent spatial features. After PointNet [41] was proposed, some methods, such as VoxelNet [4], SECOND [5], and PointPillars [7] proposed to first divide the space into partitions and encode the local features for each partition by PointNet, and then extract the global features by CNN. While most methods divide the point clouds under a Cartesian coordinate system, Zhang et al. [42] and Zhu et al. [43] proposed to describe the spatial features in polar coordinate system to equalize the distribution of points in each partition. Considering that CAD is represented in a circular form, to keep the geometric unity of the spatial features, we designed the perception module in the polar coordinate form. While the structure of feature extraction is also based on a polar coordinate grid, the proposed CADNet, designed for the discrete regression of maximum accessible depth, exhibits more accurate predictions of traversable boundary locations compared to PolarNet and Cylinder3D, designed for point cloud semantic segmentation.

Due to the sparsity of LiDAR point clouds, it is difficult to obtain sufficient spatial information from a single-frame point cloud. Thus, multiframe point clouds fusion is an important part for traversability prediction. Most of the existing multiframe point clouds fusion methods were proposed for 3-D detection [44], [45], [46], [47], where only foreground points were fused. However, such methods are not applicable for the task of traversability prediction where the LiDAR points are not classified as foreground or background. Since CAD does not contain semantic information, RNN-based method [10] used for BEV semantic map is also not applicable. Some other approaches [46], [48] attempted to eliminate the effects of dynamic objects via ray tracing [24]. One disadvantage of such methods is that it is computationally intensive, while a more serious problem is that for many outdoor environments, LiDAR may not be able to produce effective laser echoes, and thus, ray tracing cannot be calculated. Instead, we start from the perspective of feature stability and propose an attention-based multiframe point clouds fusion module to keep only the stable features.

III. Methods

This section consists of the following three parts. First, we introduce the network structure of CADNet and the processing flow of the point clouds. Then, we analyze in detail the SAM of CADNet for multiframe fusion. Finally, we elaborate the semisupervised training process and the loss functions for CADNet. The key symbols in this section are denominated as listed in the Nomenclature.

A. CADNet

CADNet aims to predict the CAD representation of the surrounding environment by extracting the spatial feature of LiDAR point clouds. Since our work is to predict the traversability of the surrounding environment centered on the UGV, the common orthogonal meshing of the point clouds conducted in the Cartesian coordinate system destroys the continuity of the UGV-centered spatial description. By comparison, predicting the probability distribution of the traversability for continuous space along the radial axis in the polar coordinates is more suitable for our task. Thus, the spatial representation and the spatial feature extraction structure of the network are both consistent with the polar coordinate system.

As shown in Fig. 1, CADNet contains of the following three main components: *points to polar feature extraction*, SAM, and *spatial feature extraction*, where SAM will be discussed in detail in the following section.

1) Problem Statement: A training dataset $N = N_l + N_u$ includes N_l labeled samples $\{(S_{ik}, p_{ik}, L_i)|i = 1, \ldots, N_l, k = 0, \ldots, f\}$, and N_u unlabeled samples $\{(S_{ik}, p_{ik})|i = 1, \ldots, N_u, k = 0, \ldots, f\}$, where $S_{ik} \in \mathbb{R}^{N_{ik} \times 4}$ is the kth frame in the ith point cloud and contains N_{ik} points. A point cloud consists of $f + 1$ frames, including one current frame and f historical frames with totally $\sum_{k=0}^{f} N_{ik}$ points. Each row of S_{ik} represents a LiDAR point with four elements including the three-dimensional coordinates of the point and its LiDAR echo intensity. p_{ik} is the pose of the LiDAR sensor when the point cloud S_{ik} is captured. L_i is the CAD label corresponding to the point cloud S_{i0} of the current frame, expressed as $L_i = \{l_{ij}|j = 1, \ldots, N_\varphi\}$, where N_φ indicates that each label contains the accessible depth for N_φ directions and l_{ij} is the depth in the index form for the jth direction. The goal of the perception module is to learn a model to minimize the mean absolute error (MAE) between the prediction and the label.

2) Points to Polar Feature Extraction: The space around the robot, a cylinder centered on the LiDAR with radius R and height H, is equally divided into $N_r \times N_\varphi$ sector pillars as shown in the *Points to Polar Feature Extraction* part in Fig. 1, where N_r and N_φ denote the number of divisions along the r-axis and the φ-axis, respectively. And the resolutions partitioned along these two axes can be denoted as $r_r = R/N_r$ and $r_\varphi = 2\pi/N_\varphi$. The division of the space also determines the shape of the spatial features. The historical points $S_{ik}, k = 1, \ldots, f$ are first transformed into the current coordinate system subject to the transformation T_{ik} between the historical pose p_{ik} and the current pose p_{i0}. Then, all the points are input into the MLP-Maxpool structure proposed by PointNet [41] to construct local spatial features with cylindrical shape. It is noteworthy that since the points belong to different frames, the max pooling should be applied to the points, which belong to the same frame and are located in the same sector pillar. Thus, the MLP-Maxpool structure creates $f + 1$ polar features $F_p^k, k = 0, 1, \ldots, f$ for $f + 1$ frames. And the overall polar features can be expressed as $\mathbf{F}_p = \{F_p^k|k = 0, 1, \ldots, f\}$, whose shape is $(f + 1, C_p, N_r, N_\varphi)$, where C_p is the number of channels.

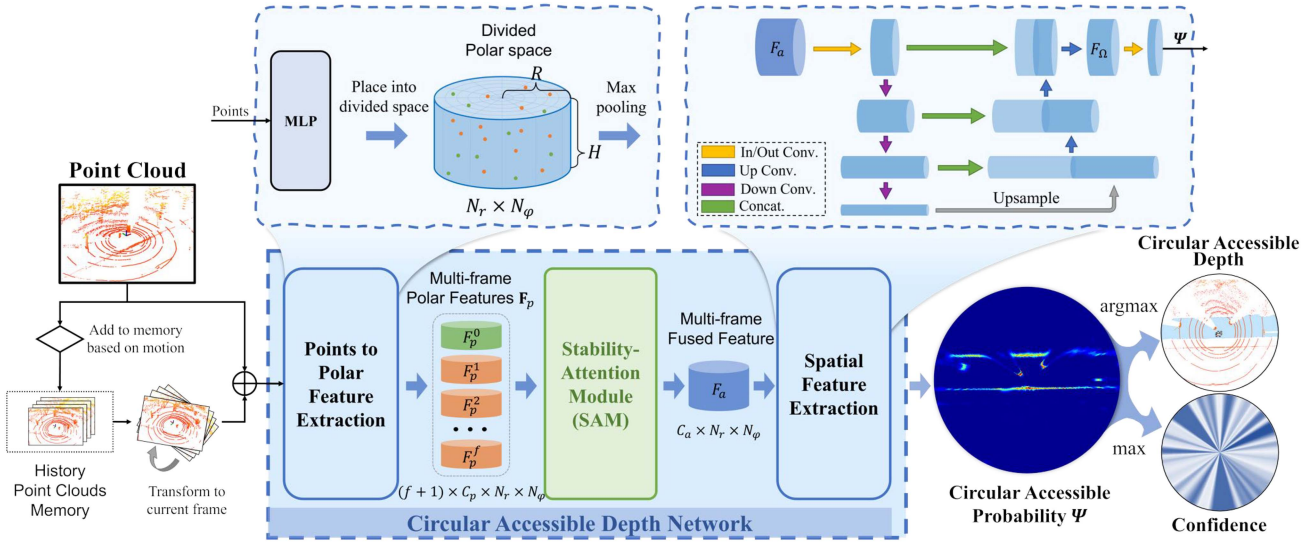


Fig. 1. Pipeline of CADNet including three main components *points to polar feature extraction*, *SAM*, and *spatial feature extraction.*

Then, $\mathbf{F}_p$ is input to the SAM and aggregated as the multiframe fused feature F_a with the shape (C_a, N_r, N_φ). The working principle and computation process of SAM will be discussed in detail in Section III-B.

3) Spatial Feature Extracting: Whether it is the polar feature extracted by the MLP-Maxpool structure or the multiframe fused feature aggregated by SAM, they are all local features with respect to the location where each sector pillar is located. To obtain the final CAD, the features of the whole space need to be encoded. Analogous to [7], [42], we adopt a UNet [49]-like encoder–decoder structure, as shown in the *spatial feature extraction* part in Fig. 1 to encode the adjacent spatial features.

The output of the network is a probability distribution, denoted as Ψ in Fig. 1, with a size consistent with the polar space divided as $(N_r \times N_\varphi)$. This probability distribution is derived by applying `softmax` on the N_r dimension of the output feature of the network. Each value in Ψ is denoted as ψ_{jd}, $j = 1, \ldots, N_\varphi, d = 1, \ldots, N_r$, and satisfies $\sum_{d=1}^{N_r} \psi_{jd} = 1$. Finally, CAD can be obtained through the `argmax` operation:

$$\begin{aligned} \text{CAD} &= \left\{ d_j^{\max} \times r_r | j = 1, \ldots, N_\varphi \right\} \\ d_j^{\max} &= \underset{d}{\operatorname{argmax}}\, \psi_{jd} \end{aligned} \tag{1}$$

where $d_j^{\max}$ corresponds to the index of the maximum probability value for the jth predicted direction, and therefore, needs to be multiplied by the resolution r_r to obtain the distance in meters. The maximum accessible depths of the N_φ predicted directions constitute CAD. Thus, CAD is a one-dimensional vector of length N_φ. And the maximum probability value in each predicted direction is obtained by the `max` operation, which serves as the predicted confidence for that direction

$$\begin{aligned} \text{Conf.} &= \left\{ \psi_j^{\max} | j = 1, \ldots, N_\varphi \right\} \\ \psi_j^{\max} &= \max_d \psi_{jd}. \end{aligned} \tag{2}$$

The predicted confidence of CAD is also a 1-D vector of length N_φ, which can be used as a reference to filter out potential erroneous predictions in practical applications.

B. Stability-Attention Module

1) Dynamic Feature Analysis: Some existing methods utilize point-wise [44], [45], [46] or pixel-wise [10] semantic information to cope with the interference of dynamic objects in multiframe fusion. However, CADNet cannot obtain semantic labels from the CAD ground truth for which points belong to dynamic objects. Thus, it is required for CAD to eliminate the points of dynamic objects in the historical frames without knowing the semantic information of the points. We think about this problem from the perspective of feature stability.

As an example, a ground should be detected by LiDAR as a low and flat point cloud, which we call a ground feature. If a pedestrian passes through the ground at a certain moment, then for this location, the spatial features at this moment are represented as a pedestrian feature. For the overall point cloud sequence at this location, the pedestrian feature is transient, while the ground feature is stable. In this way, how to eliminate the dynamic features depends on how to evaluate the feature stability. We consider the feature stability as the correlation between the feature at this moment and the feature of the whole sequence. Then, a low correlation indicates that this feature is unstable and likely to be the feature of a dynamic object. Hence, to fuse multiframe point clouds for CAD, we design an attention-based module, which calculates the correlation between features.

2) SAM Structure: The architecture of SAM is illustrated in detail in Fig. 2. The polar features $F_p^k, k = 0, 1, \ldots, f$ in the multiframe polar features $\mathbf{F}_p$ are all local features. Due to the sparsity of the point cloud, many locations have empty features, which is not conducive to correlation calculation. Therefore, we first downsample $\mathbf{F}_p$ to obtain $\mathbf{F}'_p$ with more evenly distributed features.

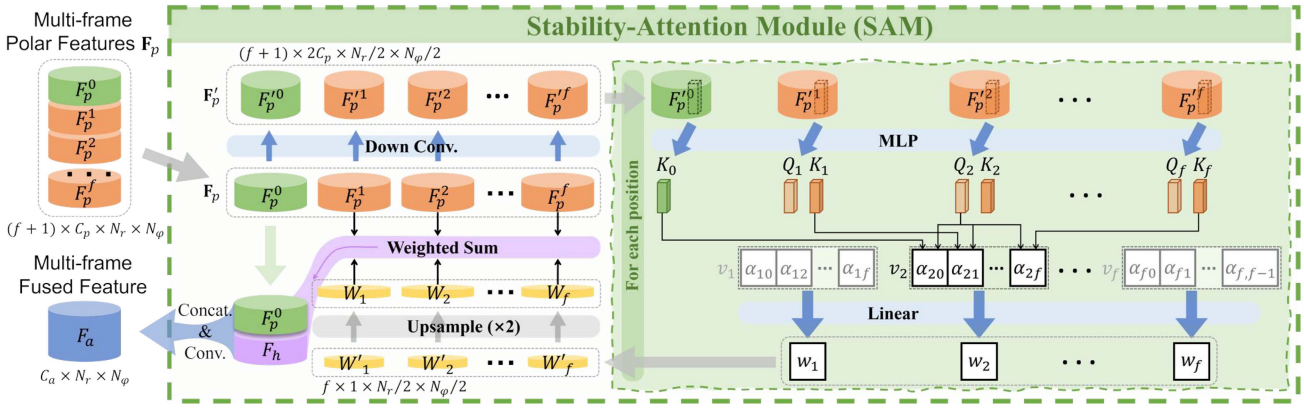


Fig. 2. SAM. SAM fuses multiframe point clouds and eliminates the unstable dynamic features in historical features. The right part of this illustration describes in detail the computing process for each position of different frames, where the computation of correlation vector takes the frame $t-2$ as an example.

For the polar feature $F_p^{'k}$ of each frame, we can obtain two embedded vectors, Q_k and K_k, through MLP. It is worth noting that since the point cloud of the current frame is absolutely accurate, the polar feature $F_p^{'0}$ related to the current frame is only used to be referred by other frames. Thus, $F_p^{'0}$ is only embedded to `key` K_0 but not `query` Q_0. Then, for each historical frame, Q_k is multiplied by K of the other frames to obtain a correlation vector v_k, expressed as

$$v_k = \{\alpha_{kn} | n = 0, \ldots, f \wedge n \neq k\}, \alpha_{kn} = Q_k \cdot K_n. \quad (3)$$

Since it is meaningless to calculate its own correlation, Q_k is not multiplied by K_k, and then the length of the correlation vector v_k is f, while the length of the multiframe feature sequence is $f+1$. Then, the final attention weight w_k is calculated by a learnable linear layer subject to the correlation vector v_k. The weight w_k represents the stability of the features of the kth frame. The more stable the feature is, the greater the value of the weight. Thus, the feature of dynamic objects is effectively removed from the historical features.

w_k is the weight for one location, and the total weights of $F_p^{'k}$ can be represented as W_k', whose shape is $(1 \times N_r/2 \times N_\varphi/2)$. Since we downsample the multiframe polar features F_p^k at the beginning, the attention weights W_k' have to be upsampled to W_k, whose size is the same as that of F_p^k. Then, the historical polar features $F_p^k, k = 1, \ldots, f$ are weighted by $W_k, k = 1, \ldots, f$ and summed up to generate the fused historical feature F_h

$$F_h = \sum_{k=1}^{f} F_p^k \cdot W_k. \quad (4)$$

As mentioned above, to ensure the completeness of the current feature F_p^0, instead of fusing them directly by a weighted summation, we use a convolutional layer to fuse F_p^0 with F_h

$$F_a = \text{conv}\left(\text{concatnate}\left(F_p^0, F_h\right)\right). \quad (5)$$

F_a is the final multiframe fused feature and will be output to the subsequent module for spatial feature extraction.

C. Semisupervised Training

Although the CAD prediction is supposed to be a regression task, in our experiments we found it extremely difficult for the network to directly regress a distance variable in a large continuous space. Therefore, we regard it as a discrete regression task, dividing each direction into N_r parts, and predicting the probability distribution of the space to which the accessible depth belongs. A schematic illustration of the predicted probability distribution of the maximum accessible depth for one prediction direction with a real-world scenario is shown in Fig. 3(a), where the red arrow indicates the ground truth in this direction, which ends in a curb. And the blue histogram indicates the predicted probability distribution for this direction, showing a possible predicting case.

Since our network predicts the probability distribution of the maximum accessible depth along the radial direction on space dimension, we constrain the overall probability distribution to improve the effectiveness of semisupervised learning. The semisupervised loss function is expressed as

$$\mathcal{L}\left(S_\Psi^l, S_Y, S_\Psi^u, t\right) = \frac{1}{N_l}\sum_{i=1}^{N_l}\mathcal{L}_l\left(\Psi_i^l, Y_i, t\right) + \lambda\frac{1}{N_u}\sum_{i=1}^{N_u}\mathcal{L}_u\left(\Psi_i^u, t\right) \quad (6)$$

where Ψ_i^l and Ψ_i^u represent the prediction probability of the ith labeled sample and unlabeled sample, respectively. Y_i is the one-hot coding form of the label L_i, and S_Ψ^l, S_Ψ^u, and S_Y are the set of training samples. t denotes the epoch of training. $\mathcal{L}_l$ and $\mathcal{L}_u$ are the loss functions of the supervised and the unsupervised parts, and λ is the balance factor for the two losses.

The loss functions for the supervised and unsupervised parts of semisupervised learning are described separately below. To show the role of the loss functions in semisupervised learning more clearly, we draw a figure for each loss function to demonstrate its effect on the predicted probability distribution, as shown in Fig. 3(b)–(e). In each subfigure, the left histogram is the predicted probability distribution, and the right one is the result

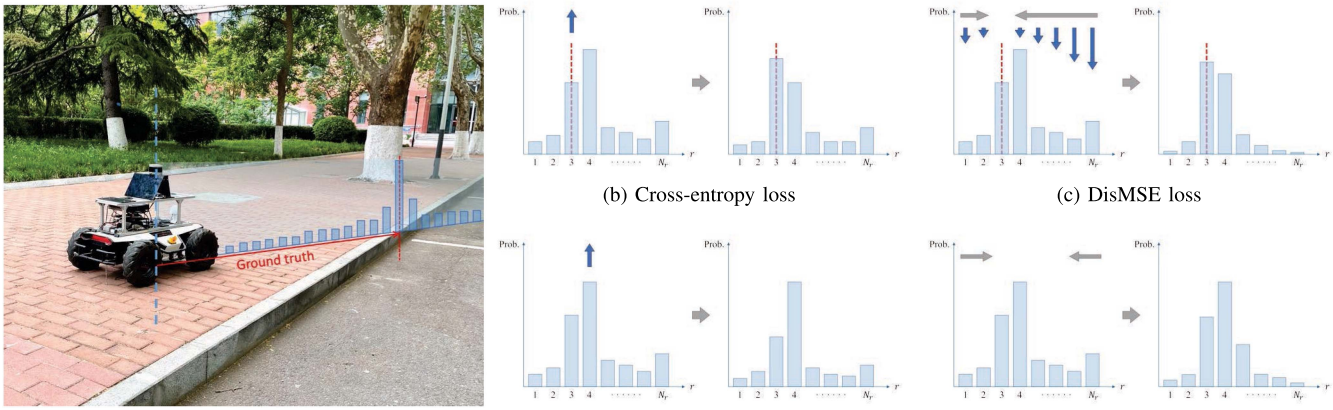


Fig. 3. Illustrations of the CAD probability distribution. (a) Schematic illustration of the predicted probability distribution in a real scenario. (b)–(e) Examples of the effect of the loss functions on the probability distribution in semisupervised training, corresponding to the supervised losses $\mathcal{L}_{\text{ce}}$ (9) and $\mathcal{L}_{\text{dm}}$ (8), and the unsupervised losses $\mathcal{L}_{\text{var}}$ (10) and $\mathcal{L}_{\text{reg}}$ (11), respectively.

of the effect of this loss function on the probability distribution. The horizontal axis represents the r-axis and the vertical axis is the probability value. The red dashed line represents the position of the ground truth, which is missing in Fig. 3(d) and (e) corresponding to the two unsupervised loss functions as the data is unlabeled in unsupervised part.

1) Supervised Training: For the CAD representation, there are correlations between the divided parts along the r-axis: the closer the prediction is to the label, the smaller the loss of the misclassification should be. Thus, for the supervised part, we cannot simply use the cross-entropy loss that only considers the label class. We design a multistage loss called CADLoss as the supervised loss, formulated as

$$\begin{aligned}\mathcal{L}_l(\Psi^l, Y, t) &= \mathcal{L}_{\text{CAD}}\left(\Psi^l, Y, t\right) \\ &= \alpha\mathcal{L}_{\text{dm}}\left(\Psi^l, Y\right) + w_{\text{ce}}(t)\mathcal{L}_{\text{ce}}\left(\Psi^l, Y\right). \end{aligned} \tag{7}$$

The CADLoss contains of the fiollowing two parts: mean square error (MSE) with the distance weight of the probabilities, namely the DisMSE loss $\mathcal{L}_{\text{dm}}$, and the cross-entropy loss $\mathcal{L}_{\text{ce}}$. α is the balance coefficient. $w_{\text{ce}}(t)$ is the weight controlling the time at which the cross-entropy loss is introduced, expressed as $w_{\text{ce}}(t) = \frac{1}{1+e^{-\sigma_1(t-\mu_1)}}$, where σ_1 and μ_1 are hyperparameters. CADNet is first trained under the DisMSE loss to concentrate on the probability distribution of the output around the label, and then the cross-entropy loss is introduced to further increase the probability value of the label class.

The DisMSE and the cross-entropy loss functions are

$$\mathcal{L}_{\text{dm}}\left(\Psi^l, Y\right) = \frac{1}{N_\varphi}\sum_{j=1}^{N_\varphi}\sum_{d=1}^{N_r} g_{jd}\left(y_{jd} - \psi^l_{jd}\right)^2 \tag{8}$$

$$\mathcal{L}_{\text{ce}}\left(\Psi^l, Y\right) = \frac{1}{N_\varphi}\sum_{j=1}^{N_\varphi}\sum_{d=1}^{N_r} -y_{jd}\log\left(\psi^l_{jd}\right) \tag{9}$$

where j and d represent the indexes of angular division and radial division, respectively. $\psi_{jd} \in [0,1]$ and $y_{jd} \in \{0,1\}$ are the prediction probability and label, subject to $\Psi = \{\psi_{jd}|j = 1,\ldots,N_\varphi, d = 1,\ldots,N_r\}$ and $Y = \{y_{jd}|j = 1,\ldots,N_\varphi, d = 1,\ldots,N_r\}$. g is the distance weight to reduce the MSE around the label, and $g_{jd} = 1 - e^{-\frac{(d-l_j)^2}{2\sigma_g^2}}, d = 1,\ldots,N_r$, where l_j is the jth depth ground truth in the index form of label $L = \{l_j|j = 1,\ldots,N_\varphi\}$ and σ_g is a hyperparameter.

It can be seen from (9) that the cross-entropy loss only considers the loss of the label class, i.e., when $y_{jd} = 1$. Consequently, as shown in Fig. 3(b), the cross-entropy loss function does not distinguish between the losses at index 4 and index N_r, while the loss at index N_r should obviously be greater than that at index 4 as it is further away from the label. Instead, the DisMSE loss function takes into account the effect of distance when calculating the loss, so that the probability distribution is concentrated around the label, as shown in Fig. 3(c). In conjunction with these two loss functions, the DisMSE loss function concentrates the probability distribution around the label from the perspective of probability distribution concentration. Then, the cross-entropy loss function improves the accuracy of the prediction from the perspective of entropy, providing an effective and reasonable supervision for the training with labeled data.

2) Unsupervised Training: For the unsupervised part, we make the probability distribution more concentrated by reducing its variance. The variance loss function is

$$\mathcal{L}_{\text{var}}\left(\Psi^u\right) = \frac{1}{N_\varphi}\sum_{j=1}^{N_\varphi}\sum_{d=1}^{N_r}\left(d - \sum_{d=1}^{N_r} d\psi^u_{jd}\right)^2\psi^u_{jd}. \tag{10}$$

However, the probability distribution is relatively uniform at the beginning of the training, resulting in a very large variance, and in the later stage of the training, the variance will become very small. Thus, simply multiplying the variance by a constant factor is not a good choice. Therefore, we first use entropy regularization [50] to obtain an initial probability distribution in the early stage of the training, and then introduce the variance loss to concentrate on the probability distribution in the radial

direction. The entropy regularization loss is

$$\mathcal{L}_{\text{reg}}\left(\Psi^u\right) = \frac{1}{N_\varphi}\sum_{j=1}^{N_\varphi}\sum_{m=1}^{N_r/b} -\psi'_{jm}\log\left(\psi'_{jm}\right)$$

$$\psi'_{jm} = \sum_{d=b(m-1)+1}^{bm} \psi^u_{jd}. \tag{11}$$

Since we only expect an approximate initial probability distribution, we combine the adjacent b positions to calculate the regularization loss, and ψ'_{jm} is the sum of the adjacent b probability values.

Thus, the total unsupervised loss is

$$\mathcal{L}_u\left(\Psi^u, t\right) = \mathcal{L}_{\text{reg}}\left(\Psi^u\right) + w_{\text{var}}(t)\beta\mathcal{L}_{\text{var}}\left(\Psi^u\right) \tag{12}$$

where β is a balance factor and $w_{\text{var}}(t)$ is responsible for controlling when the variance loss is introduced, expressed as $w_{\text{var}}(t) = \frac{1}{1+e^{-\sigma_2(t-\mu_2)}}$, where σ_2 and μ_2 are hyperparameters.

Similar to the supervised part, the unsupervised loss functions also constrain the predicted probability distribution for the unlabeled data in terms of both the entropy and the concentration of the probability distribution. Using entropy regularization alone suppresses the entropy of the probability distribution, but ignores the concentration in the spatial dimension (r-axis), as shown in Fig. 3(d). The variance loss function, on the other hand, concentrates the overall probability distribution, as shown in Fig. 3(e), which is, thus, in line with the spatially continuous nature of the predicted probabilities of CAD.

IV. Experiments

In this section, we demonstrate the robustness of traversability prediction delivered through CAD via extensive comparisons and ablation experiments. We also provide a supplemental video for demonstrating our method on a real UGV and evaluate the feasibility of CAD in real-world scenarios.

A. Perception Experiments

For CADNet, we set the maximum prediction radius to 15 m. Since the spatial feature extraction module has three pairs of encoder–decoder structures, the scale of the spatial feature should be a multiple of 8. We thus set $N_r = 128$ and $N_\varphi = 384$ so that the distance resolution and angular resolution are 0.117 m and 0.938°, respectively, which satisfies the requirements of obstacle avoidance for UGV navigation. We choose to retain four frames of historical point clouds for multiframe fusion, i.e., $f = 4$. It is worth noting that the update of the historical point cloud depends on displacement distance rather than time, and the positions of historical point clouds derived from different frames differ by 1 m. The hyperparameters of weights $w_{\text{ce}}(t)$ and $w_{\text{var}}(t)$ mentioned in (7) and (12) are set as: $\sigma_1 = 0.04$, $\mu_1 = 250$ and $\sigma_2 = 0.1$, $\mu_2 = 100$. And the two balance factors in (7) and (12) are set as $\alpha = 1$ and $\beta = 0.01$. The hyperparameter σ_g of distance weight g_{jd} in (8) is set to 9.

The perception experiments include the following four parts:

1) Comparison with BEV-based methods in terms of the robustness of traversability prediction.
2) Ablation experiments of CADNet including the use of different loss functions and different multiframe fusion methods.
3) Qualitative analysis of the proposed multiframe fusion method SAM.
4) Evaluation of the semisupervised learning scheme using data collected from multiple scenarios.

1) Comparison With BEV Semantic Map: We first quantitatively compare CAD with BEV semantic map on two datasets. The experimental results are listed in Table I.

Dataset: Semantic KITTI [52] is one of the most widely used LiDAR point cloud datasets. However, since its scenarios are for urban autonomous driving, there is a lack of negative obstacles, which often occur in UGV navigation. Thus, we build an environment and collect some data via Isaac Sim simulator [53] with a variety negative obstacles, which more closely resembles the operating scenarios of UGV and better demonstrates the robustness of our method. We conduct experiments with the two datasets: Semantic KITTI and the dataset collected in the simulator, referred to as KITTI and SIM, respectively, in the following and Table I.

We follow the setting of SemanticKITTI [52], choosing KITTI sequences 00–07 and 09–10 for training, and sequence 08 for validation. SIM is also divided into two sequences, serving as the training and the validation sets, consisting of 4200 and 3200 samples, respectively. And all the experimental results listed in Table I are produced in the validation sets.

Metrics: How to quantitatively compare CAD and BEV semantic map is a thorny issue. For CAD, the most intuitive evaluation metric is MAE of the predicted accessible depths with labels. However, the ground truth of BEV semantic map is semantic image but not the border distance of the traversable area. For the BEV semantic map method, we first compute the maximum accessible distance for N_φ directions in the BEV semantic map. This transforms the BEV prediction results into the form of CAD that can be referred to as CAD generated from BEV. However, the computed values of the generated CAD are affected by the resolution of the BEV semantic map. Thus, to ensure a fair comparison, we propose to compare the two representations by prediction accuracy. For a prediction direction, the predicted accessible distance that differs from the ground truth by no more than 0.25 m is considered as a correct prediction, while the resolution of BEV semantic map is 0.1 m. The ratio of the number of correctly predicted directions to the total number of predicted directions N_φ is defined as the prediction accuracy. We also present the MAE of the predicted accessible depths to illustrate the precision of both representations for predicting the border of traversable area. Both of Accuracy and MAE are presented with the following four situations.

1) *Curb:* The boundary between road and sidewalk is defined as curb.
2) *Negative:* Negative obstacles are not included in KITTI. In SIM, the edges of the bridge and pool without guardrails are defined as negative obstacles.
3) *Others:* Other than the above two types of obstacles, such as vehicles, pedestrians, and vegetation.
4) *Total:* All predicted samples.

TABLE I
COMPARISON WITH BEV-BASED METHODS ON KITTI AND SIM

Dataset	Methods	Accuracy ↑				MAE (m) ↓				Params	Time(ms)
		Total	Curb	Negative	Others	Total	Curb	Negative	Others		
KITTI	PillarSegNet [51]	79.48%	73.32%		91.87%	0.266	0.322		0.152	17.42M	30+110
	MASS [11]	76.20%	73.48%		78.83%	0.366	0.355		0.377	17.42M	112+110
	BEVNet [10]	85.01%	84.43%		85.56%	0.341	0.299		0.383	4.93M	170
	BEVNet-B	90.01%	87.83%		94.42%	0.177	0.202		0.127	4.93M	170
	PolarNet [42]+Agg.	82.26%	84.70%	-	77.34%	0.251	0.215	-	0.322	13.64M	200
	PolarNet [42]+Inp. [8]	71.16%	67.72%		78.09%	0.328	0.349		0.286	13.64+3.6M	200+15
	Cylinder3D [43]+Agg.	83.35%	87.12%		75.74%	0.297	0.233		0.425	55.85M	402
	Cylinder3D [43]+Inp. [8]	72.04%	69.32%		77.50%	0.324	0.327		0.317	55.85+3.6M	402+15
	CADNet(Ours)	**92.90%**	**91.87%**		**94.99%**	**0.152**	**0.175**		**0.106**	**4.19M**	**94**
SIM	PillarSegNet [51]	26.13%	35.81%	0.87%	51.98%	4.558	1.300	7.100	3.350	17.42M	**20+31**
	MASS [11]	31.34%	48.68%	1.35%	58.47%	3.660	0.914	5.827	2.610	17.42M	43+31
	BEVNet [10]	58.99%	82.87%	32.61%	77.59%	1.384	0.437	2.175	0.969	4.93M	110
	BEVNet-B	44.73%	86.72%	0.70%	74.45%	2.287	0.415	3.789	1.540	4.93M	110
	CADNet(Ours)	**91.24%**	**92.21%**	**96.45%**	**84.11%**	**0.352**	**0.390**	**0.114**	**0.628**	**4.19M**	54

The bold values represent the best performance in terms of the corresponding metric.

In addition, we also show the number of parameters and the inference time of the models to evaluate the computational performance of the methods, as shown in the last two columns in Table I.

Baselines: We first choose three methods for predicting BEV semantic map as baselines for comparison on both KITTI and SIM datasets, including PillarSegNet [51], MASS [11], and BEVNet [10], where PillarSegNet and MASS predict BEV semantic map using single-frame point cloud, while BEVNet uses fused multiframe point clouds via an RNN structure for prediction. Since the proposed CAD represents only binary traversability, we also test the binary version of BEV semantic map prediction through BEVNet, named as BEVNet-B in Table I. BEVNet, BEVNet-B, and CADNet all take 5-frame point clouds as the input of the network in this experiment.

In addition, we also design four methods for BEV semantic map generation based on point cloud semantic segmentation on the KITTI dataset for comparison. We choose PolarNet [42] and cylinder3D [43] as the point cloud semantic segmentation methods, both of which have a point cloud feature extractor based on the polar coordinate system, similar to CADNet. After obtaining the semantically segmented points, we used two ways to generate BEV semantic map using the semantic segmentation results. The first one is to directly aggregate the point clouds based on the poses of each frame to obtain the dense BEV semantic map, named as +Agg. in Table I. The second one is to generate a sparse BEV semantic map from the semantically segmented points of a single frame and complement it via an inpainting method proposed by Han et al. [8] for traversability prediction of autonomous navigation, and is named as +Inp. in Table I.

The prediction range of each method is set to 15 m, with LiDAR as the origin, i.e., $[-15, 15]$m for both x and y-axis for the Cartesian coordinate-based BEV semantic map, and $[0, 15]$m for r-axis for the polar coordinate-based CAD. In addition, to make a fair comparison with BEV-based methods, all the CADNet models evaluated in this experiment are obtained only via supervised learning.

Results: As shown in Table I, our method significantly outperforms BEV-based methods on both KITTI and SIM. It can be seen that for the methods, which directly predict BEV semantic maps, their prediction accuracy for "Others" is higher than that for "Curb," indicating that the prediction for curb is indeed challenging. This issue can be intuitively observed in the visualization shown in Figs. 4 and 5. The BEV semantic map methods have more accurate predictions for objects, such as vehicles and vegetation as they have discriminative shape features. However, both road and sidewalk are relatively flat and only have small differences in height, which causes difficulty for the BEV methods to learn the differences. In addition, it can be seen from Fig. 4 that the prediction of terrain by the BEV methods is also easily confused with sidewalk and road for the same reason. And even if it is set as a binary prediction, as demonstrated by BEVNet-B, the same issues still persist.

We also demonstrate the visualization of polar features output by PolarNet, Cylinder3D, and our CADNet, as shown in Fig. 6. Although the feature extractor of CADNet and the other two point cloud semantic segmentation methods are based on polar grids, CADNet focuses on learning the features of the curb rather than the features of the buildings behind which are more prominent. Due to the discrete regression-based representation of CAD and the spatial constraints imposed by CADLoss, CADNet provides a more robust prediction even for obstacles with obscure features like curbs. For autonomous driving, obstacles such as curbs with low height may not cause significant damage to vehicles, but for smaller UGVs, accurate prediction of curbs is crucial for navigation safety.

The prediction results for negative obstacles, such as the edge of a pool or a bridge without guardrails, are shown in Fig. 7, and the quantitative evaluation is presented in the SIM group of Table I. Due to no effective LiDAR echoes are produced, the BEV ground truth lacks sufficient semantic annotations, which results in a cluttered prediction, and also leads to the low accuracy for predicting the edge locations of negative obstacle. In comparison, the more reliable ground truth and the introduction of spatial constraints allow the CADNet to focus on predicting the edge locations of negative obstacles, which provides a more robust traversability prediction for UGV to operate in such environments.

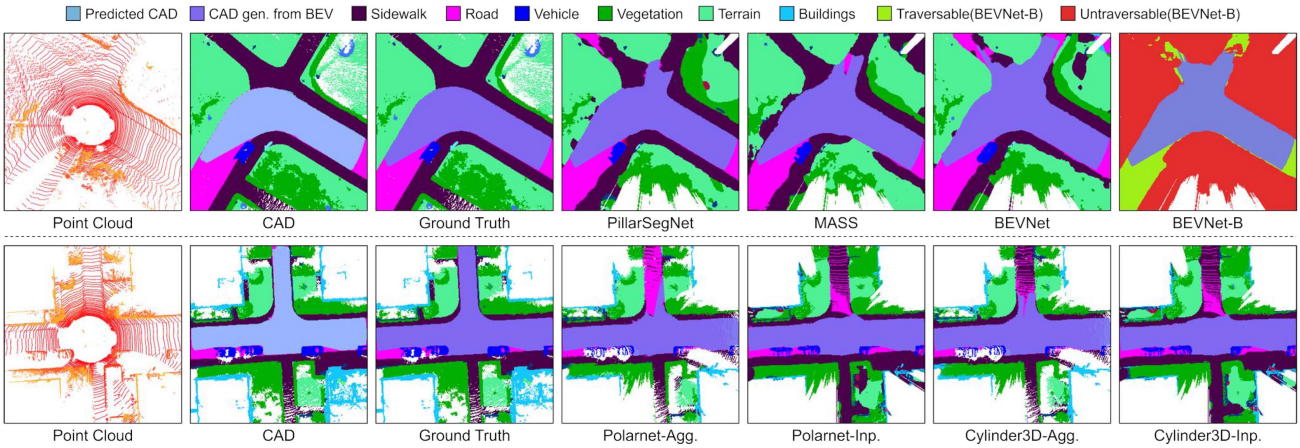


Fig. 4. Visual comparison between CAD and BEV semantic map corresponding to curbs on KITTI. The two rows correspond to two different samples. The first three columns of both rows are the point cloud, the prediction of CAD, and the ground truth. The last four columns of the two rows are the predicted results of the eight comparison baselines listed in Table I. The traversable area represented by CAD is shown in light blue. To represent CAD more clearly, the ground truth and the prediction of CAD are shown together with the BEV semantic map.

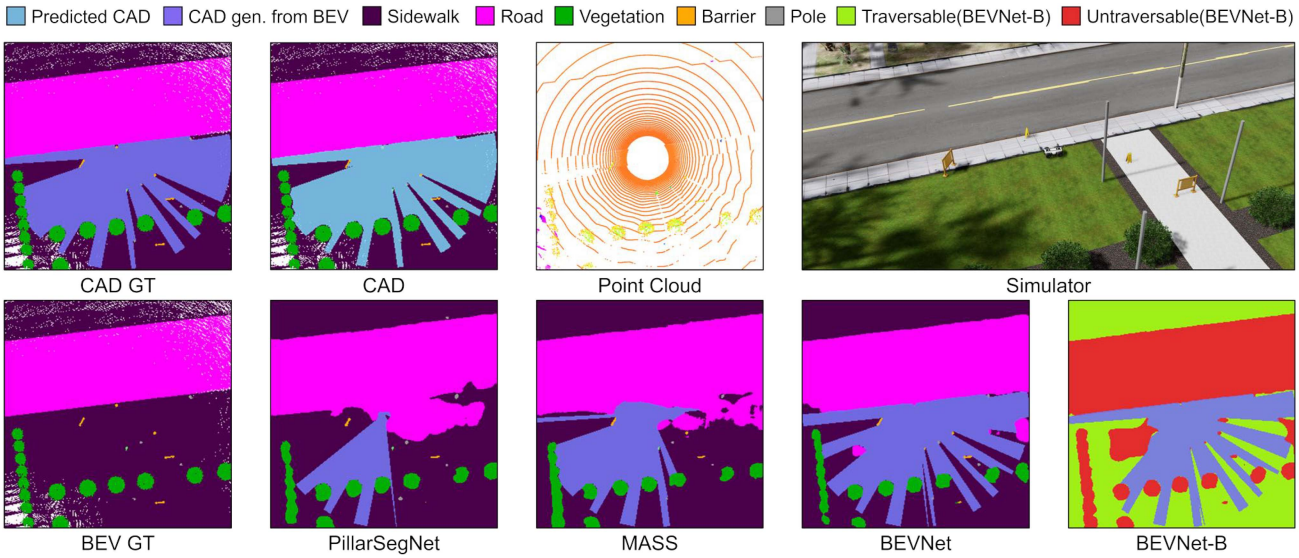


Fig. 5. Visual comparison between CAD and BEV semantic map corresponding to curbs on SIM. Top row: the ground truth of CAD, the prediction of CAD, point cloud, and simulated environment. Bottom row: the ground truth of BEV semantic map, the prediction of PillarSegNet [51], MASS [11], and BEVNet [10], and BEVNet-B.

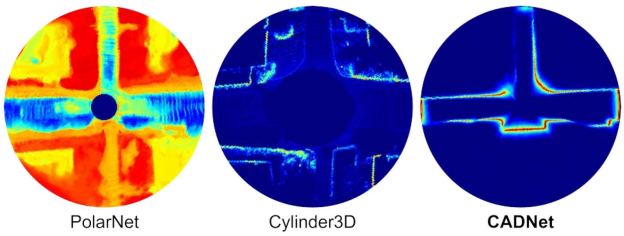


Fig. 6. Visualization of the polar features output by PolarNet [42], Cylinder3D [43], and CADNet. These features are calculated from the sample corresponding to the bottom row of Fig. 4. These visualizations are drawn by averaging the features over the channel dimensions and applying the jet colormap to them.

The number of model parameters and inference time for the competing methods are shown in the last two columns of Table I. All results are obtained on an AMD Ryzen 7 3800X CPU and an Nvidia RTX 2080Ti GPU. PillarSegNet and MASS require the visibility features calculated by ray casting [24], and thus, are recorded in the table in the form of time of inference + time of ray casting. The PolarNet and Cylinder3D combined with aggregation and inpainting were also recorded separately for number of parameters and time. Since the number of LiDAR points in SIM is less than that in KITTI, the computation consumes less time in SIM. Due to the utilization of five frames of point cloud as input, CADNet exhibits slightly higher

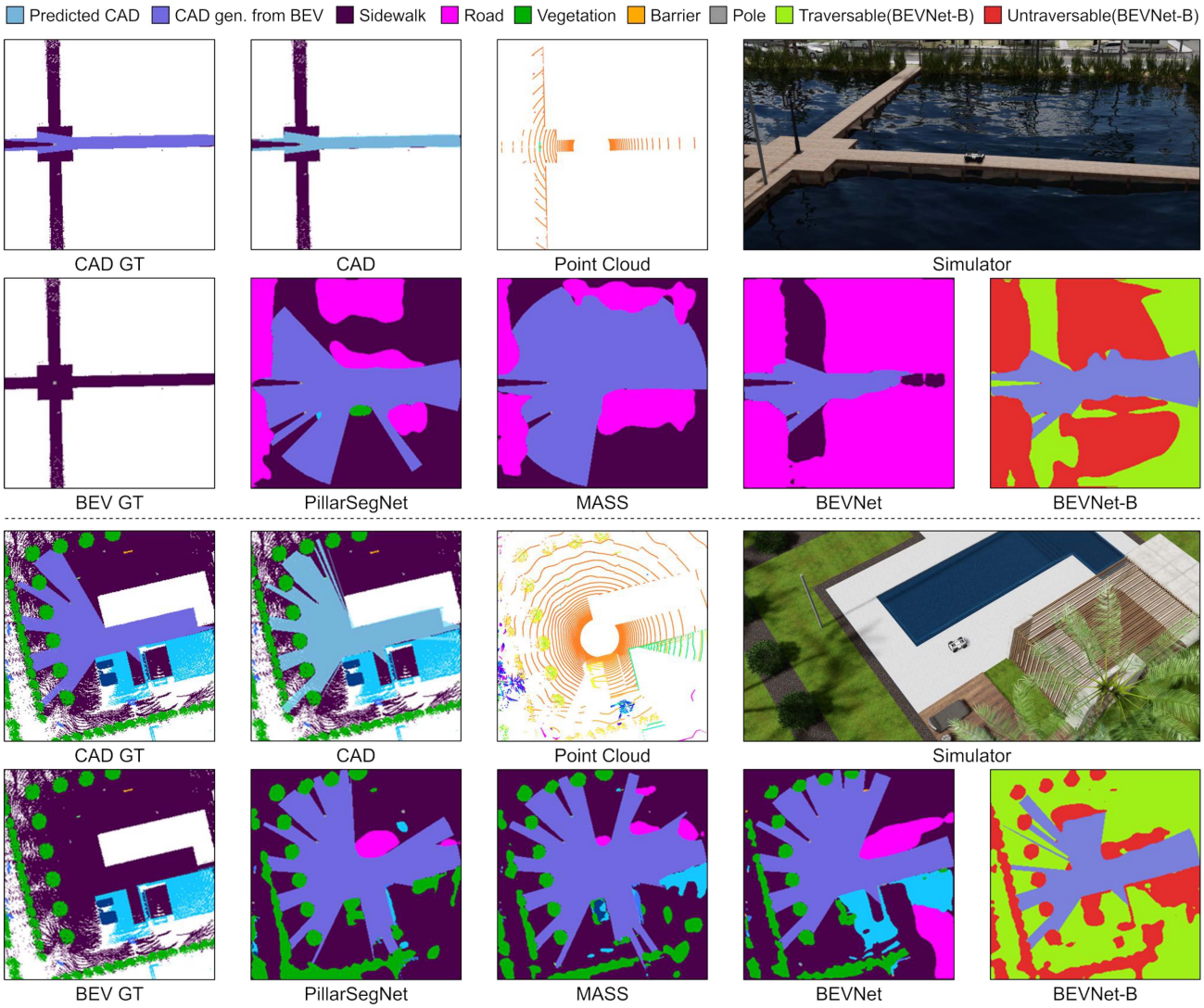


Fig. 7. Visual comparison between CAD and BEV semantic map corresponding to negative obstacles, including bridge and pool scenarios.

time complexity compared to the single-frame PillarSegNet. However, its complexity is significantly lower than that of BEVNet, PolarNet, and Cylinder3D. The small model size and less time consumption facilitate the deployment of our approach in the task of real-world UGV navigation.

2) Ablation Experiments: We evaluate the performance of CADNet with different loss functions and different multiframe fusion modules to demonstrate the effectiveness of the proposed CADLoss (7) and SAM for CAD prediction. The results of the ablation experiments are shown in Table II, where the data are obtained from the KITTI dataset. To evaluate the effect of different loss functions, all the models in this experiment are only trained in a supervised manner.

Baselines and metrics: The baselines used in the ablation study are divided into three categories. First, we compare the proposed CADLoss with the commonly used L1 loss and cross-entropy loss, as well as the two-hot form of cross-entropy loss specifically proposed for discrete regression task [54]. Then, we compare several multiframe fusion methods, including merging all points augmented with timestamp feature [46], concatenating time dimension into channel dimension to fuse temporal feature

TABLE II
RESULTS OF ABLATION EXPERIMENTS

Methods	MAE ↓		Worst-K ↓		IHD ↓	
	Total	Curb	5	20	Num.	Ratio
L1-Loss	0.222	0.175	**2.384**	1.353	-	-
CE-Loss	0.173	0.182	3.088	1.579	-	-
Two-hot [54]	0.175	0.196	3.468	1.738	-	-
Single	0.235	0.274	3.212	1.756	-	-
Merge [46]	0.149	0.162	2.769	1.363	693	1.64%
Concat. [47]	0.163	0.155	2.960	1.460	610	1.44%
SAM-DN	0.153	0.146	2.840	1.376	486	1.15%
SAM-D4	0.147	**0.135**	2.768	1.317	**337**	0.80%
SAM-9F	0.154	0.151	2.889	1.382	414	**0.72%**
CADNet	**0.144**	0.147	2.698	**1.299**	340	0.80%

The bold values represent the best performance in terms of the corresponding metric.

by 2-D CNN [47], and our attention-based SAM. Finally, we conduct ablation experiments on the downsampling operation in SAM, as illustrated by Down Conv. in Fig. 2. SAM without downsampling and downsampling into 1/4 are denoted as SAM-DN and SAM-D4 in Table II, respectively. We also show the impact of additional frames, cf. SAM-9F in the table, which takes 9 frames of point cloud as input. And the final version of CADNet with SAM downsampling into 1/2, with 5 frames of point cloud as input, and is trained with CADLoss is denoted as CADNet in the last row.

We employ three metrics, including MAE, Worst-K, and IHD, to evaluate the performance of different methods:

1) MAE is the MAE between the output CAD and the ground truth annotation. In particular, we show the MAE of the curb prediction to demonstrate the necessity of multiframe fusion.
2) Worst-K is the average of the worst K predictions, and we choose K as 5 and 20.
3) IHD is the interference by historical dynamic objects, as evidenced by two indicators: Number and Ratio. The number of IHD represents how many CAD predictions incorrectly point to the location of historical LiDAR points belonging to the dynamic objects. And the ratio of IHD represents the percentage of such errors in the prediction direction where the historical LiDAR points appear. IHD is used to quantitatively evaluate the immunity of different multiframe fusion modules to the historical points of dynamic objects.

Results: First of all, for the several loss functions shown in the top three rows of Table II, the MAE of the classification method trained with the cross-entropy loss is significantly lower than the regression method trained with the L1 loss. Thus, it is appropriate to consider the task of predicting CAD as a discrete regression task. However, although the cross-entropy loss for classification outperforms the L1 loss for regression in terms of MAE, it performs not so well in terms of Worst-K. This is because the cross-entropy loss does not consider the relationship between categories, while in the CAD, neighboring partitions have stronger spatial correlation than distant ones. Therefore, the CADNet model trained only by cross-entropy outputs some outliers, and suffers from large Worst-K in the quantitative metrics. Hafner et al. [54] proposed to solve the problem of slow training due to reward sparsity in reinforcement learning by changing the training of critic network from a regression task to discrete regression using a two-hot form of label. However, it can be seen from the third row of Table II that the two-hot label does not provide a significant improvement in the prediction of CAD. This is because the two-hot form of the cross-entropy loss function does not serve to concentrate the probability distribution as shown in 3(c), making it unsuitable for CAD prediction. For this reason, we propose CADLoss, which concentrates on the probability distribution of CAD around the label. As shown in the bottom row of Table II, CADLoss further improves the prediction precision by 17.7% and significantly reduces Worst-K, which even outperforms the regression method when K is 20.

For the multiframe fusion, the prediction results of the comparison methods are shown in Fig. 8. First, as shown in the top left image, due to the large blind spot of LiDAR, a single frame point cloud is not sufficient for accurate prediction, especially for low obstacles like curbs. Therefore, multiframe fusion is indispensable for CAD prediction. The "Merge" and "Concat" methods can solve the problem that the "Single" method suffers from. However, they are both influenced by the historical points of dynamic objects, cf. IHD in Table II, and the specific cases are marked in Fig. 8 by black circles. CADNet with the SAM multiframe fusion structure significantly reduces the interference of dynamic objects. For the downsampling setting in SAM, first, we found that downsampling into 1/2 (CADNet) significantly improves the suppression of dynamic obstacles compared to no downsampling (SAM-DN), which demonstrates the effect of the downsampling operation. We also investigated downsampling into 1/4 (SAM-D4), and found that the performance was roughly the same as that of downsampling into 1/2. Therefore, downsampling into 1/2 is sufficient. In addition, we also test the 9-frame version of SAM, and list the results in the penultimate row of Table II. The 9-frame fusion introduces more historical points, resulting in a slight increase in the number of IHD with a lower ratio, which indicates that more frames can increase the stability of SAM. And as shown in the middle row of Fig. 8, 5-frame fusion still produces a few wrong predictions, which are corrected by SAM-9F. However, with respect to the efficiency of deployment, the 9-frame version SAM takes nearly twice as long as the 5-frame version SAM. The number of frames to be fused is related to the number and the density of points in a frame, which needs to be determined empirically, and for KITTI, using 5 frames is a better choice.

3) SAM Qualitative Analysis: In the previous section, we quantitatively compare the impact of different multiframe fusion methods by the historical points of dynamic objects. In the following, we analyze the working mechanism of SAM in real-world scenarios.

We show the performance of SAM in two scenes containing some dynamic objects, as shown in Fig. 9. We overlap the multiframe point clouds as shown in the third column. We can observe that the dynamic objects, such as vehicles and pedestrians, lead to trailing shadows in the images. Then, we multiply each frame of the point cloud image separately by the attention weights generated by SAM and overlap them, as shown in the rightmost column. It can be seen that most points of dynamic objects are eliminated, and the predicted CAD has no errors at the position of the historical points of such dynamic objects as shown in the second column.

We provide an illustration for each case as shown in Fig. 10, where the two images on the left are the real-world photos and the point cloud visualization. And the one on the right is the illustration of the features of the spatial location indicated in the left images, where the green and orange parts are the polar features F_p', the yellow row represents the attention weights $w_k, k = 1, \ldots, f$, and the purple column is the fused historical feature F_h. The horizontal direction indicates different frames,

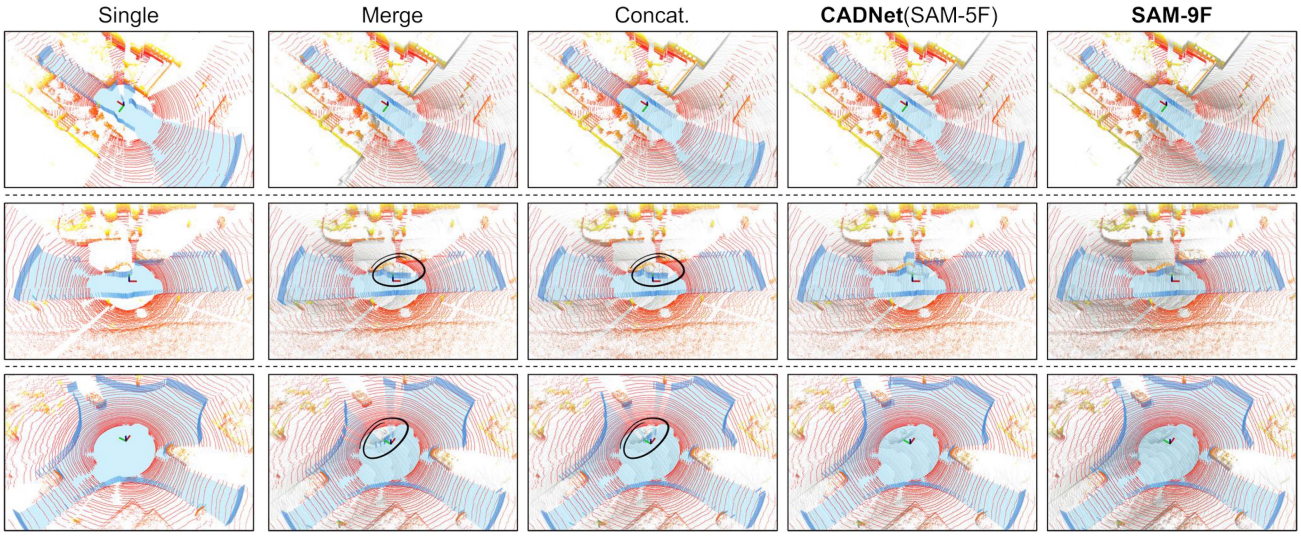


Fig. 8. Visual comparison of different multiframe fusion methods. The red and yellow points are the points of the current frame, while the gray ones belong to historical frames. The blue area represents the predicted CAD. The axes in the center of each image represent the position of LiDAR.

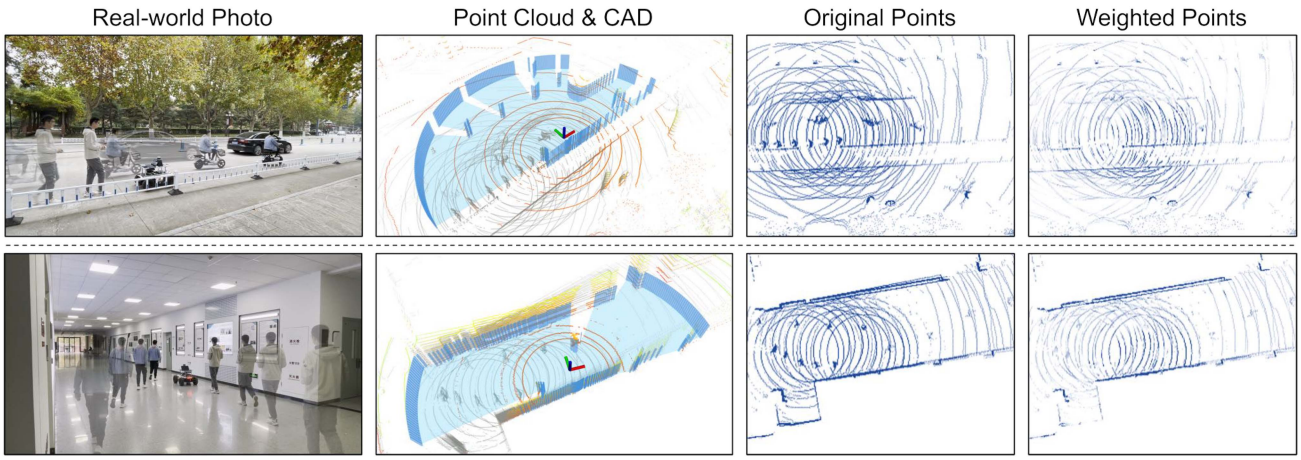


Fig. 9. Real-world tests of multiframe fusion. From left to right are real-world photos, visualization of predicted CAD with point clouds, top view of the original aggregated multiframe point clouds, and the top view of the point clouds weighted by the attention weights. The two columns of images on the right both contain only the point clouds of historical frames.

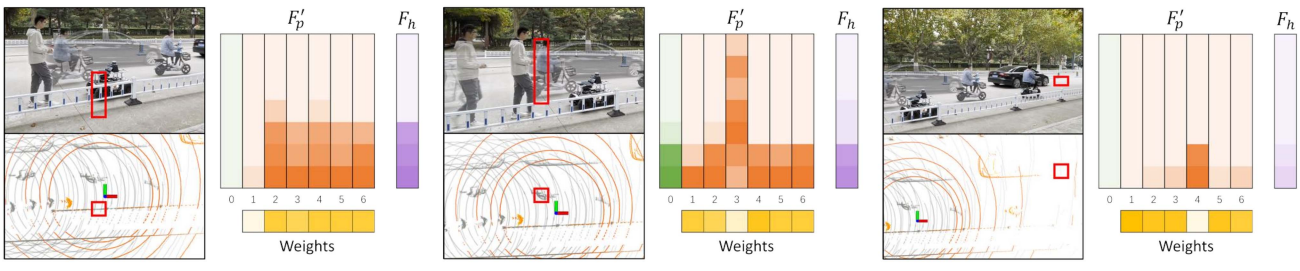


Fig. 10. Three typical cases of SAM. (a) Feature is missing in the current frame. (b) Dynamic obstacles present in the historical frame. (c) Only individual frames have features. In each subfigure, the top left image is the real-world photo, the bottom left one is the point cloud visualization of the corresponding top-down view, and the right image is the illustration of the features and weights in SAM at the location marked with red boxes in the left images.

and we take $f = 6$ as an example. The vertical direction indicates the channel of the feature. The shade of color represents the values of the features or the weights. We evaluate SAM through the following three cases.

1) Spatial feature is missing in the current frame but present in the historical ones. As shown in Fig. 10(a), low obstacles, such as curbs and barriers, are in the LiDAR blind spot belonging to this case. Although the feature of the current frame is empty, the fused feature is still present as the historical features, which completes the missing spatial feature via multiframe fusion.
2) Dynamic object features remain in the history frame. This is the main problem that SAM solves as shown in Fig. 10(b). A pedestrian appears in the third historical frame, while all other frames show the feature of ground. F_p^3 exhibits very different feature from the other frames, thus, its weight w_3 is small and its feature will not be aggregated into the fused historical feature F_h, which enables eliminating the features of dynamic objects.
3) Only individual frames have features. Locations in the distance are only detected by LiDAR at isolated times, as shown in Fig. 10(c). In this case, the only feature becomes an unstable feature, and therefore, is not added into the final fused feature F_h, as evidenced by the lightness of the color at the edges in the last column of Fig. 9, which seems to be the opposite of what we expected. However, this is actually reasonable because we cannot determine from a single frame whether such a feature is static or dynamic. Avoiding fusing such features is the most prudent choice. Since this condition usually occurs at a distance, it does not significantly affect CAD prediction.

4) Evaluation of Semisupervised Learning: Semisupervised learning enables the introduction of a large number of unlabeled samples for training to improve the generalization ability of the model. We evaluate the improvement of CADNet by semisupervised training in terms of generalization ability between samples and between scenarios. We divide the sequences from KITTI into two groups $\mathcal{A}$ and $\mathcal{B}$ based on the scenarios. Group $\mathcal{A}$ contains sequences 1, 2, 3, 4, 6, and 10, which are highway scenarios with wide roads, and has a total of 9135 samples. Group $\mathcal{B}$ contains sequences 5, 7, 8, and 9 with more complex scenarios of 9524 samples. We take one group as the primary group and another as the comparison group, and swap their roles to obtain more comprehensive experimental results. For the primary group, its samples are randomly divided into the following three parts: labeled training set, validation set, and unlabeled training set, with the proportions of 25%, 25%, and 50%, respectively. And the samples of comparison group are all considered as validation set.

The experimental results in Table III are all obtained on the validation sets of both primary and comparison groups. We compute MAE and confidence as the metrics to evaluate the semisupervised learning, where the confidence is the mean of the confidence values (2) for all predicted directions for all samples. First, when the training set includes only labeled data, the model has low prediction confidence, and the performance on the comparison group is worse than that on the primary group because of the gap in scenarios between the two groups. After adding unlabeled samples of primary group P_u for semisupervised learning, the MAE of the model decreases and the confidence improves for both groups. The improvement in confidence demonstrates that our semisupervised loss function (12) does concentrate on the output probability distribution of CADNet. Then, by adding the unlabeled samples of C_u on top of it, the performance of the model improves significantly on the comparison group. It is worth noting that the training set contains only about 15% of the labeled samples at this point, which demonstrate that our semisupervised training procedure effectively improves the generalization ability of CAD with only a small number of labeled samples.

TABLE III
COMPARISON OF SEMISUPERVISED LEARNING SCHEMES

Training set		Primary group		Comparison group	
		MAE ↓	Conf. ↑	MAE ↓	Conf. ↑
PRI $\mathcal{A}$ CMP $\mathcal{B}$	P_l	0.259	35.89%	0.337	32.61%
	P_l+P_u	0.247	62.10%	0.308	58.68%
	P_l+P_u+C_u	0.246	59.94%	0.248	65.88%
PRI $\mathcal{B}$ CMP $\mathcal{A}$	P_l	0.174	50.40%	0.318	49.03%
	P_l+P_u	0.164	70.21%	0.286	69.03%
	P_l+P_u+C_u	0.163	58.80%	0.266	70.80%

P_l, P_u and C_u represent the labeled training set of the primary group, the unlabeled training set of the primary group and the unlabeled training set of the comparison group, respectively.

B. Real-World Navigation Experiments

To demonstrate the feasibility of our method in real-world applications, we deploy CAD on a real UGV and compare it with several UGV navigation approaches.

1) Setups: We deploy our method on an AgileX SCOUT UGV, equipped with WTGAHRS1 IMU, TOP103 GPS, and Velodyne VLP-16 LiDAR. LiDAR is mounted above the center of the UGV at a height of 0.8 m from the ground. The program runs on an on-board computer with an i5-9400H CPU and a GTX-1650 GPU. It spends about 60 ms handling a frame, which satisfies the real-time requirement. In order to deploy our method in real-world environments, we developed and open-sourced a CAD data annotation tool, named CADLabeler.[1] We collected and labeled about 400 samples in real-world environments for training CADNet. We deploy the UGV to navigate along a preset path and implement the timed-elastic-band (TEB) [56] local planner as the downstream local planner for CAD.

Scenarios: We set up four real scenarios, including the outdoor ones "Driveway" and "Sidewalk," and the indoor ones "Stairhall" and "Corridor," to quantitatively evaluate our approach and the baselines. Among them, Sidewalk and Stairhall contain negative obstacles, such as downward curbs and stairs. Each competing method was experimented 12 times with static and dynamic obstacles, respectively, in each scenario.

Metrics: We evaluate the UGV's navigation performance using the following three metrics: success rate, collision, and

[1][Online]. Available: https://github.com/BruceXSK/CADLabeler

TABLE IV
QUANTITATIVE COMPARISON OF DIFFERENT NAVIGATION METHODS THROUGH REAL-WORLD EXPERIMENTS

Senarios		Methods	Success rate		Collision-Ped		Collision-Neg		Collision-Other		Avg. Speed	
			Static	Dynamic	Static	Dynamic	Static	Dynamic	Static	Dynamic	Static	Dynamic
Outdoor	Driveway	Auto. [55]	83.3%	50.0%	2 / 2	4 / 6	-	-	0 / 2	2 / 6	0.91	0.80
		STVL [14]	66.7%	50.0%	0 / 4	2 / 6	-	-	4 / 4	4 / 6	0.79	0.72
		Ours	100%	100%	0 / 0	0 / 0	-	-	0 / 0	0 / 0	0.95	0.82
	Sidewalk	Auto. [55]	0.00%	0.00%	0 / 12	3 / 12	0 / 12	0 / 12	12 / 12	9 / 12	-	-
		STVL [14]	66.7%	16.7%	0 / 4	1 / 10	4 / 4	9 / 10	0 / 4	0 / 10	0.84	0.67
		Ours	100%	100%	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0.90	0.78
Indoor	Corridor	Auto. [55]	83.3%	33.3%	0 / 2	5 / 8	-	-	2 / 2	3 / 8	0.91	0.82
		STVL [14]	100%	66.7%	0 / 0	4 / 4	-	-	0 / 0	0 / 4	0.85	0.65
		Ours	100%	100%	0 / 0	0 / 0	-	-	0 / 0	0 / 0	0.99	0.88
	Stairhall	Auto. [55]	83.3%	58.3%	2 / 2	2 / 5	0 / 2	0 / 5	0 / 2	3 / 5	0.89	0.85
		STVL [14]	58.3%	50.0%	0 / 5	0 / 6	4 / 5	4 / 6	1 / 5	2 / 6	0.76	0.63
		Ours	100%	100%	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0.95	0.84

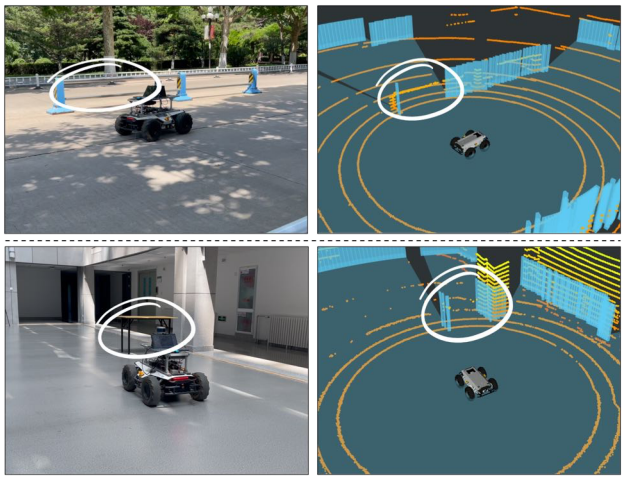


Fig. 11. Examples of incorrect predictions for a guardrail and a table. The left column shows the real-world scenes, and the right one shows the prediction results. The prediction errors are marked with white circles.

average speed. Success Rate is the ratio of the number of times the UGV reaches the navigation goal without collision. Average speed is the average speed of the UGV in all successful navigation tests. We divide the collision situation into three categories according to the objects that produce collisions, which are pedestrians, negative obstacles, and others.

Baselines: We choose two widely used and open-source autonomous navigation approaches, STVL [14] and Autoware [55], as baselines for comparison. STVL is a voxel-based method, generating 2-D costmap based on the voxel grid for the planning module. We implemented the original TEB local planner with it to complete navigation experiments. Then, we tested the Autoware [55] where the perception module is based on the popular PointPillars [7].

2) Results and Analysis: The quantitative results are listed in Table IV, and please refer to the supplemental video for the practical navigation performance.

STVL produces a 2-D costmap by projecting the LiDAR points within a preset height range into a 2-D map. This way of calculating obstacle positions through point cloud geometric relationship has relatively high stability, while the preset height threshold poses some problems for practical applications. If the threshold is too low, the LiDAR points on the ground will be also considered as obstacles, making the UGV unable to drive properly. And if the threshold is too high, the UGV will ignore some low obstacles and collide with them. From the collision-other column of Table IV, it can be seen that STVL has multiple-collisions in the driveway scenario, and most of them are with curbs. In addition, STVL cannot handle the negative obstacles, such as downward curbs in the Sidewalk scenario and the downward stairs in the Stairhall scenario.

Autoware is a comprehensive autonomous driving framework that works on high-definition maps, where the locations of curbs and stairs are predefined. The 3-D detection-based Autoware cannot detect such obstacles either, and thus, it works around the issue of negative obstacles by high precision positioning combined with high-definition maps. And in the event of a positioning deviation, it may still lead to a collision. Also, 3-D detection cannot recognize such objects as trash cans, lamp posts, and tree trunks in the sidewalk scenario, and tables, chairs, and shelves in the corridor scenario, resulting in a number of collisions.

By comparison, our method based on CAD does not require to set fixed thresholds or rely on high definition maps and can be simply applied to UGV navigation. This is because CAD uniformly represents various obstacles, including vehicles, pedestrians, curbs, and negative obstacles, and thus, provides a robust traversability prediction in different environments. In the test scenarios, our method did not collide with any object and achieved the highest success rate.

C. Limitations

Although extensive experimental results demonstrate the effectiveness of our method for UGV navigation, it still has some limitations. First, our method requires a LiDAR sensor, which is relatively expensive. Although cheap depth cameras can also capture point cloud data, their FOV and range are limited and, thus, cannot satisfy the requirement of our method. Second, our method is not good at predicting objects with low thickness. For instance, objects such as a guardrail or a table top, which are parallel to the scanning layer of the LiDAR, typically produces

a small number of LiDAR points, which may result in inaccurate predictions as shown in Fig. 11. In the future, we plan to improve our approach to accommodate diverse sensors and multiple perceptual modalities.

V. Conclusion

In this article, we propose a robust traversability representation, namely CAD, to represent the road traversability for UGV navigation. We design a CADNet with an attention-based multiframe fusion module SAM to extract the spatial features of the LiDAR points for CAD prediction. The CAD representation can be well applied to a semisupervised learning scheme with easily acquired ground truth, which significantly facilitates its deployment in practice. We demonstrate through perception and real-world experiments that CAD is more robust in traversability prediction than the existing BEV semantic maps, Autoware and STVL for UGV navigation. Our current design for CAD is based on the premise that traversability is binary, while this limits the deployment range of CAD. Thus, in the future, we plan to design a multilevel CAD to represent the traversability more appropriately in a wider range of scenarios.

References

[1] C. Badue et al., "Self-driving cars: A survey," *Expert Syst. Appl.*, vol. 165, 2021, Art. no. 113816.

[2] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE Access*, vol. 8, pp. 58443–58469, 2020.

[3] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3D object detection network for autonomous driving," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 6526–6534.

[4] Y. Zhou and O. Tuzel, "VoxelNet: End-to-end learning for point cloud based 3D object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4490–4499.

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

[6] B. Yang, W. Luo, and R. Urtasun, "PIXOR: Real-time 3D object detection from point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 7652–7660.

[7] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "PointPillars: Fast encoders for object detection from point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 12689–12697.

[8] Y. Han, J. Banfi, and M. E. Campbell, "Planning paths through unknown space by imagining what lies therein," in *Proc. Conf. Robot Learn.*, 2020, pp. 905–914.

[9] X. Yang et al., "Semantic segmentation-assisted scene completion for LiDAR point clouds," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 3555–3562.

[10] A. Shaban, X. Meng, J. Lee, B. Boots, and D. Fox, "Semantic terrain classification for off-road autonomous driving," in *Proc. 5th Annu. Conf. Robot Learn.*, 2021, pp. 619–629.

[11] K. Peng et al., "Mass: Multi-attentional semantic segmentation of LiDAR data for dense top-view understanding," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 9, pp. 15824–15840, Sep. 2022.

[12] M. H. Ng, K. Radia, J. Chen, D. Wang, I. Gog, and J. E. Gonzalez, "BEV-SEG: Bird's eye view semantic segmentation using geometry and semantic point cloud," 2020, *arXiv:2006.11436*.

[13] B. Pan, J. Sun, H. Y. T. Leung, A. Andonian, and B. Zhou, "Cross-view semantic segmentation for sensing surroundings," *IEEE Robot. Autom. Lett.*, vol. 5, no. 3, pp. 4867–4873, Jul. 2020.

[14] S. Macenski, D. Tsai, and M. Feinberg, "Spatio-temporal voxel layer: A view on robot perception for the dynamic world," *Int. J. Adv. Robot. Syst.*, vol. 17, no. 2, 2020, Art. no. 1729881420910530.

[15] C. Wang et al., "Autonomous mobile robot navigation in uneven and unstructured indoor environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 109–116.

[16] T. Hines et al., "Virtual surfaces and attitude aware planning and behaviours for negative obstacle navigation," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 4048–4055, Apr. 2021.

[17] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang, "Generative image inpainting with contextual attention," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 5505–5514.

[18] P. Ondruska and I. Posner, "Deep tracking: Seeing beyond seeing using recurrent neural networks," in *Proc. AAAI Conf. Artif. Intell.*, 2016, vol. 30, pp. 3361–3367.

[19] N. Mohajerin and M. Rohani, "Multi-step prediction of occupancy grid maps with recurrent neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 10600–10608.

[20] S. Hoermann, M. Bach, and K. Dietmayer, "Dynamic occupancy grid prediction for urban autonomous driving: A deep learning approach with fully automatic labeling," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 2056–2063.

[21] D. Nuss et al., "A random finite set approach for dynamic occupancy grid maps with real-time application," *Int. J. Robot. Res.*, vol. 37, no. 8, pp. 841–866, 2018.

[22] S. Hoermann, P. Henzler, M. Bach, and K. Dietmayer, "Object detection on dynamic occupancy grid maps using deep learning and automatic label generation," in *Proc. IEEE Intell. Veh. Symp.*, 2018, pp. 826–833.

[23] P. Hu, A. Huang, J. Dolan, D. Held, and D. Ramanan, "Safe local motion planning with self-supervised freespace forecasting," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 12732–12741.

[24] J. Amanatides et al., "A fast voxel traversal algorithm for ray tracing," *Eurographics*, vol. 87, no. 3, pp. 3–10, 1987.

[25] P. Bovbel, M. Hidalgo, and T. Foote, "pointcloud_to_laserscan." 2019. [Online]. Available: https://github.com/ros-perception/pointcloud_to_laserscan

[26] E. Shang, X. An, T. Wu, T. Hu, Q. Yuan, and H. He, "LiDAR based negative obstacle detection for field autonomous land vehicles," *J. Field Robot.*, vol. 33, no. 5, pp. 591–617, 2016.

[27] J. Larson and M. Trivedi, "LiDAR based off-road negative obstacle detection and analysis," in *Proc. IEEE 14th Int. Conf. Intell. Transp. Syst.*, 2011, pp. 192–197.

[28] X. Tian, T. Jiang, L. Yun, Y. Wang, Y. Wang, and H. Zhao, "OCC3D: A large-scale 3D occupancy prediction benchmark for autonomous driving," 2023, *arXiv:2304.14365*.

[29] C. Sima et al., "Scene as occupancy," 2023, *arXiv:2306.02851*.

[30] Z. Li et al., "FB-OCC: 3D occupancy prediction based on forward-backward view transformation," 2023, *arXiv:2307.01492*.

[31] K. Zhu, W. Chen, W. Zhang, R. Song, and Y. Li, "Autonomous robot navigation based on multi-camera perception," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 5879–5885.

[32] X. Huang, H. Deng, W. Zhang, R. Song, and Y. Li, "Towards multi-modal perception-based navigation: A deep reinforcement learning method," *IEEE Robot. Autom. Lett.*, vol. 6, no. 3, pp. 4986–4993, Jul. 2021.

[33] G. Kahn, P. Abbeel, and S. Levine, "BADGR: An autonomous self-supervised learning-based navigation system," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 1312–1319, Apr. 2021.

[34] G. Kahn, P. Abbeel, and S. Levine, "Land: Learning to navigate from disengagements," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 1872–1879, Apr. 2021.

[35] L. Wellhausen, A. Dosovitskiy, R. Ranftl, K. Walas, C. Cadena, and M. Hutter, "Where should i walk? Predicting terrain properties from images via self-supervised learning," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 1509–1516, Apr. 2019.

[36] M. V. Gasparino et al., "Wayfast: Navigation with predictive traversability in the field," *IEEE Robot. Autom. Lett.*, vol. 7, no. 4, pp. 10651–10658, Oct. 2022.

[37] J. Frey, M. Mattamala, N. Chebrolu, C. Cadena, M. Fallon, and M. Hutter, "Fast traversability estimation for wild visual navigation," in *Proc. Conf. Robot.: Sci. Syst.*, 2023.

[38] X. Cai, M. Everett, L. Sharma, P. R. Osteen, and J. P. How, "Probabilistic traversability model for risk-aware motion planning in off-road environments," 2022, *arXiv:2210.00153*.

[39] D. Z. Wang and I. Posner, "Voting for voting in online point cloud object detection," in *Proc. Conf. Robot.: Sci. Syst.*, 2015, vol. 1, pp. 10–15.

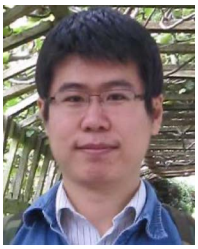
[40] M. Engelcke, D. Rao, D. Z. Wang, C. H. Tong, and I. Posner, "Vote3Deep: Fast object detection in 3D point clouds using efficient convolutional neural networks," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 1355–1361.

[41] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 77–85.

[42] Y. Zhang et al., "PolarNet: An improved grid representation for online LiDAR point clouds semantic segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 9598–9607.
[43] X. Zhu et al., "Cylindrical and asymmetrical 3D convolution networks for lidar segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 9934–9943.
[44] C. R. Qi et al., "Offboard 3D object detection from point cloud sequences," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 6130–6140.
[45] R. Huang et al., "An LSTM approach to temporal 3D object detection in LiDAR point clouds," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 266–282.
[46] P. Hu, J. Ziglar, D. Held, and D. Ramanan, "What you see is what you get: Exploiting visibility for 3D object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 10998–11006.
[47] S. Casas, A. Sadat, and R. Urtasun, "Mp3: A unified model to map, perceive, predict and plan," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 14398–14407.
[48] D. V. Lu, D. Hershberger, and W. D. Smart, "Layered costmaps for context-sensitive navigation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2014, pp. 709–715.
[49] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. image Comput. Comput.- Assist. Interv.*, 2015, pp. 234–241.
[50] Y. Grandvalet and Y. Bengio, "Semi-supervised learning by entropy minimization," in *Proc. 17th Int. Conf. Adv. Neural Inf. Process. Syst.*, 2004, vol. 17, pp. 529–536.
[51] J. Fei, K. Peng, P. Heidenreich, F. Bieder, and C. Stiller, "PillarSegNet: Pillar-based semantic grid map estimation using sparse LiDAR data," in *Proc. IEEE Intell. Veh. Symp.*, 2021, pp. 838–844.
[52] J. Behley et al., "SemanticKITTI: A. dataset for semantic scene understanding of LiDAR sequences," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 9297–9307.
[53] "NVIDIA, Isaac sim," [Online]. Available: https://developer.nvidia.com/isaac-sim
[54] D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap, "Mastering diverse domains through world models," 2023, *arXiv:2301.04104*.
[55] S. Kato et al., "Autoware on board: Enabling autonomous vehicles with embedded systems," in *Proc. IEEE/ACM 9th Int. Conf. Cyber- Phys. Syst.*, 2018, pp. 287–296.
[56] C. Rösmann, W. Feiten, T. Wösch, F. Hoffmann, and T. Bertram, "Trajectory modification considering dynamic constraints of autonomous robots," in *Proc. ROBOTIK; 7th German Conf. Robot.*, 2012, pp. 1–6.

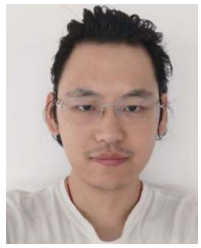
Shikuan Xie received the B.S. degree in automation and M.S. degree in control science and engineering from Shandong University, Shandong, China, in 2020 and 2023, respectively.

His research interests include robot perception, especially the LiDAR perception with deep learning, mobile robot navigation, and autonomous driving.



Ran Song (Senior Member, IEEE) received the B.Eng. degree in communications engineering from Shandong University, Jinan, China, in 2005 and the Ph.D. degree in electronic engineering from the University of York, York, U.K., in 2009.

He has been a Professor with the School of Control Science and Engineering, Shandong University, since 2020. Before his current post, he was a Senior Lecturer with the University of Brighton, Brighton, U.K. His research interests lie in 3-D shape analysis, 3-D visual perception, and 3-D vision for robotics.



Yuenan Zhao received the B.Eng. degree in data science and software engineering from Qingdao University, Qingdao, China, in 2017 and the M.S. degree in control science and engineering from Shandong University, Jinan, China, in 2020. He is currently working toward the D.Eng. degree in artificial intelligence with the Shandong University, under the supervision of Prof. Wei Zhang.

His research interests include computer vision and decision making in autonomous driving system.

Xueqin Huang received the B.S. degree in automation and M.S. degree in control science and engineering from Shandong University, Shandong, China, in 2019 and 2022, respectively.

His research interests lie in deep reinforcement learning and robot visual navigation.

Yibin Li (Member, IEEE) received the B.S. degree in automation from Tianjin University, Tianjin, China, in 1982, the M.S. degree in electrical automation from the Shandong University of Science and Technology, Jinan, China, in 1990, and the Ph.D. degree in automation from Tianjin University in 2008.

From 1982 to 2003, he was with the Shandong University of Science and Technology. Since 2003, he has been the Director of the Center for Robotics, Shandong University, Jinan, China. His research interests include robotics, intelligent control theories, and computer control systems.

Wei Zhang (Senior Member, IEEE) received the Ph.D. degree in electronic engineering from the Chinese University of Hong Kong, Hong Kong, in 2010.

He is currently a Professor with the School of Control Science and Engineering, Shandong University, Jinan, China. His research interests include computer vision and robotics.

Dr. Zhang was a program committee member and a Reviewer for various international conferences and journals.