Self-Supervised Diffusion-Based Scene Flow Estimation and Motion Segmentation With 4D Radar

Yufei Liu, Xieyuanli Chen, *Member, IEEE*, Neng Wang, Stepan Andreev, Alexander Dvorkovich, Rui Fan, *Senior Member, IEEE*, and Huimin Lu, *Member, IEEE*

***Abstract*—Scene flow estimation (SFE) and motion segmentation (MOS) using 4D radar are emerging yet challenging tasks in robotics and autonomous driving applications. Existing LiDAR- or RGB-D-based point cloud processing methods often deliver suboptimal performance on radar data due to radar signals' highly sparse, noisy, and artifact-prone nature. Furthermore, for radar-based SFE and MOS, the lack of annotated datasets further aggravates these challenges. To address these issues, we propose a novel self-supervised framework that exploits denoising diffusion models to effectively handle radar noise inputs and predict point-wise scene flow and motion status simultaneously. To extract key features from the raw input, we design a transformer-based feature encoder tailored to address the sparsity of 4D radar data. Additionally, we generate self-supervised segmentation signals by exploiting the discrepancy between robust rigid ego-motion estimates and scene flow predictions, thereby eliminating the need for manual annotations. Experimental evaluations on the View-of-Delft (VoD) dataset and TJ4DRadSet demonstrate that our method achieves state-of-the-art performance for both radar-based SFE and MOS.**

***Index Terms*—Autonomous driving, 4D radar perception, motion segmentation, scene flow estimation.**

I. Introduction

4D Millimeter-Wave (mmWave) radar [1], [2], [3], [4], [5], [6] captures both 3D positions and radial relative velocities (RRVs) of measurement points in a scene, providing additional observation information, especially on dynamic objects. Moreover, radar sensors exhibit greater robustness under adverse weather conditions compared to camera and LiDAR sensors, making them highly valuable for robots and autonomous vehicles operating in real-world environments. In light of these advantages, several studies have initiated exploration in radar-based applications [1], including semantic segmentation [2], 3D occupancy prediction [3], and object tracking [4].

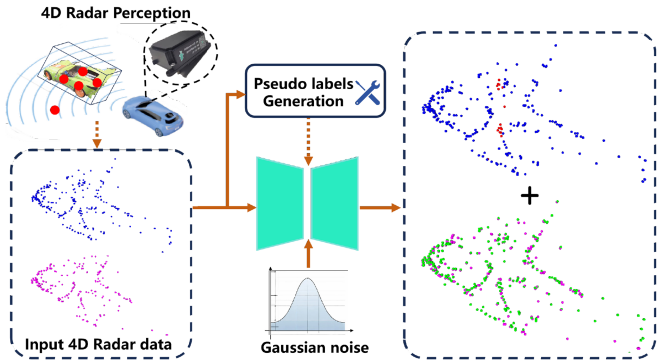

Fig. 1. RadarSFEMOS combines scene flow estimation and motion segmentation from sparse radar point cloud which takes two frames of 4D radar point cloud as input and outputs the both results.

Scene flow estimation (SFE) and motion segmentation (MOS) are critical and challenging tasks in 4D radar perception. Autonomous vehicles or robots exist in ego-motion, leading to complex motion patterns. The actual motion of the radar points is a combination of ego-motion and object motion. SFE describes a set of displacement vectors between two consecutive observations of a 3D scene, offering essential dynamic information, while MOS focus on determining the true motion state of each point in the world coordinate system. Both tasks are fundamental for achieving a comprehensive understanding of the dynamic scenes with multiple moving objects at varying speeds and directions. Although recent LiDAR- and image-based approaches [7], [8], [9] have demonstrated promising performance, directly applying these methods to radar data results in significant performance degeneration [10]. This degradation is partially due to the inherent sparsity of radar data and noise introduced by multi-path interference and specular reflections [11]. Furthermore, the high cost of labeling uneven radar data limits the scalability of supervised learning methods. Existing approaches [9], [12], [13] often rely on accurate ego-motion estimates from odometry or SLAM systems. However, such estimates may become unreliable when both the robot and surrounding objects are in motion, further complicating radar-based SFE and MOS tasks.

As shown in Fig. 1, we tackle the challenges of SFE and MOS simultaneously in sparse and noisy 4D radar point cloud

Received 13 January 2025; accepted 11 April 2025. Date of publication 22 April 2025; date of current version 1 May 2025. This article was recommended for publication by Associate Editor H. Caesar and Editor A. Valada upon evaluation of the reviewers' comments. This work was supported in part by the National Science Foundation of China under Grant 62403478, Grant 62473288, and Grant 62233013, in part by the Young Elite Scientists Sponsorship Program through CAST under Grant 2023QNRC001, and in part by the Fundamental Research Funds for the Central Universities. *(Corresponding author: Huimin Lu.)*

Yufei Liu, Xieyuanli Chen, Neng Wang, and Huimin Lu are with the College of Intelligence Science and Technology, and the National Key Laboratory of Equipment State Sensing and Smart Support, National University of Defense Technology, Changsha 410008, China (e-mail: lhmnew@nudt.edu.cn).

Stepan Andreev is with the Microwave Photonics Department, Telecommunications Center, Moscow Institute of Physics and Technology, 141700 Dolgoprudny, Moscow, Russia.

Alexander Dvorkovich is with the Multimedia Technology and Telecom Department, Telecommunications Center, Moscow Institute of Physics and Technology, 141700 Dolgoprudny, Moscow, Russia.

Rui Fan is with the College of Electronics and Information Engineering, Tongji University, Shanghai 200092, China.

The code and pre-trained weights of our method will be released at https://github.com/nubot-nudt/RadarSFEMOS.

Digital Object Identifier 10.1109/LRA.2025.3563829

by proposing novel solutions to mitigate noise and sparsity while ensuring robust performance and accurate point correspondences. To this end, we design a self-supervised framework, named RadarSFEMOS. It predicts point-wise scene flow and motion status simultaneously, eschewing the need for additional dependencies, such as manual annotations and odometric inputs. Initially, to extract key features from noisy radar point cloud, we incorporate diffusion models [14] into our network, leveraging its exceptional denoising capabilities. While point correspondences in LiDAR point clouds are often clearly discernible, radar points at long distances are not only sparse but also exhibit greater discrepancies in inter-point distances compared to closer points [10]. To address this, we employ a point-voxel correlation feature construction method to capture both short-range and long-range correspondences, effectively extracting features from all-pairs fields. Moreover, common feature encoders designed for LiDAR point clouds often struggle to perform effectively on highly sparse radar data. We design a Transformer-Based feature encoder composed of three self-attention layers, process the point cloud with the original resolution throughout the encoder to keep as much information as possible and enrich the points with high-level features. Finally, the SFE head produces the point-wise scene flow results. Based on that, RadarSFEMOS explicitly computes motion residuals by comparing scene flow estimations with rigid ego-motion, subsequently feeding this residual into a motion head for point-wise motion state prediction.

In summary, we make the following four key claims:

- We propose RadarSFEMOS, a novel self-supervised framework that simultaneously addresses radar-based scene flow estimation (SFE) and motion segmentation (MOS) trained by three task-specific losses, without relying on additional dependencies such as manual annotations or odometric information.
- We incorporate the diffusion model into our network and leverage its denoising capabilities to achieve more robust scene flow estimation, effectively addressing the noisy characteristics of 4D radar data.
- We design a novel Transformer-based 4D point cloud feature encoder that adeptly extracts key features for Point-Voxel correlation volume construction to enable effective point-level feature aggregation.
- Our proposed method achieves state-of-the-art (SOTA) performance for both the SFE and MOS tasks on the VoD dataset and TJ4DRadSet dataset.

II. Related Work

A. Scene Flow Estimation on Point Cloud

Most existing scene flow estimation methods are based on supervised learning and focus on LiDAR data. FlowNet3D [15] first proposes directly extracting point features from point cloud for scene flow estimation. Inspired by classical pyramid approaches, PointPWC-Net [16] estimates scene flow by constructing a cost volume at each feature pyramid level and refining it. FLOT [17] introduces optimal transport to establish correspondences between two point clouds. FlowStep3D [18] and PV-RAFT [19] utilize Gated Recurrent Units (GRU) to enhance prediction accuracy. Recently, Bi-PointFlowNet [20] introduces novel bidirectional layers for improved flow embedding. Some self-supervised SFE methods for point clouds have been designed mainly for dense LiDAR, with performance degrading significantly on radar data. SLIM [21] achieves the self-supervised SFE and MOS using only thresholds for distinguishing moving points. Similarly, Rigid3DSF [13] introduces instance-level rigidity in weakly supervised relying on object-level abstraction. RigidFlow [22] introduces local rigidity priors in self-supervised learning based on the assumption that a scene consists of several rigidly moving components.

B. Motion Segmentation on Point Cloud

Motion segmentation (MOS) is also called moving object segmentation. Many existing LiDAR-based methods rely on multi-frame temporal inputs and odometry information, convert the 3D raw point cloud into a 2D image plane, such as range images [23], [24], [25] or Bird's Eye View (BEV) images [21], [26] to facilitate online LiDAR point cloud processing. Chen et al. [24] first exploit the residual range images for online MOS. Their proposed method gets rid of the restraint of the pre-built maps and can be directly applied in a SLAM pipeline. 4DMOS [12] employs computationally efficient 4D convolutions to simultaneously extract spatial and temporal features, enabling the prediction of moving object confidence scores for all points in the sequence. Building on the concept of 4DMOS, Kreutz et al. [27] propose a self-supervised learning framework that leverages a 4D LiDAR representation to identify moving objects. However, their method is constrained to environments with stationary backgrounds. InsMOS [28] not only predicts point-wise motion labels but also detects instance-level information for key traffic participants. Different from existing methods, our RadarSFEMOS doesn't need explicit odometry input, making our method more robust when the odometry is not reliable.

C. 4D Radar Scene Flow and Segmentation

Due to the ability to operate in all weather conditions and the robust performance in challenging environments, 4D mmWave radar is essential for autonomous driving applications. RaFlow [10] is the first work to estimate scene flow on 4D radar which proposes three novel losses to address the challenges posed by complex and intractable radar data. Ding et al. [29] employ multiple cross-modal constraints for effective model training, successfully integrating 4D radar into modern autonomous driving architectures. In the MOS task, Radar radial velocity only reflects the relative radial velocity, while MOS aims to determine the true motion state of each point in the world coordinate system, which is especially challenging in scenarios with multiple moving objects at varying speeds and directions. RadarMOSEVE [6] introduces a novel transformer network for motion segmentation, yet its training necessitates manual annotation. Radar Velocity Transformer [30] performs single-scan MOS by incorporating the valuable velocity information throughout each module of the network. Gaussian Radar Transformer [2] includes the newly introduced Gaussian transformer layer to accurately perform sparse, single-scan segmentation. Zeller et al. [31] enhance radar scans with temporal information to tackle the challenge of moving instance segmentation. Unlike previous methods, our proposed RadarSFEMOS simultaneously addresses radar-based SFE and MOS tasks without requiring any additional annotations, greatly enhancing its practical utility.

III. Our Approach

A. Pipeline Overview and Problem Definition

As shown in Fig. 2, given two consecutive radar point cloud $\mathcal{P}_1 = \{p_i \in \mathbb{R}^3\}^N$ and $\mathcal{P}_2 = \{q_i \in \mathbb{R}^3\}^M$ with N, M

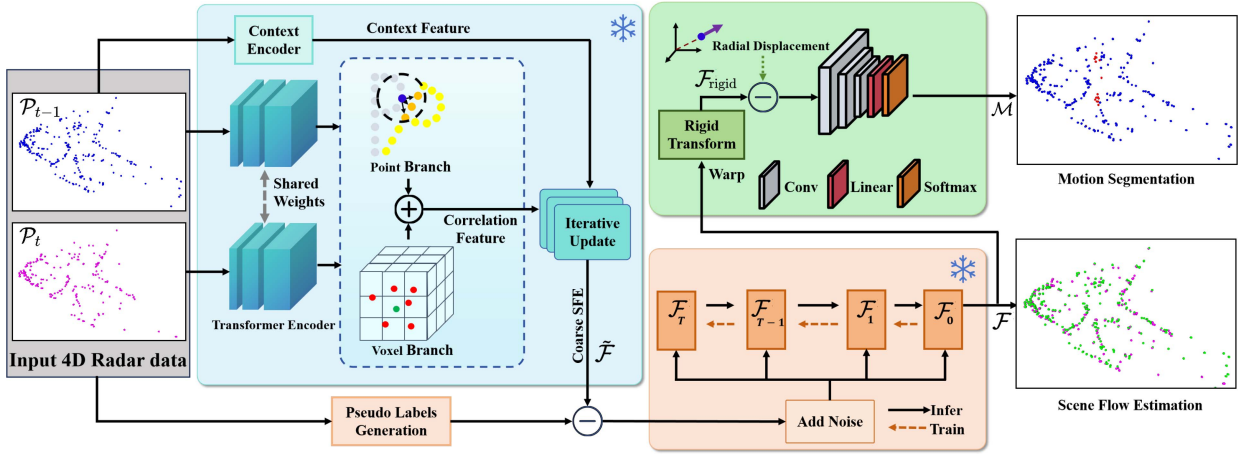

Fig. 2. Overview of RadarSFEMOS. Two point clouds $\mathcal{P}_{t-1}$ and $\mathcal{P}_t$ are input into the Transformer Encoder, generating feature for building correlation features. The coarse scene flow from the iteration update block is fed into the diffusion model for refining the estimation. As the residuals contain valuable motion information, we use a MOS head on residuals to obtain the MOS results $\mathcal{M}$ simultaneously for $\mathcal{P}_t$.

points. $\boldsymbol{p}_i, \boldsymbol{q}_j$ denote the 3D coordinates. RadarSFEMOS aims to estimate the scene flow vector $\mathcal{F} = \{\boldsymbol{f}_i \in \mathbb{R}^3\}_{i=1}^N$ from $\mathcal{P}_{t-1}$ to $\mathcal{P}_t$ and recognize the point's motion state $\mathcal{M}$, i.e., moving or static in scan $\mathcal{P}_{t-1}$. SFE and MOS are linked by the vehicle's rigid ego-motion $\boldsymbol{f}_{\text{rigid}}$, which also causes the scene flow of static objects. By establishing this connection, the scene flow $\boldsymbol{f}$ can be decomposed into two components: $\boldsymbol{f} = \boldsymbol{f}_{\text{rigid}} + \boldsymbol{f}_{\text{move}}$, where $\boldsymbol{f}_{\text{move}}$ corresponds to the motion of the surrounding moving objects, such as running vehicles and walking pedestrians. Besides, we denote the warped point of $\mathcal{P}_{t-1}$ as $\mathcal{P}'_{t-1} = \{\boldsymbol{p}'_i \in \mathbb{R}^3\}_{i=1}^N$ where $\boldsymbol{p}'_i = \boldsymbol{p}_i + \boldsymbol{f}_i$.

B. Coarse Scene Flow Estimation

1) Transformer-Based Feature Encoder: For highly sparse radar data, common LiDAR-based transformer modules typically exhibit suboptimal performance in capturing key features. To this end, we introduce a local neighbor aggregation strategy into our encoder, which can not only capture key local features even at relatively greater distances but also can optimize the point embedding.

In raw data, each point in the radar point clouds $\mathcal{P}_{t-1}$ and $\mathcal{P}_t$ is described by a C-dimensional feature vector, which encompasses not only the 3D spatial coordinates but also inherent features such as Radial Relative Velocity (RRV), Radar Cross Section (RCS), and power measurements. Due to the sparse nature, adjacent points in radar data may have relatively large distances. To effectively capture local salient features, we first design a neighbor embedding layer for extracting features from raw data, which consists of two convolution layers and two neighbor sampling and grouping layers. We employ a ball query mechanism for neighbor sampling operation, which is capable of acquiring relatively stable local features in regions with varying densities, thereby circumventing the feature extraction bias caused by differences in point cloud density. We denote the features after neighbor aggregation as $\mathbf{F}_{\text{emb}} \in \mathbb{R}^{C'}$. Subsequently, we employ a multi-attention layer to further refine the point embeddings and concatenate the multi embeddings to maintain multi-scale details. The specific operations can be represented as:

$$\mathbf{F}_1 = \text{Attention}^1(\mathbf{F}_{\text{emb}}), \tag{1}$$

$$\mathbf{F}_i = \text{Attention}^i(\mathbf{F}_{i-1}), \quad i = 2, 3, \tag{2}$$

$$\mathbf{F}_{\mathcal{P}_t} = \text{Linear}(\mathbf{F}_1 \oplus \mathbf{F}_2 \oplus \mathbf{F}_3), \tag{3}$$

where $\text{Attention}^i(\cdot)$ represents the i-th self-attention layer and Linear is the linear layer.

2) Iterative Flow Estimation: Consistent with prior research [18], [19], [21], an iterative update scheme is adopted for SFE derivation. Iterative flow estimation commences with the null initialization $\boldsymbol{f}_0 = 0$. Each iteration utilizes current estimation results and correlation features to establish precise point correspondences, thus iteratively refining the estimation.

Initially, a comprehensive correlation volume is constructed from pairwise feature similarities to derive correlation features. Point-wise features $\mathbf{F}_{\mathcal{P}_{t-1}}$ and $\mathbf{F}_{\mathcal{P}_t}$, produced by the Transformer-based feature encoder, are employed to compute a correlation map via matrix dot product: $\mathbf{C}_{\text{corr}} = \mathbf{F}_{\mathcal{P}_{t-1}} \cdot \mathbf{F}_{\mathcal{P}_t}$. The correlation volume $\mathbf{C}_{\text{corr}}$ is built only once and is kept as a lookup table for flow estimation in different steps. Thus point pairs with similar features have high correlation values. Based on this correlation map, point and voxel branches are employed for correlation feature construction as follows:

Point branch: One frequently employed approach for identifying neighboring points in 3D point cloud is the utilization of K-nearest neighbors (KNN) algorithm [15], [21]. Let $\mathcal{N}_k$ denotes the top-k nearest neighbors of $\mathcal{P}'_{t-1}$ in $\mathcal{P}_t$, the correlation feature between $\mathcal{P}'_{t-1}$ and $\mathcal{P}_t$ can be defined by:

$$\mathbf{C}_p(\mathcal{P}'_{t-1}, \mathcal{P}_t) = \max_k(\text{MLP}((\mathbf{C}_M(\mathcal{N}_k) \oplus (\mathcal{N}_k - \mathcal{P}'_{t-1}))), \tag{4}$$

$\mathbf{C}_M(\mathcal{N}_k)$ contains corresponding correlation values of $\mathbf{C}_{\text{corr}}$.

Voxel branch: The voxel branch first builds a voxel neighbor cube centered around $\mathcal{P}'_{t-1}$, which is a cube with $L = a \times a \times a$ sub-cubes. The side length of each sub-cube is r and $\mathcal{N}_r^{(l)}$ is the set of the $\mathcal{P}_{t-1}$ points that locate in the $l-$th sub-cube. The correlation values of the points inside each sub-cube are averaged to produce the sub-cube feature

$$\mathbf{e}_{\text{cube}}^{(l)} = \frac{1}{n_l} \sum_{\mathcal{N}_r^{(l)}} \mathbf{C}_M(\mathcal{N}_r^{(i)}), \tag{5}$$

where n_l is the set size of $\mathcal{N}_r^{(l)}$. The correlation feature from the voxel branch between $\mathcal{P}'_{t-1}$ and $\mathcal{P}_t$ can be expressed as:

$$\mathbf{C}_v(\mathcal{P}'_{t-1}, \mathcal{P}_t) = \text{MLP}\left(\mathbf{e}_{\text{cube}}^{(1)} \oplus \mathbf{e}_{\text{cube}}^{(2)} \oplus \ldots \oplus \mathbf{e}_{\text{cube}}^{(L)}\right). \quad (6)$$

We adopt the GRU-based [32] structure which enables iterative residual flow generation, thereby enhancing the accuracy of flow estimation. As a simplified variant of LSTM with fewer control gates, GRU can capture the long-term dependents of the context and produce better outputs than traditional RNNs, which are essential for the iterative scene flow update module. Furthermore, a content feature encoder, structurally similar to the transformer-based encoder but with independent weights, is implemented to extract context features from $\mathcal{P}_{t-1}$. These context features provide auxiliary information for GRU iterations. This enables the GRU to effectively incorporate current flow estimates and context features, iteratively refining the hidden state and enhancing flow estimation precision.

The iterative flow estimation begins with the initialize state $\hat{\boldsymbol{f}}_0 = 0$. In the $(s+1)$-th iteration, the estimation result is updated upon the current state,

$$\hat{\boldsymbol{f}}_{s+1} = \hat{\boldsymbol{f}}_s + \delta, \quad (7)$$

$$z_s = \sigma(\text{Conv}_{1d}([h_{s-1}, x_s], W_z)), \quad (8)$$

$$r_s = \sigma(\text{Conv}_{1d}([h_{s-1}, x_s], W_r)), \quad (9)$$

$$\hat{h}_s = \tanh(\text{Conv}_{1d}([r_s \odot h_{s-1}, x_s], W_h)), \quad (10)$$

$$h_s = (1 - z_s) \odot h_{s-1} + z_s \odot \hat{h_s}, \quad (11)$$

where W_z, W_r, and W_h are the weight matrices that can be learned during training. x_s is a concatenation of correlation fields, current flow $\hat{\boldsymbol{f}}_s$ and context features. $\odot$ is the Hadamard product, $[\cdot,\cdot]$ is a concatenation and $\sigma(\cdot)$ is the sigmoid activation function. Finally, the obtained hidden state h_s is fed into a small convolution layers to generate the iteration result δ. Eventually, the sequence converges to the final prediction.

C. Diffusion-Based Scene Flow Refinement

Even though we have obtained SFE results, the inherent radar noise may still affect the estimation accuracy. Inspired by the denoising capabilities of diffusion models, we integrate it into the network to effectively mitigate the noise, thus facilitating more accurate scene flow estimation.

1) Pseudo Ground Truth Generation: Diffusion model needs to add Gaussian noise to the ground truth and learn to recover the scene flow from pure noise. However, point-wise scene flow labeling is challenging. Built upon recent pseudo-label generation techniques [22], [33], we employ a piecewise rigid pseudo-label generation module. This module estimates flow for each rigid region by determining optimal rigid transformations, yielding comprehensive scene flow pseudo-labels.

Decomposing the point cloud into K rigid regions $\{\boldsymbol{G}^{(1)}, \boldsymbol{G}^{(2)}, \ldots, \boldsymbol{G}^{(K)}\}$, the piecewise rigid transform from $\mathcal{P}_{t-1}$ to $\mathcal{P}_t$ for the region $\boldsymbol{G}^{(K)}$ can be considered as an independent rigid body registration from $\boldsymbol{G}^{(K)}$ to $\mathcal{P}_t$:

$$[\boldsymbol{R}_k^*, \boldsymbol{t}_k^*] = \arg\min_{[\boldsymbol{R}_k, \boldsymbol{t}_k]} \|(\boldsymbol{R}_k \cdot \boldsymbol{G}^{(k)} + \boldsymbol{t}_k - \mathcal{P}_t)\|_2^2. \quad (12)$$

With $[\boldsymbol{R}_k^*, \boldsymbol{t}_k^*]$, the pseudo rigid scene flow estimate $\mathcal{F}^{(k)}$ for this region can be computed by:

$$\mathcal{F}^{(k)} = \boldsymbol{R}_k^* \boldsymbol{G}^{(k)} + \boldsymbol{t}_k^* - \boldsymbol{G}^{(k)}. \quad (13)$$

Combining the pseudo rigid SFE for all rigid regions, we obtain the final pseudo SFE labels for self-supervised training.

2) Forward Noising Process: Direct scene flow recovery from pure noise is difficult due to data distribution variations. We model the scene flow residual, defined as the difference between coarse scene flow and pseudo ground truth, as a diffusion latent variable. During training, a forward diffusion process progressively introduces Gaussian noise into the flow residual across T timesteps via a Markov chain:

$$q(d_{1:T}^i | d_0^i) = \Pi_{t=1}^T q\left(d_t^i | d_{t-1}^i\right), \quad (14)$$

where d_0^i indicates the flow residual for i-th point in $\mathcal{P}_t$ and d_t^i is the intermediate flow residual at timestamp t.

3) Reverse Denoising Process: To generate the flow residual with robustness to outliers, we resort to the reverse process of diffusion model. Basically, the reverse process can be represented as a parameterized Markov chain starting from a random noise $p(s_T^i)$:

$$p_\theta(d_{0:T}^i) = p(d_T^i)\Pi_{t=1}^T p_\theta(d_{t-1}^i | d_t^i), \quad (15)$$

where the reverse transition kernel $p_\theta(d_{t-1}^i | d_t^i)$ is approximated with a neural network. We follow [8] to directly learn the flow residual $\hat{d}_0^i$ by the denoising network $H_\theta(s_{t-1}^i | s_t^i, t)$.

D. Scene Flow-Guided Motion Segmentation

SFE and MOS capture complementary aspects of physical motion characteristics [34], where scene flow quantifies the magnitude of motion, while segmentation identifies the motion status. RRV measurements describe the moving speed of ambient objects relative to the sensor in the radial direction. The product of RRV measurement and time interval Δt can be seen as an approximation of the radial projection of the truth flow vector.

This insight is crucial to MOS because it can generate instructive signals for motion segmentation. Assuming the velocity of $\boldsymbol{p}_i$ keeps constant during time interval Δt between two scans, we can reach the following equation:

$$v_i^r \Delta t = \boldsymbol{f}_{\text{gt},i}^\top \cdot \frac{p_i}{||p_i||_2}, \quad (16)$$

where $\boldsymbol{f}_{\text{gt},i}$ is the true scene flow vector.

For radar or LiDAR, the majority of points in a scene are static, as the background typically has a more extensive structure compared to movable foreground instances. We intuitively assume that all points are stationary and then utilize the Kabsch algorithm [35] to get an intermediate transformation $\mathbf{T}_\text{r} \in SE(3)$. With $\mathbf{T}_\text{r}$, an intermediate rigid scene flow $\boldsymbol{f}_{\text{rigid}}$ can be computed accordingly. We utilize the radial projection of rigid flows $\boldsymbol{f}_{\text{rigid}}$ and the difference in measured RRV times Δt as residual feature:

$$\boldsymbol{f}_{\text{residual},i} = \left| \boldsymbol{f}_{\text{rigid},i}^\top \cdot \frac{p_i}{||p_i||_2} - v_i^r \Delta t \right|. \quad (17)$$

Given the residual feature for each point in the point cloud $\mathcal{P}_t$, we use an MLP consisting of three linear layers and two ReLU nonlinear layers to convert these features into the final logits for MOS.

E. Loss Function and Network Training

Since the accuracy of SFE significantly affects the MOS performance, our RadarSFEMOS employs a two-stage training process to expedite network convergence. Firstly, $\mathcal{L}_{\text{flow}}$ supervises the network to achieve precise SFE estimation. Subsequently, the SFE estimation module's parameters are frozen, and only the motion segmentation module is trained using L_{mos} supervision.

$\mathcal{L}_{\text{flow}}$ consists of two parts: Pseudo ground truth loss $\mathcal{L}_{\text{pg}}$, Doppler loss $\mathcal{L}_{\text{dop}}$. The overall loss function can be written as:

$$\mathcal{L}_{\text{flow}} = \mathcal{L}_{\text{pg}} + \mathcal{L}_{\text{dop}}. \tag{18}$$

Using the generated pseudo labels for supervision, we can achieve the self-supervised training of SFE networks with supervised loss functions:

$$\mathcal{L}_{\text{pg}} = \sum_{i=1}^{N} \|\boldsymbol{f}_{\text{est},i} - \boldsymbol{f}_{\text{pseudo}}\|_1. \tag{19}$$

Despite some inevitable measurement errors, we empirically found that RRV renders strong supervision signals for training scene flow estimators. Formally, we formulate a radial displacement loss based on (16):

$$\mathcal{L}_{\text{dop}} = \sum_{p_i \in \mathcal{P}_t} \left| \boldsymbol{f}_i^{\top} \frac{p_i}{||p_i||_2} - v_i^r \Delta t \right|. \tag{20}$$

The motion classification loss L_{mos} are optimized using a self-supervised loss that is based on the nearest neighbor (NN) errors e_i. Given a point-wise flow $\boldsymbol{f}_i$ and a nearest neighbor function as follows:

$$NN_{\mathcal{P}_t}(\boldsymbol{p}_i') = \arg\min_{\boldsymbol{p}_j \in \mathcal{P}_t} |\boldsymbol{p}_j - \boldsymbol{p}_i'|. \tag{21}$$

For every flow-corrected point, the NN-based error distance is:

$$e_i = |NN_{\mathcal{P}_t}(\boldsymbol{p}_i') - \boldsymbol{p}_i'|. \tag{22}$$

These errors are computed not only for the raw flow prediction $\mathcal{F}$, but also for its rigid counterpart $\mathcal{F}_{\text{rigid}}$, allowing for a comparison between the raw flow errors e_i and those of the rigid flow, denoted as $e_{r,i}$. To train the model, we apply the standard binary cross-entropy loss defined as follows:

$$\begin{aligned} L_{\text{mos}} = & -\sum_{p_i \in \mathcal{P}_t} [e_i < e_{r,i}] \log \sigma(m_{cls,i}) \\ & + [e_i \geq e_{r,i}] \log(1 - \sigma(m_{cls,i})). \end{aligned} \tag{23}$$

IV. Experimental Evaluation

A. Experimental Setup

Datasets: For our experiments, we use the View-of-Delft (VoD) dataset [39] and TJ4DRadSet [40]. VoD provides synchronized and calibrated data captured by co-located sensors, including a 64-beam LiDAR, an RGB camera, RTK-GPS/IMU based odometer and a 4D radar sensor. TJ4DRadSet dataset was collected across various driving scenarios, consisting of 44 consecutive sequences, all of which are thoroughly annotated with 3D bounding boxes and track IDs. We follow [29] to divide new splits from the official sets.

Ground truth labelling: We annotate ground truth scene flow by using object annotations (i.e., bounding boxes and track IDs) and ground truth radar ego-motion (calculated from the RTK-GPS/IMU based odometer). For points belonging to the static background, we label their flow vectors with the ground truth radar ego-motion. For foreground objects, we track the ID of each annotated bounding box across consecutive point clouds and compute their rigid transformation w.r.t the radar coordinate frame.

Training Details: We train SFE model for 30 epochs and MOS model for 50 epochs using the Adam optimizer [41]. The initial learning rate is set as 0.001 and decays by a factor of 0.9 after each epoch. Data augmentation is implemented by randomly rotating and translating each point cloud from the training set. During the pseudo-label generation phase, the source point cloud is decomposed into 16 supervoxels and treat these supervoxels as rigid moving regions. Our code is developed based on PyTorch [42].

B. SFE Performance Evaluation

Prior LiDAR-based SFE [15], [16], [17], [18], [21], [22] commonly use EPE, Acc Strict, Acc Relax as the major metric for evaluation. **EPE** $= \|\boldsymbol{f}_{\text{est},i} - \boldsymbol{f}_{\text{gt},i}\|_2$, is the end point error averaged on each point in meters. **Acc Strict (ACCS)** is the percentage of points whose EPE < 0.05 m or relative error $< 5\%$. **Acc Relax (ACCR)** is the percentage of points whose EPE $<$ 0.1 m or relative error $< 10\%$.

Directly applying EPE to radar point cloud lacks sensor-specific consideration, as 4D radars have lower resolution than LiDAR. Following [10], we adopt a Resolution-normalized EPE (RNE) to align our evaluation with LiDAR-based scene flow estimation. Two RNE-based accuracy metrics are: **SAS**: the percentage of points whose RNE $<$ 0.1 m or relative error $< 10\%$, and **RAS**: the percentage of points whose EPE $<$ 0.2 m or relative error $< 20\%$.

Additionally, we present results separately for static and moving points and compute their average to address class imbalance. For notation, 50-50 RNE refers to the average of Static RNE (SRNE) and Moving RNE (MRNE).

We quantitatively compare the performance of RadarSFEMOS to baselines on both VoD [39] and TJ4DRadSet [40] datasets, as shown in Tables I and II. Compared with non-learning, self-supervised, fully supervised and cross-modal supervision counterparts, RadarSFEMOS outperforms all self-supervised methods and even surpasses some fully and cross-modal supervised approaches in certain metrics. The Resolution-normalized EPE for SFE by our method is 0.074 m, with high accuracy in RAS exceeding 90%. We also present separate evaluation results for static and moving points. The qualitative results of SFE are shown in Fig. 3. As observed, our method demonstrates more accurate SFE in 4D radar data owing to our Diffusion-based network design.

C. MOS Performance Evaluation

For the MOS task, we evaluate our method against both self-supervised [10], [21] and fully supervised [6] approaches using the mean intersection-over-union (mIoU) and the accuracy of the moving objects (ACCM), which is defined as follows:

$$\text{IoU} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}}, \quad \text{ACCM} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{24}$$

TABLE I
SFE PERFORMANCE COMPARISON WITH OTHER BASELINE METHODS ON THE VOD DATASET

Methods	Sup	RNE↓	SAS↑	RAS ↑	MRNE↓	SRNE↓	50-50 RNE↓	EPE↓	ACCS↑	ACCR ↑
FlowNet3D [15]	Full	0.094	0.746	0.885	0.124	0.089	0.107	0.233	0.161	0.394
FLOT [17]	Full	0.091	0.745	0.902	0.122	0.087	0.105	0.228	0.155	0.366
CMFlow [29]	Cross	0.057	0.841	0.922	0.073	0.054	0.064	0.141	0.233	0.499
ICP [36]	None	0.138	0.523	0.843	0.148	0.137	0.142	0.344	0.019	0.106
SceneFF [37]	None	0.179	0.277	0.655	0.186	0.176	0.181	0.444	0.070	0.103
JustGW [38]	Self	0.167	0.324	0.695	0.175	0.164	0.169	0.416	0.057	0.122
PointPWCNet [16]	Self	0.173	0.516	0.795	0.123	0.179	0.151	0.433	0.048	0.163
FlowStep3D [18]	Self	0.133	0.487	0.831	0.146	0.130	0.138	0.331	0.032	0.123
SLIM [21]	Self	0.130	0.432	0.719	0.151	0.126	0.138	0.323	0.050	0.170
RaFlow [10]	Self	0.091	0.750	0.899	0.116	0.087	0.102	0.227	0.171	0.395
Ours(w.o. Diffusion)	Self	0.088	0.746	0.871	0.116	0.089	0.103	0.229	0.168	0.404
Ours	Self	**0.074**	**0.816**	**0.930**	**0.114**	**0.069**	**0.091**	**0.186**	**0.208**	**0.463**

The underlined value indicates the overall best result, while the bold value denotes the best result within self-supervised methods. Full represents full supervision and Cross indicates cross-modal supervision.

TABLE II
SFE PERFORMANCE COMPARISON WITH OTHER BASELINE METHODS ON THE TJ4DRADSET DATASET

Methods	Sup	RNE↓	SAS↑	RAS ↑	MRNE↓	SRNE↓	50-50 RNE↓	EPE↓	ACCS↑	ACCR ↑
FlowNet3D [15]	Full	0.103	0.719	0.824	0.129	0.094	0.111	0.231	0.159	0.368
FLOT [17]	Full	0.098	0.711	0.872	0.119	0.086	0.102	0.229	0.160	0.383
CMFlow [29]	Cross	0.059	0.833	0.920	0.070	0.054	0.062	0.157	0.227	0.455
ICP [36]	None	0.197	0.311	0.648	0.282	0.184	0.233	0.416	0.011	0.095
SceneFF [37]	None	0.254	0.247	0.515	0.286	0.204	0.245	0.460	0.059	0.102
JustGW [38]	Self	0.183	0.303	0.597	0.198	0.171	0.184	0.477	0.040	0.101
PointPWCNet [16]	Self	0.146	0.584	0.724	0.159	0.137	0.148	0.398	0.118	0.354
FlowStep3D [18]	Self	0.142	0.561	0.779	0.146	0.137	0.141	0.302	0.094	0.297
SLIM [21]	Self	0.170	0.411	0.693	0.201	0.146	0.173	0.355	0.048	0.140
RaFlow [10]	Self	0.095	0.719	0.856	0.121	0.093	0.107	0.223	0.169	0.386
Ours(w.o. Diffusion)	Self	0.091	0.704	0.844	0.117	0.088	0.102	0.239	0.166	0.391
Ours	Self	**0.084**	**0.793**	**0.908**	**0.115**	**0.074**	**0.094**	**0.208**	**0.192**	**0.420**

The underlined value indicates the overall best result, while the bold value denotes the best result within self-supervised methods. Full represents full supervision and Cross indicates cross-modal supervision.

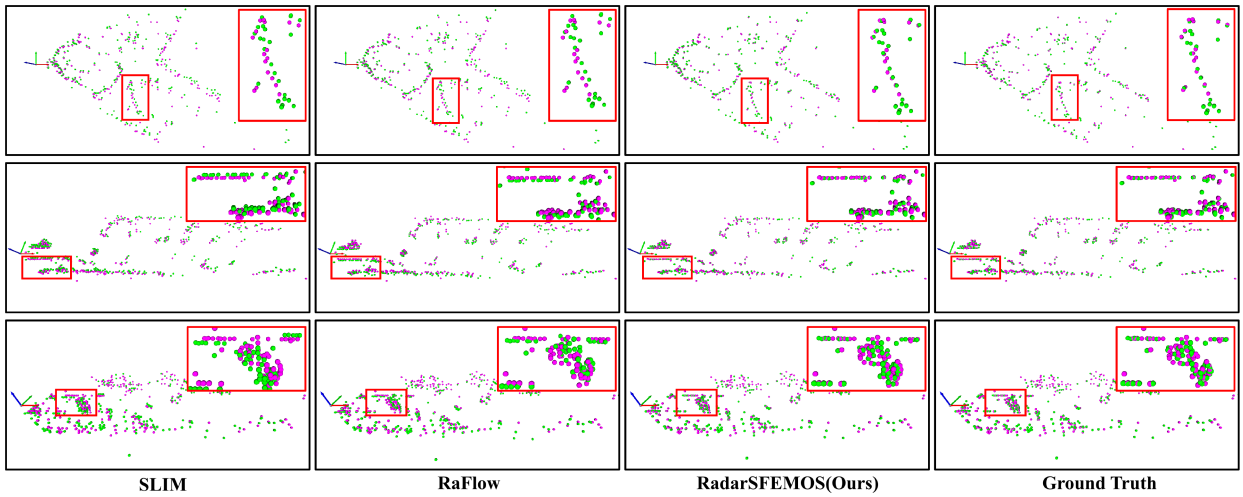

Fig. 3. Qualitative results of SFE. The pink points denote the points of $\mathcal{P}_t$. The green points represent the warped points of $\mathcal{P}_{t-1}$ by SFE result $\mathcal{F}$. Ideally, if the SFE is accurate, the green points should align well with the pink points.

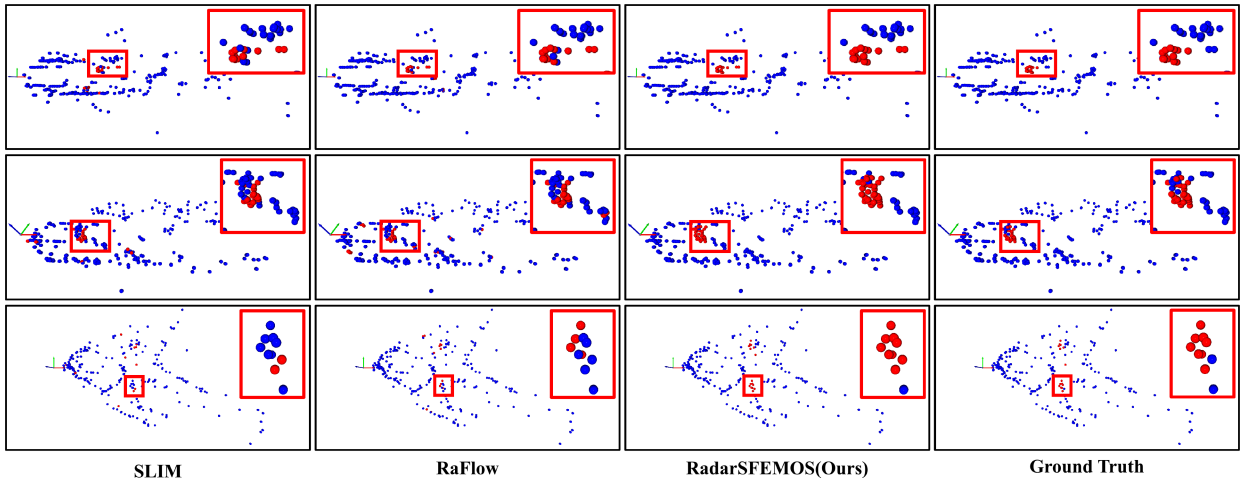

Fig. 4. Visualization of MOS compared with SLIM [21] and RaFlow [10]. Moving and static points of $\mathcal{P}_{t-1}$ are rendered as red and blue.

TABLE III
MOS PERFORMANCE COMPARISON

Methods	VoD		TJ4DRadSet	
	mIoU[%]↑	ACCM[%]↑	mIoU[%]↑	ACCM[%]↑
RadarMOSEVE†[6]	69.544	91.883	58.684	89.164
SLIM [21]	14.284	43.599	16.246	55.943
RaFlow [24]	46.652	84.796	54.287	87.355
Ours(w/o Diffusion)	49.452	85.933	56.745	88.614
Ours	**55.179**	**88.493**	**59.256**	**90.331**

†indicates fully-supervised while others are self-supervised methods.

TABLE V
ABLATION STUDIES OF DIFFERENT REFINEMENT ON VOD DATASET

	Feature Encoder	Corr Volume	Refinement	RNE↓	SAS↑	RAS↑
[F]	Radar Trans.	Point-Voxel	-	0.088	0.755	0.871
[G]	Radar Trans.	Point-Voxel	Conv	0.079	0.761	0.902
[H]	Radar Trans.	Point-Voxel	Diffusion Model	**0.074**	**0.816**	**0.930**

where TP, FP, and FN represent true positive, false positive and false negative predictions of moving points. We compare our RadarSFEMOS with multiple MOS baseline methods, including selfsupervised RaFlow [10] and SLIM [21] and supervised RadarMOSEVE [6]. To ensure fair comparisons, we adopted a consistent setup that uses two radar scans and excludes odometry information. As shown in Table III, our method significantly outperforms all self-supervised baseline methods across all evaluated metrics. Particularly in mIoU, our method achieves 55.179% and surpasses the SOTA method by 9.4%. Fig. 4 shows the qualitative results on the test set of different methods. Compared to other methods, our approach demonstrates superior capabilities in MOS and illustrates the effectiveness of our framework in simultaneously addressing radar-based SFE and MOS tasks.

D. Ablation Study

We conducted ablation studies on the VoD validation set to better understand the effectiveness of our design choices, shown in Tables IV and V. We compare three setups. The first setup shows the results of several common LiDAR-based point features encoder on radar data including ours. The second setup compares different correlation feature construction methods used in FlowNet3D [15], PointPWC-Net [16], FLOT [17], Difflow3D [8]. In both setups, we switch off the Diffusion-Based Scene Flow Refinement to show that the result differences stem from the designed modules.

Study on Diffusion Model: To evaluate the effectiveness of the diffusion model on 4D radar-based SFE task, we also report the performance after removing the diffusion model, while retaining the iterative estimator. Without the Diffusion-Based Scene Flow Refinement module, the quality of the SFE results is notably lower. However, by incorporating diffusion model into our method, we can further enhance SFE by leveraging the denoising capability of diffusion model. As in Table V, there is a 0.014 RNE decrease and 5.9% RAS increase. We also compare the Refinement module used in [19], which is composed of three convolutional layers and one fully connected layer. These results indicate the diffusion model's efficacy in reducing radar noise, yielding enhanced SFE accuracy and robustness.

TABLE VI
RUNTIME BREAKDOWN BY NETWORK MODULE

Module	Encoder	Corr Volume	Iterative Update	Refine	MOS Head
Time	9.8 ms	7.7 ms	69.2 ms	51.9 ms	5.2 ms

TABLE IV
ABLATION STUDIES OF COMPONENTS ON THE VOD DATASET

	Feature Encoder	Corr Volume	Refinement	RNE↓	SAS↑	RAS↑
[A]	PointNet++	Flow Embedding	-	0.139	0.466	0.748
[B]	PointNet++	Cost Volume	-	0.117	0.513	0.772
[C]	PointNet++	Point-Voxel	-	0.095	0.744	0.887
[D]	KpConv	Point-Voxel	-	0.094	0.737	0.886
[E]	Point Trans.	Point-Voxel	-	0.093	0.746	0.885
[F]	Radar Trans.	Point-Voxel	-	0.088	0.755	0.871

E. Runtime

We measure the runtime performance of our network using our Python implementation with a single NVIDIA RTX 3090 GPU. Our method achieves an average inference time of

143.8 ms for simultaneous SFE and MOS. A runtime breakdown by network modules, shown in Table VI, is provided for further understanding of our method's runtime performance.

V. Conclusion

In this letter, we propose a novel self-supervised joint framework for radar-based SFE and MOS without any additional reliance, greatly enhancing its robustness and practical utility. To address the inherent high sparsity and noise in radar raw data, we first design a novel transformer-based feature encoder with neighbor aggregation strategy, which effectively captures local salient features even at relatively greater distances. Furthermore, we incorporate the diffusion model into our framework to address the challenges of noise inherent in 4D radar, improving the SFE performance significantly. Extensive experiments on the VoD and TJ4DRadSet datasets demonstrate the superiority of our framework.

References

[1] Y. Cheng, J. Su, M. Jiang, and Y. Liu, "A novel radar point cloud generation method for robot environment perception," *IEEE Trans. Robot.*, vol. 38, no. 6, pp. 3754–3773, Dec. 2022.

[2] M. Zeller, J. Behley, M. Heidingsfeld, and C. Stachniss, "Gaussian radar transformer for semantic segmentation in noisy radar data," *IEEE Robot. Automat. Lett.*, vol. 8, no. 1, pp. 344–351, Jan. 2023.

[3] F. Ding, X. Wen, L. Zhu, Y. Li, and C. Lu, "RadarOcc: Robust 3D occupancy prediction with 4D imaging radar," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 37, May 2024, pp. 101589–101617.

[4] Z. Pan, F. Ding, H. Zhong, and C. Lu, "Ratrack: Moving object detection and tracking with 4D radar point cloud," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2024, pp. 4480–4487.

[5] P. Li, P. Wang, K. Berntorp, and H. Liu, "Exploiting temporal relations on radar perception for autonomous driving," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 17050–17059.

[6] C. Pang, X. Chen, Y. Liu, H. Lu, and Y. Cheng, "RadarMOSEVE: A spatial-temporal transformer network for radar-only moving object segmentation and ego-velocity estimation," in *Proc. Conf. Advancements Artif. Intell.*, 2024, pp. 4424–4432.

[7] J. Fu, Z. Xiang, C. Qiao, and T. Bai, "PT-FlowNet: Scene flow estimation on point clouds with point transformer," *IEEE Robot. Automat. Lett.*, vol. 8, no. 5, pp. 2566–2573, May 2023.

[8] J. Liu et al., "DifFlow3D: Toward robust uncertainty-aware scene flow estimation with diffusion model," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2024, pp. 15109–15119.

[9] Q. Zhang, Y. Yang, P. Li, O. Andersson, and P. Jensfelt, "SeFlow: A self-supervised scene flow method in autonomous driving," in *Proc. Eur. Conf. Comput. Vis.*, 2024, pp. 353–369.

[10] F. Ding, Z. Pan, Y. Deng, J. Deng, and C. X. Lu, "Self-supervised scene flow estimation with 4-D automotive radar," *IEEE Robot. Automat. Lett.*, vol. 7, no. 3, pp. 8233–8240, Jul. 2022.

[11] K. Harlow, H. Jang, T. D. Barfoot, A. Kim, and C. Heckman, "A new wave in robotics: Survey on recent mmWave radar applications in robotics," *IEEE Trans. Robot.*, vol. 40, pp. 4544–4560, 2024.

[12] B. Mersch, X. Chen, I. Vizzo, L. Nunes, J. Behley, and C. Stachniss, "Receding moving object segmentation in 3D LiDAR data using sparse 4D convolutions," *IEEE Robot. Automat. Lett.*, vol. 7, no. 3, pp. 7503–7510, Mar. 2022.

[13] Z. Gojcic, O. Litany, A. Wieser, L. J. Guibas, and T. Birdal, "Weakly supervised learning of rigid 3D scene flow," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 5688–5699.

[14] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 6840–6851.

[15] X. Liu, C. R. Qi, and L. J. Guibas, "FlowNet3D: Learning scene flow in 3D point clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 529–537.

[16] W. Wu, Z. Y. Wang, Z. Li, W. Liu, and L. Fuxin, "PointPWC-Net: Cost volume on point clouds for (self-) supervised scene flow estimation," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 88–107.

[17] G. Puy, A. Boulch, and R. Marlet, "FLOT: Scene flow on point clouds guided by optimal transport," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 527–544.

[18] Y. Kittenplon, Y. C. Eldar, and D. Raviv, "FlowStep3D: Model unrolling for self-supervised scene flow estimation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 4112–4121.

[19] Y. Wei, Z. Wang, Y. Rao, J. Lu, and J. Zhou, "PV-RAFT: Point-Voxel correlation fields for scene flow estimation of point clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Dec. 2021, pp. 6950–6959.

[20] W. Cheng and J. H. Ko, "Bi-PointFlowNet: Bidirectional learning for point cloud based scene flow estimation," in *Proc. Eur. Conf. Comput. Vis.*, 2022, pp. 108–124.

[21] S. A. Baur, D. J. Emmerichs, F. Moosmann, P. Pinggera, B. Ommer, and A. Geiger, "SLIM: Self-supervised LiDAR scene flow and motion segmentation," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 13106–13116.

[22] R. Li, C. Zhang, G. Lin, Z. Wang, and C. Shen, "RigidFlow: Self-supervised scene flow learning on point clouds by local rigidity prior," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2022, pp. 16938–16947.

[23] J. Kim, J. Woo, and S. Im, "RVMOS: Range-view moving object segmentation leveraged by semantic and motion features," *IEEE Robot. Automat. Lett.*, vol. 7, no. 3, pp. 8044–8051, Jul. 2022.

[24] X. Chen et al., "Moving object segmentation in 3D LiDAR data: A learning-based approach exploiting sequential data," *IEEE Robot. Automat. Lett.*, vol. 6, no. 4, pp. 6529–6536, Oct. 2021.

[25] J. Sun et al., "Efficient spatial-temporal information fusion for LiDAR-based 3D moving object segmentation," in *Proc. IEEE/RSJ Intl. Conf. Intell. Robots Syst.*, 2022, pp. 11456–11463.

[26] S. Mohapatra et al., "LiMoSeg: Real-time bird's eye view based LiDAR motion segmentation," 2021, *arXiv:2111.04875*.

[27] T. Kreutz, M. Mühlhäuser, and A. S. Guinea, "Unsupervised 4D LiDAR moving object segmentation in stationary settings with multivariate occupancy time series," in *Proc. IEEE Winter Conf. Appl. Comput. Vis.*, 2023, pp. 1644–1653.

[28] N. Wang, C. Shi, R. Guo, H. Lu, Z. Zheng, and X. Chen, "InsMOS: Instance-aware moving object segmentation in LiDAR data," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2023, pp. 7598–7605.

[29] F. Ding, A. Palffy, D. Gavrila, and C. Lu, "Hidden gems: 4D radar scene flow learning using cross-modal supervision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Mar. 2023, pp. 9340–9349.

[30] M. Zeller, V. S. Sandhu, B. Mersch, J. Behley, M. Heidingsfeld, and C. Stachniss, "Radar velocity transformer: Single-scan moving object segmentation in noisy radar point clouds," in *Proc. IEEE Intl. Conf. Robot. Automat.*, 2023, pp. 7054–7061.

[31] M. Zeller, V. S. Sandhu, B. Mersch, J. Behley, M. Heidingsfeld, and C. Stachniss, "Radar instance transformer: Reliable moving instance segmentation in sparse radar point clouds," *IEEE Trans. Robot.*, vol. 40, pp. 2357–2372, 2024.

[32] Z. Teed and J. Deng, "RAFT: Recurrent all-pairs field transforms for optical flow," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 402–419.

[33] R. Li, G. Lin, and L. Xie, "Self-point-flow: Self-supervised scene flow estimation from point clouds with optimal transport and random walk," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 15572–15581.

[34] X. Chen et al., "Joint scene flow estimation and moving object segmentation on rotational LiDAR data," *IEEE Trans. Intell. Transp. Syst.*, vol. 25, no. 11, pp. 17733–17743, Nov. 2024.

[35] W. Kabsch, "A solution for the best rotation to relate two sets of vectors," *Acta Crystallographica Sect. A*, vol. 32, pp. 922–923, 1976.

[36] P. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 2, pp. 239–256, Feb. 1992.

[37] J. K. Pontes, J. Hays, and S. Lucey, "Scene flow from point clouds with or without learning," in *Proc. Int. Conf. 3D Vis.*, 2020, pp. 261–270.

[38] H. Mittal, B. Okorn, and D. Held, "Just go with the flow: Self-supervised scene flow estimation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 11177–11185.

[39] A. Palffy, E. Pool, S. Baratam, J. F. P. Kooij, and D. M. Gavrila, "Multi-class road user detection with 3+1D radar in the view-of-delft dataset," *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 4961–4968, Apr. 2022.

[40] L. Zheng et al., "TJ4DRadSet: A 4D radar dataset for autonomous driving," in *Proc. IEEE 25th Int. Conf. Intell. Transp. Syst.*, 2022, pp. 493–498.

[41] D. P. Kingma and J. Ba, "ADAM: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations (ICLR)*, 2015.

[42] A. Paszke et al., "Pytorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 8026–8037.