Particle-Based Instance-Aware Semantic Occupancy Mapping in Dynamic Environments

Gang Chen, *Member, IEEE*, Zhaoying Wang, *Graduate Student Member, IEEE*, Wei Dong, *Member, IEEE*, and Javier Alonso-Mora, *Senior Member, IEEE*

***Abstract*—Representing the 3-D environment with instance-aware semantic and geometric information is crucial for interaction-aware robots in dynamic environments. Nevertheless, creating such a representation poses challenges due to sensor noise, instance segmentation and tracking errors, and the objects' dynamic motion. This article introduces a novel particle-based instance-aware semantic occupancy map to tackle these challenges. Particles with an augmented instance state are used to estimate the probability hypothesis density (PHD) of the objects and implicitly model the environment. Utilizing a state-augmented sequential Monte Carlo PHD filter, these particles are updated to jointly estimate occupancy status, semantic, and instance IDs, mitigating noise. In addition, a memory module is adopted to enhance the map's responsiveness to previously observed objects. Experimental results on the Virtual KITTI 2 dataset demonstrate that the proposed approach surpasses state-of-the-art methods across multiple metrics under different noise conditions. Subsequent tests using real-world data further validate the effectiveness of the proposed approach.**

***Index Terms*—Dynamic environment representation, mapping, semantic scene understanding.**

I. Introduction

SEMANTIC mapping in unknown and unstructured environments [1], [2], [3], [4], [5], [6], [7] aims to represent both geometric and semantic information of elements utilizing onboard sensor data. With the emergence of interaction-aware robots, e.g., robots that can interact with objects or other agents in the environment, it is essential to segment, track and model the individual instances with possible dynamic motions. The shape and motion of each instance should be updated consistently during the interactions to ensure the safety of the robot.

While instance segmentation [8], [9], [10] and tracking [11], [12], [13] have been thoroughly explored in computer vision, the field of instance-aware semantic mapping in dynamic environments is still nascent. This type of mapping poses considerable challenges: first, it demands the ability to manage noise not only from the sensor data but also from instance segmentation and tracking; second, it requires accounting for the dynamic motion of the instances, adding another layer of complexity to the task.

Several methods have been proposed in recent years to realize instance-aware semantic mapping in static environments [5], [6], [7]. However, the representations employed in these works, such as signed distance field or Gaussian kernels, do not account for real-time motion of objects in the environment, causing either missing object or trace noise problems [14], [15]. Alternatively, the occupancy status of dynamic environments can be modeled with particles [15], [16], [17]. While effective in estimating the occupancy status, these methods do not incorporate the semantics and instances of the objects. Moreover, they consider each object to be composed of points with individual motions, thereby disregarding object-level motion and causing non-negligible noise in the occluded areas.

In this work, we build upon the particle map work [15] by incorporating instance-aware semantic information and jointly updating the occupancy status, semantic, and instance labels using particles with an augmented instance state. The proposed approach involves constructing a world model that represents different instances as distinct random finite sets (RFSs) of points. The dynamic motions of each instance are addressed by sharing the same translation and rotation for the points within each set. In accordance with the world model, an state-augmented sequential Monte Carlo probability hypothesis density (PHD) (S^2MC-PHD) filter is proposed to use particles to efficiently estimate the PHD of the RFSs while handling the aforementioned noise. The PHD works as an implicit representation, which is further used to estimate the occupancy status and semantic and instance IDs of each voxel subspace in the map. In addition, we integrate a memory module into the filter to provide a conjecture of the occluded portion of a newly observed object, based on prior observations of objects with the same semantic label. In the experiments, the proposed method not only reaches state-of-the-art semantics and instance estimation performance in dynamic environment mapping but also improves the occupancy estimation performance by leveraging instance information.

The contributions of this article are as follows.

1) An instance-aware point-based world model that enables using the PHD to implicitly represent the occupancy

Received 22 September 2024; accepted 9 December 2024. Date of publication 6 January 2025; date of current version 30 January 2025. This work was supported by the European Union (ERC, INTERACT under Grant 101041863). This article was recommended for publication by Associate Editor O. Mees and Editor J. Civera upon evaluation of the reviewers' comments. (*Corresponding author: Gang Chen.*)

Gang Chen and Javier Alonso-Mora are with the Autonomous Multi-Robots Lab, Department of Cognitive Robotics, School of Mechanical Engineering, Delft University of Technology, 2628 CD Delft, The Netherlands (e-mail: g.chen-5@tudelft.nl).

Zhaoying Wang and Wei Dong are with the State Key Laboratory of Mechanical System and Vibration, School of Mechanical Engineering, Shanghai Jiaotong University, Shanghai 200240, China.

This article has supplementary downloadable material available at https://doi.org/10.1109/TRO.2025.3526084, provided by the authors.

Digital Object Identifier 10.1109/TRO.2025.3526084

status as well as the semantic and instance IDs of the environment.
2) An S^2MC-PHD filter that uses particles with an augmented instance state to efficiently estimate the PHD while handling the sensor and instance noise.
3) Integrating an online memory module into filter-based mapping to enhance the map's responsiveness to previously observed objects.
4) An efficient egocentric instance-aware semantic occupancy map that outperforms state-of-the-art maps in terms of occupancy, semantic, and instance estimation accuracy (Acc.) in the evaluated environments.

In terms of implementation, we also improved the data structure for particle-based mapping [15] to increase efficiency. Our code is available at https://github.com/tud-amr/semantic_dsp_map. Video: https://youtu.be/drSOYzVt2UM.

II. Related Works

A. Mapping in Dynamic Environments

Occupancy mapping in dynamic environments is a challenging task. Traditional mapping methods based on Bayesian updates [18], [19], [20] and signed distance fields [5], [21] usually suffer from the missing object or trace noise problem [14], [15] due to the motions of the dynamic objects. The missing object problem occurs if a dynamic object is observed but not represented in the map and is mistakenly considered as free space. The trace noise problem occurs if a dynamic object has left an area while the map still considers (part of) the area as occupied.

To address the problems, some works [22], [23], [24], [25], [26] detect dynamic objects and eliminate them from the map. Then the dynamic objects are separately represented using the raw point cloud in the current frame, without considering modeling their motion or multiview geometric information. To effectively model the motion and multiview geometry of both dynamic and static objects in the map, enhanced representations of the environment are needed. One popular approach in the field of computer vision is to use neural radiance fields [27], [28] or Gaussian splatting [29], [30] with additional time dimension to model each dynamic object. Although these methods provide photo-realistic rendering of the environment, they require images from different views at different time steps in advance for training and are not suitable for real-time robotic tasks.

Another approach is to use particles as the representation [15], [16], [17], [31]. Compared to regular points in a point cloud, particles can be assigned a velocity vector and a weight, which can be used to model the motion and represent the uncertainty caused by noise. Danescu et al. [16] first introduced the idea of using particles to model the objects and represent the occupancy status of the environment. Nuss et al. [17] improved the particle-based map by introducing RFS and using the PHD to represent the occupancy status. Furthermore, our previous work [15] proposed to update the particles in the continuous space with a dual structure to improve the Acc. and efficiency of the map. These particle-based methods use particles with individual velocities to model the surface points on the objects. Particles representing the points on the same object can have very different velocities. While in the currently observed area, the particles with wrong velocities can be corrected by the measurements, the particles in the occluded area often cause noise in the map due to the "particle false update" problem [15]. Compared to our previous work [15], this article introduces several key improvements. The most significant is the new S^2MC-PHD filter, which handles not only position noise but also object segmentation and tracking noise, whereas the previous filter [15] addressed only position noise. Another important aspect is that the S^2MC-PHD filter uses object-level transformations to model particle motions instead of individual particle velocities, solving the "particle false update" issue identified in [15] and reducing noise in occluded areas. In addition, a memory module is introduced to enhance the map's responsiveness to previously observed objects, and the data structure is optimized to handle object-level information and increase mapping efficiency.

B. Semantic Mapping

Semantic mapping represents the environment with both geometric and semantic information to realize better scene understanding and safer navigation. A number of works have studied semantic mapping in static environments [1], [2], [3], [4], [5], [7], [32], [33], [34], [35], [36], [37]. The representations used in these works include explicit representations, such as point cloud [3], [32], voxel [1], [2], [33], mesh [4], etc., and implicit representations, such as signed distance field [5], [37] and Gaussian kernels [7], [36]. To support the ability of planning interactions, several works [5], [6], [7], [37] added instance-aware information to the map as an additional layer. However, the motions of the instances are not considered, and thus, the maps are not suitable for dynamic environments. Recently, ConvBKI [38], [39] combines the advantages of classical probabilistic algorithms and neural networks to build a semantic voxel map that can be used in dynamic environments, but the instance information is not considered. Realizing instance-aware semantic mapping in dynamic environments is still an open problem.

Occupancy networks [40], [41], [42], [43], [44] are related to the task of semantic mapping in dynamic environments but focus on predicting both the visible and occluded areas in the current frame. They currently lack the ability to retain memory of previously seen areas [45] and are not instance-aware. In addition, they face challenges in generalizing to new environments that differ from their training data. In contrast, our approach belongs to the realm of classical mapping, where both visible areas from the current frame and memory of previously observed areas are considered. Particles are used to facilitate efficient instance-aware semantic mapping in dynamic environments, making it independent of specific scenes and capable of maintaining memory of previously observed areas.

III. Preliminary

Our map is built based on two concepts, RFS and sequential Monte Carlo (SMC)-PHD filter. This section briefly introduces

these concepts as background knowledge. Details can be found in [17], [46], and [47].

A. Random Finite Set

An RFS is a finite set-valued random variable [17]. The number and the states of the elements within an RFS are random but finite. Let X represent an RFS, and $\boldsymbol{x}^{(i)}$ denote the state vector of an element in X. Then, X is expressed as

$$\mathrm{X} = \left\{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(N)}\right\} \tag{1}$$

where $N \in \mathbb{N}$ is a random variable that represents the number of elements in X and is referred to as the cardinality of X. Specifically, when $N = 0$, X is represented as the empty set, denoted by $\emptyset$. A typical application of RFS is in the field of multiobject tracking, where $\boldsymbol{x}^{(i)}$ typically represents the state of an object, and X is a set composed of the states of all tracked objects. The value of N varies as objects appear and disappear within the tracking range.

The first moment of an RFS is the PHD, which is used to describe the multiobject density. The PHD of X at a state $\boldsymbol{x}$ is defined as follows:

$$D_{\mathrm{X}}(\boldsymbol{x}) = \mathbf{E}\left[\sum_{\boldsymbol{x}^{(i)} \in \mathrm{X}} \delta_d(\boldsymbol{x} - \boldsymbol{x}^{(i)})\right] \tag{2}$$

where $\delta_d(\cdot)$ is the Dirac delta function[1] and $\mathbf{E}[\cdot]$ denotes the expectation. The integral of the PHD corresponds to the expected cardinality of X, which can be expressed as

$$\int D_{\mathrm{X}}(\boldsymbol{x})\mathrm{d}\boldsymbol{x} = \mathbf{E}[|\mathrm{X}|] \tag{3}$$

where $|\mathrm{X}|$ is the cardinality of X. Thus, higher PHD integral $\int D_{\mathrm{X}}(\boldsymbol{x})\mathrm{d}\boldsymbol{x}$ suggests there are more elements in the RFS. If the $\boldsymbol{x}$ is constrained in a certain space in the map, it suggests that the space is more likely to be occupied by objects. Based on this property, the PHD can be used to estimate the occupancy status of the environment [15], [17].

B. SMC-PHD Filter

The PHD filter [46] was introduced to predict and update the PHD of an RFS, and was originally designed for multiobject tracking. By updating the PHD rather than tracking and updating the state of each object, the PHD filter is more computationally efficient than Bayesian filters when the number of objects is large. The SMC-PHD filter is a PHD filter that utilizes the SMC method. Specifically, particles are used to represent the PHD. Each particle is usually characterized by a state vector that is composed of position and velocity, as well as an associated weight. Let X_{k-1} represent the RFS at time $k-1$, $\tilde{\boldsymbol{x}}_{k-1}^{(i)}$ denote the state vector of the ith particle, and $w_{k-1}^{(i)}$ denote the weight of the particle. Then, the PHD of X_{k-1} at state $\boldsymbol{x}_{k-1}$ can be approximated as

$$D_{\mathrm{X}_{k-1}}(\boldsymbol{x}_{k-1}) \approx \sum_{i=1}^{L_{k-1}} w_{k-1}^{(i)} \delta_d(\boldsymbol{x}_{k-1} - \tilde{\boldsymbol{x}}_{k-1}^{(i)}) \tag{4}$$

where L_{k-1} is the number of particles at $k-1$. The approximation is utilized due to the statistical nature of the Monte Carlo method. Since achieving a good approximation requires a large number of particles, computational efficiency becomes essential for this filter.

The SMC-PHD filter estimates the PHD through prediction and updating of particles. The prediction step predicts the states changing from $k-1$ to k and is described as

$$D_{\mathrm{X}_{k|k-1}}(\boldsymbol{x}_k) = \sum_{i=1}^{L_{k-1}} P_s w_{k-1}^{(i)} \pi_{k|k-1}\left(\boldsymbol{x}_k | \tilde{\boldsymbol{x}}_{k-1}^{(i)}\right) + \gamma_{k|k-1}(\boldsymbol{x}_k) \tag{5}$$

where P_s is the survival probability indicating the probability that an object persists from $k-1$ to k, and $\pi_{k|k-1}(\boldsymbol{x}_k|\tilde{\boldsymbol{x}}_{k-1}^{(i)})$ is the state transition density of the ith particle. $\gamma_{k|k-1}(\boldsymbol{x}_k)$ is the intensity of the birth RFS. The birth RFS models the newly appeared objects in the tracking range at each time step, and its intensity controls the expected number of new objects.

The update step updates the weight of each particle with measurements $\boldsymbol{z}_k \in \mathrm{Z}_k$ and consequently updates the PHD with the following equations:

$$w_k^{(i)} = \left[1 - P_d + \sum_{\boldsymbol{z}_k \in \mathrm{Z}_k} \frac{P_d g_k(\boldsymbol{z}_k|\tilde{\boldsymbol{x}}_k^{(i)})}{\kappa_k(\boldsymbol{z}_k) + \mathrm{C}_k(\boldsymbol{z}_k)}\right] w_{k|k-1}^{(i)} \tag{6}$$

$$\mathrm{C}_k(\boldsymbol{z}_k) = \sum_{j=1}^{L_k} P_d w_{k|k-1}^{(j)} g_k(\boldsymbol{z}_k|\tilde{\boldsymbol{x}}_k^{(j)}) \tag{7}$$

$$D_{\mathrm{X}_k}(\boldsymbol{x}_k) \approx \sum_{i=1}^{L_k} w_k^{(i)} \delta(\boldsymbol{x}_k - \tilde{\boldsymbol{x}}_k^{(i)}) \tag{8}$$

where P_d is the detection probability that models the probability of an object being detected by the sensor, Z_k is the set of measurements at time k, $g_k(\boldsymbol{z}_k|\tilde{\boldsymbol{x}}_k^{(i)})$ is the likelihood, and $\kappa_k(\boldsymbol{z}_k)$ is the clutter intensity, which represents the density of false measurements in the observation. Equations (6) and (7) express the weight update of the particles. Equation (8) describes the PHD of X_k after the update step. The approximation is used for the same reason as in (4). The number of particles L_k usually differs from L_{k-1} due to the birth and death of particles.

IV. World Model and System Structure

A. World Model

We assume the map space contains only rigid objects, which may be moving. Each object has an instance ID and has a random but finite number of points representing its shape. These points are on the surface of the object and are usually observed as point cloud. Fig. 1(a) illustrates a scene with objects and points on each object. By assuming the rigidity, all the points of an object share the same motion between two time steps. We do not specifically

[1]Dirac delta function: $\delta_d(\boldsymbol{x}) = 0$, if $\boldsymbol{x} \neq \mathbf{0}$; $\int \delta_d(\boldsymbol{x})\mathrm{d}\boldsymbol{x} = 1$.

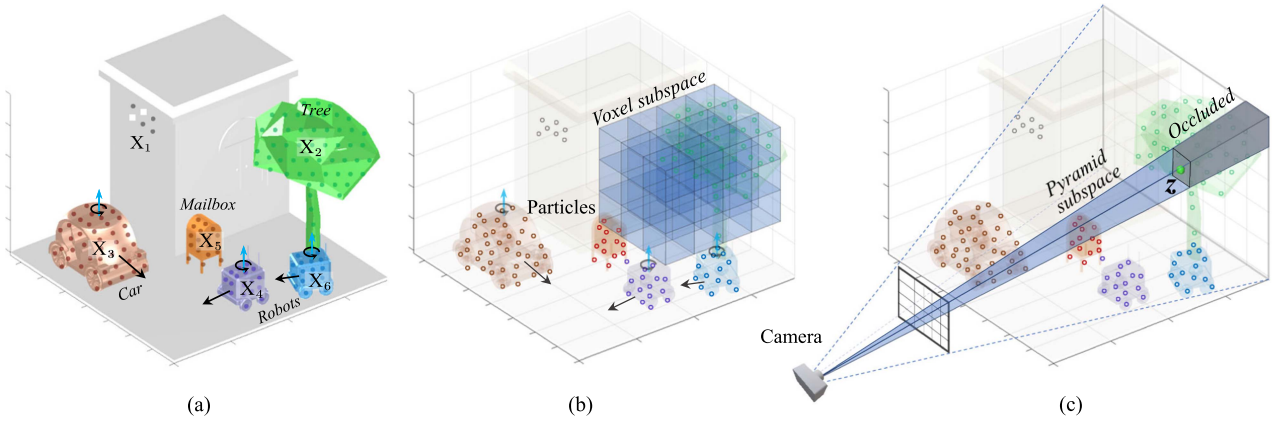


Fig. 1. World model illustration. (a) Example scenario with both objects of interests and background objects. The wall and ground belong to the background object with ID 1. The tree belongs to the background object with ID 2. The car and the two robots belong to the objects of interest with ID 3, 4 and 6, respectively, and are dynamic objects. The mailbox with ID 5 is an example static object of interests. Each object is assumed to be composed of a set of points on its surface. The set of the Ith object is marked as X_I and the points within the set share the same color. (b) Particles (hollow points) that are used to model the PHD of the points. Particles with different IDs are shown in different colors. Particles with the same ID share the same motion. The particles are stored in voxel subspaces [15], which are also used for resampling and occupancy estimation. (c) Camera pinhole model used in this work to formulate the pyramid subspaces [15], which are used to distinguish the observed area and occluded area in the continuous space and to accelerate the update process. The green point is a measurement point in a pyramid subspace. The gray area behind the measurement point is occluded. Only a part of the points in X_1 in (a), voxel subspaces in (b), and pyramid subspaces in (c) are shown for clear illustration.

model nonrigid objects, such as pedestrians, but treat them as rigid objects with a simplified motion model.

To associate each point with its corresponding object and estimate the semantics and instances, we use a state vector composed of the 3-D position and an augmented instance ID dimension for each point. The state vector is expressed as

$$\boldsymbol{x} = [x, y, z, id]^T \tag{9}$$

where $id \in \mathbb{N}^+$ is the instance ID, and x, y, $z \in \mathbb{R}$ are the 3-D position coordinates of the point in the map space, which is a cubic space centered at the location of the robot. While id is a discrete state variable, it can be regarded as a narrowed continuous state variable, and the Dirac delta function $\delta_d(\cdot)$ to calculate the PHD can still be used. Each instance is assigned with a semantic label when the instance is created.

All the points in the map form an RFS, which can be represented as

$$\mathrm{X} = \{\underbrace{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(n_1)}}_{\mathrm{X}_1}, \underbrace{\boldsymbol{x}^{(n_1+1)}, \ldots, \boldsymbol{x}^{(n_1+n_2)}}_{\mathrm{X}_2}, \ldots, \underbrace{\boldsymbol{x}^{(n_1+n_2+\cdots,n_{N-1}+1)}, \ldots, \boldsymbol{x}^{(n_1+n_2+\cdots,n_N)}}_{\mathrm{X}_N}\} \tag{10}$$

where $\mathrm{X}_I, I \in \{1, 2, \ldots, N\}$ under each brace is a sub-RFS composed of the points belonging to instance I. n_I represents the number of points in X_I, and N denotes the total number of instances present in the map. The value of n_I changes at each time step, reflecting the observation of new part of the Ith instance or the removal of part of the instance from the map. Similarly, the value of N changes as objects enter or exit the map's boundaries. For the points within X_I, their associated id is always I.

The instances can be categorized into two groups: instances of interest and background instances. Instances of interest may exhibit either static or dynamic behavior at one moment. Each instance of interest has a distinct ID and is segmented and tracked. The background instances, encompassing both unlabeled objects and labeled objects that are not of interest, are static. In the context of a navigation task, background instances may refer to features like the ground, walls, trees, etc. We assume the same kind of background objects have the same instance ID and belong to one sub-RFS. For example, we can presume that all walls in the map belong to X_1, and all trees in the map belong to X_2.

The measurements of X consist of point clouds accompanied by their respective instance IDs. These points are typically acquired from a stereo/RGB-D camera or Lidar. The instance IDs are derived from instance segmentation and tracking. The measurements have the similar form as X

$$\mathrm{Z} = \{\underbrace{\boldsymbol{z}^{(1)}, \boldsymbol{z}^{(2)}, \ldots, \boldsymbol{z}^{(m_1)}}_{\mathrm{Z}_1}, \underbrace{\boldsymbol{z}^{(m_1+1)}, \ldots, \boldsymbol{z}^{(m_1+m_2)}}_{\mathrm{Z}_2}, \ldots, \underbrace{\boldsymbol{z}^{(m_1+m_2+\cdots,m_{M-1}+1)}, \ldots, \boldsymbol{z}^{(m_1+m_2+\cdots,m_M)}}_{\mathrm{Z}_M}\} \tag{11}$$

where $\mathrm{Z}_J, J \in \{1, 2, \ldots, M\}$ under each brace is a sub-RFS composed of the measurement points belonging to object J, m_J is the number of points in Z_J, and M is the number of objects observed in the measurements of this time step. Each point in the measurement is also composed of a 3-D position and an instance ID

$$\boldsymbol{z} = [x, y, z, id]^T. \tag{12}$$

Note that due to the sensor's limited field of view (FOV) and inevitable occlusions between objects. Only a portion of objects and points in X has observations in Z. Furthermore, the

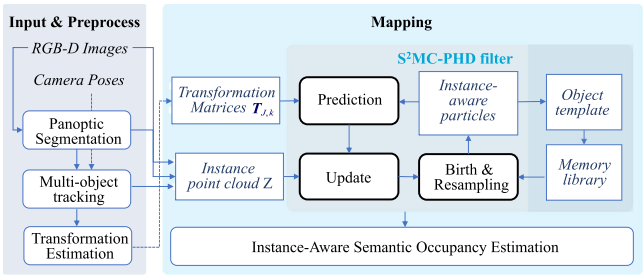


Fig. 2. System structure. The left side shows the input and preprocessing modules, which generate data composed of two parts: Transformation matrices and instance point cloud. The generated data, which contains noise, is used in the mapping on the right side. The core of the mapping part is the S^{2}MC-PHD filter.

measurements contain noise in both position and instance ID. The noise in position comes from the sensor measurement noise. The noise in instance ID arises from missing instances, merging instances, misclassification, etc., in instance segmentation, and incorrect data association in tracking, and manifests as the incorrect instance IDs. When updating the map, noise in position and instance should both be addressed.

Our map employs particles with IDs to approximate the PHD of X, and Z is utilized to update the particles. Compared to using particles to track and update the state of each point individually, modeling the PHD is more computationally efficient and can better handle occlusions. We illustrate the particles in Fig. 1(b) and (c) with hollow points. Following our previous work [15], we store the particles in voxel subspaces and use pyramid subspaces to differentiate between the currently observed and occluded areas in the continuous space. To realize efficient pyramid subspace division when the robot is moving, this article adopts the Pinhole model for pyramid subspace division, as illustrated in Fig. 1(c). Details about the subspace divisions can be found in Section VI-A. The following presents the system structure and the filter used to update the PHD in the observed area of each time step.

B. System Structure

The system structure is shown in Fig. 2. The modules on the left detail the input and preprocessing requirements. The inputs are RGB-D image pairs from stereo/RGB-D cameras[2] and the corresponding poses of the camera. The preprocessing modules encompass panoptic segmentation, multiobject tracking and transformation estimation. At each time step, the panoptic segmentation module [8], [9], [10] segments the instances in the image, and the multiobject tracking module [48], [49] tracks the instances of interest in the image sequence. Recent advancements in 4-D panoptic segmentation have proposed an alternative solution for obtaining both segmentation and tracking results using a single network [50]. The transformation estimation module estimates the motion of instances in tracking between two frames and can be realized by joint localization and object motion estimation [51], [52], [53] or pose tracking methods [11], [12], [13]. We consider these preprocessing modules off-the-shelf and do not delve into them in this article. However, these modules inevitably introduce noise to the measurements, which should be considered when updating the map. In Section VI, we present a practical implementation solution for the modules. The output of the preprocessing modules encompasses the measurement point RFS Z, and the estimated transformation matrices of objects of interest between two consecutive frames.

An S^{2}MC-PHD filter, detailed in the next section, is proposed to update the PHD of X at the sub-instance level, and estimate the occupancy status of the map. This filter mainly consists of particle prediction, update, and birth and resampling modules. The prediction module adopts the transformation matrices of the objects of interest to predict the new positions of the points. The update module uses the measurements Z to update the PHD represented by particles. With the updated PHD, the occupancy status and instance labels of the map can be estimated. The birth and resampling module generates new particles and prevents degeneracy. The memory module is introduced to enhance the map's responsiveness to previously observed objects and to provide a conjecture of the occupancy status in the occluded portion of an object.

V. S^{2}MC-PHD Filter

The S^{2}MC-PHD filter is built upon the world model described in Section IV-A and uses particles to approximate the PHD of X. In addition to the position state and weight, we augment the state vector of a particle in the SMC-PHD filter with an instance ID. Each particle can be regarded as a hypothesis of the point in the world model. A particle with index i at time step k is represented by

$$P_k^{(i)} = \left\{ \tilde{\boldsymbol{x}}_k^{(i)}, w_k^{(i)} \right\} = \left\{ \left[x_k^{(i)}, y_k^{(i)}, z_k^{(i)}, id^{(i)} \right]^T, w_k^{(i)} \right\}. \tag{13}$$

In Fig. 1(b) and (c), the particles are shown with hollow points. Different colors indicate different instance IDs.

A. Prediction

The prediction step predicts the PHD distribution of X based on the estimated instances' transformation matrices given by the preprocessing module. By using the transformation matrices, the 6-D motion of the instance between two time steps can be tackled. Let $\boldsymbol{T}_{J,k}$ denote the transformation matrix estimated for the Jth observed instance at time k. The transformation matrix contains the rotation matrix and translation vector and is a 4×4 matrix. Let $f(\boldsymbol{T}_{J,k}, \boldsymbol{x})$ denote the function that uses $\boldsymbol{T}_{J,k}$ to transform a point $\boldsymbol{x}$ that belongs to object J from time $k-1$ to k. Then, the predicted prior state of this point is

$$\begin{aligned} \boldsymbol{x}_{k|k-1} &= f(\boldsymbol{T}_{J,k}, \boldsymbol{x}_{k-1}) + [\boldsymbol{\xi}, 0]^T \\ &= \boldsymbol{T}_{J,k} \begin{bmatrix} \boldsymbol{x}_{k-1}(1:3) \\ 1 \end{bmatrix} + \begin{bmatrix} \boldsymbol{0}_{3\times 1} \\ \boldsymbol{x}_{k-1}(4) \end{bmatrix} + \begin{bmatrix} \boldsymbol{\xi} \\ -1 \end{bmatrix} \end{aligned} \tag{14}$$

[2]Using Lidar point cloud can also acquire the input required in our world model. As panoptic segmentation and tracking are more accessible with RGB images, we take RGB-D images as input in this article.

where $\boldsymbol{\xi}$ represents the position noise caused by the inaccuracy in the transformation matrix estimation. We assume the noise follows a Gaussian distribution with a zero mean and a covariance matrix $\boldsymbol{Q}_{3\times3}$, denoted as $\boldsymbol{\xi} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q})$.

Then, the state transition function from a particle at time $k-1$ to a prior point state at time k can be formulated as a Gaussian probability density function

$$\pi_{k|k-1}\left(\boldsymbol{x}_k|\tilde{\boldsymbol{x}}_{k-1}^{(i)}\right) = \mathcal{N}\left(\boldsymbol{x}_k; f(\boldsymbol{T}_{J,k}, \tilde{\boldsymbol{x}}_{k-1}^{(i)}), \boldsymbol{Q}\right). \quad (15)$$

By substituting the state transition function into (5), the predicted PHD of X can be obtained.

Due to the limited FOV of the sensor and inevitable occlusion between objects, some objects cannot be observed, and their transformation matrices are not available. In this case, we employ a constant velocity model to predict the transformation matrix. The ego motion of the sensor is handled with the data structure described in Section VI-A.

B. Update

The update procedure updates the PHD distribution of X_k by calculating the particle weights using the latest measurements Z_k at time k. The update is performed only for the particles in the visible space. In this section, our primary focus is on updating the particle weight utilizing measurements Z_k while mitigating the effects of noise discussed in Section IV-A.

The miss-detection and clutter noise in raw sensor measurements has been taken into account by the detection probability P_d and the clutter intensity $\kappa_k(\boldsymbol{z}_k)$ in the original SMC-PHD filter introduced in Section III-B. If the panoptic segmentation and tracking are very reliable, and consequently, there is no instance noise, the weight of the particle with the augmented state vector can be updated using a straightforward method, i.e., individual filtering (IF) method: updating the particles exclusively with measurements sharing the same instance ID. In other words, particles with $id^{(i)} = J$ are updated only with the measurements in Z_J. Essentially, the filter comprises multiple independent SMC-PHD filters, where each filter works for an RFS of a specific instance in (10).

However, if there is instance noise, the IF method suffers from a missing object problem. For example, when some measurement points Z_J of instance J are mislabeled with another existing or new instance ID J' due to the misclassification, inaccurate segmentation or wrong data association, the weights of the particles whose $id^{(i)} = J$ will be decreased. Simultaneously, the particles whose $id^{(i)} = J'$ will be created in the particle birth step (see Section V-C) but will only have a low weight. Consequently, there is a sudden drop in the PHD at the positions of these points. The region is susceptible to being inaccurately classified as free space, posing a high risk of collision for the robot. We illustrate this issue in Fig. 3[Row (a)] with a single measurement point and single particle situation. At $k-2$, the measurement point on an instance is mislabeled, and the estimated occupancy result at this step is wrong because both particles have very low weights. The occupancy status of the space where the point is located will be falsely estimated as free space.

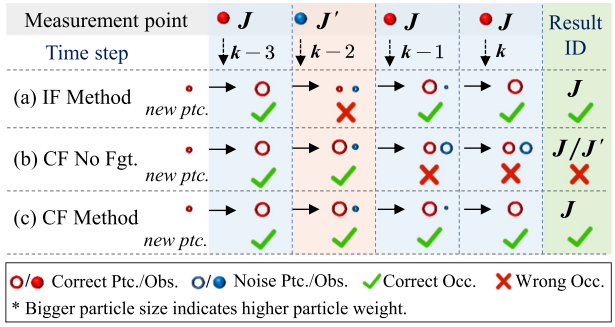


Fig. 3. Illustration of the updated particles of filters (a)–(c) when a noise observation with a wrong ID is given. With method IF, the particles' weights at $k-2$ are too small, and the occupancy status of the space is thus falsely treated as free. With method CF no forgetting function, the instance ID after $k-1$ is ambiguous.

To address the instance noise, we further propose collective filtering (CF) method: updating the particles collectively with all the measurements Z_k by using a specialized likelihood function. The likelihood function is formulated as

$$g_k(\boldsymbol{z}_k|\tilde{\boldsymbol{x}}_k^{(i)}) = F_{gt}(\tilde{\boldsymbol{x}}_k^{(i)}) \cdot T_r(\boldsymbol{z}_k, \tilde{\boldsymbol{x}}_k^{(i)}) \cdot \mathcal{N}\left(\boldsymbol{z}_k; \tilde{\boldsymbol{x}}_k^{(i)}, \boldsymbol{\Sigma}\right) \quad (16)$$

where $T_r(\cdot)$ represents an instance ID transition function and $F_{gt}(\cdot)$ is a forgetting function. $\mathcal{N}(\cdot)$ is the Gaussian probability density for position transition, used to model the position noise, whose covariance matrix is assumed to be $\boldsymbol{\Sigma}$.

$T_r(\boldsymbol{z}_k, \tilde{\boldsymbol{x}}_k^{(i)})$ is defined as

$$T_r\left(\boldsymbol{z}_k, \tilde{\boldsymbol{x}}_k^{(i)}\right) = \begin{cases} 1, & \text{if } \tilde{\boldsymbol{x}}_k^{(i)}(4) = \boldsymbol{z}_k(4) \\ P_{\text{tr}}\left(\boldsymbol{z}_k(4), \tilde{\boldsymbol{x}}_k^{(i)}(4)\right), & \text{Otherwise} \end{cases} \quad (17)$$

where $P_{\text{tr}}(\boldsymbol{z}_k(4), \tilde{\boldsymbol{x}}_k^{(i)}(4)) \in [0, 1)$ characterizes the likelihood of an instance being identified as or associated with another instance. This parameter can be determined as a function of instance labels and positions if the performance of instance segmentation and tracking is known. For generality and computational simplicity, we treat it as a constant in the experiments.

The forgetting function $F_{gt}(\tilde{\boldsymbol{x}}_k^{(i)})$ is elaborated as a truncated Ebbinghaus curve of forgetting [54]

$$F_{gt}(\tilde{\boldsymbol{x}}_k^{(i)}) = \begin{cases} e^{-\frac{\Delta k^{(i)}}{S}}, & \text{if } \Delta k^{(i)} \le \Delta\bar{k} \\ 0, & \text{if } \Delta k^{(i)} > \Delta\bar{k} \end{cases} \quad (18)$$

where e represents the Euler's number. $\Delta k^{(i)} \in \mathbb{N}$ denotes the time interval between the current time step k and the last time step when the ith particle was updated with a measurement sharing the same ID. $\Delta\bar{k}$ is a threshold that controls the maximum time interval that the particle can be updated with the measurement that has a different ID. The constant $S > 0$ governs the forgetting speed, with a smaller S resulting in a faster rate of forgetting.

The ID transition function allows particles with a different ID from the measurement to still be updated if the measurement's position is close. Then, the aforementioned missing object problem can be avoided. However, if an object with ID J is

persistently labeled with ID J' afterward in the tracker or if it is relabeled as J after mislabeled as J', both particles with ID J and J' will have large weights, which causes confusion in labeling the space occupied by the object. We illustrate the situation where the object is mislabeled with ID J' and then relabeled as J in Fig. 3[Row (b)]. The result contains ambiguous ID choices. Therefore, the forgetting function becomes crucial. This function reduces the weight of particles not updated with measurements sharing the same ID. When $\Delta k^{(i)} > \Delta\bar{k}$, the particle's weight experiences a rapid and substantial decrease, finally being removed in the resampling step. As a result, the map turns to trust more on consistent measurements. If J is permanently mislabeled as J', the particles with ID J will be removed after a few observations, and the space taken by the object will only be labeled with J'. If P_{tr} is set to be zero and the forgetting function is not used, the CF method will be equivalent to the IF method.

Updating each particle with each measurement is computationally expensive, as discussed in [15]. To accelerate the update process while considering the occluded space, we incorporate the Pinhole model-based pyramid subspaces and activation bounding boxes to confine the particles that should be updated with each measurement point. The details can be found in Section VI and the Appendix.

C. Particle Birth, Resampling, and Occupancy Estimation

Particles are typically born from the measurements Z_k. For each measurement point $\boldsymbol{z}_k \in \mathrm{Z}_k$, we generate L_b newborn particles. The state of each newborn particle $P_{b,k} = \{\tilde{\boldsymbol{x}}_{b,k}, w_{b,k}\}$, where the subscript b suggests "born," is given by the following equations:

$$\tilde{\boldsymbol{x}}_{b,k}(1:3) = \boldsymbol{z}_k(1:3) + \boldsymbol{\sigma}, \boldsymbol{\sigma} \in \mathcal{N}(0, \boldsymbol{\Sigma}) \tag{19}$$

$$\tilde{\boldsymbol{x}}_{b,k}(4) = \boldsymbol{z}_k(4), \quad w_{b,k} = \frac{v_{b,k|k-1}}{M_k L_b} \tag{20}$$

where $v_{b,k|k-1} = \int \gamma_{k|k-1}(\boldsymbol{x}_k)\mathrm{d}\boldsymbol{x}_k$ is a parameter that controls the expected number of newborn points from $k-1$ to k. The instance ID of a newborn particle is the same as the measurement point. Suppose the measurement point of an object is labeled differently at a new time, in which case instance noise occurs, the instance ID of the newborn particle will also change, and new instance hypotheses will be generated. These hypotheses are updated with the measurements in the subsequent time steps to filter out the incorrect ones with the ID transition function and the forgetting function. With the newborn particles, (6)–(8) need to be changed to separately update survived and newborn particles. Details of the changed equations can be found in [55].

The resampling step is to mitigate the particle degeneracy problem and control the number of particles. We still use the rejection sampling [56] approach for each voxel subspace in the map [15]. Particles in one voxel subspace are resampled by their weights, regardless of the instance ID. Particles with higher weights are more likely to survive or be duplicated in the resampling process, while the particles with lower weights are more likely to be removed, which is the "death" of the particles. The overall weight of the particles in one voxel subspace is the same before and after resampling. Therefore, the resampling process does not affect the occupancy status estimation. Particles newly born in the same step are excluded from the resampling process. To reduce computational cost, resampling is triggered to halve the number of particles only when a voxel becomes full, thereby freeing up space for the insertion of new particles.

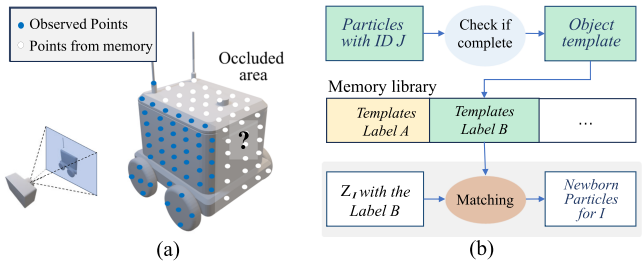


Fig. 4. Illustration of the memory enhancement. In (a), the blue points are currently observed while the white points are occluded and are aimed to be conjectured. In (b), the memory enhancement structure is shown. The first row shows the process of adding a template in the memory library, which is shown in the middle row. The last row shows the process of matching the particles with a template in the memory library. The green background indicates that the particles or templates correspond to the same semantic label. (a) Memory illustration. (b) Memory operations structure.

Considering (3), (8)–(10), the cardinality expectation of points that has $id = I$ in a voxel subspace $\mathbb{V}$ (constrained by position dimensions) is calculated by

$$\mathbf{E}\left[|\mathrm{X}_I^{\mathbb{V}}|\right] = \iiint_{\mathbb{V}} D_{\mathrm{X}_I}\mathrm{d}x\mathrm{d}y\mathrm{d}z = \sum_{i=1}^{L_I^{\mathbb{V}}} w_k^{(i)} \tag{21}$$

where the subscript or superscript $\mathbb{V}$ suggest the point or particle is in the voxel subspace. $L_I^{\mathbb{V}}$ is the number of particles with $id = I$ in the voxel subspace. Time step k is omitted in the equation for simplicity. The cardinality expectation of all points in the voxel subspace is

$$\mathbf{E}\left[|\mathrm{X}^{\mathbb{V}}|\right] = \sum_I \mathbf{E}\left[|\mathrm{X}_I^{\mathbb{V}}|\right]. \tag{22}$$

We adhere to an "occupancy first and ID second" strategy, prioritizing the occupancy status over the object ID in navigation tasks. Therefore, the occupancy status of the voxel subspace is first estimated by applying a threshold on $\mathbf{E}[|\mathrm{X}^{\mathbb{V}}|]$. If the voxel subspace is determined occupied, then its ID is estimated by finding the ID with the largest $\mathbf{E}[|\mathrm{X}_I^{\mathbb{V}}|]$.

D. Memory Enhancement

Since objects with the same semantic label in one environment usually geometrically resemble each other, memory of previously observed objects can be used to conjecture the occluded portion of an object. For example, in Fig. 4(a), the surface facing the camera is observed while the remaining area on the object is occluded. We store particles from previously observed objects as templates, which serve as a memory to account for the occluded parts of newly observed instances that have the same semantic label. This conjecture is important for navigation tasks because it can help the robot avoid planning in the space that is likely

to be occupied. In addition, it accelerates the map's response to previously observed objects by bearing particles to the occluded portion in advance.

1) Structure: Fig. 4(b) illustrates the structure of the memory enhancement module. The memory library is illustrated in the middle row of the figure. During the mapping process, we have particles with different instance IDs. Suppose an instance J is well observed from various directions and is completely modeled. In that case, the particles with ID J are stored as a template with the semantic label of J in the memory library. Each label in the library can encompass several templates with distinct shapes. In practical navigation scenarios, a robot rarely observes an object from all directions. We evaluate the completeness (Comp.) of the instance by uniformly sampling rays from the mass center of the voxels of this instance and calculating the percentage of rays that intersect with the voxels. If this percentage exceeds a predefined threshold, the instance is considered completely modeled, and its particles are stored as a template.

The memory is integrated into the particle birth step. When a new instance with measurement points Z_I is observed and the number of measurement points exceeds a threshold (e.g., five thousand), we match these points with templates of the same semantic label and generate additional newborn particles based on the best-matched template. Let $\bar{\mathrm{T}}_J = \{P_J^{(1)}, P_J^{(2)}, \ldots, P_J^{(L_T)}\}$ denote the template generated from instance J, containing L_T particles. We then add L_T newborn particles in addition to the newborn particles in Section V-C. Each newborn particle $P_b^{(i)}$ has the same position as the particle $P_J^{(i)}$ but weight $w_{b,k}$, which turns to

$$w_{b,k} = \frac{v_{b,k|k-1}}{M_k L_b + L_T}. \tag{23}$$

In the update step, the weight of these particles is updated with lateral observations, the same as the other particles, so that the conjecture can be corrected. If a voxel subspace is not determined occupied but contains nonupdated conjectured particles, it is labeled "speculatively occupied."

2) Matching: The matching algorithm matches the particles in the template with the measurement points of a new instance. The matching algorithm should be efficient since there can be multiple new instances at a time. Unlike the matching between two regular point clouds, the matching between the particles and the measurement points should consider the nature of the particles representing the PHD and having different weights. In addition, the measurement points also contain hidden information, which is, the space between the camera and the measurement points should be free space. Therefore, we introduce a PHD-based matching algorithm to match the particles with the measurement points.

The algorithm relies on a similarity score defined with the property of PHD described in (3). Suppose the measurement points in Z_I are in a bounding box space $\mathbb{S}_I$, where I is the instance ID, and the boundary of $\mathbb{S}_I$ can be found by searching the minimum and maximum coordinates of the measurement points. We divide $\mathbb{S}_I$ into voxel subspaces $\{\mathbb{S}_I^{(i)}, i \in N_I\}$, where N_I is the number of subspaces. Suppose $h(\mathbb{S}_I^{(i)})$ is the expected point number in $\mathbb{S}_I^{(i)}$. If $\mathbb{S}_I^{(i)}$ contains at least one measurement point, $h(\mathbb{S}_I^{(i)})$ is considered as 1. If $\mathbb{S}_I^{(i)}$ is observed to be free space (determined by raycasting), $h(\mathbb{S}_I^{(i)})$ is considered as -1. Otherwise, $h(\mathbb{S}_I^{(i)}) = 0$. Then, we iterate over the voxel subspaces and calculate the similarity score with the J_{th} template $\bar{\mathrm{T}}_J$ using the following equation:

$$\mathrm{Score}(\mathrm{Z}_I, \bar{\mathrm{T}}_J) = \frac{1}{N_I} \sum_{i=1}^{N_I} \min \left\{ \int_{\mathbb{S}_I^{(i)}} D_{\bar{\mathrm{T}}_J}(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}, 1 \right\} \cdot h(\mathbb{S}_I^{(i)}) \tag{24}$$

where $D_{\bar{\mathrm{T}}_J}(\boldsymbol{x})$ is the PHD of $\bar{\mathrm{T}}_J$ at position $\boldsymbol{x}$. $\int_{\mathbb{S}_I^{(i)}} D_{\bar{\mathrm{T}}_J}(\boldsymbol{x}) \mathrm{d}\boldsymbol{x}$ is the PHD integral of $\bar{\mathrm{T}}_J$ in $\mathbb{S}_I^{(i)}$ and equals the weight summation of the template particles in $\mathbb{S}_I^{(i)}$ according to (3) and (8). The integral and $h(\mathbb{S}_I^{(i)})$ are both normalized to one to avoid the influence caused by the voxel size, which is determined by a balance between the Acc. and the efficiency of the matching algorithm. With the similarity score, the matching is performed using the RANSAC algorithm [57] to search for transformation matrices that maximize the score.

VI. Implementation Details

This section describes two implementation details of the proposed system: the data structure and the measurement points generation method. The former is to realize an efficient S^{2}MC-PHD filter, and the latter describes our choices of existing methods to implement the preprocessing modules.

A. Data Structure

As described in Section IV-A, we use voxel subspaces to store the particles and use pyramid subspaces to distinguish the occluded area in the continuous space and accelerate the update process with an activation space as in [15]. To represent the voxel subspaces and the pyramid subspaces in practice, our previous work [15] uses a regular array and a dynamic vector, respectively, resulting in a high computational cost. To address this issue and consider the instance ID, we propose a new data structure composed of the following three parts:

1) a 3-D circular buffer;
2) an instance hash map;
3) an update indices image.

The data structure is illustrated in Fig. 5.

1) 3-D Circular Buffer: We store the particles in a 3-D circular buffer indexed by Morton Code for efficiency. Each element in the buffer represents a voxel subspace and contains the particles within the subspace. With the circular buffer, only the indices of the voxel subspaces need to be updated when the robot moves. Following Ewok [19], at the start of each time step, the indices are updated using the localization data, accounting for the relative motion of background particles to the sensor. For instances of interest, their particles are relocated to new voxels by recalculating their voxel indices based on their predicted positions, as described in (14). A position vector is maintained to

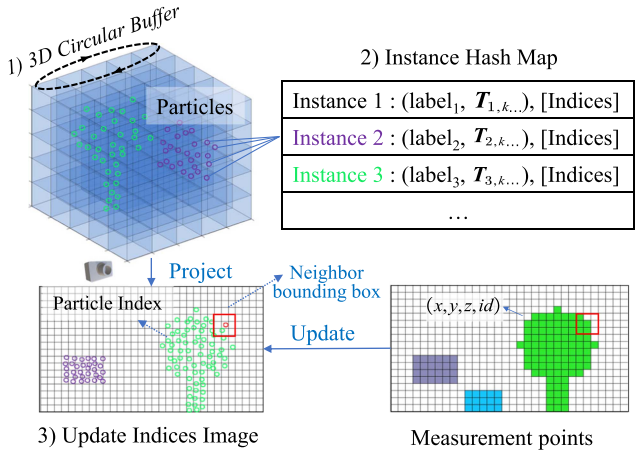


Fig. 5. Data structure. Different colors of particles and measurement points represent different instance IDs. The blue object is newly observed. The red rectangle represents the neighbor bounding box area (activation space) of the pixel with a particle in red.

compensate for position errors caused by the limited resolution of the voxel.

Each voxel has a fixed particle capacity. If particles move to a voxel that has already reached its maximum capacity, resampling, as described in Section V-C, is applied to this voxel to reduce the number of particles and free up space. If after resampling the space remains insufficient, the excess particles are discarded. Each particle contains the states described in (13), namely position, weight, and instance ID, along with additional attributes such as a timestamp and a validity flag. The timestamp records the moment when the particle is updated with a measurement sharing the same ID, and it is used to calculate the forgetting function in (18). The validity flag indicates whether the particle is valid, facilitating efficient particle deletion by setting the flag to false.

2) Instance Hash Map: A hash map is employed to store the instance ID along with the corresponding instance-level states. These states encompass each instance's semantic label, previous transformation matrices, and particle indices in the circular buffer. The particles' indices are used to quickly locate the particles in the buffer and predict the particles' positions in the prediction step (see Section V-A). The previous transformation matrices are used to predict a new transformation matrix in the prediction step when the instance is occluded.

3) Update Indices Image: Since only a part of the particles in the circular buffer are visible to the camera and should be updated, we present an update indices image to leverage the Pinhole model to find these particles and store the indices. Specifically, a breadth-first search is used to identify the voxel subspaces in the FOV. In each voxel subspace in the FOV, the particles are projected to the image with the camera intrinsics and extrinsics. If the particle's depth is smaller than the depth of the corresponding measurement (indicating that the particle is not occluded), the particle's index is stored in a pixel. In this context, each pixel serves a role similar to the pyramid subspace in [15], as is illustrated by Fig. 1(c). Subsequently, the weights of particles in each pixel are updated using only measurement points in a neighboring bounding box area, which act as the activation space in [15], to accelerate the update process. In other words, the particles outside the bounding box area of a measurement point do not need to be updated with this measurement point. In Fig. 5, the red rectangle represents the neighbor bounding box area of the pixel with a particle in red. The size of the bounding box area is determined by the noise model and the distance from the camera to the measurement points. We present the derivation of the bounding box area in Appendix A.

B. Preprocessing

Generating the measurement points requires preprocessing modules in Fig. 2. As discussed in Section IV-B, these preprocessing modules can be implemented using existing methods. The panoptic segmentation is realized by Mask2Former [58] using the OpenMMLab [59] framework. For object tracking and transformation estimation, we employ Superpoint [60] and Superglue [61] to extract feature points for the instances and match them between two frames. Tracking is then implemented by voting the matched points, while transformation estimation is accomplished by applying RANSAC [57] on the matched feature points (considering depth) of each object. When an object is partially occluded, the transformation estimation can still be conducted with the matched feature points in the visible area.

VII. Experiment

This section presents the experimental results. We first compare the proposed map with state-of-the-art mapping systems in terms of occupancy, semantic, and instance estimation. The occupancy information is the foundation for the robot to realize safe navigation, while the semantics and instances of the occupied space are crucial for the robot to understand the environment. Then, the ablation study is conducted to compare the results of the proposed map with different update methods, and with and without memory enhancement. Moreover, the efficiency of the proposed system is evaluated with computational time. Finally, real-world data is used to demonstrate the effectiveness of the proposed system in realistic scenarios.

A. Occupancy Estimation

We adopt Virtual KITTI 2 [62], [63] for evaluation. The cars, vans, etc., with possible dynamic motions in the dataset, are objects of interest. The dataset was chosen because it provides the ground truth about depth, panoptic segmentation, instance ID, and object poses over time, which is essential to creating the ground truth of occupancy and instance-aware semantics estimation in the dynamic environment. Creating the ground truth local instance-aware semantic occupancy map involves a two-step process. The first step is accumulating the point cloud in the global coordinate using each frame's depth image, panoptic segmentation, and ego-pose, attributing semantic and

TABLE I
GENERAL COMPARISON OF THE MAPPING SYSTEMS

Map	Base	Semantics	Instance	Dynamic
ewok [19]	Raycasting	No	No	No
k3dom [31]	Particles	No	No	**Yes**
dsp map [15]	Particles	No	No	**Yes**
voxblox++ [5]	TSDF	**Yes**	**Yes**	No
kimera-s. [4]	TSDF	**Yes**	No	No
Ours	Particles	**Yes**	**Yes**	**Yes**

TABLE II
OCCUPANCY ESTIMATION COMPARISON

Depth Image	GT depth			Depth with noise		
Map	**AHD↓**	**F1↑**	**ADm↓**	**AHD↓**	**F1↑**	**ADm↓**
ewok [19]	0.460	0.871	0.534	0.504	0.738	0.405
k3dom [31]	1.006	0.643	0.425	3.858	0.467	0.508
dsp map [15]	0.316	0.908	0.596	0.549	**0.747**	0.747
voxblox++ [5]	0.998	0.641	0.468	1.624	0.444	0.596
kimera-s. [4]	0.256	0.878	0.423	0.461	0.634	0.364
ours	**0.067**	**0.945**	**0.276**	**0.195**	0.689	**0.242**
voxblox++* [5]	0.999	0.647	0.475	1.535	0.412	0.639
kimera-s.* [4]	0.255	0.878	0.424	0.462	0.634	0.364
ours*	**0.066**	**0.945**	**0.318**	**0.201**	**0.682**	**0.278**

$*$ indicates the case using segmentation and tracking from Section VI-B.

instance IDs to individual points. As new frames emerge, points associated with the moving objects are transformed to their new positions using the ground truth object poses. Subsequently, at each time step k, the accumulated point cloud from 0 to k is divided into voxel subspaces and the occupancy status of each voxel in the local map range is determined by whether a point is present. The semantic label of the voxel subspace is determined by the majority of the points in the subspace.

The occupancy estimation is evaluated using the average Hausdorff distance (AHD), the F1 score (F1), and the average distance of movable objects (ADm). The AHD measures the average distance between the center points of the estimated occupancy voxels and those of the ground truth. A smaller AHD indicates better surface reconstruction performance. The F1 is the harmonic mean of precision and recall. A higher F1 shows better occupancy classification performance. The ADm is utilized to specifically measure the average Euclidean distance of the movable objects to the closest occupied voxels in the map. The movable objects in our scene are the aforementioned the objects of interests with possible dynamic motions. A smaller ADm indicates a better mapping performance of these objects. Note the abovementioned metrics are evaluated for the local map with the ego-vehicle moving. Therefore, the metrics are calculated for each frame and then averaged over all frames. All the five sequences from Virtual KITTI 2 are used for evaluation and the results are averaged. These sequences contain $\{93, 17, 15, 21, 127\}$ vehicles, with the percentage of moving vehicles being $\{9.7, 52.9, 93.3, 100.0, 60.6\}\%$, respectively.

The comparison is conducted with five state-of-the-art (SOTA) maps: ewok [19], k3dom [31], dsp map [15], kimera-semantic [4], and voxblox++ [5]. Table I shows a general comparison of the maps. Bold "Yes" indicate that the corresponding function is available in the specified map. In these maps, k3dom and dsp map are particle-based methods designed for dynamic environments, while Ewok uses raycasting for mapping without special consideration for dynamic objects, and none of the three methods considers semantics. Kimera-semantic and voxblox++ are both TSDF-based semantic maps. Voxblox++ is instance-aware. In comparison, our map is an instance-aware semantic map and considers dynamic objects. Each map is evaluated in the situation with ground truth depth and noised depth based on a real-world noise model introduced in [64]. Furthermore, two cases are evaluated for the map with semantics: one with the ground truth semantics and tracking, and the other using the method in Section VI-B. The lateral contains segmentation and tracking noise and is represented with a superscript $*$ in the result tables.

The voxel resolution in the test is 0.2 m, and the map size is (51.2, 51.2, 51.2) m (with 2^8 voxels on each dimension). All the maps are compared in the voxelized form. For the TSDF-based maps [4], [5], a distance threshold is applied to determine the occupied voxels. We tested different distance thresholds with a sampling step of 0.05 m for the TSDF-based maps, and different occupancy thresholds with a sampling step of 0.1 for the rest maps to find the best performance of each map. The remaining mapping parameters for our method are detailed in Appendix C, while the parameters for other maps are kept at their default values as specified in their respective released code. The input images are from front camera with ID 0 in the dataset. The image size is 1242×375 pixels. K3dom [31] uses CUDA parallel computing and is tested with an NVIDIA RTX 2060S GPU. The rest of the maps are tested with an AMD Ryzen 9 5900X CPU with single-core computing. Since kimera-semantic and voxblox++ are global maps, we crop out a local map to compare with the ground truth. The results are shown in Table II. The arrows after each metric indicate the direction of the improvement. The best performance is in bold. Fig. 6 presents an example mapping result of three time steps.

When using ground truth inputs, our map achieves the best performance in terms of all three metrics. The AHD and ADm are 73.8% and 34.8% smaller than the second-best map kimera-semantics, respectively, which indicates that our map has significantly improved the overall surface reconstruction and the movable object mapping performance in dynamic environments. The F1 does not show a distinct difference between our map and the dsp map, but ours is still 4.1% higher. From Fig. 6, it can be seen that our map does not suffer from the missing object problem and the trace noise problem as the other maps do.

When the depth with noise [64] is used, our map is the third best in terms of F1 but is at least 57.7% and 33.5% better than other maps in terms of AHD and ADm, respectively. For the case that uses nonground-truth segmentation and tracking (with superscript $*$), our map also shows a significant advantage over kimera-semantic and voxblox++. Fig. 7 further illustrates the mapping result of the three maps using nonground-truth segmentation and tracking.

Overall, our map shows the best performance in terms of the occupancy estimation and is more robust to the noise in depth image, and segmentation and tracking.

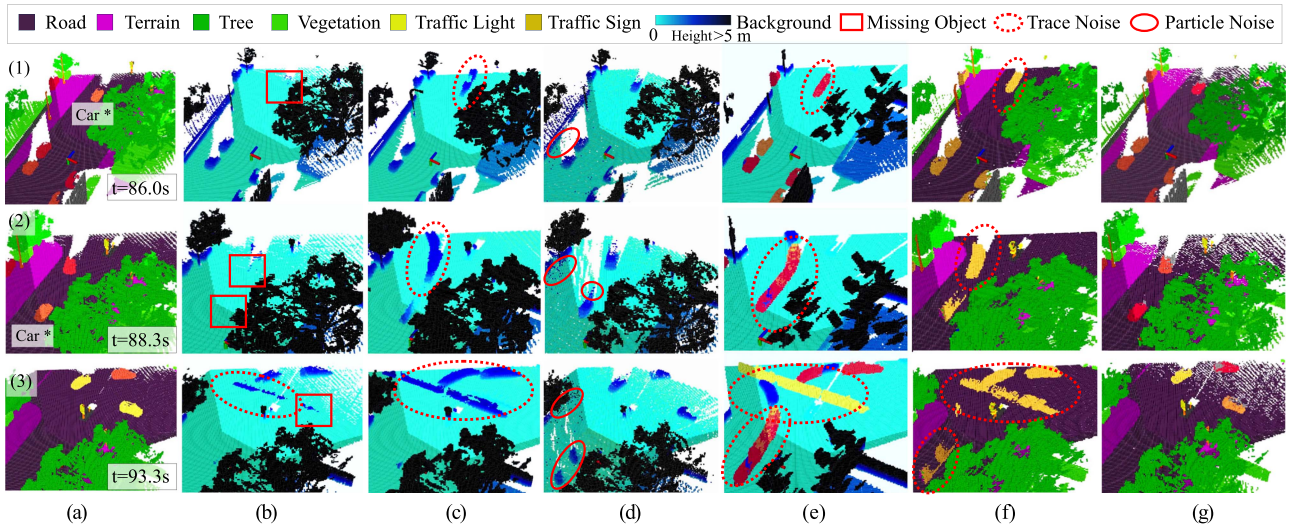


Fig. 6. Mapping result comparison when using ground truth segmentation and tracking. The first column shows the ground truth map at three time steps, $t = \{86.0, 88.3, 93.3\}$s. The rest columns show the result of different maps, among which ours is presented in the last column. Voxels in the FOV are brighter than the rest to illustrate the currently observed area. In (e) voxblox++ [5] and (g) ours, the vehicles are painted in random colors to show their instance-awareness. The meaning of the rest of the colors is illustrated in the legend above. If no semantic meaning is provided, the voxel is painted light blue to dark blue according to the height. Axes in the subfigures indicate the pose of the ego-vehicle. The red rectangles indicate the missing objects, while the red dashed ellipses illustrate the trace noise in the compared maps. The red solid ellipses in (d) suggest the noise in the occluded space caused by the individual particle motion model used in dsp map [15]. In row (1) and (2) in column (a), a car in orange is marked with "car*." The motion of this car causes the missing object or trace noise problem in the compared maps. (a) Ground truth. (b) Ewok [19]. (c) K3Dom [29]. (d) Dsp map [15]. (e) Voxblox++ [5]. (f) Kimera-s. [4]. (g) Ours.

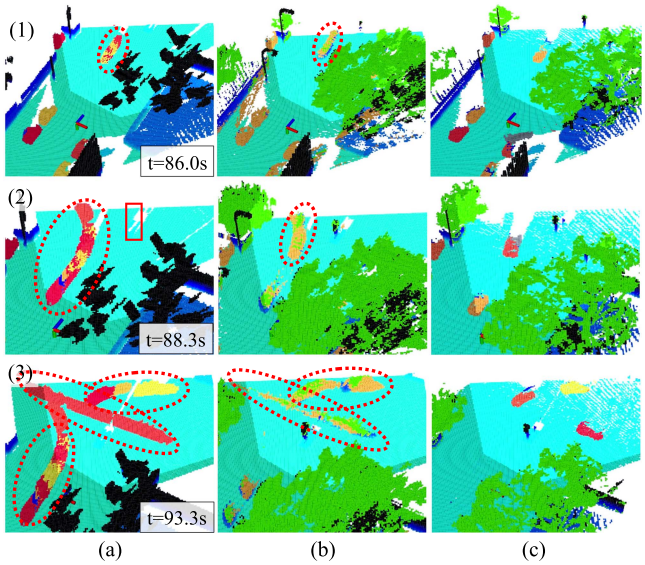


Fig. 7. Mapping result comparison when segmentation and tracking are realized with the method described in Section VI-B. Columns (a) to (c) show the mapping result of voxblox++ [5], kimera-semantic [4] and our map, respectively. The three time steps, the ground truth mapping result, and the legend are the same as in Fig. 6. Voxblox++ [5] and kimera-semantic [4] suffer from the trace noise problem. Voxblox++ [5] also misses details like the traffic sign and branches of the trees. (a) Voxblox++ [5]. (b) kimera-s [4]. (c) Ours.

TABLE III
SEMANTIC SEGMENTATION RESULTS OF STATIC OBJECTS

Depth Image	GT depth		Depth with noise	
Map	**mIoU↑**	**mIoU 3-D↑**	**mIoU↑**	**mIoU 3-D↑**
kimera-s. [4]	0.586	0.494	0.339	0.233
ours	**0.629**	**0.834**	**0.468**	**0.322**
kimera-s.* [4]	0.613	0.245	0.165	0.110
ours*	**0.636**	**0.473**	**0.355**	**0.245**

TABLE IV
SEMANTIC SEGMENTATION RESULTS OF MOVABLE OBJECTS

Depth Image	GT depth		Depth with noise	
Map	**mIoU↑**	**mIoU 3-D↑**	**mIoU↑**	**mIoU 3-D↑**
voxblox++ [5]	0.360	0.112	0.394	0.087
kimera-s. [4]	0.464	0.212	0.365	0.101
ours	**0.680**	**0.596**	**0.660**	**0.256**
voxblox++* [5]	0.364	0.112	0.382	0.085
kimera-s.* [4]	0.462	0.161	0.353	0.085
ours*	**0.685**	**0.367**	**0.591**	**0.198**

B. Semantic and Instance Estimation

Since our map is instance-aware, we evaluate both semantic segmentation and instance segmentation performance. The semantic segmentation performance is evaluated with 2-D and 3-D mean intersection over union (mIoU) of 15 classes in the Virtual KITTI 2 [62], [63] dataset when ground truth segmentation is used. For the case that uses the OpenMMLab framework for segmentation, only the trees and cars are evaluated because the other classes are annotated differently or unavailable with the tested pre-trained segmentation model [65]. In the 2-D case, the labeled voxels in the map are projected to the image and compared with the ground truth segmentation image. In the 3-D case, the labeled voxels are compared with the ground truth labeled voxels generated with the steps described in Section VII-A. We present the results of static objects and movable objects separately in Tables III and IV to show the

TABLE V
INSTANCE SEGMENTATION RESULTS

Depth image	GT depth		Depth with noise	
Map	**mF1↑**	**mF1 3-D↑**	**mF1↑**	**mF1 3-D↑**
voxblox++ [5]	0.381	0.154	0.174	0.000
ours	**0.667**	**0.588**	**0.409**	**0.186**
voxblox++* [5]	0.372	0.208	0.169	0.055
ours*	**0.606**	**0.402**	**0.345**	**0.177**

performance of the map towards different types of objects. Voxblox++ [5] receives only the instance segmentations, which are available just for movable objects in the dataset, and thus, is not included in the Table III. In both tables, our map has the best performance regarding each metric. The advantage is distinctive regarding movable objects and the 3-D mIoU metric. The 2-D and 3-D mIoU for movable objects are at least 45% better than the second-best map, regardless of whether the depth, segmentation, and tracking have noise, in the tested cases.

The instance segmentation performance is evaluated with the mean F1 of the instances in different frames. An instance in the ground truth and the estimated map is considered matched if the IoU is larger than 0.5. If no matching is found, the F1 of the instance is 0. The results are shown in Table V. Compared with voxblox++, our map shows over 60% improvement in terms of 2-D and 3-D F1 in all cases.

Overall, our map has a significant advantage in terms of both semantic segmentation and instance segmentation. The noise in depth image and segmentation and tracking affects more on the 3-D metrics than the 2-D metrics for all maps, which is reasonable because the noise in 2-D is further amplified in 3-D depending on the distance from the camera.

C. Ablation Study

The presented results use the memory enhancement and the CF method in Section V-B. In this section, we further conduct an ablation study to compare the difference of using IF and CF methods for the update step and show the effect of using the memory enhancement.

When the ground truth segmentation and tracking are used, the IF and CF methods show very similar performances. When the segmentation and tracking method in Section VI-B is used, CF shows 7.1% higher performance on ADm but 1.6% lower performance on mF1 3-D. The difference is insignificant because the instance noise takes a small portion of the instance estimations. However, if an instance in the measurement is allocated with a wrong ID due to missing detection, mislabeling or mismatching, the IF method suffers from the missing object problem that is highly detrimental to safety. An example mapping result is shown in Fig. 8(a). Therefore, CF is preferred in the proposed system.

With memory enhancement, the map's response to the newly observed space on an instance is improved. Fig. 9 shows the 2-D and 3-D mF1 score changing curve when the same instance is and is not matched with a template. At Step 0, a template is matched for the blue curve. Then, from Step 0 to 1, the F1 increase of the blue curve is 12.9% and 18.8% faster than the red curve in 2-D and 3-D, respectively. The results indicate that memory enhancement is beneficial for the map in responding to the newly observed area on an instance. We illustrate some conjectured occupied voxels in Fig. 8(c) using the gray color. These conjectured voxels have also been proven helpful in motion planning [66], [67].

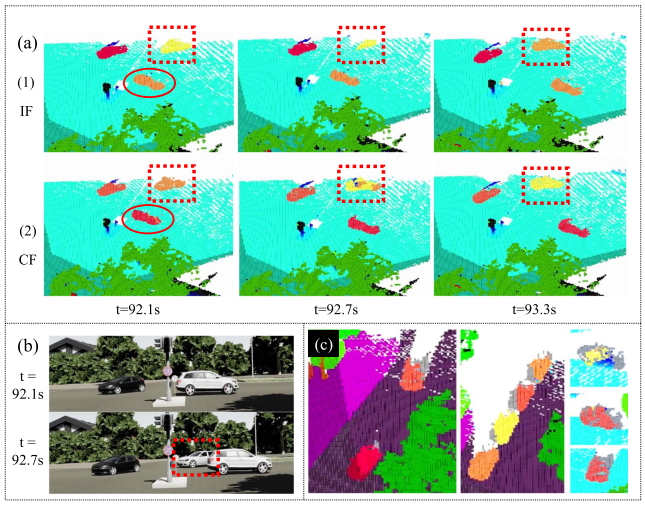


Fig. 8. (a) Illustration of the missing object problem with the IF method. At 92.0 s, the car in the dashed red rectangle is occluded by the car in the red ellipse. At time 92.7 s, the former car is redetected and allocated with a new ID, which causes instance noise and is mostly missing in the map when using IF. In comparison, CF correctly updates the car. (b) Corresponding RGB images. The red rectangle indicates the occluded car. (c) Presents several instances that have been matched with templates. The gray voxels show the voxels that are not observed but are "speculatively occupied."

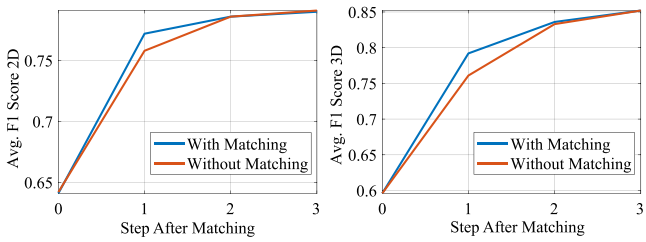


Fig. 9. Response comparison with and without memory enhancement.

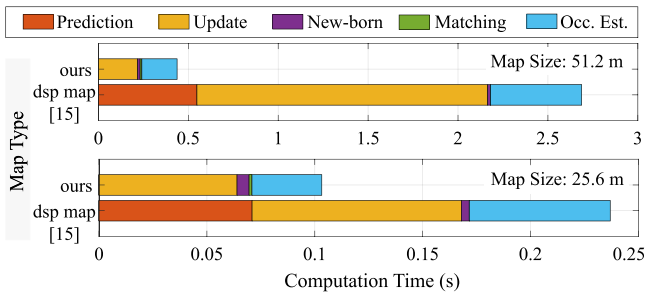


Fig. 10. Average computation time two particle-based maps: Our map and the dsp map [15]. When the map size is 25.6 m with voxel resolution 0.2 m, our mapping approach takes 103.2 ms per frame, reaching near real-time performance. The time consumption of different mapping steps is shown in different colors. The hardware is specified in Section VII-A.

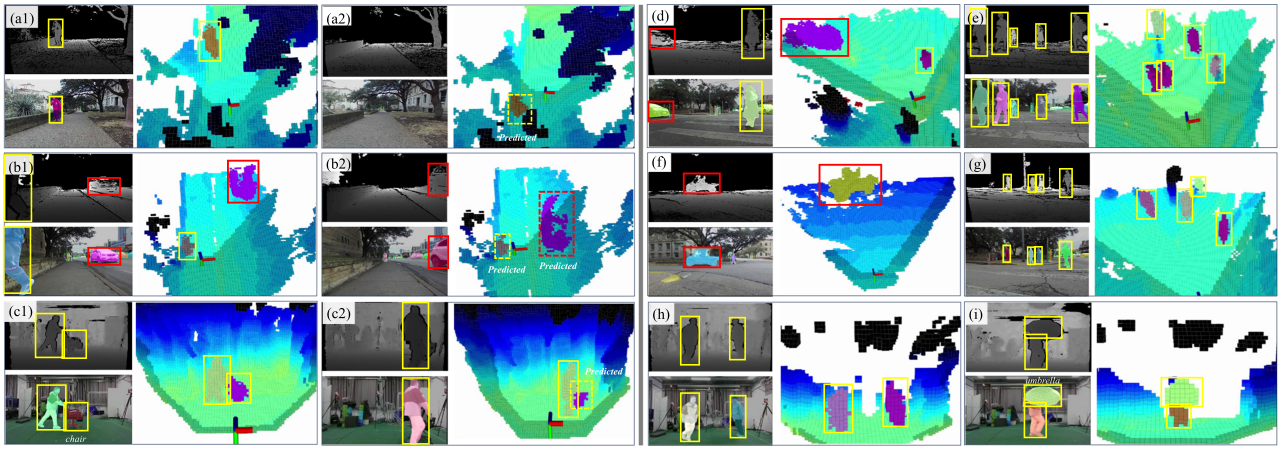


Fig. 11. Mapping result in the real world. On the left side of the figure, three scenes (a)–(c) are presented. Each scene has two frames, for example, (a1) and (a2), captured at two different time steps. In each frame, a depth image and an RGB image with segmentation masks are presented on the left, while the mapping result is shown on the right. In the map view, the objects of interest are depicted in random colors, and the background is painted in green to light blue and then to dark blue according to the height. The camera's pose is illustrated by axes, with areas within the FOV appearing brighter than the surrounding areas. In (a1), a human is running towards the camera and exits the FOV in (a2), where the map gives a predicted occupancy status of the human. Similarly, (b1) and (b2) show a scenario with a car and a human. In (c1), both the human and the chair are moving, while the chair is occluded in (c2), and the predicted occupancy status of the chair is shown. Red and yellow rectangles outline the cars and the other objects of interest, respectively, with dashed rectangles indicating the objects that are occluded or out of the FOV. Additional scenes are shown in (d)–(i) on the right side of the figure. Scenes (a), (b) and (d)–(g) are from a ZED 2 camera used in the UT campus object dataset, while the rest are recorded by ourselves with a RealSense D455 camera.

D. Computation Time

We compared the computational time of the proposed map and the dsp map [15], in Fig. 10. An additional map size of 25.6 m is tested, while the rest settings are the same as used in the previous tests. The hardware has been specified in Section VII-A, and both maps use only a single core of the CPU. The time consumption of different mapping steps is shown in different colors. The preprocessing modules adopt existing works and can be adjusted with different models and, thus, is not included. It can be seen from Fig. 10 that the most time-consuming step is the update step. The template matching step in our map takes 9.8ms with a big map size and 1.3 ms with a small map size. In total, the proposed map takes 437.7 ms on average per frame when the map size is 51.2 m, which is six times faster than the dsp map. If the map size is reduced to 25.6 m, the proposed map takes 103.2 ms while the dsp map takes 236.9 ms. This demonstrates that the proposed map with the improved data structure described in Section VI-A, our map is much more efficient than the dsp map though we have added more complex semantics and instance information.

Ewok [19] takes less than 100 ms per frame to process, even with large map sizes, but it is not suitable for dynamic environments and does not contain semantic information. Kimera-semantic [4] and voxblox+ [5] are static global maps whose computation time increases from around 100 ms to over 1000 ms as the map grows. K3dom [31], a particle-based dynamic local map similar to the DSP map and ours, requires over 26 s per frame when conducting parallel computing on an RTX 2060S GPU when the map size is 25.6 m. Overall, our map emerges as the most efficient map among the compared dynamic maps and particle-based maps.

E. Real-World Experiment

The real-world experiment is conducted with the RGB-D image pairs and pose estimation data from the UT campus object dataset [68], [69] and some additional RGB-D image pairs recorded by ourselves. The image pairs in the UT campus object dataset came from a ZED 2 camera[3] mounted on a mobile robot, while those recorded by ourselves came from a RealSense D455 camera. Fig. 11 shows scenes of the original images and the constructed map, where the brighter part illustrate the area in the FOV. Scenes (a), (b), and (d)–(g) are from the UT campus object dataset and contain cars and people. Scene (c) and (h)–(i) are recorded by ourselves and contain people and objects moved by people, such as a chair and an umbrella. In scenes (a), (b), and (c), the first frame shows a time step when the moving objects can be observed, while the second frame shows a while later when the moving objects cannot be observed, our map gives a predicted occupancy status of the objects. Although people have moving joints and are not rigid, we treat them as rigid bodies and consider the shape changes caused by the joint motions as noisy measurements. Instance segmentation and tracking are realized with the method described in Section VI-B. The computation frequency of the mapping component reaches 10 Hz with the hardware specified in Section VII-D when the map size is 12.8 m. More results, including real-time tests conducted in Delft using a ZED 2 camera, can be found in the attached video. We also tested our method on the semantic mapping task of the KITTI-360 dataset [70]. The results show that our method achieves the highest mIoU performance. Details can be found in Appendix B.

[3]ZED 2 Camera: [Online]. Available: https://www.stereolabs.com/products/zed-2

VIII. Conclusion

This article presents a dynamic instance-aware semantic map based on an S^{2}MC-PHD filter. Experimental results on the Virtual KITTI 2 dataset demonstrate significant improvements in semantic and instance segmentation performance for possibly moving objects, such as cars, by over 45% and 60%, respectively, in both 2-D and 3-D, compared to SOTA maps. Moreover, the segmentation performance of static objects also excels, benefiting from the multihypotheses nature of the proposed method. Regarding occupancy estimation, our map outperforms SOTA maps by at least 30% in terms of AHD and ADm under different noise conditions while maintaining a comparable F1. In addition, an online memory enhancement module has been introduced and shown to improve the map's response to previously observed objects by 12.9% and 18.8% in 2-D and 3-D, respectively. It is worth noting that the proposed method introduces a way to cohesively integrate the filtering-based mapping method with object motion prediction and memory-based geometric information conjecture, respectively, in the prediction and particle birth steps. While a constant velocity model and an online template matching method are used in the current implementation, extensions to learning-based motion prediction and 3-D shape estimation or generation methods will be considered in future works, aiming to combine robustness and consistency inherent in the filtering-based method with the scalability and adaptability of learning-based methods.

Appendix A
Activation Bounding Box

The activation space is used to determine the range of pyramid subspaces whose particles should be updated with a measurement point [15]. The general idea is to ignore the particles in the pyramid subspaces that make the following condition true: for any particle $\tilde{\boldsymbol{x}}_k^{(i)}$ in the pyramid subspace, $g_k(\boldsymbol{z}_k|\tilde{\boldsymbol{x}}_k^{(i)}) \leq \epsilon$, where $g_k(\boldsymbol{z}_k|\tilde{\boldsymbol{x}}_k^{(i)})$ is the likelihood described in (16) and $\epsilon \approx 0$ is a threshold. As is mentioned in Section IV-A, the Pinhole model is used to realize the pyramid subspace division. Thus, each pixel in the image plane corresponds to a pyramid subspace, as Fig. 1(c) shows, and we can calculate an activation bounding box in the image plane to define the activation space. Any particle whose projection on the image plane is outside the bounding box can be ignored, while the weights of particles whose projections are inside the bounding box should be either increased or decreased in the update step. We illustrate the Pinhole model and the activation bounding box in Fig. 12.

Since $F_{gt}(\tilde{\boldsymbol{x}}_k^{(i)}) \leq 1$ and $T_r(\boldsymbol{z}_k, \tilde{\boldsymbol{x}}_k^{(i)}) \leq 1$, the activation bounding box can be determined by finding the range of pixel position (u, v), whose corresponding $\tilde{\boldsymbol{x}}_k^{(i)}$ can possibly satisfy $\mathcal{N}(\boldsymbol{z}_k; \tilde{\boldsymbol{x}}_k^{(i)}, \boldsymbol{\Sigma}) \leq \epsilon$, according to (16). Let $P = (x_1, y_1, z_1)$ and $Q = (x_2, y_2, z_2)$, $z_1, z_2 > 0$, be the position points of a measurement point $\boldsymbol{z}_k$ and a particle $\tilde{\boldsymbol{x}}_k^{(i)}$ in the camera frame, respectively. $P' = (u_1, v_1)$ and $Q' = (u_2, v_2)$ are the projections of P and Q on the image plane. As is used in [15], we assume the covariance matrix $\boldsymbol{\Sigma}$ as a diagonal matrix with the same diagonal value $\rho(z_1)$, where $\rho(z_1)$ can be estimated by experiments with a sensor [64]. Then, the particles that satisfy the condition $\mathcal{N}(\boldsymbol{z}_k; \tilde{\boldsymbol{x}}_k^{(i)}, \boldsymbol{\Sigma}) > \epsilon$ are in a sphere with radius l

$$l = \sqrt{2\rho^2(z_1) \ln\left(\frac{1}{(2\pi)^{\frac{3}{2}}\rho^3(z_1)\,\epsilon}\right)}. \tag{25}$$

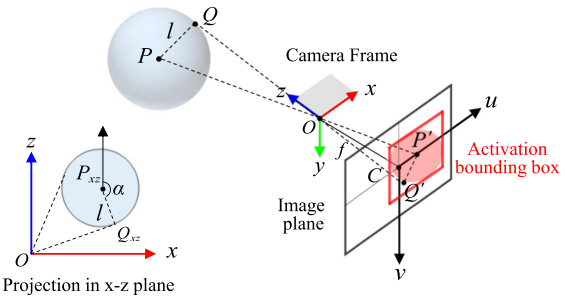


Fig. 12. Illustration of calculating the activation bounding box when using the Pinhole model. P is the position of a measurement point, and Q represents the position of a particle. P' and Q' are the projections of P and Q on the image plane. C is the center of the image. f is the focal length of the camera. l is the radius of the sphere which represents the surface where the Gaussian probability density is the same as a given threshold. The activation bounding box of the measurement point is the bounding box of the sphere's projected ellipse on the image plane. A red rectangle is used to illustrate the bounding box. The projection of the sphere on the x–z plane is shown in the left bottom.

The projection of this sphere on the image plane can be proved to be an ellipse. The activation bounding box is calculated by finding the minimum and maximum values of u_2 and v_2 in this ellipse, given $P = (x_1, y_1, z_1)$ and l. Take u_2 as an example. Since $u_2 = f\frac{x_2}{z_2}$, where f is the focal length, the minimum and maximum values of u_2 are equivalent to those of $\frac{x_2}{z_2}$. Considering the projection of the sphere on the x–z plane, $\frac{x_2}{z_2}$ can be represented by

$$\mathrm{f}(\alpha) = \frac{x_2}{z_2} = \frac{x_1 + l\sin(\alpha)}{z_1 + l\cos(\alpha)} \tag{26}$$

where α is the angle shown in Fig. 12. By solving $\mathrm{f}'(\alpha) = 0$, the minimum and maximum values of $\mathrm{f}(\alpha)$ is taken when

$$\alpha = 2\arctan\left(\frac{x_1 \pm \sqrt{x_1^2 + z_1^2 - l^2}}{z_1 - l}\right) \tag{27}$$

if $x_1^2 + z_1^2 - l^2 \geq 0$ and $z_1 \neq l$. These conditions hold because otherwise, the origin is in the sphere, and Q can be behind the camera. Then, the minimum and maximum values of u_2 can be calculated by $u_2 = f \cdot \mathrm{f}(\alpha)$. The minimum and maximum values of v_2 can be calculated similarly.

Appendix B
Additional Results

This section presents the performance of our map in the semantic mapping task on the KITTI-360 dataset [70]. The task evaluates global static semantic mapping performance across four test sequences using metrics such as Acc., Comp., F1, and mIoU over classes. To generate the global map, we assume all objects are static and accumulate the voxels from our map into a global map. Localization data is obtained from

TABLE VI
SEMANTIC MAPPING RESULTS IN KITTI-360 DATASET [70]

Method	Acc.	Comp.	F1	mIoU
ORB-SLAM2 [71] + PSPNet [74]	81.77	**74.89**	**78.15**	32.48
SUMA++ [26]	**90.98**	64.19	75.27	19.40
Ours + Preprocessing [71], [72]	79.15	72.45	75.64	**37.59**

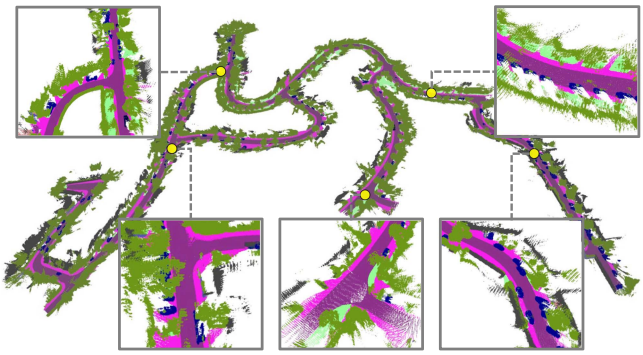


Fig. 13. Semantic mapping result of our method on test sequence 03 of the KITTI-360 [70] semantic mapping challenge. Five specific locations, marked with yellow spots, are enlarged and illustrated for a detailed view. The color of each label follows the scheme used in the Cityscapes dataset [65].

TABLE VII
PARAMETERS USED IN EXPERIMENTS

Param.	Value	Param.	Value	Param.	Value
P_d	0.98	P_s	1	κ_k	0.01
$\boldsymbol{Q}$	$0.01\boldsymbol{I}$	$\boldsymbol{\Sigma}$	$(10^{-3}d + 10^{-2})\boldsymbol{I}$	P_{tr}	0.5
S	1	$\Delta\bar{k}$	5	L_b	5
$w_{b,k}$	0.001	$\mathrm{Thr}_{\mathrm{score}}$	0.6	l_{voxel}	0.2 m
$\mathrm{Thr}_{\mathrm{occ}}$	0.8	$N^{\mathbb{V}}_{\mathrm{max}}$	8	$N^{\mathrm{bbox}}_{\mathrm{pix}}$	5 pix

ORB-SLAM2 [71], and the segmentation model used is CMNext [72], with pretrained weights on KITTI-360. Depth images are generated using SGM [73], without applying any filters. The benchmark results are summarized in Table VI. Our map achieves the best performance in terms of mIoU, outperforming the second-best map by 5.11. Fig. 13 illustrates the result of our map on the Test Sequence 03.

We further conducted an ablation study by comparing the results of using our map with directly accumulating semantic points acquired using ORB-SLAM2, CMNext, and SGM, in the first global map of Sequence 0. Our map significantly improves accuracy from 29.6 to 72.8 and the F1 from 44.9 to 76.5, demonstrating its denoising capability.

APPENDIX C
PARAMETERS

The parameters used in the experiments are listed in Table VII. To enhance computational efficiency, several simplifications were made: the clutter intensity $\kappa_k(\boldsymbol{z}_k)$ was simplified to a constant κ_k, the prediction covariance $\boldsymbol{Q}$ was simplified to a constant diagonal matrix, and the positional noise covariance $\boldsymbol{\Sigma}$ of the depth camera was simplified to a diagonal matrix that changes linearly with the depth d. $\mathrm{Thr}_{\mathrm{score}}$ represents the matching score threshold used in Section V-D2. $N^{\mathrm{bbox}}_{\mathrm{pix}}$ denotes the pixel number from the center of the neighbor bounding box in Section VI-A3 to its edges. l_{voxel} is the size of the voxel subspace, and $N^{\mathbb{V}}_{\mathrm{max}}$ specifies the maximum number of particles per voxel.

The occupancy threshold $\mathrm{Thr}_{\mathrm{score}}$ was determined through uniform sampling, as described in Section VII-A, and was identified to be 0.2 for the tests with depth noise. The rest of the parameters were tuned based on experience. Specifically, κ_k, $\boldsymbol{\Sigma}$, and $\boldsymbol{Q}$ need to be adjusted to account for different depth camera noise characteristics. In the tests with depth noise, $\kappa_k = 0.4$, $\boldsymbol{\Sigma} = (0.03d + 0.1)\boldsymbol{I}$, and $\boldsymbol{Q} = 0.05\boldsymbol{I}$ were used. In real-world experiments (see Section VII-E), κ_k and $\boldsymbol{\Sigma}$ were further adjusted to 0.5 and $(0.02d + 0.3)\boldsymbol{I}$, respectively. Since the panoptic segmentation and tracking errors usually remain consistent between simulation and real-world environments, parameters related to these processes, such as P_{tr}, S, and $\Delta\bar{k}$, were not adjusted.

ACKNOWLEDGMENT

Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.

REFERENCES

[1] B.-S. Kim, P. Kohli, and S. Savarese, "3D scene understanding by Voxel-CRF," in *Proc. IEEE/CVF Comput. Vis. Pattern Recognit. Conf.*, 2013, pp. 1425–1432.
[2] S. Yang, Y. Huang, and S. Scherer, "Semantic 3D occupancy mapping through efficient high order CRFs," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 590–597.
[3] N. Sunderhauf, T. T. Pham, Y. Latif, M. Milford, and I. Reid, "Meaningful maps with object-oriented semantic mapping," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2017, pp. 5079–5085.
[4] A. Rosinol et al., "Kimera: From SLAM to spatial perception with 3D dynamic scene graphs," *Int. J. Robot. Res.*, vol. 40, no. 12–14, pp. 1510–1546, 2021.
[5] M. Grinvald et al., "Volumetric instance-aware semantic mapping and 3D object discovery," *IEEE Robot. Autom. Lett.*, vol. 4, no. 3, pp. 3037–3044, Jul. 2019.
[6] G. Narita, T. Seno, T. Ishikawa, and Y. Kaji, "PanopticFusion: Online volumetric semantic mapping at the level of stuff and things," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 4205–4212.
[7] D. Seichter, B. Stephan, S. B. Fischedick, S. Müller, L. Rabes, and H.-M. Gross, "PanopticNDT: Efficient and robust panoptic mapping," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2023, pp. 7233–7240.
[8] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2017, pp. 2980–2988.
[9] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "YOLACT: Real-time instance segmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 9156–9165.
[10] C. Lyu et al., "RTMDet: An empirical study of designing real-time object detectors," 2022, *arXiv:2212.07784*.
[11] B. Wen and K. Bekris, "Bundletrack: 6D pose tracking for novel objects without instance or category-level 3D models," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 8067–8074.
[12] C. Wang et al., "6-PACK: Category-level 6D pose tracker with anchor-based keypoints," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2020, pp. 10059–10066.
[13] B. Wen et al., "BundleSDF: Neural 6-DoF tracking and 3D reconstruction of unknown objects," in *Proc. IEEE/CVF Comput. Vis. Pattern Recognit. Conf.*, 2023, pp. 606–617.
[14] J. Wilson et al., "MotionSC: Data set and network for real-time semantic mapping in dynamic environments," *IEEE Robot. Autom. Lett.*, vol. 7, no. 3, pp. 8439–8446, Jul. 2022.
[15] G. Chen, W. Dong, P. Peng, J. Alonso-Mora, and X. Zhu, "Continuous occupancy mapping in dynamic environments using particles," *IEEE Trans. Robot.*, vol. 40, pp. 64–84, Oct. 2023.

[16] R. Danescu, F. Oniga, and S. Nedevschi, "Modeling and tracking the driving environment with a particle-based occupancy grid," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 4, pp. 1331–1342, Dec. 2011.

[17] 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, 2017.

[18] A. Hornung, M. W. Kai, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Auton. Robots*, vol. 34, no. 3, pp. 189–206, 2013.

[19] V. Usenko, L. V. Stumberg, A. Pangercic, and D. Cremers, "Real-time trajectory replanning for MAVs using uniform b-splines and 3D circular buffer," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 215–222.

[20] J. Tordesillas, B. T. Lopez, M. Everett, and J. P. How, "FASTER: Fast and safe trajectory planner for navigation in unknown environments," *IEEE Trans. Robot.*, vol. 38, no. 2, pp. 922–938, Apr. 2022.

[21] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3D euclidean signed distance fields for on-board MAV planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 1366–1373.

[22] L. Schmid, O. Andersson, A. Sulser, P. Pfreundschuh, and R. Siegwart, "Dynablox: Real-time detection of diverse dynamic objects in complex environments," *IEEE Robot. Autom. Lett.*, vol. 8, no. 10, pp. 6259–6266, Oct. 2023.

[23] H. Wu, Y. Li, W. Xu, F. Kong, and F. Zhang, "Moving event detection from LiDAR point streams," *Nature Commun.*, vol. 15, no. 1, Jan. 2024, Art. no. 345.

[24] L. Schmid, M. Abate, Y. Chang, and L. Carlone, "Khronos: A unified approach for spatio-temporal metric-semantic SLAM in dynamic environments," Feb. 2024, *arXiv:2402.13817*.

[25] B. Mersch, T. Guadagnino, X. Chen, I. Vizzo, J. Behley, and C. Stachniss, "Building volumetric beliefs for dynamic environments exploiting map-based moving object segmentation," *IEEE Robot. Autom. Lett.*, vol. 8, no. 8, pp. 5180–5187, Aug. 2023.

[26] X. Chen, A. Milioto, E. Palazzolo, P. Giguère, J. Behley, and C. Stachniss, "SuMa++: Efficient LiDAR-based semantic SLAM," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 4530–4537.

[27] J.-W. Liu et al., "DeVRF: Fast deformable voxel radiance fields for dynamic scenes," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 36762–36775.

[28] A. Cao and J. Johnson, "HexPlane: A fast representation for dynamic scenes," in *Proc. IEEE/CVF Comput. Vis. Pattern Recognit. Conf.*, 2023, pp. 130–141.

[29] J. Luiten, G. Kopanas, B. Leibe, and D. Ramanan, "Dynamic 3D Gaussians: Tracking by persistent dynamic view synthesis," in *Proc. IEEE Int. Conf. 3D Vis.*, 2024, pp. 800–809.

[30] G. Wu et al., "4D Gaussian splatting for real-time dynamic scene rendering," in *Proc. IEEE/CVF Comput. Vis. Pattern Recognit. Conf.*, 2024, pp. 20310–20320.

[31] M. Youngjae, K. Do-Un, and C. Han-Lim, "Kernel-based 3-D dynamic occupancy mapping with particle tracking," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2021, pp. 5268–5274.

[32] Z. Zeng, Y. Zhou, O. C. Jenkins, and K. Desingh, "Semantic mapping with simultaneous object detection and localization," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2018, pp. 911–918.

[33] C. Yu et al., "DS-SLAM: A semantic visual SLAM towards dynamic environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2018, pp. 1168–1174.

[34] Y. Nakajima and H. Saito, "Efficient object-oriented semantic mapping with object detector," *IEEE Access*, vol. 7, pp. 3206–3213, 2019.

[35] L. Gan, R. Zhang, J. W. Grizzle, R. M. Eustice, and M. Ghaffari, "Bayesian spatial kernel smoothing for scalable dense semantic mapping," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 790–797, Apr. 2020.

[36] D. Seichter, P. Langer, T. Wengefeld, B. Lewandowski, D. Hochemer, and H.-M. Gross, "Efficient and robust semantic mapping for indoor environments," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2022, pp. 9221–9227.

[37] L. Schmid et al., "Panoptic Multi-TSDFs: A flexible representation for online multi-resolution volumetric mapping and long-term dynamic scene consistency," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2022, pp. 8018–8024.

[38] J. Wilson et al., "Convolutional Bayesian kernel inference for 3D semantic mapping," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2023, pp. 8364–8370.

[39] J. Wilson et al., "ConvBKI: Real-time probabilistic semantic mapping network with quantifiable uncertainty," *IEEE Trans. Robot.*, vol. 40, pp. 4648–4667, Sep. 2024.

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

[41] Y. Ding, L. Huang, and J. Zhong, "Multi-scale Occ: 4th place solution for CVPR 2023 3D occupancy prediction challenge," 2023, *arXiv:2306.11414*.

[42] C. Sima et al., "Scene as occupancy," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 8372–8381.

[43] X. Tian et al., "Occ3D: A large-scale 3D occupancy prediction benchmark for autonomous driving," in *Proc. Adv. Neural Inf. Process. Syst.*, 2023, vol. 36, pp. 64318–64330.

[44] H. Jiang et al., "Symphonize 3D semantic scene completion with contextual instance queries," in *Proc. IEEE/CVF Comput. Vis. Pattern Recognit. Conf.*, 2024, pp. 20258–20267.

[45] Y. Li et al., "SSCBench: A large-scale 3D semantic scene completion benchmark for autonomous driving," in *Proc. IEEE/RSJ Intl. Conf. Intell. Robot. Syst.*, 2024, pp. 13333–13340.

[46] R. P. S. Mahler, "Multitarget Bayes filtering via first-order multitarget moments," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 39, no. 4, pp. 1152–1178, Oct. 2003.

[47] B. Ristic, *Particle Filters for Random Set Models*. Berlin, Germany: Springer, 2013.

[48] G. Maggiolino, A. Ahmad, J. Cao, and K. Kitani, "Deep OC-sort: Multi-pedestrian tracking by adaptive re-identification," in *Proc. IEEE Int. Conf. Image Process*, 2023, pp. 3025–3029.

[49] S. Xu et al., "PP-YOLOE: An evolved version of YOLO," 2022, *arXiv:2203.16250*.

[50] M. Aygun et al., "4D panoptic LiDAR segmentation," in *Proc. IEEE/CVF Comput. Vis. Pattern Recognit. Conf.*, 2021, pp. 5523–5533.

[51] J. Huang, S. Yang, T.-J. Mu, and S.-M. Hu, "ClusterVO: Clustering moving instances and estimating visual odometry for self and surroundings," in *Proc. IEEE/CVF Comput. Vis. Pattern Recognit. Conf.*, 2020, pp. 2165–2174.

[52] B. Bescos, C. Campos, J. D. Tardós, and J. Neira, "DynaSLAM II: Tightly-coupled multi-object tracking and SLAM," *IEEE Robot. Autom. Lett.*, vol. 6, no. 3, pp. 5191–5198, Jul. 2021.

[53] Y. Qiu, C. Wang, W. Wang, M. Henein, and S. Scherer, "Airdos: Dynamic SLAM benefits from articulated objects," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2022, pp. 8047–8053.

[54] H. Ebbinghaus, "Memory: A contribution to experimental psychology," *Ann. Neurosciences*, vol. 20, no. 4, 2013, Art. no. 155.

[55] B. Ristic, D. Clark, and B. N. Vo, "Improved SMC implementation of the PHD filter," in *Proc. Int. Conf. Inf. Fusion*, 2010, pp. 1–8.

[56] G. Casella, C. P. Robert, and M. T. Wells, "Generalized accept-reject sampling schemes," *Lecture Notes-Monograph Ser.*, vol. 45, pp. 342–347, 2004.

[57] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun.*, vol. 24, no. 6, pp. 381–395, 1981.

[58] B. Cheng, I. Misra, A. G. Schwing, A. Kirillov, and R. Girdhar, "Masked-attention mask transformer for universal image segmentation," in *Proc. IEEE/CVF Comput. Vis. Pattern Recognit. Conf.*, 2022, pp. 1280–1289.

[59] K. Chen et al., "MMDetection: Open mmlab detection toolbox and benchmark," 2019, *arXiv:1906.07155*.

[60] D. DeTone, T. Malisiewicz, and A. Rabinovich, "SuperPoint: Self-supervised interest point detection and description," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops*, 2018, pp. 337–349.

[61] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, "SuperGlue: Learning feature matching with graph neural networks," in *Proc. IEEE/CVF Comput. Vis. Pattern Recognit. Conf.*, 2020, pp. 4937–4946.

[62] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig, "VirtualWorlds as proxy for multi-object tracking analysis," in *Proc. IEEE/CVF Comput. Vis. Pattern Recognit. Conf.*, 2016, pp. 4340–4349.

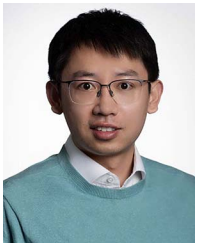
[63] Y. Cabon, N. Murray, and M. Humenberger, "Virtual KITTI 2," 2020, *arXiv:2001.10773*.

[64] A. Handa, T. Whelan, J. McDonald, and A. J. Davison, "A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2014, pp. 1524–1531.

[65] M. Cordts et al., "The cityscapes dataset for semantic urban scene understanding," in *Proc. IEEE/CVF Comput. Vis. Pattern Recognit. Conf.*, 2016, pp. 3213–3223.

[66] L. Wang, H. Ye, Q. Wang, Y. Gao, C. Xu, and F. Gao, "Learning-based 3D occupancy prediction for autonomous navigation in occluded environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 4509–4516.

[67] A. Elhafsi, B. Ivanovic, L. Janson, and M. Pavone, "Map-predictive motion planning in unknown environments," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2020, pp. 8552–8558.
[68] A. Zhang et al., "Towards robust robot 3D perception in urban environments: The UT campus object dataset," *IEEE Trans. Robot*, vol. 40, pp. 3322–3340, May 2024.
[69] A. Zhang et al., "UT campus object dataset (CODA)," 2023, doi: 10.18738/T8/BBOQMV.
[70] Y. Liao, J. Xie, and A. Geiger, "KITTI-360: A novel dataset and benchmarks for urban scene understanding in 2D and 3D," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 3, pp. 3292–3310, Mar. 2023.
[71] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras," *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1255–1262, Oct. 2017.
[72] J. Zhang et al., "Delivering arbitrary-modal semantic segmentation," in *Proc. IEEE/CVF Comput. Vis. Pattern Recognit. Conf.*, 2023, pp. 1136–1147.
[73] D. Hernandez-Juarez, A. Chacón, A. Espinosa, D. Vázquez, J. C. Moure, and A. M. López, "Embedded real-time stereo estimation via semi-global matching on the GPU," *Procedia Comput. Sci.*, vol. 80, pp. 143–153, 2016.
[74] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proc. IEEE/CVF Comput. Vis. Pattern Recognit. Conf.*, 2017, pp. 6230–6239.



Gang Chen (Member, IEEE) received the B.E. and Ph.D. degrees in mechanical engineering from Shanghai Jiao Tong University, Shanghai, China, in 2016 and 2022, respectively.

He is currently a Postdoctoral Researcher with the Cognitive Robotics Department, Delft University of Technology, Delft, The Netherlands. His research interests include perception and perception-aware planning for the navigation of single- and multirobot systems, with a special focus on dynamic environments.

Zhaoying Wang (Graduate Student Member, IEEE) received the B.E. degree in mechanical engineering from Wuhan University, Wuhan, China, in 2019. He is currently working toward the Ph.D. degree with the State Key Laboratory of Mechanical System and Vibration, Shanghai Jiao Tong University, Shanghai, China.

His research interests include visual inertial odometry, SLAM, and multirobot relative localization and mapping.

Wei Dong (Member, IEEE) received the B.S. and Ph.D. degrees in mechanical engineering from Shanghai Jiao Tong University, Shanghai, China, in 2009 and 2015, respectively.

He is currently a tenured Associate Professor with the Robotic Institute, School of Mechanical Engineering, Shanghai Jiao Tong University. In 2022, he was selected into the Shanghai Rising-Star Program for distinguished young scientists. His research interests include active perception and cooperation of unmanned systems.

Javier Alonso-Mora (Senior Member, IEEE) received the Ph.D. degree in robotics from ETH Zurich, Zurich, Switzerland, in 2014.

He is currently an Associate Professor with the Cognitive Robotics Department, Delft University of Technology, Delft, The Netherlands, where he leads the Autonomous Multi-Robots Lab. Before joining TU Delft, he was a Postdoctoral Associate with the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA. His main research interest is in navigation, motion planning, learning, and control of autonomous mobile robots, and teams thereof, that interact with other robots and humans in dynamic and uncertain environments.

Dr. Alonso-Mora is the recipient of multiple awards, including the IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING Best Paper Award (2024), an ERC Starting Grant (2021), and the ICRA Best Paper Award on Multi-Robot Systems (2019). He is currently an Associate Editor for IEEE TRANSACTIONS ON ROBOTICS and for *Springer Autonomous Robots*.