TARS: Traffic-Aware Radar Scene Flow Estimation

Jialong Wu[1,3] Marco Braun[3] Dominic Spata[3] Matthias Rottmann[1,2]
[1]University of Wuppertal [2]Osnabrück University [3]Aptiv Services Deutschland GmbH

Abstract

Scene flow provides crucial motion information for autonomous driving. Recent LiDAR scene flow models utilize the rigid-motion assumption at the instance level, assuming objects are rigid bodies. However, these instance-level methods are not suitable for sparse radar point clouds. In this work, we present a novel Traffic-Aware Radar Scene-Flow (TARS) estimation method, which utilizes motion rigidity at the traffic level. To address the challenges in radar scene flow, we perform object detection and scene flow jointly and boost the latter. We incorporate the feature map from the object detector, trained with detection losses, to make radar scene flow aware of the environment and road users. From this, we construct a Traffic Vector Field (TVF) in the feature space to achieve holistic traffic-level scene understanding in our scene flow branch. When estimating the scene flow, we consider both point-level motion cues from point neighbors and traffic-level consistency of rigid motion within the space. TARS outperforms the state of the art on a proprietary dataset and the View-of-Delft dataset, improving the benchmarks by 23% and 15%, respectively.

1. Introduction

Scene flow estimates displacement vectors that describe point motion between two point cloud frames, which facilitates subsequent decision making in autonomous driving.

Early point cloud scene flow methods [22, 32] typically extract point-level or point-patch features and then estimate the flow by aggregating neighboring information from two frames. Considering only point-level information can result in points from the same object moving in different directions with varying magnitudes [21]. Recent LiDAR scene flow methods focus on the rigid-motion assumption [12]: objects move rigidly without deformation and a scene can be viewed as multiple rigidly moving objects and stationary parts. These LiDAR-based methods often begin with foreground segmentation, followed by clustering and matching to obtain instance pairs of the same object in two frames [14]. Then they predict the rigid motion at the instance level for each object pair [8] or derive the instance-wise transformation using optimization-based methods [13, 20].

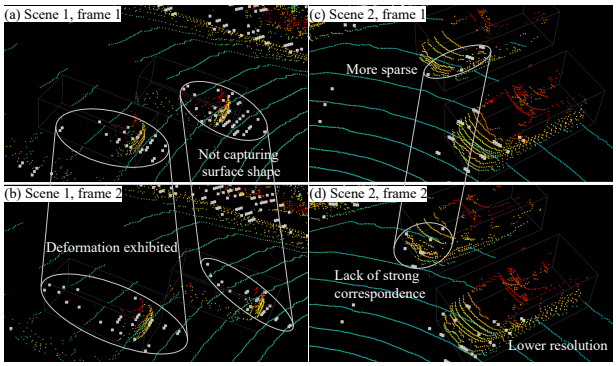

Figure 1. Challenges in radar scene flow. LiDAR points are shown in color, with corresponding radar points overlaid as larger gray points. (a-b) and (c-d) are two pairs of consecutive frames.

However, these instance-level methods are not suitable for radar scene flow due to the inherent sparsity of radar data (cf. Fig. 1). Radar is more robust under different weather conditions and is typically an order of magnitude less expensive than LiDAR [30]. Nevertheless, radar point clouds are considerably sparser than LiDAR ones and fail to capture object shapes. These challenges lead to a lack of reliable correspondences for matching instance pairs. Objects in radar point clouds may even exhibit deformations between frames due to the sparsity. Inferring rigid motion at the instance level may misinterpret such abnormal deformations as motion. Moreover, even nearby objects may have only a few reflection points (Fig. 1d), making optimization-based methods [20, 21] ineffective.

In this work, we focus on reconciling the contradiction between the rigid-motion assumption and the sparsity of radar point clouds. Although rigid motion of objects under radar is difficult to capture at the instance level, we believe this motion rigidity still exists within the space occupied by objects. Therefore, we aim for a higher-level scene understanding, beyond the instance level, focusing on the traffic level to capture rigid motion hidden in the traffic context. When estimating the scene flow, we not only consider the

motion cues propagated from neighboring points, but also the consistency of spatial context. This still adheres to the rigid-motion assumption. In fact, previous methods also assume that motion rigidity exists in Euclidean space when performing clustering and pairing of LiDAR points [12, 21]. However, to address the aforementioned challenges in radar scene flow, we do not further refine this assumption down to the instance level.

On the other hand, perceiving the environment and road users in the traffic is beneficial for motion prediction. Moreover, scene flow complements object detection with crucial motion information, enabling a more comprehensive perception. Therefore, we perform scene flow estimation and object detection jointly. We provide traffic cues to scene flow estimation through the object detector's feature map, which has been trained with detection losses and contains all relevant features about road users and the environment.

In our network, we achieve the traffic-level scene understanding by building a traffic vector field (TVF). We define the TVF as: a discrete grid map that incorporates traffic information about road users and the environment, with each cell containing a vector representing the motion. A conceptual diagram is shown in Fig. 5b. Note that we do not define an explicit 2D vector field; instead, we embed this concept within the feature space. The traffic information is extracted from the feature map of the object detection branch, and the motion information is passed through the hierarchical architecture of our scene flow branch. In each level of the architecture, we extract point-level motion cues from point neighbors while also paying attention to traffic-level motion consistency in Euclidean space. We use a coarse-grid TVF to achieve a high-level scene understanding, rather than falling into point-level details.

We evaluate our model TARS on a proprietary dataset and the View-of-Delft (VOD) dataset [23]. Quantitative results demonstrate that TARS exceeds the state-of-the-art (SOTA) scene-flow accuracy on these datasets by 23% and 15%, respectively. Radar scene flow heavily relies on radar sensors' ability to measure object velocities. However, radar can only measure radial velocity, leading to significant underestimation of tangential motion. Qualitative results show that TARS effectively captures objects' rigid motion while mitigating the tangential motion challenge.

Our main contributions are summarized as follows:

- We present TARS, the traffic-aware radar scene flow model that addresses the challenges of radar by leveraging the object detection feature map to obtain traffic context.
- We design the traffic vector field encoder and decoder modules to encode traffic-level motion understanding into the TVF and capture rigid motion in Euclidean space.
- TARS achieves SOTA accuracy by a large margin on both the proprietary dataset and VOD dataset.

2. Related Work

Point cloud scene flow. Scene flow for LiDAR point clouds has been widely studied over the past few years. Early methods [18, 22, 32] use PointNet [24] as the point or patch feature extractor and calculate flow embeddings based on neighborhood information. Bi-PointFlowNet [4] uses the bidirectional flow embeddings to capture the motion context between frames. PV-RAFT [29] extracts point and voxel features to capture local and long-range correspondences. HALFlow [27] applies a double attention mechanism to aggregate information from neighbors. DeFlow [33] employs a gated recurrent unit (GRU) to transfer voxel features to points and improves efficiency on large-scale point clouds. Flow4D [17] fuses multiple frames into 4D voxels and extracts spatio-temporal features. These methods infer scene flow only at the point-patch level, without considering higher-level motion consistency.

WsRSF [12] and PCAccumulation [14] utilize the aforementioned rigid-motion assumption at the instance level. They segment instance pairs and then regress the motion between each pair. Based on this approach, Dong et al. [8] introduce the nearest neighbor error minimization into a GRU to iteratively update the scene flow. Meanwhile, RigidFlow [20] uses the network output as the initialization for ICP (Iterative Closest Point) and refines the results. SCOOP [19] trains a pure correspondence model and computes scene flow via optimization with smoothness prior. Let-It-Flow [25] enforces rigidity in object clustering. MBNSF [26] encourages multi-body rigidity by adding a regularization term to the Chamfer loss. ICP-Flow [21] is a non-learning method using clustering to obtain instance pairs and a histogram-based approach to initialize ICP. Although effective for LiDAR, these instance-level rigidity methods cannot handle the challenges in radar scene flow (Fig. 1).

Radar scene flow. Radar point cloud scene flow has only gained attention in recent years. MilliFlow [7] estimates human motion. RaFlow [5] employs motion segmentation and applies the Kabsch algorithm [15] to estimate the ego-motion transformation as the static flow. It treats static points collectively, but lacks a higher-level understanding of dynamic points. CMFlow [6] leverages additional cross-modal information as supervision signals to enhance the performance of radar scene flow. In this work, we provide additional traffic information for radar scene flow and capture the rigid motion of objects at the traffic level.

Joint scene flow estimation & object detection. Combining these two tasks enables comprehensive perception in autonomous driving. Erçelik et al. [10] train a shared backbone with alternating task-specific heads, while PillarFlowNet [9] uses a voxel representation and a multi-task head to jointly estimate scene flow and detect objects. PointFlowNet [3] further infers point-wise motion from voxels. These fine-grained voxel-based methods are limited

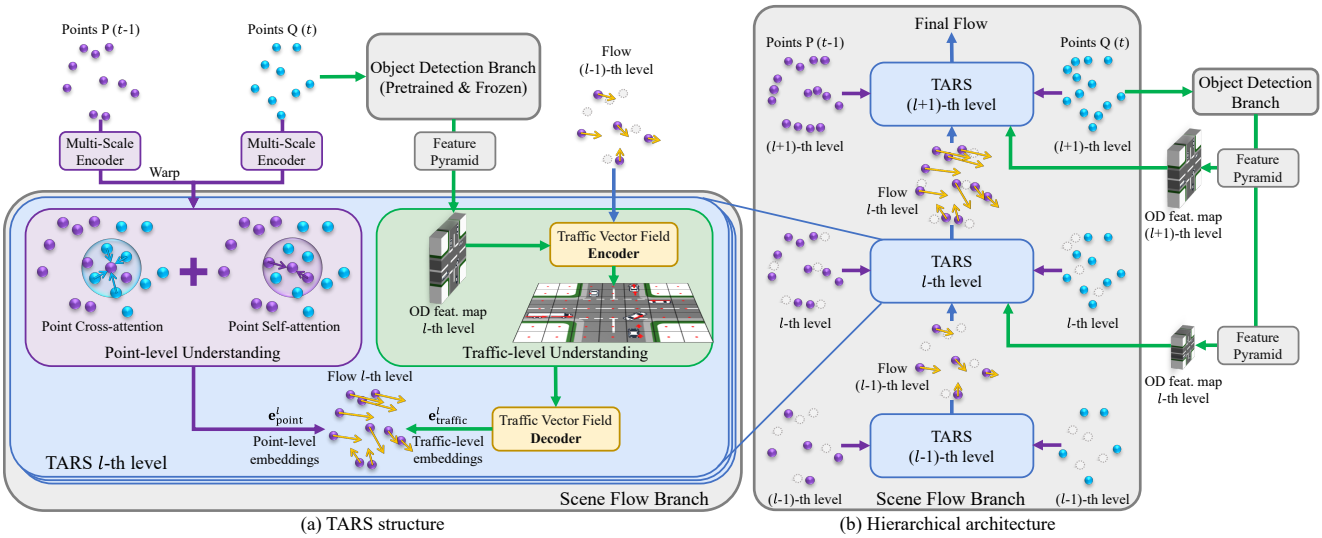

Figure 2. Overview of TARS. TARS employs a hierarchical architecture. At each level, it infers point-level motion cues using a double attention mechanism, while the TVF encoder leverages the OD feature map to build a traffic-level motion representation. The TVF decoder then extracts rigid motion cues in Euclidean space. Finally, dual-level flow embeddings are combined to estimate the scene flow.

to point-patch-level motion understanding, with task interaction primarily via sharing backbone features. In contrast, our work provides a holistic traffic-level understanding for scene flow through synergistic interactions in the feature space. TrackFlow [16] directly derives LiDAR scene flow from object detection and tracking results. However, radar object tracking is significantly less accurate. Our approach perceives traffic context by leveraging feature maps, which reduces the reliance on object detection accuracy.

3. TARS Architecture

We introduce TARS in a top-down approach. First, we present the hierarchical architecture that progressively refines scene flow (Sec. 3.1). Next, we use the l-th level to explain the structure of TARS and its dual-level motion understanding (Sec. 3.2). Thereafter, we dive into details of our TVF encoder, which encodes traffic and motion context into the TVF, achieving a traffic-level motion understanding (Sec. 3.3). The TVF decoder captures rigid motion hidden in the surrounding context, and the scene flow head combines point and traffic-level flow embeddings to predict the scene flow (Sec. 3.4). Finally, we briefly describe our recurrent module that leverages temporal cues (Sec. 3.5).

3.1. Hierarchical Architecture

Following the hierarchical architecture in prior works [27, 32], TARS has L levels and predicts scene flow in a coarse-to-fine fashion, progressively refining the prediction. Perceiving the environment and road users in the traffic is beneficial for motion prediction, as they provide a traffic-level motion prior. The object detection (OD) feature map, trained with detection losses, contains all relevant features about road users and the environment. Therefore, we jointly perform scene flow estimation and object detection (see Fig. 2b). We focus on enhancing the performance of radar scene flow, while the OD branch can be any detector, as long as it can generate bird's-eye view feature maps to provide traffic information for the scene flow branch.

The input to our scene flow branch consists of two point clouds $P \in \mathbb{R}^{N\times(3+2)}$ and $Q \in \mathbb{R}^{M\times(3+2)}$, with 5D initial features: x, y, z coordinates plus RRV (relative radial velocity) and RCS (radar cross-section) [34]. The multi-scale point encoder [24] is first applied to both point clouds for point feature extraction. Farthest point sampling is also performed to downsample the point clouds, yielding input point set pairs for each hierarchical level $\{P^l \in \mathbb{R}^{N_l\times(3+C)}, Q^l \in \mathbb{R}^{M_l\times(3+C)}\}_{l=1}^{L}$, where C is the dimension of extracted point feature, N_l and M_l are downsampled by a factor γ. Then, starting from the smallest point set, the (l-1)-th level of TARS computes flow embeddings $\mathbf{e}^{l-1} \in \mathbb{R}^{N_{l-1}\times D}$ and generates a coarse scene flow $F^{l-1} \in \mathbb{R}^{N_{l-1}\times 3}$, which are then used as input for the next level. After refining the flow using multiple TARS levels, we obtain the final scene flow for the full point set.

From the second-lowest level, we enhance scene flow estimation by incorporating traffic information from the OD branch (green arrows in Fig. 2b). Here, we feed the point cloud Q to the OD branch. The reason for using the feature map from Q rather than P is that: in our hierarchical architecture, the point cloud P is gradually warped toward the corresponding positions in Q by each level's flow prediction. Therefore, using Q's feature map allows for more accurate alignment between points and object features.

3.2. TARS Structure

In radar scene flow, capturing motion rigidity at the instance level is challenging (cf. Fig. 1). However, we believe that rigid motion still exists regions occupied by objects. Therefore, with the help of the OD branch, we aim to achieve a traffic-level scene understanding to reveal the rigid motion hidden in the traffic context. Meanwhile, point-level matching information remains crucial for motion estimation. By integrating both point-level and traffic-level insights, TARS achieves a comprehensive motion understanding and enhances radar scene flow estimation.

Taking the l-th level as an example, Figure 2a shows the structure of TARS. The point-level understanding extracts motion cues from neighboring points, while the traffic-level understanding is achieved by building a TVF (defined in Sec. 1) to capture the motion consistency.

3.2.1. Point-Level Motion Understanding

Point motion can be inferred from the matching information between neighboring points across consecutive point cloud frames [32]. Previous studies [22, 32] use a multi-layer perceptron (MLP) to encode this point-level matching information, known as the cost volume, into the flow embeddings. However, we observed that these MLP-based methods are unstable in sparse radar point clouds due to larger point spacing and fewer reflections per object. Therefore, we use a double attention mechanism [27] to adaptively extract the matching information from radar point clouds.

For a point $p_i^l \in \mathbb{R}^3$ in P^l, we compute cross-attention between p_i^l and its K nearest neighbors in Q^l (blue circle in Fig. 2a); then we apply self-attention between p_i^l and its neighboring points in P^l (purple circle in Fig. 2a). Unlike HALFlow [27], we remove the direction vector to mitigate the point spacing issue, and we employ heterogeneous keys and values to obtain fully attentive flow embeddings.

Specifically, we first warp the point cloud P^l closer to its neighbors in Q^l using the upsampled coarse flow from the previous level: $P^l_{\text{warp}} = P^l + \text{Interp}(F^{l-1})$. For simplicity, we omit this in the equations below. Let $\mathbf{p}_i^l, \mathbf{q}_j^l \in \mathbb{R}^C$ represent the point features of p_i^l, q_j^l. We compute the cross-attentive matching embeddings $\mathbf{e}_{\text{cross}}(p_i^l)$ for each point p_i^l:

$$\mathbf{e}_{\text{cross}}(p_i^l) = \text{Attention}(\mathbf{p}_i^l, \mathcal{N}_Q(p_i^l), \mathcal{N}_Q(p_i^l)), \quad (1)$$

$$\text{Attention}(\cdot, \diamond, \star) = \text{softmax}(\frac{\text{Q}(\cdot)\text{K}(\diamond)^T}{\sqrt{d_k}})\text{V}(\star), \quad (2)$$

where $\mathcal{N}_Q(p_i^l) = \text{KNN}(p_i^l, Q^l)$ denotes the K nearest neighbors of p_i^l in Q^l. $\text{Q}(\cdot), \text{K}(\cdot), \text{V}(\cdot)$ are linear layers.

Then, the point-level flow embeddings $\mathbf{e}_{\text{point}}(p_i^l)$ for a point p_i^l is computed via self-attention as:

$$\mathbf{e}_{\text{point}}(p_i^l) = \text{Attention}(\mathbf{e}_{\text{cross}}(p_i^l), \mathcal{N}_{\mathbf{e}}(p_i^l), \mathcal{N}_{\mathbf{e}}(p_i^l)), \quad (3)$$

where $\mathcal{N}_{\mathbf{e}}(p_i^l) = \text{KNN}(p_i^l, \mathbf{e}_{\text{cross}})$ fetches the matching embeddings for the neighbors of p_i^l in P^l.

3.2.2. Traffic-Level Scene Understanding

Our goal is to reconcile the rigid-motion assumption with the sparsity of radar point clouds. Instead of applying this assumption at the instance level [12, 21], we capture the rigid motion at the traffic level to address the challenges in radar. To achieve this, we design a TVF encoder that builds a traffic-level scene understanding. Then we employ a TVF decoder to capture rigid motion hidden in the traffic context.

Specifically, the feature map χ_{od}^l from the OD branch contains traffic information, and the flow embeddings $\mathbf{e}^{l-1}$ passed from the previous level carry motion information. The TVF encoder combines and encodes them into a coarse-grid TVF in the feature space, enabling traffic-level scene understanding (green & blue arrows in Fig. 2a). The TVF decoder employs cross-attention between points and TVF grids to perceive rigid motion and generate the traffic-level flow embeddings. In the TVF encoder, we apply global attention to build a holistic traffic-level understanding within the TVF. In contrast, the TVF decoder restricts the cross-attention to a local area, which helps to capture the rigid motion in spatial context. Finally, we combine point-level $\mathbf{e}_{\text{point}}$ and traffic-level flow embeddings $\mathbf{e}_{\text{traffic}}$ to enhance scene flow estimation (purple & green arrows in Fig. 2a).

3.3. Traffic Vector Field Encoder

Our TVF encoder integrates traffic and motion information into the TVF, and it progressively updates this traffic-level motion representation across TARS's hierarchical levels.

To this end, the TVF encoder employs two stages: (i) **Scene update**: A GRU [1] leverages the OD feature map to update the TVF from the previous level; (ii) **Flow painting**: Point-to-grid self-attention adaptively paints the coarse flow from the previous level onto the scene representation. Finally, we generate the new $\text{TVF}^l \in \mathbb{R}^{H \times W \times D_{\text{TVF}}}$ using global attention. Figure 3 illustrates these two stages. The TVF maintains the same shape across each level and is configured to a coarse grid, *e.g.*, $2\text{m} \times 2\text{m}$, enabling a high-level scene understanding without falling into point-level details.

Scene update. In this stage, we update the scene representation by leveraging traffic features from the l-th level of the OD feature pyramid to refine the previous level's TVF^{l-1}.

First, we apply CNN and pooling layers to the feature map χ_{od}^l to adapt the object features and match the predefined shape of the coarse TVF. Then we use χ_{od}^l as the input to a GRU, with TVF^{l-1} as the hidden state. The GRU is applied in an inter-level manner for updating the scene representation $\mathbf{X}_{\text{traffic}}^l$ across hierarchical levels:

$$\tilde{\mathbf{X}}_{\text{traffic}}^l = \tanh(\mathbf{W}_{\text{G}} * \chi_{\text{od}}^l + \mathbf{U}_{\text{G}} * (\mathbf{r}^l \odot \text{TVF}^{l-1})), \quad (4)$$

$$\mathbf{X}_{\text{traffic}}^l = \mathbf{z}^l \odot \text{TVF}^{l-1} + (1 - \mathbf{z}^l) \odot \tilde{\mathbf{X}}_{\text{traffic}}^l, \quad (5)$$

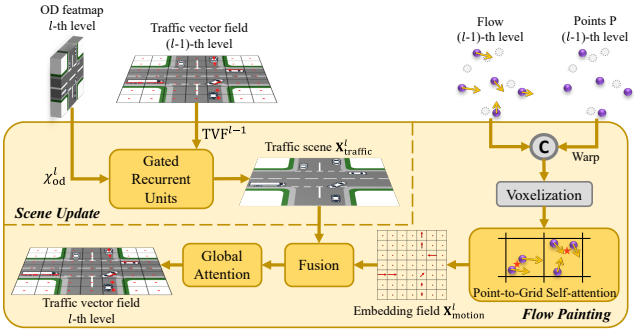

Figure 3. TVF encoder. Scene update: updates traffic information on the TVF using GRU; Flow painting: incorporates motion information into the TVF and build a holistic traffic representation.

where $*$ is the convolution operation, $\odot$ is the element-wise multiplication, $\mathbf{W}_{\text{G}}, \mathbf{U}_{\text{G}}$ are 2D convolution kernels, $\mathbf{r}^l, \mathbf{z}^l$ are the reset and update gates, see [1] for details.

Flow painting. In this stage, we project the coarse flow from the previous level onto the grid, using point-to-grid self-attention. Next, we fuse the motion representation with the scene representation and apply global attention to build a high-level scene understanding.

Specifically, we concatenate the (l-1)-th level's flow embeddings $\mathbf{e}^{l-1}$ with the point features $\mathbf{p}^{l-1}$ extracted by the multi-scale encoder, and then project them onto the predefined 2D grid. Because our TVF grid is coarse, each cell may contain multiple points with varying motion patterns (see Fig. 3). Therefore, we apply point-to-grid self-attention to adaptively extract motion features. We perform both channel-wise and point-wise self-attention within each grid cell and obtain the motion embedding field $\mathbf{X}^l_{\text{motion}}$.

Next, we fuse the traffic feature $\mathbf{X}^l_{\text{traffic}}$ and motion feature $\mathbf{X}^l_{\text{motion}}$ using spatial attention [31]. We concatenate the two feature maps and process them through CNN layers followed by a pixel-wise softmax to generate spatial attention weights. The fused traffic-level feature $\mathbf{X}^l_{\text{fusion}}$ is obtained as a weighted sum of $\mathbf{X}^l_{\text{traffic}}$ and $\mathbf{X}^l_{\text{motion}}$.

High-level scene understanding should not be limited to local areas. A global receptive field is crucial for modeling dependencies of rigid-body motions in traffic, *e.g.*, the motion patterns of vehicles in the same lane. We use axial attention [28] to provide the global vision and build the traffic-level motion understanding. It splits a standard attention block into separate row-wise and column-wise components, reducing complexity yet preserving global context. By stacking ω axial attention blocks, we enhance the fused features $\mathbf{X}^l_{\text{fusion}}$ and obtain the final TVF^l.

3.4. TVF Decoder & Scene Flow Head

Although radar point clouds are sparse, motion rigidity persists in Euclidean space. When building the traffic-level motion understanding, we encoded the rigid motion cues into the coarse-grid TVF. We then use the TVF decoder to perceive rigid motion hidden in the traffic context.

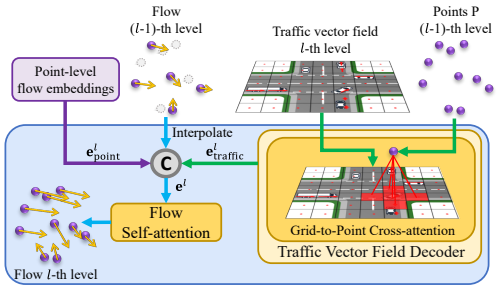

Figure 4. TVF decoder & scene flow head: capture motion rigidity in spatial context; combine dual-level embeddings for prediction.

Specifically, we apply grid-to-point cross-attention between each point p^l_i and its surrounding traffic context in the TVF, thereby integrating rigid motion cues into the flow embedding. To focus on relevant local rigid motion, the attentive receptive field is restricted to the nearby region around each point (see Fig. 4). We interpolate the coarse flow embeddings $\mathbf{e}^{l-1}$ and concatenate with the point feature $\mathbf{p}^l$ as the query, and the TVF grids as keys and values. This enables the resulting traffic-level flow embeddings $\mathbf{e}_{\text{traffic}}(p^l_i)$ to be aware of motion consistency in the traffic context:

$$\hat{\mathbf{p}}^l_i = \text{Concat}(\text{Interp}(\mathbf{e}^{l-1})_i, \mathbf{p}^l_i), \tag{6}$$

$$\mathbf{e}_{\text{traffic}}(p^l_i) = \text{Attention}(\hat{\mathbf{p}}^l_i, \mathcal{N}_{\text{TVF}}(p^l_i), \mathcal{N}_{\text{TVF}}(p^l_i)), \tag{7}$$

where $\mathcal{N}_{\text{TVF}}(p^l_i) = \text{KNN}(p^l_i, \text{TVF}^l)$ fetches the surrounding K cells of p^l_i from TVF^l.

Combining point-level and traffic-level motion understanding, we obtain the final flow embeddings $\mathbf{e}^l = \text{Concat}(\mathbf{e}_{\text{point}}, \mathbf{e}_{\text{traffic}}, \text{Interp}(\mathbf{e}^{l-1}))$. Finally, we apply another self-attention as in Eq. (3) on $\mathbf{e}^l$, reduce it back to C channels, and predict the final scene flow F^l.

3.5. Temporal Update Module

Recurrent layers can leverage long-range temporal information to enhance radar scene flow. CMFlow uses a GRU to retain flow embeddings across frames but experiences a slight drop in accuracy [6]. In contrast, we employ PointGRU layers [11] as the temporal module (distinct from TVF

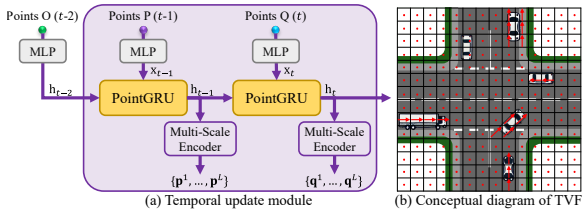

Figure 5. (a) Temporal update module: leverages low-level point dependencies using PointGRU. (b) Conceptual diagram of TVF.

encoder's inter-level GRU) to capture dependencies in low-level point features. We initialize the hidden state with the features of point cloud O at time $t-2$ and update it between the current point cloud pairs (see Fig. 5a). During training, we sample sequences of T frames as mini-clips.

4. Experiments

4.1. Experimental Setup

Dataset. We conducted experiments on the VOD dataset [23] and a proprietary dataset from Aptiv. Both datasets provide synchronized radar, LiDAR, camera and GPS/IMU odometry data. The VOD dataset contains primarily urban traffic scenes recorded with a low-resolution 4D radar, which captures about 256 radar points per frame and includes 4,662 training samples and 2,724 test samples. The proprietary dataset comprises urban, suburban and highway scenes, merging data from multiple high-resolution radars to obtain about 6K radar points per frame and containing 107,382 training samples and 24,198 test samples.

Metrics. On the VOD dataset, we adopt the evaluation metrics from CMFlow [6]: 1. *EPE*: mean end-point-error (L_2 distance) between the ground truth (GT) and the predicted scene flow. 2. *AccS/AccR*: Strict/Relaxed Accuracy, the percentage of points with EPE $< 0.05/0.1$ m or a relative error $< 5\%/10\%$. 3. *RNE*: resolution normalized EPE, to accommodate low-resolution radar. 4. *MRNE* and *SRNE*: RNE computed separately for moving and static points.

The proprietary dataset was collected using high-resolution radars. Therefore, we omit RNE. Since moving objects are critical in real-world autonomous driving, we focus on the accuracy of moving points. For moving points, we measure the following: 1. *MEPE*: EPE of moving points. 2. *MagE* and 3. *DirE*: magnitude and direction error between the GT and prediction. 4. *AccS/AccR* of moving points. For static points, we only calculate 5. *SEPE*; and we use 6. *AvgEPE*, the mean of MEPE and SEPE, as an overall metric. Metric details are provided in Appendix D.1.

4.2. Implementation Details

Model details. The hyperparameters for both datasets are listed in Tab. 1. Since the VOD dataset has only 256 points per frame, we do not perform downsampling. Our TVF uses a coarse grid to gain a high-level understanding rather than being confined to point-level details. On VOD, we use the Adam optimizer with a learning rate 10^{-3}, a decay rate of 0.9 per epoch, over 60 epochs. On the proprietary dataset, which is $20\times$ larger, we set the learning rate to 10^{-4} with a decay of 0.8 per 30K steps, training for 3 epochs. In the Appendix, we provide details of the OD branch (A.7, B.1), runtime analysis (C) and dual-task training strategy (A.6).

On the proprietary dataset, we simulate real-world autonomous driving by providing **all** models with ego-motion

Table 1. Params for two datasets. L, N, M, T: number of levels, points or mini-clips. γ: downsampling factor. C, D, D_{TVF}: feature channels of points, flow embeddings, or TVF. ω: number of axial attention blocks. $\mathcal{N}_Q, \mathcal{N}_{\text{TVF}}$: KNN points or TVF cells. TVF grid: shape $[H, W]$ and grid size. ego-info: availability of ego-motion.

Dataset	L	N, M	γ	C	D	D_{TVF}	ω	$\mathcal{N}_Q$	$\mathcal{N}_{\text{TVF}}$	T	TVF grid	ego-info
VOD	4	256	1	64	256	128	4	16	9	5	[40,40] 1.28m	Sup.
proprietary	4	6K	2	64	256	128	4	8	9	12	[70,40] 2.0m	Input

Table 2. Model variants and supervision signals.

TARS	EM (ego-motion Ω) availability			Supervision	
	train	test	Note	Label	Note
ego	✓	✗	as GT to train an EM head	Cross	w/ all losses
superego	✓	✓	as input to compensate EM	Cross$^+$	w/o $\mathcal{L}_{\text{seg}}, \mathcal{L}_{\text{ego}}, \mathcal{L}_{\text{opt}}$
no-ego	✗	✗	no EM operation	/	w/o $\mathcal{L}_{\text{seg}}, \mathcal{L}_{\text{ego}}, \mathcal{L}_{\text{opt}}$
FlowStep3D [18], RaFlow [5], etc.				Self	w/ only $\mathcal{L}_{\text{sc}}, \mathcal{L}_{\text{ss}}, \mathcal{L}_{\text{rd}}$

$\Omega \in \mathbb{R}^{4\times4}$ from the GPS/IMU sensor as known input. We apply ego-motion compensation to align P and Q into the same coordinate system. In this case, the GT for static points is the zero vector. On the VOD dataset, we test our model under two setups: (i) **TARS-ego:** following CMFlow [6], using the ego-motion transformation to train an additional ego-motion head for a fair comparison; (ii) **TARS–superego:** using ego-motion as known input and applying compensation, same as on the proprietary dataset. For details of the setups, please see Appendix A.2 and A.3.

Weakly-supervised training. Since annotating scene flow is extremely difficult, we adopt the self-supervised losses in [5] and cross-modal losses in [6], and introduce an additional background loss for static points. Detailed loss functions are given in Appendix D.2. On both datasets, we apply the following losses: 1. soft Chamfer loss $\mathcal{L}_{\text{sc}}$: aligns P_{warp} and Q by minimizing nearest-point distances, handling outliers via probabilistic matching; 2. spatial smoothness loss $\mathcal{L}_{\text{ss}}$: enforces neighboring points to have similar flow vectors, weighted by distance to ensure spatial smoothness; 3. radial displacement loss $\mathcal{L}_{\text{rd}}$: constrains the radial projection of predicted flow vectors using radar's RRV measurements; 4. foreground loss $\mathcal{L}_{\text{fg}}$: derives the pseudo scene flow GT from a LiDAR multi-object tracking model, applied to the predicted flow F^l_{fg} of foreground moving points at each level; 5. additionally, we employ a background loss $\mathcal{L}_{\text{bg}}$: using the ego-motion transformation as pseudo GT $\hat{F}^l_{\text{bg}}$ for static points. The overall loss $\mathcal{L}_{\text{all}}$ is formulated as:

$$\mathcal{L}_{\text{all}} = \mathcal{L}_{\text{sc}} + \mathcal{L}_{\text{ss}} + \mathcal{L}_{\text{rd}} + \sum_{l=1}^{L} (\mathcal{L}^l_{\text{fg}} + \lambda_{\text{bg}} \mathcal{L}^l_{\text{bg}}), \quad (8)$$

where $\mathcal{L}^l_{\text{fg}}$ and $\mathcal{L}^l_{\text{bg}}$ are computed for each level, $\lambda_{\text{bg}} = 0.5$.

When training TARS-ego on the VOD dataset, we incorporate cross-modal losses [6]: motion segmentation loss $\mathcal{L}_{\text{seg}}$: uses pseudo segmentation GT (derived from odometer and RRV measurements) to train a motion-segmentation head; ego-motion loss $\mathcal{L}_{\text{ego}}$: uses GT ego-motion to train an ego-motion head; and optical flow loss $\mathcal{L}_{\text{opt}}$: projects the

Table 3. Scene flow evaluation on the VOD dataset. Mean metric values across the test set are reported. "Sup." indicates the supervision signal, Self: training with only self-supervised losses [5], Cross: with additional cross-modal losses [6], Cross$^+$: setup for TARS-superego, without $\mathcal{L}_{\text{seg}}$, $\mathcal{L}_{\text{ego}}$, and $\mathcal{L}_{\text{opt}}$ while using ego-motion Ω as known input for ego-motion compensation, same as on the proprietary dataset.

		Overall				Moving	Static
Method	Sup.	EPE [m]↓	AccS [%]↑	AccR [%]↑	RNE [m]↓	MRNE [m]↓	SRNE [m]↓
PointPWC-Net [32]	Self	0.422	2.6	11.3	0.169	0.154	0.170
SLIM [2]	Self	0.323	5.0	17.0	0.130	0.151	0.126
FlowStep3D [18]	Self	0.292	3.4	16.1	0.117	0.130	0.115
Flow4D-2frame [17]	Cross	0.255	10.0	26.2	0.103	0.125	0.098
RaFlow [5]	Self	0.226	19.0	39.0	0.090	0.114	0.087
DeFlow [33]	Cross	0.217	11.8	31.6	0.087	0.098	0.085
CMFlow [6]	Cross	0.130	22.8	53.9	0.052	0.072	0.049
TARS-ego (ours)	Cross	**0.092** (-0.038)	**39.0** (+16.2)	**69.1** (+15.2)	**0.037** (-0.015)	**0.061** (-0.011)	**0.034** (-0.015)
TARS-superego (ours)	Cross$^+$	**0.048**	**76.6**	**86.4**	**0.019**	**0.057**	**0.014**

Table 4. Scene flow evaluation on the proprietary dataset. Mean metric values across the test set are reported. Ego-motion compensation and Cross$^+$ setup are applied to **all** models, making them "superego". PointGRU is applied in all models except for the one in the first row.

		Moving					Overall	Static
Method	PointGRU	MEPE [m]↓	MagE [m]↓	DirE [rad]↓	AccS [%]↑	AccR [%]↑	AvgEPE [m]↓	SEPE [m]↓
PointPWC-Net [32]	✗	0.453	0.363	1.218	44.2	52.2	0.244	0.036
PointPWC-Net [32]	✓	0.213	0.178	0.762	49.0	60.5	0.124	**0.035**
HALFlow [27]	✓	0.170	0.135	0.721	50.9	63.8	0.104	0.038
TARS (ours)	✓	**0.069** (-0.101)	**0.059** (-0.076)	**0.599** (-0.122)	**69.8** (+18.9)	**86.8** (+23.0)	**0.054** (-0.05)	0.038

scene flow onto the image plane and is trained with pseudo optical flow labels. Tab. 2 summarizes the variants of TARS and their supervision signals.

4.3. Comparison with State of the Art

Experiments on the VOD dataset. We compare our model TARS with SOTA scene flow methods on the VOD dataset (see Tab. 3). TARS clearly outperforms the previous SOTA model CMFlow [6] across all evaluation metrics in both setups. Under the same setup as CMFlow, our TARS-ego reduces the overall EPE from 0.13m to 0.092m, marking a 0.038m reduction and achieving a new milestone by bringing EPE below the AccR threshold of 0.1m. Moreover, TARS-ego improves AccS and AccR, two accuracy metrics computed based on EPE, by 16.2% and 15.2%, respectively. RNE is computed as $RNE = \frac{EPE}{r_R / r_L}$, where $\frac{r_R}{r_L}$ is the ratio of radar to LiDAR resolution (on average 2.5). Therefore, reductions in RNE metrics are numerically smaller. Nevertheless, TARS-ego shows substantial improvements in all three RNE-related metrics. We also compare TARS-ego with latest LiDAR models [17, 33]. These models perform worse because their fully-voxel representation, designed for large-scale LiDAR point clouds, is unsuitable for radar scene flow. Ego-motion from the odometer is a simple $\mathbb{R}^{4\times4}$ matrix yet effective booster. Assuming real-world autonomous driving with an available odometer, TARS-superego applies ego-motion compensation and reduces EPE down to 0.048m, MRNE to 0.057m, and boosts AccS and AccR to 76.6% and 86.4%, respectively.

Experiments on the proprietary dataset. The proprietary dataset includes more complex and high-speed scenarios such as highways and suburban scenes, which makes radar scene flow estimation particularly challenging. Therefore, to simulate real-world autonomous driving, we apply ego-motion compensation to **all** models and focus on moving points during evaluation (see Tab. 4). Furthermore, we integrate our temporal update module (Sec. 3.5) into other models for a fair comparison. Our model TARS outperforms the previous SOTA model HALFlow [27] by a large margin, reducing MEPE from 0.17m to 0.069m and improving AccS and AccR by 18.9% and 23%, respectively. Qualitative results in Fig. 6 show that TARS effectively captures the rigid motion of radar points on the same object and mitigates the tangential motion challenge mentioned in Sec. 1.

Table 5. Ablation study on the proprietary dataset. Point Level: point-level motion understanding. OD Featmap: using OD feature map. Coarse Grid: using coarse-grid TVF. Glob Attn: global attention applied. The chosen setup of TARS is highlighted in blue.

No.	Point Level	OD Featmap	Coarse Grid	Glob Attn	Moving MEPE↓	Moving AccS↑	Moving AccR↑	Overall AvgEPE↓	Static SEPE↓
1	✓				0.178	47.9	61.6	0.106	0.033
2	✓	✗	✓	✓	0.144	45.0	63.3	0.093	0.041
3	✓	✓	✗	✓	0.104	51.4	69.9	0.076	0.049
4	✓	✓	✓	✗	0.074	65.6	84.2	**0.053**	**0.031**
5	✓	✓	✓	✓	**0.069**	**69.8**	**86.8**	0.054	0.038
6	Decoder $\mathcal{N}_{\text{TVF}} = 5$				0.071	67.2	86.5	0.064	0.057
7	Decoder $\mathcal{N}_{\text{TVF}} = 9$				**0.069**	**69.8**	**86.8**	**0.054**	**0.038**
8	Decoder $\mathcal{N}_{\text{TVF}} = 13$				0.077	59.9	84.0	0.060	0.043

4.4. Ablation Studies

Importance of key components. We demonstrate the effectiveness of the key components of TARS on the proprietary dataset. Ablation study *No. i* refers to the i-th row of Tab. 5. *No. 1*: We completely remove the traffic level, causing the model to revert to inferring motion cues solely at the point level, resulting in MEPE of 0.178m, close to HALFlow (Tab. 4). *No. 2*: We activate the traffic level but without the OD feature map, meaning no scene update is performed in the TVF encoder. This reduces our model to simply combining point- and voxel-based motion features,

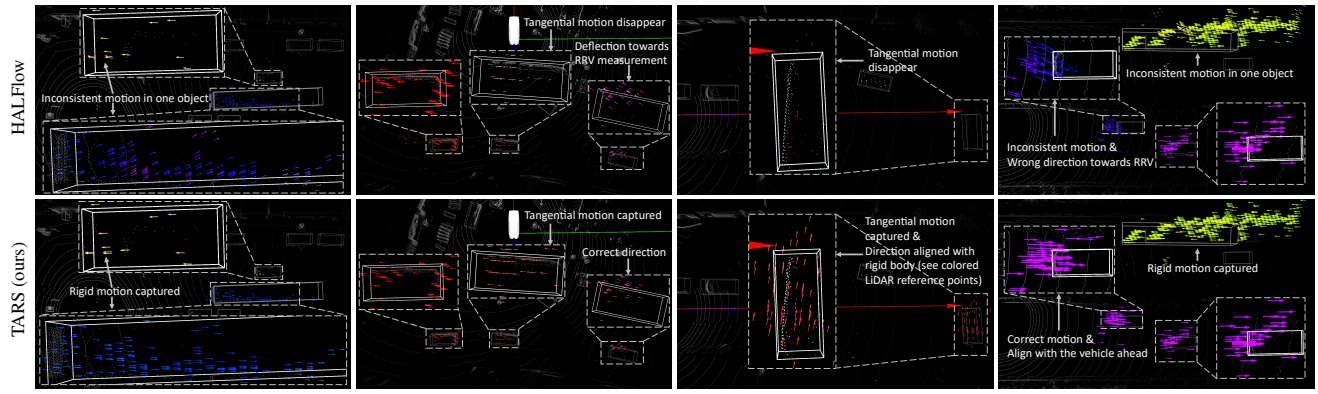

Figure 6. Qualitative results on the proprietary dataset, compared with HALFlow [27]. LiDAR point clouds serve as reference. Arrows indicate the predicted scene flow. After ego-motion compensation, static points are expected to yield zero vectors if predicted correctly. By perceiving traffic-level motion cues, TARS effectively captures the rigid motion in Euclidean space, as well as tangential movements.

leading to unsatisfactory AccR of 63.3%. *No. 3*: We enable the OD feature map and scene update, but change the coarse grid of TVF to a fine grid, from $2\text{m} \times 2\text{m}$ to $1\text{m} \times 1\text{m}$ with shape $[140, 80]$. In this case, our traffic-level understanding falls into point-level details, resulting in a reduction of MEPE by 0.04m, yet the improvement in accuracy is limited (AccR 69.9%). *No. 4*: We apply a coarse TVF to achieve high-level understanding, which further reduces MEPE by 0.03m and boosts the accuracy; but we replace the global attention in the TVF encoder with local convolutions. This makes the TVF focus on local areas, achieving the lowest SEPE of 0.031m, while limiting its ability to capture global traffic information (e.g., motion of vehicles in the same lane). *No. 5*: We construct a holistic traffic model using global attention. Compared to No. 4, this improves AccS by 4.2% and AccR by 2.6%. Although SEPE increased, all models maintain a low SEPE (below 0.05m) due to ego-motion compensation, and we prioritize dynamic objects in autonomous driving. *No. 6-8* show the effect of $\mathcal{N}_{\text{TVF}}$: the number of TVF cells that a point query attends to, when capturing motion context. In the cross-attention, considering more spatial context $\mathcal{N}_{\text{TVF}} = 9$ (surrounding neighbors) improves AccS by 2.6% and reduces SEPE by 0.019m, compared to $\mathcal{N}_{\text{TVF}} = 5$ (only direct neighbors). However, expanding to $\mathcal{N}_{\text{TVF}} = 13$ (second-order neighbors) increases SEPE and significantly reduces AccS due to the inclusion of irrelevant parts in the attention. Moreover, on PointPWC-Net in Tab. 4, we demonstrate the importance of applying PointGRU to utilize temporal information.

Effect of losses. We evaluate the impact of losses on the VOD dataset (Tab. 6). Group *No. 1*: we test the impact of the proposed background loss $\mathcal{L}_{\text{bg}}$ using a TARS-no-ego model, which excludes the ego-motion head of TARS-ego as well as three losses $\mathcal{L}_{\text{seg}}$, $\mathcal{L}_{\text{ego}}$, and $\mathcal{L}_{\text{opt}}$. The experiments show that setting the background weight $\lambda_{\text{bg}} = 0.5$ results in the lowest MRNE of moving points. Further increasing λ_{bg} could inflate AccS&AccR as they reflect overall accuracy on the VOD dataset, where static points dominate the scene. However, it undermines the MRNE. We advocate emphasizing dynamic objects in radar scene flow, which is why we set $\lambda_{\text{bg}} = 0.5$. Group *No. 2*: we include the ego-motion head but initially without $\mathcal{L}_{\text{bg}}$, which yields AccR of 62.4%. After enabling $\mathcal{L}_{\text{bg}}$, both moving and static points get improved. Group *No. 3*: we test the impact of $\mathcal{N}_{\text{TVF}}$. However, it did not yield significant differences among the three setups, because each frame in the VOD dataset contains only 256 points, resulting in a highly sparse TVF.

Table 6. Ablation study on the VOD dataset. TARS-no-ego: without ego-motion head and supervision signals $\mathcal{L}_{\{\text{seg, ego, opt}\}}$. The chosen setup of TARS-ego is highlighted in blue.

No.	TARS	$\{\mathcal{L}_{\text{seg}}, \mathcal{L}_{\text{ego}}, \mathcal{L}_{\text{opt}}\}$	$\mathcal{L}_{\text{bg}}$ or λ_{bg}	Overall				Moving	Static
				EPE↓	AccS↑	AccR↑	RNE↓	MRNE↓	SRNE↓
1	no-ego	✗	0.25	0.124	23.6	54.7	0.050	0.066	0.048
	no-ego	✗	0.50	0.111	28.5	59.3	0.045	**0.065**	0.043
	no-ego	✗	0.75	0.103	32.5	63.2	0.042	0.066	0.039
	no-ego	✗	1.00	**0.098**	**34.3**	**65.8**	**0.040**	0.067	**0.036**
2	ego	✓	✗	0.107	32.5	62.4	0.043	0.062	0.040
	ego	✓	0.50	**0.092**	**39.0**	**69.1**	**0.037**	**0.061**	**0.034**
3	ego	Decoder $\mathcal{N}_{\text{TVF}} = 5$		0.094	38.8	68.5	0.038	0.062	0.034
	ego	Decoder $\mathcal{N}_{\text{TVF}} = 9$		**0.092**	**39.0**	**69.1**	**0.037**	**0.061**	0.034
	ego	Decoder $\mathcal{N}_{\text{TVF}} = 13$		0.093	38.1	68.7	0.037	0.062	0.034

5. Conclusion

We introduced TARS, a traffic-aware radar scene flow model. Leveraging traffic information from an object detector, TARS employs the traffic vector field encoder to build a holistic traffic-level scene understanding and uses the TVF decoder to perceive motion rigidity in the traffic context. Quantitative results show that TARS significantly outperforms SOTA on both the proprietary dataset and VOD dataset. Ablation studies highlight the effectiveness of key components in our design, such as incorporating OD feature maps, using a coarse-grid TVF, and applying global attention. Qualitative results demonstrate TARS's ability to capture rigid motion and tangential movements.

Acknowledgements

J.W. and M.R. acknowledge support by the German Federal Ministry of Education and Research within the junior research group project “UnrEAL” (grant no. 01IS22069).

References

[1] Nicolas Ballas, Li Yao, Chris Pal, and Aaron Courville. Delving deeper into convolutional networks for learning video representations. *arXiv preprint arXiv:1511.06432*, 2015. 4, 5, 2

[2] Stefan Andreas Baur, David Josef Emmerichs, Frank Moosmann, Peter Pinggera, Björn Ommer, and Andreas Geiger. Slim: Self-supervised lidar scene flow and motion segmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 13126–13136, 2021. 7

[3] Aseem Behl, Despoina Paschalidou, Simon Donné, and Andreas Geiger. Pointflownet: Learning representations for rigid motion estimation from point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7962–7971, 2019. 2

[4] Wencan Cheng and Jong Hwan Ko. Bi-pointflownet: Bidirectional learning for point cloud based scene flow estimation. In *European Conference on Computer Vision*, pages 108–124. Springer, 2022. 2, 4

[5] Fangqiang Ding, Zhijun Pan, Yimin Deng, Jianning Deng, and Chris Xiaoxuan Lu. Self-supervised scene flow estimation with 4-d automotive radar. *IEEE Robotics and Automation Letters*, 7(3):8233–8240, 2022. 2, 6, 7, 1

[6] Fangqiang Ding, Andras Palffy, Dariu M Gavrila, and Chris Xiaoxuan Lu. Hidden gems: 4d radar scene flow learning using cross-modal supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9340–9349, 2023. 2, 5, 6, 7, 1, 3, 4

[7] Fangqiang Ding, Zhen Luo, Peijun Zhao, and Chris Xiaoxuan Lu. milliflow: Scene flow estimation on mmwave radar point cloud for human motion sensing. In *European Conference on Computer Vision*, pages 202–221. Springer, 2024. 2

[8] Guanting Dong, Yueyi Zhang, Hanlin Li, Xiaoyan Sun, and Zhiwei Xiong. Exploiting rigidity constraints for lidar scene flow estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12776–12785, 2022. 1, 2

[9] Fabian Duffhauss and Stefan A Baur. Pillarflownet: A real-time deep multitask network for lidar-based 3d object detection and scene flow estimation. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10734–10741. IEEE, 2020. 2, 3

[10] Emeç Erçelik, Ekim Yurtsever, Mingyu Liu, Zhijie Yang, Hanzhen Zhang, Pınar Topçam, Maximilian Listl, Yılmaz Kaan Caylı, and Alois Knoll. 3d object detection with a self-supervised lidar scene flow backbone. In *European Conference on Computer Vision*, pages 247–265. Springer, 2022. 2

[11] Hehe Fan and Yi Yang. Pointrnn: Point recurrent neural network for moving point cloud processing. *arXiv preprint arXiv:1910.08287*, 2019. 5, 2

[12] Zan Gojcic, Or Litany, Andreas Wieser, Leonidas J Guibas, and Tolga Birdal. Weakly supervised learning of rigid 3d scene flow. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5692–5703, 2021. 1, 2, 4

[13] Xiaodong Gu, Chengzhou Tang, Weihao Yuan, Zuozhuo Dai, Siyu Zhu, and Ping Tan. Rcp: Recurrent closest point for point cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8216–8226, 2022. 1

[14] Shengyu Huang, Zan Gojcic, Jiahui Huang, Andreas Wieser, and Konrad Schindler. Dynamic 3d scene analysis by point cloud accumulation. In *European Conference on Computer Vision*, pages 674–690. Springer, 2022. 1, 2

[15] Wolfgang Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 32(5):922–923, 1976. 2, 1

[16] Ishan Khatri, Kyle Vedder, Neehar Peri, Deva Ramanan, and James Hays. I can’t believe it’s not scene flow! In *European Conference on Computer Vision*, pages 242–257. Springer, 2025. 3, 4, 7

[17] Jaeyeul Kim, Jungwan Woo, Ukcheol Shin, Jean Oh, and Sunghoon Im. Flow4d: Leveraging 4d voxel network for lidar scene flow estimation. *IEEE Robotics and Automation Letters*, 2025. 2, 7, 3

[18] Yair Kittenplon, Yonina C Eldar, and Dan Raviv. Flowstep3d: Model unrolling for self-supervised scene flow estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4114–4123, 2021. 2, 6, 7, 4

[19] Itai Lang, Dror Aiger, Forrester Cole, Shai Avidan, and Michael Rubinstein. Scoop: Self-supervised correspondence and optimization-based scene flow. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5281–5290, 2023. 2

[20] Ruibo Li, Chi Zhang, Guosheng Lin, Zhe Wang, and Chunhua Shen. Rigidflow: Self-supervised scene flow learning on point clouds by local rigidity prior. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16959–16968, 2022. 1, 2

[21] Yancong Lin and Holger Caesar. Icp-flow: Lidar scene flow estimation with icp. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15501–15511, 2024. 1, 2, 4

[22] Xingyu Liu, Charles R Qi, and Leonidas J Guibas. Flownet3d: Learning scene flow in 3d point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 529–537, 2019. 1, 2, 4

[23] Andras Palffy, Ewoud Pool, Srimannarayana Baratam, Julian FP Kooij, and Dariu M Gavrila. Multi-class road user detection with 3+ 1d radar in the view-of-delft dataset. *IEEE Robotics and Automation Letters*, 7(2):4961–4968, 2022. 2, 6

[24] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on

point sets in a metric space. *Advances in neural information processing systems*, 30, 2017. 2, 3

[25] Patrik Vacek, David Hurych, Karel Zimmermann, and Tomáš Svoboda. Let-it-flow: Simultaneous optimization of 3d flow and object clustering. *IEEE Transactions on Intelligent Vehicles*, 2024. 2

[26] Kavisha Vidanapathirana, Shin-Fang Chng, Xueqian Li, and Simon Lucey. Multi-body neural scene flow. In *2024 International Conference on 3D Vision (3DV)*, pages 126–136. IEEE, 2024. 2

[27] Guangming Wang, Xinrui Wu, Zhe Liu, and Hesheng Wang. Hierarchical attention learning of scene flow in 3d point clouds. *IEEE Transactions on Image Processing*, 30:5168–5181, 2021. 2, 3, 4, 7, 8, 5

[28] Huiyu Wang, Yukun Zhu, Bradley Green, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Axial-deeplab: Stand-alone axial-attention for panoptic segmentation. In *European conference on computer vision*, pages 108–126. Springer, 2020. 5

[29] Yi Wei, Ziyi Wang, Yongming Rao, Jiwen Lu, and Jie Zhou. Pv-raft: Point-voxel correlation fields for scene flow estimation of point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6954–6963, 2021. 2, 4

[30] Zhiqing Wei, Fengkai Zhang, Shuo Chang, Yangyang Liu, Huici Wu, and Zhiyong Feng. Mmwave radar and vision fusion for object detection in autonomous driving: A review. *Sensors*, 22(7):2542, 2022. 1

[31] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018. 5, 1

[32] Wenxuan Wu, Zhi Yuan Wang, Zhuwen Li, Wei Liu, and Li Fuxin. Pointpwc-net: Cost volume on point clouds for (self-) supervised scene flow estimation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16*, pages 88–107. Springer, 2020. 1, 2, 3, 4, 7

[33] Qingwen Zhang, Yi Yang, Heng Fang, Ruoyu Geng, and Patric Jensfelt. DeFlow: Decoder of scene flow network in autonomous driving. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2105–2111, 2024. 2, 7, 3, 4

[34] Yi Zhou, Lulu Liu, Haocheng Zhao, Miguel López-Benítez, Limin Yu, and Yutao Yue. Towards deep radar perception for autonomous driving: Datasets, methods, and challenges. *Sensors*, 22(11):4208, 2022. 3