RVLoss: Runoff Vote Loss for Self-Supervised LiDAR Scene Flow Estimation

Shiming Wang[1] **Liangliang Nan**[1] **Julian F.P. Kooij**[1] **Holger Caesar**[1] **Yancong Lin**[2,*]
[1]TU Delft, the Netherlands [2]University of Nottingham, UK

Abstract

LiDAR scene flow estimates point-wise motion between two consecutive scans, referred to as the source and target. Leading self-supervised methods typically minimize the Chamfer loss, the nearest-neighbor distance between the flow-compensated source and the target. However, nearest-neighbor search does not enforce motion rigidity, often leading to inconsistent flows within object instances. Existing approaches address this issue with additional regularization terms, but flow consistency among points remains limited, especially for large objects. We propose **RVLoss**, a self-supervised loss that incorporates motion rigidity by design through a *runoff vote* mechanism. Our key observation is that the point-wise motion, calculated from nearest-neighbor search, can often be grouped into a small set of dominant flow candidates by voting (top-k voting). Furthermore, when compensating the source by these candidates, the flow that best represents the underlying rigid motion often yields the highest consensus after a second voting (top-1 voting). Based on this insight, we incorporate the two-stage *runoff vote* into loss design and create cluster-wise rigid flows and free-form flows as pseudo-labels for self-supervised learning. RVLoss can be seamlessly integrated into existing feedforward architectures. Experiments on the Argoverse2 2026 Scene Flow Challenge show that models trained with RVLoss achieve state-of-the-art performance among self-supervised approaches, outperforming baseline models trained with alternative loss designs by 20% and reducing the mean dynamic normalized end-point error from 0.202 to 0.163. Moreover, cross-dataset evaluations demonstrate consistent performance improvements across four additional datasets. Code is available at `https://github.com/tudelft-iv/RVLoss`.

1 Introduction

LiDAR scene flow is the per-point 3D motion between two consecutive point clouds, and estimating scene flow is essential for many downstream tasks in autonomous driving, such as object discovery [1, 2], dynamic object reconstruction [3, 4], and BEV feature alignment [5]. Self-supervised methods have recently gained increasing attention, as they reduce reliance on costly manual annotations. Progress has been substantial, as demonstrated by the annual Argoverse2 Scene Flow Challenge [6, 7], where leading methods achieve both high accuracy and real-time inference. The majority of leading works follow a feedforward design and use the Chamfer distance as the loss function, where the goal is to minimize the spatial distance between a point and its nearest neighbor across time after flow compensation [8–10]. However, nearest-neighbor search breaks motion rigidity (a common property for traffic participants in autonomous driving scenarios), as shown in Fig. 1. As a result, there are often inconsistent flows, particularly for large objects such as cars, buses, and trucks.

A common strategy for enhancing motion rigidity is to introduce additional regularization terms. For instance, Zhang et al. [3, 8] apply regularization to static, dynamic, and clustered points, where point

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

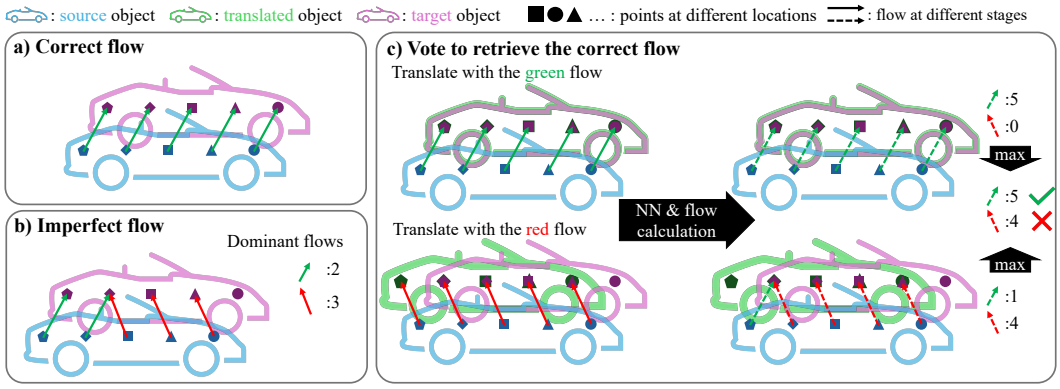


Figure 1: **Overview of a runoff vote.** a) A car moves rigidly and thus exhibits consistent flows among points. b) The widely used Chamfer distance breaks motion rigidity due to the local nearest-neighbor search. As a result, multiple flow candidates emerge within the object. Although the correct flow is among these candidates, it may not be the top-voted one (1^{st} stage). c) We translate the source by those flow candidates and measure how well it aligns with the target by an additional voting (2^{nd} stage). The flow receiving the highest number of votes is selected as the rigid flow, representing the underlying motion of the car.

labels are precomputed using off-the-shelf algorithms. Beyond explicit regularization, Lin et al. [9] propose a differentiable module that enforces motion rigidity from an architectural perspective by aggregating features over predefined local neighborhoods to capture collective motion cues. While these approaches improve performance, the resulting flow predictions still exhibit inconsistent motion within rigid objects.

In this paper, we investigate how to incorporate motion rigidity directly into the loss design, eliminating the need for delicate architectural modification or multiple regularization terms. Our key observation is that point-wise motions of rigidly moving points, calculated from nearest-neighbor search, can often be grouped into a small set of dominant flows (top-k), and crucially, the ideal flow (top-1) that best characterizes the underlying rigid motion typically resides among these candidates, as shown in Fig. 1 b). Intuitively, the top-k candidates provide a limited number of viable solutions, which can be further evaluated one by one by shifting the source point cloud toward the target and by checking for consistency among clustered points, as illustrated in Fig. 1 c). Both the selection of top-k candidates and the identification of the top-1 can be implemented via *voting*, i.e., aggregating the total number of points that share the same motion. Based on this insight, we propose **RVLoss**, a self-supervised loss function built on two-stage cascaded voting for scene flow estimation, also known as a *runoff vote*. RVLoss generates cluster-wise rigid flows and free-form flows, which are treated as pseudo-labels for self-supervised learning.

RVLoss differs from directly selecting the top-1 dominant flow via a single-stage voting, which often underestimates the actual motion. For large objects (e.g., trucks) that exhibit significant spatial overlap after motion, nearest-neighbor search tends to favor correspondences with minimal displacement. Such matches, however, fail to capture the true object motion, especially on geometrically featureless surfaces. Selecting the top-k candidates instead accounts for multiple plausible displacements shared across points, yielding a compact set of hypotheses and enabling efficient downstream voting.

Our experiments on the Argoverse2 2026 Scene Flow Challenge[1] demonstrate consistent performance improvement of RVLoss over other loss designs. Cross-dataset evaluation confirms the strong generalization ability of models supervised by RVLoss, yielding top performance on four additional datasets. To summarize, our contributions are as follows:

- We propose **RVLoss** for self-supervised scene flow learning based on a *Runoff Vote* process. As its name suggests, RVLoss uses two-stage cascaded voting to identify cluster-wise rigid motions, and uses these as pseudo-labels for self-supervised learning. This way, motion rigidity is encoded directly by the pseudo-labels, rather than as a learning objective of a separate loss term.

[1] https://www.argoverse.org/sceneflow.html

- RVLoss is architecture-agnostic and can be seamlessly integrated with mainstream scene flow backbones. Models supervised by RVLoss achieve leading performance on the Argoverse2 2026 Scene Flow Challenge [11] in both in-domain test and cross-domain evaluation across four other datasets, in particular on rigid moving object categories.

2 Related Work

Our work builds on four lines of research: efficient backbone design, self-supervised loss design, rigid motion modeling, and voting-based point cloud processing.

Backbone design for LiDAR scene flow estimation. LiDAR scene flow methods commonly adapt feedforward architectures to predict per-point motion [12–20]. However, early models mainly operate on downsampled point clouds, typically up to 8k points, and therefore struggle with full-scale LiDAR scans [17, 21–23]. FastFlow3D [24] addresses this limitation as the first feedforward model for real-time large-scale scene flow, and subsequent methods build on this design for further improvement [3, 9, 25–27]. Recent works extend feedforward estimation from frame pairs to LiDAR sequences [28, 29]. In parallel, test-time optimization also demonstrates strong performance [30–34]. However, the time-consuming inference limits practical deployment. In this work, we build on established feedforward backbones [26, 28] and focus on improving supervision through RVLoss.

Loss design for self-supervised scene flow learning. Early scene flow methods are predominantly supervised, requiring precise annotations from real-world datasets [12, 13, 22, 35] or synthetic data [36]. To reduce reliance on manual annotation, self-supervised methods commonly adopt cycle-consistency-based objectives [21, 30, 37, 38]. However, these objectives typically estimate per-point motion independently and lack an explicit object-level rigidity prior, leading to intra-object motion inconsistency under sparse LiDAR sampling. Recent works address this issue with additional regularizers [3, 8, 32], while TeFlow [10] strengthens supervision through multi-frame temporal consistency. In contrast, we directly incorporate motion rigidity into loss design through a *runoff vote* mechanism, eliminating the need for additional regularization terms.

Rigid motion priors. Rigid motion is a common property in autonomous driving and has been incorporated into scene flow estimation through architectural design and loss regularization. From the model-design perspective, ICP-Flow [34] estimates rigid transformations of clustered points with ICP [39]. RigidFlow [23] integrates ICP into a learning-based framework. PointPWC [21] promotes local consistency through cost volumes and neighborhood feature aggregation. More recently, VoteFlow [9] introduces a differentiable voting module to identify shared translations within clustered points. Unlike these methods that encode rigidity through architectural components, we enforce motion rigidity directly through loss design. From the loss-design perspective, prior works typically encourage motion consistency through local regularization terms [3, 8, 32, 40]. In contrast, RVLoss uses a *runoff vote* to encode the rigidity prior directly in the pseudo-labels used to train a scene-flow backbone, without modifying its architecture or adding a separate regularization term.

Voting mechanisms for point cloud processing. Voting is a classical technique for extracting geometric primitives [41–44]. It has been widely used in point cloud processing, including segmentation [45], detection [46], tracking [47], and pose estimation [48], where sparse local evidence is aggregated into object-level hypotheses. Recent scene flow methods have also incorporated voting mechanisms. ICP-Flow [34] uses voting to identify dominant translations, while VoteFlow [9] extends voting into feature space. Specifically, VoteFlow [9] accumulates votes over displacement bins according to the similarity of learned pillar features, and this voting module remains part of the network at inference. RVLoss instead incorporates voting into the training loss: a first vote over nearest-neighbor displacements proposes a small set of candidate translations, each candidate guides a new geometric matching, and a second vote over the matched displacements yields the supervision. Consequently, RVLoss trains existing backbones without architectural changes.

3 Methodology

This section describes the problem definition and our proposed RVLoss: a self-supervised loss for feedforward scene flow estimation, compatible with state-of-the-art backbone designs.

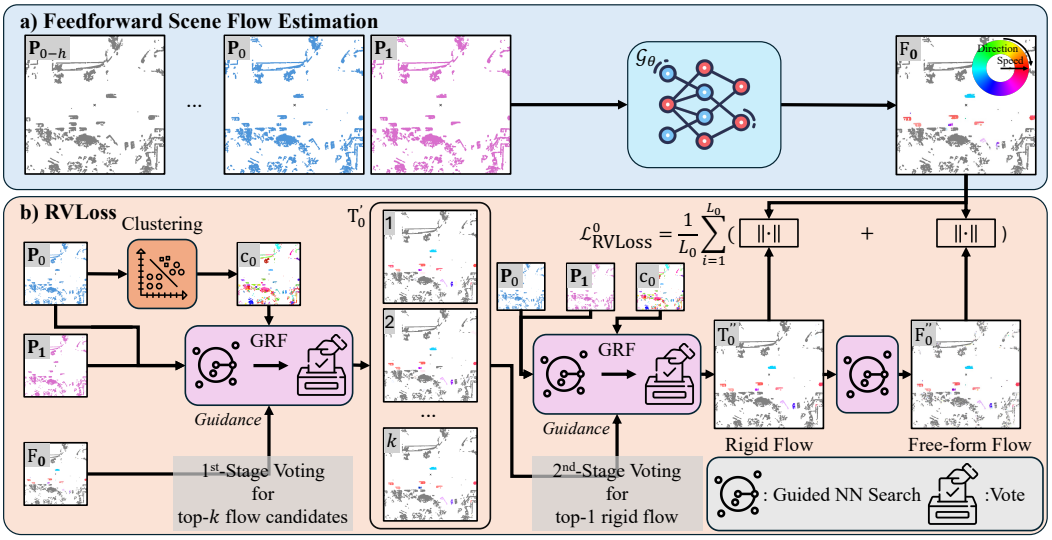


Figure 2: **Overview of RVLoss.** a) A typical feedforward scene flow estimator $\mathcal{G}_\theta$ takes as input a sequence of LiDAR scans (≥ 2 frames), $\mathbf{P}_{0-h}, \ldots, \mathbf{P}_0, \mathbf{P}_1$, and predicts per-point flows $\mathbf{F}_0$. b) RVLoss employs a *runoff vote* mechanism to estimate rigid and free-form flows over clustered points during each forward pass, and uses the resulting flows as pseudo-labels. The *runoff vote* consists of a two-stage voting procedure built upon a shared operation, termed *Guided Rigid Flow Estimation (GRF)*. Specifically, GRF first uses the predicted flow $\mathbf{F}_0$ as guidance to select the top-k rigid flow candidates $\mathbf{T}_0^{'}$ (Stage 1). Subsequently, GRF takes these top-k candidates as guidance and identifies the top-1 rigid flow $\mathbf{T}_0^{''}$ (Stage 2). In addition to rigid flow, RVLoss also computes a free-form flow $\mathbf{F}_0^{''}$ as complementary supervision to relax the rigid-motion assumption and to mitigate the quantization error introduced by voting. The objective is to minimize the residual between the predicted flow and two supervision flows: the top-1 rigid flow and the free-form flow.

3.1 Problem Definition

Given a pair of consecutive LiDAR point clouds, compensated by ego motion, $\mathbf{P}_0 \in \mathbb{R}^{L_0 \times 3} = \{\mathbf{p}_0^i \in \mathbb{R}^3\}_i^{L_0}$ and $\mathbf{P}_1 \in \mathbb{R}^{L_1 \times 3} = \{\mathbf{p}_1^j \in \mathbb{R}^3\}_j^{L_1}$, a neural network $\mathcal{G}_\theta$ produces a per-point flow field, $\mathbf{F}_0 \in \mathbb{R}^{L_0 \times 3} = \{\mathbf{f}_0^i \in \mathbb{R}^3\}_i^{L_0}$, such that $\mathbf{P}_1 \approx \mathbf{P}_0 + \mathbf{F}_0$. L_0 and L_1 denote the number of points in $\mathbf{P_0}$ and $\mathbf{P_1}$. Optionally, multiple historical frames $\mathbf{P}_{0-h}, \cdots, \mathbf{P}_0$ can be incorporated as input to improve performance, where h is the number of historical frames [9, 28]. The training objective of the feedforward scene flow estimation is defined as:

$$\mathbf{F}_0 = \mathcal{G}_\theta\left(\mathbf{P}_{0-h}, \cdots, \mathbf{P}_0, \mathbf{P}_1\right), \tag{1}$$

$$\theta^* = \arg\min_{\theta \in \Theta} \mathbb{E}_{(\mathbf{P}_0, \mathbf{P}_1) \sim \mathcal{D}}\left[\mathcal{L}\left(\mathbf{P}_0, \mathbf{P}_1, \mathbf{F}_0\right)\right], \tag{2}$$

where $\mathcal{D}$ denotes the distribution of consecutive point cloud pairs, and $\mathcal{L}$ represents a self-supervised loss. The bidirectional Chamfer loss $\mathcal{L}_{CD}$ is widely used in self-supervised scene flow estimation [8–10, 30–33, 49], i.e.:

$$\mathcal{L}_{\text{CD}} = \frac{1}{L_0}\sum_{i=1}^{L_0}\mathcal{L}_{\text{NN}}(\mathbf{p}_0^i + \mathbf{f}_0^i, \mathbf{P}_1) + \frac{1}{L_1}\sum_{j=1}^{L_1}\mathcal{L}_{\text{NN}}(\mathbf{p}_1^j, \mathbf{P}_0 + \mathbf{F}_0), \tag{3}$$

where $\mathcal{L}_{\text{NN}}$ calculates the distance between a point and its nearest neighbor in another point cloud, defined by $\mathcal{L}_{\text{NN}}(\mathbf{p}_0, \mathbf{P}_1) = \min_{\mathbf{p}_1^j \in \mathbf{P}_1} \|\mathbf{p}_0 - \mathbf{p}_1^j\|$.

Despite its popularity, $\mathcal{L}_{\text{CD}}$ inherits a critical limitation from nearest-neighbor search: the lack of motion rigidity, which leads to inconsistent flow predictions. To address this issue, recent works [3, 8] introduce multiple regularization terms to encourage coherent motion within each cluster. More importantly, adding extra regularization does not guarantee motion rigidity, demonstrating the need for an inherently motion-rigid loss function and thus motivating our RVLoss.

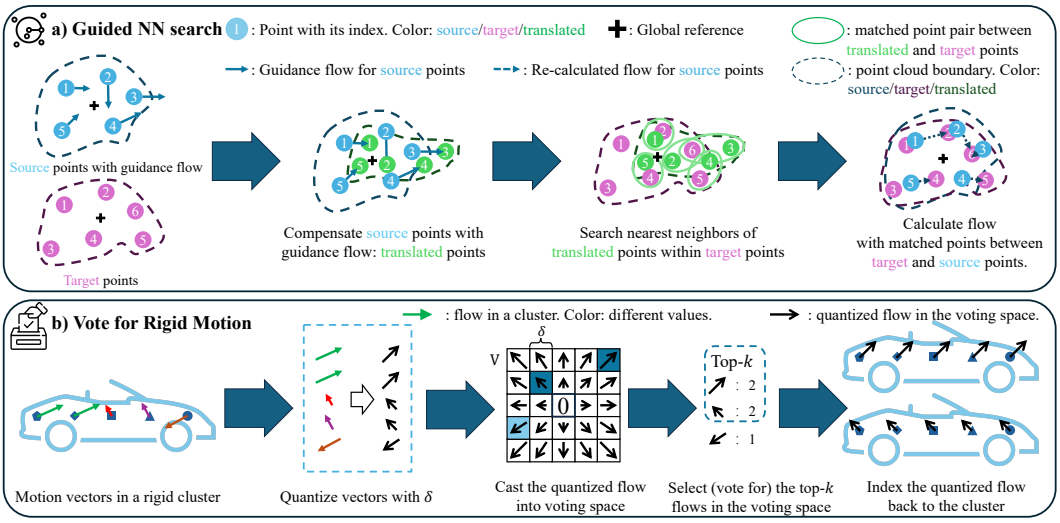


Figure 3: **Overview of GRF. a) Guided NN Search** first translates the source points using the guidance flow and then finds their nearest neighbors in the target point cloud. The point-wise motion between each source point and its matched target point is used for the subsequent processing. **b) Vote for Rigid Motion** quantizes the point-wise flows within each cluster and casts them into a predefined voting space. The top-k flow candidates are then selected and assigned to all points within the cluster.

3.2 RVLoss

Our proposed RVLoss enables self-supervised learning by generating pseudo flow labels during training. It aims to create increasingly more reliable rigid flow estimates during every forward pass, by (1) using the model's increasingly more accurate flow predictions as guidance to calculate the point-wise motion from observed data, and (2) pooling motions across pre-computed point clusters to estimate top-k cluster-wise rigid flows via voting. The clustering of points is done beforehand by HDBSCAN following common practice [8, 50, 51]. We refer to this procedure as *Guided Rigid Flow Estimation* or *GRF* in short, as shown in Fig. 3. A pseudo-implementation of GRF (Algorithm 1) is also available in Sec. A.1.

In practice, we apply GRF twice (see Fig. 2), resulting in a cascaded runoff vote process. In the 1$^{\text{st}}$ stage, GRF uses the model's predicted flow as guidance to generate the top-k most likely rigid flow candidates per cluster (top-k flows). In the 2$^{\text{nd}}$ stage, GRF uses each of the k flow candidates as guidance, letting each point vote k times to find consistent object motion across all points within the cluster. The flow with the highest vote count is deemed the rigid flow (top-1 flow). We also add a free-form loss term to relax the rigid motion assumption and to handle the quantization errors in voting by performing an additional nearest-neighbor lookup using the top-1 rigid flow as guidance. With the rigid and free-form flows established, the forward pass is supervised by a distance-based loss, e.g., ℓ_1 loss, between the predicted flow and pseudo flows. The following subsections first describe GRF, and then elaborate on its use in both stages to create pseudo flows.

3.2.1 Guided Rigid Flow Estimation (GRF)

GRF takes as input the source $\mathbf{P} \in \mathbb{R}^{L_p \times 3}$, the target $\mathbf{Q} \in \mathbb{R}^{L_q \times 3}$, the cluster indices of the source $\mathbf{c} \in \mathbb{Z}^{L_p}$, a guidance flow $\mathbf{F} \in \mathbb{R}^{L_p \times 3}$, and a hyperparameter $k \in \mathbb{Z}^+$ that determines the number of top-k flow candidates per cluster. GRF consists of the following two steps:

Guided nearest-neighbor search. It checks where each point in $\mathbf{P}$ moves to according to the guidance flow $\mathbf{F}$, and identifies the nearest point in $\mathbf{Q}$. Then it computes point-wise flows $\mathbf{M} \in \mathbb{R}^{L_p \times 3} = \text{NN}(\mathbf{P} + \mathbf{F}, \mathbf{Q}) - \mathbf{P}$, where NN indicates the nearest-neighbor search. The whole process is shown in Fig. 3 a).

Vote for rigid motion. Points within each cluster cast a vote, and the top-k flow candidates per cluster are returned, as shown in Fig. 3 b). To be specific, we create a uniformly spaced 2D voting

space over x and y dimensions[2], bounded by the minimal and maximal motion along each dimension, resulting in $\mathbf{V} \in \mathbb{Z}^{c \times m \times n}$, where $m \times n$ are the spatial dimensions of the voting space over x and y, and c is the total number of clusters. In practice, we set the voting step δ, i.e., the distance between two adjacent bins in the voting space, to be 1 cm. For the i^{th} motion vector $\mathbf{M}_i \in \mathbb{R}^3$, we cast a vote into the corresponding bin where it falls after quantization. All votes are equally weighted, i.e., each vote increases the bin count by 1. After voting, we select the top k (quantized) flows $\mathbf{D} \in \mathbb{R}^{c \times k \times 3}$ by the number of votes for each cluster, after non-maximum suppression. We further index $\mathbf{D}$ using cluster indices $\mathbf{c}$ to obtain the point-wise flow candidates $\mathbf{T} \in \mathbb{R}^{L_p \times k \times 3}$. For simplicity, we use $\mathbf{T} = \mathrm{GRF}(\mathbf{P}, \mathbf{Q}, \mathbf{F}, \mathbf{c}, k)$ to illustrate GRF.

3.2.2 1st stage voting for top-k rigid flows

The goal of this stage is to generate a set of top-k flow candidates using GRF. The input to GRF consists of the source $\mathbf{P}_0 \in \mathbb{R}^{L_0 \times 3}$, the target $\mathbf{P}_1 \in \mathbb{R}^{L_1 \times 3}$, cluster indices $\mathbf{c}_0 \in \mathbb{Z}^{L_0}$, initial flow prediction $\mathbf{F}_0 \in \mathbb{R}^{L_0 \times 3}$ (the guidance flow) by a neural network $\mathcal{G}_\theta$, and the number of top-k flow candidates. The output of GRF is top-k rigid flow candidates $\mathbf{T}_0^{'} \in \mathbb{R}^{L_0 \times k \times 3}$, representing k most likely rigid flows per cluster. We represent the 1^{st} stage by $\mathbf{T}_0^{'} = \mathrm{GRF}(\mathbf{P}_0, \mathbf{P}_1, \mathbf{F}_0, \mathbf{c}_0, k)$.

3.2.3 2nd stage voting for top-1 rigid flow

The goal of this stage is to select, from the top-k candidates (i.e., $\mathbf{T}_0^{'} \in \mathbb{R}^{L_0 \times k \times 3}$), the one that best represents the collective motion of clustered points, resulting in $\mathbf{T}_0^{''} \in \mathbb{R}^{L_0 \times 1 \times 3}$. $\mathbf{T}_0^{'}$ becomes the guidance flow in this stage and each source point votes k times. This can be done efficiently by tensor broadcasting operations (see Algorithm 2). For simplicity, we omit the details of broadcasting. Thus, we can re-use GRF for the 2nd stage voting, parameterized by $\mathbf{T}_0^{''} = \mathrm{GRF}(\mathbf{P}_0, \mathbf{P}_1, \mathbf{T}_0^{'}, \mathbf{c}_0, 1)$.

3.2.4 Loss formulation

With the obtained top-1 rigid flow, two loss terms can be computed to complete the forward pass.

Rigid flow. $\mathbf{T}_0^{''}$ is the quantized flow that indicates where clustered points move. We consider $\mathbf{T}_0^{''}$ as the rigid flow supervision.

Free-form flow. To avoid quantization error, we further conduct a guided nearest-neighbor search to retrieve the nearest neighbor of $\mathbf{P}_0$ from $\mathbf{P}_1$ under the guidance of $\mathbf{T}_0^{''}$. As a result, we obtain the free-form point-wise flow $\mathbf{F}_0^{''} \in \mathbb{R}^{L_0 \times 3}$ depicting the motion from $\mathbf{P}_0$ to $\mathbf{P}_1$, i.e., $\mathbf{F}_0^{''} = \mathrm{NN}(\mathbf{P}_0 + \mathbf{T}_0^{''}, \mathbf{P}_1) - \mathbf{P}_0$. We treat $\mathbf{F}_0^{''}$ as the free-form flow supervision.

The total loss, which consists of rigid-flow and free-form-flow supervision, is defined as follows. The corresponding pseudo-implementation is provided in Sec. A.2.

$$\mathcal{L}^0_{\text{RVLoss}} = \frac{1}{L_0} \sum_{i=1}^{L_0} \big(\underbrace{\|\mathbf{T}_0^{''} - \mathbf{F}_0\|}_{\text{Rigid flow}} + \underbrace{\|\mathbf{F}_0^{''} - \mathbf{F}_0\|}_{\text{Free-form flow}} \big). \tag{4}$$

Bidirectional supervision. Similar to the bidirectional Chamfer loss (Eq. (3)), we also calculate the rigid and free-form flow from $\mathbf{P}_1$ to $\mathbf{P}_0$, $\mathbf{F}_1^{''}$ and $\mathbf{T}_1^{''}$ to ensure cycle-consistency. To be specific, we first calculate the nearest-neighbor index of $\mathbf{P}_1$ with respect to $\mathbf{P}_0 + \mathbf{F}_0$, and then retrieve the predicted flow accordingly, denoted as $\mathbf{F}_1$, which characterizes the motion from $\mathbf{P}_1$ to $\mathbf{P}_0$. Afterwards, we repeat the aforementioned 1st stage voting and 2nd stage voting procedures by swapping $\mathbf{P}_1$ and $\mathbf{P}_0$, and obtain $\mathbf{F}_1^{''}$ and $\mathbf{T}_1^{''}$. When bidirectional optimization is adopted, the overall loss becomes: $\mathcal{L}_{\text{RVLoss}} = \frac{1}{L_0} \sum_{i=1}^{L_0} \big(\|\mathbf{T}_0^{''} - \mathbf{F}_0\| + \|\mathbf{F}_\mathbf{0}^{''} - \mathbf{F}_0\| \big) + \frac{1}{L_1} \sum_{j=1}^{L_1} \big(\|\mathbf{T}_1^{''} - \mathbf{F}_1\| + \|\mathbf{F}_\mathbf{1}^{''} - \mathbf{F}_1\| \big)$. Within each training iteration, the resulting rigid and free-form flows are treated as fixed supervision, and no gradients are propagated through the clustering, nearest-neighbor assignment, or voting operations.

[2]We follow the flat ground assumption and thus assume no motion along the z axis.

Table 1: In-domain performance on the AV2 *test* split. Numerical results on scene flow prediction come from the AV2 2026 Scene Flow Challenge leaderboard. '#F_{input}' denotes the number of input frames to the network. '#F_{loss}' denotes the number of input frames during loss calculation. RT stands for runtime per sequence (around 157 frames) in seconds, retrieved from TeFlow [10]. (**Best**, Second Best)

Backbone	Loss	#F_{input}	#F_{loss}	RT (s)	Dynamic Normalized EPE ↓					Three-way EPE (cm) ↓			
					Mean	Car	O.V.	Ped.	VRU	Mean	FD	FS	BS
Supervised methods													
DeltaFlow	GT	5	-	8.0	0.103	0.083	0.132	0.122	0.075	1.76	3.16	1.31	0.81
SSF	GT	2	-	-	0.160	0.115	0.170	0.217	0.137	2.36	4.71	1.69	0.67
Self-supervised methods													
DeFlow	SeFlow	2	2	7.2	0.313	0.236	0.295	0.441	0.273	4.76	10.93	2.23	1.12
VoteFlow	SeFlow	2	2	13.0	0.290	0.226	0.301	0.383	0.252	4.52	10.36	2.09	1.12
DeFlow++	SeFlow++	3	3	10.0	0.283	0.202	0.308	0.387	0.234	4.20	9.49	**2.02**	**1.08**
DeFlow	RVLoss (ours)	2	2	7.2	0.256	0.160	0.254	0.391	0.218	3.63	7.14	2.16	1.60
DeltaFlow	TeFlow	5	5	8.0	0.202	0.167	0.221	0.273	0.146	3.54	7.02	2.18	1.42
DeltaFlow	RVLoss (ours)	5	2	8.0	**0.163**	**0.116**	**0.154**	**0.242**	**0.141**	**2.95**	**4.92**	2.12	1.82

4 Experiments

We conduct benchmarking on the Argoverse2 (AV2) 2026 Scene Flow Challenge to demonstrate the advantage of RVLoss over other loss designs.

4.1 Datasets and Evaluation Metrics

Datasets. The AV2 2026 Scene Flow Challenge has been released to benchmark models across a variety of datasets. The key difference to previous challenges is the multi-dataset evaluation, composed of five mainstream autonomous driving datasets: Argoverse2 (AV2) [6], Waymo [24, 52], nuScenes [53], TruckScenes [54] and AevaScenes [55].

Evaluation Metrics. Following the challenge protocol, we compare the Dynamic Normalized End-Point Error (EPE) [7] within a 35 m radius around the ego vehicle. Unlike standard EPE, which calculates the L2 distance between predicted and ground-truth flow vectors, Dynamic Normalized EPE normalizes EPE by the mean speed within predefined motion buckets, providing a fairer comparison across multiple categories with different velocities. The four categories are: Car, Other Vehicle (O.V.), Pedestrian (Ped.), and Wheeled Vulnerable Road User (VRU).

4.2 Implementation Details

Baselines. We make comparisons with leading feedforward models, including SeFlow [8], VoteFlow [9], SeFlow++ [3], TeFlow [10], SSF [27], and DeltaFlow [28]. We take the numerical results directly from the leaderboard if provided; otherwise, we report results using the publicly released checkpoints by the authors. The optimization-based models are currently absent from the leaderboard, and we are unable to test them due to high computational cost[3].

Implementation of our models. We use RVLoss to supervise the training of two scene flow backbones: multi-frame DeltaFlow [28] and two-frame DeFlow [26]. We follow the official training schemes provided by OpenSceneFlow[4]. We take $k = 10$ by default in the 1^{st} stage voting and discuss the selection of k in Sec. A.4. We adopt the bidirectional optimization by default. Training DeltaFlow+RVLoss takes approximately 3 days on 4 NVIDIA A40 GPUs (48GB VRAM).

4.3 Quantitative Comparison

In-domain evaluation. Tab. 1 demonstrates the in-domain test on the AV2 *test* split, which is part of the AV2 2026 Scene Flow Challenge. All models are trained on the AV2 *train* split. We refer to all models by BACKBONE + LOSS. DeltaFlow+RVLoss achieves the best performance among all self-supervised models when comparing mean dynamic normalized EPE. Particularly,

[3]The leading optimization-based model, EulerFlow [33], takes 24 hours on one NVIDIA V100 16GB GPU for a *single* AV2 sequence.

[4]https://github.com/KTH-RPL/OpenSceneFlow

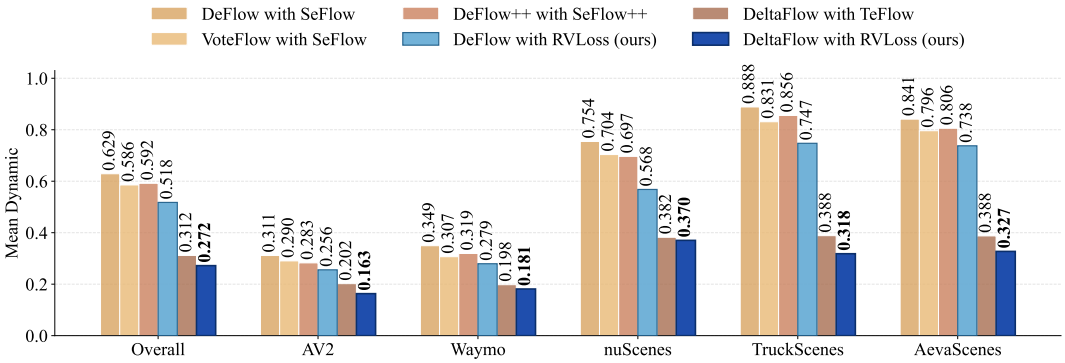


Figure 4: **Cross-domain evaluation performance.** We report the mean dynamic normalized EPE (Mean Dynamic) on multiple datasets used in the AV2 2026 Scene Flow Challenge [11]. DeltaFlow+RVLoss demonstrates consistent performance improvement over all baselines.

given the DeltaFlow backbone, RVLoss outperforms the TeFlow loss on Car and Other Vehicle by approximately 5 p.p. (percentage points) and 7 p.p., demonstrating the advantage of RVLoss over other self-supervised loss functions. Similarly, given the DeFlow [26] backbone, RVLoss is able to deliver better results than the SeFlow [8] loss, verifying the effectiveness of the *runoff vote* design. Despite substantial improvement on Car and Other Vehicle, the gain on Wheeled VRU is less prominent, as the rigid motion assumption no longer stands and point density decreases. When adopting RVLoss during training, the DeltaFlow backbone reduces the mean dynamic normalized EPE by a large margin (up to 9 p.p.) over the DeFlow backbone, demonstrating the benefit of using temporal information. Although the performance gain induced by RVLoss is encouraging, the gap to fully supervised baselines, e.g., DeltaFlow+GT, remains, particularly on Pedestrian and Wheeled VRU. We also perform in-domain evaluation on Waymo [24] and nuScenes [53] in Sec. A.3.

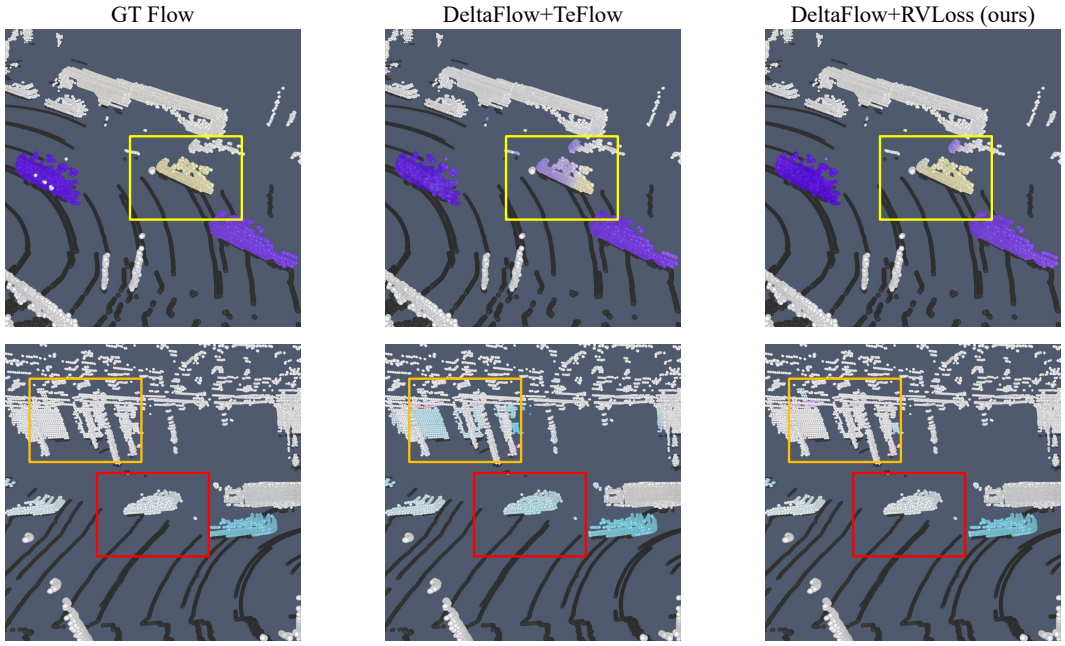


Figure 5: **Qualitative comparison**. Visualization of scene flow predictions from the DeltaFlow model supervised by the TeFlow loss and the proposed RVLoss. Point colors denote flow direction, and color saturation indicates flow magnitude. Objects are expected to exhibit coherent color patterns. Compared with the TeFlow [10] loss, RVLoss produces more uniform coloring within each object (highlighted in boxes), demonstrating more coherent scene flow estimation.

Table 2: Ablation of each loss component on the AV2 *val* split [6]. (**Best**, Second Best)

Free-form Loss	Rigid Loss	Bidirect Optim	Dynamic Normalized EPE ↓					Three-way EPE (cm) ↓			
			Mean	Car	O.V.	Ped.	VRU	Mean	FD	FS	BS
✓	✓	✗	0.284	0.129	0.267	0.442	0.299	3.59	7.31	1.99	1.48
✗	✓	✓	0.338	0.138	0.268	0.604	0.346	3.70	8.27	**1.69**	**1.14**
✓	✗	✓	0.275	0.139	0.266	**0.405**	0.288	4.68	7.48	3.62	2.94
✓	✓	✓	**0.262**	**0.121**	**0.246**	**0.405**	**0.277**	**3.47**	**6.84**	2.05	1.53

Cross-domain evaluation. We make further comparisons on multiple datasets adopted in the AV2 2026 Scene Flow Challenge, as shown in Fig. 4. The models in this comparison are identical to those in Tab. 1, i.e., all are trained on the AV2 *train* split. DeltaFlow+RVLoss demonstrates consistent performance improvement over the others on all five datasets. Notably, the error on nuScenes is nearly twice as high as on AV2 and Waymo. We speculate that the reduced point density is the main cause, as nuScenes uses a *single* 32-beam LiDAR, resulting in $2\times$ fewer beams than the other two. The TruckScenes dataset differs from AV2, Waymo, and nuScenes, as it uses a truck as the ego vehicle and collects data mainly on highways. Despite the use of multiple 64-beam LiDAR sensors, the error remains high, suggesting that models trained on urban-centric and car-based datasets need further fine-tuning to better adapt to truck views and highway scenarios. The AevaScenes dataset uses high-resolution FMCW LiDARs that differ substantially from rotating LiDAR sensors in the other four datasets. All models exhibit relatively high errors on AevaScenes, indicating a notable domain gap. Overall, DeltaFlow+RVLoss delivers not only the best result in the in-domain test but also in the cross-domain test, validating its generalization ability over other variants.

4.4 Ablation Study on Loss Composition

We validate our design choices in Tab. 2, using the DeFlow backbone [26] as it takes less training time than DeltaFlow. The comparison between the first and last rows shows that bidirectional optimization improves performance by enforcing cycle consistency. While the free-form flow alone achieves the second-best performance on mean dynamic normalized EPE, it performs worst on static background regions, as the model tends to overestimate motion and predict most points as moving. Adding the rigid flow loss effectively reduces errors on static points. Therefore, we use both rigid and free-form flows, together with bidirectional optimization, as our default configuration.

4.5 Clustering Sensitivity

RVLoss constructs supervision from precomputed point clusters. To examine how clustering affects performance, we compare five clustering settings using DeFlow [26] on the AV2 *val* split. Following prior work [3, 8], *Default* uses HDBSCAN with minimum cluster size and minimum samples both set to 20, a cluster-selection threshold of $0.7\,\text{m}$, and $\alpha = 1.1$. *Over* uses DBSCAN with radius $0.7\,\text{m}$ and minimum samples 5. *Under* changes only *Default*'s selection threshold to $1.2\,\text{m}$. *Over* and *Under* result in over-segmentation and under-segmentation overall, respectively. We additionally consider

Table 3: **Clustering sensitivity.** Left: mean positive-cluster counts (and proportions) per category per frame on the AV2 *train* set. Right: Dynamic Normalized EPE and Three-way EPE (cm). (**Best**, Second Best); *Oracle* and *Oracle FG* are excluded from ranking. BG: background.

Clustering	Positive clusters per frame						Dynamic Normalized EPE ↓					Three-way EPE (cm) ↓			
	BG	Car	O.V.	Ped.	VRU	Total	Mean	Car	O.V.	Ped.	VRU	Mean	FD	FS	BS
Under	68.22 (80.1%)	13.04 (15.3%)	1.51 (1.8%)	2.10 (2.5%)	0.34 (0.4%)	85.22	0.288	0.128	0.257	0.477	0.290	3.57	7.32	**1.93**	**1.46**
Default	108.24 (84.0%)	14.97 (11.6%)	1.97 (1.5%)	3.21 (2.5%)	0.46 (0.4%)	128.85	0.262	**0.121**	0.246	0.405	0.277	**3.47**	**6.84**	2.05	1.53
Over	188.47 (86.9%)	20.18 (9.3%)	2.95 (1.4%)	4.46 (2.1%)	0.78 (0.4%)	216.85	**0.256**	0.122	**0.236**	**0.395**	**0.272**	3.59	6.85	2.22	1.71
Oracle	0.12 (0.4%)	18.42 (60.2%)	1.64 (5.4%)	8.49 (27.7%)	1.94 (6.3%)	30.61	0.299	0.132	0.280	0.471	0.314	3.40	7.43	1.79	0.99
Oracle FG	115.89 (74.0%)	25.88 (16.5%)	2.70 (1.7%)	10.03 (6.4%)	2.07 (1.3%)	156.57	0.257	0.119	0.253	0.377	0.280	3.50	6.69	2.16	1.65

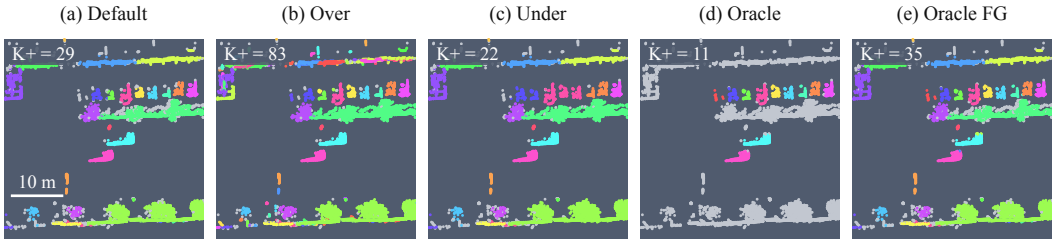


Figure 6: **Qualitative comparison of five clustering settings.** All panels show the same 40×40 m point-cloud crop. Colors distinguish positive-ID clusters within each panel; gray indicates points with label 0. K_+ counts the positive-ID clusters visible in the crop.

two oracle clustering settings based on the GT cuboid annotations. *Oracle* groups foreground points by GT cuboid instance and assigns all remaining non-ground points to a single background cluster with label 0. For a closer comparison with *Default*, *Oracle FG* preserves the same GT foreground clusters but uses *Default*'s local clusters outside the cuboids. Each foreground or background cluster receives a unique positive ID within the frame, while unclustered points retain label 0. Fig. 6 visualizes all five settings, and Tab. 3 reports mean per-category cluster counts per frame.

Tab. 3 shows that *Over* increases positive-cluster counts across all reported categories while maintaining a mean dynamic normalized EPE comparable to *Default* ($0.262 \rightarrow 0.256$). *Under* produces fewer clusters but increases mean dynamic normalized EPE to 0.288, with pedestrians exhibiting the largest degradation. Finer clusters may better satisfy RVLoss's translation approximation for dynamic objects: rigid rotation produces spatially varying displacements that can be approximated by piecewise translations. This interpretation is consistent with *Over*'s improvements on Other Vehicle.

For static regions, larger clusters provide more votes and may reduce sensitivity to LiDAR sampling noise, partial visibility, and occlusion, whereas small fragments may yield less stable estimates. This is consistent with *Under* achieving lower EPE-BS and EPE-FS than *Default* and *Over*. Similarly, *Oracle* achieves the lowest EPE-BS and EPE-FS, but its performance on the foreground is the worst.

Notably, *Default* achieves a comparable mean dynamic normalized EPE to *Oracle FG* (0.262 vs. 0.257), suggesting that RVLoss does not require exact object-level foreground clustering. This result demonstrates that, despite variations in performance, RVLoss does not depend on a specific clustering configuration and is generally robust to over-segmentation.

5 Conclusions

We present **RVLoss**, a self-supervised loss function for LiDAR scene flow estimation using a *runoff vote* mechanism. Unlike the widely used Chamfer loss, RVLoss explicitly incorporates the rigid-motion prior into the design without demanding a separate regularization term. Specifically, RVLoss estimates cluster-wise rigid flows and point-wise free-form flows via a *runoff vote* and uses them as pseudo-labels for self-supervised learning. Extensive experiments on the AV2 2026 Scene Flow Challenge demonstrate that RVLoss consistently improves flow estimation in both in-domain and cross-domain evaluations. For in-domain evaluation, DeltaFlow+RVLoss achieves state-of-the-art performance on the AV2 *test* set, reaching a mean dynamic normalized EPE of 0.163 and substantially outperforming DeltaFlow+TeFlow (0.202). For cross-domain evaluation, DeltaFlow+RVLoss achieves an average mean dynamic normalized EPE of 0.272 across five datasets, outperforming DeltaFlow+TeFlow by 4 percentage points. The advantage on Car and Other Vehicle validates the importance of a rigid-motion prior in loss design and demonstrates the effectiveness of the *runoff vote* mechanism.

Limitations and Future Work. Both voting stages in RVLoss require precomputed cluster labels. Future work could infer voting groups directly from predicted flow. Another limitation is that the rigid-motion assumption generally holds for large traffic participants, such as cars, buses, and trucks, but may not hold for non-rigid or articulated agents, such as pedestrians and other vulnerable road users. Extending the motion prior beyond strict rigidity, e.g., toward part-level or deformable motion consistency, is an important direction for future work.

Acknowledgments

This work was supported by the 3D Urban Understanding (3DUU) Lab funded by the TU Delft AI Initiative.

References

[1] Mahyar Najibi, Jingwei Ji, Yin Zhou, Charles R Qi, Xinchen Yan, Scott Ettinger, and Dragomir Anguelov. Motion inspired unsupervised perception and prediction in autonomous driving. In *ECCV*, 2022.

[2] Ted Lentsch, Holger Caesar, and Dariu M Gavrila. UNION: Unsupervised 3d object detection using object appearance-based pseudo-classes. *NeurIPS*, 2024.

[3] Qingwen Zhang, Ajinkya Khoche, Yi Yang, Li Ling, Sina Sharif Mansouri, Olov Andersson, and Patric Jensfelt. Himo: High-speed objects motion compensation in point clouds. *T-RO*, 2025.

[4] Junhwa Hur, Charles Herrmann, Songyou Peng, Philipp Henzler, Zeyu Ma, Todd Zickler, and Deqing Sun. UFO-4D: Unposed feedforward 4D reconstruction from two images. In *ICLR*, 2026.

[5] Shiming Wang, Holger Caesar, Liangliang Nan, and Julian FP Kooij. Asyncbev: Cross-modal flow alignment in asynchronous 3d object detection. In *ICLR*, 2026.

[6] Benjamin Wilson, William Qi, Tanmay Agarwal, John Lambert, Jagjeet Singh, Siddhesh Khandelwal, Bowen Pan, Ratnesh Kumar, Andrew Hartnett, Jhony Kaesemodel Pontes, Deva Ramanan, Peter Carr, and James Hays. Argoverse 2: Next generation datasets for self-driving perception and forecasting. In *NeurIPS*, 2021.

[7] Ishan Khatri, Kyle Vedder, Neehar Peri, Deva Ramanan, and James Hays. I can't believe it's not scene flow! In *ECCV*, 2024.

[8] Qingwen Zhang, Yi Yang, Peizheng Li, Olov Andersson, and Patric Jensfelt. Seflow: A self-supervised scene flow method in autonomous driving. In *ECCV*, 2024.

[9] Yancong Lin, Shiming Wang, Liangliang Nan, Julian Kooij, and Holger Caesar. Voteflow: Enforcing local rigidity in self-supervised scene flow. In *CVPR*, 2025.

[10] Qingwen Zhang, Chenhan Jiang, Xiaomeng Zhu, Yunqi Miao, Yushan Zhang, Olov Andersson, and Patric Jensfelt. TeFlow: Enabling multi-frame supervision for self-supervised feed-forward scene flow estimation. In *CVPR*, 2026.

[11] Siyi Li, Qingwen Zhang, Ishan Khatri, Kyle Vedder, Deva Ramanan, and Neehar Peri. Uniflow: Towards zero-shot lidar scene flow for autonomous vehicles via cross-domain generalization. *arXiv preprint arXiv:2511.18254*, 2025.

[12] Aseem Behl, Despoina Paschalidou, Simon Donné, and Andreas Geiger. Pointflownet: Learning representations for rigid motion estimation from point clouds. In *CVPR*, 2019.

[13] Xingyu Liu, Charles R Qi, and Leonidas J Guibas. Flownet3d: Learning scene flow in 3d point clouds. In *CVPR*, 2019.

[14] Gilles Puy, Alexandre Boulch, and Renaud Marlet. FLOT: Scene Flow on Point Clouds Guided by Optimal Transport. In *ECCV*, 2020.

[15] Xiuye Gu, Yijie Wang, Chongruo Wu, Yong Jae Lee, and Panqu Wang. Hplflownet: Hierarchical permutohedral lattice flownet for scene flow estimation on large-scale point clouds. In *CVPR*, 2019.

[16] Xingyu Liu, Mengyuan Yan, and Jeannette Bohg. Meteornet: Deep learning on dynamic 3d point cloud sequences. In *ICCV*, 2019.

[17] Yair Kittenplon, Yonina C Eldar, and Dan Raviv. Flowstep3d: Model unrolling for self-supervised scene flow estimation. In *CVPR*, 2021.

[18] Haiyan Wang, Jiahao Pang, Muhammad A Lodhi, Yingli Tian, and Dong Tian. Festa: Flow estimation via spatial-temporal attention for scene point clouds. In *CVPR*, 2021.

[19] Wencan Cheng and Jong Hwan Ko. Bi-pointflownet: Bidirectional learning for point cloud based scene flow estimation. In *ECCV*, 2022.

[20] Zirui Wang, Shuda Li, Henry Howard-Jenkins, Victor Prisacariu, and Min Chen. Flownet3d++: Geometric losses for deep scene flow estimation. In *WACV*, 2020.

[21] 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 *ECCV*, 2020.

[22] Zhao Jin, Yinjie Lei, Naveed Akhtar, Haifeng Li, and Munawar Hayat. Deformation and correspondence aware unsupervised synthetic-to-real scene flow estimation for point clouds. In *CVPR*, 2022.

[23] 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 *CVPR*, 2022.

[24] Philipp Jund, Chris Sweeney, Nichola Abdo, Zhifeng Chen, and Jonathon Shlens. Scalable scene flow from point clouds in the real world. *RA-L*, 2021.

[25] Kyle Vedder, Neehar Peri, Nathaniel Chodosh, Ishan Khatri, Eric Eaton, Dinesh Jayaraman, Yang Liu, Deva Ramanan, and James Hays. Zeroflow: Fast zero label scene flow via distillation. In *ICLR*, 2024.

[26] Qingwen Zhang, Yi Yang, Heng Fang, Ruoyu Geng, and Patric Jensfelt. Deflow: Decoder of scene flow network in autonomous driving. In *ICRA*, 2024.

[27] Ajinkya Khoche, Qingwen Zhang, Laura Pereira Sánchez, Aron Asefaw, Sina Sharif Mansouri, and Patric Jensfelt. SSF: Sparse long-range scene flow for autonomous driving. In *ICRA*, 2025.

[28] Qingwen Zhang, Xiaomeng Zhu, Yushan Zhang, Yixi Cai, Olov Andersson, and Patric Jensfelt. Deltaflow: An efficient multi-frame scene flow estimation method. In *NeurIPS*, 2025.

[29] Jaeyeul Kim, Jungwan Woo, Ukcheol Shin, Jean Oh, and Sunghoon Im. Flow4d: Leveraging 4d voxel network for lidar scene flow estimation. *RA-L*, 2024.

[30] Xueqian Li, Jhony Kaesemodel Pontes, and Simon Lucey. Neural scene flow prior. *NeurIPS*, 2021.

[31] Xueqian Li, Jianqiao Zheng, Francesco Ferroni, Jhony Kaesemodel Pontes, and Simon Lucey. Fast neural scene flow. In *ICCV*, 2023.

[32] David T Hoffmann, Syed Haseeb Raza, Hanqiu Jiang, Denis Tananaev, Steffen Klingenhoefer, and Martin Meinke. Floxels: Fast unsupervised voxel based scene flow estimation. In *CVPR*, 2025.

[33] Kyle Vedder, Neehar Peri, Ishan Khatri, Siyi Li, Eric Eaton, Mehmet Kocamaz, Yue Wang, Zhiding Yu, Deva Ramanan, and Joachim Pehserl. Neural eulerian scene flow fields. In *ICLR*, 2025.

[34] Yancong Lin and Holger Caesar. ICP-Flow: Lidar scene flow estimation with icp. In *CVPR*, 2024.

[35] Shengyu Huang, Zan Gojcic, Jiahui Huang, Andreas Wieser, and Konrad Schindler. Dynamic 3d scene analysis by point cloud accumulation. In *ECCV*, 2022.

[36] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *ICCV*, 2015.

[37] 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 *ICCV*, 2021.

[38] Himangi Mittal, Brian Okorn, and David Held. Just go with the flow: Self-supervised scene flow estimation. In *CVPR*, 2020.

[39] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, 1992.

[40] Kavisha Vidanapathirana, Shin-Fang Chng, Xueqian Li, and Simon Lucey. Multi-body neural scene flow. In *3DV*, 2024.

[41] Richard O Duda and Peter E Hart. Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 1972.

[42] Dana H Ballard. Generalizing the hough transform to detect arbitrary shapes. *PR*, 1981.

[43] Yancong Lin, Ruben Wiersma, Silvia L Pintea, Klaus Hildebrandt, Elmar Eisemann, and Jan C van Gemert. Deep vanishing point detection: Geometric priors make dataset variations vanish. In *CVPR*, 2022.

[44] Yancong Lin, Silvia L Pintea, and Jan C Van Gemert. Deep hough-transform line priors. In *ECCV*, 2020.

[45] Fausto Milletari, Seyed-Ahmad Ahmadi, Christine Kroll, Christoph Hennersperger, Federico Tombari, Amit Shah, Annika Plate, Kai Boetzel, and Nassir Navab. Robust segmentation of various anatomies in 3d ultrasound using hough forests and learned data representations. In *MICCAI*, 2015.

[46] Alain Lehmann, Bastian Leibe, and Luc Van Gool. Fast prism: Branch and bound hough transform for object class detection. *IJCV*, 2011.

[47] Fausto Milletari, Wadim Kehl, Federico Tombari, Slobodan Ilic, Seyed-Ahmad Ahmadi, Nassir Navab, et al. Universal hough dictionaries for object tracking. In *BMVC*, 2015.

[48] Min Sun, Gary Bradski, Bing-Xin Xu, and Silvio Savarese. Depth-encoded hough voting for joint object detection and shape recovery. In *ECCV*, 2010.

[49] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *CVPR*, 2017.

[50] Leland McInnes, John Healy, and Steve Astels. hdbscan: Hierarchical density based clustering. *The Journal of Open Source Software*, 2017.

[51] Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia conference on knowledge discovery and data mining*, 2013.

[52] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *CVPR*, 2020.

[53] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. In *CVPR*, 2020.

[54] Felix Fent, Fabian Kuttenreich, Florian Ruch, Farija Rizwin, Stefan Juergens, Lorenz Lechermann, Christian Nissler, Andrea Perl, Ulrich Voll, Min Yan, et al. Man truckscenes: A multimodal dataset for autonomous trucking in diverse conditions. *NeurIPS*, 2024.

[55] Gautham Narayan Narasimhan, Heethesh Vhavle, Kumar Bhargav Vishvanatha, and James Reuther. Aevascenes: A dataset and benchmark for fmcw lidar perception, 2025. URL `https://scenes.aeva.com/`.

[56] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *NeurIPS*, 2019.

A Technical appendices and supplementary material

A.1 Pseudo-implementation of Guided Rigid Flow Estimation

Algorithm 1: Guided Rigid Flow Estimation (PyTorch-style [56])

Input: $\mathbf{P} \in \mathbb{R}^{L_P \times 3}$, $\mathbf{Q} \in \mathbb{R}^{L_Q \times 3}$, $\mathbf{F} \in \mathbb{R}^{L_P \times 3}$, clusters $\mathbf{c} \in \mathbb{Z}^{L_P}$, $\delta \in \mathbb{R}^+$, $k \in \mathbb{Z}^+$
Output: Top-k rigid flow per point $\mathbf{T} \in \mathbb{R}^{L_P \times k \times 3}$

Guided nearest-neighbor search

$\mathbf{M} \in \mathbb{R}^{L_P \times 3} \leftarrow \texttt{NN}(\mathbf{P} + \mathbf{F}, \mathbf{Q}) - \mathbf{P}$; `// Point-wise motion`

Vote for top-k rigid motion

$c \leftarrow \mathbf{c}.\texttt{max}() + 1$; `// Number of clusters`
$\mathbf{b}_x \in \mathbb{R}^m \leftarrow \texttt{torch.arange}(\mathbf{M}[:,0].\texttt{min}(), \mathbf{M}[:,0].\texttt{max}(), \delta)$; `// Bins over` x
$\mathbf{b}_y \in \mathbb{R}^n \leftarrow \texttt{torch.arange}(\mathbf{M}[:,1].\texttt{min}(), \mathbf{M}[:,1].\texttt{max}(), \delta)$; `// Bins over` y
$\mathbf{V} \in \mathbb{R}^{c \times m \times n} \leftarrow \texttt{torch.zeros}(c, m, n)$ `// Voting space over` x `and` y
$\mathbf{v}_x \in \mathbb{Z}^{L_P} \leftarrow \texttt{torch.bucketize}(\mathbf{M}[:,0], \mathbf{b}_x)$; `// Flow indices after binning`
$\mathbf{v}_y \in \mathbb{Z}^{L_P} \leftarrow \texttt{torch.bucketize}(\mathbf{M}[:,1], \mathbf{b}_y)$; `// Flow indices after binning`
$\mathbf{V}[\mathbf{c}, \mathbf{v}_x, \mathbf{v}_y] += 1$; `// Majority voting`
$\mathbf{V} \leftarrow \texttt{F.max_pool2d}(\mathbf{V}.\texttt{unsqueeze}(0), 3, 1, 1).\texttt{squeeze}(0)$; `// Non-max suppression`
$\mathbf{I} \in \mathbb{Z}^{c \times k} \leftarrow \texttt{torch.topk}(\mathbf{V}.\texttt{view}(c, -1), k, -1)$; `// Top-`k `indices per cluster`
$\mathbf{D}_x \in \mathbb{R}^{c \times k} \leftarrow \mathbf{b}_x[\mathbf{I} \,\%\, m]$; `// Rigid flow over` x
$\mathbf{D}_y \in \mathbb{R}^{c \times k} \leftarrow \mathbf{b}_y[\mathbf{I} \,//\, m \,\%\, n]$; `// Rigid flow over` y
$\mathbf{D}_z \in \mathbb{R}^{c \times k} \leftarrow \texttt{torch.zeros}(c, k)$; `// Zero flow over` z
$\mathbf{D} \in \mathbb{R}^{c \times k \times 3} \leftarrow \texttt{torch.stack}([\mathbf{D}_x, \mathbf{D}_y, \mathbf{D}_z], -1)$; `// Top-`k `rigid flow per cluster`
$\mathbf{T} \in \mathbb{R}^{L_P \times k \times 3} \leftarrow \mathbf{D}[\mathbf{c}]$; `// Top-`k `rigid flow per point`
return $\mathbf{T}$;

A.2 Pseudo-implementation of RVLoss

Algorithm 2: RVLoss

Input: $\mathbf{P}_0 \in \mathbb{R}^{L_0 \times 3}$, $\mathbf{P}_1 \in \mathbb{R}^{L_1 \times 3}$, $\mathbf{F}_0 \in \mathbb{R}^{L_0 \times 3}$, $k \in \mathbb{Z}$, $\delta \in \mathbb{R}$
Output: $\mathbf{T}_0^{''} \in \mathbb{R}^{L_0 \times 3}$, $\mathbf{F}_0^{''} \in \mathbb{R}^{L_0 \times 3}$

Clustering

$\mathbf{c}_0 \in \mathbb{Z}^{L_0} \leftarrow \texttt{HDBSCAN}(\mathbf{P}_0)$; `// Point clustering`

Top-k rigid flows (1st stage)

$\mathbf{T}_0^{'} \in \mathbb{R}^{L_0 \times k \times 3} \leftarrow \mathrm{GRF}(\mathbf{P}_0, \mathbf{P}_1, \mathbf{F}_0, \mathbf{c}_0, \delta, k)$; `// Top-`k `rigid flows`

Top-1 rigid flow (2nd stage)

$\mathbf{P}_0^{'} \in \mathbb{R}^{L_0 k \times 3} \leftarrow \mathbf{P}_0[:, \texttt{None}, :].\,\mathrm{expand}(L_0, k, 3).\,\mathrm{view}(L_0 \times k, 3)$; `// Broadcasting`
$\mathbf{c}_0^{'} \in \mathbb{R}^{L_0 k} \leftarrow \mathbf{c}_0.\,\mathrm{repeat_interleave}(k)$; `// Broadcasting`
$\mathbf{T}_0^{'} \in \mathbb{R}^{L_0 k \times 3} \leftarrow \mathbf{T}_0^{'}.\,\mathrm{view}(L_0 \times k, 3)$; `// Reshaping`
$\mathbf{T}_0^{''} \in \mathbb{R}^{L_0 \times 1 \times 3} \leftarrow \mathrm{GRF}(\mathbf{P}_0^{'}, \mathbf{P}_1, \mathbf{T}_0^{'}, \mathbf{c}_0^{'}, \delta, 1)$; `// Top-1 rigid flow`
$\mathbf{F}_0^{''} \in \mathbb{R}^{L_0 \times 3} \leftarrow \texttt{NN}(\mathbf{P}_0 + \mathbf{T}_0^{''}.\,\mathrm{squeeze}(1), \mathbf{P}_1) - \mathbf{P}_0$; `// Free-form flow`
return $\mathbf{T}_0^{''}, \mathbf{F}_0^{''}$;

A.3 In-domain evaluation on nuScenes and Waymo

Besides Argoverse2, we conduct further in-domain tests on nuScenes and Waymo datasets. The dynamic normalized EPE on these two datasets is shown in Tab. 4. Note that the results of TeFlow [10] are reproduced using the released pretrained models, whereas the results of other methods are taken

Table 4: Dynamic Normalized EPE (↓) on the nuScenes [53] *val* set and the Waymo [24] *valid* set. (**Best**, Second Best)

Method	Loss	nuScenes [53]					Waymo [24]			
		Mean	Car	O.V.	Ped.	VRU	Mean	Car	Ped.	VRU
DeFlow	SeFlow	0.544	0.396	0.653	0.726	0.419	0.351	0.212	0.551	0.289
VoteFlow	SeFlow	0.538	0.355	0.605	0.780	0.410	0.347	0.197	0.548	0.298
DeFlow++	SeFlow++	0.509	0.327	0.583	0.716	0.409	0.323	0.201	0.521	0.247
DeltaFlow	TeFlow	0.433	0.347	0.492	0.628	**0.266**	0.285	0.164	**0.486**	0.206
DeltaFlow	RVLoss (ours)	**0.388**	**0.291**	**0.319**	**0.578**	0.365	**0.275**	**0.124**	0.507	**0.194**

from the TeFlow paper, as their checkpoints are not publicly available. Overall, DeltaFlow+RVLoss achieves state-of-the-art performance on both datasets in terms of mean dynamic normalized EPE. At the category level, DeltaFlow+RVLoss consistently outperforms DeltaFlow+TeFlow on Car and Other Vehicle. However, RVLoss occasionally underperforms on Pedestrian and Wheeled VRU. A possible explanation is that these objects often exhibit non-rigid or articulated motion, which violates the rigid-motion assumption. This experiment further supports our claim that incorporating motion rigidity into loss design enhances the quality of flow estimation.

A.4 Ablation Study on Voting Components

Table 5: Ablation of k in voting modules on the AV2 *val* split [6]. (**Best**, Second Best)

Stage 1	Stage 2	Dynamic Normalized EPE ↓					Three-way EPE (cm) ↓			
		Mean	Car	O.V.	Ped.	VRU	Mean	FD	FS	BS
-	1	0.499	0.373	0.667	0.540	0.418	7.19	18.5	**1.81**	**1.29**
3	1	0.317	0.176	0.303	0.474	0.316	4.44	9.84	2.03	1.47
5	1	0.277	0.137	0.260	0.418	0.291	3.73	7.75	1.98	1.46
10	1	**0.262**	**0.121**	**0.246**	**0.405**	**0.277**	**3.47**	**6.84**	2.05	1.53
20	1	0.297	0.127	0.276	0.500	0.285	3.67	7.41	2.04	1.57

We validate our design choices using the DeFlow backbone [26]. Tab. 5 presents the ablation study of the *runoff vote* process, including the necessity of two-stage voting and the choice of top-k. The variant that only uses a single stage produces the highest mean error, which is more than 10 p.p. higher than the other two-stage variants, showcasing the demand for two-stage voting. Empirically, we find that $k = 10$ achieves the best performance.

A.5 Geometric Runoff Voting on VoteFlow Features

We perform a two-stage voting analysis on VoteFlow [9]. VoteFlow and RVLoss vote over different evidence. RVLoss votes over geometric translations, whereas VoteFlow's in-model voting module aggregates cosine similarities between source and target pillar features. For each source pillar, we inspect VoteFlow's voting space and calculate the hit rate between the Top-1/Top-10 candidate flows and the ground-truth flows using a threshold $\tau = 0.1$ m, which accounts for quantization errors. *VF Top-1* evaluates the highest-scoring bin in the feature space, whereas *VF Top-10* measures whether at least one of the ten highest-scoring bins is a hit.

Following the two-stage voting principle of RVLoss, we use the Top-10 bins as the first-stage translation candidates. Their bin centers are decoded into metric translations and used to warp the source points toward the target. For each candidate, we perform GRF (Guided Rigid Flow Estimation) under the guidance of the candidate translation. The resulting point-level translations then cast unit votes in a geometric voting space, where the highest-scoring bin is selected as *RV Top-1*. As a reference, we also report *VF decoded*, VoteFlow's point-wise flow predictions, evaluated on the same points using the same hit criterion.

As shown in Tab. 6, VF Top-1 achieves a hit rate of 24.2%, while VF Top-10 achieves a hit rate of 59.9%. VoteFlow struggles to identify the optimal candidate, as the performance gap between VF

Top-10 and VF Top-1 remains substantial. When using the VF Top-10 translation candidates for geometric runoff voting, RV Top-1 consistently improves over VF Top-1 across all categories. This demonstrates that the runoff mechanism can effectively select a better candidate from the given set.

VF Top-10 is an oracle coverage metric rather than a prediction. The 6.4 p.p. gain applies to the raw VF Top-1 decision; it does not establish an improvement or superiority over its full learned voting representation. This experiment provides mechanistic evidence for geometric runoff at the intermediate voting stage. Determining whether it improves final scene-flow predictions requires integration into VoteFlow, retraining, and controlled evaluation.

Table 6: **Voting on frozen VoteFlow feature candidates.**

Category	VF Top-1 (%)	VF Top-10 (%)	RV Top-1 (%)	Gain (p.p.)	VF decoded (%)
Car	21.5	55.2	28.0	+6.5	65.7
O.V.	25.2	63.2	28.1	+3.0	48.4
Ped.	40.5	75.2	68.8	+28.4	84.1
VRU	40.1	81.8	62.4	+22.3	57.7
All foreground	24.2	59.9	30.6	+6.4	59.5

A.6 RVLoss for offline flow refinement

Optionally, the runoff vote process in RVLoss can also be used as an offline refinement module that adjusts the predicted flow from a pretrained model, producing both refined rigid flow and refined free-form flow. We compare the predicted and refined flows in Tab. 7. For Car and Other Vehicle, the refined rigid flow consistently improves over the predicted flow, demonstrating the benefit of *runoff vote* in enhancing motion rigidity. However, it may degrade performance for Pedestrian, highlighting its limitations in handling non-rigid motion. The refined free-form flow exhibits varying behavior depending on the quality of the initial predictions. For DeFlow and DeFlow++ models, it generally improves over the predicted flow. In contrast, when applied to DeltaFlow, the refined free-form flow no longer yields gains, suggesting that it is approaching an upper performance bound (mean dynamic normalized EPE around 0.28).

A.7 Stage-wise Analysis of Pseudo-label Construction

We analyze RVLoss's pseudo-label construction using DeFlow trained with RVLoss on AV2 *val* split. We compare the model prediction, first-stage rigid flow candidates, second-stage rigid flow, and final free-form flow against the GT flow. Mean dynamic normalized EPE (Dynamic Mean) is reported.

Stage definitions. *Model prediction* is the direct output of the DeFlow backbone. *Top-k oracle flow* evaluates the candidate set produced by the first voting stage (Sec. 3.2.2), with $k = 10$. Specifically, we select a single candidate flow per cluster that minimizes the mean 3D endpoint distance to the GT flow. The selected candidate is shared by all points within the cluster. *Top-1 rigid flow* is the cluster-shared translation selected by the second voting stage (Sec. 3.2.3). *Free-form flow* is the subsequent point-wise flow obtained through nearest-neighbor retrieval (Sec. 3.2.4).

Overall behavior. Tab. 8 shows Top-k oracle reduces mean dynamic normalized EPE from 0.262 to 0.152, indicating that the first-stage candidate set contains valid flow hypotheses for each cluster.

Table 7: RVLoss as offline Flow Refinement. We use RVLoss to refine the predicted flows and thus obtain rigid flows and free-form flows. We compare these flows on the AV2 *val* split [6] using mean dynamic normalized EPE. (**Best**/Second Best in each row and category)

Method	Loss	Predicted flow↓					(Refined) rigid flow↓					(Refined) free-form flow↓				
		Mean	Car	O.V.	Ped.	VRU	Mean	Car	O.V.	Ped.	VRU	Mean	Car	O.V.	Ped.	VRU
DeFlow	SeFlow	0.342	0.205	0.340	0.510	0.315	**0.237**	**0.140**	**0.273**	**0.348**	**0.189**	0.293	0.194	0.294	0.411	0.272
DeFlow++	SeFlow++	0.300	0.180	0.340	0.395	0.288	**0.236**	**0.131**	**0.260**	**0.363**	**0.189**	0.289	0.187	0.282	0.416	0.270
DeltaFlow	TeFlow	0.223	0.140	0.268	**0.304**	0.179	**0.220**	**0.119**	**0.243**	0.354	**0.162**	0.278	0.178	0.267	0.414	0.255

Table 8: **Stage-wise pseudo-label accuracy on AV2 *val* split.** Δ_{sel} measures the selection gap between second-stage voting and the GT-assisted reference (Top-1 − Top-k oracle). Δ_{ret} measures the error change after final point-wise nearest-neighbor retrieval (Free-form − Top-1). Positive differences indicate degradation.

Category	Model prediction	Top-k oracle	Top-1 rigid	Free-form flow	Δ_{sel}	Δ_{ret}
Car	0.121	0.067	0.109	0.171	+0.042	+0.062
O.V.	0.246	0.123	0.221	0.246	+0.098	+0.025
Ped.	0.407	0.287	0.345	0.409	+0.058	+0.064
VRU	0.275	0.132	0.176	0.260	+0.044	+0.084
Dynamic Mean	0.262	0.152	0.213	0.272	+0.061	+0.059

Table 9: **Runtime breakdown.** Times are milliseconds per bidirectional frame pair. 2nd GRF share is the 2nd GRF time divided by the Sum column. Dynamic Mean is the mean dynamic normalized EPE from the separate training ablation in Tab. 5.

k	1st GRF	2nd GRF	Free-form KNN	Sum	2nd GRF share	Dynamic Mean ↓
3	22.16	60.14	22.00	104.30	57.7%	0.317
5	22.35	96.79	21.98	141.13	68.6%	0.277
10	22.61	188.25	21.92	232.77	80.9%	**0.262**
20	22.34	352.71	21.84	396.90	88.9%	0.297

Second-stage selection yields a mean dynamic normalized EPE of 0.213, and free-form retrieval further increases it to 0.272. The dominant source of diagnostic error differs by category.

Pedestrian. Pedestrian retains the highest Top-k oracle error among the evaluated categories (0.287), even when ground truth is used to select the best cluster-shared candidate. This indicates that its performance gap is already present in the first-stage voting. This behavior is consistent with articulated or heterogeneous intra-cluster pedestrian motion. The subsequent top-1 selection and free-form retrieval introduce additional increases of 0.058 and 0.064, respectively. We therefore attribute the gap on the Pedestrian category primarily to limited candidate/shared-motion representation.

Wheeled VRU. In contrast, degradation on the Wheeled VRU category comes mostly from the free-form retrieval, increasing the error by 0.084 – the largest such increase among the evaluated categories. This identifies point-wise nearest-neighbor retrieval as the principal bottleneck for Wheeled VRU. Sparse and thin geometry, partial visibility, and rider–vehicle motion differences may make cross-frame point correspondences ambiguous, although isolating their individual effects requires further analysis.

Discussion. These results show that the underperforming categories do not share a single failure mode: performance on Pedestrian is primarily influenced by the Top-k rigid flow candidates, whereas performance on Wheeled VRU is primarily influenced by the final free-form retrieval.

A.8 Runtime Analysis of Runoff Voting

We analyze a detailed runtime breakdown of the two-stage runoff-voting procedure for $k \in \{3, 5, 10, 20\}$ (Tab. 9). The 1st GRF represents the 1st stage voting for top-k rigid flows, which uses the predicted flow to find nearest neighbors, then votes for k candidate translations. The 2nd GRF is the 2nd stage voting for top-1 rigid flow, which repeats the nearest-neighbor search for each candidate and votes for one translation per cluster. A final nearest-neighbor search guided by this translation yields the free-form flow. We profile 20 AV2 validation pairs on an NVIDIA A100 80GB PCIe GPU, using two warm-up and seven measured repetitions for each k. For each stage, we sum forward and inverse CUDA-event times per repetition. Timing covers pseudo-label construction, excluding the backbone, loss reduction, backpropagation, and data loading.

At the default $k = 10$, the runtimes sum to 232.77 ms per bidirectional frame pair. The 2nd GRF accounts for 80.9% of this sum and is therefore the primary computational bottleneck. Its runtime

grows approximately linearly with k, increasing from 60.14 to 352.71 ms as k increases from 3 to 20, consistent with performing a nearest-neighbor search for each candidate branch. The free-form retrieval remains nearly constant and takes 21.84–22.00 ms across these settings, which introduces only a marginal additional cost. Importantly, this cost is incurred only during training, when the runoff procedure constructs pseudo-labels for the self-supervised objective. Neither runoff voting nor its additional nearest-neighbor searches are executed during standard inference. Consequently, RVLoss introduces no additional inference-time operations or latency and leaves the inference procedure of the underlying scene-flow backbone unchanged.

A.9 Broader Impact

We introduce a self-supervised loss for scene flow estimation that is compatible with existing feedforward backbones. By removing the need for manual annotations, it has the potential to substantially reduce data collection and labeling costs. At the same time, it improves scalability by enabling training on large-scale unlabeled LiDAR datasets commonly available in autonomous driving. More accurate dynamic scene understanding can enhance the safety and reliability of point cloud-based perception systems, particularly in autonomous driving and robotics applications. However, increased scalability may come at the cost of higher computational demands, potentially raising the carbon footprint of model training. In addition, there is a risk that scene flow models could be misused for unauthorized tracking or surveillance. To mitigate these risks, it is essential to adopt responsible data governance, enforce access controls, and adhere to established ethical principles, ensuring the technology is deployed for societal benefit.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims made in the abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We include the limitation discussion in the Conclusion part of this paper.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: We do not include theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Code and models are publicly available at `https://github.com/tudelft-iv/RVLoss`.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Code and models are publicly available at `https://github.com/tudelft-iv/RVLoss`.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We elaborated the training and test details in the paper and appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We reported performance over multiple commonly used datasets within the community.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Computer resources are reported in the paper.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conducted in the paper conforms, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss the societal impact of autonomous driving in our submission.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: Our submission has no such concerns.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All datasets and assets used in this work (e.g., AV2, Waymo, and nuScenes) are publicly available and used in compliance with their respective licenses and terms of use. We properly cite the original sources in the paper. No proprietary, private, or restricted data or code is used.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The source code in this work is well documented, and code and models are publicly available at https://github.com/tudelft-iv/RVLoss.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This work does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This work does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.

- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLM is used only for writing, editing, or formatting purposes.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.